COLLEGES BRETAGNE LOIRE SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

THÈSE DE DOCTORAT DE

.

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE - IMT ATLANTIQUE
DÉLIVRÉE CONJOINTEMENT AVEC
ESCUELA COLOMBIANA DE INGENIERÍA
"JULIO GARAVITO"

ÉCOLE DOCTORALE N° 602
SPIN: Sciences pour l'Ingénieur et le Numerique
Spécialité: *Computer Science*

Par

Wilmer GARZÓN-ALFONSO

Secure distributed workflows for biomedical data analysis

Thèse présentée et soutenue à Nantes, le 30 août 2023
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N)

Rapporteurs avant soutenance :

M. Eddy CARON — Professeur, École Normale Supérieure de Lyon (France)
M. Jorge DUITAMA — Professeur, Universidad de los Andes (Colombia)

Composition du Jury :

Président :	Mme. Françoise BAUDE	Professeure, Université Côte d'Azur (France)
Examinateurs :	Mme. Françoise BAUDE	Professeure, Université Côte d'Azur (France)
	M. Mauricio SOLAR	Professeur, Universidad Técnica Federico Santa María (Chile)
Dir. de thèse :	M. Mario SÜDHOLT	Professeur, IMT Atlantique (France)
Co-dir. de thèse :	M. Daniel BENAVIDES	Professeur, Escuela Colombiana de Ingeniería "Julio Garavito" (Colombia)

Invité(s) :

M. Alban GAIGNARD — Research Engineer at L'institut du Thorax in Nantes (France)

ABSTRACT

Title: Secure distributed workflows for biomedical data analysis

Keywords: Distributed biomedical analyses, fully distributed collaborations, workflow specification language, multi-site analyses, distributed workflow analyses.

Abstract: In recent years, the amount of biomedical data collected and stored has grown significantly. Analyzing these extensive amounts of data can no longer be done by individuals or single organizations. Thus, the scientific community is creating global collaborative efforts to study these data. However, biomedical data is subject to several legal and socioeconomic restrictions hindering the possibilities for research collaboration.

In this thesis, we first investigate and show that researchers require new tools and techniques to address the restrictions and needs of global scientific collaborations over geo-distributed biomedical data. In particular, we identify three kinds of constraints related to global collaborations, namely, technical, legal, and socioeconomic constraints. We also investigate the state of the art of current tools for distributed global biomedical analyses, including tools using machine learning techniques, and show their limitations.

From these findings, we propose Fully Distributed Collaborations (FDC), which are research endeavors that harness means to exploit and analyze massive biomedical information collaboratively while respecting legal and socioeconomic restrictions. Here, we investigate the concept, properties, and features of FDCs, as well as the architectural requirements and security and privacy needs. As a first example of the design of FDC-based tools, we propose a fully distributed machine learning strategy. The strategy considers a random forest training algorithm where multiple geo-distributed sites, with their own private data, compute a global model collaboratively without sharing private information. The proposed algorithm, called MuSiForest, improves over other existing multi-site forest approaches by ameliorating computation time and reducing the amount of shared data while having a training precision close to that of centralized random forest techniques.

Finally, we investigate how workflow systems have been widely used to specify biomedical data analyses, and we show the current limitations of those tools. We show how they offer limited means to define, deploy, and execute multi-site studies in today's distributed infrastructure while respecting data ownership and privacy restrictions. We then propose FeDeRa, a language to specify, deploy, and execute FDC-compliant multi-site scientific workflows. The language is enriched with abstractions to deploy analysis in geo-distributed cloud infrastructures and with abstractions to define complex workflow patterns across multi-site boundaries. FeDeRa supports dataflow programming and declarative concurrency natively. We also present the implementation of a runtime engine supporting the execution of FeDeRa workflows and experiments deployed on cloud infrastructure.

RÉSUMÉ

Titre: Flux de travail distribués sécurisés pour l'analyse des données biomédicales

Mot clés: Analyses biomédicales distribuées, collaborations entièrement distribuées, langage de spécification de flux de travail, analyses multi-sites, analyses de flux de travail distribuées.

Résumé: Ces dernières années, la quantité de données biomédicales collectées et stockées a considérablement augmenté. L'analyse de ces grandes quantités de données ne peut plus être effectuée par des individus ou des organisations uniques. Ainsi, la communauté scientifique crée des efforts de collaboration mondiaux pour étudier ces données. Cependant, les données biomédicales sont soumises à plusieurs restrictions légales et socio-économiques entravant les possibilités de collaboration en recherche.

Dans cette thèse, nous étudions et montrons d'abord que les chercheurs ont besoin de nouveaux outils et techniques pour répondre aux restrictions et aux besoins des collaborations scientifiques mondiales sur les données biomédicales géodistribuées. En particulier, nous identifions trois types de contraintes liées aux collaborations mondiales, à savoir les contraintes techniques, juridiques et socio-économiques. Nous étudions également l'état de l'art des outils actuels d'analyses biomédicales globales distribuées, y compris les outils utilisant des techniques d'apprentissage automatique, et montrons leurs limites.

À partir de ces résultats, nous proposons des collaborations entièrement distribuées (FDC), qui sont des efforts de recherche qui exploitent des moyens pour exploiter et analyser de manière collaborative des informations biomédicales massives tout en respectant les restrictions légales et socio-économiques. Ici, nous étudions le concept, les propriétés et les fonctionnalités des FDC, ainsi que les exigences architecturales et les besoins en matière de sécurité et de confidentialité. Comme premier exemple de conception d'outils basés sur FDC, nous proposons une stratégie d'apprentissage automatique entièrement distribuée. La stratégie considère un algorithme de formation de forêt aléatoire où plusieurs sites géo-distribués, avec leurs propres données privées, calculent un modèle global en collaboration sans partager d'informations privées. L'algorithme proposé, appelé MuSiForest, s'améliore par rapport aux autres approches forestières multi-sites existantes en améliorant le temps de calcul et en réduisant la quantité de données partagées tout en ayant une précision d'apprentissage proche de celle des techniques forestières aléatoires centralisées.

Enfin, nous étudions comment les systèmes de workflow ont été largement utilisés pour spécifier les analyses de données biomédicales, et nous montrons les limites actuelles de ces outils. Nous montrons comment ils offrent des moyens limités pour définir, déployer et exécuter des études multi-sites dans l'infrastructure distribuée d'aujourd'hui tout en respectant la propriété des données et les restrictions de confidentialité. Nous proposons ensuite FeDeRa, un langage pour spécifier, déployer et exécuter des workflows scientifiques multi-sites conformes à la FDC. Le langage est enrichi d'abstractions pour déployer

l'analyse dans des infrastructures cloud géo-distribuées et d'abstractions pour définir des modèles de flux de travail complexes à travers les frontières multi-sites. FeDeRa prend en charge la programmation par flux de données et la concurrence déclarative de manière native. Nous présentons également l'implémentation d'un moteur d'exécution supportant l'exécution des workflows FeDeRa et des expérimentations déployées sur une infrastructure cloud.

RESUMEN

Título: Flujos de trabajo distribuidos y seguros para análisis de datos biomédicos

Palabras Clave: Análisis biomédicos distribuidos, colaboraciones totalmente distribuidas, lenguaje de especificación de flujo de trabajo, análisis multi sitio, análisis de flujo de trabajo distribuido.

Resumen: En los últimos años, la cantidad de datos biomédicos recopilados y almacenados ha crecido significativamente. El análisis de estas grandes cantidades de datos ya no puede ser realizado por individuos u organizaciones individuales. Por lo tanto, la comunidad científica está creando esfuerzos de colaboración global para analizar estos datos. Sin embargo, los datos biomédicos están sujetos a varias restricciones legales y socioeconómicas que dificultan las posibilidades de colaboración en investigación.

En esta tesis, primero investigamos y mostramos que los investigadores requieren nuevas herramientas y técnicas para abordar las restricciones y necesidades de las colaboraciones científicas globales sobre datos biomédicos geo distribuidos. En particular, identificamos tres tipos de restricciones relacionadas con las colaboraciones globales, a saber, restricciones técnicas, legales y socioeconómicas. También investigamos el estado del arte de las herramientas actuales para análisis biomédicos globales distribuidos, incluidas herramientas que utilizan técnicas de aprendizaje automático, y mostramos sus limitaciones.

A partir de estos hallazgos, proponemos colaboraciones totalmente distribuidas FDC (definido en inglés como Fully Distributed Collaborations), como esfuerzos de investigación que aprovechan los medios para explotar y analizar información biomédica de forma masiva y colaborativa respetando las restricciones legales y socioeconómicas. Nosotros investigamos el concepto, las propiedades y las características de los sistemas FDC, así como los requisitos de arquitectura y las necesidades de seguridad y privacidad. Como primer ejemplo del diseño de herramientas basadas en FDC, proponemos una estrategia de aprendizaje automático completamente distribuida. La estrategia considera un algoritmo de entrenamiento de bosque aleatorio donde varios sitios distribuidos geográficamente, mantienen sus propios datos privados, entrenan un modelo global en colaboración sin compartir información privada. El algoritmo propuesto, llamado MuSiForest, mejora con respecto a otros enfoques existentes de bosques multi sitio al mejorar el tiempo de cómputo y reducir la cantidad de datos compartidos mientras tiene una precisión de entrenamiento cercana a la de las técnicas centralizadas de bosques aleatorios.

Finalmente, investigamos cómo los sistemas de flujo de trabajo se han utilizado ampliamente para especificar análisis de datos biomédicos y mostramos las limitaciones actuales de esas herramientas. Mostramos cómo ofrecen medios limitados para definir, implementar y ejecutar estudios de sitios múltiples en la infraestructura distribuida actual, respetando la propiedad de los datos y las restricciones de privacidad. A continuación, proponemos FeDeRa, un lenguaje para especificar, implementar y ejecutar flujos de traba-

jo científicos multi sitio compatibles con FDC. El lenguaje está enriquecido con abstracciones para implementar análisis en infraestructuras de nube distribuidas geográficamente y con abstracciones para definir patrones de flujo de trabajo complejos a través de límites de múltiples sitios. FeDeRa admite la programación de flujo de datos y la concurrencia declarativa de forma nativa. También presentamos la implementación de un motor de tiempo de ejecución que admite la ejecución de flujos de trabajo y experimentos de FeDeRa implementados en la infraestructura de la nube.

Acknowledgements

Firstly, I would like to thank God for everything. Then, I would like to express my sincere gratitude to my family for their unconditional support. My parents, María and José, and my two wonderful daughters, **Sharon Nalieth** and **Jael Mariana**, for their trust, patience, understanding, and sacrifice to go so far, and Clara for her perseverance and support. Many thanks to my grandmother Ana for her incessant prayers and good wishes. In memory of my grandparents: *Jael*, *Samuel*, and *Pedro*. To my brother Cristian, and my whole family's example and advice were essential to accomplishing my Ph.D.

I am extremely grateful to my supervisors: Professor Mario Südholt allowed me to explore, experiment, and learn, while his advice empowered me to face challenges that helped me overcome my limits. Professor Luis Daniel Benavides, for his constant help and always being available to give me great advice to improve my research. I also want to mention Alban Gaignard for his collaboration, recommendations, and knowledge sharing to develop my research.

I also thank my alma mater, Escuela Colombiana de Ingeniería, and its directors board, who supported the development of this thesis under cotutelle modality. In Escuela, Dr. Eduardo Silva, who is in the afterlife, thanks for so much trust, and Alicia Guzmán, who both supported me since my undergraduate studies. Myriam Angarita, thanks for your trust to develop my Ph.D. studies. Finally, thanks to the entire community of the Escuela for their support and Oswaldo Castillo for his encouragement to make this possible.

In Nantes, thanks also to the IMT Atlantique and STACK team, who made part of my funding and gave me the fantastic opportunity to live in France and learn so many beautiful things about its people, culture, and food. I also want to thank all my colleagues for many moments of sharing in 'FrenGlish' and also for their productive contributions to my work. Although my gratitude goes out to all of you, I want to thank Fatima-Zahra, my temporary partner during my stay in France, for the guidance, advice, and willingness to share with me during our lunches with the team members.

Finally, I would like to thank my friends near and far, especially: Samuel, Rosita, Edgar, Vladimir, Paula, María, and Simon, because life in Nantes would not have been half as fun without you.

Table of Contents

Abstract **3**

Résumé **4**

Resumen **6**

Acknowledgements **8**

Notations and Acronyms **13**

List of figures **18**

List of tables **19**

1. Introduction **21**
1.1. Context and Motivation 21
1.2. Contributions and structure of the thesis 23
1.3. Publications 24

I Related Work and Concepts **26**

2. The Problem of Global Collaboration in Biomedical Analyses **27**
2.1. Data Sharing, Privacy and Protection in Biomedical Analyses 28
2.1.1. Biomedical Data Categories 29
2.1.2. Data-Privacy: Global Landscape 31
2.1.3. Guiding Principles for Data-Sharing 35
2.2. The ICAN project: a Case of Collaborative Biomedical Analyses 36
2.2.1. The IntraCranial ANeurysm ICAN project 37
2.3. Constraints on Biomedical Collaborations 38
2.3.1. Technical Constraints 39
2.3.2. Legal Constraints 39

2.4. Extending I-CAN to EU and non-EU partners . . . 40
2.5. Conclusions . . . 43

3. Distributed Biomedical Analyses 45
3.1. Distributed Biomedical Analyses . . . 46
3.1.1. Biomedical problems, data analytic techniques, and tools . . . 46
3.1.2. Support for Research Collaborations . . . 48
3.1.2.1. Workflow Description Language . . . 49
3.1.2.2. Experiment Reproducibility . . . 50
3.1.2.3. Workflow System's Interoperability . . . 54
3.1.3. Distributed Architectural Features . . . 56
3.1.3.1. Data and Computation Placement . . . 57
3.1.3.2. Privacy and security . . . 60
3.1.3.3. Architecture and Quality Attributes . . . 61
3.1.4. Distributed Workflow Systems . . . 64
3.2. Machine Learning-Based Analysis . . . 66
3.2.1. Understanding Supervised Learning . . . 67
3.2.2. Ensemble Learning . . . 70
3.2.2.1. Random Forests . . . 73
3.2.3. Multi-Site Forests . . . 74
3.2.4. Distributed Machine Learning . . . 78
3.3. Conclusions . . . 82

II Contributions 87

4. Fully Distributed Collaborations 91
4.1. The FDC concept . . . 92
4.1.1. FDC Properties . . . 92
4.1.2. Data as first-class citizens . . . 95
4.2. FDC-based analyses supported . . . 96
4.3. The Architecture of FDCs . . . 99
4.4. Privacy and Security . . . 99
4.5. Conclusions . . . 100

5. Fully Distributed Random Forests (MuSiForest) 101
5.1. Random Forests Training . . . 102
5.1.1. Training a Model . . . 103
5.1.1.1. Building a Decision Tree . . . 103
5.1.1.2. Decision Tree Application . . . 107
5.1.2. Estimating the Prediction Error . . . 109
5.1.2.1. Metrics for Classification Models . . . 109
5.1.2.2. Out-of-Bag (OOB) Error Estimation . . . 111
5.1.3. Complexity of Random Forests . . . 113
5.2. A Fully Distributed Random Forests Algorithm . . . 113
5.2.1. MuSiForest Methodology . . . 113
5.2.2. The MuSiForest Algorithm . . . 114
5.2.2.1. Local Model Construction . . . 115
5.2.2.2. Model Aggregation . . . 117
5.2.2.3. Evaluating the Collaborative Model . . . 121
5.3. Privacy-Preserving Bias Correction . . . 122
5.3.1. Secure Data Containers (SDCs) . . . 123
5.3.2. Secure Tree Data (STD) . . . 125
5.3.3. Secure Validation Dataset (SVD) . . . 127
5.4. Architecture and Implementation . . . 129
5.4.1. Component-based Architecture . . . 130
5.4.2. System Implementation . . . 132
5.4.3. MuSiForest Topologies . . . 133
5.4.3.1. Sequential Aggregation . . . 134
5.4.3.2. Hierarchical Aggregation . . . 135
5.4.3.3. Fully Distributed Aggregation . . . 136
5.5. Experiments and Results . . . 138
5.5.1. Dataset Description . . . 139
5.5.2. The Grid'5000 testbed . . . 139
5.5.3. Experimental Setup . . . 141
5.5.4. Experimental Results . . . 142
5.5.5. Bias-Correction Evaluation . . . 148
5.5.6. Discussion . . . 151
5.6. Conclusions . . . 154

6. Workflow Language for FDCs (FeDeRa) 155
6.1. Workflow Languages for Biomedical Analyses . . . 156
6.1.1. Distributed Workflow Languages . . . 156
6.1.2. Baseline Workflow Languages . . . 157
6.2. Workflow-based Machine Learning Analysis . . . 159
6.3. FeDeRa: Distributed and Declarative Workflows . . . 161
6.3.1. FeDeRa Language Features . . . 162
6.3.1.1. Dataflow Programming . . . 163
6.3.1.2. Declarative Concurrency . . . 164
6.3.2. Syntax and Semantics . . . 165
6.3.2.1. FeDeRa Instructions . . . 165
6.3.2.2. Transformational semantics of FeDeRa instructions . . . 168
6.3.2.3. Specification of the Collaborative Scenario . . . 171
6.4. Architecture and Implementation . . . 174
6.4.1. Implementation overview . . . 174
6.4.2. FeDeRa Runtime Architecture . . . 176
6.4.3. Deployment . . . 177
6.5. Evaluation and Results . . . 179
6.5.1. Expressiveness Level . . . 185
6.5.2. Concurrency and Distribution . . . 186
6.6. Conclusion . . . 187

7. Conclusions 191
7.1. Future work . . . 193

Glossary 197

Bibliography 201

Notations and Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Networks
API	Application Programming Interface
Bagging	Bootstrap Aggregating
BATTs	Biomedical Analytical Tools and Techniques
BestMdn	Best Aggregated Model by Median
BestPerc	Best Aggregated Model by Percentage
BLM	Build Local Model
BWT	Burrows-Wheeler transform
CCPA	The California Consumer Privacy Act
CDPA	The Virginia Consumer Data Protection Act
CMJoin	Collaborative Model by Joining
CMMdian	Collaborative Model by Median Precision
CMPerc	Collaborative Model by Percentage
DBA	Distributed Biomedical Analyses
DC	Distributed Controller
DML	Distributed Machine Learning
DNNs	Deep Neural Networks
DP	Differential Privacy
DT	Decision Tree
D_{TE}	Testing Data Set
D_{TS}	Training Data Set
DWFS	Distributed Workflow System
EBI	European Bioinformatics Institute
FDC	Fully Distributed Collaboration
FDC-RF	Fully Distributed Random Forest
FDCs	Fully-Distributed Collaborations
FeDeRa	Fully and Enriched Language for Distributed Analysis

FHE	Fully Homomorphic Encryption
FL	Federated Learning
G5K	Grid'5000 platform
GDPR	General Data Protection Regulation
GPU	Graphics Processing Units
GWA	Genome-Wide Association studies
HE	Homomorphic Encryption
HGP	The Human Genome Project
ICAN	The IntraCranial Aneurysms Project
IG	Information Gain
inBag set	In-bag data set
KNN	K-Nearest Neighbor
MC	Model Controller
MEO	Model Evaluation and Outcomes
ML	Machine Learning
MLP	Multilayer Perceptron
MMD	Merge Models
MPI	Message Passing Interface
MuSiForest	Multi-Site Random Forest
NCBI	The National Center for Biotechnology Information
NGS	Next-Generation Sequencing
oob set	Out-of-bag data set
P2P	Peer-to-peer
PCAWG	The Pan-Cancer Analysis of Whole Genomes
RF	Random Forest
RNA-Seq	RNA-Sequencing
SC	Site Controller
SDC	Secure Data Containers
SMC	Secure MultiParty Computation
SSS	Single-Share Store
STD	Secure Tree Data
SVM	Support Vector Machine
SWFMSs	Scientific WorkFlow Management Systems
VIM	Variable importance
WDL	Workflow Description Language

List of Figures

2.1. Data privacy regulations in the global landscape 31
2.2. Hospitals, data flows and processing sites of the ICAN project 37
2.3. Data analyses process among Nantes and Rennes in the ICAN project . . . 38
2.4. Scenario for distributed processing analysis in Fully Distributed Collaborations across four sites 41

3.1. High-Throughput Sequencing (HTS) data analysis protocol (based on [De +12]) 46
3.2. General idea of the learning process approach. 68
3.3. Supervised and unsupervised learning. Supervised learning trains a model to best separate two classes $\{+, -\}$. Unsupervised learning seeks to cluster samples based on common features. 69
3.4. Bootstrapping strategy, from the initial dataset D with m records, then m samples are selected with replacements to generate one bootstrapped data set. 71
3.5. Bagging strategy for training and evaluating multiple classifiers in an ensemble learning model. 71
3.6. Representation of a data set, in two forms, by a decision tree with two attributes (x_1, x_2) and one binary class $\{+, -\}$, and by decision boundaries. 72
3.7. Random Forests algorithm. 73
3.8. Distributed Machine Learning aggregation strategies. 78
3.9. Horizontal Data Partitioning. 80
3.10. Vertical Data Partitioning. 80

4.1. FDC analysis between four cities distributed in two countries. 93
4.2. FDC Machine Learning over multi-site workflows. 97
4.3. FDC Fully-Distributed Architecture. 99

5.1. Decision tree modeling a data training process with thousands of genes, multiple classes, and five paths from the root to leaves. 108

5.2. An illustration of OOB error estimation for each decision tree 112
5.3. Construction and evaluation of the RF model under Fully Distributed Collaborations. The process involves s sites. From each dataset a training (D_{TR}) and testing (D_{TE}) set are generated that are used on the local and collaborative models privately. 114
5.4. Secure Data Container strategy to share partial data securely. 123
5.5. Secure Tree Data strategy based on FHE to build each level tree securely. . 125
5.6. Bias-correction strategy based on a secure global validation dataset. 127
5.7. Architecture of the MuSiForest engine. 130
5.8. Execution Architecture the MuSiForest algorithm. 131
5.9. Diagram of deployment components among three sites. 132
5.10. Sequential aggregation, each site builds the local model and shares it with others until all sites participate sequentially. 135
5.11. Hierarchical aggregation with one intermediate level, each site builds a local model and shares with the intermediate sites. The intermediary collects the models to the top and then shares the collaborative model with all sites. . 136
5.12. Example of Fully distributed aggregation, through peer-to-peer communication, the sites share the models until the participation of all results in the collaborative version. 137
5.13. Grid'5000 ($G5K$) nodes distribution. 140
5.14. Experiments deployed with the participation of s sites versus the centralized training. 142
5.15. Comparison of metrics for collaborative and centralized models in 3 Sites. . 143
5.16. Comparison of metrics for the models: BestPercCollaborative (BCoM) and Centralized (CeM) in different distributed scenarios. 144
5.17. FScore metrics for collaborative and centralized models in different distributed settings (H:Hundreds, K:Thousands) based from Table 5.2. 146
5.18. Amount of data shared between collaborative models and moved raw data in the centralized case. 147
5.19. Computational Time Training a Model: centralized versus collaborative versions. 148
5.20. Comparison of local models per site versus the Secure Data Containers (SDC) strategy for three sites. 150

6.1. Workflow for collaborative training between CO and FR (described in Section 2.4). (BLM: Build a Local Model, GLM: Group Local Models, SLD: Share Local Data, MMD: Merge Models, SLM: Share Local Models, SCM: Share Collaborative Models, MEO: Model Evaluation, REO: Receive Evaluation and Outcomes). 160
6.2. Dataflow representation between French and Colombian sites that share information on one variable E. 163
6.3. Syntax of FeDeRa language. 166
6.4. A distributed machine learning scenario among three sites applying the steps: Build a Local Model (BLM), Merge Models (MMD), and Model Evaluation (MEO). 167
6.5. FeDeRa analysis workflow among three sites from Figure 6.4 168
6.6. FeDeRa compilation process. 169
6.7. Transformation of Single Share Store functionality 169
6.8. Implementation of the subscriber site to the dataflow variable varName as *Python* code 170
6.9. Implementation for binding a dataflow variable in *Python* code 170
6.10. FeDeRa specification for the global conditions of the collaborative scenario presented in Figure 6.1 (introduced on page 40) 171
6.11. FeDeRa specification for Colombian site in the collaborative scenario illustrated in Figure 6.1 172
6.12. FeDeRa specification for French sites in the collaborative scenario presented in Figure 6.1 173
6.13. Communication between two sites via the Single Share Store *SSS*. 174
6.14. Example of a Wait-For-Graph (WFG) strategy for handling deadlocks between three sites. 175
6.15. FeDeRa distributed runtime architecture. 176
6.16. FeDeRa communication distributed architecture. 177
6.17. Grid'5000 (G5K) nodes distribution with FeDeRa and FDC components. . 178
6.18. Sample of configuration file to deploy multi-site analyses on *G5K* 178
6.19. Workflow to split a single input into four independent files on which word frequencies are calculated output as a single file. 180
6.20. Specification in Swift for the workflow analysis presented in Figure 6.19 . . 181
6.21. Specification in Snakemake for the workflow analysis presented in Figure 6.19182
6.22. Specification in Pegasus for the workflow analysis presented in Figure 6.19 183

6.23. Specification in FeDeRa for the workflow analysis presented in Figure 6.19 184

List of Tables

3.1. Characterization of BATTs for genomic data analyses 47
3.2. Workflow description characteristics of the SWFMSs 50
3.3. Computational reproducibility characteristics in the SWFMSs 53
3.4. Interoperability properties in the SWFMSs 55
3.5. A categorization of BATTs and SWFMSs according to our architectural features proposed 57
3.6. Sequential and cluster-based BATTs for genomic data (complements Table 3.1) 62
3.7. Classification of SWFMSs according to the level of distribution offered. . . 64
3.8. Comparison of some Supervised Machine Learning models 75

5.1. Confusion matrix for a binary classification problem. 110
5.2. Evaluation of the FScore metric for the trained models 145
5.3. Description of each data set in each site experimented in the Secure Data Containers' strategy. 149

Chapter 1

Introduction

1.1. Context and Motivation

The amount of data collected continues to grow every day [Ste+15]. According to the European Bioinformatics Institute (EBI) archive, at the end of 2020, raw data reached over 390 petabytes of data storage [Can+22]. Biotechnological advances like Next-Generation Sequencing (NGS) technologies have contributed to this growth [PST11]. For example, sequencing a whole genome nowadays requires only one day, in contrast to the more than ten years invested in the Human Genome Project (HGP) between *1990* and *2003* [BT13]. Therefore, biomedical data, particularly genetic data, is voluminous and often too difficult or costly to transfer, store at, and analyze on a single site [Pap+18a].

Today's predominant architectural model for biomedical collaborative analyses consist of centralizing the underlying data and performing analyses through supercomputers or cluster infrastructures located at a single or a small number of organizations [Sch+10a]. Furthermore, this collaborative model is very restrictive and rigid. Recently, the need for more widely distributed collaborations has been noted [BS18; Bou+19; PCA15; Li+16]. Several arguments favor a higher degree of distribution: more and more organizations dispose of high-performance infrastructures for large-scale analyses, biomedical local data should be kept private, and massive data transfers are too time-consuming. We refer to biomedical analyses as those workflows composed of multiple steps, even distributed, to analyze data from medical applications: images, medical records, and data resulting from genomics analyses.

Approaches and tools for fully distributed collaborative biomedical analyses are rare today. Most existing approaches and tools process data at individual or a small number of locations using efficient frameworks for large-scale computations, such as MapReduce [Cat+17a; Sal+16; Guo+18]. MapReduce-based frameworks like Hadoop and Spark are highly efficient for distributed processing across a large number of machines orchestrated by a central node. But, these frameworks have limitations when processing geographically

distributed data while maintaining quality attributes such as scalability, consistency, and performance [Dol+17b]. Similarly, workflow systems, another popular tool used in the biomedical field, allow scientists to define analyses in terms of tasks as well as task dependencies and dataflows [Liu+15]. They also frequently support portability across different execution environments like grids and clusters [YB05; RB17]. But, current workflow systems do not support FDC scenarios because they are designed to cope with current state-of-the-art infrastructure. They are also not designed to cope with the complex requirements of international biomedical cooperation, lacking mechanisms for decentralization, distributed computations, and security/privacy requirements [PCA15].

These restrictions on computational capacity and processing mechanisms are complemented by security/privacy requirements and data-sharing restrictions on biomedical data. Security and privacy risks include data leakage and re-identification concerns [Are+18]. Therefore, different governments and official entities are trying to regulate biomedical data sharing. A prime example is the General Data Protection Regulation (GDPR) [CC16], implemented in *2018* by the European Union. The GDPR defines rules for the treatment and sharing of personal data, including, among others, health data, genetic, and biometric data.

In order to address these restrictions, we propose fully-distributed collaborations (denoted as FDCs) to process biomedical data. FDCs are research endeavors that harness means to exploit and analyze massive biomedical information collaboratively over geo-distributed infrastructures. Thus, FDCs require tools and techniques for collaboration that can use advanced distributed (data and computation) architectures to cope with complex socio-technical constraints and heterogeneous networks. FDCs promise to enable more powerful biomedical analyses defined in distributed workflows operating over large volumes of shared public and private data. FDC analyses will promote cooperation among geo-distributed research groups or organizations, each typically subject to, possibly local, constraints stemming from legal frameworks, security constraints, sensitive private data, and locally-available infrastructures. Thus, FDCs require collaboration beyond traditional social/legal ones, such as confidentiality agreements or copyrights contracts.

In this thesis, we propose an approach to analyze biomedical data in a fully distributed setting. An FDC-based scenario implements a supervised learning algorithm to analyze biomedical data collaboratively. The FDC scenario considers the global training of a Random Forests model through the aggregation of models trained locally without sharing data entirely [Liu+20b]. Random Forests [Bre99] is a popular learning model applied to

biomedical data for classification and prediction problems [GPB11; CI12; Bou+12]. We implement this FDC-based scenario and experiment with data resulting from the analysis of genomic data. Additionally, we complement this algorithm with an enriched language to specify instructions declaratively. The language syntax offers means to improve expressiveness and interpretability that are very useful for biomedical users.

1.2. Contributions and structure of the thesis

In this thesis, we provide the following contributions:

- We motivate the use of FDC scenarios as an improvement of centralized biomedical analyses. We show current limitations and constraints in the context of the ICAN project [ANR19], a large collaborative project that involves *34* french hospitals and research centers working together to understand the pathology of intracranial aneurysms.

- We present a taxonomy of tools and systems for distributed genomic data processing. The taxonomy classifies from three different perspectives: (i) biomedical problems being solved on genomic data, (ii) the tool support provided for biomedical analyses, and (iii) support for distributed cooperation, notably in terms of types of distribution offered, interoperability properties, and reproducibility properties.

- We propose the **MuSiForest** algorithm to build a collaborative model between multiple sites through different model aggregation strategies applying different security levels. The algorithm supports the collaboration among several geo-distributed research centers involving distributed biomedical data and computational infrastructures. Furthermore, the implementation offers some strategies to handle the bias in the trained collaborative model sharing partial data securely.

- We also propose declarative language to specify FDC-based analysis. The declarative workflow language, **FeDeRa**, specifies collaborative biomedical analysis and is implemented across multiple geo-distributed sites. Furthermore, the language adopts concurrency and distributed mechanisms to mitigate current workflow language restrictions, enhancing the specification-level analysis definitions required by biomedical users.

- Finally, we propose multi-site analyses executed in an international context between Colombia and France to evaluate our algorithm and the proposed specification language. Each site disposes of private data, and computing and storage infrastructures at the different sites are heterogeneous. The collaborative analysis is defined as a workflow over computations steps and data movements. The MuSiForest algorithm is deployed among geo-distributed sites, and through collaboration, we train a collaborative model, which is evaluated against expensive and unfeasible centralized training. Furthermore, we have specified this scenario with our declarative language contrasting the expressiveness level and deployment mechanisms provided versus some workflow languages.

This thesis comprises two parts in seven chapters:

Part I: The first part, we present the related work and basic concepts to comprehend the thesis in two chapters. Chapter 2 presents relevant features in the problem of global collaboration in biomedical analyses. We also present a motivational case based on real biomedical scenarios. Next, Chapter 3 presents the state and concepts related to distributed biomedical analyses with traditional analysis and based on Distributed Machine Learning strategies.

Part II: The second part presents the three contributions of the thesis. First, chapter 4 presents Fully-Distributed Collaborations (FDCs) and their relevant processing features. We then present a collaborative learning approach following the FDC concept in Chapter 5. Thirdly, Chapter 6 describes the distributed workflow specification language for biomedical analyses. Finally, we present the dissertation's conclusion and potential directions for future work in Chapter 7.

1.3. Publications

The thesis's development has led to papers, some in revision and others in co-authorship, through collaborative research on topics of interest in this dissertation.

- **Garzón, Wilmer** and Benavides, Luis and Gaignard, Alban and Redon, Richard and Südholt, Mario. (2022, July). A taxonomy of tools and approaches for distributed genomic analyses. Informatics in Medicine Unlocked. https://doi.org/10.1016/j.imu.2022.101024

- Sanabria-Ardila, M., Navarro, L. D. B., Diaz-Lopez, D., & **Garzón-Alfonso, W**. (2020). A Semantic Framework for the Design of Distributed Reactive Real-Time Languages and Applications. IEEE Access, vol. 8, pp. 143862-143880, 2020. https://doi.org/10.1109/ACCESS.2020.3010697

- Boujdad, F. Z., Gaignard, A., Südholt, M., **Garzón-Alfonso, W.**, Navarro, L. D. B., & Redon, R. (2019, May). On distributed collaboration for biomedical analyses. CCGrid-Life 2019 Workshop on Clusters, Clouds and Grids for Life Sciences (CCGRID) (pp. 611-620). IEEE. https://doi.org/10.1109/CCGRID.2019.00079

- **Garzón, Wilmer** and Benavides, Luis and Gaignard, Alban and Redon, Richard and Südholt, Mario. A Fully-Distributed Random Forest approach for biomedical data analysis. (revision process).

Part I

Related Work and Concepts

Chapter 2

The Problem of Global Collaboration in Biomedical Analyses

The volume of biomedical information collected is growing every day [Ste+15]. Individuals or single organizations often can no longer analyze such amounts of data. Researchers are thus organizing collaborative research efforts to address this issue. An example of such an effort is the ICAN project [Bou+17; ANR19], a collaboration of more than 30 French medical institutions in a research project that targets pathologies pertaining to intracranial aneurysms. In the ICAN project, each participant shares large amounts of biomedical data (including clinical records, medical images, and genomic data from the sequencing of biological samples). Although multiple institutions provide data, the biomedical data is analyzed by only a small number (2) of participating entities.

In this thesis, we consider biomedical analyses defined as distributed workflows employed to analyze data from medical applications: images, medical records, and data resulting from genomics analyses, such as gene expression levels[1]. Analyzing this data, even in a collaborative manner, will allow insights from complex and voluminous data to help researchers and health professionals make evidence-based decisions to improve human health.

Biomedical data is subject to more numerous and more severe limitations on data sharing than most other kinds of information. Governments and other organizations are frequently imposing regulations and policies to protect biomedical data, such as the European Union's General Data Protection Regulation (GDPR) [CC16]. However, such restrictions frequently make conducting research collaborations more difficult.

In this chapter we investigate how to address the data protection problem while provid-

1. Gene expression levels determine how genes are transcribed into functional gene products such as functional RNA or proteins [Kor+14; Liñ+19].

ing flexible means for distributed data analyses. In Sec. 2.1, we study the characteristics of biomedical data, notably limitations pertaining to the privacy, protection, and the sharing limitations of such data. In Sec. 2.2, we present the ICAN project in detail as a case study how such affect research efforts. Section 2.3 highlights and classifies the problems of research collaboration into three categories: technical constraints, legal constraints, and socio-economic constraints. Section 2.4 discusses how another research collaboration scenario among several geo-distributed parties may benefit from more flexible tools for collaboration. Finally, Sec. 2.5 presents a conclusion.

2.1. Data Sharing, Privacy and Protection in Biomedical Analyses

Sharing biomedical data presents considerable challenges due to its privacy and confidentiality requirements compared to other data types. Governments and research organizations have promoted policies and practices to share data responsibly to meet these requirements. For example, the Genomic Data Sharing (GDS) policy [Hea21] and the Strategic Plan for Data Science (SPDS) [Hea+18], both proposed by the National Institutes of Health (NIH) in the U.S. The GDS policy is an initiative to share genomic data among NIH-funded projects. While the SPDS aims to interconnect data generated from different NIH projects integrating and standardizing tools and algorithms to manage the data efficiently. Other research agencies have taken similar initiatives as ELIXIR-EXCELERATE [ELI14], proposed in 2015 by ELIXIR, which seeks to integrate several bioinformatics resources supporting research and development in the European community. Other actors are governments or government-like entities that often define policies to limit data sharing. The most relevant regulation is the EU's GDPR [CC16] which defines rules on sharing and processing data from Europeans. In the U.S, the California Consumer Privacy Act (CCPA) [Leg] defines California residents' privacy guarantees and protection mechanisms. Therefore, collaborative biomedical data analyses must consider government policies while favoring those practices defined by research funding agencies.

Data sharing regulations also limit international biomedical cooperations. A notable example are Genome-wide association studies (Genome-Wide Association (GWA)) that help scientists identify genes associated with a particular disease. GWAS operate on a group of patients' entire genome set (i.e., DNA), analyzing small variations called Single Nucleotide Polymorphisms (SNPs). The GWAS catalog [Bun+19] contains a vast amount

of genomic data such as SNPs associations. It is managed by the NHGRI (National Human Genome Research Institute) and the EBI (European Bioinformatics Institute). The data is hosted on central sites managed by them, but they facilitate the participation of researchers by sharing and downloading statistical analyzes of genomic data. The General Data Protection Regulation (GDPR) also restricts genomic data sharing such as those contained in the GWAS catalog due to strict rules for de-identification and consent [Pel+20]. These restriction rules are motivated by case studies that have shown a direct linkage with patients from GWAS data, such as analysis from summary statistics [Hom+08] or quantitative traits [Im+12]. However, researchers have sought practices to promote international cooperation taking into account data sharing restrictions as defining codes of conduct for genomic data sharing [MK20]. The codes of conduct seek to respect regulations to delegate genomic research institutes as those responsible for their control and processing. Nevertheless, the definition of these codes has required a significant dedication of time and resources because, in many cases, it is not clear how to resolve the differences in data-sharing responsibilities between the parties [Uff+21].

2.1.1. Biomedical Data Categories

Many Data sharing regulations, such as the GDPR, define how data is shared based on categories of biomedical data. Therefore, it is important to know the different types of biomedical data because some of these regulations handle different types differently.

Personal Identifiable Data

Personal Identifiable Data (PID) is a set of information (attributes) that directly or indirectly permits to identification a person, as defined by the GDPR Article 4. In the U.S., this kind of data is known as Personal Identifiable Information (PII) [Lab21]. Some examples of identifiable data are attributes such as name and identification number that directly identify a person. However, other data, such as demographic information, also permits the identification of individuals. For example, Sweeney et al.. [Swe00] identified part of the U.S. population by combining demographic data such as gender, date of birth, and zip code. They thus demonstrated that, while such data is not sensitive, it can be used to determine a person's identity.

Sensitive Data

Sensitive data is a special case of personal data that should have a greater protection because it reveals sensitive information, and its disclosure is limited. The GDPR in articles 9 and 10 defines various types of sensitive data such as racial/ethnic, genetic and biometric data to identify a natural person uniquely, and information concerning health data. Sensitive data is subject to specific processing according to GDPR conditions. These conditions are a set of rules that apply to sensitive data before processing, such as anonymizing, pseudonymizing, or encrypting the data whenever possible. The GDPR suggests defining a data controller responsible for defining the methods and practices to process sensitive data. The controller is the principal researcher or the funding entity that regulates how sensitive data should be processed. On the other hand, the data processor is the one who processes them on behalf of the controller.

Anonymous Data

According to recital 26 of the GDPR, anonymous data does not lead to the identification of persons. The scope of the GDPR is outside of anonymous data. Therefore, biomedical analyses must implement efficient anonymization methods to offer more freedom to process biomedical data collaboratively. These methods consider various strategies and seek to keep the data anonymous.

De-identification methods. The first approach is based on de-identification strategies seeking to avoid revealing a patient's identity, such as removing patient identifiers from data. The de-identification methods comply with HIPAA rules to preserve the patient's identity. These rules suggest excluding specific identifiers from the data before sharing. HIPAA rules recommend removing as many as 18 identifiers from the data, such as patient names, demographic information, unique numbers like IDs or phones, and biometric identifiers, including finger and voiceprints. Nevertheless, de-identifying patient information does not always guarantee to keep the data secure because these methods only partially anonymize the data.

Anonymization methods. These methods seek to preserve personal data by removing or encoding attributes that can lead to identifying a natural person. The application of these techniques should maintain the inherent knowledge of the attributes of the data set as far as possible. For example, anonymization techniques are inappropriate when the analysis result seeks to guide or personalize a specific treatment because the data is

modified in an anonymized version.

Pseudonymous Data

Pseudonymization is a method to exchange the original value by a pseudonym. The methods reduce the risk of sharing sensitive data. Regulations such as the GDPR promote pseudonymization to preserve the intrinsic relationship between the attributes while allowing researchers to extract knowledge from pseudonymous data (Article 4, GDPR [CC16]). Pseudonymization methods are reversible and allow de-identification of the raw data. In contrast, anonymization limits the reversion process because breaks the relationships between the attributes. A simple example of a pseudonymization algorithm might be to change a patient's first and last name to an encrypted numeric value, that is, replace these two attributes with a pseudonym value.

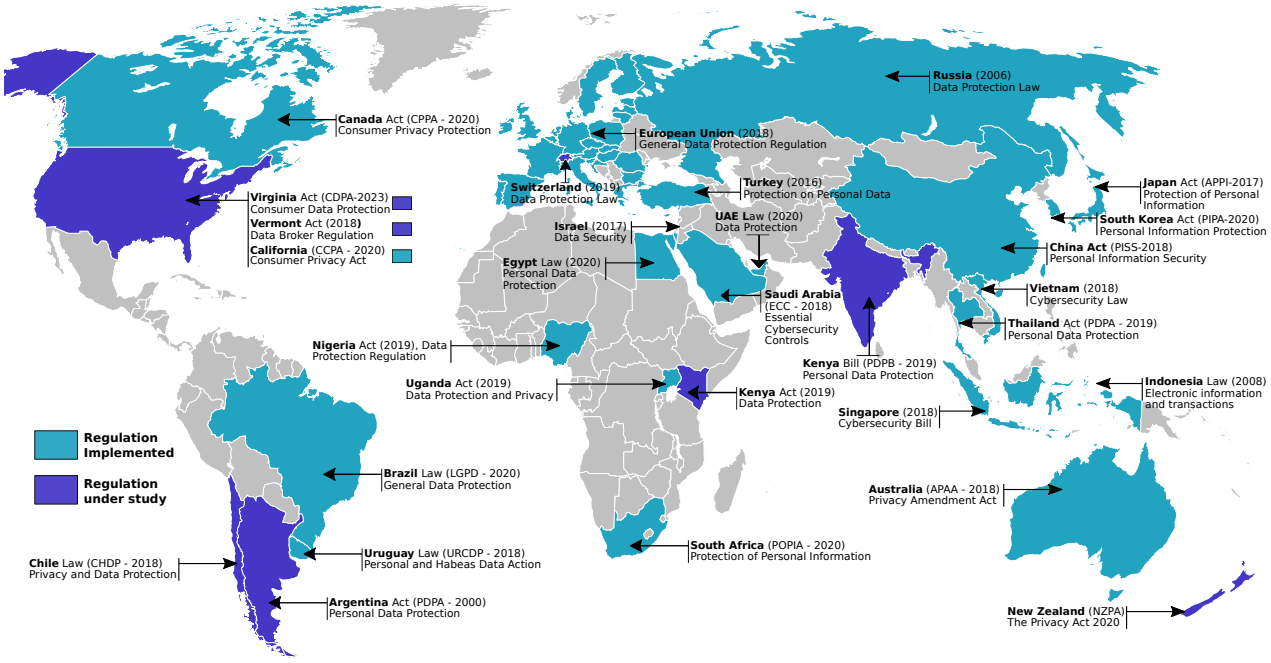


Figure 2.1 – Data privacy regulations in the global landscape

2.1.2. Data-Privacy: Global Landscape

Figure 2.1 illustrates the most relevant protection regulations on data sharing worldwide and the status of each one, implemented or under study. The information presented comes from different sources such as data-privacy reports [Glo; Sana; Sanb; Map] and

the website of each illustrated regulation. Some of these regulations are discussed in this section.

The General Data Protection Regulation (GDPR)

In 2018, in Europe, the General Data Protection Regulation (GDPR) [CC16] was proposed as the first data regulation worldwide. The GDPR defines directives and rules concerning the data privacy of the personal data of European Union residents. The GDPR has replaced the Data Protection Directive 95/46/E.C. [PC95] introduced in the E.U. in 1995. The evolution of the directive resulted from more than four years of work between the European Parliament and the European Council.

The GDPR establishes good practices for handling personal data that lead to people's direct or indirect identification. These practices are proposed through nine principles: legality, equity, transparency, purpose limitation, data minimization, accuracy, storage limitation, integrity and confidentiality, and accountability. The main objective is to safeguard the privacy of European data, being strict with the free processing by laboratories and companies.

GDPR has a world scope since it regulates how non-European research groups must analyze European patient information. Therefore, international cooperations including research teams and medical institutions must define protect-internal frameworks to respect these regulations. The non-compliance with GDPR directives leads to penalties for companies and research teams, such as monetary sanctions. The restriction also extends to international websites that must strive for compliance with the GDPR. A notable case was the Los Angeles Times newspapers that blocked European users' access. In addition to the legal impact, the GDPR recognizes the importance of implementing systems that adopt these legal frameworks to regulate the sharing of personal data.

Data Privacy Laws in the U.S.

The U.S., in the 1970s, pioneered federal restrictions on how government entities access data using database systems due to its popularity. Two decades later, the NIH has promoted the collaborative analyses of biomedical data satisfying the privacy rules formalized as part of the HIPAA (Health Insurance Portability and Accountability Act) [EAH18], encouraging companies to adopt good handling practices. Additionally, each state is defining its regulations. For example, the California Consumer Privacy Act (CCPA) [Leg] gives California residents the right to know what companies know about them, and they can

disagree when companies seek economic benefits from their data. CCPA was the first state-level regulation in the United States and turned into effect on January 1, 2020. The CCPA recognizes as personal information all information that makes it possible to identify a California resident. Biomedical information such as biometric and health data are part of this category. The law applies to California residents who act individually or as a family but it excludes those who sell data in business contexts. However, residents can know, suppress, and revoke prior consent on handling their data, in addition to other rights, such as ensuring non-discrimination and personal safety. Entities that process data of California residents must define good practices of applying the CCPA standard, thus ensuring compliance and complete protection of the personal data of their residents.

Similarly, the Virginia Consumer Data Protection Act (CDPA) [Leg20b], will come into 2022 and applies to all companies or institutions that manage the data of citizens of Virginia state. The law is flexible and excludes all organizations except government, financial, education, or research laboratories. The CDPA only considers direct personal data of the person that leads to the identification omitting aside indirect data or derived data. Genetic or biometric data are considered sensitive and therefore regulated. In addition, the law regulates the sale of personal data, such as exchanging personal data for monetary gain by controlling the maximums permitted during annual transactions.

The Brazilian General Data Protection Law (BGDPL)

In South America, a relevant regulation is the Brazilian General Data Protection Law [Rep18] in force since 2020. Brazilian law also covers companies and institutes that process the personal data of their residents inside and outside Brazil. The BGDPL defines personal information, similarly to GDPR, as any direct or indirect data that leads to the identification of a person, including data processed electronically. The Brazilian regulation extends the GDPR concerning research data and medical procedures, considering, among others, that research teams must guarantee the anonymization of personal data whenever possible. Furthermore, data protection applies to the result of medical procedures in Brazil. Regarding data transfer, it maintains the provisions of the GDPR. However, it makes international sharing more flexible as long as an inter-institutional agreement guarantees responsible comparison, respecting the privacy management established in Brazilian law [EU20].

The Act on the Protection of Personal Information (APPI)

The Act on the Protection of Personal Information (APPI) in Japan [JAP20], proposed in 2017, follows the world's concerns about data protection. The law applies to national and foreign institutions that process data of Japanese citizens. The APPI and GDPR have defined a reciprocal adaptation agreement on restrictions to protect their residents' data in each territory [Eur20; GG20]. The E.U. considers a security list of Japanese companies that responsibly protect data, and reciprocally, Japan has created a list of trusted E.U. companies. The APPI seeks to protect the personal data of living Japanese residents and personal data such as race, social creed, and medical records. The processing of personal data must be regulated, and the informed consent of each Japanese citizen must define the scope. However, the regulation establishes an exception to consent in the different cases determined by the government, such as protecting any risk to life or national security and improving public health policies. The law focuses on commercial companies and defines some as the notification or publication of the purpose of personal data through informed consent.

The Protection of Personal Information Act (POPIA)

Finally, the Protection of Personal Information (POPIA) in South Africa [Rep20] was enacted in 2020. POPIA regulates living persons' data and considers data of juridic persons domiciled in South Africa. It considers medical and biometric information as personal data. However, POPIA does not refer to the handling of genetic data obtained from biological samples or derived from genomic data. POPIA does not directly reference data generated from scientific research or data obtained from the analysis for research purposes. Unlike the GDPR, the law does not consider pseudonymization mechanisms. The legislation does not indicate the transfer of cross-border data. The POPIA regulation has significant gaps compared to others, nor does it indicate how foreign companies should process data from their citizens.

Discussion

GDPR is the most complete regulation on data privacy and has been used as a reference worldwide. It was the first example of how data controllers and processors should protect the data of citizens inside and outside a territory. Many countries have defined less strict and vague regulations in this same direction. In contrast, state-level legislation

in Virginia prioritizes business interests such as sales and economic remuneration. On the other hand, the legislation in China defines rules of common interests for the government where the policies seem to serve a totalitarian purpose. To conclude, there is a diversity of regulations, but many neglect specific policies on biomedical data, while others are quite rigorous. Nevertheless, research organizations should promote practices for responsible data sharing within international cooperations aligned with data privacy regulations, and computational tools must be flexible enough to accommodate easily those considerations.

2.1.3. Guiding Principles for Data-Sharing

During the last few years, there has been a notable interest in sharing genomic data to achieve remarkable advances in science for the benefit of human health. Sharing biomedical data is governed by legal restrictions and a lack of interoperability protocols. The need to share responsible data inspired the genomics research community to establish the Global Alliance for Genomics and Health (GA4GH) in 2014 [Ter14]. The major goal of GA4GH is to benefit genomics and health science through standard policies for sharing scientific data responsibly. As part of GA4GH's commitments, they have adopted the FAIR principles to promote standard protocols in data sharing.

The FAIR Guiding Principles

The FAIR guiding principles provide guidelines to improve the findability, accessibility, interoperability, and reuse of scientific data.

- *Findable*: Advancement in research benefits when researchers can find data and metadata from other scientific analyses. Finding the data should be an easy task for scientists in an automated way. findable is one of the most important qualities in FAIR principles for the consequences tasks after the discovery.

- *Accessible*: The researchers must access the data found, possible through controlled access with prior registration.

- *Interoperable*: The data must be integrated with other data sources or external applications to enable more extensive analyses due to the participation of different sources. The data must follow standards that permit reuse and exchange with others.

- *Reusable*: The reusable data allows new findings or the refutation of previous research results. Therefore, data and metadata must be annotated to be understood by other researchers.

The FAIR principles ignore technology aspects of scientific data analyses. This responsibility is delegated to the researchers and the entity responsible for financing the project. Nonetheless, in 2016, the G20 member countries endorsed the principles as a relevant guiding force for research data. Similarly, research funding entities have been motivated to follow FAIR principles on research in scientific data. In addition, the FAIR principles have been adopted in various international initiatives for managing and sharing biomedical data. For example, the NIH Big Data to Knowledge (BD2K) initiative [Mar+14] adopts the principles to maximize the use of biomedical data to extract knowledge (value added to raw data) for individual researchers and the scientific community as a whole. Since the NIH is a part of the world's data ecosystem, this kind of effort between research project funding organizations and policy-making entities will lead to standard models to manage data and obtain new knowledge collaboratively. Similarly, in Europe, ELIXIR [ELI19] comply with the FAIR principles as part of its effort for the efficient management of biomedical data by publicly funded projects. All these efforts demonstrate the interest of different entities to open data responsibly while promoting research.

2.2. The ICAN project: a Case of Collaborative Biomedical Analyses

The volumes of biomedical data, in particular genomic data, have grown remarkably [Ste+15], such as the data collected in the EMBL-EBI (European Bioinformatics Institute) archive, including clinical and genomic data [Can+22]. The EMBL-EBI reached over 390 PetaBytes of raw data storage by the end of 2020, from approximately 70 PB in 2015 and 160 PB in 2018 [Coo+19]. The availability of such data presents new opportunities and challenges to extract actionable insights from data for biomedical research.

Biomedical collaborations can have a global scope, such as the GWA studies, but are also highly relevant in national and regional scenarios. One corresponding example is the IntraCranial ANeurysm ICAN project [Bou+17] project in France. This project involves 34 collaborating French hospitals. Although the project is a French national effort financed by the government, data sharing with non-project members is severely restricted by legal

restrictions and technical challenges. Part of these challenges can be mitigated by an extension of the ICAN project's distributed computing infrastructure supported by the idea of multi-site analyses discussed in this thesis.

2.2.1. The IntraCranial ANeurysm ICAN project

The project aims at developing diagnostic and predictive tools for the risk of intracranial aneurysm formation and rupture from three types of data [Bou+17]: clinical data (data related to environmental risk factors are collected for each included patient), imaging data (MRI/MRA data[2] from imaging scans), and biological samples (blood samples taken from each patient whose genetic information has been sequenced).

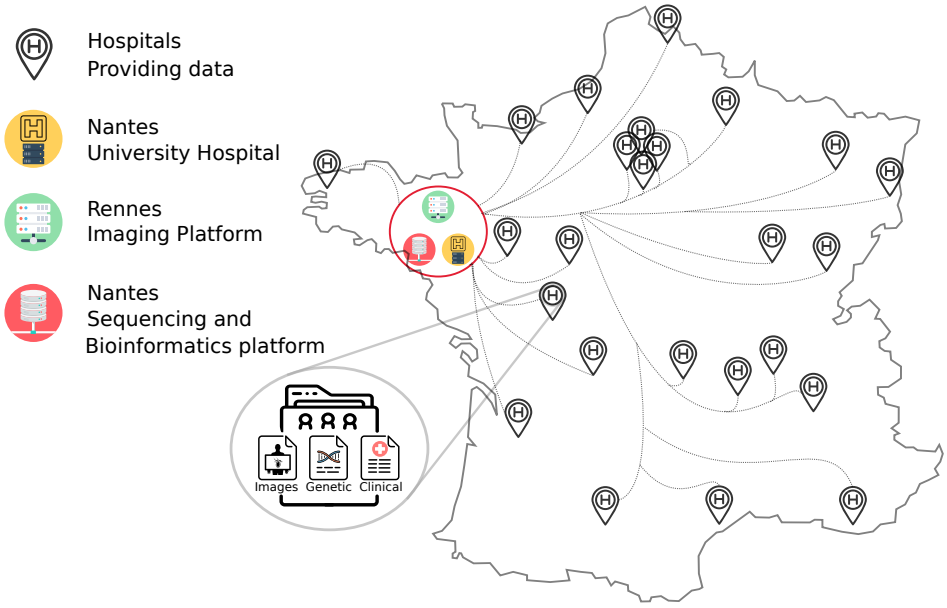


Figure 2.2 – Hospitals, data flows and processing sites of the ICAN project

Figure 2.2 shows the parties involved in the project. The figure shows that most sites provide three types of data, while only hospitals and research institutions in two cities (Nantes and Rennes) ensure their processing and (intermediate) storage.

Figure 2.3 details the process performed on computing platforms (different clusters) located in Nantes and Rennes. The medical images and their metadata are uploaded using the Shanoir-NG (SHAring iN vivO Imaging Resources, Next Generation) neuroimage data

2. Types of magnetic resonance examinations: MRI (Magnetic Resonance Imaging) and MRA (Magnetic Resonance Angiography).

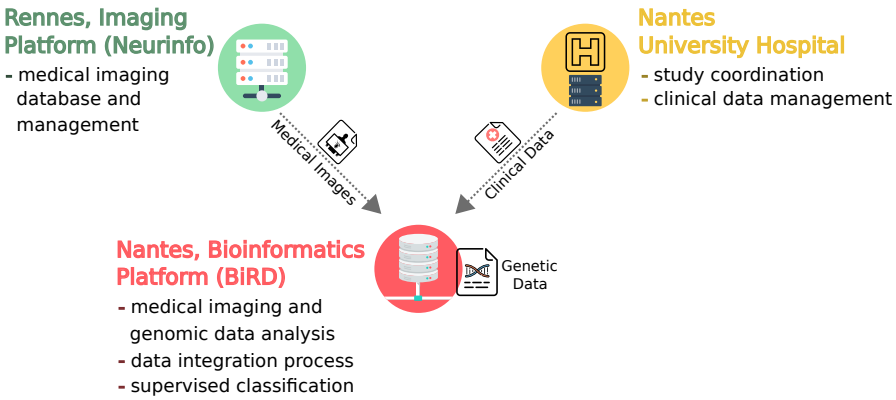


Figure 2.3 – Data analyses process among Nantes and Rennes in the ICAN project

sharing platform [Bar+16], and these are managed and stored on an imaging platform (Neurinfo) in Rennes. Subsequently, the images are transferred to Nantes. In Nantes, the analysis occurs at two places: at the university hospital and on a supercomputing cluster BiRD (Bioinformatics institute Research and Development) of an independent research institution.

The BiRD platform also enables the analysis of transferred images and processes genetic data. Genetic data is obtained on-site through High-Throughput Sequencing and array genotyping techniques from biological samples sent by all hospitals. Thus, although the ICAN project involves multiple hospitals, the data analysis process is performed at only two sites. As discussed below, the project structure described above already poses several technical, legal, and socioeconomic challenges, foremost related to data storage, data sharing, and computational requirements. In addition, replicating the study abroad or sharing data with other countries would require overcoming legal constraints and considering alternatives to centralizing storage and processing.

2.3. Constraints on Biomedical Collaborations

Similar to the ICAN project, real-world cooperations and the corresponding body of scientific work in almost all of today's cases are strongly limited. Most calculations are implemented on infrastructure on a single site, and few data are effectively shared without constraints, especially in international settings. There are multiple reasons for the current restrictions, technical ones but also legal ones discussed below based on the ICAN project.

2.3.1. Technical Constraints

Biomedical cooperations are subject to significant technical challenges. For example, the ICAN project requires processing large volumes of data and performing complex and time-consuming computations. Therefore, the data is centralized and processed only in two sites (illustrated in Figure 2.3) because the other sites do not dispose of the necessary hardware and software infrastructures. The data volumes grow at least linearly with the number of involved patients. In this context, storage, and communication bottlenecks arise easily [Pap+18a], leading to major questions related to the localization/placement of computation and data, notably for performance reasons.

Second, the ICAN project uses a simple distributed collaboration architecture because data is generated at all sites but then centralized and processed at only two sites out of 34. More widely distributed architectures between the participating sites would have led to larger distributed executions of analyses performing less data movements and thus could have been more efficient and cost-effective. They could, however, not be employed because of insufficient computational facilities at many sites. In general, current computational infrastructures and data storage methods lack support for the efficient distributed processing of massive biomedical data [FHL14].

Third, because of its mostly centralized execution architecture, the ICAN project has a simple architecture in terms of security and privacy where security services are delegated to the infrastructure layer. However, in general, biomedical analyses must satisfy security and privacy properties that are much stronger than those applicable to other domains and have to be enforced in heterogeneous (computational and regulatory) environments [DCD03; PCX11; CD05].

2.3.2. Legal Constraints

Biomedical data and the projects using them are most frequently subject to much stronger legal restrictions, as described in the previous section. As already discussed, some regulations are much stricter with biomedical data than others that consider it just personal data. In addition to standard rules to ensure and preserve data privacy, biomedical-related data must have stricter storage, transfer, and access control due to sensitivity level restrictions.

The ICAN project is limited by French regulations that impose several constraints on biomedical data sharing, even between institutions located inside the country, in addition

to the restrictions defined by the GDPR. French regulations impose stringent constraints on privacy preservation if data is shared with third parties: public cloud infrastructures can only be used in exceptional circumstances and after undergoing specific accreditation procedures. Currently, the ICAN partners are considering an extension to an international partnership across and beyond the European Union. However, for more deeply integrated cooperations, the regulatory situation becomes much more complicated because of the different laws governing data privacy in general and health-related data in different countries. Since governments are aware of the potential benefits of such cooperations, they have made efforts to regulate data privacy without affecting the flexibility of the research initiatives [Fed+15].

2.4. Extending I-CAN to EU and non-EU partners

Collaborative analyses through international cohorts can help collect more information about a disease, in particular aneurysm risks. However, the technical and legal restrictions on moving human data are still more severe. As a continuation of the ICAN project, an extension to EU and non-EU partners is currently in preparation.

With our partners from Nantes University Hospital, we have investigated the following extension scenario, presented in the form of a distributed workflow:

1. In France, medical images are analyzed to select interesting patients,

2. The genetic information of the selected patients is shared with partners with whom data-sharing agreements exist. Otherwise, learning models trained on local data can be shared without sharing raw data.

3. In both cases, the shared information of these relevant genomes could build an international study cohort for aneurysm research.

Figure 2.4 presents such an international cooperation as a distributed workflow analysis, in the presence of data sharing restrictions on biomedical data. This scenario does not stem from a real project but has been defined in terms of realistic constraints that apply to international cooperations and has been validated by our medical partners. The figure shows a workflow involving four sites, three in France, and one in Colombia, where each site disposes of local data, and the computing and storage infrastructures at the different sites are heterogeneous.

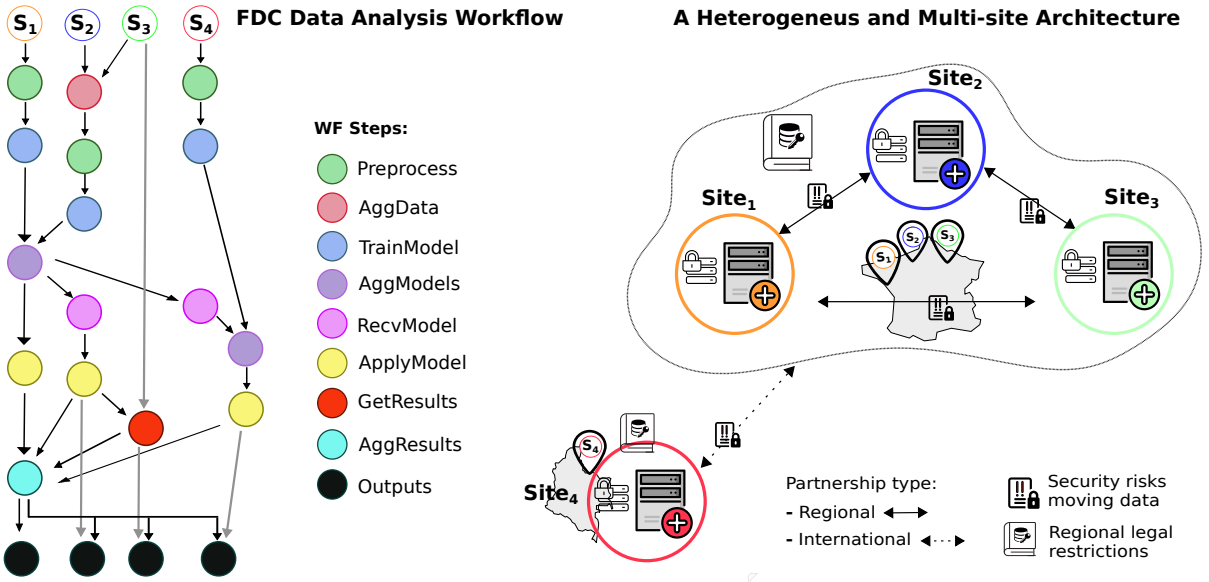


Figure 2.4 – Scenario for distributed processing analysis in Fully Distributed Collaborations across four sites

The analysis considers training models on local data. Local models are shared to build a global model to determine the likelihood of a patient having a disease, such as an aneurysm. The left part in Figure 2.4 shows a distributed multi-site workflow constructed from nine steps, each represented by a different color. It aggregates local models by building a collaborative learning model [Liu+20b]. The analysis is defined as a workflow over computations and data movements. On the right-hand side, the four sites are separated into two groups according to region-specific rules governing the cooperation, such as data-sharing restrictions, policies for data privacy protection, and data ownership. For example, the sites located in France may share data as in the genomes of the patients in the ICAN project. Similarly, European sites may cooperate under standard rules like the GDPR. In this scenario, a collaborative analysis must also be compatible with Colombian data protection law. Colombian law allows the transfer of personal data only to countries with defined data protection standards [Col12], such as members of the European Union.

In addition, each site has different computing and network access capacities. Sharing data on a central site is also inappropriate due to technical restrictions such as computing capacity and bandwidth limitations. For example, the memory capacity required to train a model is notably affected by the size of the data, and the execution time further increases rapidly with increasing data size [Yu+12]. Moreover, transferring raw data versus trained models makes a significant difference in terms of network capacity. For instance, 1.7 TB of

data may lead to a trained (compressed) model of 400 MB [Can+16], which corresponds to a noticeable reduction in the amount of data transferred.

The illustrated scenario avoids sharing raw data among sites, as it was required by the prevailing models for data analysis. On the contrary, the multi-site scenario contemplates training a model that shares only the data of learning models. These trained models use much less space than raw data. Therefore, sharing models reduce data transfer by at least 80% compared to moving raw data (as will be presented in Chapter 5). This remarkable reduction impacts the transfer times and reduces the requirements for the channel's capacity.

Generally, such workflows have to respect numerous constraints, including data ownership, bandwidth limits, infrastructure heterogeneity, and availability, as well as security risks. An FDC-based analysis workflow should facilitate the automation of such complex workflow under the corresponding restrictions. Support for FDCs requires mechanisms to deal with such constraints, and biomedical analysis tools should thus provide them. From these considerations, we propose three architectural key features to support our FDC approach:

- **Data and Computation Placement**. Frequently moving complete datasets between sites is inefficient due to constraints on network capacity and bandwidth costs. Alternatively, data can be processed locally and only partially transferred. Similarly, different sites may share computational resources to optimize computations.

- **Privacy and Security**. Distributed biomedical analyses must ensure strong security and privacy properties, notably confidentiality and integrity properties throughout the complete processes. However, delegating security checks to third parties, *e.g.*, cloud providers, is frequently insufficient because of non-compliance with legal constraints and robust privacy requirements. Third parties provide other mechanisms to address security and confidentiality, such as confidentiality agreements, but these do not grant data privacy and confidentiality when data is moved to third-party facilities.

- **Performance and Scalability**. In multi-site scenarios, computational resources and requirements may differ at each site. Thus, scaling biomedical computations represents a serious challenge. Biomedical analysis tools designers have frequently opted to rely on infrastructure scaling capabilities to address the issue, *e.g.*, elastic computing in the cloud.

We consider these three features fundamental for FDCs. This motivational case will serve to present the thesis contributions. The approach proposed in this thesis is to support biomedical analyses through distributed workflows that share models trained privately from local data.

2.5. Conclusions

Data sharing goes beyond the data sets available for analysis. Effective sharing is achieved by overcoming legal and technical restrictions on biomedical data to analyze it collaboratively. The legal restrictions include regulations that strictly limit medical-source data, and others define rules on personal data in a general way. The reference regulation for many worldwide countries has been the GDPR which defines rules for the treatment and sharing of personal data, including, among others, health data, in particular genetic and biometric data. Meanwhile, research funding entities such as NIH promote policies to share data responsibly, at least data corresponding to self-funded research projects. In the same sense, strategies such as the FAIR principles have been defined to promote more accessible biomedical data for the progress of science. Technical challenges complement legal restrictions due to the size and time required to move large amounts of data to a central site. A real biomedical case where such restrictions apply is the ICAN project which involves the participation of multiple sites, but the analysis occurs in two of them. The project will be favored by collaboration between medical sites interested in understanding the pathology of intracranial aneurysms. The collaboration is focused on collecting knowledge relevant from other patients to make remarkable progress in studying the disease. We propose to analyze data through Fully-Distributed Collaborations (FDCs) mitigating the restrictions on moving and sharing biomedical data. The FDC analyses promise to improve biomedical analysis supported by distributed techniques that favor multi-site collaborative research. The FDC-based analysis will dynamically coordinate the execution of complete workflows exploiting and combining heterogeneous computing facilities.

Chapter 3

Distributed Biomedical Analyses

Biomedical data acquisition has reached surprising volumes in recent years, comparable to astronomical data and social networks. Zachary et al. [Ste+15] estimate that genomic data will require between 2 and 40 exabytes of storage capacity, exceeding astronomical data projections for 2025. This accelerated growth has created new data collection, storage, analysis, and processing challenges. Analyzing and processing such amounts of information poses big challenges to researchers and the infrastructure supporting the experiments. Traditional centralized data analysis scenarios are no longer appropriate due to the constraints discussed in the previous chapter.

In this chapter, we present a taxonomy of existing Biomedical Analytical Tools and Techniques (BATTs) and classify them concerning three different essential criteria for collaborative research focusing on distributed processing:

- First, we study tools and techniques for genomic data analyses, such as SNP identification and gene expression analysis.
- We then investigate how workflow systems support collaborative research regarding workflow specification tasks, interoperability properties, and computational reproducibility[1].
- Finally, we classify BATTs according to three architectural features identified in the motivation section: data and computation placement support, privacy and security properties, as well as scalability and performance properties.

Finally, as a complement to these BATTs, we examine proposals for distributed analyses based on learning models, which have been widely used in recent years. Techniques based on distributed learning emerged in the last decade and could mitigate some of the technical and legal restrictions on collaborative research. However, it is a very active field with diverse challenges that are discussed in this chapter.

1. Computational reproducibility refers to the system's properties to ensure that experiments can be executed repeatedly in different computational environments.

This chapter is structured as follows. Section 3.1 presents a taxonomy focusing on tools and approaches for genomics data analyses. Section 3.2 introduces relevant Machine Learning concepts to understand the collaborative training process. We present, in particular, current learning approaches based on the random forest model in multi-site scenarios. Finally, Section 3.3 concludes the review process's conclusions on distributed biomedical analyses.

3.1. Distributed Biomedical Analyses

This section presents a taxonomy of existing Biomedical Analytical Tools and Techniques (BATTs) for distributed genomic analyses. We have restricted the study of biomedical analysis to genomics due to its importance in biomedical research, the vast amount of unprocessed data that has been generated in recent years, and because we think it is representative enough of the practices of research collaborations in the entire biomedical field. We denote as biomedical analyses those supported by distributed workflows, that is, formulated in terms of the execution of steps between multiple processing sites.

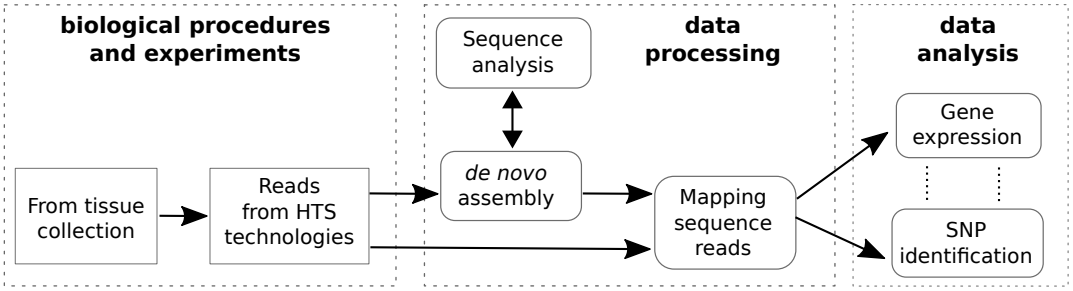


Figure 3.1 – High-Throughput Sequencing (HTS) data analysis protocol (based on [De +12])

3.1.1. Biomedical problems, data analytic techniques, and tools

We classify the data analysis techniques and tools biomedical researchers use to solve common problems when processing genomic data. Such processes can generally be represented by the High-Throughput Sequencing workflow, as shown in Figure 3.1, and consist of three processing steps:

- In the first stage, researchers prepare the tissue samples using biophysical and biochemical methods (for more details on these methods, see [Kim19]). These samples are then sequenced to create libraries of millions of reads sequences that are stored in computer files (see [De +12] for a detailed explanation).
- In the next stage, *data processing*, the reads are assembled to generate a genome sequence. If the studied organism is not associated with any known genome, then "de novo" assembly and *sequence analysis* techniques are used to search links with some known species and better approximate the studied sequence. In contrast, if a reference genome is available, the *mapping sequence reads* techniques are used to generate the sequence.
- Finally, the researchers have enough organized data to test biomedical hypotheses through different genome data analyses in the third stage. For example, researchers may study the relation of specific genes with particular diseases employing *Single Nucleotide Polymorphism (SNP) identification* or *Gene Expression Analysis*.

Table 3.1 – Characterization of BATTs for genomic data analyses

Problem	Analytic Technique	BATT
Gene expression analysis	Statistical methods	ExAtlas [SSK15] Myrna [LHL10] Sparkhit [HKS18] SEQSpark [Zha+17]
	Gene data sets	YunBe [Zha+12a]
	Clustering methods	VariantSpark [OBr+15] Sparkhit
SNP identification	Haplotype blocks	CloudTSS [Hun+11]
	Bayesian approaches	SOAPsnp [Li+09] Crossbow [Lan+09]
	Sequencing methods	Crossbow GATK [McK+10]
	p-value test	BlueSNP [HTP13]

Gene expression is the process by which the instructions encoded in a gene are converted into a functional product, such as a protein. Gene expression analysis is widely used in modern biology to investigate the transcriptional behavior of systems and classify diseased cells [Lov+12]. Microarray and RNA sequencing (RNA-Seq) are the most widely used transcriptome methods. The microarray method is based on DNA hybridization rules on a solid-based platform (i.e., a glass slide), and then the microarray is scanned to measure the expression of each gene. The RNA-Seq methods have further advantages due to not being limited by the printed material on the microarray. Furthermore, RNA-Seq methods are becoming more popular due to the remarkable development of Next-Generation Sequencing technologies (an extensive discussion on transcriptomic technologies is presented in [Zho+16]).

Nowadays, there are different analytic techniques for gene expression analysis, some of them are classified in Table 3.1, but an extensive review of these is presented in [Gar+22]. These techniques must evolve for collaborative research to support processing such voluminous data and promote cooperation among geo-distributed research groups. Regarding the computational architecture supported by the reviewed BATTs, they consider analyzing data using sequential or parallel cluster-based strategies. These computational features are discussed below, and the shortcomings reveal opportunities to improve processing strategies, especially by distributing data and computations, notably as part of FDC scenarios.

3.1.2. Support for Research Collaborations

Nowadays, global multi-party collaborations are crucial in multiple biomedical domains, such as cancer treatment [Kuh+07]. Such collaborations require large geo-distributed analyses to be applied to huge amounts of distributed data. Furthermore, they should be reproducible to support independent validation by the biomedical community. In order to support such collaborations, BATTs must provide mechanisms and abstractions for the application of algorithms in distributed data and support the participation of independent and partially competing researchers. Therefore, we argue that the appropriate tools must possess at least the following three corresponding characteristics:

- They must provide a *workflow description language*, providing a common interaction language for users with different backgrounds.

- They must provide explicit *computational* abilities means to *reproduce* experiments. Thus, the definition must include enough information for other teams to replicate the experiment based on computational settings.
- The tools must provide *interoperability* mechanisms such as Application Programming Interfaces (APIs).

In the following we investigate how Scientific Workflow Management Systems (SWFMSs), popular tools in biomedical research, meet these research collaboration characteristics.

3.1.2.1. Workflow Description Language

The most common way of defining data-driven biomedical analyses is using workflow description languages (WDLs). A workflow definition uses atomic tasks that perform specific calculations on available data [Zha+12b; Atk+17] and the execution order of (atomic and composed) tasks. The tasks may be executed sequentially or concurrently depending on the definition of the WDL, the orchestration components available and the infrastructure capabilities. The workflow is represented using a directed graph. The nodes represent the tasks, and the edges represent the control flow. Biomedical analyses also require the definition of data flows between tasks. Most data flow definitions are implicit, that is, atomic tasks assume that their input data is ready for processing via the corresponding data sources when their tasks are invoked. The input data for a specific task may be prepared by the previous task. This is, however, not always the case: in general, the workflow (control flow) edges represent data flows (or better data dependencies) only implicitly.

The workflow specification form depends on each system and can be done in two ways: by a graphical interface or textual. The specification form is relevant since end-users usually have limited technical capabilities, making it more accessible through graphic means. For example, some workflow description languages provide a graphical user interface (see Galaxy [Goe+10] and Kepler [Bar+10]), while others, such as Snakemake [KR12] and Nextflow [Di +17], use scripting languages.

Table 3.2 shows the most important workflow systems, classified with respect to the type of graph supported for the definition of the workflows, their user interface, and the specification language used or generated by the tool. The column *Workflow graph* indicates if the tool supports the definition of workflows using Directed acyclic graphs (DAG) or Directed Cyclic Graphs (DCG). The user interface column indicates how the workflow

Table 3.2 – Workflow description characteristics of the SWFMSs

SWFM	Workflow Graph	User Interface	Spec. Language
Galaxy	cyclic graph	graphic	N/A
Kepler	cyclic graph	graphic	XML-based
Knime	acyclic graph	graphic	N/A
Nextflow	acyclic graph	textual	Groovy-based
Pegasus	acyclic graph	textual	XML-based
Swift	cyclic graph	textual	Objective-C-based
Taverna	acyclic graph	both	SCUFL-based
Triana	cyclic graph	graphic	XML-based
Snakemake	acyclic graph	textual	Python-based
Wings	acyclic graph	graphic	OWL-based

definition is specified via a textual interface or a graphical one to represent the tasks and the control flow. Finally, the *Spec. Language* column indicates the language used for the specification or the language generated by the graphical tool to specify and store the workflow. For example, Snakemake provides a scripting language that is built on Python. Taverna [Wol+13] supports data flows expressed in terms of the Simple Conceptual Unified Flow Language (SCUFL), an XML-based language. SCUFL provides three main abstractions: processors, data links, and restrictions applied during the workflow execution. The Nextflow [Di +17] specification is based on the Groovy programming language, and its syntax thus is Java-compatible. Finally, Wings [Gil+10] uses the Web Ontology Language (OWL) to model tasks and workflow constraints. OWL was proposed by the World Wide Web Consortium (W3C) and is a computational logic-based language for representing and sharing computational ontologies.

3.1.2.2. Experiment Reproducibility

Experiment reproducibility is an important feature in any research field. It is essential for global research collaborations studying biomedical questions. However, designing reproducible experiments with biological data has proven to be challenging. Some researchers even claim that life sciences are experiencing a reproducibility crisis [Fry+15]. The non-reproducibility of experiments nowadays may lead to the rejection of otherwise sound research papers. For example, in 2007, researchers from the M.D. Anderson Cancer Center[2] found inconsistencies due to data handling errors in an acclaimed paper on cancer

2. https://www.mdanderson.org/

treatment from the previous year [Hut10].

Since then, the research community has put a lot of pressure on researchers to ensure that their experiments and results are reproducible [HG13a]. Funding agencies, in particular, have established policies to promote reproducibility, such as NIH guidance for addressing rigor and reproducibility in their funded projects [Hea+21]. Similarly, the European Union's Horizon 2020 program has implemented measures to make research data accessible for reproduction by others.

Executing analyses in distributed environments increases difficulties for the reproducibility of experiments; for instance, distributed systems consist of many components working together, and in those systems cause challenging problems such as heterogeneity, scale, and instability [WCW08; Wan06]. In the case of FDC-based analyses, reproducibility is particularly challenging because multiple sites host executions, each potentially with different, computing capacity, dynamically-changing topologies, security, and privacy parameters. These technical challenges limit the capacity of current tools to support the reproducibility of geo-distributed analyses.

This section investigates the level of reproducibility provided by the workflow systems that have been considered above. We start from two different reproducibility perspectives advocated by various research initiatives, namely, *computational reproducibility* and *experiment reproducibility*. We then then describe the approaches and present a taxonomy to classify and understand the current state of the tools. We discuss *computational reproducibility* putting a focus on three main relevant features: data provenance management, support for the export of workflow components for reuse in another systems, and the ability to document or modify analyses phases by means of annotations.

Several papers have evaluated the computational reproducibility in scientific workflows [Coh+17; SP15; GFI16; FBS12; LZ11]. These studies have concluded that for the reproduction of data analysis experiments, researchers need at least three things: the workflow description, the data, and the description of the technical configuration of the experiment. The findings seem natural and sound; however, we argue that the last requirement, *technical configuration of the experiment*, needs some discussion. In particular, it is essential to note that the information of "technical configuration" is required for performance comparisons and for the detection of possible sources of errors. However, the reproduction of the analysis must be independent from the hardware and middleware configuration. Workflows describe a computation performed over input data, and are, in principle, independent from the hardware and middleware on which they are executed.

Thus, it is important to differentiate the configuration of the underlying hardware and middleware (for example, the operating system) from that of the active components of the analysis, for example, a deep learning engine.

Other researchers have studied reproducibility from a more abstract perspective, emphasizing experiment (analysis) reproducibility [Dee+09]. These researchers advocate for a more general set of requirements for reproducibility: data provenance information, workflow exchange mechanisms, and workflow annotations. Data provenance information documents where data came from and what transformations it has suffered. The exchange mechanisms describe the standards and methods supported to define workflows, and workflow annotations provide information about the execution of the scientific analysis. We think that this last perspective is more suitable for the study of experiment reproducibility over FDCs. In particular, because it addresses the problem from the perspective of knowledge transfer and abstract computations (workflows). In other words, it allows for a suitably high-level description of the design and the execution of the scientific analyses. Interestingly, these issues are similar to those commonly investigated in the software development community, where design documentation and algorithm definition are crucial for reproducibility. For instance, design documentation via annotations in the workflow, metadata to track transformation in scientific data sets, and standard platforms for "code/experiment" sharing have already been studied by the software engineering community [Dee+09; Kar+18; Coh+17]. Industry standards and corresponding platforms, such as *myExperiment*, *BioSharing*, and *WorkflowHub*, improve communication among researchers and facilitate exchanging knowledge.

Table 3.3 – Computational reproducibility characteristics in the SWFMSs

SWFM	Data Provenance	WF Exchange	Annotations
Galaxy	Yes	GA(v19), Format2 (>v19)	Yes
Kepler	Yes	KAR	Yes
Knime	No	KNWF	No
Nextflow	No	NF	No
Pegasus	No	DAX(v4.0) and YAML(v5.0)	No
Swift	Yes	Swift	Yes
Taverna	Yes	SCUFL	Yes
Triana	No	XML	No
Snakemake	No	Snakefiles	No
Wings	Yes	RDF/SWRL	Yes

Table 3.3 presents the three features that characterize the support for experiment reproducibility in the SWFMSs. The first column indicates if the system provides some data provenance mechanism. SWFMSs labeled with 'Yes' comply in some way with practices and recommendations, like those proposed by World Wide Web Consortium (W3C) [Mor+13], or they provide some proprietary provenance strategy. For example, Taverna represents metadata at the dataflow level based on the PROV[3] language defined by W3C. However, the information stored does not provide details of intermediate data during the workflow analysis [SAG16]. On the other hand, Kepler records all details during the workflow execution, including data evolution and step details. Provenance features also enable researchers to debug transformations, *e.g.*, to search for errors. However, there are still many open challenges regarding data provenance; for example, incomplete data provenance records limit the reproducibility of workflows [Kha+19].

The column *"WF Exchange"* shows the supported formats for exchanging workflow definitions or their components. Pegasus abstractly describes the workflow using a DAX format (Directed Acyclic Graph in XML) up to version 4.0, and the following versions use a serialization format YAML. Galaxy is based on a format that is not readable and a more recent, experimental one. In Taverna, the workflow and its components are represented using the SCUFL language based on the XML format. Kepler exports the workflow components to the Kepler Archive (KAR) file based on the JAR file format from Java. Knime

3. See https://www.w3.org/TR/prov-overview/ for information on PROV.

exports to format KNWF; it is a KNIME Workflow data based on an XML/JSON format. Nextflow has its own format that represents the specification of the workflow based on the Groovy programming language. Swift, through the Swift language, represents the workflows in a format based on C-like syntax. Triana defines the workflows using files based on XML representing the name, specifications, and parameters. Snakemake uses Snakefiles, Python-based rules including input and output data between them. Wings expresses the workflow components and dependencies in workflow templates using the Resource Description Framework (RDF) and the Semantic Web Rule Language (SWRL). Finally, the table shows if a platform supports annotations in the workflow. Annotations help scientists to understand the design of the experiment. For example, Taverna and Galaxy provide means to annotate the workflow using free text or labels, in contrast to Nextflow, which does not offer an annotation mechanism.

The table shows that support for experiment reproducibility is still limited in the current workflow systems. In fact, in 2017, Kanwal et al. studied reproducibility and provenance tracking in workflows for genomic analyses and pointed out that there is still an incomplete understanding of reproducibility requirements, and thus a lack of support on current tools. They also remarked the complexity of reproducibility of distributed workflows as an open research problem. Similarly, other researchers have identified *workflow decay* [De +11; Zha+12b] due to volatile third-party resources, missing example data, missing information about execution environments, and insufficient descriptions of workflows as a complication for experiment reproducibility.

3.1.2.3. Workflow System's Interoperability

We define interoperability as the ability of workflow systems to communicate with other systems to exchange data, share functionality, or delegate responsibility. SWFMSs may interoperate by providing language bindings or by exposing an API at runtime. Programming language bindings allow scientists to use their preferred programming languages to create complex analytical routines to extend, enhance, or leverage native functionality. On the other hand, an exposed API helps scientists to interconnect several analytical engines to delegate responsibilities to specialized hardware and software. For example, a scientific workflow may execute most of its computations in local infrastructure and delegate a specialized machine learning computation to a GPU cluster deployed in the cloud.

In this section, we investigate interoperability mechanisms by identifying each tool's

different programming language bindings and by studying the presence and reach of an API exposed to be invoked at runtime. In particular, we have studied the presence of REST APIs and the exposed functionality. Some tools provide APIs to support execution monitoring, while others allow full control of the workflow life cycle (definition, debugging, deployment, and execution). During the discussion, we use the term API to refer mostly to the programming interface provided via libraries for specific languages (language bindings). REST API refers to the functionality exposed at runtime using REST web services.

Table 3.4 – Interoperability properties in the SWFMSs

SWFM	Language Binding	REST API capabilities
Galaxy	Python, PHP, Java, JavaScript	Low-level: interact with data, runtime tools and WFs, handling histories
Kepler	Java, R	No
Knime	Python scripts via UI	Interaction by UI for Extract, Transform, and Load (ETL), and score a model for analysis prediction
Nextflow	No	No
Pegasus	Python, Java, R	Monitoring, defining and executing workflows.
Swift	No	No
Taverna	Java scripts via UI	Monitoring and executing workflows
Triana	Java scripts via UI	No
Snakemake	Python	No
Wings	Java, Matlab, and Python scripts via UI	No

Table 3.4 presents the classification of SWFMSs with respect to their interoperability mechanisms. The second column, *Language Binding*, lists the programming languages that are supported by each tool. A system with more language bindings is supposed to provide better interoperability properties. For example, Galaxy provides Python, PHP, Java, and

JavaScript libraries, allowing researchers to define their analytical experiments in their preferred language. Knime, on the other hand, supports the inclusion of Python scripts as custom code in the workflow steps defined in their graphical user interface but does not provide language bindings to use the tool from a programming language. Similarly, Taverna and Triana enable customizing Java code through the graphical interface during the workflow design. In contrast, Nextflow and Swift do not provide libraries to extend to programming languages, and they use their proprietary specification language to define workflows.

The third column, *REST API*, provides information on interoperability with external sources through REST services. Galaxy's API supports the interaction with external programs or libraries at runtime. The Galaxy API allows interaction with data sets, executing and monitoring workflows, and monitoring relevant information. Secure communication is achieved by supporting the HTTPS protocol. Galaxy is migrating the current version of the API to an improved version supporting the standard of FastAPI[4]. Knime provides interoperability mechanisms, although limitedly, through the licensed component Knime Server. It enables calling web services as part of the workflow design by means of a graphical interface. Knime performs ETL (Extract, Transform, Load) operations via REST services and integrates with external visualization tools. Finally, Pegasus and Taverna provide functionalities through REST web services to support monitoring, design, and specification of workflows.

The table shows that interoperability mechanisms are limited in current workflow systems. Other researchers have identified these shortcomings; see the research presented in [EHT10; AIG12; Sil+17]. Sarah et al. [Coh+17] also identified interoperability between workflow systems as an open challenge, suggesting a standard intermediate model between the specification and execution layers to interoperate between different workflow systems.

3.1.3. Distributed Architectural Features

We now investigate the support of BATTs for the architectural features required to address fully distributed collaborations. Concretely, we will investigate how the current tools address *data and computation placement*, *privacy and security*, and *performance and scalability*.

4. https://fastapi.tiangolo.com/

3.1.3.1. Data and Computation Placement

Analyzing large amounts of data scattered over several sites requires transferring the data to a single site for analysis, delegating computations to the sites where data is stored, or a hybrid solution that partially distributes storage and computations. The choice of the strategy to use depends entirely on the specific context of the collaboration.

Table 3.5 – A categorization of BATTs and SWFMSs according to our architectural features proposed

Name	Paradigm	D/C Placement		Privacy and Security	Perform & Scalability	
		Data	Computation		Architecture	Abstraction
Taverna	Workflow	loc+, ext	dynamic	+	distribut	explicit
Galaxy	Workflow	loc+, ext	dynamic	+	distribut	explicit
Kepler	Workflow	loc++	dynamic	++	distribut	explicit
Knime	Workflow	local	static	delegated	distribut	explicit
Nextflow	Workflow	loc+, ext	dynamic	delegated	distribut	delegated
Pegasus	Workflow	loc+, ext	static	++	distribut	delegated
Swift	Workflow	loc++, ext	dynamic	delegated	distribut	delegated
Triana	Workflow	loc++	dynamic	delegated	distribut	delegated
Snakemake	Workflow	loc+, ext	dynamic	delegated	distribut	delegated
Wings	Workflow	loc+	static	delegated	distribut	minimal
Sparkhit	MapReduce	loc++	dynamic	delegated	PCluster	explicit
Crossbow	MapReduce	loc+	static	delegated	PCluster	delegated
MetaSpark	MapReduce	loc++	static	delegated	PCluster	explicit
CloudBurst	MapReduce	loc+	static	delegated	PCluster	explicit
Halvade	MapReduce	loc+	static	delegated	PCluster	delegated
DistMap	MapReduce	loc+	static	delegated	PCluster	delegated
Myrna	MapReduce	loc+	static	delegated	PCluster	delegated
SparkBLAST	MapReduce	loc++	static	delegated	PCluster	delegated
K-mulus	MapReduce	loc	static	delegated	PCluster	delegated
CloudSW	MapReduce	loc++	static	delegated	PCluster	explicit
SOAPsnp	Sequential	loc	local	N/A	standalone	N/A
S–MART	Sequential	loc	local	N/A	standalone	N/A
CAFE	Sequential	loc	local	N/A	standalone	N/A

The first column refers to a platform supported based on MapReduce or workflow system. Data placement covers three strategies: only local data, allocation by a simple partition strategy (+), and dynamic allocation of data (++). Computing placement allocates resources to process data before execution (static) or reallocate during execution (dynamic). Privacy and Security cover three mechanisms: a simple (delegated in the cloud or locally), complement delegate security with own functionalities (+), and advanced strategies (++). Finally, the architecture implemented and the abstraction level supported by each one, explicit or delegated.

The problem of allocating storage, bandwidth, and computational resources to optimize the performance of a specific scientific workflow has been amply studied generally for cluster and grid computing settings. A common strategy is to address the problem of data placement in order to minimize total data transfer cost and optimize the execution time (computation time) in a given architecture, as proposed, for instance, by Van Huang and Chuanhe [VC11] as well as by Cope *et al.* [Cop+09]. In contrast, Li *et al.* [Li+16] claim that such cost minimization strategies are insufficient and propose a two-stage approach, first preallocating datasets to specific data centers during workflow build-time, and then dynamically distributing newly generated datasets at runtime. Other authors have proposed other heuristics to address the placement problem [Ebr+15; Zha+16; Che+21; Xie08]. However, finding the *optimal* data placement strategy is an NP-hard problem [LD11]. Studies on computation placement are rare: very few approaches address the problem of selecting the best computation facility for a given task, and most rely on cloud environments guaranteeing the availability of (almost) arbitrary computational resources required by collaborative scenarios. Similarly, studies relying on MapReduce and workflow systems lack the functionality necessary for multi-site processing [Dol+17b; PCA15]. Therefore, current approaches are not appropriate for FDCs for biomedical analyses for two reasons:

- They do not support geo-distributed architectures that constitute a much more complex setting for computation and data placement [PCA15; Liu+18; Pin+16; Liu+17].
- They do not consider the larger set of constraints of biomedical applications that involve legal and socio-economic constraints.

This problem is manifest because most BATTs harness partitioning data simply by distributing equal chunks among multiple nodes. Data is often loaded onto a single node and then split into chunks stored and distributed on a cluster of nodes using randomization strategies [Xie+10]. Table 3.5 starts by classifying the BATTs according to the paradigm and data location type offered. For example, Halvade or BioPig use such strategies based on Hadoop. Others use different frameworks, notably Spark, but use similar data and computation placement strategies. Sparkhit [HKS18] can process data from different clusters located in three geographic regions, but the data needs to be moved to one place. MetaSpark [Zho+17], SparkBlast [Cas+17], VariantSpark [OBr+15], and SparkSW [ZLS15] are also based on Spark architecture. By contrast, a few BATTs, like S-MART [ZQ11] or CAFE [Lu+17], harness sequential analysis without any distributed process-

ing. Therefore, all these BATTs lack the functionality to support fully distributed and collaborative work, especially to process data across different geographic sites.

Other tools as workflow systems have more explicit data and computation placement features. For instance, Pegasus [Dee+15] supports three data placement approaches, shared file systems, remote ones and non-shared ones. In addition, Pegasus uses DAGMan [Tan+01] and HTCondor [TTL05] to model task-based workflows submitted to a pool of resources in HPC clusters. Kepler [Bar+10] includes prebuilt components (referred to as actors) to model external data sources and grid facilities. Finally, Taverna uses the SCUFL language to explicitly define the data flow between processors to model the passing of data between services associated with bioinformatic atomic tasks. Pegasus and Triana implement the strategy "replica location service" [Yua+10] that allows access to information about copies in different physical locations to support scalability, reliability, and security during distributed executions. The same approach is used by the Giggle (GIGa-scale Global Location Engine) framework [Che+02].

The second column of Table 3.5 shows the classification of BATTs according to their data location and computation strategies. The data placement column indicates if the corresponding BATT supports local data processing (label `loc`). If, for example, the local data placement strategy may distribute data in a cluster, such as in Hadoop, we add a plus symbol (`+`). If the tool supports even more sophisticated methods, such as dynamic allocation offered by Spark, we add a doubled plus symbol (label `++`). Finally, the label `ext` shows if the BATT can handle external data sources located, for example, in cloud repositories like Amazon's S3 cloud. The column *Placement - Computing* describes how computing resources are allocated. If the tool allocates computing resources at definition or configuration time, we classify it as employing `static` allocation. If the allocation or reallocation of resources is done at execution time, we classify the tool as having `dynamic` allocation. The `dynamic` and `static` labels are related to each workflow system's workflow scheduling strategy. For example, Swift, Galaxy, and Triana systems exploit different dynamic scheduling strategies based on publish/subscribe patterns or adaptive methods during workflow execution [Liu+15]. In contrast, systems like Pegasus provide mechanisms resulting in static scheduling [BL13], without automated strategies to adapt computing resources during execution across multiple nodes.

3.1.3.2. Privacy and security

Distributed biomedical analyses are subject to many security risks and, frequently, to a much higher risk of privacy issues than other domains since the potential loss of personal and sensitive information implies more severe consequences. Security and privacy-related properties that have to be satisfied comprise, but are not limited to, authentication, authorization, integrity during access and control of biomedical data. BATTs, therefore, have to provide means for the stringent enforcement of security and privacy-preservation properties, or at least be able to harness corresponding means that are provided by their environments.

In Table 3.5 (on page 57), we classify BATTs according to the security mechanisms provided or delegated (either in the cloud or locally). If a BATT provides simple security mechanisms, we mark it with a single plus symbol (`+`). Taverna, for example, provides simple security mechanisms, such as authentication based on web services. Similarly, Galaxy offers limited capabilities such as libraries to make a secure connection through a web API. If the tool provides a more advanced security mechanism, we mark it with two-plus symbols (`++`). For example, some BATTs provide explicit mechanisms to secure data during storage, transfer, and processing. For instance, in Kepler, the Security Analysis Package (SAP) provides information security mechanisms such as input validation, data integrity, and remote access validation. The package triggers an alarm when a potential alteration of the information is detected [KV16]. However, some researchers have pointed out that the SAP package is not yet part of the current release of Kepler due to its runtime overhead [Ryn+19]. Similarly, in Pegasus, the Scientific Workflow Integrity Project[5] (SWIP), proposed by NSF, seeks to ensure integrity and security during workflow execution. The SWIP project implements some cryptographic mechanisms to check provenance metadata to detect input and output data changes during the workflow execution.

Many BATTs delegate security to the hosting (cloud or local) infrastructure provider; this is indicated in the table by `delegated`. For example, Snakemake, Knime, Sparkhit, and MetaSpark do not offer mechanisms to analyze data securely. The security aspects are delegated to the infrastructure provider. In contrast, Taverna and Galaxy endow the API with authentication strategies through REST services, although the premise is that the processing nodes are part of trusted environments.

Overall, the table shows that only a few BATTs provide specialized advanced security and privacy mechanisms. Moreover, security and privacy issues have also to be handled

5. https://cacr.iu.edu/projects/swip/index.html

at other levels, such as storage, sharing, processing, and publication of results. The first BATTs were designed to work with distributed file systems, such as GPFS [SH02] or PVFS [RT+00], and had to provide simple interfaces to high-performance environments implemented on top of grids [Liu+14]. In such scenarios, information security and privacy were often ensured by not sharing data and strongly limiting access to research facilities. Later on, the exponential growth of biomedical data and the emergence of accessible cloud infrastructures motivated BATT systems to redesign their architecture. However, most of them tried to achieve this by minimal changes to adopt the cloud as a preferred execution environment and delegating security to the cloud providers. For example, tools implementing a MapReduce strategy, such as CloudBurst or MetaSpark, delegate security features to the infrastructure provider. In addition, privacy and security are major concerns of its own in cloud environments. There are still many open challenges to comprehensive security during FDC analyzes involving shared computing facilities and data storage. Some initiatives, though limited, address security with a higher priority, such as the SWIP project. However, these challenges are most frequently present because of a lack of support for security concerns in most workflow systems [Kar+18; NV08].

3.1.3.3. Architecture and Quality Attributes

Flexible and robust architectures are needed to support FDC scenarios. In this section, we study the architectural features of BATTs, focusing on two distinctive characteristics, namely the support for distributed architectures and the mechanisms provided to address complex quality attributes such as performance and scalability. Concretely, we classify them first according to their execution architecture, identifying those that support standalone execution, distributed execution on clusters of computers, and distribution over several geographically separated sites.

We then study how those tools support complex quality attributes, identifying if they provide explicit mechanisms to address them or if they delegate the final configuration to the deployment phase, where engineers rely on the capabilities of the underlying infrastructure. Here, we are interested in the explicit support for distribution from within the tool. For example, as part of a complex workflow, a complex algorithm may be configured at deployment time to be executed in high-performance computing services on Amazon Web Services (AWS). However, we are interested in tools providing explicit means to model and manipulate such high-performance computation facilities.

Table 3.5 presents our findings. First, the column Architecture shows if the tools

are stand-alone, run on a parallel cluster, or support complex distributed architectures. By complex distributed architectures, we mean complex workflows deployed on top of the complex geo-distributed infrastructure. The next column shows if complex quality attributes are supported explicitly by abstractions provided by the tool or if they are delegated to the deployment phase and the underlying computing infrastructure. The remainder of this section provides additional insights on this classification.

Stand-alone Solutions on a Single Machine

Stand-alone applications are used to process small datasets on a single machine. For example, CAFE implements alignment-free sequence analysis on single-machine architectures. Decades ago, DNA sequences were aligned using algorithms based on dynamic programming, but this strategy turned out to be inefficient over time due to the length of the investigated sequences. More generally, due to a large amount of data to be analyzed in many cases, stand-alone applications are limited. Therefore, implementations based on parallel approaches such as MPI techniques, GPU computing, or MapReduce frameworks are preferred nowadays.

Table 3.6 – Sequential and cluster-based BATTs for genomic data (complements Table 3.1)

Architecture	BATTs
Sequential	ExAtlas
Cluster-based	VariantSpark, CloudTSS, BlueSNP, Myrna, Sparkhit, SEQSpark, Crossbow, GATK

Table 3.1 is complemented by Table 3.6, where we present the main processing architecture differentiating the BATTs that use sequential computations and those that support processing on parallel clusters.

Cluster-based Distribution

Most current BATTs are implemented on top of popular frameworks such as Hadoop and Spark. These tools are extensively used for cluster-based processing of large data sets. For instance, Halvade executes pipelines parallelly on a multi-node architecture or in a multi-core configuration using Spark on a cluster and storing data in files using the Hadoop Distributed File System (HDFS). Similarly, DistMap and Myrna use Hadoop to

execute statistical models on multiple processors in cluster scenarios. Several other tools follow similar approaches providing parallel execution over a computing cluster, including CloudTSS, SEQSpark, VariantSpark, CloudBLAST, CloudSW, or CloudBurst.

The MapReduce paradigm has also been integrated into workflow systems. For example, Kepler over Hadoop provides an architecture that supports the execution of MapReduce applications during the workflow execution in Kepler [WCA09]. Similarly, Hi-WAY executes scientific workflows using Hadoop YARN [Bux+17]. Other SWFMSs employ naive ad hoc strategies to parallelize tasks and distribute data over available resources [BL13]. These strategies are specific for each system and are usually provided by the task management layer of each workflow system. The corresponding BATTs that we have surveyed are categorized in Table 3.6 as *cluster-based architecture*.

Distribution of Computations over Several Sites

One may argue that some of the tools harnessing parallel infrastructures can support fully distributed collaborations by exploiting, in addition, direct manipulation of the underlying distributed infrastructures such as those based on MapReduce. However, we have identified several limitations as part of our analysis. Most importantly, the tools rely on the configuration at deployment time to address complex collaborations, resulting in a complex deployment phase that restricts flexibility and reproducibility. For example, Taverna provides components supporting remote execution of locally defined workflows that are published and controlled by web services. Taverna can be executed on clusters, grids, and clouds, and it can be made to interoperate with other workflows like Galaxy. However, those configurations require extensive technical knowledge by the deployer and flexible features from the underlying infrastructure.

Kepler supports the use of programs written in R or C for remote execution, harnessing distributed execution threads via web and grid services. In the same way, Nextflow [Di +17] and Snakemake [KR12] support GRID platforms, *e.g.*, SGE (Sun Grid Engine) or LSF (Load Sharing Facility). In addition, Snakemake supports the interaction with other tools via web services executing jobs in distributed environments, such as clusters or batch systems. But again, most of these tools rely on the configuration knowledge of the deployer. Pegasus has taken a more explicit approach for distribution and supports execution on individual machines, remote clusters, distributed infrastructures, and clouds. Nevertheless, Pegasus does support such architectures with explicit abstractions or components. For example, it incorporates HTCondor to enable the management of resources in dedicated

or distributed computers. Also, Pegasus incorporates the Glideins component that allows adding machines from different domains and HPC centers.

Researchers have noted that BATTs must adapt their architectures for processing in multi-site scenarios, such as multi-cloud technologies. Nevertheless, it is necessary to have an orchestrator between the workflow layers and the cloud architectures. This way, the technical challenges of distributed computing, such as resource allocation, virtualized systems, fault tolerance, and task monitoring, can be supported [Zha+15c; Sen+18].

3.1.4. Distributed Workflow Systems

Table 3.7 – Classification of SWFMSs according to the level of distribution offered.

SWFM	WF Scheduling	Data Management	WF Partitioning	Distribution Level
Taverna	centralized	centralized	Yes	Partial
Galaxy	centralized	centralized	Yes	Partial
Kepler	centralized	all		Partial
Knime	centralized	centralized	Yes	Limited
Nextflow	centralized	centralized	Yes	Partial
Pegasus	centralized	mediated	Yes	Limited
Swift	decentralized	all		Fully
Triana	decentralized	peer-to-peer		Fully
Snakemake	centralized	centralized	Yes	Partial
Wings	centralized	mediated	No	Partial

The first column represents the workflow scheduling architecture: central workflow (centralized) or supported by multiple schedulers (distributed). Data management indicates the strategies during data processing, moving it to a centralized site, mediated by a distributed data management system, and transferring data point-to-point in a P2P fashion. The partitioning process generates workflow fragments where each is defined to be executed on a specific site or time. The last column, Distribution Level distinguishes between three possible options: Limited, Partial, or Fully Distributed.

In this section, we study the use of workflow systems for distributed collaborative scenarios. Table 3.7 presents additional information about SWFMSs, classifying them by workflow scheduling approach, data management strategy, support of workflow partitioning, and fully distributed collaboration.

Workflow scheduling strategies. Workflow scheduling strategies allocate tasks to computational resources such as processing nodes during workflow execution [Liu+15]. Yu and Buyya [YB05] present two scheduling strategies for workflow systems. In contrast

to centralized scheduling architectures, decentralized ones enable multiple schedulers to manage tasks. They highlight the importance of scheduling schemes to achieve scalability and performance in workflow systems. They survey Galaxy, Nextflow, Swift, and Triana as examples of decentralized architectures. For instance, Galaxy implements the GridWay framework to provide multiple schedulers, and Swift integrates the Karajan workflow engine [Wil+11]. In [Liu+15], centralized architectures are identified as having bottlenecks in the master node, and peer-to-peer (P2P) systems are proposed to mitigate this problem. The first column in Table 3.7 shows the systems' workflow scheduling strategies of the surveyed SWFMSs

Data Management. Another critical feature for FDC support is the data management strategy. Yu *et al.* [YB05] study three mechanisms: centralized, mediated, and P2P-based data management. In a centralized data management strategy, a master node manages the data; mediated strategy, the underlying system is responsible for data management; a P2P-based scenario, data is distributed among all available nodes without an intermediary. Some SWFMSs offer several strategies, for example, Kepler. Triana implements a decentralized architecture applying two distribution policies for parallel and pipeline execution [Tay+07a]. The data management strategies are summarized in the second column of Table 3.7.

Workflow Partitioning. Clustering is a technique using several tasks to partition a workflow horizontally. This technique may improve the performance in distributed scenarios. For instance, Pegasus [Dee+15] implements different clustering techniques that improve the execution time significantly for short tasks. Following Liu *et al.* [Liu+15], the partitioning process generates workflow fragments where each one is programmed to be executed on a specific site.

Discussion

The taxonomy presented in this section concludes with the classification of the workflow systems with respect to their support for fully-distributed collaborations. As discussed during the section, existing tools, and approaches for distributed biomedical analysis limit functionality that supports the three architectural features for analyzing data in an FDC setting: data and computation placement, privacy and security, and performance and scalability. Additionally, the functionality provided by workflow systems is also limited around

interoperability and reproducibility. Therefore, we identified an opportunity in designing systems to analyze data in multi-site environments, such as FDC scenarios, due to the limited support of current ones in collaborative research and geo-distributed processing.

As discussed in this section, existing tools and approaches for distributed biomedical analysis limit the functionality of the three main features for FDC analyses: data and computation placement, privacy and security, as well as performance and scalability. Additionally, the functionality provided by workflow systems is also limited with respect to interoperability and reproducibility. The last column in Table 3.7 classifies current workflow systems in three categories according to the functionality provided by each one focused on FDC features: limited, partial, and fully distributed. Finally, we have identified opportunities in analyzing biomedical data across multiple sites, such as FDC-based analyses. Current biomedical tools and workflow systems must redesign their architecture and processing strategies adopting multi-site distribution strategies due to limitations in centralized processing. This is reaffirmed by the classification presented throughout the section.

3.2. Machine Learning-Based Analysis

Artificial Intelligence (AI) techniques, which emerged in the 50s, recently have been much developed as well as proven popular and useful in many domains. In the medical field, AI systems have emerged, starting from expert systems. A generation of rule-based expert systems assisted and supported medical diagnoses [Mil94; SPS88; Sho12]. The rule-based systems were a great success for the problems of the time, such as diagnosing diseases and defining treatments, leading in many cases to hypotheses to be explored through clinical and medical research. However, the heterogeneity in the data collected over time revealed shortcomings with the rule-based systems of the time. Today, machine Learning algorithms aim to provide the machine with the power of inference from data via AI techniques. ML algorithms are already frequently used in computer vision, speech and image recognition, natural language processing, prediction, and recommender systems.

In the biomedical domain, machine learning has recently also become popular, mainly due to the widespread application of machine learning models. The popularity stems from recent advances in computing technologies to scale computing resources to train models on large-scale sets of now available biomedical data. ML techniques have been applied to diagnose and prognose of different diseases using various biomedical data sources, such as

genomes, images, clinical records [Ver+20; LN15; YBK18].

Over the years, technological progress and new biomedical problems motivated the development of learning techniques to alleviate the challenges faced by heterogeneous and high-dimensional data [YBK18]. For example, learning implementations assist the medical imaging diagnosis in radiology, dermatology, and ophthalmology [Est+19], such as early detection of breast cancer based on prediction models analyzing tomographic images [Cic+17; Koo+17]. Similarly, the interpretation of genomes [LN15] has benefited from ML because the volume of available genomic data has much increased over the last years. For instance, learning algorithms are employed to predict the cancer type from biopsies [Cri+19] and how a certain kinds of cancer progress in patients [Car+18].

However, there are multiple challenges for learning-based analyses, in particular, reducing the computing capacity needed to train large volumes of data for neural networks. Another important issue consists in the need of removing noise from data, while preserving inherent relationships among data attributes [FHL14].

3.2.1. Understanding Supervised Learning

Machine learning encompasses quite different algorithms to learn (infer knowledge) from large volumes of data. Traditionally learning algorithms required that all the data was stored centralized on a single site. However, centralized algorithms are inappropriate for many analyses of biomedical data due to the data-sharing restrictions discussed in the previous chapter (see Sec. 2.3). Therefore, it is necessary to propose learning techniques that promote the distributed participation of multiple sites aligned with the current restrictions for processing and sharing biomedical data.

This section presents basic concepts to understand the training of supervised models. First, we explain the general learning process from the data, then supervised learning, how to evaluate a model, and the possible biases of the trained model. We then explain the ensemble learning strategy by composing multiple supervised models. Finally, we present models based on random forests that we use later on.

Learning from Data

Learning from data is the process of discovering underlying knowledge from large datasets [AML12]. For instance, data may contain input values, such as a symptoms, and outputs indicate explicitly (as part of *supervised learning* that we employ, see Fig. 3.3)

whether patients have or have not a specific disease. The input values are also known as attributes or features, and the output is typically expressed using labels or class values that are associated with each set of features. Learning then means inferring a relationship between the input data and the output variable, assuming that the output is related to the input attributes. The relationship inferred from the data is called a learning model.

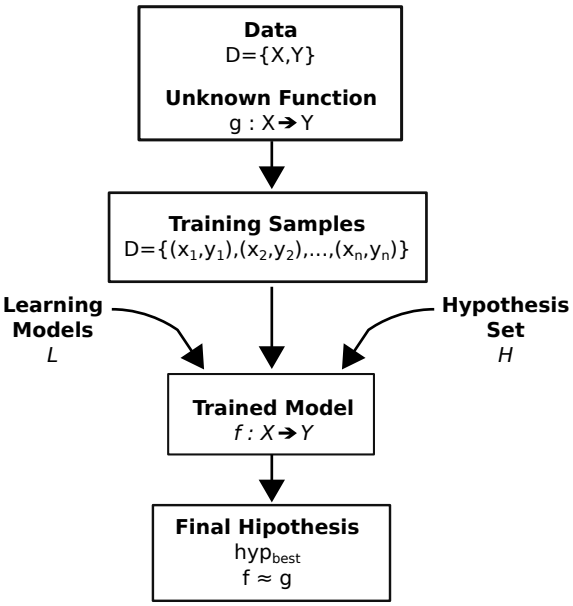


Figure 3.2 – General idea of the learning process approach.

The learned model can be represented as a function $M : X \rightarrow Y$ that approximates an unknown target function $g(x)$, where X is the input set (features), Y the output set (value classes). The labeled dataset D is composed of sets of input and output values, denoted as X and Y, as well as $D = \{(x_j, y_j)\}_{j=1}^{m} = \{(x_1, y_1), ..., (x_m, y_m)\}$, where m is the number of records, $x_j \in X$, $y_j \in Y$. In D, each record $\mathsf{x_j}$ has one output value $\mathsf{y_j}$, where $\mathsf{1 \leq j \leq m}$.

Figure 3.2 shows a general approach to learning from data. First, the learning process (algorithm) selects M from a set of candidates under multiple considerations called a hypothesis (*hyp*) [AML12]. Then, the inferred function M corresponds to the best mapping from the hypotheses, considered as the best hypothesis, hyp_B. The objective of the learning process is to infer a learning model, the best hypothesis, from the data set.

Scoring a Learning Model. The main objective of learning is to determine the best model from the data, i.e., that best maps the input space onto the output space. The

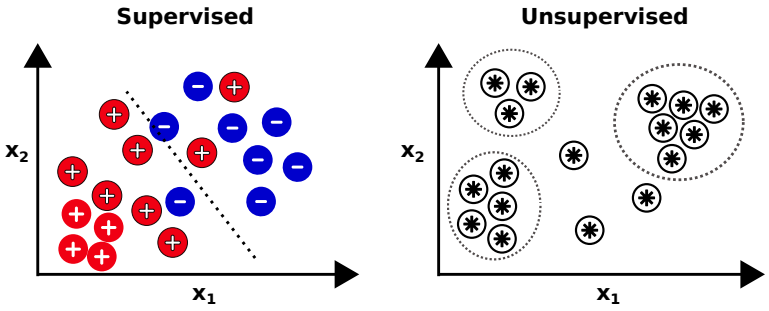


Figure 3.3 – Supervised and unsupervised learning. Supervised learning trains a model to best separate two classes $\{+, -\}$. Unsupervised learning seeks to cluster samples based on common features.

model always seeks to minimize the error of the result obtained by the inferred function M. For each sample, x_j, of the data set, $D = \{(x_j, y_j)\}_{j=1}^{m}$, it is possible to predict (classify) the output value class with the learned model, such that, $M(x_j) = \hat{y}_j$. The value of $\hat{y}_j$ corresponds to the predicted value for the record x_j using the model represented by the function M. Based on this premise, the output of $M(x_j) = \hat{y}_j$ should be the most similar value to the true class, the value of y_j. The classification error in supervised learning seeks to minimize the error defined as:

$$Error(g) = \mathbb{E}_{\{X,Y\}}\Big[\mathbb{I}\Big(Y, f(X)\Big)\Big] \tag{3.1}$$

where $\mathbb{E}(X)$ is the expected value of X, and $\mathbb{I}$ is the indicator function defined as:

$$\mathbb{I}\Big(Y, f(X)\Big) = \begin{cases} 0 & \text{if} \quad Y = f(X) \\ 1 & \text{if} \quad Y \neq f(X) \end{cases} \tag{3.2}$$

Using the Equation 3.2, the classification error can also be defined as:

$$Error(g) = \mathbb{E}_{\{X,Y\}}\Big[\mathbb{I}\Big(g(X) \neq f(X)\Big)\Big] \tag{3.3}$$

It is calculated after training the model for a subset of D denoted as the testing data set, $\mathsf{D_{TE}}$. From this error, it is possible to calculate the model's classification score, which corresponds to the portion of well-classified samples, calculated as $\mathsf{predAcc} = 1 - \mathsf{Error}$.

Bias in Machine Learning

In Machine Learning, bias is the deviation of model predictions from true values or desired outcomes. The bias problem exists because the training data is skewed, leading to a model based on incorrect assumptions and inaccurate predictions. For example, if most of the training data corresponds to information from a minority group, the trained model will be biased toward people from that group. The bias in machine learning models leads to unfair predictions due to the deviation of the model from the training data [Mit80].

In geo-distributed processing, bias can appear in trained models, for example, associated with the location of data, where knowledge underlies the geographic location of the samples. However, this knowledge cannot be common to other geographic regions. The bias can affect the trained model, especially under our proposal to compose privately and collaboratively trained models on the data of each geographically distributed site.

The bias problem has been studied in traditional scenarios, some bias-correction strategies have been explored in federated models [Bia+20]. In this thesis, we propose strategies for bias correction by securely sharing partial data to have a fairer collaborative model. However, there is a paradox in training a collaborative model: Collaborative scenarios provide greater diversity in the global model by promoting data protection for each organization participating while sacrificing, to some extent, the aggregate model's fairness and accuracy.

3.2.2. Ensemble Learning

Ensemble learning trains a model composed of multiple supervised models (classifiers), and the final prediction of each sample is by a combination of the results of each classifier in the ensemble. The main idea of ensemble learning is that combining multiple models compensates for the error induced by a single supervised model. Their combination can improve the prediction by aggregating multiple simple classifiers [SR18], due to delegate the learning to a single classifier tends to build models with low accuracy compared to combining multiple trained models [Zho21].

The ensemble training starts from an initial data set, D, from which random samples of data sets are generated to train the multiple classifiers that make up the ensemble model. Each data set to train each classifier is obtained using resampling techniques such as the statistical strategy Bagging (Bootstrap AGGregatING), proposed by Breiman [Bre96b]. Bagging refers to two tasks, bootstrapping and aggregating, for vari-

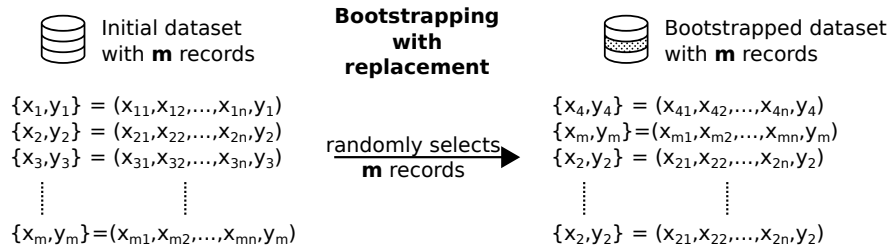


Figure 3.4 – Bootstrapping strategy, from the initial dataset D with m records, then m samples are selected with replacements to generate one bootstrapped data set.

ance reduction of the ensemble model. Bootstrapping is a statistical manner to evaluate the accuracy of a prediction through resampling samples from a single dataset to create multiple datasets [Has+09].

Figure 3.4 illustrates the bootstrapping method. There is an initial data set D with m records. bootstrapping generates a new data set with m records by sampling from D uniformly and with replacement. The new data set by resampling has m records and can consider repeated records of the initial data set. For example, in the figure, the record $\{x_2, y_2\}$ is repeated in the generated data set, but the total number of records, m, is equal to the initial data set. The resampling data set is known as the bootstrapped data set. For each classifier in the ensemble model, it is necessary to generate multiple bootstrapped data sets, one for each classifier. Finally, aggregating combines the prediction results of each classifier through a joint decision of the models.

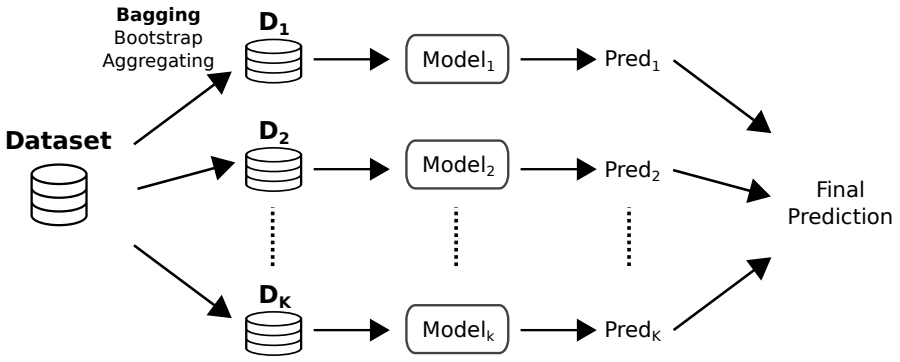


Figure 3.5 – Bagging strategy for training and evaluating multiple classifiers in an ensemble learning model.

Figure 3.5 presents the bagging strategy with an ensemble model composed of k classifiers. The first part generates independent bootstrapped data sets to train each classifier. Each classifier, $\mathsf{Model_i}$, is trained from each bootstrapped dataset, D_i. Next, each model

is used to make predictions, and a combination of the outputs determines the final prediction. Computationally, the ensemble learning approach is highly parallelizable due to the random sampling from the data, it is generated independently during the construction of each classifier. A popular ensemble learning method is known as Random Forest. It is an ensemble composed of multiple individual classifiers known as decision trees.

Decision Trees

Decision trees are represented by a hierarchical structure composed of nodes representing rules and leaves indicating classes. It is a supervised learning model and has been widely applied for two decades [SL91].

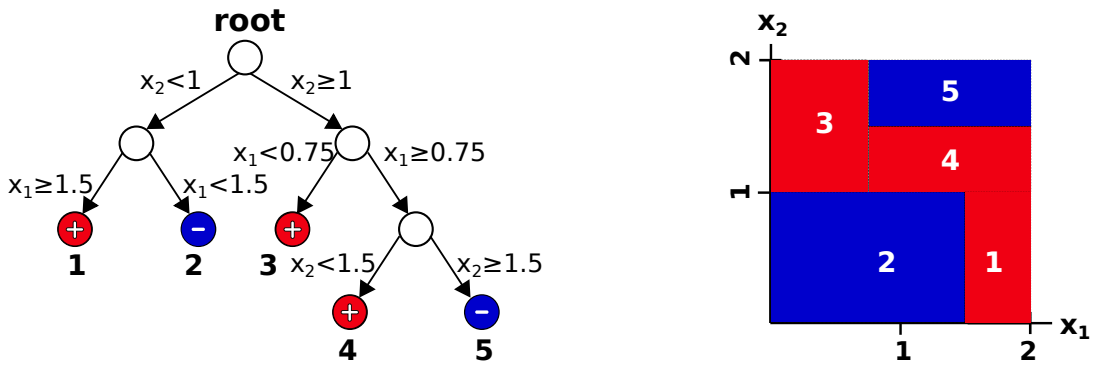


Figure 3.6 – Representation of a data set, in two forms, by a decision tree with two attributes (x_1, x_2) and one binary class $\{+, -\}$, and by decision boundaries.

Figure 3.6 is a simple example of a decision tree representing a data set with two attributes, (x_1, x_2), and one binary class, $\{+, -\}$. The tree has five paths from the root to the leaves. A rule of conditions represents each path, for example, path 1 can be defined as: $x_2 < 1$ and $x_1 \geq 1.5$, then the class value is, $+$.

Graphically, on the right-hand side, the decision tree divides the features (attributes) space into axis-parallel regions, where each one is associated with the tree paths. The classification of new records using the decision tree is performed by traversing the tree from the root to the leaf with the class; the path depends on satisfying each node's conditions. For instance, the record, $(x_1 = 0.8, x_2 = 0.2)$, is classified as negative '$-$' following the path 2. The challenge of building a decision tree is determining the tree levels and the split values at each node to define the left and right attributes. The construction of optimal trees is an *NP-complete* problem. Therefore, the levels and the split values are computed using heuristics based on statistical measurements to build near-optimal trees.

3.2.2.1. Random Forests

A forest is a group of trees, and they seek together to improve a single tree's learning score. Random Forests (RF) [Bre99], proposed in 2001, is an ensemble machine-learning algorithm composed of individual decision trees. RF has been widely used to analyze biomedical data [GPB11; CI12] and mainly to perform two tasks [Bou+12]: build a classifier model and obtain ranking variables by their importance level calculated during the tree's construction. The Random Forests algorithm has been very popular in analyzing biomedical data mainly for three reasons [GPB11]:

1. The final classification has high precision due to the composition of randomized decision trees.

2. RF is robust with respect to parameter settings, usually requiring three values, *ntree* (number of trees), *mtry* (number of attributes considered in each division), and *depth* (size of trees). The parameter selection affects the precision of the model.

3. The model is interpretable due to its individual hierarchical structure: non-experts can understand the decisions of each tree based on its structure, contrary to some learning models that are black boxes [Fre14].

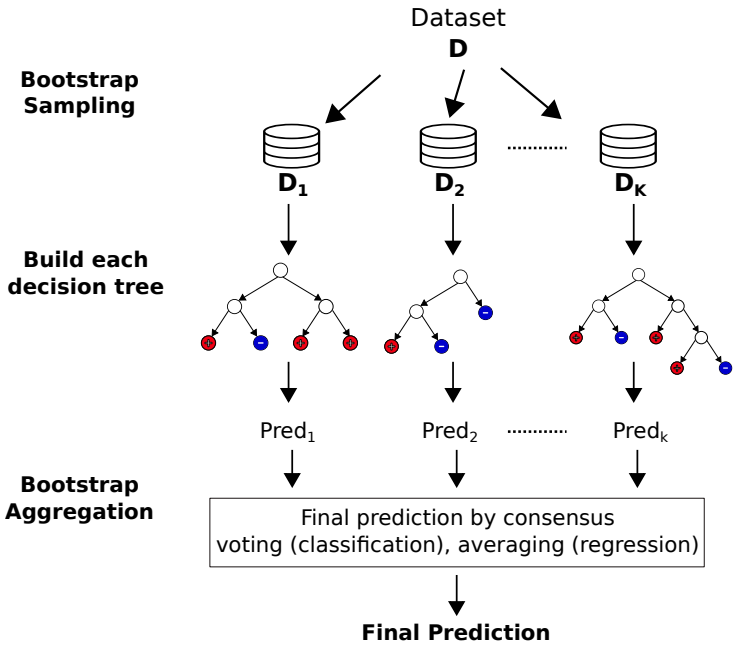


Figure 3.7 – Random Forests algorithm.

Figure 3.7 presents the idea of training and applying the random forest algorithm. Each tree is built from the bootstrap dataset of the original ensemble (using a bagging strategy) and a random sampling of features, each contributing to the error classification of the forest. If the class type of the data is a numeric value, then the training corresponds to a regression task, and the final consensus of the forest is achieved by averaging the results of each tree. On the contrary, if the class corresponds to a discrete value, it is a classification task. The final prediction is calculated by voting among the individual predictions of each tree.

Random forest mitigates the shortcomings of a single decision tree. RF improves instability by building a more accurate and diverse model considering the prediction of multiple randomized decision trees. The above aggregation considers all decisions with the same weight during the final prediction. However, the final prediction can be modified such that the prediction of each tree has a different weight to achieve a weighted decision of the forest (an extensive discussion of voting mechanisms is presented in [Rok09; BS16]).

Table 3.8 describes five classification methods using six properties to compare different kinds of models (and motivate the importance of random forests): predictive power, interpretability, scalability, computation speed, and ability to adjust to overfitting and underfitting. These last two refer to an undesired effect in the model. Overfitting is when the trained model perfectly fits the training data but cannot satisfactorily predict unknown/-future samples. While underfitting refers to the poor generalization of the model from the training data, the inferred function does not satisfactorily capture the relationship underlying the data [BN06]. These two effects can be treated in some models using parameters or by the nature of each model. For example, Random Forests by resampling, as Bagging, controls overfitting due to having more randomization trees. The symbol $\{+, -\}$ indicates how much the model favors the considered property, and the computational column indicates the cost during the construction and application of each model.

3.2.3. Multi-Site Forests

Very few Random Forests implementations have been proposed for the collaboration of geo-distributed trees. Here, we present three proposals with similar objectives as our approach implemented in this thesis.

- Federated Forests [Liu+20a] build the global model between multiple sites that share client IDs, but each site maintains different client attributes. A central site, known

Table 3.8 – Comparison of some Supervised Machine Learning models

Model	**Description**	**Pred.**	**Inter.**	**Scal.**	**Comp.**	**Over.**	**Under.**
Decision Trees	Builds a hierarchical flow (tree) and classifies traversing from root to leaves.	-	++	++	fast	sensible	sensible
K-Nearest Neighbors	Estimates the similarity between data vectors by choosing the nearest neighbors.	+	-	-	expensive	sensible	sensible
Random Forests	Sets of decision trees as weak classifiers aggregating the individual results of each tree.	+	+	++	fast	treatable	treatable
Support Vector Machines	Computes the most optimal hyperplane to separate classes.	++	-	-	high memory	treatable	treatable
Deep Neural Networks	Multiple layers with weighted connections produce an output from input data.	++	-	-	expensive	treatable	treatable

as the master, coordinates the model construction and stores all the data that is going to be distributed to the sites. The master first sends randomly chosen samples and attributes to each site. Then each site determines the best split attribute for the data returning the gained information value. The attribute's name and the gained information value are then shared with the master. Finally, the master compares the clients' values, identifying the highest value and the respective name of the split attribute. This process is between all sites to define the levels of each tree until the forest is completely trained. The master is responsible for coordinating the process by choosing the best attribute for each tree and performing the complete construction of the forest. Storing data in a centralized is not appropriate for analyzing biomedical data. This kind of implementation does not satisfy the current restrictions regarding biomedical data processing nor the collaborative analysis of biomedical data between multiple sites.

- DfedForest [Sou+20] is a distributed machine-learning system to build and share decision trees. The proposal trains models under Breiman's strategy of sharing models between the parties. They use blockchain technology to prevent malicious participants from altering the model during sharing. Before combining the local model with the one shared by others, the received (shared) model is validated on the local data. The validation determines if the received model increases the local model's accuracy. If so, the two models are combined, otherwise, it is omitted. The idea of omitting trees from other sites can disregard inherent knowledge in each site's data, leading to bias in the federated model. For instance, suppose that site i trains an unstable model $\mathsf{M_i}$ due to training data sets of small size. The site i then shares $\mathsf{M_i}$ with site j, but $\mathsf{M_i}$ can have a large variance in the probability of misclassification on data of the site j [SD98]. Thus, $\mathsf{M_i}$ is omitted due to its low accuracy on the data of j. However, part of the $\mathsf{M_i}$ model can be useful to build the collaborative model. The model $\mathsf{M_i}$ can still represent knowledge of patterns associated with the data of site i. Other limitations associated with blockchain as the construction of trees that must be ordered sequentially and inefficiency due to multiple users performing concurrent operations over large volumes of data [GR18].

- Multicenter Random Forest (MRF) [Li+20a] implements RF at multiple sites to predict the prognosis of clinical data. MRF generates synthetic data from local data based on Generative Adversarial Networks (GAN) and then shares it with other

sites to validate each local model. The sharing process employs DP combined with the GANs known as DPGAN methods. Each local model is validated using synthetic data to find parameters that provide better model accuracy. However, generating synthetic data using GAN methods requires various preprocessing steps that affect the quality of the synthetic data [MO14]. Besides, DPGAN-based methods control data leakage, but there must be a trade-off between preserving privacy and the quality of synthetic data. Furthermore, DPGAN methods do not provide precise techniques to resolve all security risks during synthetic data generation [BDR19]. Therefore, MRF has limitations introduced by the generation of synthetic data, which can affect the construction of the model. Unlike our approach, which does not use synthetic generation methods, it only shares models with the trained trees at each site. Our approach is complemented by three strategies to securely share data to correct bias without generating synthetic data.

Discussion and Main Challenges

Learning techniques are an emerging area widely used in the biomedical domain. Current restrictions on sharing biomedical data have promoted federated initiatives to keep the data on each site. As a result, the trained models can be diverse and even have an ensemble model composed of diverse classifiers. This subsection presents the reasons for implementing Random Forests mainly due to their properties and application over biomedical data. Additionally, we present relevant identified challenges regarding the training of the collaborative forest and the state of the art.

Finally, we identified an opportunity to implement RF in multi-site scenarios due to its popularity for analyzing biomedical data and the current state of multi-site random forests. The multi-site RF implementation poses several challenges:

(I) implementing ensemble learning strategies applied to fully distributed architectures,

(II) aggregating the best trees to each site to have a global model with similar learning performance to the centralized version,

(III) designing flexible and scalable distributed workflows to analyze biomedical data,

(IV) incorporating sharing strategies while preserving data privacy and security.

The implementation chapter considers the above challenges to validate that our collaborative forest approach satisfies those identified in existing multi-site random forests.

3.2.4. Distributed Machine Learning

Distributed Machine Learning (DML) has emerged as an alternative to analyze data in a distributed way, between multiple sites, avoiding sharing data to a central processing site [Ver+20; PG13], motivated mainly by three reasons:

- the ability to scale computational resources for the training of models,
- the use of distributed and parallelization strategies during the learning process, and
- the need to ensure data privacy during the training of model.

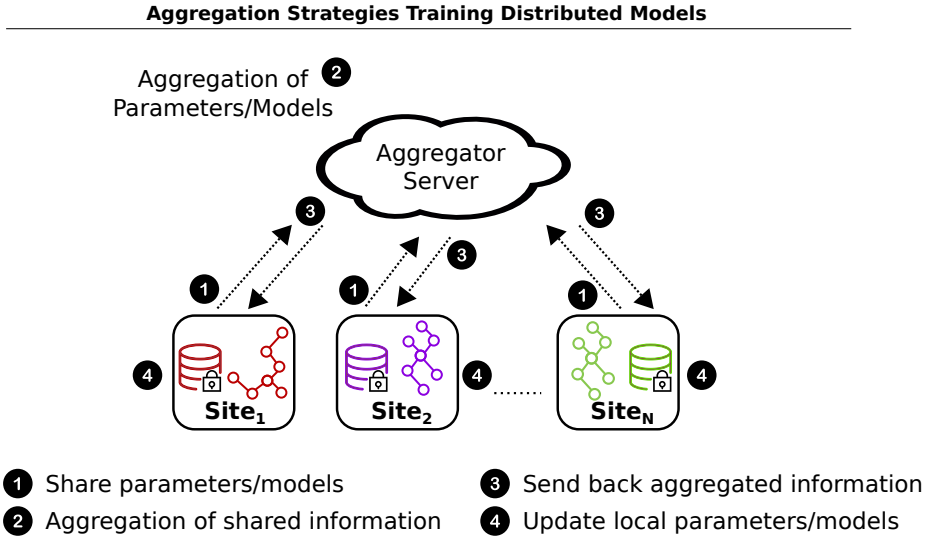


Figure 3.8 – Distributed Machine Learning aggregation strategies.

The distributed learning considers the participation of multiple sites through two collaborative strategies by sharing intermediate model parameters and sharing trained models privately at each site [Liu+20b]. In the first strategy, each site trains a local model and shares the parameters of its model with others. Then, through parameter aggregation, one site aggregate all parameters to build a global model based on these parameters shared by all. In contrast, the second aggregates local models shared by each site, at the end, a global model is built by the participation of multiple sites. Figure 3.8 presents a general aggregation strategy, where the steps 3 and 4 change depending on the selected strategy. Federated Learning (FL) [McM+17] is a notable example of a distributed learning based on the first strategy, parameter aggregation. In FL, the collaborative model is adjusted

by sharing the parameters of each local model by participation rounds. This strategy considers the collaboration between the sites to share parameters such as, for example, the weights of a deep neural network and thus, at the end, have a trained neural network by sharing local model parameters.

On the other hand, model aggregation can considers the execution of heterogeneous workflows between multiple sites that share partial information, such as trained supervised models privately. For instance, train a global random forests by aggregating trained forests at each site. The sites are heterogeneous, and therefore models at each site are trained on different machine capabilities where data-sharing and communication have different restrictions and capacities.

For today's biomedical data analysis needs, distributed machine learning is an active and relevant area in the coming years with many challenges such as improving traditional statistical techniques, ensuring efficient communication between multiple sites, enhancing current privacy and security techniques, enabling the processing of heterogeneous sites, and seeking a good precision of the global model [Ver+20; LWH; Xu+21].

Parallelism Strategies for DML

Different initiatives have been proposed to alleviate the workload training on large volumes of data, such as applying parallelism strategies during different phases of the learning process, parallelism at the level of data, models, and hybrid techniques [Ver+20; Liu+22; Yan+19].

Data Parallelism. The most intuitive way to apply parallelism is on the data. The training is applied in parallel on different machines from each dataset. Model training is data-centric, and during training, multiple independent devices can communicate synchronously or asynchronously to train the global model. The communication considers the sharing of parameters or intermediate models between the parallel devices; for example, implementations based on MapReduce have synchronous communication, while the parameter server architecture trains asynchronously. The communication does not consider sharing raw data between the parties, only intermediate values or models to apply some aggregation strategy to train the global model.

Data Parallelism is appropriate for horizontally partitioned data, where multiple sites contribute with intermediate parameters to build the final model. Horizontal data partitioning works on the same feature space over different samples at each site. Figure 3.9

illustrates the horizontal data partition strategy in federated scenarios. Assuming that three sites have the same attributes (feature space), but the samples correspond to unique patients from each site. For example, three sites analyze symptoms variables (x_1, x_2, x_3) of a given disease y (yes or not), but the attributes correspond to exclusive patient samples from each site (unique sample IDs). Each site trains a local model on its privately partitioned data; then, it can be shared to be aggregated globally.

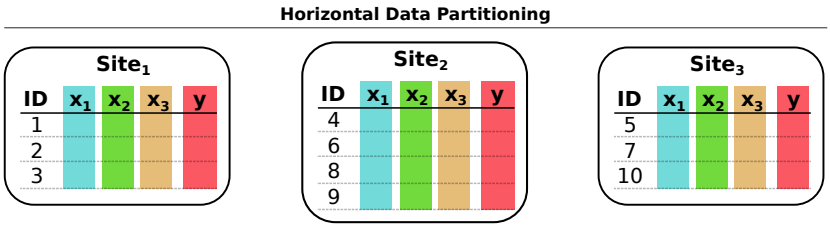


Figure 3.9 – Horizontal Data Partitioning.

Model Parallelism. Model Parallelism considers the participation of different processing devices, each one contributing independently to the construction of the model. Model Parallelism is also known as the model-centric training strategy. The execution on each device is independent from the other tasks. Model Parallelism is suitable for vertically partitioned data, where each of the sites trains the model on different feature space to share knowledge from each set of attributes. Each site is thus similar to an independent processing device. Figure 3.10 shows the vertical data partition strategy. Three sites use the sample ID space but differ in the attributes (different feature space). For example, three independent medical institutions have samples from the same patients, but variables correspond to different data; i.e., the data comes from different sources. In the end, the global model collectively generalizes relevant knowledge for each patient sample.

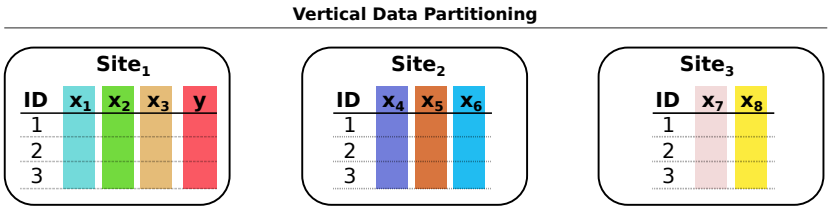


Figure 3.10 – Vertical Data Partitioning.

Hybrid Parallelism. A hybrid approach combines the two previously-discussed parallelism strategies to train the model collaboratively, meeting the conditions of each approach. Different implementations take advantage of data and model parallelism during collaborative training models such as Stochastic Gradient Descent (SGD) [Dea+12] and distributed deep networks framework [SCJ17]. They partition the data between different machines, and the processing on independent data in each one contributes to the parameters of the global model.

Data Privacy in Distributed Machine Learning

One of the benefits of Distributed Machine Learning is minimizing the privacy and security risks of biomedical data in its storage and sharing. The aggregation strategies do not consider sharing raw data, only shared intermediate data such as parameters or models. Furthermore, these strategies can incorporate security techniques. Privacy risks are reduced by not sharing sensitive data between the parties. The shared intermediate information can adopt security techniques to shield leakage of this data during collaboration. An important premise during data aggregation is that all sites are considered trusted parties.

Security methods over data encompass different levels depending on the type of processing. The methods may contemplate strategies to safeguard the privacy of the data from its storage, processing, and sharing. For example, secure collaboration between sites is facilitated in distributed learning scenarios because the DML strategies avoid sharing sensitive data like confidential patient information [Li+20b]. Furthermore, distributed scenarios can preserve the privacy of shared data through techniques applied to these prior to sharing, such as Differential Privacy (DP) or based on Homomorphic Encryption (HE) techniques. Moreover, the collaboration can also be done through the participation of a trusted third-party site to share partial information, such as implemented in Secure Multi-Party Computation (SMC). These strategies are briefly explained below, but we present details of each one in Chapter 5.3. There, these strategies are detailed as a complement to the fully distributed implementation proposed in this dissertation. Implementing these techniques is not exclusive; therefore, we combine them to improve security and guarantee data privacy [Hao+19; Zha+20].

Differential Privacy (DP). Differential Privacy [Dwo+06] introduces noise to the training set. The goal is to introduce values that distort the original data value in a

controlled form. The noise introduction is controlled and seeks to maintain the precision of the learning models as if they were trained on original data. DP techniques are applied to select records to generate training set noisily but without losing intrinsic knowledge about the data. This strategy has been widely implemented in Distributed Machine Learning algorithms [Aba+16; Dwo08] to mislead the adversary in the face of improper access. However, the great challenge in differential privacy techniques is to guarantee performance of model training in the presence of noise.

Homomorphic Encryption (HE). Homomorphic Encryption [RAD+78] was proposed in the 70s to compute ciphertext without applying decryption techniques. HE techniques have been used to perform calculations on encrypted values [Van+10]. For example, sharing numerical values of model parameters during training as gradient values. Analyses based on HE techniques share encrypted numerical values to be analyzed during the collaboration [MG19]. If an intruder obtains part of the shared data, these values are completely encrypted. There are three main HE schemes based on the definition and complexity of the mathematical functions proposed in each one (a broad explanation is presented in [Yan+19]).

Secure Multi-Party Computation (SMC). Secure multi-party computation [Yao86] was proposed in the 80s and used to train collaborative models. SMC techniques allow for secure computing between multiple parties through the participation of a trusted third party during model training [Fen+19; Ohr+16]. The process involves the participation of keys with public and private values to each site, which is not entirely shared but partially ensures security. However, SMC techniques have not been much used in training models on large volumes of data due to computational inefficiency due to the time and capacity required to share data with the third party. Finally, some implementations combined between DP and SMC offer better security guarantees than only the SMC techniques [GX15; Ria+18]. Nevertheless, they have the limitation of efficiency due to the cost of transferring big data among the participation of multiple sites.

3.3. Conclusions

Most of the analytical tools and techniques reviewed provide only mechanisms for partial collaboration among geo-distributed sites. Most of them are based on collaborative

strategies where one of the sites acts as a central master and the others are slaves nodes. There are still numerous open issues, including explicit data and computation placement management, dynamic scheduling of resources, robust security and privacy features, and flexible data placement strategies. Additionally, ethical and legal constraints must be considered during the execution of biomedical analyses [Cor+18; MEO13]. As a solution to these issues, we have advocated Fully-Distributed Collaborations (FDCs) in this dissertation. The FDC analyses are appropriate for processing biomedical data geo-distributedly, respecting the restrictions and promoting collaborative work. This section presents the open issues in detail, which motivate the following chapters corresponding to the thesis's contribution.

Data and Computation Placement

To address the problem of data and computation placement, tools must place data and computations in the optimal location according to specific criteria, e.g., putting data in the infrastructure best suited to analyze it or moving data while preserving data locality and security constraints. This problem is known to be *NP-hard* in computational complexity, and it is still actively studied [LD11].

The development of heuristics with efficient implementations for data and computation placement is an ongoing endeavor. As we have shown before, current BATTs offer only some basic strategies such as data analysis in a centralized setting, uniform data partition among multiple nodes, and computation based on cluster distribution on homogeneous nodes. Therefore, other research directions consist in providing relevant abstractions to handle multi-site biomedical data analyses.

A recent and exciting field of research is the application of Machine Learning techniques to optimize distributed computations on scattered data that cannot be shared. The application of learning models on these data without moving them to a central site. On the contrary, consider sharing local model parameters or trained local models without revealing sensitive data [Liu+20b]. A prime example, federated learning [Xin+15; McM+17], has been applied to analyze large genomic data sets spread over multiple sites worldwide. However, there are many challenges, such as communication latency, systems heterogeneity, heterogeneous data, and privacy preservation. Nowadays, Distributed Machine Learning [Ver+20; PG13] seems a promising future research domain.

Privacy and Security

Research collaborations must comply with strict security policies imposed by governments over biomedical data. These policies may be enforced in a national context (as those implemented by national governments), supranational (*e.g.*, policies enforced by the European Union), or even globally. However, as we have shown before, the current set of tools and techniques used to analyze biomedical data address the problem of security and privacy only partially. They either delegate security to the computing infrastructure (*e.g.*, a cloud provider) or provide very basic mechanisms. Therefore, biomedical researchers must address security by means external to the tools they use. In order to support FDCs, BATTs have to better support privacy concerns. Thus, several researchers are investigating more sophisticated security mechanisms. For example, secure containers encapsulating confidential data allow data to be used in complex computations without exposing participating stakeholders [BS18]. Other approaches investigate data watermarking techniques augmented with encryption and cryptography operations [Bou+22]. However, the security in BATTs is still a great challenge, especially by security and privacy risks encompassing large parts of complex workflows, as mentioned by the authors from the NSF's SWIP project.

Scalability and Performance

The exponential growth of biomedical data has forced scientists to scale their experiments and applications. Several techniques have been used to address the problem of scaling up these analytical experiments. A common practice consists in implementing dedicated High-Performance Computing (HPC) systems. Most of the world's supercomputers belong to this category. Another common technique to improve performance and scalability is deploying scientific experiments on clusters of computers, on on-premise facilities, or in the cloud. Some authors propose to mimic the architectures for massive computations, and data storage proposed by internet corporations such as Amazon, Facebook, or Google to create Data-Intensive Scalable Computing (DISC) facilities for scientific applications [BV16; Val+18].

Each strategy has its benefits and drawbacks. For example, HPC systems offer centralized computing resources focused on high-performance applications with many floating-point operations. But it requires all the data to be available for the computation, and it also requires a significant investment in infrastructure. These infrastructures may also

be costly to maintain, and they can pose problems regarding data ownership and confidentiality. On the other hand, private and hybrid clouds for the deployment of scientific experiments constitute flexible means to configure multiple computing scenarios. However, data ownership, security, and legal restrictions hinder the implementation of massive biomedical experiments on the cloud. On the other hand, DISC systems, *e.g.*, based on Hadoop and Spark, enable processing large volumes of data and the distribution and scaling of resources in clusters or clouds. They include fault-tolerance mechanisms and are generally controlled by the owner of the DISC system. However, these infrastructures may be costly to maintain and pose problems regarding data ownership, data confidentiality, and data localization.

None of the above-discussed systems will support Fully-Distributed Collaborations (FDCs) as presented in Section 3.1.3. Instead, they all propose some essentially centralized solution. They lack the means to address the three FDC requirements: decentralized computations and flexible data placement, compliance with security and data protection policies, and performance and scalability based on the interests of participating sites. Thus, numerous research opportunities exist to develop solutions addressing these requirements while providing performance and scalability benefits.

We found some computational approaches that may support basic, Fully-Distributed patterns. For example, hybrid strategies, exploiting the MapReduce benefits in the cloud [HKS18; ZLS15; Cha+12; Xu+17b]. Similarly, some workflow systems also have adopted combinations of these paradigms [Liu+15; Wol+13; Tay+07b]. However, as seen in the review, these solutions address some primitive form of technical flexibility, and few provide explicit abstractions to handle all the desired requirements. Our taxonomy also revealed another opportunity, for example, to strive to implement flexibly configurable systems that enable high scalability. The current methods to address high scalability and performance require some form of centralized control.

Need for Fully Distributed Collaborations

This chapter has identified several research opportunities in data and computation placement, data privacy and security, and performance and scalability. These areas of investigation make sense for fully distributed collaborations only when they are considered together. In the discussion, we explain how to achieve fully distributed collaborations.

First, the CAP theorem [GL02] applied in the same logic to distributed systems indicates that it is impossible simultaneously to have the three properties in a distributed

system: consistency, availability, and partition tolerance. Therefore, a trade-off has to be enacted between the three properties to achieve a practical system supporting a distributed database in an FDC system. Similarly, the scalability trilemma [BFV19] states that highly distributed systems cannot simultaneously combine: decentralization, scalability, and security. Therefore, again, there should be a balance between the levels of support for each of these three properties. The classification presented in Tables 3.5 and 3.7 categorizes the tools and approaches into properties such as resource location, privacy and security, and scalability. It is clear that many cover two of the three properties; therefore, there is an open question about solving them together. There is still a great challenge to achieve all three simultaneously because researchers seek to compensate for two of the properties indicated in the CAP theorem or the scalability trilemma.

Nevertheless, not only technical constraints will affect the development of FDCs. Legal and socio-economic misconceptions should be overcome before having a full implementation. We expect that in FDCs scientists will not have to disregard security or legal restriction to achieve high scalability and performance. On the contrary, an FDC solution will coordinate dynamically the execution of complete workflows exploiting and combining heterogeneous computing facilities.

Part II

Contributions

CONTRIBUTIONS

The second part describes the contributions of this thesis which are presented in the remaining chapters.

- Chapter 4 presents the general motivation for Fully Distributed Collaborative biomedical analyses, defined as Fully Distributed Collaborations (FDCs). The case presented motivates the need to process biomedical data following the FDC approach. During the chapter, we present relevant FDC features, distributed processing architectures, security and privacy mechanisms.

- Chapter 5 describes the MuSiForest algorithm, which implements a particular scenario that adopts the FDC properties. The MuSiForest algorithm takes advantage of distributed collaborative learning strategies. The development of the chapter presents relevant concepts associated with distributed machine learning, followed by the technical components of the implementation, and complemented by the experimentation and results for the proposed scenario.

- Chapter 6 introduces the language to specify distributed and collaborative analysis. The distributed workflow specification language is defined as FeDeRa. The language supports the specification of analyzes driven by the need for data. FeDeRa is enriched with mechanisms to specify multi-site analyses that are more expressive and intuitive.

Chapter 4

Fully Distributed Collaborations

Introduction

There have been many forms of cooperation among interdisciplinary groups that involve sharing knowledge and the collaborative generation of results [MK03; Mil00]. For instance, the BIOINFOMED [Mar+04] project is an initiative funded by the European Commission (EC) to facilitate the collaboration between experts in medical informatics and bioinformatics. Similarly, the ICAN project [ANR19] (presented in Section 2.2.1) involves collaborative work between 34 French entities where each member provides clinical records, medical images, and genetic data collected from patients, notably through biological samples [Bou+17]. However, the predominant architectural model for biomedical collaborative analyses still consist of centralizing and sharing the underlying data and performing analyses through supercomputers or cluster infrastructures at a single or small number of organizations [Sch+10a].

Recently, the need for more widely distributed collaborations has been noted [BS18; Bou+19; PCA15]. Several arguments favor a higher degree of distribution: more and more organizations dispose of high-performance infrastructures for large-scale analyses, local data may be preferred to be kept private, massive data transfers are too time-consuming, etc. During the last years, federated analyses have emerged as a strategy to train models on mobile devices without the need to centralize data [Kon+16a; Kon+16b]. These initiatives promote collaborative analysis and protect sensitive information such as biomedical data. In this chapter, we argue that these kinds of collaborations are part of a wider spectrum of collaborations that we call *Fully Distributed Collaborations* (FDCs).

FDCs are research endeavors that harness means to exploit and analyze massive amounts of information collaboratively over geo-distributed infrastructures while respecting social, human, and legal constraints. Thus, FDCs require tools and techniques for collaboration that can use advanced distributed (data and computation) architectures while coping with complex sociotechnical constraints and heterogeneous networks. FDCs

promise to enable more powerful biomedical analyses defined in terms of distributed workflows operating over large volumes of private data. FDC approaches will support cooperation between geo-distributed research groups or organizations that are generally subject to several restrictions derived from legal frameworks, data privacy regulations, and available local computing infrastructures.

This chapter introduces the concept of Fully Distributed Collaborations and investigates their characteristics and the features needed by tools supporting such research endeavors. Concretely, The chapter is structured as follows. Section 4.1 presents the definition of Fully Distributed Collaborations (FDCs). Section 4.2 describes collaborative research cases that may be supported by FDCs, including some based on machine Learning algorithms. Section 4.3 presents the distributed architectures supported by FDCs. Finally, Section 4.4 describes the security strategies enforced by FDCs, and the conclusions are presented in Section 4.5.

4.1. The FDC concept

Nowadays, analyzing data in multi-site scenarios is common in real projects due to known restrictions on transferring biomedical data to central processing sites. Traditional data analyses are processed in centralized environments, which are often too limited for the current needs for biomedical data analyses. Overcoming these restrictions in collaborative research settings requires improving tools, applications, and infrastructures for research collaboration. We argue that new tools for scientific collaboration should be supported by geographically distributed architectures that support distribution, concurrency, and security mechanisms driven by data needs. We thus propose a method for scientific collaboration called Fully Distributed Collaborations (FDCs). As stated previously, FDCs are research endeavors that harness means to exploit and analyze massive amounts of information collaboratively over geo-distributed infrastructures while respecting information ownership, data privacy, and social and legal constraints.

4.1.1. FDC Properties

We advocate more flexible and dynamic research cooperations through FDC analyses. FDCs involve several geographically distributed research centers with their biomedical data, and their computational infrastructure. The central role of FDC analyses is to

model tasks across heterogeneous cross-site workflows. An example of these heterogeneous analysis workflows is illustrated in Figure 4.1. The workflow involves the participation of two countries that cannot (reasonably) be implemented through traditional systems due to centralized dependence on data placement and computer processing. This workflow may correspond to the extension proposed for the ICAN project (discussed on page 40). In the workflow, France has three member sites processing data located in Nantes, Rennes, and Paris, while Colombia has a single site processing data in Bogotá. Each of the two countries applies its own restrictions on sharing data, such as the GDPR [CC16] in France and the "habeas data" law in Colombia [Col12]. In this scenario, the FDC analyses satisfy each regulation member's requirements. Colombian law considers responsible data sharing with countries that dispose of clearly defined regulations, such as the GDPR. Therefore, the proposed workflow can be reasonable through FDC analysis while complying with legal restrictions.

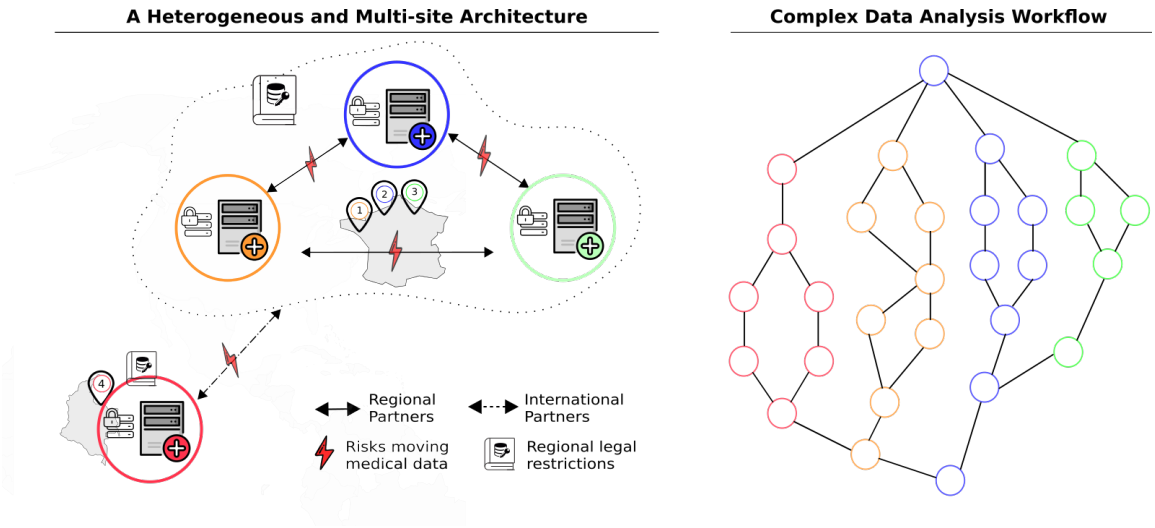


Figure 4.1 – FDC analysis between four cities distributed in two countries.

The scenario illustrated in Figure 4.1 can consider sharing models trained from the private data of each site. Assume that each site has genomic data corresponding to its private patient samples. For example, Bogotá stores *1.7TB* of data corresponding to Colombian samples, while in France, Paris, Nantes, and Rennes store *3.0TB*, *1.2TB*, and *1.0TB*, respectively. One relevant consideration, from *1.7TB* of raw data, the compressed trained model can be stored in *400MB* [Can+16]. Therefore, if Bogotá wants to share information with France, it does not need to share almost *2.0TB*; on the contrary, Bogotá needs to share only *400MB* through learning models. This notable data size reduction impacts the

French sites similarly due to the notable reduction in sharing raw data versus trained models. In addition, the reduced size of shared data favorably impacts technical aspects such as communication channel capacity and required transfer time. Regarding legal aspects, multi-site analyses are limited by the agreements established in such regulations. Therefore, an FDC-based analysis should facilitate the automation of complex workflows, such as the one presented between France and Colombia, while respecting diverse restrictions.

Our FDC proposal aims to alleviate legal and technical considerations to analyze data collaboratively, like the scenario illustrated previously. The FDC approach supports three significant architectural properties to satisfy complex and heterogeneous workflows:

- **Data and computation placement**: in traditional scenarios, data moves to the computing site, typically a centralized site; on the contrary, we propose that computations move to/are performed at data-owning sites, thus keeping data completely private.
- **Privacy and Security**: transferring data to central processing sites generates risks associated with the privacy and security of biomedical data. We propose security mechanisms when data sharing is necessary between sites during the FDC analyses. The transferred data may be intermediate results obtained during the workflow execution.
- **Performance and Scalability**: traditional scenarios ignore computational facilities installed at each site, often resulting in a large computational overhead at the central site. In contrast, we propose using the local infrastructure available at each site, allowing the integration of heterogeneous tasks supported by different scalable architectures.

These three properties are essential to satisfy the analysis based on FDCs and are taken into account in the context of the implementation of our algorithm:

- *Dataflow-oriented programming* guarantees communication based on data availability between independent tasks, that is, analyses by data-driven execution.
- *Declarative concurrency* extends the declarative model to deal with concurrency, mainly designing workflows on demand for data on independent tasks.
- *Message passing* patterns between independent and active tasks that interact via messages asynchronously.

- *Distributed programming* mechanisms ensure communication and interaction between tasks; although they are located in independent sites, they can depend on data availability to fulfill their execution.

4.1.2. Data as first-class citizens

In a programming language, a first-class citizen refers to an entity that supports other entities' operational properties [Sco00]. However, in treating data as first-class, workflow-based analyses do not always have this consideration. Therefore, in FDCs, data as first-class citizens means that data can be passed as an argument, returned in workflow steps, and assigned as a value to a variable, even across multiple sites. Therefore, the data is treated as a first-class citizen, and their flow drives the analysis execution. The sites can require sharing information during the execution through dataflow variables. These variables pass data as arguments to other steps and can also be returned and assigned by the site responsible for its creation. This behavior is inspired by processing based on dataflow modeling.

The FDC approach implements data-driven modeling with the need to drive execution and avoid errors due to the unavailability of data. External tasks must wait for the owner's data to bind them and continue executing the paused task. Treating data as a first-class citizen is advantageous compared to current biomedical data analysis systems, which require external routines and complex custom plugins to indicate the interaction with other processing devices.

Relevant Dataflow Features

We support sharing different kinds of data during the analysis, raw data, trained models, or partial results to be operated by other sites. Our approach to FDC analysis incorporates relevant dataflow modeling features into the workflow analysis execution:

- **Data on demand**. The interaction between the site that owns the data (who provides it) and those who need it (who require it) is on demand. The owner site publishes the data while the requester consumes it as input to the local process. The data can be consumed as long as it is made available by the owner.

- **Immutable data**. The data shared from one site with others is immutable. The site owner binds the data and is responsible for publishing it to others. The immutability

property minimizes errors due to different values for the same variable. For example, if multiple sites access the same variable, the same variable may have different values due to multiple assignments in parallel. Therefore, the immutability of variables guarantees secure operations in terms of the uniqueness of the shared information.

- **Dynamic and heterogeneous analysis**. This characteristic suggests that the execution of the local tasks is independent and must be adjusted to the capabilities of each site. The execution does not depend on what happens on other sites unless others require the data. If the local execution throws an error in any site, the site that provides data must guarantee data availability for those who require it. Furthermore, each site may have different configurations, *e.g.*, machine capacity, and installed base software. Therefore, FDC systems need to be scalable and flexible for heterogeneous on-site installations.

- **Semi-synchronous mechanisms**. This mechanism provides mediation through intra-site or inter-site depending on the workflow task type. Our FDC approach implements features of synchronous and asynchronous communication. For example, intra-site tasks take advantage of synchronous communication since local execution at each site does not usually require external resources. In this sense, the tasks are executed at each site regardless of what is happening at the other sites. In contrast, inter-site communication adopts asynchronous communication. The interactions with other sites use asynchronous mechanisms, where data availability activates on-demand tasks that require external data. The execution of FDC-based workflows employs both mechanisms.

4.2. FDC-based analyses supported

FDCs support diverse types of analysis as long as it takes advantage of the local computing at each site while respecting data-sharing restrictions. These analyses involve tasks that run sequentially at each site or in a distributed manner where participants can share information, such as trained models in the case of machine Learning algorithms. We present abstract cases of FDCs covering various dimensions based on learning analyses and traditional scenarios such as single-site processing to cross-site analysis.

Based on Learning models

Retaking the collaborative scenario illustrated in Figure 4.1 between Colombia and France. The outcome analysis can correspond to training a federated model to improve specific disease prevention. The collaborative model refers to a federated classifier to predict the risk of having or not having a specific disease. For example, the ICAN project extension can consider international cooperation to share knowledge and geo-distributed data from patients with aneurysm disease around the world (see Sec. 2.4).

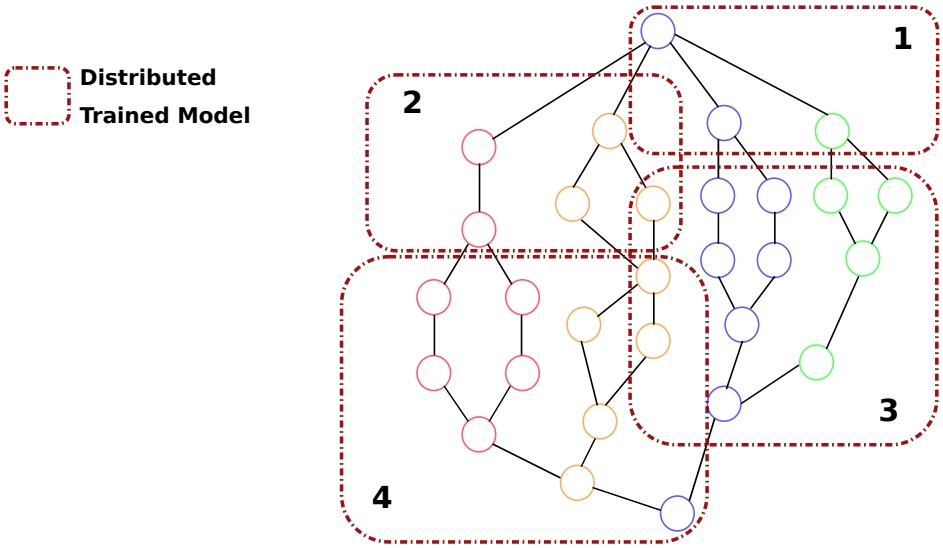


Figure 4.2 – FDC Machine Learning over multi-site workflows.

In this sense, Figure 4.2 presents four sites where various learning steps are computed within the workflow analysis. The four sites have tasks unique to each site and other tasks requiring external data from other parties. Therefore, some tasks like *site1* occur sequentially, while others occur in parallel and partially share data. The distributed workflow scenario illustrated corresponds to a specific analysis case based on machine learning techniques that seek to build a final federated model as part of a biomedical collaboration. However, FDC-based analyses also comprise simple cases to more complex analysis scenarios. For example, processing at a single site through local distribution, adding more sites for new cooperation partners and reaching more complex scenarios over multi-site workflows. This scenario is also preferred compared to the previous one (see Figure 4.1) for the data size kept during the training of the models, impacting the technical restrictions favorably due to the amount of data to be shared. FDC-based analyses are adaptable to

particular needs regarding biomedical data, considering the data type to be shared, local infrastructures, and the number of sites participating in the process.

A Collaborative analysis between multiple sites

The analysis between multiple sites often requires sharing information securely between them. A collaborative analysis can train an ensemble learning model between multiple geo-distributed sites [Zho21]. The first site trains a classifier, such as Random Forests, while the others train classifiers as Support Vector Machine (or any model described in Table 3.8). Each one trains a classifier on local data private to each site. After the local training of each classifier, one of the sites is responsible for aggregating the models to build a collaborative model. The data type at each site may be different; for example, some sites may have genetic information while others have medical images. Although these data types are different, they are useful for studying specific diseases, such as the ICAN project. The outcome is an aggregated model that brings together knowledge from the two parties to predict future disease cases from two different data types.

FDC-based analyses are most useful in this case with many sites; each site has a local workflow and does not depend on the executions on other sites. However, there may be data dependencies between sites. In addition, FDC allows for different computation capacities. Therefore, some sites may have to share their data with another site to be analyzed. The multiple sites may have different computing capabilities corresponding to each site's infrastructure. Heterogeneous factors such as each site's computing capacity and the diversity of the biomedical data. Our proposal to analyze biomedical data through workflows such as those supported by FDC considers the previous two factors, such as the geo-distributed scenario illustrated in Fig. 4.1. Additionally, we support variety in the local tasks executed in each site, following the analysis driven by data sharing information securely among all. The FDC analyses are heterogeneous and seek to expand the constraints of current workflow systems and applications used to analyze biomedical data because they provide limited support to the geo-distributed analysis and even support heterogeneous data and architectures [Gar+22].

FDCs also provide flexible and friendly management and configuration mechanisms by extending a specification language (presented in Chapter 6). Usually, biomedical tools require custom routines designed by computer experts to deploy distributed analysis, which is an obstacle for biomedical experts designing biomedical analyzes. For example, heterogeneous software environments, diverse computing devices, and complex analyses

require specialized workflow languages.

4.3. The Architecture of FDCs

FDC systems support traditional collaboration architectures such as sequential and hierarchical ones. Furthermore, FDCs support fully distributed topologies, as illustrated in Figure 4.3. The FDC-based architectures are inspired by traditional topologies in distributed environments [Bar64; Ver+20]. They seek to support heterogenous complex analysis and computational capacities that are installed on different site.

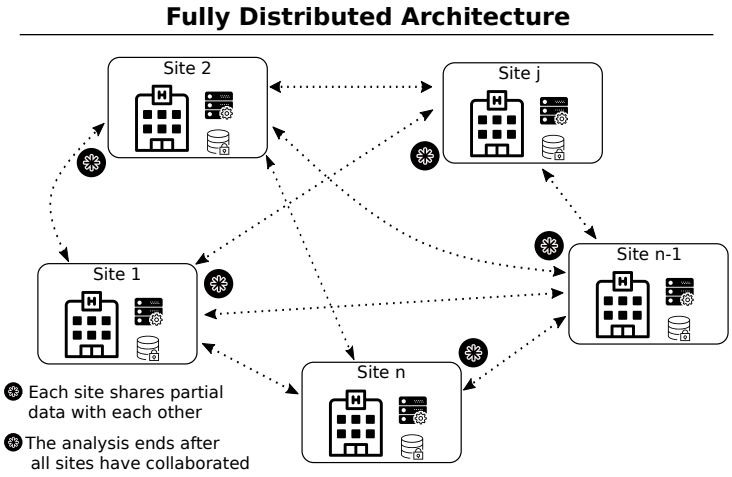


Figure 4.3 – FDC Fully-Distributed Architecture.

The Fully-Distributed architecture performs point-to-point communications between the sites that all have the same role. Therefore, all sites have the same importance during the process. A fully distributed analysis scenario can share transformed data as trained models with each neighbor's site to achieve full participation and construct a global model. This architecture is more scalable than sequential and hierarchical one; it also eliminates dependencies on a single site in the event of possible failures during the process.

4.4. Privacy and Security

Our FDC approach considers sharing information secure satisfying legal privacy restrictions during the analysis. Technologies based on encryption and cryptography methods safeguard data privacy during storage, processing, and sharing. For example, public-

key encryption methods to securely store and share genomic data [Kan+08], as well as based on homomorphic encryption techniques to protect biomedical data [Rai+18] during their sharing and processing. FDC-based analyses include strategies that seek to satisfy data privacy depending on the type of information to be shared and privacy-enhancing technologies over biomedical data [Are+18]. These strategies are presented in the next chapter on the implementation of a Fully Distributed Machine Learning algorithm.

4.5. Conclusions

This chapter described our FDC approach, collaborative research endeavors that enforce confidentiality and ownership properties of biomedical data while providing means to exploit and analyze information collaboratively. FDCs take advantage of current architecture for distributed processing adaptable to multi-site biomedical analysis. The objective is to support real biomedical projects that involve the participation of geo-distributed research groups and entities. FDCs go beyond simple human solutions such as confidentiality agreements or copyrights contracts. We are looking to provide computational tools for research collaboration respecting all the technical, legal, and socioeconomic restrictions imposed on biomedical data while maintaining high computational efficiency, flexibility, and ease of use. FDC analyses offer different ways to analyze data using traditional scenarios and multi-site collaborations. Moreover, we propose the adoption of analyzes based on machine learning models through the participation of multiple sites. The construction of federated models is a trending topic due to its primary benefit of reducing the risk of sharing sensitive data. Our approach proposes workflow analysis driven by data need, where the data is considered as a first-class citizen. Data-driven features differ from the current biomedical distributed systems, especially to analyze biomedical data in geo-distributed settings. The analysis through FDC systems will enhance biomedical data processing by promoting collaborative and secure research and promoting current restrictions by sharing data, supporting the real needs in biomedical projects.

Chapter 5

Fully Distributed Random Forests (MuSiForest)

Introduction

During the last two decades, there has been remarkable growth in the usage of Machine Learning (ML) for data analyses. Machine learning analyses have notably impacted areas such as healthcare and biomedical research. For example, ML techniques have been applied to model diagnoses and prognoses of different diseases by analyzing various biomedical data sources such as genomes, images, and clinical records [LN15; YBK18]. Machine learning is particularly attractive because it uses methods such as inferential statistics, accelerated computing, and distributed computing to efficiently process and extract knowledge from large amounts of data.

However, despite the rapid evolution of ML techniques, collaboration for the exploitation of large amounts of data is still hindered by social and technical constraints. Thus, instead of exploiting parallel computations on globally distributed environments or elastically scaling resources on global cloud providers, several of the current approaches for ML rely on sharing full data sets.

During the last few years, there has been a need to adopt mechanisms to analyze data using learning algorithms and distributed computing while maintaining data at its original location and respecting data ownership constraints. Distributed machine learning has emerged as a strategy to analyze geo-distributed data [PG13; Ver+20]. In 2016, the first geo-distributed analysis proposed was Federated Learning [Kon+16a; Kon+16b] to train a collaborative model from data hosted on several mobile devices. However, current approaches do not completely support FDCs due to the inherent restrictions in biomedical data and the computational capacity required to analyze large volumes of data. Therefore, FDC-compliant methods require novel technical approaches, tools, methodologies, and collaboration strategies.

This chapter presents an approach to Fully Distributed Collaboration built on top of a Random Forest algorithm. Our FDC-compliant approach, MuSiForest (Multi-Site Random Forest), can be used to implement a collaboration of several geographically distributed research centers with their own data and their own computational infrastructure. We investigate techniques and tools to build distributed models without violating sharing restrictions, in contrast to traditional techniques that require full data access. The major contributions of this chapter are the following:

- A Fully Distributed Random Forest algorithm to analyze data collaboratively across sites while ensuring strict confidentiality and ownership properties of biomedical data.
- A software implementation of the proposed MuSiForest algorithm to build a global classification forest with the participation of multiple sites.
- Strategies for sharing partial data to reduce bias in the collaborative model.
- Experiments with the proposed algorithm analyzing labeled data corresponding to gene expression levels (RNA-Seq) to classify patient samples among five cancer types.
- Experiments using the proposed security strategies relaxing the sharing constraints to address bias correction in the collaborative model.

This chapter is structured as follows. Section 5.1, presents an overview of the construction process of models based on Random Forests. Section 5.2, describes the proposed MuSiForest distributed algorithm to train a collaboration forest. Section 5.3, presents strategies to secure bias correction. Section 5.4, details the implementation and the architecture of the prototype, and discusses some of the more common deployment topologies. Section 5.5 presents the experiments and corresponding results. Finally, Section 5.6 presents the conclusions of the chapter.

5.1. Random Forests Training

In Chapter 3, we introduced machine learning concepts to explain the process of learning from data using supervised models such as decision trees (see page 67). This section extends these concepts and explains the relevant steps necessary to train a random

forest model. A random forest model is a classifier composed of multiple decision trees. Using an ensemble of trees for the model has several benefits, namely, it is a way of reducing the risk of overfitting, we can use distributed and parallel computing to train the model, and the resulting model is highly interpretable by users because its hierarchical structure is comprehensible and its structure indicates the level of importance among the data variables.

5.1.1. Training a Model

Training a model follows the learning process illustrated in Figure 3.2 and seeks to infer a model that best represents the underlying knowledge from the labeled data set, $\mathsf{D} = \{\mathsf{X}, \mathsf{Y}\}$, where X is a set of attributes, and Y is a set of labels for each one. Usually, a proportion of records from D is used to train a model (often 70% of random samples), and the remainder to evaluate the model. Therefore, the data set D is split into training ($\mathsf{D_{TR}}$) and testing ($\mathsf{D_{TE}}$) data sets.

This section describes the construction (training) and application (test) tasks of each decision tree that makes up the forest. Finally, we present the computational complexity of training a decision tree.

5.1.1.1. Building a Decision Tree

The decision tree is built top-down recursively by partitioning the attributed spaces of the training set. Thus, the construction strategy is based on divide and conquer. The data set attributes contribute to the tree levels and the cut-off value at each node that determines the subsequent levels (see explanation in Section 3.2.2). The root node is the most important attribute, and subsequent levels are defined by decreasing levels of importance in the remaining attributes. Statistical measurements determine each attribute's importance level: the most commonly used ones are the Gini index and entropy measurements [RS04]. The decision tree construction is presented below as Algorithm 1, which follows similar strategies as those presented in [MR14].

Growing the tree

Algorithm 1 implements the idea of training a decision tree where the required data set has the form, $D = \{X, Y\} = \{(x_j, y_j)\}_{j=1}^{m}$, composed of m samples. From these data, the tree growth is based on the following steps:

Algorithm 1 Building a Decision Tree

Input: Dataset D with samples and a list of attributes (variables) A.
Output: Decision Tree

```
1: TreeGrowing(D,A)
2: if stop_condition(D) then
3:    return leaf with the most common value of class in D as a label.
4: else
5:    bestA ← best_split_attribute(D, A)
6:    tree ← new tree with root bestA
7:    for each value v_k in bestA do
8:       D_exs ← {e : e ∈ A and e.bestA = v_k}
9:       child ← TreeGrowing(D_exs, A − bestA)
10:      add child a branch to tree with label (A = v_k)
11:   end for
12: end if
13: return tree
```

The algorithm trains a decision tree. The process starts from the training data set, determining the level of importance of each attribute and resulting in a trained model, a decision tree.

1. Select the best attribute x_i in D, employing statistical measurements.

2. Define the cut-off value from x_i to divide D into two subsets (left and right branches).

3. Repeat the previous steps until the stop condition is fulfilled, i.e., the samples in the partitioned subset have the same class or no more records to be divided exist.

The main challenge of these steps is determining the best attribute to split and its cut-off value. We now present two relevant strategies for measuring the attributes and determining the cut-off value based on statistical measurements to identify the best attributes.

Splitting Attributes

Selecting the best attributes leads to identifying the tree levels, a recursively applied process until reaching the stop criteria, e.g., the nodes are purity. Purity means that all samples of the left have the same class, and the right node contains the records of the other class, i.e., a pure node groups all samples of the same class at each node. After finding the first split attribute, the data is split into two subsets, each with one or more classes. Then, the process is repeated until the nodes are pure in terms of the resulting class. The best attribute is selected through the following statistical measurements that

determine the Information Gain (IG) level of each attribute in the data set:

1. **Gini impurity** is the probability of incorrectly classifying a randomly chosen sample if it was randomly classified according to the class distribution. Given the data set $D = \{(x_1, y_1), ..., (x_m, y_m)\}$, $y_i \in classes$. For each value class c, the dataset D_c is defined as $D_c = \{(x_1, c), ..., (x_k, c)\}$, where $y_i = c$, and $D_c \subset D$. Gini impurity value is defined as:

$$Gini(D) = \sum_{j=1}^{C} p_j \Big(1 - p_j\Big), \tag{5.1}$$

where C is the number of classes present in a node; p_j is the portion of samples in D with class value j, that is, $p_j = \frac{|D_j|}{|D|}$. Assuming the dataset D is split into two subsets D_1 and D_2 according to attribute A, Gini impurity is defined as:

$$Gini_A(D) = \frac{|D_1|}{|D| \cdot \ Gini(D_1)} + \frac{|D_2|}{|D| \cdot Gini(D_2)}, \tag{5.2}$$

where $\mathsf{D} = \mathsf{D_1} \cup \mathsf{D_2}$, $\mathsf{D_1} \cap \mathsf{D_2} = \emptyset$, and $|\mathsf{D_i}|$ denotes the number of samples in $\mathsf{D_i}$.

During this document, without loss of generality, the operations $\cup$ (union) and $\cap$ (intersection) follow the traditional operations between sets, defined as:

$$A \cup B = \{x : x \in A \ \ or \ \ x \in B\} \tag{5.3a}$$

$$A \cap B = \{x : x \in A \ \ and \ \ x \in B\} \tag{5.3b}$$

The Equation 5.2 is applied recursively to the dataset's attributes until the stopping criterion is fulfilled. The selection of each node performed according to the lowest Gini value, and the root represents the attribute with the lowest value. The root is statistically the most important attribute. The process is repeated with part or all of the attributes until the decision tree is built. Algorithm 2 presents how to compute the Gini impurity value [Loh11].

2. **Entropy** determines the degree of uncertainty in a data set. Entropy is used to calculate the Information Gain (IG) to characterize the impurity of a dataset. The IG measure seeks to minimize the entropy level in the data set. During the tree construction, the attributes preferred are those with the lowest IG value. Entropy

Algorithm 2 Gini Impurity estimation

Input: The partitioned subsets D_1 and D_2 by A.
Output: Gini impurity value

```
for each subset D_i in {D_1, D_2} do
    Determine the percentage of samples in D_i
    for each class c in D_i do
        Calculate the probability of class c in D_i
        Calculate the Gini(D_i) value as in Equation 5.1
    end for
end for
Calculate Gini_A(D) following the Equation 5.2
return Gini_A(D)
```

The algorithm presents the strategy to calculate the value of the Gini impurity. The process assumes a data set D and estimates the statistical measurement for the attribute A. The process is repeated for each of the attributes in D.

is defined as

$$Entropy(D) = S(D) = -\sum_{j=1}^{C} p_j \ log_2(p_j), \tag{5.4}$$

where p_j is the portion of samples in D with class value j as defined before. The process of calculating entropy is presented in the instructions in Algorithm 3.

Algorithm 3 Entropy estimation

Input: The partitioned subsets D_1 and D_2 by A.
Output: Entropy value

```
for each subset D_i in {D_1, D_2} do
    Determine the percentage of samples in D_i
    for each class c in D_i do
        Calculate the probability of class c in D_i
        Multiply the probability in logarithmic value by (−1)
    end for
end for
Calculate Entropy(D) as the Equation 5.4
return Entropy(D)
```

The algorithm describes the strategy to calculate the Entropy value. The process assumes a dataset D and estimates the Entropy for the attribute A.

Consequently, the Information Gain uses the Entropy Equation 5.4 to be defined as:

$$IG(D, A) = S(D) - S(D|A), \tag{5.5}$$

where $\mathsf{S}(\mathsf{D}|\mathsf{A})$ corresponds to the entropy value in the data set resulting from dividing the data set D by the attribute A.

Algorithm 4 Information Gain estimation

Input: Dataset D and list of attributes A.
Output: IG value, Attribute
1: $entropy \leftarrow Entropy(D)$
2: **for each** attribute a in A **do**
3: Compute the expected information as Equation 5.5
4: **end for**
5: Find the highest IG value and the respective attribute $highest\_a$
6: **return** $highest\_IG$, $highest\_a$

The algorithm presents the strategy to calculate the Information Gain (IG) value. The process assumes a data set D and estimates the IG for the attribute A. The process is repeated for each of the attributes in D.

Algorithm 4 presents a procedure for calculating information gain value. Algorithms 3 and 4 are based on current literature on decision trees' impurity measure estimations [MR14; Has+09].

5.1.1.2. Decision Tree Application

After the tree construction, the tree is used to classify (predict) data using the trained model. The application of the model consists of predicting the classes for (in particular, testing) records to determine if the predicted class is equal to the true class. The prediction is based on the decision tree's hierarchical structure, the predicted class is reached by traversing from the root to the leaf while performing the intermediate node validations.

Figure 5.1 presents a tree trained from real data corresponding to gene expression levels for thousands of genes, almost 20 thousand attributes with a labeled class. The data is labeled and correspond to five types of cancer represented by the leaves of the tree with an integer value class between 1 and 5. The hierarchical structure provides information about the trained model. Each complete path corresponds to one conditional rule. The figure also shows relevant information for the construction of the tree. For example, information such as the Gini value, the number of records analyzed, the distribution of records by class analyzed, and the predominant class in each node. The root node corresponds to the best attribute (gen10896) with a Gini value of 0.758, from 252 of five classes with predominant class 1. Similarly, the remaining nodes present this information based on

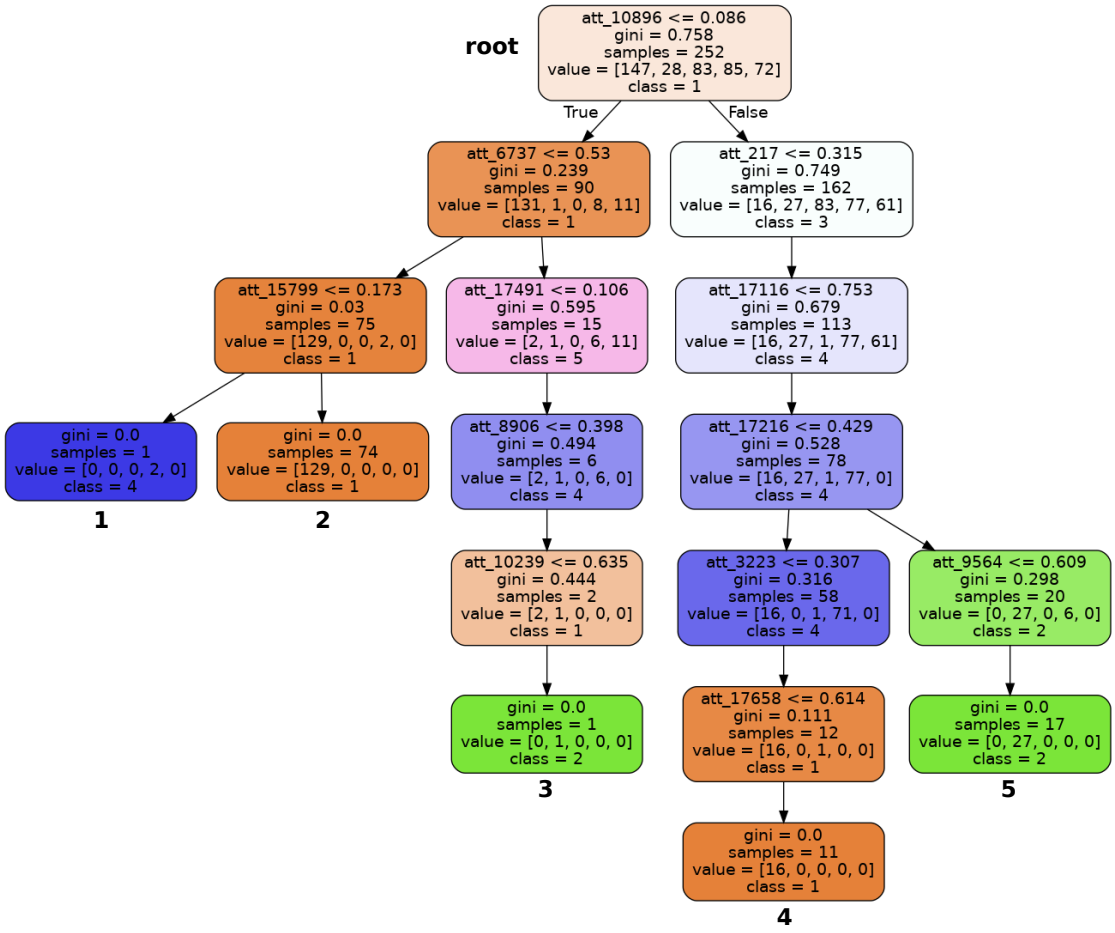


Figure 5.1 – Decision tree modeling a data training process with thousands of genes, multiple classes, and five paths from the root to leaves.

node split evaluation. The color of each node indicates the majority class of the analyzed records.

The tree application (prediction process) traverses the tree from the root to a leaf. Suppose, we are interested in classifying the sample record x into one of the classes using the constructed tree: $\mathsf{x} = (\mathsf{att\_8906} = 0.21, \mathsf{att\_6737} = 0.18, \mathsf{att\_3223} = 0.415, \mathsf{att\_15799} = 0.52, \mathsf{att\_10896} = 0.02, \mathsf{att\_217} = 0.12)$. The process starts from the root and chooses the corresponding path based on the values of each attribute of x. In this case, the root chooses the left branch until it reaches the leaf represented with path 2. Therefore, the record x is classified with a class value 1.

5.1.2. Estimating the Prediction Error

As explained in Section 3.2.2.1, a random forest is composed of multiple trees, each trained following the previously presented algorithm. In the random forest, each tree can be used independently. The forest prediction is achieved by a majority vote.

The prediction (classification) error in the Random Forest model is calculated based on the error of each decision tree. The classification error is defined as:

$$error(RF, D_{TE}) = \frac{1}{|D_{TE}|} \sum_{(x_i, y_i) \in D_{TE}} \mathbb{I}\Big(y_i \neq RF(x_i)\Big), \tag{5.6}$$

where RF is the forest, and $\mathsf{D_{TE}}$ is the testing data set. Remember that the forest is built from labeled records, the record x_i has a known class, $\mathsf{y_i}$ (a true class for each record). The value of $\mathsf{RF(x_i)}$ corresponds to the predicted value determined by most trees for the record $\mathsf{x_i}$. We refer to this value as $\hat{\mathsf{y}}_\mathsf{i}$ (a predicted class).

The main objective of training a model is that the value of $\mathsf{y_i}$ is close to or equal to the value of $\hat{\mathsf{y}}_\mathsf{i}$, for each record, $\mathsf{x_i}$, of the testing set. Thus, the classification error of the forest can be redefined as:

$$error(RF, D_{TE}) = \frac{1}{|D_{TE}|} \sum_{(x_i, y_i) \in D_{TE}} \mathbb{I}\Big(y_i \neq \hat{y}_i\Big), \tag{5.7}$$

where $\hat{\mathsf{y}}_\mathsf{i}$ is the most frequent class predicted by the trees of the forest for each record $\mathsf{x_i}$, and $|\mathsf{D_{TE}}|$ is the number of records in dataset $\mathsf{D_{TE}}$. From this classification error, it is possible to calculate the classification accuracy as: $\mathsf{classAcc} = 1 - \mathsf{error}$. However, there are different classification metrics to evaluate the model. We consider to four of them, the most popular measurements in classification models: accuracy, precision, recall, and F-score, which will be used in the experimental phase.

5.1.2.1. Metrics for Classification Models

The results of the classification models can be visualized through a matrix represented by two dimensions, "Actual" and "Prediction". The matrix rows indicate the actual class (a true class: $\mathsf{y_i}$), and the columns determine the class predicted by the model, $\hat{\mathsf{y}}_\mathsf{i}$.

Table 5.1 – Confusion matrix for a binary classification problem.

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	*TP*	*FN*
	Negative	*FP*	*TN*

Table 5.1 presents an example of a confusion matrix for a binary classification problem with only two output classes (e.g., positive and negative). The four entries are calculated based on the correct and incorrect records predicted by the model:

- *TP* (True Positive) is the number of predictions in which the model correctly predicts the positive class.
- *TN* (True Negative) is the number of predictions where the model correctly predicts the negative class.
- *FP* (False Positive) is the number of incorrect predictions, where the model incorrectly predicts the positive class when the actual class is negative.
- *FN* (False Negative) is the number of incorrect predictions, where the model incorrectly predicts the negative class when it is positive.

Based on these four values, the total number of correct predictions is given by $\mathsf{TP} + \mathsf{TN}$ and the total number of incorrect predictions by $\mathsf{FP} + \mathsf{FN}$. The matrix gives an intuitive visualization of the results of the classification model applied to testing records. Nevertheless, the confusion matrix is not a performance measure. However, classification metrics can be defined based on the confusion matrix values (an extensive discussion on these metrics is explained in [SL09]). We use four classification metrices, which are presented later in the implementation section, to evaluate the trained models.

Accuracy

Accuracy is the proportion of samples for which the model predicts the correct result. It can be defined as:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} \mathbb{I}\Big(\hat{y}_i = y_i\Big), \tag{5.8}$$

where $\mathsf{y_i}$ is the true class, and $\hat{\mathsf{y}}_\mathsf{i}$ is the predicted class for the record $\mathsf{x_i}$. The accuracy can also be defined from the confusion matrix as:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.9}$$

In contrast, the classification error can also be defined as follows using the matrix entries:

$$error = \frac{FN + FP}{TP + FP + FN + TN} \tag{5.10}$$

Precision

Precision is defined as the portion of positive samples correctly classified divided by the total number of samples classified as positive. This metric measures how many samples predicted as positive are actually positive, defined as:

$$precision = \frac{TP}{TP + FP} \tag{5.11}$$

Recall

Recall indicates the number of positive samples classified as positive by the model. The recall value is the ratio between all well-predicted positive samples divided by the total number of positives, defined as:

$$recall = \frac{TP}{TP + FN} \tag{5.12}$$

F-score

Also known as F-measure, it combines the previous metrics, Precision and Recall, into a single value. F-score is defined as:

$$F - score = 2 \times \frac{precision \times recall}{precision + recall} \tag{5.13}$$

5.1.2.2. Out-of-Bag (OOB) Error Estimation

The four classification metrics defined above evaluate the random forest and estimate a measure of the quality of the prediction of the trained model. However, it is possible to determine the error of each tree independently without the need to evaluate all trees in the forest together.

Remember that before training the forest, the Bagging (Bootstrap AGGregatING) strategy is applied to select a different resampling data set to build each tree independently (see explanation in Section 3.2.2). A bootstrapped set consists of random samples with replacements from the initial training set (D_{TR}). Approximately two-thirds of the training records are used to train each tree (inBag set), and the remainder is left Out-Of-Bag (OOB set) to evaluate each local tree independently.

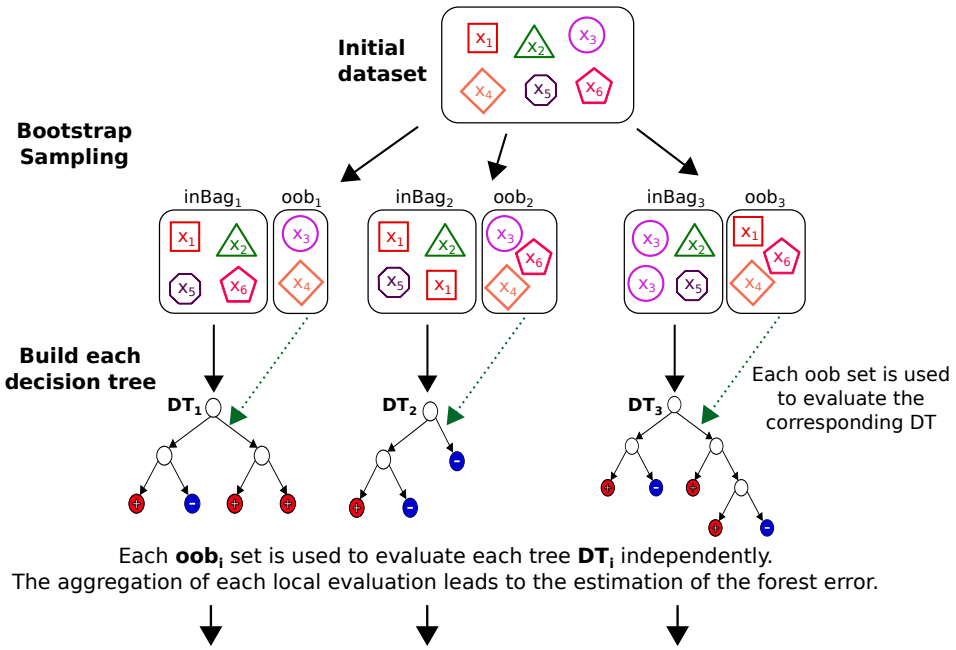


Figure 5.2 – An illustration of OOB error estimation for each decision tree

Breiman [Bre96c] proposed the out-of-bag estimation procedure to compute the ensemble tree's classification error based on two data sets, inBag and OOB. The procedure illustrated in Figure 5.2 computes the error for each tree individually. From the training data set, multiple bootstrapping samples are selected according to the number of trees in the forest. In the example, the forest has three trees, therefore, a pair of sets inBag and OOB is necessary for each tree. Each inBag set leads to training each tree, and each OOB set is a validation set for each trained tree. This OOB set works as a testing set.

To estimate the Out-of-Bag error, first, each tree predicts each sample from OOB, then the true class is compared against the predicate class, and finally, the classification quality is estimated for each tree. The training and testing process occurs independently for each tree. The sets inBag and OOB are specific to each tree. This strategy seeks to evaluate the local precision of each tree from the records not used for training each tree.

5.1.3. Complexity of Random Forests

Several parameters may influence the construction of decision trees. For example, one may tune parameters such as the measurement *impurity function* (i.e., Gini impurity or Entropy), *depth* (maximum depth of the tree), and *mtry* (the number of features to consider for the best split). The runtime complexity of the decision tree depends on these parameters. Assuming a dataset with numerical values, the runtime complexity of building a decision tree is $\mathcal{O}(n \cdot m \cdot \log(m))$, where m is the number of samples, and n is the number of attributes. The time complexity of testing a DT model is determined by the depth of the tree, $\mathcal{O}(depth)$ (number of edges from the root to the leaf node).

Random forest implementations also consider tuning parameters such as measurement impurity functions (e.g., Gini or Entropy), *depth* (maximum tree depth), and *mtry* (the number of features to consider for the best split). Remember that the complexity of training a single tree is given by the Equation $\mathcal{O}(n \cdot m \cdot \log(m))$ with m samples and n attributes. Thus, if the forest has n_t trees, then the time complexity for building the forest is $\mathcal{O}(n_t \cdot n \cdot m \cdot \log(m))$. Therefore, the time complexity applying a random forest model is given by the number of trees and the depth of the tree $\mathcal{O}(n_t \cdot depth)$.

5.2. A Fully Distributed Random Forests Algorithm

We propose a Fully Distributed Random Forests Algorithm (MuSiForest) to build a federated and collaborative model from data located in multiple sites. The proposal's main objective is to train a classification model while ensuring confidentiality and ownership properties of biomedical data.

This section presents the MuSiForest algorithm's methodology with its phases and the corresponding steps composing each phase. Furthermore, the proposed strategies to aggregate independent trees, the way to share the model, and how to apply and evaluate the collaborative forests.

5.2.1. MuSiForest Methodology

A random forest is a set of multiple trained trees where each tree predicts the class of a specific instance. Due to the forest's composition, the training of each tree can be computed in parallel, each tree can be trained from an independent bootstrapped data sample, as presented in Algorithm 5 (based on Bagging, discussed on page 70).

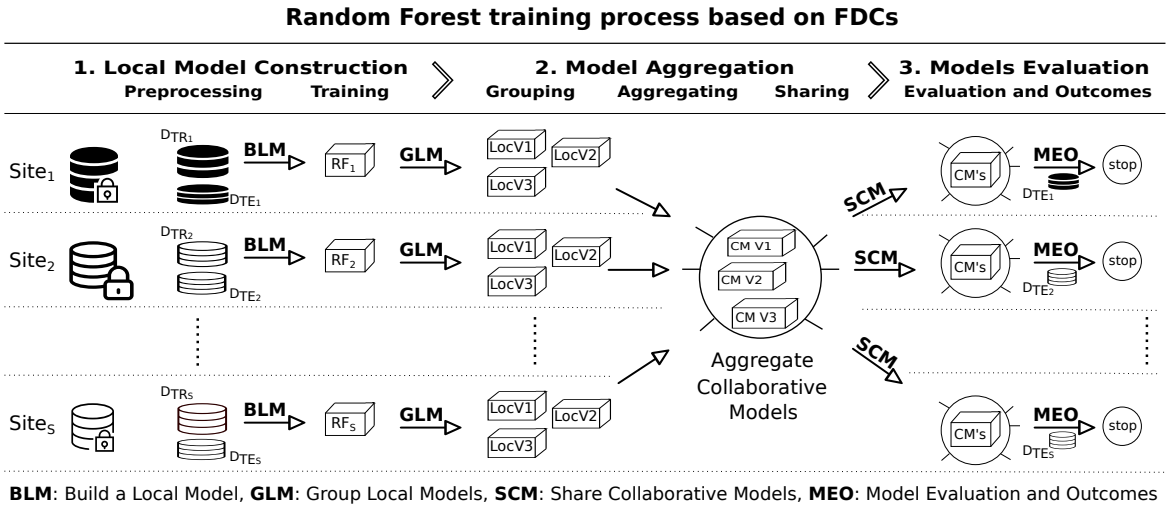


Figure 5.3 – Construction and evaluation of the RF model under Fully Distributed Collaborations. The process involves s sites. From each dataset a training ($\mathsf{D_{TR}}$) and testing ($\mathsf{D_{TE}}$) set are generated that are used on the local and collaborative models privately.

The MuSiForest methodology uses the Bagging strategy. But in a multi-site scenario, where each site builds the forest locally, we propose the following:

- First, the trees are aggregated locally using three strategies into ensemble diverse forests.
- Next, the forests are shared to be aggregated and build a collaborative forest.
- Finally, each site has a collaborative forest trained jointly and a local forest trained privately and does not collect information from the other sites.

Figure 5.3 shows the phases of the proposed methodology (upper line) and the corresponding steps (remaining part).

5.2.2. The MuSiForest Algorithm

Datasets: Assume there are s geo-distributed sites, and each has a dataset denoted as D_i, $(1 \leq i \leq s)$. Each site has a strict data-sharing policy, so they cannot share any raw data. The dataset D_i contains information for m_i samples and n_i attributes. The samples at each site are unique and private, but they may have common attributes (following the horizontal partition depicted in Figure 3.9). Consider, for example, a group of medical institutions interested in building a collaborative model to predict cancer types from gene

expression levels. Each hospital has private samples from its patients, but the samples have common information, such as the expression level for each gene in the human genome.

Retaking the collaborative case between France and Colombia introduced in Section 2.4 as an extension idea of the ICAN project. Each participating member has private information about their patients. However, the samples have data in common (attributes). In this case, the samples have common attributes, the genes, but they correspond to different patients, unique to each site.

The model construction and application process are based on traditional learning models. However, it is complemented with the proposed aggregation strategies to obtain a diverse and precise collaborative forest, mainly by:

- Construction of the local forest,
- Aggregation of trees to produce diverse forests, and
- Evaluation of local and collaborative forests.

These phases belong to the proposed algorithm, MuSiForest, and are detailed below, along with the steps to meet the desired objective in each phase.

5.2.2.1. Local Model Construction

The first phase of the algorithm, the construction of the local forest, follows two steps, the traditional training of the forest, next the evaluation of local data.

Step1: Model Training

The dataset has the form $\mathsf{D_i} = \{(\mathsf{x_j^i}, \mathsf{y_j^i})\}_{\mathsf{j=1}}^{\mathsf{m_i}} = \{(\mathsf{x_1^i}, \mathsf{y_1^i}), ..., (\mathsf{x_{m_i}^i}, \mathsf{y_{m_i}^i})\}$ on each site i, where X_j^i is a vector of attributes, and Y_j^i is the vector class (target variable) for the j^{th} record. Two new sets are generated from this data set D_i, training ($\mathsf{D_{TR}}$) and testing ($\mathsf{D_{TE}}$) sets, following the explained above (see section 5.1.1).

Algorithm 5 presents the instructions for training each tree in the the forest. The algorithm follows the traditional training in learning models [MR14]. The data partition between training and testing is presented in line 1 in the function *split_into_training_testing*. The bagging strategy generates a bootstrapped dataset (bstrap) from $\mathsf{D_{TR}}$ to build each tree, it is implemented in line 4, and the remainder is an Out-Of-Bag set (OOB) used to evaluate each tree individually. The evaluation of each tree is using the OOB set to determine an error value for each one. Line 6 computes the internal error rate for each tree

Algorithm 5 Model Training

Input: Dataset D_i composed of m_i labeled samples $\{(x_1, y_1), ..., (x_m, y_m)\}$, $ntree$: number of trees, $feat$: number of features, $depth$: maximum depth reaches, $mtry$: number of attributes considered at each split

Output: RF Model, $\mathsf{D_{TE}}$

1: $D_{TR},\ D_{TE} \leftarrow split\_into\_training\_testing(D_i)$
2: $RF \leftarrow empty\_set$
3: **do in parallel** {for each tree in RF }
4: $\quad bstrap, oob \leftarrow generate\_bootstrap\_sample(D_{TR})$
5: $\quad tree_j \leftarrow build\_DT(bstrap, feat, mtry, depth)$
6: $\quad oobError_j \leftarrow tree\_prediction\_error(tree_j, oob)$
7: $\quad RF \leftarrow RF \cup (tree_j,\ oobError_j)$
8: **end**
9: **return** $RF,\ D_{TE}$

The algorithm presents the instructions to train the forest. The input required is the dataset, number of trees, number of attributes, maximum depth of each tree, and number of attributes to consider at each split. The algorithm returns the trained forest and the testing set for the subsequent evaluation of the models.

(oobError). The oobError is calculated using the dataset bstrap, and the trained tree following the explained in the Out-Of-Bag procedure (see subsection 5.1.2.2). The algorithm ends at line 7, appending information from each tree, where RF represents the forest. The union operation, $\cup$, follows the traditional on sets, defined in Equation 5.3a. Due to the design of the algorithm, this is highly parallelizable during the construction of the forest. Therefore, the construction process can take advantage of the computing facilities at each site by efficiently exploiting distributed resources on site. In summary, the local forest training consists of:

(I) Generating a training dataset $\mathsf{D_{TR}}$, by choosing a percentage of random samples from the local dataset D_i,

(II) Resampling n different bootstrap samples, bstrap, to train each tree separately from the training dataset $\mathsf{D_{TR}}$,

(III) Training each tree to find a function RF that better maps the input values, bstrap, to the class.

At the end of the execution of these steps, the forest RF is composed of trees denoted as $\mathsf{RF_i} = \{\mathsf{t_1}, \mathsf{t_2}, \ldots, \mathsf{t_n}\}$ composed of with n trees for the site i. The algorithm finally returns

the testing set, $\mathsf{D_{TE}}$, that will be used to evaluate all random forest models (local and collaborative), using the same data set to obtain comparable metrics.

Step2: Model Evaluation

The testing dataset, $\mathsf{D_{TE}}$, is used to evaluate the trained forest, where $D_{TE} \subseteq D_i$, thus, the elements have the same structure $\{X_j^i, Y_j^i\}$. The n trees of the forest are used to evaluate the classification error of the trained model. Remember that the classification for each test sample is done by consensus among all trees (see Section 5.1.2 for more details). This process is done for each trained forest, both for the locally trained model and the collaborative versions. The details about the construction of the collaborative versions is explained later. In summary, the evaluation of the trained model $RF = \{t_1, t_2, \ldots, t_n\}$ consists of:

(I) The n trees of the forest RF are applied on each record $\mathsf{x_i} \in \mathsf{D_{TE}}$ to predict n classes, $\{\hat{\mathrm{Y}}_1, \hat{\mathrm{Y}}_2, ..., \hat{\mathrm{Y}}_n\}$, one for each record x_i.

(II) From these classes, the majority class, $\hat{\mathrm{Y}}_i$, is computed to determine the predicted value for the record x_i.

(III) The classification error is computed based on the predicted classes $\hat{\mathrm{Y}}_i$, the computation is based on Equation 3.1.

The two previous steps, model training and evaluation, lead to reaching the first phase of the MuSiForest scenario. The construction of the forest from the training set and evaluation process using a testing set of the forest at each site. These steps are applied independently on each site on the local data without sharing any information.

5.2.2.2. Model Aggregation

The construction of the collaborative model groups the best locally trained trees based on the $\mathsf{oobError}$ of each tree at each site (see Out-of-Bag error estimation procedure explained on page 111). The proposed approach to aggregating local trees addresses five relevant aspects:

- Building an ensemble model via site collaboration.
- Having a classification scoring similar to other federated proposals.
- Minimizing data leakage risks by sharing models and not raw data.

- Taking advantage of the computational facilities at each site.
- Reducing the amount of information transferred between the parties.

The model aggregation process consists of three steps. The first step groups the best local trees by employing three strategies. The second shares the grouped models with other sites. Lastly, the generation of collaborative models by aggregating the trees in a global forest. The generation of collaborative models leads to three versions according to the strategies of the first step. The details of the second phase steps are illustrated in Figure 5.3 (on page 114) and explained below.

Step1: Grouping Trees

After the first phase, the local construction of each forest, each site i has the forest $\mathsf{RF_i} = \{\mathsf{t_1}, \mathsf{t_2}, \ldots, \mathsf{t_n}\}$. The local aggregation groups the best locally trained trees through three strategies based on the local evaluation of each tree using the $\mathsf{oobError}$ calculated in line 6 of Algorithm 5. Based on this evaluation metric, the three aggregation strategies are: joining without distinction (aggregating all local trees), choosing a percentage of the best trees, and those whose precision is higher than the median accuracy value.

1. *Joining without distinction:* the trees are aggregated together following the natural idea of the random forest algorithm. The aggregation groups all local trees following Breiman's approach to produce the $\mathsf{RF_i}$ model at each site.

2. *Joining by percentage:* the trees have associated the values $\mathsf{oobError}$ and $\mathsf{classAcc}$ calculated during the construction phase. This strategy selects a percentage of the best local trees of each site by its $\mathsf{classAcc}$ value. We leave out the worst trees for the individual $\mathsf{oobError}$ of each one. We denote this set of trees in site i as $\mathsf{Best_{Per}Trees_i}$.

3. *Joining by median value:* this strategy selects those whose precision is higher than the forest's median classification value. We use the median value since it represents the best value of central data distribution, especially when the distribution is skewed and has outliers. The forest built with this strategy is denoted as $\mathsf{Best_{Mdn}Trees_i}$ for site i.

At the end of this grouping step, the local trees on each site have been grouped by three strategies: RF, $\mathsf{BestPerc}$ and $\mathsf{BestMdn}$. They are, respectively, denoted as $\mathsf{LocV1}$, $\mathsf{LocV2}$, and $\mathsf{LocV3}$ in Figure 5.3 on page 114. Algorithm 6 presents the strategies implemented to generate three forest versions on each site.

Algorithm 6 Model Aggregation

Input: perc: percentage of best trees, Random Forest RF, $\mathsf{RF} = \{(\mathsf{tree_j}, \mathsf{oobError_j})\}$, where $1 \leq \mathsf{j} \leq \mathsf{NoSites}$.
Output: Three aggregated forests from local trees

```
1: medianclassAcc ← get_median_accuracy_trees(RF)
2: threshold ← get_threshold_best_perc_trees(RF)
3: BestPerc, BestMdn ← empty_sets
4: for j ← 1 to ‖RF‖ do
5:    classAcc ← (1 − oobError_j)
6:    if (classAcc ≥ threshold) then
7:       BestPerc ← BestPerc ∪ {tree_j} // ∪ operator defined in 5.3a
8:    end if
9:    if (classAcc ≥ medianclassAcc) then
10:      BestMdn ← BestMdn ∪ {tree_j}
11:   end if
12: end for
13: return RF, BestPerc, BestMdn
```

From the trained forest composed of trees and the classification error of each one, three versions of the forest are aggregated (RF, BestPerc, BestMdn), leaving the worst trees out of the collaborative models.

Step2. Sharing the Grouped Models

The MuSiForest algorithm shares models from multiple sites to the aggregating site. The location of the aggregator depends on the deployed architecture and is detailed in the next section. The selection of the aggregator depends on the desired level of distribution and conditions of the sites, such as:

(I) One site may have better computer capacity installed than others. The site configurations can be heterogeneous due to processing and communication capacity. This ability allows dynamically assigning the aggregator site like the one with the best computing facilities.

(II) A site equidistant from others. Grouping sites can favor aggregation due to their geographical location to improve transmission. The aggregator site can be the one most central to all in terms of channel capacity.

(III) By agreements between some sites to share information responsibly. The aggregator may be according to any specific region, respecting the regional data-sharing restrictions.

Under these considerations, the architecture is determined and, therefore, the number of aggregating sites. The shared information corresponds to trained models whose structure is the hierarchical structure represented in decision trees. The model's structure corresponds to the tree's levels by the nodes and the conditions in each of the branches. The tree level are represented from the root (most significant attribute) to the leaves (dataset classes). The information share in trained models do not reveal sensitive patient data [Yin+11]. The data privacy of patient samples is preserved because it is not part of the trained model. Similarly, if we consider sharing statistical information during tree construction, such as sum and average, it does not compromise sensitive patient information [AS00].

Step3: Aggregating the Collaborative Model

The responsible site for receiving the models also aggregates them to build the collaborative model. The aggregator site receives three forests from each site: $\mathsf{RF_i}$, and $\mathsf{Best_{Per}Trees_i}$, and $\mathsf{Best_{Mdn}Trees_i}$. These models are sets of individual trees. Next, it creates three different collaborative models joining the forests of each type:

- $\mathsf{CMJoin} = \bigcup_{i=1}^{s} \mathsf{F_i}$, is a large forest composed of the forests for s sites.
- $\mathsf{CMPerc} = \bigcup_{i=1}^{s} \mathsf{Best_{Per}Trees_i}$,
- $\mathsf{CMMdian} = \bigcup_{i=1}^{s} \mathsf{Best_{Mdn}Trees_i}$.

Algorithm 7 presents the generation of these collaborative models. The algorithm is executed on the aggregation site depending on the deployed architecture. Figure 5.3 (on page 114) illustrates these three collaborative versions, such as CMJoin, CMPerc, and CMMdian.

The idea of aggregating local trees by the individual level of precision is presented in [CJL07]. They discard some of the worst trees to build a model called *Trimmed Bagging*. Based on experimental results, they show better precision than the traditional forest but in a local site, without collaboration with others. We apply this idea of selecting the best local trees on each site to next aggregate into collaborative forests. We seek an optimal classification close to the central one but only share knowledge inherent to data location without risking data confidentiality. Finally, the aggregator site returns the collaborative models to each site to be evaluated and applied to the local data to complete the aggregation step.

Algorithm 7 Generate Collaborative RF

Input: Set of RF models $\{(\mathsf{RF_j}, \mathsf{BestPerc_j}, \mathsf{BestMdn_j})\}$, where $1 \leq \mathsf{j} \leq \mathsf{NoSites}$
Output: Collaborative RF models

```
1: CMJoin, CMPerc, CMMdian ← empty_sets
2: for i ← 1 to NoSites do
3:     CMJoin ← CMJoin ∪ {RF_i} // ∪ operator defined in 5.3a
4:     CMPerc ← CMPerc ∪ {BestPerc_i}
5:     CMMdian ← CMMdian ∪ {BestMdn_i}
6: end for
7: sendToAll CMJoin, CMPerc, CMMdian
```

The aggregator site makes the aggregate from the models aggregated and shared by each site to form three forest models from the three proposed strategies. In the end, the aggregator sends the three collaborative forests to each site: CMJoin, CMPerc, CMMdian.

5.2.2.3. Evaluating the Collaborative Model

At this point, each site has received three collaborative forests from the aggregation site. Therefore, each site has four models, one built locally, $\mathsf{RF_i}$, and three by collaboration: CMJoin, CMPerc, and CMMdian. Next, the four models are evaluated separately on the testing dataset $\mathsf{D_{TE_i}}$. The evaluation follows the process explained in the section on model evaluation, see Step3 in 5.2.2.1.

Discussion

The MuSiForest algorithm comprises three main phases: local construction, aggregation models, and application and evaluation of each model. Due to the nature of training an ensemble model as a random forest, the three phases are highly parallelizable by its design computing on distributed infrastructures at each site. Moreover, our strategies for aggregating the collaborative models are also highly parallelizable through the multi-site distribution, according to the steps presented in the algorithms discussed above. The experimentation and results address the challenges presented at the beginning of the chapter (listed on page 77). We show favorability in each of them by implementing the MuSiForest algorithm to analyze geographically distributed data; as a first scenario, we train a collaborative forest to classify and predict samples.

5.3. Privacy-Preserving Bias Correction

Bias in machine learning is a phenomenon that affects the training process, as explained previously (see Sec 3.2.1). Bias leads to inaccurate models due to incorrect data assumptions during training. These assumptions can be caused by data with a dominant class, which induces a bias in the learned model towards such classes. Moreover, the training can lead to bias due to the characteristics of the geographical location of the patients. Consequently, a multi-site analysis can be affected by these two reasons. Some sites have majority classes and others particular values due to the geographical location of the samples. The bias is evidenced in the experimental results by achieving less accurate models between more distributed sites than the centralized model over all data, as will be shown later.

This section presents three strategies for bias correction induced by each site's data in the collaborative model. These strategies seek a balance between the collaborative model's privacy and accuracy. The strategies consider sharing partial data securely by implementing known techniques for the collaborative model based on privacy-enhancing technologies over biomedical data [Are+18].

However, there is a paradox behind a collaborative model: collaboration leads to diversity by gathering knowledge from all sites and promoting data security and confidentiality. The collaborative model can also induce biases that affect the prediction of the results in contrast to the traditional centralized training.

The three bias-correction strategies are:

1. The first strategy is based on secure data containers. It considers a global data set referencing the samples at each site, but the samples are stored private at each site. The idea is to generate and collect resampling samples of the global references from each site's records and then follow the bagging strategy to train the global forest.

2. The second strategy considers the collaborative construction of each tree, level by level, through the aggregator. The construction of each tree requires information gain, a statistical measurement of each set of attributes locally. This strategy considers two sharing approaches: a trusted aggregator without encrypting the values and an untrusted aggregator requiring homomorphic encryption operations.

3. Finally, the third strategy combines the misclassified samples at each site during the evaluation phase. These samples are securely shared with neighbors' sites via

public and private keys. These samples are part of the validation data set to be safely shared with others. The goal of sharing a validation dataset is to extend the diversity of the global forest by aggregating misclassified samples. This strategy is based on the idea of privacy-preserving classification proposed by Raphael et al. [Bos+14].

These strategies seek aim at sharing partial data while preserving the privacy and confidentiality of the information.

5.3.1. Secure Data Containers (SDCs)

The first proposed strategy is named Secure Data Containers (SDC) due to sharing securely partial data during the collaborative training. The secure data container is a means for securing data and learning models that are shared with others. The aggregator site has information about the number of records for each site, which are kept privately stored in each one. Further, the aggregator has a consolidated data set with pointers (references) to identifiers of the records of each site. The identifiers can be consecutive numbers, but they are not sensitive to patient information. The consolidated data set stores each pointer referenced to each site's sample identifiers. The objective of this consolidated set is to apply the bagging strategy to determine the random sampling of pointers and then safely share each site with these samples that will comprise the training set. Figure 5.4 represents the aggregator site with the consolidated set with only the pointers to each site's records.

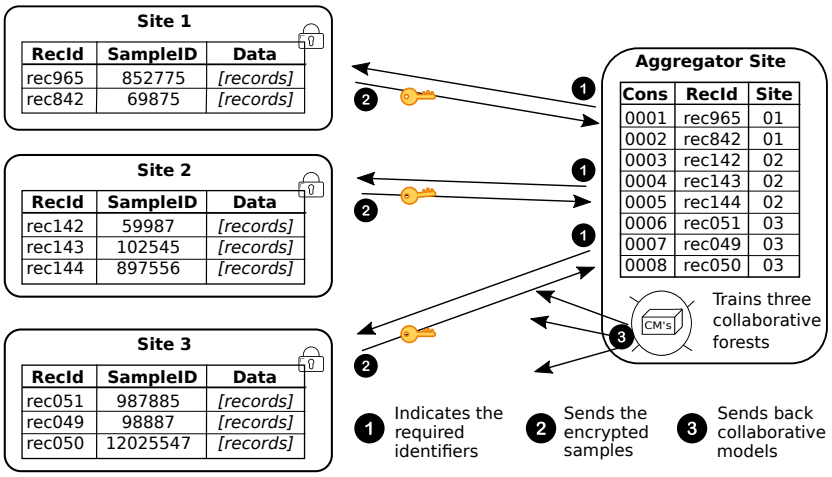


Figure 5.4 – Secure Data Container strategy to share partial data securely.

This bias-correction strategy is illustrated in Figure 5.4 and follows three steps:

1. The aggregator site stores all records to each site's identifiers *(RecId)*. Identifiers are unique values to identify each site sample, but they are not sensitive data values. From this set, the aggregator chooses a random sample of references using the bagging strategy to build a set of random references pointing to each site's identifier records. Then, based on this set of references, the aggregator site sends the required records, *RecId*, to each indicated site.

2. After receiving the list of identifiers required by the aggregator, each site generates a data set with the required samples from such identifiers. Then, each site packages the data required in a container and, applying some encryption techniques, gets a secure container to be shared with the aggregator. The container is encapsulated based on biomedical encryption techniques such as those presented in [Azi+19; Are+18].

3. Finally, after receiving the secure data containers from each site, the aggregator is responsible for aggregating the consolidated data set from the samples received from each site. For this, the aggregator must decrypt the samples to generate that will be the training set of the collaborative forests. Then, the aggregator trains the forest using the three strategies of the MuSiForest algorithm. In the end, it sends back the three collaborative forests to apply privately to all data at each site.

After applying the SDC strategy, each site has four models, the one built locally and the three collaborative versions shared by the aggregator. The encrypted data shared by the aggregator does not correspond to the total records. It is only a portion of the records according to the bagging technique.

The Secure Data Container strategy can reduce the model's bias by having a higher representation of the data distribution for each site. The collaborative models better generalize the sample population from the sites through statistical sampling mechanisms such as bagging. Moreover, the models increase the diversity compared to the version trained on each site. One site is limited to knowing the data of another site, but this is exclusive to the aggregator site, which accesses the partial records securely.

In summary, the SDC strategy can reduce the bias of the collaborative model due to the greater availability of data to train the forest at a single site. However, this strategy requires more flexibility in terms of security between the sites to ensure the sharing of containers.

5.3.2. Secure Tree Data (STD)

The second strategy considers the collaborative construction of each tree in the forest among all the sites. Remember that each tree has a hierarchical structure comprised of levels from the root to the leaves. The tree hierarchy is defined based on statistical measurements to determine the gain of information on the attributes of each site, following the explained in Section 5.1.1.1. This second strategy, STD, involves two approaches to sharing data. The first approach is to encrypt the data with Fully Homomorphic Encryption (FHE) techniques on each site before sharing data. The encrypted data corresponds to the statistical measurement, assuming the non-existence of a trusted third party. While the second is to share the values without encryption, assuming the existence of a trusted site. These two approaches increase the sharing scenarios depending on the level of security offered and the computing facilities of each site that does collaborative analysis.

FHE-based Sharing

The FHE-based approach applies when there is no trusted third-party site to share data. For instance, the aggregator is not part of the collaborating members, but it can be used for its high computing capacity appropriate for the analysis. Encryption based on fully homomorphic techniques supports mathematical operations such as addition, multiplication, and comparing encrypted data [Van+10]. However, homomorphic operations are computationally-intensive, decreasing the performance. Nevertheless, the FHE-based approach can be considered to share encrypted numerical values with the non-existence of a trusted site.

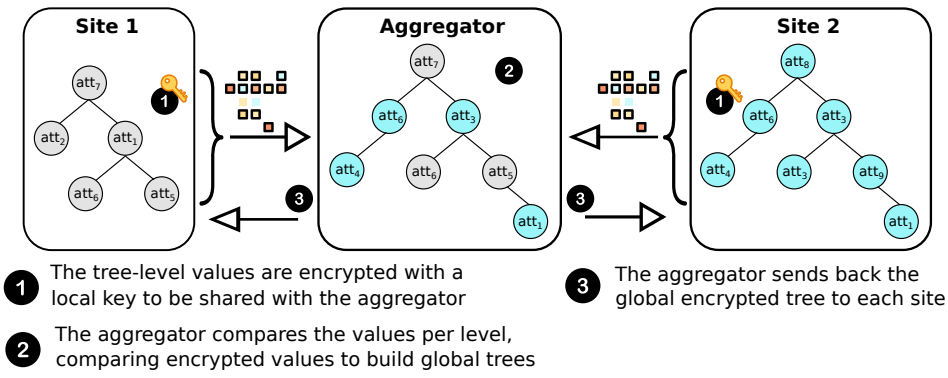


Figure 5.5 – Secure Tree Data strategy based on FHE to build each level tree securely.

This approach applies FHE techniques based on other homomorphic techniques (as

presented in [Aka+22]) based on the construction of each tree in the forest described in 5.1.1.1. The main idea is to share encrypted statistical values with the aggregator (untrusted site) to build each tree level by level. Then, the aggregator builds each tree from the statistical information shared by each attribute (identifier, gain level, and split value). This strategy is illustrated in Figure 5.5, and the idea is summarized in the following steps.

1. Assume that two sites want to build a collaborative model, tree by tree. Each site shares tree-level values by training each tree from local data. The tree-level values, Information Gain (IG), result from the splitting procedure (see explanation in 5.1.1.1). The values resulting are the most important attributes and their splitting values. This information is represented by the set $\mathsf{IG}^{\mathsf{s}} = \{\mathsf{att}_1, \ldots, \mathsf{att}_\mathsf{p}\}$ that represents the information for p attributes in the site s. Each one of the elements of IG $(\mathsf{att}_\mathsf{t})$ contains the identifier (id_t), the Information Gain (ig_t), and the split value (cv_t) for the attribute t.

   First, each site creates the respective data set, IG^1 and IG^2, for its local data set. Next, create an encrypted version from these values using a local key to be shared with the aggregator.

2. The aggregator receives the encrypted values from two sites and compares the gain values ig_t^1 and ig_t^2, selecting the major value. This value corresponds to each level of the global tree. This procedure is applied until it reaches each tree's depth.

3. Finally, the aggregator sends the global encrypted tree back to each site. The previous process is repeated, and the collaboration finishes until the forest is complete.

As already mentioned, this approach assumes data sharing with an untrusted site. Therefore, the data shared is encrypted. However, this strategy can increase the latency time between the sites because the collaboration is done level by level for each tree. Moreover, FHE techniques are subject to some limitations in handling large numbers because of the computational effort required by the homomorphic evaluation [ZZK16].

TrustedSite-based Sharing

This strategy follows the explanation of the previous part, but it is applied to cases when the aggregator is trusted. Therefore, this scenario does not consider encrypting the

data to be shared. Instead, the process follows the collaborative construction of each tree level, sharing information about the statistical gain from each site (as explained in Section 5.1.1.1). The sites only share unencrypted information about the hierarchical structure of the trees. The strategy does not consider sharing raw data. The idea is the same as illustrated in Figure 5.5, but removing the encryption process on each site because the aggregator is a trusted site. In the end, The aggregator site has a global model composed of each tree to be sent back to each site.

5.3.3. Secure Validation Dataset (SVD)

The last strategy, SVD, seeks to create a global validation data set from the misclassified samples in each site forest. This global validation set will be the training set to apply the MuSiForest algorithm. The idea of having a global validation dataset aims to gather misclassified samples to have a dataset with more knowledge from data from all sites. Gathering this diverse knowledge will be a way to correct the bias of the collaborative model. The SVD strategy is based on the idea of privacy-preserving classification proposed by Raphael et al. [Bos+14].

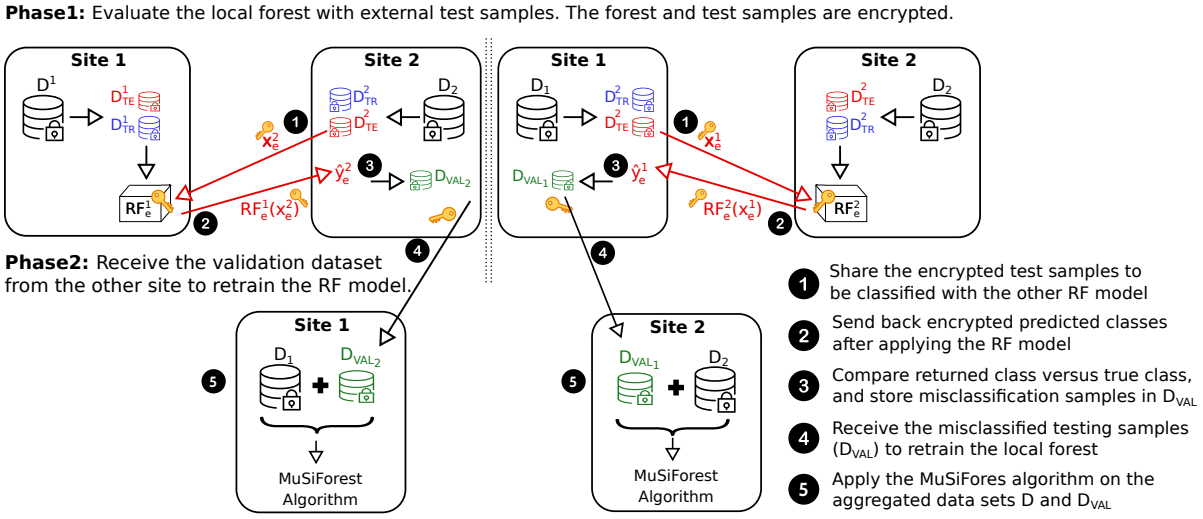


Figure 5.6 – Bias-correction strategy based on a secure global validation dataset.

This strategy is illustrated in Figure 5.6, composed of two main phases, evaluating the local forest with external test samples and receiving the validation dataset from the other site to retrain the models. Assuming that the strategy is applied between two sites (site_1 and site_2), these two phases proceed as follows:

1. In the first step, site_1 receives from site_2 the encrypted testing samples $\mathsf{D}^2_{\mathsf{TE}}$. With these samples, site_1 predicts with the encrypted forest $\mathsf{RF}^1_{\mathsf{e}}$ each testing sample $\mathsf{x}^2_{\mathsf{e}}$ from site_2 to obtain $\hat{\mathsf{y}}^2_{\mathsf{e}}$. Similarly, site_2 receives the encrypted testing samples $\mathsf{x}^1_{\mathsf{e}}$ from site_1 to apply the local forest $\mathsf{RF}^2_{\mathsf{e}}$. The encryption and privacy classification is based on Raphael et al. [Bos+14].

   At this point, each site obtains predicted classes encrypted with the opposite RF on its testing samples. For instance, site_1 has the predicted classes for site_2, such that, $\mathsf{RF}^1_{\mathsf{e}}(\mathsf{x}^2_{\mathsf{e}}) = \hat{\mathsf{y}}^2_{\mathsf{e}}$, and vice-versa, site_2 has the predicted classes for site_1, as $\mathsf{RF}^2_{\mathsf{e}}(\mathsf{x}^1_{\mathsf{e}}) = \hat{\mathsf{y}}^1_{\mathsf{e}}$.

2. Each site sends the predicted classes to the respective site. For instance, site_1 privately receives the predicted classes $\hat{\mathsf{y}}^1_{\mathsf{e}}$ from site_2, and site_2 sends $\hat{\mathsf{y}}^1_{\mathsf{e}}$ to site_1. The two sites keep the model and samples encrypted; neither site learns anything from the other because the RF model and samples are always encrypted.

3. On each site, the predicted class is compared against the true class of each record to identify misclassified records. The set of misclassified records constitutes a validation dataset. For instance, $\mathsf{D}_{\mathsf{VAL}_1}$ is shared encrypted with site_2, and $\mathsf{D}_{\mathsf{VAL}_2}$ with site_1.

4. Next, site_1 receives the validation set from site_2, $\mathsf{D}_{\mathsf{VAL}_2}$, and site_2 receives the $\mathsf{D}_{\mathsf{VAL}_1}$ dataset from site_1. Each validation set comprises samples from the other site misclassified by each local forest, where the local model was wrong in those predictions.

5. Finally, site_1 expands the initial dataset D_1 with $\mathsf{D}_{\mathsf{VAL}_2}$. This new dataset is used to train the collaborative model based on the MuSiForest algorithm. Similarly, datasets D_2 and $\mathsf{D}_{\mathsf{VAL}_1}$ are joined for use in site_2. The local data and validation set from the other site are combined privately to train the forest following our approach illustrated in Figure 5.3 (on page 114).

The bias-correction strategy, SVD, seeks to improve model bias by extending the original dataset with partial data from other sites. The diversity in the new training dataset will lead to an unbiased model by collecting more samples from other sites extending the local knowledge of the samples with data from other geo-located patients. Moreover, the third strategy uses a recent method for machine learning classification over encrypted data, a promising area in the coming years to use global encrypted models without moving data to predict unknown records, such as machine learning as a service [Bos+14; GLN13; Fri+18].

Discussion

The fairness[1] of model training is an open question in Federated Learning [Kai+21]. The models must meet current regulations for data sharing and improve fairness and accuracy without having access to private data at each site. Nowadays, few approaches have been proposed to build fairness models. For example, agnostic federated learning [MSS19] attempts to train a centralized model optimized for any possible target distribution conformed by a mixture of client distributions. PATE [Pap+18b] combines multiple trained models applying DP techniques by introducing controlled noise into the data. Federated fairness is an open field of study. One starting point is to validate whether the centralized bias correction methods apply to geo-distributed environments and determine each method's assumptions. Regarding learning model bias, there is a trade-off between privacy, fairness, and precision, where the local models offer guarantees to comply with these features, limiting collaborative efforts. In our experiments, we implement one of the three bias correction techniques proposed and show how the classification scoring of the model improved sharing of partial data securely.

These three scenarios are applied to realistic projects analyzing biomedical data, such as the proposed collaboration between Colombia and France (see Sec. 2.4). These collaborative scenarios will improve the model's scoring by reducing the bias by collecting more information specific to the site of each patient through one of the three strategies described. Furthermore, the three techniques enrich the MuSiForest algorithm through encryption, sharing, and data processing techniques. These bias-correction strategies enrich this type of international cooperation favoring privacy in biomedical data and promoting collaborative research.

5.4. Architecture and Implementation

The MuSiForest engine's architecture aims for high scalability and flexibility despite the workload distribution and heterogeneity properties for the processing among multiple participants. Figure 5.7 presents the architecture with the components installed in each site and their interactions. The interaction between these components supports the deployment in three distributed processing topologies: sequential, hierarchical, and fully distributed. These topologies are based on the architecture described in FDCs (see Section

1. We refer to fairness as the absence of generalization about the training data and the model's preference for a particular set of features.

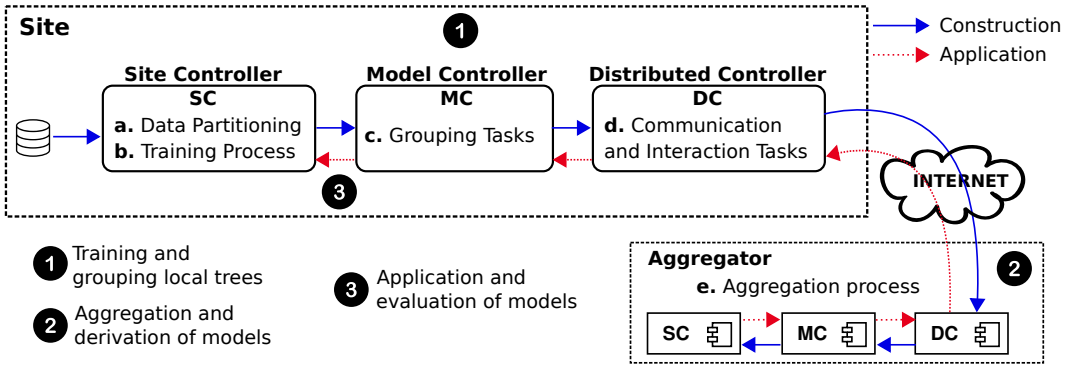


Figure 5.7 – Architecture of the MuSiForest engine.

4.3).

As shown in the Figure, the architecture includes three components (site, model, and distribution controller) that are part of each site and the aggregator site. These components are installed on each site, and it is possible to configure aggregator roles before deploying a multi-site analysis. The execution of these components is based on data-driven demand as an advantage of our FDC approach. Therefore, these components are independent at each site and are adjusted to each configuration and computational capacity.

These tasks are presented in three main processes illustrated in Figure 5.7. They interact through two processes: models' construction (blue arrow) and their application (red arrow) over data from each site. The deployment of these components supports the MuSiForest algorithm execution offering a flexible processing architecture adjusted to heterogeneous conditions of a multi-site analysis, such as the collaborative scenario between France and Colombia.

5.4.1. Component-based Architecture

The FDC architecture uses independent components between functional and logical modules based on data-driven demand. The architecture components contribute to modular, reusable, and extensible deployment scenarios. In our system, the FDC architecture supports these properties, considering three specific controllers:

- *The Site Controller* (SC) is responsible for local tasks such as building local trees and grouping them by the three aggregation strategies proposed (RF, BestPerc, BestMdn). Furthermore, the SC is exclusively responsible for accessing the data to

build the models at each site. Finally, it is responsible for applying the models (local and collaborative versions) to obtain the evaluation and the results over local data.

- The *Model Controller* (MC) oversees the aggregation of local models during the ensemble process. Furthermore, it aggregates the local models with those received from other sites. The MC is an intermediary between the SC and the following component, Distribution Controller (DC), responsible for interacting with others.

- The *Distribution Controller* (DC) is responsible for distributing and communicating with other sites. It also ensures the exchange of local aggregated models and receives the collaborative models. In addition, the DC controller is responsible for the interaction for data sharing between the sites. The models that others share are received by the DC controller and delivered to the MC component.

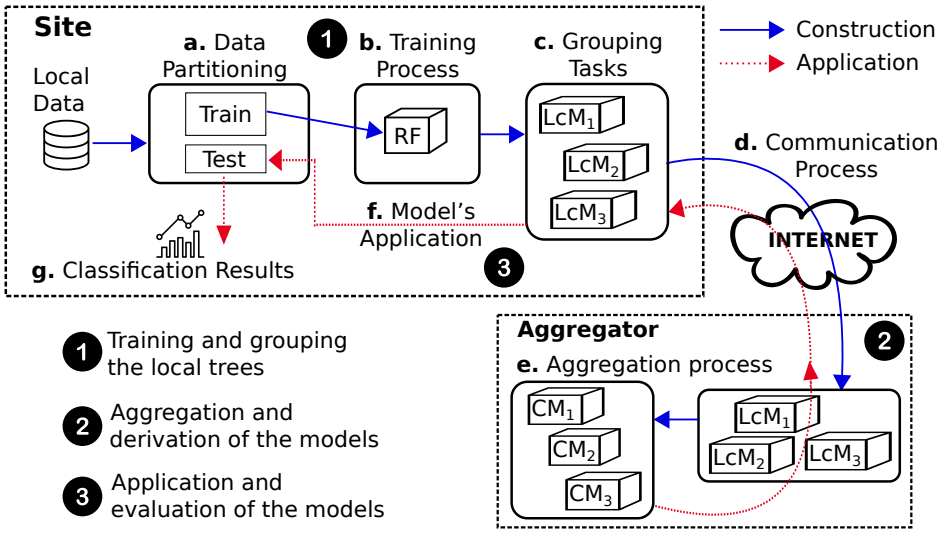


Figure 5.8 – Execution Architecture the MuSiForest algorithm.

Figure 5.8 complements the architecture diagram presenting the execution flow between the system components. The diagram illustrates the execution architecture of the MuSiForest algorithm. The figure details the architecture components with two execution flow between the site and the aggregator. The first flow is in charge of the model's local and global construction process, while the second sends back the collaborative models and ensures the application of these on local data. In Figure 5.8, the three main processes are detailed in tasks (numbered from a. to g.) delegated to each controller. These controllers

communicate through interfaces ensuring interoperability between them. The component-based architecture of our system provides a higher level of abstraction appropriate for the distribution level considered in the FDC architectures.

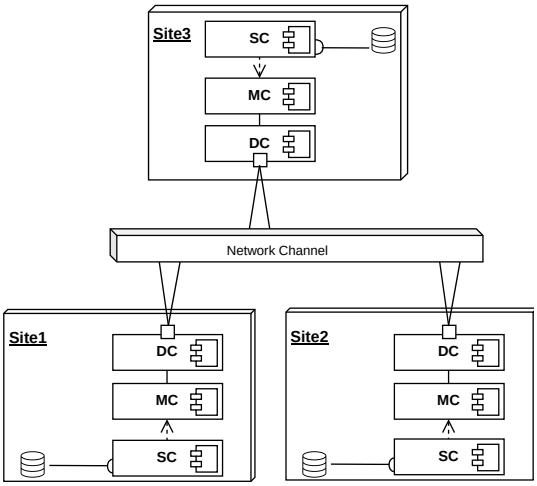


Figure 5.9 – Diagram of deployment components among three sites.

Each of the three controllers makes up a processing site of the multi-site system, and their integration permits the execution of the MuSiForest algorithm. Figure 5.9 shows the installation at each site, where they all have the same components communicated through a channel. The Figure shows an abstraction of a deployment scenario with three sites (the deployment model applies to N sites without loss of generality). The sites collaborate through the interface provided by the distributed controller to build a shared model. The architecture may involve a trusted communication channel between the parties. However, the shared data can also be secured using the techniques described in Section 5.3. The communication can be done differently depending on the desired deployment model, such as using point-to-point or one-to-many connections.

5.4.2. System Implementation

Our system is implemented through *Python 3.7* in well-defined modules packaged according to the responsibility of each architecture component. The components incorporate specialized libraries according to the tasks of each controller. For instance, the site controller tasks extend the *Scikit-learn*[2] library by our native implementation to incorporate the strategies proposed during the trees' construction and their classification metrics.

2. https://scikit-learn.org/

Scikit-learn is a popular library used for machine-learning tasks in *Python*. Moreover, in this component, we have implemented some routines in parallel during forest construction, such as building the trees and estimating the corresponding errors. The following controller, MC, incorporates native programming to implement the ensemble strategies proposed in the MuSiForest algorithm. In the first data flow (arrows depicted in Figure 5.8), the implementation groups the local models to be shared by the DC controller. In the other flow, it is also in charge of aggregating the models received with the local models to be applied by the site controller. Finally, the distribution controller communicates, shares, and interacts with other sites through the three libraries:

- *ZeroMQ*[3] is used to pass messages between the sites according to the availability of shared data.
- *p2pNetwork*[4] allows decentralized peer-to-peer communication among all sites.
- *Redis*[5] is a distributed data store used for sharing data between sites.

Our system's implementation includes native code and integrates the four popular *Python* libraries mentioned: *Scikit-learn*, *ZeroMQ*, *p2pNetwork*, and *Redis*. The implementation ensures the correct execution of the MuSiForest algorithm in the two flows (construction and application) during the phases proposed in our approach.

5.4.3. MuSiForest Topologies

Our system architecture supports three distributed topologies adapted to multi-site scenarios to analyze biomedical data. The topologies presented are a particular analysis scenario conforming to the FDC architectures discussed in Section 4.3: Sequential, Hierarchical, and Fully Distributed processing. Federated Learning has adopted these processing topologies proposed in distributed communications decades ago [Bar64]. Furthermore, the aggregation strategies in federated training consider different data partitioning strategies (as discussed in Section 3.2.4) and processing architectures, as discussed in this thesis. Therefore, the three proposed topologies are based on traditional federated processing and have been widely discussed by others, as presented in [Liu+22; Ver+20; Li+20b].

3. https://zeromq.org/
4. https://pypi.org/project/p2pnetwork/
5. https://redis.io/

The MuSiForest algorithm can be deployed on each architecture, changing the form of communication and aggregation between multiple processing nodes. Each node can be a single computational device or a sub-network of geographically distributed nodes. The main nodes of each topology can have two roles: processing and aggregator site, or even both. Each topology maintains the three software components discussed: site, model, and distribution controllers. The difference in each is due to the form of communication and sharing between them. In this subsection, we investigate three configurations based on the strategy used to ensemble the collaborative models:

- *Sequential*: in this topology, the aggregation strategies are executed sequentially between all sites. The sequential topology is similar to a pipeline execution, orchestrated by the sequential execution of site-by-site tasks.
- *Hierarchical*: this topology considers the aggregation of the model in a more decentralized way than in the previous case. Hierarchical aggregation considers processing nodes for partial and intermediate aggregation of collaborative models.
- *Fully Distributed*: this strategy considers fully distributed participation among all sites, similar to peer-to-peer communication until everyone participates during the construction of collaborative models.

Each proposed topology is associated with the proposed aggregation strategies in our MuSiForest algorithm, previously discussed in Section 5.2.2.2. Hence, each topology is discussed and detailed below based on the three indicated aggregation strategies.

5.4.3.1. Sequential Aggregation

In the sequential topology (pipe), the model aggregation is executed sequentially over the sites, as shown in Figure 5.10. The first site shares its local trees with the second one. Then, the second one aggregates them with its local versions and then shares the new aggregated model with the third site, and so on, until the `s` sites are participated in building the aggregated models. In the end, the site `s` is responsible for sharing the aggregated model with all sites via each other's distributed controller.

An application case of this topology can consider the training of the forest by building each tree level by level. For example, consider a scenario where each site provides the information for each tree level and sequentially defines each tree level for each tree that composes the forest.

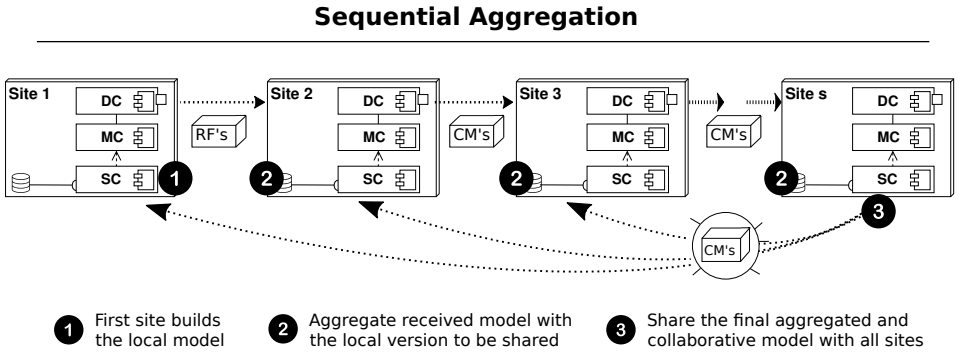


Figure 5.10 – Sequential aggregation, each site builds the local model and shares it with others until all sites participate sequentially.

The algorithm's execution by sequential topology has an important property extended to all topologies, the independent composition; that is, regardless of the sequence order of the sites, the aggregate model will always be the same.

The operation applied during the grouping models (before sharing) justifies the independence property. Remember that the models are grouped through the three forms of aggregation (see explanation in Section 5.2.2.2) employing the union of local trees. The union operation on sets always satisfies the associative property. Therefore, if three sites have the forests RF_1, RF_2, and RF_3, respectively, then the aggregated model conformed by the union of all as $\Big((\mathsf{RF}_1 \cup \mathsf{RF}_2) \cup \mathsf{RF}_3\Big)$ is equivalent to $\Big(\mathsf{RF}_1 \cup (\mathsf{RF}_2 \cup \mathsf{RF}_3)\Big)$. Therefore, the execution order (aggregation) does not matter because the collaborative models will always be the same. The independence property is maintained even when the percentage of trees in each site changes because the collaborative forest follows the associative property during the aggregation process. Our algorithm's associative property on sets and independence composition leads to a faster model ensemble by considering partial aggregations as in the following topology, maintaining the final result, the final collaborative model.

5.4.3.2. Hierarchical Aggregation

Hierarchical aggregation is illustrated in Figure 5.11. The model aggregation is implemented in a more decentralized way than in the previous case. The aggregation process uses intermediate sites responsible for partially aggregating the model on its delegated sites. The collaborative model is shared with all sites when the process covers all levels to the top. The topology provides different levels generating a hierarchy.

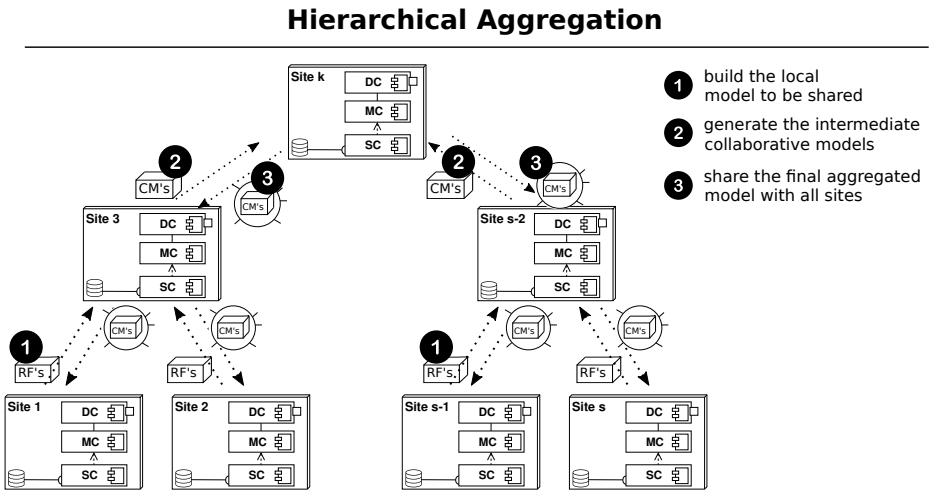


Figure 5.11 – Hierarchical aggregation with one intermediate level, each site builds a local model and shares with the intermediate sites. The intermediary collects the models to the top and then shares the collaborative model with all sites.

A hierarchical deployment can be used if some criteria can group sites. For example, in Figure 5.11, sites 1 and 2 can communicate efficiently as sites $\mathsf{s}-1$ and s , for technical reasons or by geographic location. Then each pair aggregates the model through the intermediate site and so on until all sites are collaborating. For example, in the collaborative scenario between Colombia and France (detailed in Section 2.4), the final aggregation can be responsible for an equidistant site located in France, which has the aggregator role and receives the regional models.

This topology preserves the compositional independence property. Regardless of the number of intermediaries or the hierarchy established during aggregation, the final collaborative model will always be the same. As discussed in the topology above, the associative property argues the reason for the union of sets.

5.4.3.3. Fully Distributed Aggregation

An example of a fully distributed architecture is illustrated in Figure 5.12. This scenario aggregates the model in a fully distributed way to build the collaboration model where all sites have the same role during the process. The scenario removes the hierarchy and intermediate sites from the previous scenario. When all the sites have participated in the process, each has a version of the collaboration model applied locally. The sites communicate using peer-to-peer protocols. Therefore, this topology is completely decen-

tralized than the two presented above.

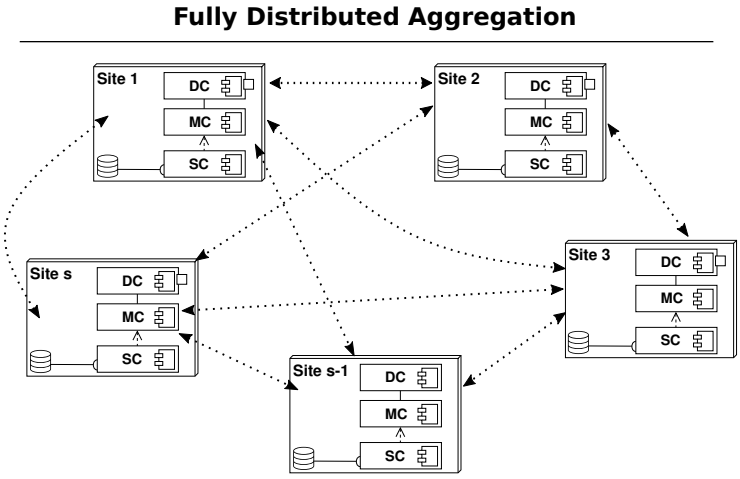


Figure 5.12 – Example of Fully distributed aggregation, through peer-to-peer communication, the sites share the models until the participation of all results in the collaborative version.

The fully distributed topology considers the same role for all the sites. If all the sites participate in the collaboration, the global model still complies with the independence property. The only way to not fulfill the property is if a site fails to participate during the collaboration for some technical reason. Otherwise, this topology also satisfies the set-associative property, fulfilling the independence execution property.

Discussion

The architecture of our system considers diverse topologies where sites can be deployed in three different ways depending on the analysis type and its restrictions. In each topology, a site is a processing node where local tasks can be processed in a distributed way, such as those delegated to the site controller during model building and evaluation. The data demand drives the communication between the sites. When external information is required, the local execution stops until the site offering the data fulfills the required information. The three proposed topologies provide flexibility and scalability to our MuSi-Forest algorithm's execution, such as the Fully Distributed approach, even for large-scale processing data.

Additionally, the topologies preserve the independence property allowing optimal model aggregation during the collaboration. For example, it is possible to consider the geograph-

ical distances among sites. Furthermore, the participation order of sites is indifferent to the final global model during the analysis deployment since the independence property guarantees the same aggregation of the final model during the indistinct participation of all sites. Finally, the topology selection depends on on-site computing capacity, communication channels, geographical location, and legal considerations. The architecture and topologies proposed are completely aligned with FDCs; therefore, these three topologies are highly appropriate for multi-site biomedical analysis.

5.5. Experiments and Results

We conducted the experiments to verify that the challenges of multi-site random forest implementations, see Sec. 3.2.3, are met:

(I) implementing ensemble learning strategies applied to fully distributed architectures,

(II) aggregating the best trees to each site to have a global model with similar learning performance to the centralized version,

(III) designing flexible and scalable distributed workflows to analyze biomedical data,

(IV) incorporating sharing strategies while preserving data privacy and security.

These challenges are fulfilled in the experiments of our implementation, in particular, we seek to evaluate three aspects:

1. The performance of the classification model as evaluated using metrics (see Sec. 5.1.2.1).

2. The size of data shared by trained models.

3. The computing capacity required at each site to train the models privately.

We contrast the corresponding results with a traditional centralized training algorithm, that stores all data at a single site. This requires the property to be satisfied that biomedical data can be freely moved and shared on a central site.

The models trained in our experiments correspond to a supervised classification problem. The dataset used is labeled and corresponds to gene expression levels for five types of cancer (RNA-seq data). Our learning objective is to train a multiclass classification

model to predict the cancer type from numerical values corresponding to gene expression levels.

Concretely, the experiment compares the training of prediction models using a random forest on a centralized setting versus collaborative models built using the MuSiForest implementation in a distributed setting. The distributed environments use configurations of 3, 5, and 8 geo-distributed sites. Execution environments (algorithms, data, ...) have been deployed on the grid-cluster platform for distributed systems Grid'5000[6](*G5K*) [Cap+05].

5.5.1. Dataset Description

The experimented data (RNA-Seq) correspond to gene expression levels for five types of cancer: Breast (BRCA), Renal Kidney (KIRC), Colon (COAD), Lung (LUAD), and Prostate (PRAD). The dataset is a part of the information published in the PCAWG (Pan-Cancer Analysis of Whole Genomes) project [Por; Who20]. The experimental dataset has gene expression levels of $20,531$ genes (attributes) for $10,400$ patient samples stored in approximately 1.8 Gigabytes.

The gene expression levels correspond to values associated with the degree of transcription of each gene. Each value is obtained by applying quantification methods on biological tissues over normal and mutated cells (see [Kor+14; Liñ+19] for more details about these quantification methods). Gene expression analysis allows to understand the biological principles to establish variations in the phenotypes related to human diseases.

For the centralized scenario, all the data is stored on one site for model training. In the distributed environments, the patient samples are randomly horizontally partitioned into smaller data sets used to train the models on each site.

5.5.2. The Grid'5000 testbed

Grid'5000[7] (*G5K*) is a large-scale, geo-distributed, highly re-configurable experimental grid testbed for computer science experiments, scientific algorithms, and distributed infrastructures. *G5K* is maintained and financed by the French and European informatics communities, notably through the EU project and infrastructure slices. Figure 5.13 presents the physical architecture of *G5K* hosted at (currently) 12 sites in France and Luxembourg. At each site one or a small number of clusters is installed, totaling approximately

6. https://www.grid5000.fr/
7. https://www.grid5000.fr/

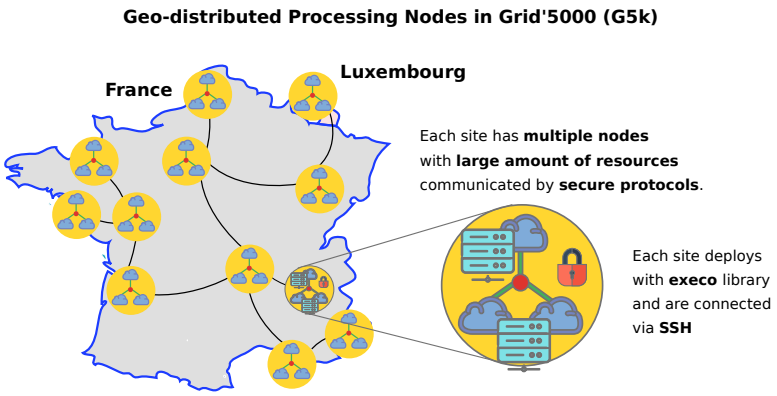


Figure 5.13 – Grid'5000 (*G5K*) nodes distribution.

800 compute nodes and 15000 cores. Employing advanced virtualization techniques, geo-distributed applications can be executed practically and efficiently on hundreds of thousands of virtual machines. Therefore, *G5K* allows us to simulate geo-distributed biomedical analyses in a controlled environment.

Deploying on Grid'5000 (*G5K*)

The execution time of our experiments on *G5K* was, on average, between 3 and 5 days. The duration depends on the size of the data and the number of sites deployed. We complement our implementation with a module to deploy the experiments in a controlled and automated way employing the `execo`[8] library. `Execo` provides an API for local or remote, standalone or parallel, process execution and also enables large-scale running experiments on distributed systems. With this deployment module, we reserve the desired number of sites and the number of nodes per site to simulate the geo-distributed environment. Each site and each node can have a different configuration which we support through our deployment module on *G5K*. The machine's reservation in *G5K* is associated with jobs, and they have a duration in time. We monitored each site and its nodes to have information on the process status and guarantee the complete execution of tasks. Our deployment module supports controlled execution between the different sites between France and Luxembourg supported on *G5K*.

8. http://execo.gforge.inria.fr/

5.5.3. Experimental Setup

We deployed and executed the experiments on Grid'5000. With our deployment module, we load configuration files and deploy all the FDC dependencies on each geo-distributed site in *G5K*. First, we set up the geo-distributed environment with each node's initialization. Next, we install the FDC components on each node. Finally, Next, we move the libraries required and install them on each node deployed. At this point, the geo-distributed environment is ready to analyze data according to the multi-site workflow design.

We used a hierarchical topology with one aggregator site for the experiments with distributed architectures. The hierarchical deployment allows the experimentation of the collaborative scenario between Colombia and France, which considers regional grouping in each country and global aggregation in any of the sites. However, without loss of generality, the deployed scenario can consider any of the topologies supported by our system architecture explained in the previous section.

We deploy our experiments on machines with Intel E5 Xeon Processor 2.2GHz with 32CPUs, each running Ubuntu 18. This configuration is the same for all sites, even for the centralized scenario. Each machine's system is installed, including our architecture based on the three components. The experimental data is randomly divided for each distributed configuration, and each site keeps the data private by never sharing patient samples. Moreover, each site has a local configuration file to specify hyper-parameters learning (e.g., number of trees) and connection parameters (e.g., IP addresses).

The experimentation contemplates executing the MuSiForest algorithm following the methodology explained for the aggregation and construction of the collaborative models (explained in Section 5.2.2). The deployment environments consider three distributed settings with the participation of 3, 5, and 8 sites. The first scenario, with 3 sites, has two forest configurations with 100 and 1000 trees, while the centralized version independently trains models with 300 and 3000 trees on the aggregated data. The second comprises 5 sites that train a forest with 200 trees. Finally, the last scenario, with 8 sites, trains each local forest with 125 trees. The total number of aggregated trees equals the number of trees in the centralized model, 3000 trees, to compare different forests under the same conditions.

The idea of the deployed experiment is presented in Figure 5.14. It follows the MuSiForest phases detailed in Section 5.2.2. First, each site trains the random forest model locally following the MuSiForest algorithm. Then, the construction of each model starts

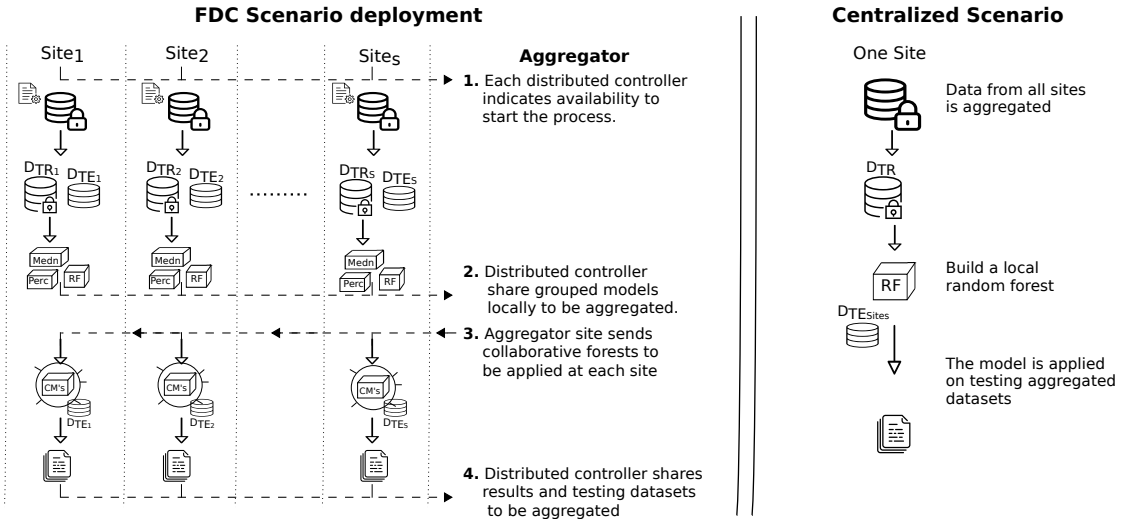


Figure 5.14 – Experiments deployed with the participation of s sites versus the centralized training.

using training data, $\mathsf{D}_{\mathsf{TR}_i}$, and the testing set $\mathsf{D}_{\mathsf{TE}_i}$, is used to evaluate the model. Each site trains local RF models and then groups them using the three strategies to be shared with the responsible site. The experiment considers the hierarchical topology to aggregate the model. However, the aggregator's role can be randomly assigned under any other architecture since any site can run as the aggregating site during the algorithm's execution.

5.5.4. Experimental Results

The experiments focused on three relevant aspects to show the advantages of FDC analyses versus the centralized scenario:

(i) evaluation of each collaborative model,

(ii) size of data shared during collaboration, and

(iii) computational required time for model training.

Evaluating these three aspects, we partially alleviate the challenges in the current implementations of multi-site forests. The implementation and the proposed deployment topologies complemented the full achievement of these challenges discussed at the end of the experimental results.

Models Evaluation

We conducted experiments for different distributed environments by changing the number of sites and setting up random forests with different numbers of trees. The evaluation of the models considered four classification metrics: accuracy, precision, recall, and FScore, which were explained in Section 5.1.2.1.

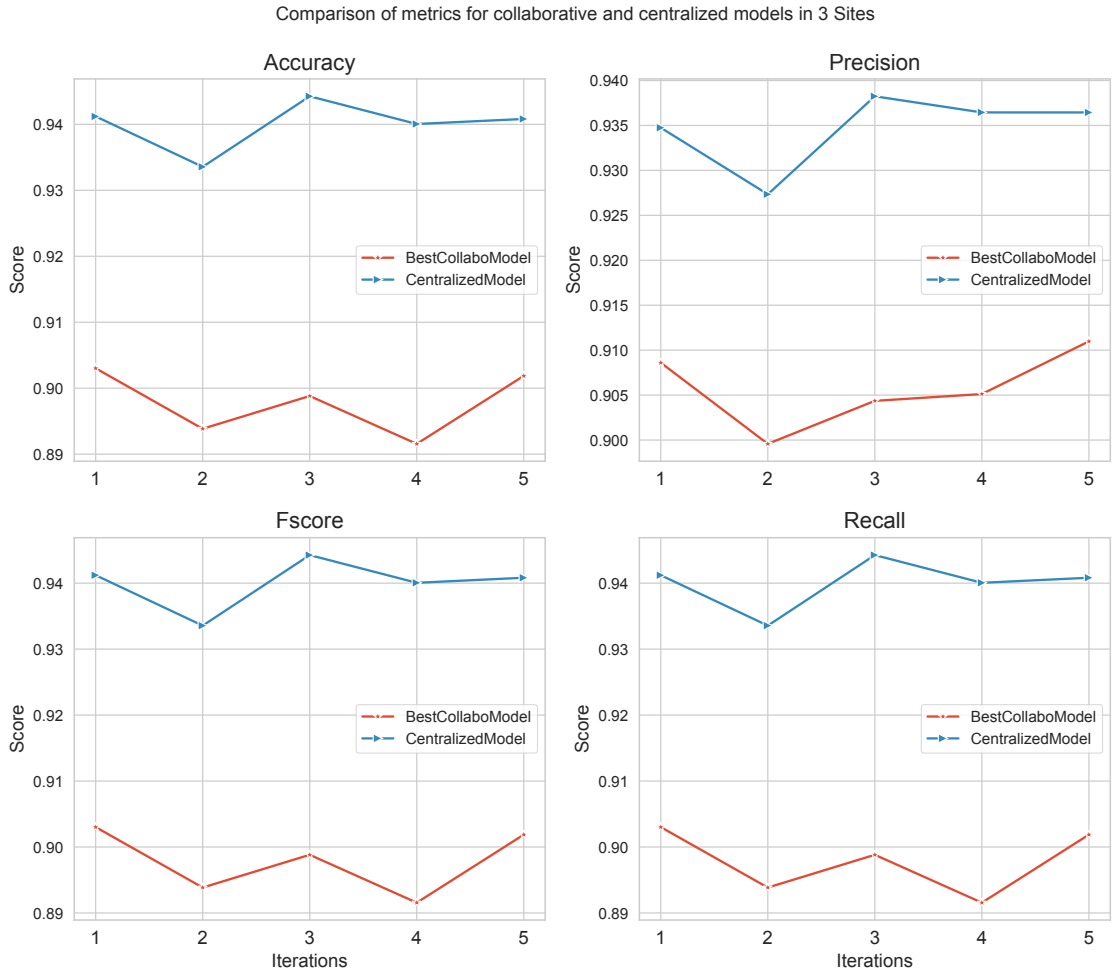


Figure 5.15 – Comparison of metrics for collaborative and centralized models in 3 Sites.

Figure 5.15 presents these four metrics for the experiment deployed on three sites. In this experiment, each site builds a forest of 100 local trees, and the size of the collaborative model is 300 trees. Similarly, the centralized model comprises 300 trees but is trained over aggregated data on one site. The x-axis indicates the number of iterations, corresponding to the number of times each model was trained independently. The centralized model does better on all four metrics than the collaborative model because it trains the model

on all the aggregated data on one site. However, the movement and central processing are highly costly, as we will show later based on the experimental results. Figure 5.15 shows a difference between 0.2 and 0.4 points between the centralized and collaborative models' for the FScore. We consider this difference tolerable due to the unfeasibility and expensive centralized training as shown by the experimental results.

To compare the classification performance of the models, we select the FScore to find a balance between precision and recall using a single metric. Remember that precision refers to how the model behaves regarding the number of predicted positives. However, the precision does not include how many observations of actual positive classes were predicted to be in the negative class. In contrast, recall measures the ability to find all the relevant cases within a data set. Therefore, the selected metric, FScore, seeks to compare performance by combining precision and recall to mitigate the impact of false positives and false negatives calculated between multiple iterations.

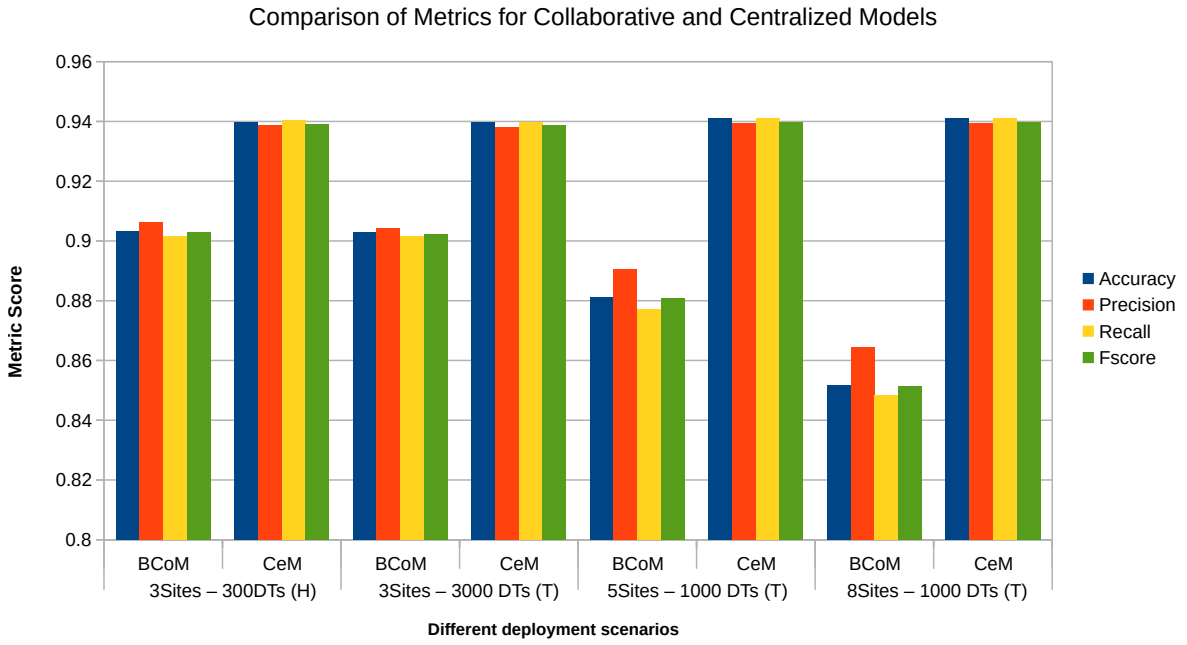


Figure 5.16 – Comparison of metrics for the models: BestPercCollaborative (BCoM) and Centralized (CeM) in different distributed scenarios.

Figure 5.16 presents these classification metrics for more distributed experiments for two random forest versions, Best Median Collaborative (BCoM) and Centralized Model (CeM). The x-axis values indicate the number of deployed sites and the configuration of each forest represented by Hundreds (H) or Thousands (K) of trees. In these other

Table 5.2 – Evaluation of the FScore metric for the trained models

Sites	FScore							
	Join		BestPerc		BestMedn		Centralized	
	Score	DTs	Score	DTs	Score	DTs	Score	DTs
3	0.9027	300	0.8978	246	**0.8983**	160	0.9388	300
3	0.9020	3000	0.8966	817	**0.8985**	528	0.9385	3000
5	0.8809	1000	0.8701	811	**0.8734**	538	0.9396	1000
8	0.8514	1000	0.8436	818	**0.8457**	530	0.9396	1000

distributed scenarios, the evaluation metrics always show a better result in the centralized model versus the collaborative version. Furthermore, while the number of sites increases, the difference with the centralized model also increases.

Table 5.2 presents the FScore metrics for the three collaborative model versions compared to the centralized training. The evaluation considered different forest configurations determined by the number of trees (DTs). The table shows forests with different tree sizes due to the threshold value determined during the execution to select the best trees in each site according to the three aggregation strategies. The collaborative model version that showed the best precision is *BestMedn*, which reduces the difference with respect to the centralized model. In the Table, the first record value in bold indicates that the best score obtained from the *BestMdn* version was 89.78% versus 93.88% for the centralized model. The delta value, the difference between the precision of the centralized model and the MuSiForest algorithm, is almost four points, with the centralized slightly better. However, the difference is minimal, in contrast to the MuSiForest versions.

Figure 5.17 presents more detail on these FScores between the central and collaborative models. The three plotted results show each of the three aggregation strategies of our algorithm against the respective version of the central forest. The difference between centralized and collaborative versions lies between 0.3 and 0.6 points, depending on the number of distributed sites. While the number of distributed sites increases, the difference between the two precision scores is the same as the delta value. The main reason is the degree of distribution of the data; more distribution leads to a model with lesser generalization capabilities than the centralized model.

Shared Data Size

The second aspect evaluated corresponds to the amount of data shared through the MuSiForest strategies and the centralized one. The MuSiForest algorithm considers sharing

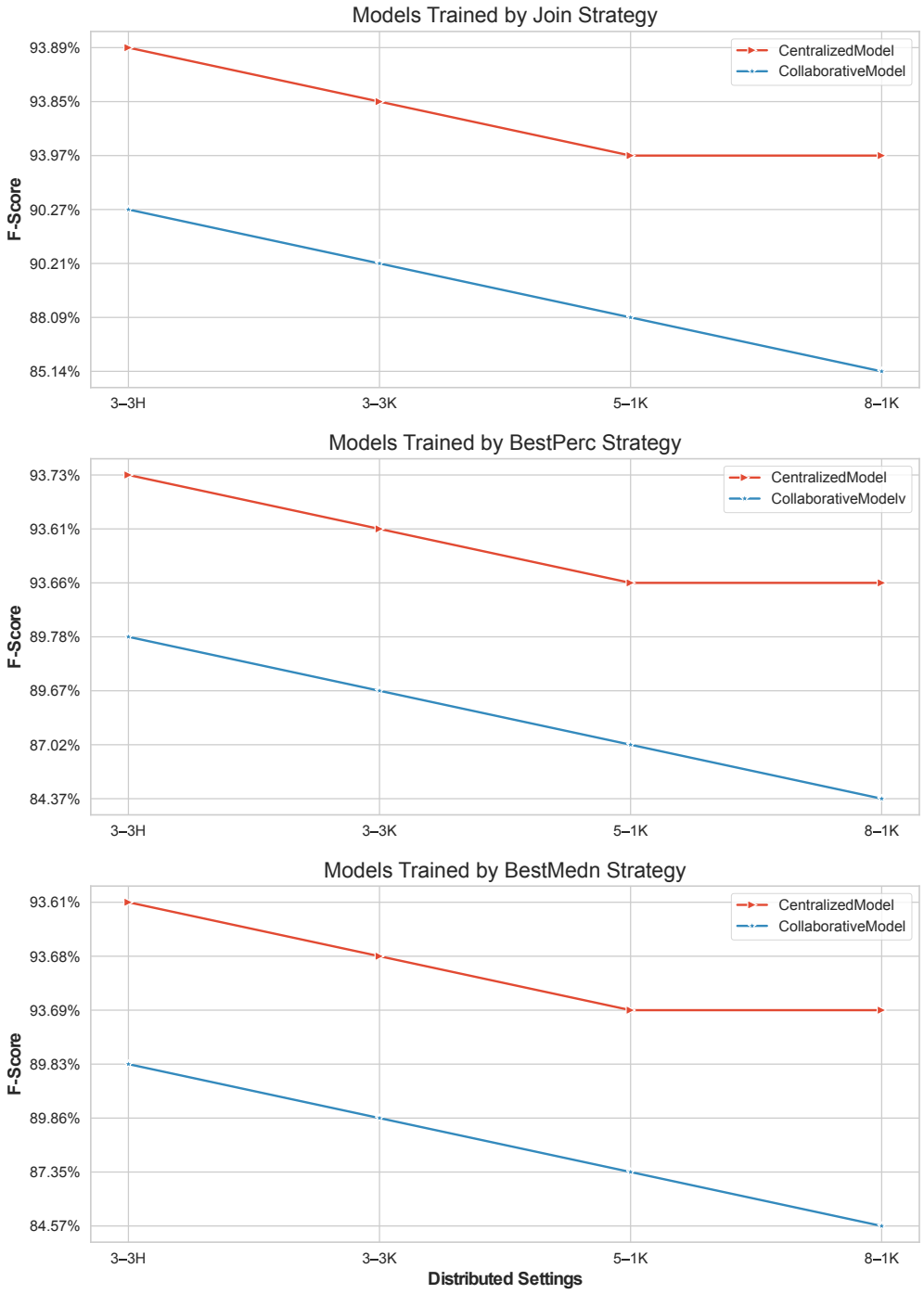
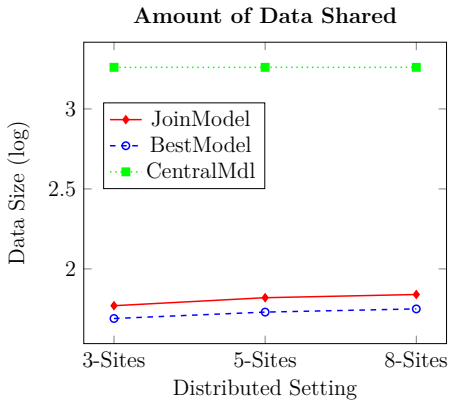


Figure 5.17 – FScore metrics for collaborative and centralized models in different distributed settings (H:Hundreds, K:Thousands) based from Table 5.2.

models that are trained privately, while the centralized one requires moving all raw data to a single site. Figure 5.18 illustrates the shared data sizes during model training for the

different distributed settings. Each site has part of the data due to the initial partitioning process of the experimental data, in total, corresponds to 1.8Gb.



No. Sites	DT	Data Size Shared (MB) per Site		
		JoinMdl	BestMdl	Centralized
3	300	59.5	49.4	1800
5	200	65.5	53.8	
8	125	69.1	55.8	

Figure 5.18 – Amount of data shared between collaborative models and moved raw data in the centralized case.

The evaluation of the distributed scenarios considers different forest compositions due to the corresponding number of decision trees. The centralized model has been handled using the same conditions: aggregate data set, forest composition, and machine configuration conditions. The *join* version shares trained models between 60MB and 70MB of data, and data sharing of the *best* version amounts to 49MB and 55MB.

In contrast to sharing models, the centralized version requires moving all the data to a single site, in our case, almost 2Gb of data. Our proposal to share models significantly reduces the size of the shared data. The model's size is directly related to the number of trees in the forest. As a result, the stored space in the forest is also much smaller than the size of the shared raw data. The data size also impacts the required time and bandwidth capacity to move the data (models) depending on the learning scenario. This notable reduction has to be a trade-off against the model's precision score, the cost of moving raw data to a single site, and potential legal restrictions.

Computational Time

The last aspect evaluated is the training time of each model version in each distributed setting and centralized one. The time required to train the MuSiForest versions and the centralized model is presented in Figure 5.19. The computational training time of models at different sites significantly reduces the computational resources required compared

to the central forest. Figure 5.19 illustrates the computational time required to train the models in the two scenarios. The first scenario, the distributed one, corresponds to our algorithm deployed among several sites, and the second contrast the forest trained traditionally in a central site on aggregated data. The centralized model requires more computation time to train the model. We have evaluated all cases with the same conditions concerning machine configuration and model parameters in both scenarios. The MuSiForest algorithm reduces the computational time required by taking advantage of the local computation capacity installed at each site. The last aspect constitutes another advantage of analyzing data using our FDC approach.

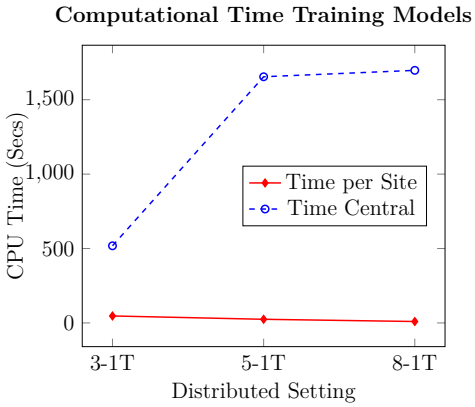


Figure 5.19 – Computational Time Training a Model: centralized versus collaborative versions.

5.5.5. Bias-Correction Evaluation

The last challenge to evaluate is the scenario where partial data is shared securely to mitigate the bias of the fully distributed trained model according to the last challenge: incorporating sharing strategies while preserving data privacy and security. Therefore, we implement the first proposed strategy for bias correction, defined as *Secure Data Containers* (SDC) in Section 5.3.

In brief, the strategy considers a trusted site, an aggregator, with a global data set. This data set has the number of samples from each site, but the data samples are stored privately on each site. Although the global set has an identifier corresponding to each sample at each site, this number is not sensitive data. It simply corresponds to a unique numerical value in the global data set.

The aggregator starts the process by resampling all these identifiers, then sends the required identifiers to each site. Next, each site selects the samples with that identifiers and packages and encrypts them to be shared with the aggregator. Finally, the aggregator follows the traditional training model with the decrypted data and generates models from the three aggregation strategies. We used the same data set described at the beginning of the section and held the random partitioning at each site.

We implement the strategy with three sites. The objective is to compare the training model's learning performance using the secure data container strategy versus the collaborative model aggregated with MuSiForest. The *Python* implementation uses the *Fernet*[9] library to generate encrypted samples. As the aggregator is a secure site, then each site shares the key with the aggregator to consolidate the global set and generate the training samples.

Dataset Description

The data set is the same used in the experiments described in the previous section. The dataset samples correspond to 1.8Gb of gene expression levels for five types of cancer comprised of $10,471$ samples and $20,533$ genes. In our experiment, we consider three sites where the data is randomly partitioned based on the horizontal data strategy.

Table 5.3 – Description of each data set in each site experimented in the Secure Data Containers' strategy.

Data Description	Size	No.Samples	No.Genes
Central Data	1.8Gb	10471	20533
Data per site	∼549MB	∼3491	
Data required per site	∼84KB	∼2400	
Data unencrypted per site	∼280MB		
Data encrypted per site	∼370MB		

Table 5.3 presents the description of each site's datasets and the information of required data sets (unencrypted and encrypted). The required data sets corresponding to the total number of samples resampled by the aggregator at each site corresponds to 70% of the samples, which follows the traditional training model learning. The data size per site is averaged based on the number of sites. The number of samples required corresponds to

9. https://cryptography.io/en/latest/fernet/

70% of samples randomly determined by the aggregator. The table shows different data sizes, but only encrypted data is shared with the aggregator. In our experiment, the data transferred by each site corresponds to an average of 370MB. This size results from an average of 68% of the raw data stored on each site. These figures regarding the number of samples and genes correspond to the horizontal partitioning strategy on all data and the data set used for centralized training. The data set encrypted by each site is shared with the aggregator to train the model following the three aggregation strategies proposed with our MuSiForest algorithm.

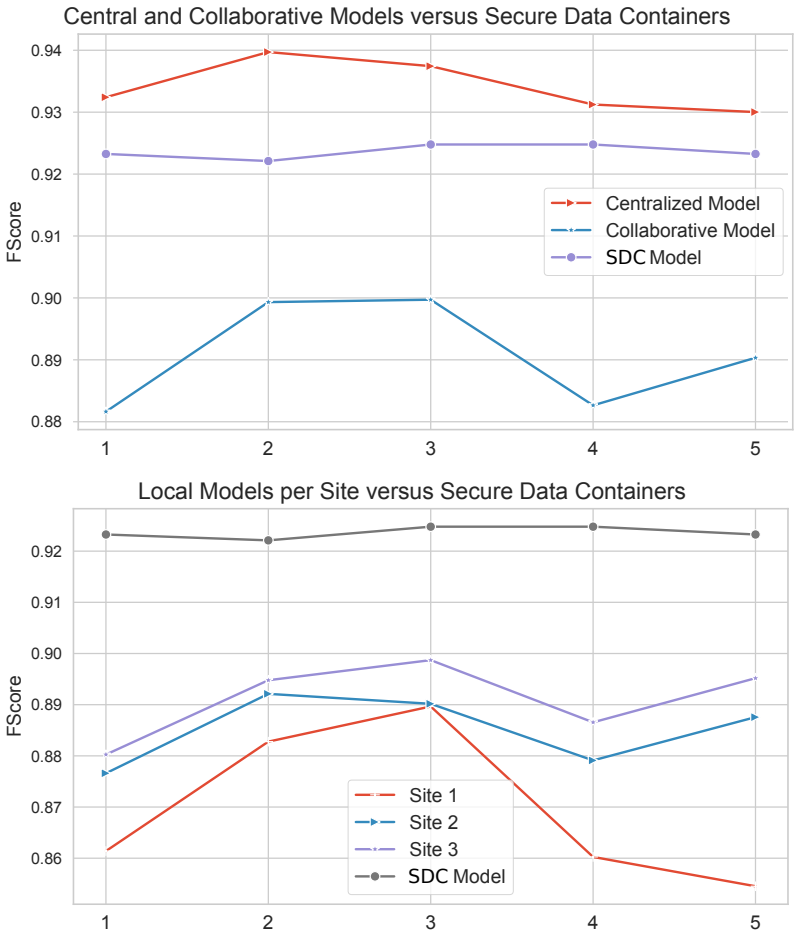


Figure 5.20 – Comparison of local models per site versus the Secure Data Containers (SDC) strategy for three sites.

Figure 5.20 presents the model's performance trained on a central and collaborative version versus the Secure Data Containers (SDC) strategy between three sites. The upper part shows three models: the centralized (trained over aggregated data), the trained

through secure data sharing strategy and the collaborative (based on the MuSiForest strategy). The centralized model presents better precision than the other two, notable for the five iterations experienced. The difference between the central model (red line) and the collaborative model (blue line) follows that shown in the experiments (see Figure 5.17). The difference corresponds to 2 or 3 percentage points. In contrast, we compared the central model with the proposed secure model (purple line) and found a reduction in the difference between these two learning performances.

The central model continues with better performance, but the difference is reduced compared with the SDC model and the MuSiForest proposal. The difference corresponds to between 0.3 and 1.4 points. This performance learning result improves the model aggregated by secure sharing SDC versus the collaborative MuSiForest model. Therefore, our safe sharing strategy to improve the bias of the global model is feasible according to these experimental figures. Since the performance of the SDC model (purple line) is improved versus the fully distributed proposal, the MuSiForest algorithm (blue line).

In Figure 5.20, the lower part shows the result of the model trained using the SDC strategy versus the local model trained at each site. The SDC model shows a better performance concerning the locally trained ones compared to our MuSiForest approach. The SDC model shows a better performance concerning the locally trained ones and is also better with respect to the one trained without data sharing, our MuSiForest approach.

5.5.6. Discussion

As a result of the experiments and the evaluation outcomes, the following insights can be emphasized:

- The learning performance, FScore metric, of the collaborative model is between 0.2 and 0.4 points lower than that of the centralized version. The better value of the centralized model is due to performing model training based on all data aggregated on one site.

- The centralized model shows expensive training in terms of computation capacities and communication restrictions moving all data. Our algorithm significantly reduces the computing capacity required to train the model on each site.

- Our algorithm significantly reduces the amount of data shared due to sharing only trained models. Furthermore, the aggregation strategies proposed significantly re-

duces the amount of information shared, in contrast to the often infeasible process of moving all the data to one site.

- We have experimented with one bias-correction strategy to improve the precision score of the collaborative model (Secure Data Containers strategy, see Sec. 5.3). This strategy shows an improvement the learning metric compared to the centralized model.

- To improve our classification performance, we propose strategies to share partial data securely, experimentally showing that it improves classification performance.

- Our experimental results are similar to current approaches to training multi-site random forests [Liu+20a; Sou+20]. The central model always has better precision than the multi-site. However, training on aggregated data is computationally expensive and infeasible due to restrictions on biomedical data.

- According to Giacomelli et al. [Gia+19], our difference values with the centralized model are similar for models trained collaboratively, centrally, and entirely locally without data sharing.

Our FDC-based Forest training approach offers computational relief and guarantees data confidentiality but maintains lower learning performance than the costly and unfeasible centralized training. Moreover, due to the previously mentioned insights, the approach is beneficial to promote collaborations around training prediction models based on ensemble learning techniques.

Furthermore, the experimental results fulfill the challenges stated at the beginning of Section 3.2.3 (copied below and shown in *italics*) with respect to existing multi-site Random Forest implementations:

(I) *Implementing ensemble learning strategies applied to fully distributed architectures.* Our algorithm is designed and implemented through architectural components that support independent and highly distributed execution. Furthermore, the implementation supports topologies such as hierarchical and fully distributed ones used in Federated Learning that can be combined to take full advantage of the distributed resources at each site. Therefore, our implementation appropriates the FDC features to provide a powerful multi-site analysis over biomedical data.

(II) *Aggregating the best trees to each site to have a global model with similar learning performance to the centralized version.*
The proposed aggregation strategies select the best trees from each site to build collaborative models. As a result, the precision of our collaborative model is lower than the expensive centralized model. However, we consider the difference acceptable concerning the other benefits obtained with our algorithm. For example, our collaborative model can assist medical decisions and supports medical hypotheses. Nevertheless, we never consider substituting the medical expert because the expert will always be responsible for the patient's final decision. However, we implemented one bias-correction strategy sharing secure partial data, and our collaborative model improves the precision performance.

(III) *Designing flexible and scalable distributed workflows to analyze biomedical data.*
The architecture implemented complies with two principles: flexibility and scalability. Our proposal is flexible since it adjusts to the sites' heterogeneous capabilities and communication capacities. The implemented topologies provide flexible analyses with the ability to adapt to site conditions. On the other hand, the independence of the components makes them highly scalable to satisfy the processing capacity on site. Additionally, tasks on the controllers can be executed by leveraging distributed processing at each site. Therefore, our implementation is more robust with these two principles in the face of heterogeneous processing workflows between multiple sites.

(IV) *Incorporating sharing strategies while preserving data privacy and security.*
Our implementation is extended through three strategies to improve the performance learning of the model seeking to reduce bias. In the experimental phase, we showed that one of the strategies improves the precision of the model through the secure sharing of partial data. The three proposed strategies are based on existing techniques for securely sharing biomedical data. Therefore, we use these secure techniques to enrich our approach to building collaborative models.

In conclusion, the MuSiForest implementation offers favorable properties for analyzing biomedical data supported by a flexible and scalable architecture compared to traditional centralized training. Furthermore, the FDC features alleviate data sharing restrictions, reduce the computational time training centralized models, minimize the amount and type

of data sharing, and favor biomedical data privacy. Therefore, the FDC-based analyses offer a good balance of these aspects versus the expensive training in a centralized way.

5.6. Conclusions

In this chapter, we propose a novel approach, MuSiForest, to build a collaborative model in terms of forests composed of multiple trees that are trained privately among multiple sites. We have performed, evaluated and analyzed multiple experiments consisting of analyses over labeled data over gene expression levels in order to classify patient samples among five cancer types. We include three strategies to collaborate between the parties that share models without revealing sensible data from each site. We only use partial data represented in trees to collaborate among all and build a global model. The proposed approach mitigates restrictions on sharing data, such as technical, legal, and ethical ones. In addition, we proposed bias correction techniques to improve the precision of models seeking to be fair in learning with more representation of the knowledge of the sites. Experiments have shown a reduced amount of shared data, decreased computational overhead, and a noticeable reduction with our distributed processing approach. However, this minimal difference with the expensive centralized training does not affect the prediction results since these learning models seek to assist the medical experts based on insights from the data analysis. Therefore, the trained models will never replace medical experts because they are only assistance tools. For this reason, we consider having an efficient training approach in a distributed setting, respecting data privacy restrictions, and providing more rapid data processing than the time-consuming and non-viable centralized training over biomedical data. FDC-based analyses are a promising solution for collaborative data processing, such as the proposal presented in this chapter, the MuSiForest algorithm, as opposed to expensive and often inappropriate centralized processing.

Chapter 6

Workflow Language for FDCs (FeDeRa)

Introduction

Workflow systems are frequently used in biomedical research to model scientific experiments. They enable scientists, who are often laymen in informatics, to model analyses in terms of tasks, dependencies, and dataflows [Liu+15]. Workflow systems also frequently support portability across different execution environments, such as grids and clusters [YB05; RB17]. However, current workflow systems do not support global multi-site collaborations, such as FDCs, because they lack coordination mechanisms for decentralization, distributed computations, and security and privacy models [PCA15].

Current workflow systems offer mechanisms to specify scientific analyses, but they have limited means for configuring deployments on complex and heterogeneous distributed environments. In particular, they lack mechanisms to support geo-distributed processing. In this sense, FDC-compliant workflows must provide intuitive and flexible distributed workflow specification languages suitable to be used by researchers that are not experts in computer science.

In this chapter, we first investigate the expressiveness level and ease of use of current workflow specification languages. We then identify shortcomings in current workflow languages to specify workflow analysis in multi-site scenarios. Based on this analysis, we propose FeDeRa, a language to specify multi-site analyses. The FeDeRa language also provides abstractions to fine-tune the deployment across distributed machines.

This chapter is structured as follows. Section 6.1 introduces workflow languages and presents some popular workflow systems used in biomedical analyses. Section 6.2 presents a workflow involving a multi-site machine learning scenario. Section 6.3 introduces our distributed and declarative language with its features, syntax, and semantics. Section 6.4 presents the implementation, architecture, and deployment features. Section 6.5 compares

our language to other workflow languages. Finally, in Section 6.6, we present a conclusion.

6.1. Workflow Languages for Biomedical Analyses

Workflow systems are popular tools used to analyze biomedical data, allowing scientists to define tasks as well as dependencies between tasks [Liu+15]. The workflow task definitions and their dependencies define the steps and the execution order of those steps, required to complete a data analysis. The workflow specification can be represented in graphical form or using scripting text-based languages. Workflow languages typically adopt one of two strategies to task dependencies [Tay+07b]. The data-flow strategy defines interactions in terms of data provided by one task and needed by others. The control-flow strategy defines task dependencies in terms of the transfer of control between tasks. Typical control-oriented operators execute tasks sequentially, depending on conditions or iteratively [Zha+12b].

The popularity of workflow systems has increased remarkably in recent years. For instance, the Common Workflow Language (CWL) website[1] lists over 200 workflow systems. Biomedical data processing contributes to this popularity. Workflow languages support, in particular, the management and automation of analysis pipelines. The researchers can modularize their pipelines, facilitating problem-solving and maintainability. Moreover, some workflow languages integrate specialized libraries for common tasks, such as sanitizing data or particular genomic tasks. These benefits can reduce the time required to code pipelines, supporting more complex but also more comprehensible designs as well as supporting reproducible analyses.

The diversity in workflow systems has proliferated in diverse specification languages since there is no standard specification language among all systems. Therefore, each workflow system offers different ways of specifying and how analyses are executed and processed, some systems and their properties were discussed in Section 3.1.2.1. Therefore, researchers can choose the most appropriate one according to particular needs.

6.1.1. Distributed Workflow Languages

Workflow systems for geo-distributed processing involving biomedical data have to observe data-sharing restrictions. Nowadays, workflow systems support distributed pro-

1. https://www.commonwl.org/

cessing between different cloud sites and distributed infrastructures. For example, Taverna [Wol+13] can be executed on clusters, grids, and clouds. However, those configurations require extensive technical knowledge by the deployer. Nextflow [Di +17] and Snakemake [KR12] support grid platforms, e.g., SGE (Sun Grid Engine) and LSF (Load Sharing Facility). Pegasus [Dee+15] has taken a more explicit approach to distribution and supports execution on individual machines, remote clusters, distributed infrastructures, and clouds. The execution on multiple sites is usually harnessing the platforms' virtualization properties [Liu14]. The processing of biomedical data across multiple sites has to be taken into account limitations related to the definition and execution of distributed workflows. The high-level specification means partially depend on the other workflow system layers, for instance, on operational aspects, such as resource allocation. A major resulting limitation, pointed out by different studies, consists in the limited functionality to support geographically distributed processing in workflow systems, notably among geo-distributed cloud sites [PCA15]. These features directly impact the ability to specify biomedical analyses among multiple sites.

6.1.2. Baseline Workflow Languages

Due to the wide variety of workflow systems, we select three of them: Swift [Wil+11], Snakemake [KR12], and Pegasus [Dee+15], which have been regularly used for biomedical analyses, such as processing genomics data [Bux18]. We want to compare our specification proposal against the mechanisms offered by these three widespread systems. The main objective is to show the advantages of our proposal, especially concerning multi-site analyses, such as those supported by FDCs.

Systems like Snakemake and Swift offer much more lightweight text-based specification languages than complex systems like Pegasus [Bux+17]. Snakemake is based on the GNU make tool that supports the definition of task-oriented workflows. While Swift, through its scripting language, allows the simple definition of parallel instructions on the data. These specification forms allow the scalable execution of workflows on distributed architectures but leave out features such as data partitioning, as is supported by Pegasus. Pegasus provides distributed execution capabilities to efficiently use installed resources for distribution, monitoring, and groupings [Bux18]. In the following we provide the main characteristics of Swift, Snakemake, and Pegasus (so that we can later compare them to our proposal).

Swift

Swift [Wil+11] offers a scripting language that supports large-scale and parallel/distributed workflows. Swift has been widely used for biomedical analyses, notably by taking advantage of on-site distributed processing [Wil+09; DeB+10]. The workflow execution translates the steps into executing tasks, which can be performed distributively through the Turbine distributed run-time system. However, the tasks access the data by reading or writing on a shared data system, leading to a bottleneck for large-scale processing. Swift specifications are based on the C language, that is, instructions are defined in an imperative style, a control-flow oriented style, as opposed to dataflow-based analysis.

Snakemake

The Snakemake [KR12] workflow system supports the creation of reproducible and scalable data analyses. For example, different biomedical data analyses have been implemented in Snakemake, such as analyses of cancer genomics data [MGG10]. Snakemake specifies workflows in Snakefiles that describe rules with Python-like syntax, including input and output data between them. However, Snakemake does not provide its own distributed execution environment. Therefore, distributed workflow analyzes lead to complex specifications due to the interaction between multiple files between their rules and dependencies. The Snakemake specifications harnesses the benefits of *Python* syntax; however, configuring multi-site workflows are very difficult to define because each site must be defined by one rules file.

Pegasus

Pegasus [Dee+15] is a scalable workflow management system that takes a more explicit approach to distribution, supporting the execution on local machines, remote clusters, distributed computing environments, and clouds. Pegasus has also been used in several biomedical tasks, such as analyzing genomic data. Although Pegasus supports multiple distributed environments, it is limited in specifying and executing workflows or parts of them across different geo-distributed distributed sites.

Workflow Languages Problems

The three workflow systems described above provide different mechanisms to implement tasks between multiple machines on-site and specification forms according to the

execution capabilities of each system. However, the limitation of workflow systems to process multi-site analysis across sites leads to limitations in specifying workflow across geo-distributed sites. For instance, specifying collaborative analyses is complex due to the lack of mechanisms to combine rules with external data sources required during an analysis. Additionally, the complexity of specifying and allocating distributed computing resources requires highly-specialized technical expertise. This constitutes a prime motivation to equip our FDC approach with a new specification language that supports the definition of collaborative workflows for expert and non-expert users. An extensive discussion of the functionality and limitations of multi-site workflows is presented in [Gar+22].

Finally, workflow systems should provide mechanisms for experiment reproducibility, which is essential for collaborative research studying biomedical issues. Some researchers even claim that the life sciences are experiencing a reproducibility crisis [Fry+15]. The research community has pressured researchers to ensure their experiments and results are reproducible [HG13b]. Current workflow mechanisms have limitations in specifying geo-distributed analyses and limit the reproducibility of experiments due to the distribution of multiple resources, leading to problems such as heterogeneity, scale, and instability [Wan06; WCW08]. Therefore, reproducibility is another property to consider besides specifying multi-site workflow analyses.

6.2. Workflow-based Machine Learning Analysis

In order to concretely motivate our language contribution, we reconsider the collaborative scenario between Colombia and France (that has been first presented on page 40). The scenario contemplates a collaborative training of local random forest models, sharing partial information between the sites. However, this collaborative analysis is challenging for current workflow systems because they lack mechanisms for multi-site analysis on heterogeneous architectures. Similarly, workflow specification languages between multiple geo-distributed sites are lacking coordination mechanisms for decentralization and distributed computations (also see the discussion in Chapter 3).

This collaborative scenario comprises different sites with differing communication capacities and, potentially, heterogeneous infrastructures. We use this scenario to show the benefits of our declarative approach compared to the specification mechanisms offered by other workflow systems.

Figure 6.1 shows a workflow representation of the scenario, which considers the training

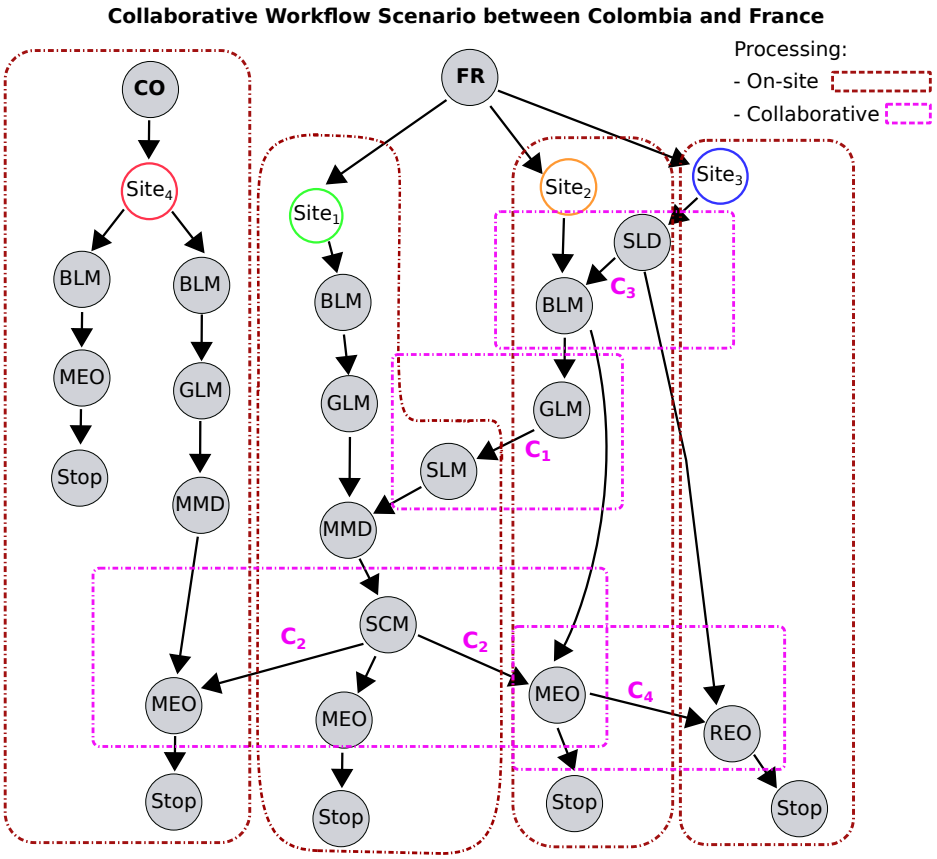


Figure 6.1 – Workflow for collaborative training between CO and FR (described in Section 2.4). (BLM: Build a Local Model, GLM: Group Local Models, SLD: Share Local Data, MMD: Merge Models, SLM: Share Local Models, SCM: Share Collaborative Models, MEO: Model Evaluation, REO: Receive Evaluation and Outcomes).

of distributed models based on the execution of tasks in each site and some collaborations. The workflow illustrated is heterogeneous, from executing the steps at each site to combining results and supporting different data types. This kind of workflow is dynamic and has a heterogeneous behavior.

The illustrated workflow has two local flows within site_4. On that site, in the right flow, Colombia participates with France to analyze common data, for example, genomic data. As part of the other flow (left side), site_4 can consider other types of analysis for local patient samples, such as SNP identification analysis. The parallel and independent analyses triggered in site_4 can be extended to other sites and combines various data sources to complement the analysis of local heterogeneous samples, such as genomics data and SNP information. The four sites include steps/tasks that are executed in an entirely local

fashion without data sharing, but others involve cooperation.

A cooperation case is presented in four collaboration steps during the analyses that we denoted as C_1, C_2, C_3, and C_4. The collaboration steps share data between the tasks at each site. For example, in the collaboration denoted as C_1, site_1 aggregates the models of the other French sites in the merge models step (MMD1). site_1 can be selected due to having more computing capacity than the other two sites. In another step C_3, site_3 requires the collaboration of site_2 to complete the analysis because site_3 does not dispose of the necessary infrastructure. However, it can share data with site_2 due to agreements established between the parties.

In summary, the proposed scenario involves different tasks between the four sites, which are executed partially sequentially, partially in a parallel fashion between different parts of the analysis workflow. The execution of each task is specific to each site, and the collaboration is achieved by sharing data during the multi-site analysis.

6.3. FeDeRa: Distributed and Declarative Workflow Language

We propose the FeDeRa language (Fully EnricheD languagE for distRibuted Analysis) to specify workflow-based analyses on FDCs. FeDeRa enables, in particular, the definition of workflows between multiple sites. FeDeRa is implemented as a *Python* library that provides a declarative programming language for specifying multi-site analyses. FeDeRa provides control-flow and data flow programming strategies, treating data as a first-class citizen. It also provides special concurrency and distribution mechanisms to support the execution of tasks. FeDeRa implements data flows by means of a shared information store that is accessible by all sites involved in the data flow. A workflow task can provide the data to a participating site that requires it by using a shared variable that will be consumed by said site that requires the data.

The FeDeRa language presents three abstractions to support declarative specifications based on the dataflow during analysis:

- We incorporate future variables to share information between cross-site tasks. The interaction is done through data-driven tasks mediated by a variable shared common to all sites.

- We implement the binding operator that allows to fill data required by others by data-need during collaborative execution.

- Finally, we synchronize through controlled and synchronous waits according to the availability of the data.

These abstractions are made explicit in the language's syntax and are transformed into executable *Python* code. The workflow tasks at different sites are executed independently; local execution does not affect the execution of the other sites. With these abstractions, we seek to provide three capabilities that are very rarely supported by other workflow languages:

1. User-friendly specification through declarative statements focused on non-expert users, notably biomedical engineers and techniciens.

2. Natural transformation from diagram-expressed workflows into FeDeRa syntax without using complex constructions.

3. High level of expressiveness for defining tasks on multiple sites and their dependencies compared to other specification approaches.

Finally, we also consider deadlocks detection since it can occur due to mutual dependency between sites, such as between data flow variables. Therefore, we have harnessed a graph-based strategy for deadlock detection [KS11] to build a global graph representing the tasks deployed across all sites and the dependencies between dataflow variables. This construction is usually feasible in the case of biomedical collaborations due to a mutual agreement between all the participants that typically must be established prior to the cooperation. Furthermore, requiring dependencies to be known in advance allows potential deadlocks to be identified and removed before starting the collaborative analysis.

6.3.1. FeDeRa Language Features

In order to realize the above abstractions and capabilities, we propose a language incorporating dataflow programming and declarative concurrency mechanisms.

6.3.1.1. Dataflow Programming

Workflows almost always define tasks in terms of control, indicating that a task can be executed only after completing the previous task. In contrast, using data-driven flows, task execution is triggered by the data required availability. Therefore, the data-driven approach also fits the definition of multiple concurrent tasks well. Nowadays, the most popular data processing systems based on a data flow approach are those based on MapReduce. However, current control flow-based and dataflow-based approaches are subject to limitations concerning multi-site analyses and do not support geo-distributed analyses [PCA15; Dol+17a]. Therefore, dataflow programming drives the execution through a series of operations and transformations on the data. The data flow can be represented by a directed graph where the nodes represent the operations, and the edges between them show how data flows among operations. For example, a node executes an operation when its input data is available and produces output data for the next node.

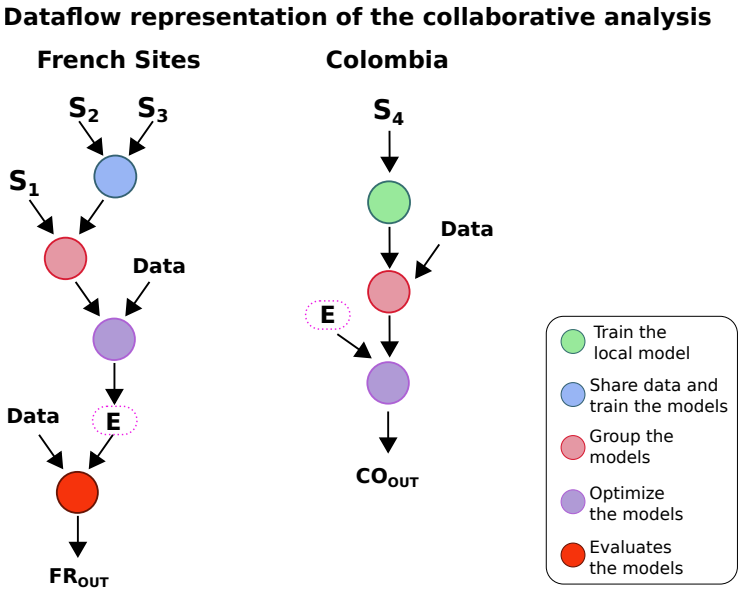


Figure 6.2 – Dataflow representation between French and Colombian sites that share information on one variable E.

Figure 6.2 is a part of the collaborative workflow proposed between France and Colombia (illustrated in Figure 6.1). Figure 6.2 presents a dataflow between multiple sites that execute independent tasks but sometimes require collaboration. The left part of the dataflow corresponds to the French sites and the right side to the Colombian process. The French sites can share data and models as indicated in each step. Some steps are

executed on a specific site according to the installed computing capacity or data-sharing agreements. For instance, the Colombian site requires the variable E, and a French site binds that variable. Once E is bound, the Colombian site continues with the local process. The variable E is a one-time modifiable entity, that is immutable after the initial binding. It is the owner's responsibility to bind its value. In other terms, the variable E is a so-called future that supplies a value for a "hole" on another site.

Dataflow-based tasks may be executed in parallel if each of them does not depend on the output data of the other. This often allows many tasks in dataflow-based workflows to be executed in parallel. This relationship between parallel and dataflow programming enables us to provide mechanisms appropriate for multi-site analysis. Dataflow-based parallelism provides a natural specification means, in particular, for non-expert users.

6.3.1.2. Declarative Concurrency

FeDeRa also includes declarative concurrent programming features. It adopts relevant mechanisms focused on concurrent processing and distributed mechanisms, such as those proposed in the Oz language [HSW93] and its computation model [Con13; VH04]. FeDeRa combines declarative and concurrency techniques seeking to maintain instructions independently, satisfying the needs of each site during multi-site analyses.

Concurrency extends the functionality to perform analyses independently across multiple sites, where they interact only when necessary. The interactions are performed through single-share variables referenced in a common space accessible to all. Dataflow variables are stored on an external site called the *Single-Share Store (SSS)*. These variables have two states: initially, they are undefined and their status is unbound. Next, the status changes to bound when a value is assigned to the variable.

The Single Share Store (SSS)

FeDeRa implements a variable store, a common space to share information between participating sites. Variables in the *SSS* are initially declared and have no initial value assigned. The purpose of these variables is to share the required information between the sites to support multi-site analyses. The *SSS* stores dataflow variables that maintain the unbound (defined) and bound (assigned) states. These variables are assignable only once and then immutable. The site that defines the dataflow variable is the only one responsible for changing its value. The value assigned can correspond to a primitive type, a learning model, a data set, or an encrypted object. The location of the *SSS* store can be fixed

to one of the participating members (that is, for instance, located approximately "in the middle" of all participants) or a third party trusted by the parties.

The *SSS* store can constitute a bottleneck during processing due to thousands of requests from geo-distributed sites, which generate latency and communication problems. Nevertheless, we think that real biomedical problems almost always involve fewer participants than in other areas, such as business or social media. For example, the ICAN project involves 34 participating sites. Our scheme can be improved upon if bottlenecks arise by splitting the *SSS* into several such spaces that manage subsets of the shared variables whose users are, for example, located close to one another. We do not consider this, relatively straightforward, extension further in this thesis.

Futures

Futures are dataflow variables that share information between parties, enabling on-demand data analysis. The future variables are implicitly defined during the workflow specification, and the synchronization is done during the call for sites that require it. The adoption of future variables increase the independence between calculations that require values from other sites and takes advantage of local computing facilities installed at each site.
These features, directly related to declarative and dataflow-oriented concurrency programming, are included in the FeDeRa language implementation to facilitate the specification of biomedical analyses at multiple sites and mitigate some of the specification problems in current workflow systems.

6.3.2. Syntax and Semantics

We define the FeDeRa syntax by extending the provided for *Python*[2]. First, FeDeRa's syntax defines the most important properties of our language. Next, the semantics is defined in terms of transformations from our declarative programming instructions into *Python* standard programs.

6.3.2.1. FeDeRa Instructions

The syntax of the FeDeRa language is presented in Figure 6.3. The grammar is defined using Extended Backus-Naur Form (EBNF) and Parsing Expression Grammar (PEG). The syntax is similar to the *Python* grammar. In particular, the syntax statements in our

2. https://docs.python.org/3/reference/grammar.html

```
//FeDeRa Program statements
FeDeRaProgram := FeDeRaStmt+
FeDeRaStmt := FeDeRaStmtDecl | PythonStmts

//FeDeRa statement declaration
FeDeRaStmtDecl := DFVarDeclaration | DFVarBindings | FlowDecl

//DataFlow Variable Declaration
DFVarDeclaration := "var", varName+
varName := PythonVariableIdentifier

//DataFlow Variable Bindings
DFVarBindings := varName, ":=", value
value := PythonExpression

//Flow and Workflow definition
FlowDecl := "flow", SiteIdentifier, "(",[Paramlist],")", ":", [FeDeRaStmt]
WorkflowDef := "workflow (", SiteIdentifier, {"|", SiteIdentifier}, ")"

//Workflow invocation
StrtInv := WorkflowObj, ".start", "(",[Paramlist],")"
WorkflowObj := WorkflowDef | objectName
Paramlist = Param, {"," Param}
Param := varName | value

//Python statements
PythonStmts correspond to Python grammar
```

Figure 6.3 – Syntax of FeDeRa language.

language follow *Python's* indentation rules[3] to delimit structure, establish code blocks, and take advantage of the benefits of code readability.

The grammar mainly introduces three statements that enrich our language for multi-site analyses: parallel composition of multi-site workflows (represented by the grammar non-terminal *WorkflowDef*), local flow declaration (non-terminal *FlowDecl*), dataflow variables declaration (*DFVarDeclaration*), and a binding operation for dataflow variables (*DFVarBindings*). The parallel composition between multi-site workflows is synchronized through the shared variables implemented in our language.

3. https://peps.python.org/pep-0008/

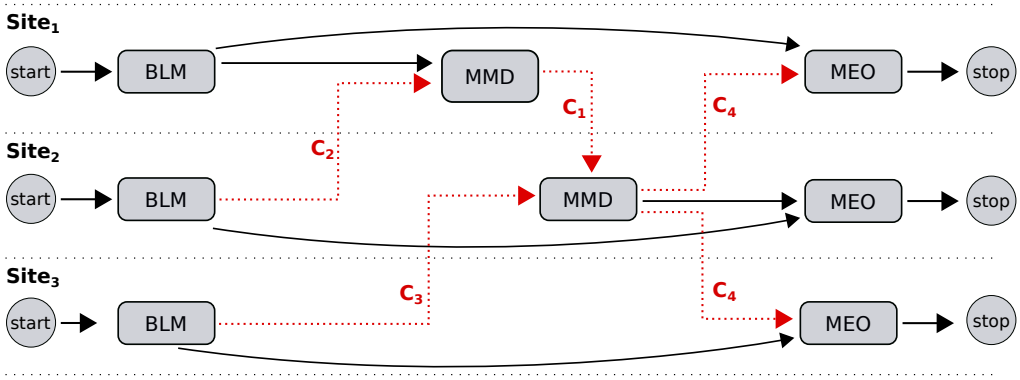


Figure 6.4 – A distributed machine learning scenario among three sites applying the steps: Build a Local Model (BLM), Merge Models (MMD), and Model Evaluation (MEO).

To show how to use the FeDeRa syntax, we use the workflow diagram illustrated in Figure 6.4. This workflow is a part of the collaborative analysis discussed on page 92, which will be defined later. The workflow illustrated in Figure 6.4 represents an analysis among three sites. They share data at different times through dataflow variables (represented by C_1, C_2, C_3, and C_4). The scenario consists of three steps: BLM (Build Local Model) runs entirely local on each site, while the steps MMD (Merge Models) and MEO (Model Evaluation and Outcomes) share data between sites. The step MMD1 requires two models, the local one, from $Site_1$, and the other from $Site_2$. For communication, $Site_1$ requires the future variable C_2, bound by $Site_2$. Therefore, the execution of the MMD1 task must wait until the value of C_2 is assigned.

This collaborative workflow among three sites is specified using FeDeRa's syntax, as shown in Figure 6.5. The specification has two parts, a global part for all the sites and a part specific for each one. First, the global workflow definition provides the information required by all sites, see lines 2–5 in Figure 6.5. The second part corresponds to the algorithm executed on each workflow site. For the case study, three sites are declared separately. In Figure 6.5, the flows on the three sites are defined in lines 9–33.

The languages capabilities offered are used as follows in this context. We define four dataflow variables in line 3. These dataflow variables allow communication between the sites in order to share information required by the parties during the multi-site execution of tasks. On line 4, we start three tasks on three different sites. During the execution, they communicate through passing message patterns and share future variables using a single share store.

Then come three flow definitions for the three sites (lines 9–33). On each site, the **flow**

```
1  # Global workflow specification for all sites
2  flow allSites():
3    var c1,c2,c3,c4                # Define dataflow variables
4    workflow(Site1|Site2|Site3)    # Invoke the multi-site workflow
5    print("Analysis performed among three sites")
6
7  # Local flow specifications for each site
8  flow Site1():
9    dataS1 = loadData()
10   blm = buildLocalMdl(dataS1)
11   mmd = mergeMdl(blm,c2)
12   c1 := mmd    # Bind future c1
13   meolc = evalMdl(blm)
14   meo = evalMdl(c4)
15   print("Model Evaluation Results")
16   print("Local Model",meolc, "Collab M.", meo)
17
18 flow Site2():
19   dataS2 = loadData()
20   blm = buildLocalMdl(dataS2) # Bind future c2
21   c2 := blm
22   mmd = mergeMdl(c1,c3)
23   c4 := mmd    # Bind future c4
24   print("Model Evaluation Results")
25   print("Local Model",evalMdl(blm), "Collab M.", evalMdl(mmd))
26
27 flow Site3():
28   dataS3 = loadData()
29   blm = buildLocalMdl(dataS3)
30   c3 := blm    # Bind future c3
31   print("Model Evaluation Results")
32   print("Local Model",evalMdl(blm), "Collab M.", evalMdl(c4))
```

Figure 6.5 – FeDeRa analysis workflow among three sites from Figure 6.4

statement in form of *Python*-style programs. Finally, the dataflow variables are bound (using FeDeRa's binding operator operator `':='`) on lines 12, 21 and 30.

6.3.2.2. Transformational semantics of FeDeRa instructions

The FeDeRa statements are defined by transforming the FeDeRa's new constructs into executable (standard) *Python* code. The main idea of this semantics is to define the future-based by-need execution among multiple sites.

Figure 6.6 illustrates FeDeRa's compilation process, transforming FeDeRa's proper declarative or *Python* instructions into *Python* executable code. The former is transformed

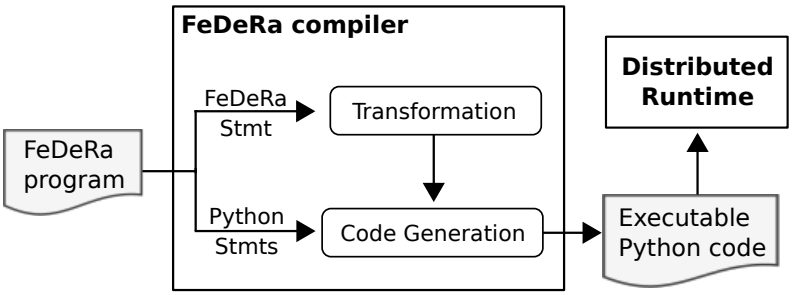


Figure 6.6 – FeDeRa compilation process.

into plain *Python*, as shown in the following; the latter can be interpreted directly. This transformation is achieved through FeDeRa modules developed in *Python* for each FeDeRa component explained below.

The Single Share Store (SSS) is the channel through which sites share information based on the dataflow variables (futures). In the example specification presented in Figure 6.5, line 3 defines four dataflow variables in the *SSS*. Without loss of generality, we assume that such variables must be unique.

```
#Global workflow specification for all sites
flow allSites():
  #Define dataflow variables
  var c1
```

Implementation of ***var*** definition for the variable ***c1*** into Python code

```
1 #ZMQ is a high-performance asynchronous messaging library
2 context = zmq.Context()
3 subscriber = context.socket(zmq.SUB)
4 subscriber.connect("tcp://" + ip_SSS+ ":" + port)
5 subscriber.setsockopt_string(zmq.SUBSCRIBE, "c1")
```

Figure 6.7 – Transformation of Single Share Store functionality

In Figure 6.7, we present the transformation of the *SSS* into *Python* code that uses message passing between the sites and the site responsible for the store. The synchronization of declares and binds operations is through publisher and subscriber patterns. We implement this functionality based on the asynchronous messaging library, *ZeroMQ*[4].

Future assignment (:=) is the second relevant feature of our language. This bind operation is defined in the syntax as *DFVarDeclaration* and *DFVarBindings*. The operator

4. https://zeromq.org/

```
#statement to assign a future varName
varName := value
```

Implementation of the **bind** operator **':='** for the dataflow variable **varName**

```
def wait_needed(varName):
  """Parameters:
    - varName: the name of the future variable
  Returns:
    - bound value: returns the value of the varName variable until
  it is bound somewhere"""

  #ip_SSS: IP address where is located the SSS
  #port: specific port for communication
  subscriber.connect("tcp://"+ ip_SSS+ ":" + port)
  subscriber.setsockopt_string(zmq.SUBSCRIBE, varName)
  topic, msgContent = subscriber.recv_string().split(",")
  if topic == varName: return msgContent
```

Figure 6.8 – Implementation of the subscriber site to the dataflow variable varName as *Python* code

':=' is transformed into the '`wait_needed()`' *Python* function as presented in Figure 6.8. The function subscribes to the site interested in accessing the future named varName. When a site requires the variable and is not bound, the site waits until the variable's state is bound. Otherwise, the subscriber continues the local execution if the future variable is bound.

```
#statement to bind a dataflow variable
call_function(varName,result)
```

Implementation of the **bind** operation for the dataflow variable **varName** with the value **result**

```
def bind_variable(varName,result)
  """Parameters:
    - varName: the name of the future variable to be bind
    - result: the value to assign to the variable VarName
  Returns:
    - assignment success: True or False """

  publisher = context.socket(zmq.PUB)
  #ip_SSS: IP address where is located the SSS
  #port: specific port for communication
  subscriber.connect("tcp://"+ ip_SSS+ ":" + port)
  publisher.send_string(varName, result)
```

Figure 6.9 – Implementation for binding a dataflow variable in *Python* code

The binding operator is implemented using a function '`bind_variable()`' presented in Figure 6.9. The function publishes the `result` in the future `varName`. The binding process is done through the publisher patterns. The future `varName` binding is implicit whenever some function refers to it. For example, the '`call_function()`' implicitly assigns a `varName` the `result` value.

Finally, as we already mentioned, the keyword **workflow** defines the parallel composition of the flows, and **flow** is transformed into executable routines in *Python*. The workflow definition allows deploying the analysis between the multiple sites indicated as arguments. In contrast, the flow corresponds to the executable code composed of *Python* and FeDeRa statements executed on each site.

6.3.2.3. Specification of the Collaborative Scenario

As a second workflow case to specify, we reconsider the collaborative scenario illustrated in Figure 6.1 (on page 160). The corresponding FeDeRa specification is also composed of two parts. The first part corresponds to the global specification illustrated in Figure 6.10. The second part defines once again each site's flow steps, represented in Figure 6.11 for the Colombian and Figure 6.12 for the French sites.

```
1 #Global specification for all sites
2 flow allSites():
3   #define dataflow vars in the single share store
4   var c1,c2,c3,c4
5   #The first three sites correspond to French members, and the rest
      to the process in Colombia.
6   workflow(Site1|Site2|Site3|Site4a|Site4b)
7   print("Analysis performed among the four sites")
```

Figure 6.10 – FeDeRa specification for the global conditions of the collaborative scenario presented in Figure 6.1 (introduced on page 40)

The Colombian and French sites dispose of information derived from genomic data, such as gene expression levels. Moreover, Site_4 processes another independent task that does not depend on third parties, such as analyzing SNP information to complement the analyses of the samples of Colombian patients. For this reason, in Figure 6.1, Site_4 has two local processes, as explained above.

The global specification, presented in Figure 6.10, defines the participation of four sites that compose the collaborative analysis. In the specification, line 4 defines four dataflow

variables used during the collaborative analysis to share data and accomplish the process according to the workflow diagram. Next, line 6 refers to the different process sites, and the last value, Site4b, is the independent process in Colombia that, in particular, processes SNP information.

```
1  #Local specification for Colombian site
2  flow Site4a():
3    dataS4a = loadData()
4    localMdl = buildLocalMdl(dataS4a)
5    groupMdl = groupModel(localMdl)
6    mergeMdl = mergeModes(groupMdl)
7    print("Model Evaluation Results")
8    print("Local Model",evalMdl(localMdl))
9    print("Collab Models",evalMdl(c2)
10
11 flow Site4b():
12   dataS4b = loadData()
13   localMdl = buildLocalMdl(dataS4b)
14   print("Model Evaluation Results")
15   print("Local Model",evalMdl(localMdl))
```

Figure 6.11 – FeDeRa specification for Colombian site in the collaborative scenario illustrated in Figure 6.1

For the specification on the Colombian site, Figure 6.11 defines two parallel workflows. The workflow described in the Site4a routine corresponds to the collaboration with France. All steps are defined using new FeDeRa constructs or standard *Python* code. In this specification, all steps call routines that are defined in *Python*. In contrast, line 9 uses the dataflow variable C_2, bound by $Site_1$. The variable C_2 corresponds to a future to share information.

Finally, the specification of the three French sites is shown in Figure 6.12. They are also composed of calls to routines defined in native *Python* code. For example, lines 7, 16, 23, and 31 refer to the variables C_1, C_2, C_3 defined in the single share store to share required information. The binding operations of these variables occur in lines 9, 19, and 29.

Until now, we have specified two collaborative workflows using our provided syntax. The specification is based on the simplicity provided by *Python*. The workflow illustrated in Figure 6.4 was defined as the FeDeRa code shown in Figure 6.5. Second, the collaborative case introduced at the beginning of the thesis, diagrammed in Figure 6.1 was transformed into the FeDeRa syntax shown in Figures 6.10, 6.11, and 6.12.

In both cases, the specification form from the workflow diagram shows one of the ben-

```
#Local specification for French sites
flow Site1():
  dataS1 = loadData()
  localMdl = buildLocalMdl(dataS1)
  groupMdl = groupModel(localMdl)
  #wait_needed(c1), wait until glm on site2 is bound
  mergeMdl = mergeModes(groupMdl,c1)
  #binding the dataflow variable c2 to mergeMdl value
  c2 :=  mergeMdl
  print("Model Evaluation Results")
  print("Local Model",evalMdl(localMdl))
  print("Collab Models",evalMdl(mergeMdl))

flow Site2():
  dataS2 = loadData()
  localMdl = buildLocalMdl(dataS2,c3)
  groupMdl = groupModel(localMdl)
  #binding the dataflow variable c1 to groupMdl value
  c1 :=  groupMdl
  print("Model Evaluation Results")
  print("Local Model",evalMdl(localMdl))
  #binding the dataflow variable c4
  c4 :=  evalMdl(c2)
  print("Collab Models",c4)

flow Site3():
  dataS3 = loadData()
  #binding the dataflow variable c3 to dataset dataS3
  c3 :=  dataS3
  print("Model Evaluation Results")
  print("Collab Models",c4)
```

Figure 6.12 – FeDeRa specification for French sites in the collaborative scenario presented in Figure 6.1

efits desired with our language, mentioned at the beginning when we define our language (on page 161). The natural transformation from diagrammed workflow into FeDeRa syntax without using complex elements. These specification features are complemented later in the experimental stage, comparing other capabilities of FeDeRa with respect to other workflow specification languages.

6.4. Architecture and Implementation

FeDeRa language implementation supports FDCs with two main features: dataflow-oriented workflows supported by declarative means to handle concurrency. FeDeRa has been implemented using *Python* (version 3.7) and some specialized libraries to achieve the desired functionality mentioned in this section. Our implementation offers high portability, a relevant feature in biomedical analysis since there is no standard platform for distributed analyses.

6.4.1. Implementation overview

The Single Shared Store (*SSS*) supports communication between all sites through the interaction between data flow variables. The *SSS* can be configured and located at an arbitrary site that participates in the analysis or a separated trusted third party. The interaction between the *SSS* and the sites is handled via message passing using an asynchronous messaging library (ZeroMQ[5]). ZeroMQ is a concurrent library that provides brokerless messaging through an asynchronous message queue. In addition, the ZeroMQ library also handles low-level tasks such as managing system-level communication sockets.

Figure 6.13 – Communication between two sites via the Single Share Store *SSS*.

Figure 6.13 presents the interaction with the *SSS* through three sites. First, Site1 defines the variable `varName` to be used as a future for others. Site2 is responsible for hosting the Single Share Store. Finally, Site3 requires the content of `varName` at some point during the workflow execution.

We mainly rely on the Publisher/Subscriber pattern for interactions that use different ports for different variables. The Publisher/Subscriber functionality is implemented using the functions wait_needed and bind_variable introduced in previous section (on page 170). The implementation supports point-to-point communication that is well aligned with the distributed topologies supported by FDCs. Finally, the concurrent functionality is implemented through the `asyncio` library[6], which provides high-level functionality to

5. https://zeromq.org/
6. https://docs.python.org/es/3/library/asyncio.html

ensure asynchronous tasks during the communication.

Handling deadlocks

Distributed processing involves sharing resources in FeDeRa-based FDC systems, such as dataflow variables. At some point, there can be competition for access to these variables, generating deadlocks. For example, a deadlock in FeDeRa specification can occur when a workflow task requires dataflow variables held by another site. Potential deadlocks are unavoidable and difficult to avoid, and they are always latent in distributed and concurrent systems such as FDCs. Therefore, we propose using a well-known deadlock identification employing cyclic dependencies in global directed graphs, known as Wait-For-Graph (WFG) [KS11].

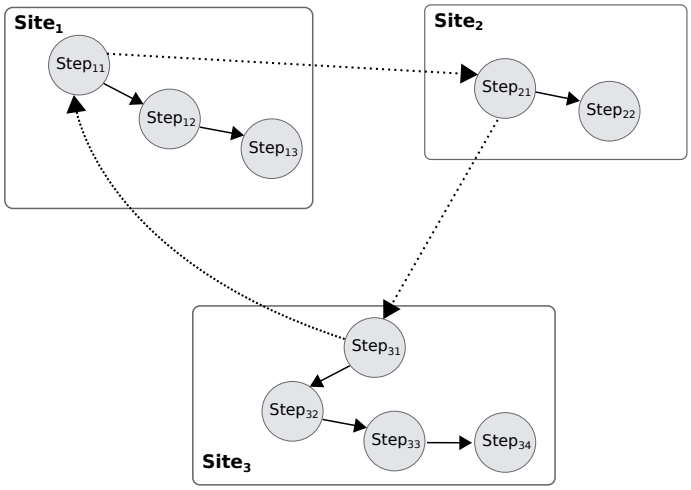


Figure 6.14 – Example of a Wait-For-Graph (WFG) strategy for handling deadlocks between three sites.

This approach is useful in the context of FDCs for biomedical analyses — and typically feasible in contrast to other domains where distributed algorithms are harnessed — because their stringent security and privacy requirements entail that complete knowledge about the system is available at some point. As a consequence, the dependency graph can effectively be constructed.

A multi-site distributed analysis can be modeled through a directed graph whose nodes represent tasks, and the edges represent directed communications between them. Nodes can have two possible states running or waiting. The running status corresponds to the correct assignment of resources to finish the task. In contrast, waiting indicates that the

step is acquiring some future value. The form of representation of the WFG corresponds to the indicated in the literature handling deadlocks using graphs [Sin89].

Figure 6.14 shows a WFG graph composed of tasks at three sites, where the nodes are tasks, and the edges represent the wait state for the resource. For instance, the directed edge from node Step_{11} to node Step_{21} means Step_{11} is blocked and is waiting for Step_{21} to bind a variable. WFG representation allows identifying if the system has deadlocks as long as there is a cycle between nodes. For example, Step_{11}, Step_{21}, and Step_{31} form a cycle. Therefore, these steps form a deadlock. Algorithms, for instance, that are presented in [KS11], must satisfy two conditions:

I. detection of deadlocks in a finite time, avoiding the occurrence of new deadlocks,

II. ensure that it is a real lock and does not correspond to false deadlocks.

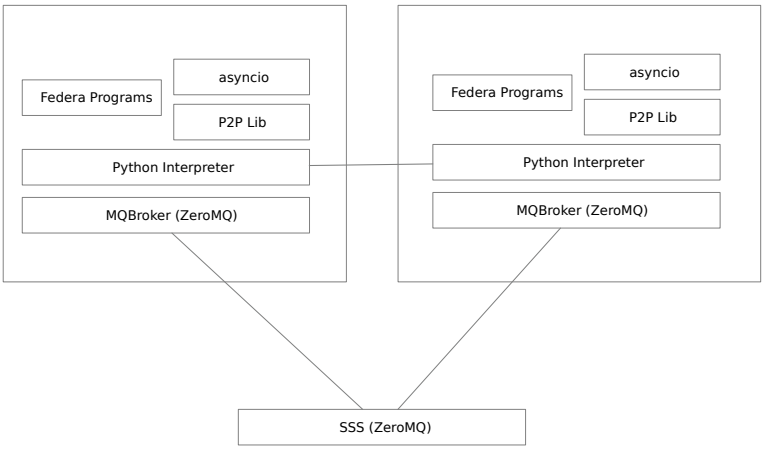


Figure 6.15 – FeDeRa distributed runtime architecture.

6.4.2. FeDeRa Runtime Architecture

The FeDeRa runtime architecture supports the execution of multi-site analyses collaboratively and coordinatedly. Figure 6.15 illustrates the distributed runtime architecture of FeDeRa and its handling of interactions between multiple sites. At each site, the implemented FeDeRa code is deployed, and the specialized libraries to achieve local execution and synchronization between sites are defined. In the distributed runtime architecture, the sites communicate through the availability of variables hosted in the Single Share Store (*SSS*). The *SSS* is initialized in one site that communicates with all to share future data.

The communication between sites can be point-to-point when the sites require some direct communication. The services implemented in the architecture support multi-site execution of the analyses specified with the FeDeRa language.

The mentioned libraries in the runtime architecture correspond to specialized tasks supported in *Python*. For example, the message queue on each site is implemented through the `ZeroMQ` library. Point-to-point communication is performed through the `p2pnetwork` library[7]. It supports peer-to-peer connection and the modeling of dynamic networks, such as the workflows supported by FDC systems.

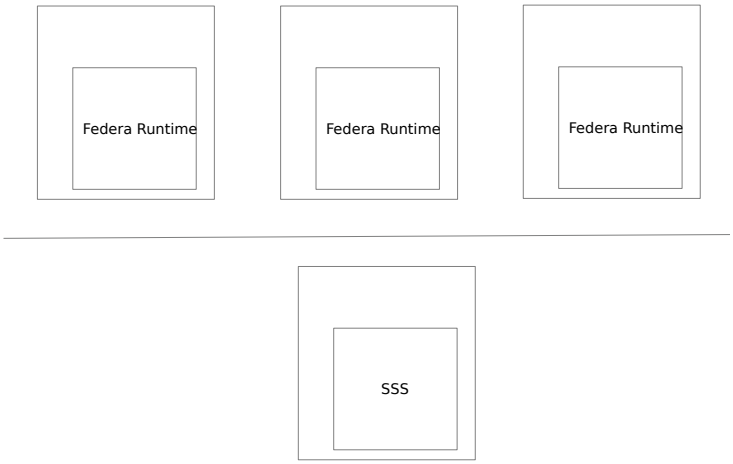


Figure 6.16 – FeDeRa communication distributed architecture.

Finally, Figure 6.16 corresponds to the distributed architecture between the sites, where the *SSS* can be located at one of the sites participating in the process or even at trusted third party during the execution of the distributed analysis.

6.4.3. Deployment

We are interested in deploying FeDeRa in multi-cloud environments over distributed sites in order to support FDC analyses. Its implementation library can be deployed on an arbitrary number of sites. After installation of some configuration files and the setup and parameterization of the single share store on a trustworthy site, FDC applications are ready to be executed.

7. https://github.com/macsnoeren/python-p2p-network

Deploying on Grid'5000 (*G5K*)

Following the deployment mentioned in the previous chapter in Section 5.5.2 (on page 139), FeDeRa-specified analyses were also deployed in the Grid'5000 environment. The deployment is automated and controlled through the `execo` library. This library supports the automated execution from the local console on hundreds of nodes deployable on *G5K*.

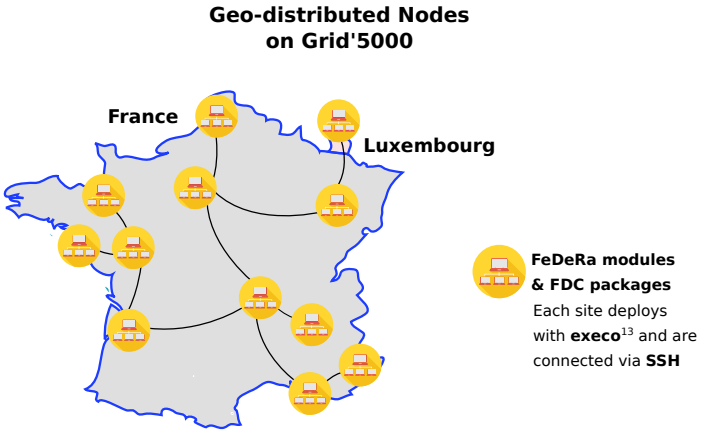


Figure 6.17 – Grid'5000 (G5K) nodes distribution with FeDeRa and FDC components.

We first initialize each node required during the multi-site analysis from the specification in FeDeRa. Next, we set up the geo-distributed environment based on the nodes specified. We then move and install the FeDeRa and FDC components on each node. Finally, we start the execution according to the multi-site workflow specification. Figure 6.17 presents the distributed deployment of the modules and FeDeRa components installed in each site, where they can communicate point-to-point using the `networkx` library.

```
<DeployConfig>
  <ProviderName>G5K</ProviderName>
  <NoSites>8</NoSites>
  <SiteConfig>
    <Site>Nantes</Site>
    <NoNodes>2</NoNodes>
    <OS>ubuntu1804-x64-min</OS>
    <Requisites>python3.7,scikit-learn,pandas</Requisites>
  </SiteConfig>
  ...
</DeployConfig>
```

Figure 6.18 – Sample of configuration file to deploy multi-site analyses on *G5K*

Environment settings

The deployment is handled mainly by means of a configuration file that enables customizing the relevant site data, such as the arrangement of nodes and the configuration of the machines involved in the multi-site analysis. On each node, it is possible to indicate the operating system and the required libraries. The environment is first initialized at each site, and then the base software is installed. Finally, each site executes the specified workflow that implements the FDC. Figure 6.18 presents a part of the configuration file (in XML format) for deployment between multiple sites members of *G5K*. This per-site file may define the following fields:

- *ProviderName*: is the name of the environment to implement the distributed analysis. Until now, we only have considered *G5K*, but other geo-distributed environments could be considered an extension of the language.
- *NoSites*: total number of sites on which the FDC is deployed.
- *Site*: name or ID of each site (in our case, the cities providing *G5K* clusters).
- *NoNodes*: number of nodes to be deployed on each site.
- *OS*: operating system to set up and configure on each node.
- *Requisites*: list of desired packages and libraries to install on each node.

The configuration file is required prior to the deployment of the analysis. This setup reduces the required technical knowledge for deploying current multi-site analyses in distributed processing environments such as *G5K*. Current analyses, such as those based on distributed workflow systems, require intricate specialized code to achieve similar deployment types. In this way, our language offers another advantage, reducing the complex technical knowledge of multi-site deployments required in traditional workflow systems.

6.5. Evaluation and Results

In this section, we evaluate the new features of FeDeRa by comparison with the three baseline workflow languages Swift, Snakemake, and Pegasus, mentioned in this chapter on page 157. We specify a workflow common to all and then present a qualitative evaluation of the abstractions provided by our language versus the specification of the three baseline systems. We focus on three properties:

- **Expressiveness** refers to the capability of how natural and concise multi-site analyses can be defined with the new language mechanisms.
- **Concurrency** strategies to specify multi-site tasks in the presence of data-flow dependencies during collaborations among sites.
- **Distribution** mechanisms to specify the use and share of resources required for processing between multiple sites.

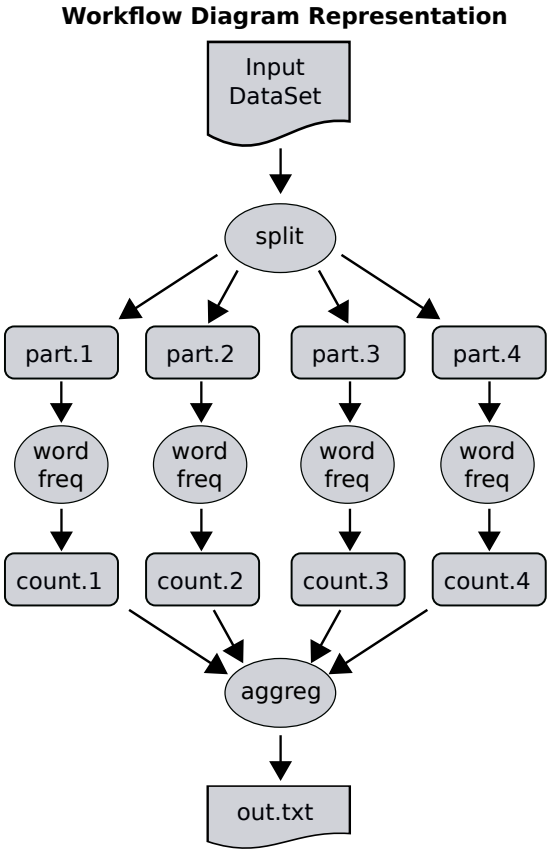


Figure 6.19 – Workflow to split a single input into four independent files on which word frequencies are calculated output as a single file.

The comparison among FeDeRa and the three baseline workflows employ the same workflow analysis common to all, represented in Figure 6.19, and specified separately in each syntax provided. The workflow splits an input file into several independent tasks. Later, the number of frequencies of the words is counted in each file. Then the frequency counts are returned and aggregated into a single output file. This workflow can be used to

parallelize across multiple compute resources, allowing multiple jobs to run independently. For example, processing large data sets or sharing trained models across multiple sites, such as the FDC-based learning approach.

```
type file;

app (file out) split_job (string FInput)
{
  split "--input" FInput stdout=filename(o);
}

app (file out) wordfreq_job (file part_file)
{
  wordfreq "--input" part_file stdout=filename(out);
}

app (file out) merge_job (file s[])
{
  aggregate filenames(s) stdout=filename(out);
}

file FInput <"InputDS.txt">;
file outputs[];

foreach i in [0:3] {
  file FPart <single_file_mapper; file=strcat("output/part.",i)>;
  FPart = split_job(FInput);

  file FCount <single_file_mapper; file=strcat("output/count.",i)>;
  FCount = wordfreq_job(FPart);
  outputs[i] = FCount;
}

file out<"output/output.txt">;
out = merge_job(outputs);

```

Figure 6.20 – Specification in Swift for the workflow analysis presented in Figure 6.19 .

Figure 6.20 presents the specification in Swift for the workflow illustrated in Figure 6.19. The specification presented is from the documentation provided by the Swift language workflow[8]. The syntax is based on scripting language similar to C-syntax, and each statement acts as a "shell" language instruction, a syntax highly suitable for developers. However, this is a disadvantage for non-technical users, such as biomedical researchers,

8. https://github.com/swift-lang/swift-tutorial

because learning the syntax can be complex compared to other languages, such as FeDeRa. The complexity of learning the Swift language can be a barrier for non-technical users [Ahm+19]. In addition, another drawback is the complex configuration and technical knowledge required to implement analyses in distributed environments using Swift.

```
inFiles = "data.1 data.2 data.3".split()

rule all:
  input:
    "results.txt"

rule count:
  input:
    "{myFile}.txt"
  output:
    txt="{myFile}.count"
  shell:
    "cat {input} | wc -w > {output}"

rule merge_job:
  input:
    expand('{f}.count', f=inFiles)
  output:
    "results.txt"
  shell:
    "cat {input} > {output}"

```

Figure 6.21 – Specification in Snakemake for the workflow analysis presented in Figure 6.19 .

Figure 6.21 corresponds to the specification in Snakemake for the workflow presented in Figure 6.19. The specification presented is also from the documentation provided on the Snakemake website[9]. The specification shows the essence of the Snakemake syntax, the rule-based Snakefile. The workflow defines the analysis of the data from rules, composed of a name, input files, output files, parameters, and the shell responsible for generating the output from the input. The syntax provided by Snakemake is based on *Python*, which makes the language easy to write, read and maintain. However, the relationship between the rules can be difficult for complex analyses since the connection between the input and output of the rules directs the workflow execution. In addition, if the analysis requires more than one Snakefile, the biomedical engineer is responsible for correctly designing

9. https://snakemake.readthedocs.io/

the workflow among requirements between rules in multiple files, dealing with the dependencies between them and the particularities of the execution of each one. Therefore, this creates difficulties in deploying complex analyses such as multi-site and distributed analyses supported by the FDCs.

```
from Pegasus.DAX3 import *
# Create a abstract DAG
dax = ADAG("sample-workflow")

#Add input file to the DAX-level
FInput = File("InputDS.txt")
dax.addFile(a)
#Add a remote or local split job
split_job = Job(name="split")
split_job.addArguments("-l 1",FInput,"part.")
split_job.uses(FInput, link=Link.INPUT)
dax.addJob(split_job)

#Define merge job to aggregate each counts
output_file = File("output.txt")
merge_job = Job(name="aggregate")\
            .add_outputs(output_file)
dax.add_jobs(merge_job)
outputs = []

for f in range(1,5):
  #Generate each part files
  part = File("part.%s" % f)
  split.uses(part, link=Link.OUTPUT)
  FPart = File(part)
  dax.addFile(FPart)

  ##Add a remote or local word frequent job, one for each input file
  wordfreq_job = Job(name="wordfreq")
  FCount = File("count.%s" % f)
  wordfreq_job.addArguments(FPart, FCount)
  wordfreq_job.uses(FPart, link=Link.INPUT)
  wordfreq_job.uses(FCount, link=Link.OUTPUT)
  outputs.append(FCount)
  dax.addJob(wordfreq_job)

#Merge jobs with the counts files
merge_job.add_inputs(outputs)
dax.write()

```

Figure 6.22 – Specification in Pegasus for the workflow analysis presented in Figure 6.19 .

The last baseline workflow specification is based on Pegasus, presented in Figure 6.22. The specification assumes execution on a single site. Therefore partitioning and file merging can be parallelized on-site. The specification presented is based on the documentation provided by Pegasus[10]. Pegasus abstractly describes the workflow using a DAX (Directed Acyclic Graph in XML) format based on diverse programming languages such as *Python* and *Java*. Pegasus supports dependency management of complex tasks through explicit configuration, e.g., a user-produced file in XML format. Pegasus has taken a more explicit approach to distribution and supports running on single machines, remote clusters, distributed infrastructures, and clouds. Therefore, it is usually optimal for execution in HPC environments. However, the specification form for multi-site analysis is complex, as the configuration for multi-site deployments requires detailed technical knowledge.

```
#Global specification for all sites
flow allSites():
  #Define futures variables to receive data from all
  var count.1, count.2, count.3, count.4
  FInput = load("InputDS.txt")
  FPart = split(FInput)
  workflow(Site0|Site1|Site2|Site3).start(FPart[0],FPart[1],FPart[2],
   FPart[3])

  #Merge jobs with the counts files
  res = merge_job.add_inputs(count.1, count.2, count.3, count.4)
  genResults("output.txt",res)
  print("Analysis performed among four sites")

flow Site0():
  #Invoke a local function to compute word frequencies
  count.1 := wordfreq(FPart.0)

flow Site1():
  count.2 := wordfreq(FPart.1)

flow Site2():
  count.3 := wordfreq(FPart.2)

flow Site3():
  count.4 := wordfreq(FPart.3)

```

Figure 6.23 – Specification in FeDeRa for the workflow analysis presented in Figure 6.19 .

10. https://github.com/pegasus-isi/

Finally, we specify the same workflow using FeDeRa, illustrated in Figure 6.23. Our specification assumes that a global site is responsible for splitting the files, sending these to each site, and then receiving them back for aggregation. We share the information between the four sites and the aggregator through future variables, defined on line 4, and each is filled in within each site. The execution on line 10 follows the natural execution of the workflow after each site sends the information about the local count. Finally, this information is combined in a single output file. The routines, split, wordfreq, and genresults are external scripts specialized in a particular task, similar to invoke in each rule in Snakemake. This specification provides for a natural formulation using the features of our language, the future variables, their declaration, and filling during the workflow execution.

6.5.1. Expressiveness Level

FeDeRa seeks to offer a high level of expressiveness and ease of use than current workflow languages. We evaluate this property based on two principles proposed by Kiepusewski et al. [Kie03] to achieve a good modeling language:

- **Suitability** is related to the level of knowledge the user is required to possess in order to specify the workflow steps, that is, how natural it is to specify programs with the syntax provided.
- **Expressiveness** is the ability to provide domain-specific notations and constructs through language specification.

Due to the diversity of workflow systems, there is no standard way to compare them; they lack standards in the design, specification, and execution of workflows. Therefore, our evaluation corresponds to a subjective assessment, as used also by other authors [KTB00; WWG21]. A means to evaluate the expressiveness level can be to determine the suitability of the language's mechanisms versus the concepts required from the problem domain. In our case, transforming graphical workflow diagrams into workflows is the most natural procedure: FeDeRa directly supports this method because it does not require complex knowledge of the language or computation to realize such transformations. Only the three language abstractions we have introduced and *Python* code blocks are required.

The level of expressiveness is obvious from the different scenarios shown in this chapter. First, the collaborative scenario illustrated in Figure 6.4 and expressed using FeDeRa in Figure 6.5 shows a natural way to transform the graphical workflow into FeDeRa syntax.

Second, the split/merge workflow specified in the three baselines workflows (respectively presented in Figures 6.22, 6.21, 6.20) shows the benefits of our specification form. For example, in contrast to Swift, FeDeRa allows to define the steps more intuitively without requiring programming concepts like in *C* language. Similarly, with respect to Snakemake, although we have a similar syntax based on *Python*, the relationships between rules and their requirements can be a problem. This problem is if we consider multi-site analysis, where the manipulation of Snakefiles and its dependencies often become very complex. Finally, regarding the Pegasus scenario, the simplicity of writing is similar. However, communicating tasks between multiple sites requires complex middleware configurations, which is highly complex to perform for non-technical users.

Therefore, our proposal uses futures between the sites frees users from responsibilities since the Single Share Store coordinates and mediates the data during workflow execution. This type of future sharing is not provided in current workflow systems, which require previously defining resources such as data in each site. In addition, our functionality allows for more comfortable sharing data (i.e., raw data or learning models) between multiple sites, improving the ability to specify and configure versus others. Finally, the level of expressiveness and abstraction provided allows the creation of reusable, scalable, and flexible modules, which is highly convenient for biomedical users. Although the level of expressiveness is one feature provided in our proposal, there are other aspects to consider since the configuration required to deploy multi-site analysis is also relevant, especially for the thesis scope, our interest in processing data in FDCs. Overall, our approach allows for a more efficient and understandable multi-site workflow design for biomedical users.

6.5.2. Concurrency and Distribution

The last two features desired with our language are concurrency and distribution mechanisms. According to concurrency, our specification supports using data flow variables that can be executed in different sites, and, in the end, the analysis preserves the final result. Therefore, the multi-site task specification allows for adding concurrency mechanisms without stressing about data races between sites. Instead, each site interacts with others through the controlled passage of messages through the single share store. Additionally, the scenarios specified in our language, including the collaborative analysis between Colombia and France (presented on page 40), show an example of the use of futures supporting concurrent operations such as splits and joins.

Distributed processing is favored by some of the benefits of the mechanisms provided in

the language. FeDeRa's syntax and its mechanisms provided for deployment in multi-site environments allow controlled execution of multi-site analyses where sites communicate with each other by the need of data. The specified analyses can run in distributed environments where each site can have a different configuration. The concurrency mechanisms are aligned with distributed processing between sites. Our multi-site specification considers the distribution of tasks, resources, and data sharing through a common space. Each site can have its local resources, even those different from others. FeDeRa supports flexible and hybrid designs where some tasks can be decentralized and others entirely localized.

Nevertheless, a limitation in the availability of distributed resources can generate deadlocks. One way to manage this is to build a global graph (Wait-For-Graph explained on page 175) from the steps of all sites, including the required resources at each one. From this graph, cycles, and deadlocks are previously calculated and then corrected before starting the analysis. Therefore, our language provides mechanisms in favor of distributed processing, such as distributed computation models, distribution of declarative data, and commonly distributed processing architectures.

Finally, the controlled deployment of multi-site analysis in the *G5K* deployment case shows the ease of supporting FDCs, which require a simple configuration file (explained on page 179). We also offer a model for controlled deployment across multiple *G5K* sites to complement our language.

6.6. Conclusion

This chapter presents some workflow systems and their forms to specify distributed analysis, especially focused on expressiveness, concurrency, and distribution mechanisms relevant to multi-site analysis. From our survey, we identified shortcomings in such mechanisms. For example, Snakemake defines the analysis using association rules in text files, limiting multi-site analysis due to multiple dependencies between those rules. Swift offers a C language-based syntax, which is hard to understand, nor does it make it easy to distribute the analysis across multiple sites. Pegasus supports multiple distributed environments but is limited in specifying and executing workflows or parts of them across different geo-distributed distributed sites. Finally, these systems lack mechanisms that facilitate easy deployment in cloud environments, often requiring technical knowledge from the user to achieve such deployment.

Based on these shortcomings, we propose FeDeRa, a language to specify multi-site

analyses. The FeDeRa language also provides abstractions to fine-tune the deployment across distributed machines. FeDeRa enables, in particular, the definition of workflows between multiple sites. FeDeRa is implemented as a *Python* library that provides a declarative programming language for specifying multi-site analyses. FeDeRa provides control-flow and dataflow programming strategies, treating data as a first-class citizen. Our language is complemented with two features: dataflow programming and declarative concurrency.

FeDeRa uses specialized libraries to ensure data sharing between sites through variable dataflow, using message-passing patterns, and achieving point-to-point communication between each site. The syntax offered in FeDeRa is transformed into executable code in *Python*, which is deployed in a controlled way on each site. The architecture provided is flexible and aligned to multi-site analysis, like those previously presented in FDC systems.

Finally, we present a qualitative evaluation of our approach to specifying distributed analyzes versus other workflow systems based on three properties: expressiveness, concurrency, and distribution. By comparing three current systems, we identified benefits in our language in each property, which are appropriate for biomedical analysis and their participants. The mechanisms provided in our language make it easy to specify analyses and deploy them in multi-site environments.

Chapter 7

Conclusions

This thesis addresses the problem of designing, implementing, and deploying information systems to analyze massive amounts of biomedical data using a globally distributed infrastructure while respecting data ownership and privacy. Concretely, we argue that due to the amount of biomedical data, and the legal and socio-economic restrictions on these data, scientific analyses can no longer be done by individuals or single organizations. Instead, researchers need to design Fully Distributed Collaborations which are research endeavors that harness means to exploit and analyze massive biomedical information collaboratively over geo-distributed infrastructures. However, in their current state, computer tools are not prepared to support such collaborations. In particular, they do not provide means to address legal and socio-economic restrictions on data. In most cases, researchers are forced to request full access to the data of all parties involved in the collaboration, and they design their experiments under this assumption.

In chapter 2, we first investigated the issues and challenges of collaborative research on biomedical data. We studied different types of biomedical data and the restrictions that apply to each. We also summarized the global landscape of current data-sharing regulations, which determine how and when to share personal data, including biomedical information. We also presented biomedical projects that involve collaboration between geo-distributed teams, their restrictions, and the opportunities associated with the current tools for biomedical analysis, such as workflow systems. Finally, we presented in detail a collaborative scenario involving the participation of sites in Colombia and France. This scenario is used during the thesis development to present and discuss our contributions.

In chapter 3, we presented the state of the art of Biomedical Analytical Tools and Techniques. First, we investigated how these analytical tools were used to solve biomedical problems. In particular, we investigated how they were used to solve five relevant problems during the DNA sequencing process. We then categorized a subset of these tools, namely workflow systems, into three categories: the specification of tasks, the ability to reproduce experiments, and interoperability properties. Finally, we studied them

from the perspective of three architectural features: data and computation placement, privacy and security, and architecture and distribution level. During the investigation, we found limited support to address the needs and requirements of FDCs.

From the revision of the literature, we identify several research opportunities for the distributed processing of biomedical data [Gar+22]. Current tools and systems for distributed genomic analyses have several limitations, including implementing efficient heuristics for the placement of computations and data according to the restrictions specific to each experiment. Similarly, regarding privacy and security, most current systems do not provide native means to address security issues and delegate responsibility to the computing infrastructure (e.g., the cloud infrastructure). Finally, the scalability and performance quality attributes have limited support to configure and execute complex distribution patterns over heterogeneous infrastructure and across data privacy and ownership boundaries. They use efficient frameworks to analyze data in distributed environments but neglect biomedical, legal, and socio-economic considerations.

Thus, in Chapter 4, we introduce the concept of Fully Distributed collaborations (FDCs), which constitute collaborative research endeavors where the confidentiality and ownership of biomedical data are satisfied while providing means to analyze and exploit the information collaboratively. FDC-based analyses take advantage of local computing facilities at each site, minimize security risks in sharing data, and favor global biomedical data restrictions. FDCs promise to enable more powerful biomedical analyzes defined in distributed workflows operating over large volumes of shared public and private data. Three topologies complement our collaborative approach for distributed processing based on sequential, hierarchical, and fully distributed architectures. We also extend our FDC approach with two security strategies for sharing data when it is necessary to share data securely between the parties. The first is to share data in secure containers, and the second is to share and operate over the encrypted data, maintaining the data encrypted. The FDC features mitigate technical, legal, and socio-economic constraints by adopting secure mechanisms, expressive definition languages, and a rich set of architectural features that promote reliable collaborations.

Chapter 5 proposes a biomedical analysis algorithm compliant with the FDC requirements and based on Random Forest. The scenario considers a novel implementation of the Random Forests algorithm in a fully distributed multi-site setting. Our fully distributed random forest algorithm (MuSiForest) offers strategies to use private data to train models at each site without sharing raw information and aggregate and optimize them accord-

ing to the distributed processing topology. MuSiForest shows better results regarding the shared information size and the required computation infrastructure. The performance learning rate is close to but lower than centralized versions of the Random Forest algorithm. We also present strategies to improve the performance of the MuSiForest model by sharing partial information securely. Allowing the algorithm to share partial information produce models with less bias than the initial proposal of not sharing data between sites. We experiment with geo-distributed data corresponding to gene expression levels for five types of cancer. The goal is to build a global model for prediction from the aggregation of privately trained local models at each site. Our algorithm was efficient in computing overhead and amount of shared data, and our learning performance was close to expensive and unfeasible centralized training.

Finally, Chapter 6 presents a language to specify fully distributed collaborative analyses in a declarative way. Our language combines FeDeRa and *Python* statements, where the need for the data drives the analysis. FeDeRa is a language to support FDCs with two main features: declarative and concurrent, extending *Python* with share assignment variables and dataflow variables as futures. The language provides instructions that are transformed into executable code to be interpreted in *Python*. The language was tested against three current workflow systems to assess the level of expressiveness and distributed and concurrency mechanisms. Our experiments found that the syntax and way of specifying and deploying multi-site analysis are appropriate for biomedical users in contrast to the technical knowledge required in current systems to configure and deploy analysis in multi-site environments. The results also show that workflows are defined more simply and naturally than the other workflow languages. For example, a notable reduction in the number of instructions to specify analysis with our proposal compared to others shows expressiveness benefits. As result, we offer a user-friendly approach, eliminating the complex practices that current systems provide for specifying multi-site analyses.

7.1. Future work

The results of this dissertation reveal opportunities for future research in biomedical data analysis in geo-distributed environments. Here we discuss possible future works in three categories: the implementation of fully distributed algorithms for data analysis, automated compilation from FeDeRa into *Python* code, more expressiveness in workflow languages, and secure sharing of biomedical data.

Fully distributed processing algorithms

The global restrictions on sharing biomedical data motivate research on more fully distributed processing algorithms, such as those based on distributed and federated learning. In this thesis, we presented a strategy based on vertical data partitioning. However, as a complement, new approaches can be based on horizontal and hybrid data partitions. In addition, combining other data sources on the same samples can reveal more findings in the study of each patient, such as jointly processing images and genomic data.

As a future implementation, other FDC scenarios can consider the composition of other local models (classifiers) resulting from algorithms such as Support Vector Machines [Hea+98], K-Nearest Neighbor [CH67], and Neural Networks [JMM96]. This composition generates different aggregated models, which could improve the precision due to the training strategy of each algorithm. This idea would seek to combine the benefits of each classifier to train a global model that best approximates the performance of centralized training.

Finally, during the prediction process, a weighted voting mechanisms strategy could be implemented [WFB13; Zhu+18] in contrast to our approach, where all trees have the same weight during the prediction. The weight could be determined by the level of contribution of each site. For example, the weight used during the global model could be determined based on the number of samples from each site. However, the collaborative model can explore other statistical measurements to find the best voting mechanism during the prediction phase.

Improve FeDeRa functionalities

We have applied the transformation manually for each of the FeDeRa statements into *Python* code. In future work, we propose the automated generation of executable code from our declarative specifications. To achieve this, it must be necessary to define the automatic compilation with the validation of our syntax and the correct transformation of our instructions into native code to be executed by the *Python* interpreter. In addition, we propose the automated implementation of deadlock handling mechanisms as future work. This implementation will permit early detection of deadlocks between resources required from multiple sites.

Moreover, improving the expressiveness of language mechanisms in terms of supporting more distributed processing environments is necessary. For example, we implemented

a module to deploy automated geo-distributed analyses in Grid'5000 (*G5K*), but it is important to propose other scripts for automated deployment in popular cloud environments. Finally, we consider to improve our specification forms to have more expressiveness, such as designing graphical user interfaces, but without requiring the configuration of specialized routines through the interface, as in traditional workflow systems.

Secure sharing of biomedical data

As evaluated in state of the art, current tools and workflow systems partially address security and privacy issues. They either delegate security to the computing infrastructure (e.g., a cloud provider) or provide basic mechanisms. Therefore, it is necessary to improve the ways to protect the data during processing during all analysis phases, from the collection to results generation. These efforts should be focused on multi-site analysis, where recently, there have been more and more restrictions to sharing biomedical data. Furthermore, the analysis must consider processing encrypted data using complementary techniques to the costly homomorphic encryption. Such as training models from encrypted data focused on global supervised services where consumers send encrypted samples, and the model is also applied in an encrypted way [Bos+14]. The idea is to extend various models through machine learning as a service approach [Taf+17].

GLOSSARY

bagging considers weak classifiers that are trained in a parallel way independently and then combined using some aggregation strategy. 70, 71, 74, 112–114

BATTs is an abbreviation for Biomedical Analytical Tools and Techniques. 45

bias results from making incorrect assumptions, which leads to a model trained under the wrong premises. 23, 122

bootstrapping is a statistical sampling method based on random sampling with replacement focused on improving learning algorithms' diversity and accuracy. 15, 70, 71

CAP theorem indicates that it is impossible in a distributed system to guarantee at the same time: Consistency, Availability, and Partition tolerance. 85, 86

classification refers to supervised learning problem where the training dataset relates independent and output (label) variables. 19, 23, 73, 74, 109, 110

decision trees are a supervised learning method used for classification and regression tasks based on a hierarchical structure. 72, 73

Distributed Machine Learning is a strategy to mitigate algorithm complexity and memory limitation through large-scale distributed processing.. 24, 78, 81–83

ensemble learning is a machine learning strategy where multiple models are trained to solve a classification problem and then combined into one model to be used in prediction tasks. 15, 70–72, 77, 98, 152

entropy is the amount of noise or disorder in a data set. 105, 106, 113

expressiveness is the ability to provide domain-specific notations and constructs through language specification. 23, 162, 180, 185

federated Learning is a distributed machine learning technique to train models across multiple decentralized mobile maintaining data on site. 78, 101, 129, 152

Fully-Distributed Collaborations (FDCs) is a research endeavors where the confidentiality and ownership of biomedical data are satisfied while providing means to analyze and exploit the information collaboratively. 43, 83, 85

gene expression level determines how genes are transcribed into functional gene products such as functional RNA species or proteins. 27, 107, 138, 139

General Data Protection Regulation (GDPR) is the strictest privacy and security regulation for data sharing in the world. 22, 27, 29

Genome-Wide Association (GWA) is a research approach to identify genomic variants that are statistically associated with a particular disease.. 28

genomic data is a set of genomes and DNA data of an organism and the resulting analyzes from these. 28, 29, 158, 160, 171

gini impurity is the probability of incorrectly classifying a randomly chosen sample if it was randomly classified according to the class distribution. 105, 106, 113

ICAN project is the IntraCranial ANeurysms (ICAN) project aims to develop diagnostic and predictive tools for the risk of intracranial aneurysms. 9, 27, 36–39, 43, 97, 98, 165

Information Gain is the amount of information gained by each attribute to determine the tree's levels by its information gained. 105, 106, 126

machine Learning involves artificial intelligence methods allowing organizations to infer knowledge from data autonomously. 46, 66, 83, 92, 96

MapReduce is a framework that supports parallel computing on large volumes of data distributed between processing workers. 79, 85

MuSiForest is a our approach for MUlti-SIte random Forest. 23

Next-Generation Sequencing (NGS) refers to large-scale DNA or RNA sequencing technology to study diseases and their biological relationships. 21

overfitting is when the trained model perfectly fits the training data but cannot predict unknown/future samples satisfactorily. 74

Random Forests are an ensemble machine learning algorithm composed of individual decision trees. 22

regression refers to supervised learning problem where the training dataset relates independent and output (continuous) variables. 74

suitability of the language is related to the level of knowledge required by the user to specify the workflow steps. 185

underfitting refers to the poor generalization of the model from the training data, the inferred function does not satisfactorily capture the relationship underlying the data. 74

BIBLIOGRAPHY

[Aam+13] Harald Aamot et al., «Pseudonymization of patient identifiers for translational research», *in*: *BMC medical informatics and decision making* 13.*1* (2013), pp. 1–15.

[Aba+16] Martin Abadi et al., «Deep learning with differential privacy», *in*: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 308–318.

[Abu+16] José M Abuín et al., «SparkBWA: speeding up the alignment of high-throughput DNA sequencing data», *in*: *PloS one* 11.*5* (2016), e0155461.

[AÇ15] Anas Abu-Doleh and Ümit V Çatalyürek, «Spaler: Spark and graphx based de novo genome assembler», *in*: *2015 IEEE International Conference on Big Data (Big Data)*, IEEE, 2015, pp. 1013–1018.

[Act96] Accountability Act, «Health insurance portability and accountability act of 1996», *in*: *Public law* 104 (1996), p. 191.

[ADW10] Francisco Azuaje, Yvan Devaux, and Daniel R Wagner, «Integrative pathway-centric modeling of ventricular dysfunction after myocardial infarction», *in*: *PLoS One* 5.*3* (2010), e9661.

[Ahm+19] Azza E Ahmed et al., «Managing genomic variant calling workflows with Swift/T», *in*: *PloS one* 14.*7* (2019), e0211608.

[AIG12] Mohamed Abouelhoda, Shadi Alaa Issa, and Moustafa Ghanem, «Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support», *in*: *BMC bioinformatics* 13 (2012), pp. 1–19.

[Aka+22] Adi Akavia et al., «Privacy-preserving decision trees training and prediction», *in*: *ACM Transactions on Privacy and Security* 25.*3* (2022), pp. 1–30.

[Alm+12] Jonas S Almeida et al., «Fractal MapReduce decomposition of sequence alignment», *in*: *Algorithms for Molecular Biology* 7.*1* (2012), pp. 1–12.

[Alt+90] Stephen F Altschul et al., «Basic local alignment search tool», *in*: *Journal of molecular biology* 215.*3* (1990), pp. 403–410.

[AML12] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin, *Learning from data*, vol. 4, AMLBook New York, 2012.

[ANR19] ANR, *IntraCranial ANeurysms: From familial forms to pathophysiological mechanisms – I-CAN*, `http://www.agence-nationale-recherche.fr/Project-ANR-15-CE17-0008`, [Online; accessed 10-Oct-2019], 2019.

[APT10] Mohamed Radhouene Aniba, Olivier Poch, and Julie D Thompson, «Issues in bioinformatics benchmarking: the case study of multiple sequence alignment», *in*: *Nucleic acids research* 38.*21* (2010), pp. 7353–7363.

[Are+18] April Moreno Arellano et al., «Privacy policy and technology in biomedical data science», *in*: *Annual review of biomedical data science* 1 (2018), pp. 115–129.

[AS00] Rakesh Agrawal and Ramakrishnan Srikant, «Privacy-preserving data mining», *in*: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 439–450.

[Atk+17] Malcolm Atkinson et al., *Scientific workflows: Past, present and future*, 2017.

[Azi+19] Md Momin Al Aziz et al., «Privacy-preserving techniques of genomic data—a survey», *in*: *Briefings in bioinformatics* 20.*3* (2019), pp. 887–895.

[Bal+13] Daniel Balouek et al., «Adding virtualization capabilities to the Grid'5000 testbed», *in*: *Cloud Computing and Services Science: Second International Conference, CLOSER 2012, Porto, Portugal, April 18-21, 2012. Revised Selected Papers 2*, Springer, 2013, pp. 3–20.

[Bar+10] Derik Barseghian et al., «Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis», *in*: *Ecological Informatics* 5.*1* (2010), pp. 42–50.

[Bar+16] Christian Barillot et al., «Shanoir: applying the software as a service distribution model to manage brain imaging research repositories», *in*: *Frontiers in ICT* 3 (2016), p. 25.

[Bar64] Paul Baran, «On distributed communications networks», *in*: *IEEE transactions on Communications Systems* 12.*1* (1964), pp. 1–9.

[BBL76] Barry W Boehm, John R Brown, and Myron Lipow, «Quantitative evaluation of software quality», *in*: *Proceedings of the 2nd international conference on Software engineering*, 1976, pp. 592–605.

[BDR19] Steven M Bellovin, Preetam K Dutta, and Nathan Reitinger, «Privacy and synthetic datasets», *in*: *Stan. Tech. L. Rev.* 22 (2019), p. 1.

[BFV19] Mirko Bez, Giacomo Fornari, and Tullio Vardanega, «The scalability challenge of ethereum: An initial quantitative analysis», *in*: *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, IEEE, 2019, pp. 167–176.

[BH07] Peter Bühlmann and Torsten Hothorn, «Boosting algorithms: Regularization, prediction and model fitting», *in*: (2007).

[Bia+20] Jiang Bian et al., «Mp2sda: Multi-party parallelized sparse discriminant learning», *in*: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14.*3* (2020), pp. 1–22.

[BL13] Marc Bux and Ulf Leser, «Parallelization in scientific workflow management systems», *in*: *arXiv preprint arXiv:1303.7195* (2013).

[BN06] Christopher M Bishop and Nasser M Nasrabadi, *Pattern recognition and machine learning*, vol. 4, 4, Springer, 2006.

[Bod+15] Ulrich Bodenhofer et al., «msa: an R package for multiple sequence alignment», *in*: *Bioinformatics* 31.*24* (2015), pp. 3997–3999.

[Bos+14] Raphael Bost et al., «Machine learning classification over encrypted data», *in*: *Cryptology ePrint Archive* (2014).

[Bou+12] Anne-Laure Boulesteix et al., «Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics», *in*: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.*6* (2012), pp. 493–507.

[Bou+17] Romain Bourcier et al., «Understanding the pathophysiology of intracranial aneurysm: the ICAN project», *in*: *Neurosurgery* 80.*4* (2017), pp. 621–626.

[Bou+19] Fatima-Zahra Boujdad et al., «On distributed collaboration for biomedical analyses», *in*: *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, 2019, pp. 611–620.

[Bou+22] Fatima-zahra Boujdad et al., «A Hybrid Cloud Deployment Architecture for Privacy-Preserving Collaborative Genome-Wide Association Studies», *in*: *Digital Forensics and Cyber Crime: 12th EAI International Conference, ICDF2C 2021, Virtual Event, Singapore, December 6-9, 2021, Proceedings*, Springer, 2022, pp. 342–359.

[Bra+16] Nicolas L Bray et al., «Near-optimal probabilistic RNA-seq quantification», *in*: *Nature biotechnology* 34.*5* (2016), pp. 525–527.

[Bre+84] L Breirnan et al., *Classification and regression trees*, 1984.

[Bre96a] Leo Breiman, *Arcing classifiers*, tech. rep., Technical report, University of California, Department of Statistics, 1996.

[Bre96b] Leo Breiman, «Bagging predictors», *in*: *Machine learning* 24 (1996), pp. 123–140.

[Bre96c] Leo Breiman, «Out-of-bag estimation», *in*: (1996).

[Bre99] L Breiman, «Random forests», *in*: *University of California: Berkeley, CA, USA* (1999).

[Bro00] Michael W Browne, «Cross-validation methods», *in*: *Journal of mathematical psychology* 44.*1* (2000), pp. 108–132.

[BS16] Gérard Biau and Erwan Scornet, «A random forest guided tour», *in*: *Test* 25 (2016), pp. 197–227.

[BS18] Fatima-Zahra Boujdad and Mario Südholt, «Constructive privacy for shared genetic data», *in*: *CLOSER 2018: 8th International Conference on Cloud Computing and Services Science*, 2018, pp. 1–8.

[BT13] Sam Behjati and Patrick S Tarpey, «What is next generation sequencing?», *in*: *Archives of Disease in Childhood-Education and Practice* 98.*6* (2013), pp. 236–238.

[Bun+19] Annalisa Buniello et al., «The NHGRI-EBI GWAS Catalog of published genome-wide association studies, targeted arrays and summary statistics 2019», *in*: *Nucleic acids research* 47.*D1* (2019), pp. D1005–D1012.

[Bur+05] Alexandre Bureau et al., «Identifying SNPs predictive of phenotype using random forests», *in*: *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* 28.*2* (2005), pp. 171–182.

[Bux+17] Marc Bux et al., «Hi-way: Execution of scientific workflows on hadoop yarn», *in*: *20th International Conference on Extending Database Technology, EDBT 2017, 21 March 2017 through 24 March 2017*, OpenProceedings. org, 2017, pp. 668–679.

[Bux18] Marc Nicolas Bux, «Scientific Workflows for Hadoop», *in*: (2018).

[BV16] Carlyna Bondiombouy and Patrick Valduriez, «Query processing in multistore systems: an overview», *in*: *International Journal of Cloud Computing* 5.*4* (2016), pp. 309–346.

[BW08] Raymond PL Buse and Westley R Weimer, «A metric for software readability», *in*: *Proceedings of the 2008 international symposium on Software testing and analysis*, 2008, pp. 121–130.

[Can+16] Ignacio Cano et al., «Towards geo-distributed machine learning», *in*: *arXiv preprint arXiv:1603.09035* (2016).

[Can+22] Gaia Cantelli et al., «The european bioinformatics institute (EMBL-EBI) in 2021», *in*: *Nucleic Acids Research* 50.*D1* (2022), pp. D11–D19.

[Can20] Parliament of Canada, *BILL C-11*, `https://parl.ca/DocumentViewer/en/43-2/bill/C-11/first-reading`, [Online; accessed 21-July-2021], 2020.

[Can85] Justice Law of Canada, *The Privacy Act*, `https://laws-lois.justice.gc.ca/eng/acts/p-21/fulltext.html`, [Online; accessed 21-July-2021], 1985.

[Cap+05] Franck Cappello et al., «Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform», *in*: *SC'05: Proc. The 6th IEEE/ACM International Workshop on Grid Computing Grid'2005*, hal number inria-00000284, IEEE/ACM, Seattle, USA, Nov. 2005, pp. 99–106, URL: `https://hal.inria.fr/inria-00000284`.

[Car+18] Giulio Caravagna et al., «Detecting repeated cancer evolution from multi-region tumor sequencing data», *in*: *Nature methods* 15.*9* (2018), pp. 707–714.

[Cas+17] Marcelo Rodrigo de Castro et al., «SparkBLAST: scalable BLAST processing using in-memory operations», *in*: *BMC bioinformatics* 18 (2017), pp. 1–13.

[Cat+17a] Giuseppe Cattaneo et al., «An effective extension of the applicability of alignment-free biological sequence comparison algorithms with Hadoop», *in*: *The Journal of Supercomputing* 73 (2017), pp. 1467–1483.

[Cat+17b] Giuseppe Cattaneo et al., «Mapreduce in computational biology-a synopsis», *in*: *Advances in Artificial Life, Evolutionary Computation, and Systems Chemistry: 11th Italian Workshop, WIVACE 2016, Fisciano, Italy, October 4-6, 2016, Revised Selected Papers 11*, Springer, 2017, pp. 53–64.

[Cat19] GWAS Catalog, *GWAS Catalog*, `https://www.ebi.ac.uk/gwas/`, [Online; accessed 20-Sept-2019], 2019.

[CC16] European Commission and Council, *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data*, `http://data.europa.eu/eli/reg/2016/679/2016-05-04`, Apr. 2016.

[CD05] Brecht Claerhout and Georges JE DeMoor, «Privacy protection for clinical and genomic data: The use of privacy-enhancing techniques in medicine», *in*: *International Journal of Medical Informatics* 74.*2-4* (2005), pp. 257–265.

[CDK05] George F Coulouris, Jean Dollimore, and Tim Kindberg, *Distributed systems: concepts and design*, pearson education, 2005.

[CG16] Tianqi Chen and Carlos Guestrin, «Xgboost: A scalable tree boosting system», *in*: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[CG18] Sam Corbett-Davies and Sharad Goel, «The measure and mismeasure of fairness: A critical review of fair machine learning», *in*: *arXiv preprint arXiv:1808.00023* (2018).

[CH67] Thomas Cover and Peter Hart, «Nearest neighbor pattern classification», *in*: *IEEE transactions on information theory* 13.*1* (1967), pp. 21–27.

[Cha+12] Yu-Jung Chang et al., «A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework», *in*: *BMC genomics*, vol. 13, 7, BioMed Central, 2012, pp. 1–17.

[Che+02] Ann Chervenak et al., «Giggle: A framework for constructing scalable replica location services», *in*: *SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, IEEE, 2002, pp. 58–58.

[Che+21] Zheyi Chen et al., «Effective data placement for scientific workflows in mobile edge computing using genetic particle swarm optimization», *in*: *Concurrency and Computation: Practice and Experience* 33.*8* (2021), e5413.

[CI12] Xi Chen and Hemant Ishwaran, «Random forests for genomic data analysis», *in*: *Genomics* 99.*6* (2012), pp. 323–329.

[Cic+17] Mark Cicero et al., «Training and validating a deep convolutional neural network for computer-aided detection and classification of abnormalities on frontal chest radiographs», *in*: *Investigative radiology* 52.*5* (2017), pp. 281–287.

[CJL07] Christophe Croux, Kristel Joossens, and Aurélie Lemmens, «Trimmed bagging», *in*: *Computational statistics & data analysis* 52.*1* (2007), pp. 362–368.

[CMH83] K Mani Chandy, Jayadev Misra, and Laura M Haas, «Distributed deadlock detection», *in*: *ACM Transactions on Computer Systems (TOCS)* 1.*2* (1983), pp. 144–156.

[Coh+17] Sarah Cohen-Boulakia et al., «Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities», *in*: *Future Generation Computer Systems* 75 (2017), pp. 284–298.

[Col+11] Marc E Colosimo et al., «Nephele: genotyping via complete composition vectors and MapReduce», *in*: *Source code for biology and medicine* 6 (2011), pp. 1–10.

[Col12] Congress of Colombia, *Colombian Data Protection Law*, Last accessed 16 September 2021, 2012, URL: `https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981`.

[Com] European Commission, *Data protection in the EU*, `https://ec.europa.eu/info/law/law-topic/data-protection/data-protection-eu_en`.

[Con+14] DS Consortium et al., «Genome-wide trans-ancestry meta-analysis provides insight into the genetic architecture of type 2 diabetes susceptibility», *in*: *Nature genetics* 46.*3* (2014), pp. 234–244.

[Con13] Mozart Consortium, *The Mozart Programming System*, `http://www.mozart-oz.org/`, [Online; accessed 01-January-2021], 2013.

[Coo+19] Charles E Cook et al., «The European Bioinformatics Institute in 2018: tools, infrastructure and training», *in*: *Nucleic acids research* 47.*D1* (2019), pp. D15–D22.

[Coo+20a] Charles E Cook et al., «The European Bioinformatics Institute in 2020: building a global infrastructure of interconnected data resources for the life sciences», *in*: *Nucleic acids research* 48.*D1* (2020), pp. D17–D23.

[Coo+20b] Charles E Cook et al., «The European Bioinformatics Institute in 2020: building a global infrastructure of interconnected data resources for the life sciences», *in*: *Nucleic acids research* 48.*D1* (2020), pp. D17–D23.

[Cop+09] Jason M Cope et al., «Robust data placement in urgent computing environments», *in*: *2009 IEEE International Symposium on Parallel & Distributed Processing*, IEEE, 2009, pp. 1–13.

[Cor+18] Manuel Corpas et al., «A FAIR guide for data providers to maximise sharing of human genomic data», *in*: *PLoS computational biology* 14.*3* (2018), e1005873.

[Cou08] Jennifer Couzin, *Whole-genome data not anonymous, challenging assumptions*, 2008.

[Cri+19] Stephen Cristiano et al., «Genome-wide cell-free DNA fragmentation in patients with cancer», *in*: *Nature* 570.*7761* (2019), pp. 385–389.

[DCD03] GJE De Moor, B Claerhout, and FILIP De Meyer, «Privacy enhancing techniques», *in*: *Methods of information in medicine* 42.*02* (2003), pp. 148–153.

[De +11] David De Roure et al., «Towards the preservation of scientific workflows», *in*: *Proc. 8th Intl. Conference on Preservation of Digital Objects*, 2011.

[De +12] Pierre De Wit et al., «The simple fool's guide to population genomics via RNA-Seq: an introduction to high-throughput sequencing data analysis», *in*: *Molecular ecology resources* 12.*6* (2012), pp. 1058–1067.

[Dea+12] Jeffrey Dean et al., «Large scale distributed deep networks», *in*: *Advances in neural information processing systems* 25 (2012).

[DeB+10] Joe DeBartolo et al., «Protein structure prediction enhanced with evolutionary diversity: SPEED», *in*: *Protein Science* 19.*3* (2010), pp. 520–534.

[Dec+15] Dries Decap et al., «Halvade: scalable sequence analysis with MapReduce», *in*: *Bioinformatics* 31.*15* (2015), pp. 2482–2488.

[DEC21] DECIPHER, *DatabasE of genomiC varIation and Phenotype in Humans using Ensembl Resources*, `https://www.deciphergenomics.org/about/overview`, [Online; accessed 21-July-2021], 2021.

[Dee+09] Ewa Deelman et al., «Workflows and e-Science: An overview of workflow system features and capabilities», *in*: *Future generation computer systems* 25.*5* (2009), pp. 528–540.

[Dee+15] Ewa Deelman et al., «Pegasus, a workflow management system for science automation», *in*: *Future Generation Computer Systems* 46 (2015), pp. 17–35.

[Di +17] Paolo Di Tommaso et al., «Nextflow enables reproducible computational workflows», *in*: *Nature biotechnology* 35.*4* (2017), pp. 316–319.

[Dob+13] Alexander Dobin et al., «STAR: ultrafast universal RNA-seq aligner», *in*: *Bioinformatics* 29.*1* (2013), pp. 15–21.

[Dol+17a] Shlomi Dolev et al., «A survey on geographically distributed big-data processing using MapReduce», *in*: *IEEE Transactions on Big Data* 5.*1* (2017), pp. 60–80.

[Dol+17b] Shlomi Dolev et al., «A survey on geographically distributed big-data processing using MapReduce», *in*: *IEEE Transactions on Big Data* 5.*1* (2017), pp. 60–80.

[Dom+19] Josep Domingo-Ferrer et al., «Privacy-preserving cloud computing on sensitive data: A survey of methods, products and challenges», *in*: *Computer Communications* 140 (2019), pp. 38–60.

[Don+17] Gaifang Dong et al., «An accurate sequence assembly algorithm for livestock, plants and microorganism based on Spark», *in*: *International Journal of Pattern Recognition and Artificial Intelligence* 31.*08* (2017), p. 1750024.

[Dwo+06] Cynthia Dwork et al., «Calibrating noise to sensitivity in private data analysis», *in*: *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, Springer, 2006, pp. 265–284.

[Dwo08] Cynthia Dwork, «Differential privacy: A survey of results», *in*: *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings 5*, Springer, 2008, pp. 1–19.

[EAH18] Peter F Edemekong, Pavan Annamaraju, and Micelle J Haydel, «Health insurance portability and accountability act», *in*: (2018).

[Ebr+15] Mahdi Ebrahimi et al., «Bdap: a big data placement strategy for cloud-based scientific workflows», *in*: *2015 IEEE First International Conference on Big Data Computing Service and Applications*, IEEE, 2015, pp. 105–114.

[EHT10] Erik Elmroth, Francisco Hernández, and Johan Tordsson, «Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment», *in*: *Future Generation Computer Systems* 26.*2* (2010), pp. 245–256.

[El 10] Khaled El Emam, «Risk-based de-identification of health data», *in*: *IEEE Security & Privacy* 8.*3* (2010), pp. 64–67.

[ELI14] ELIXIR, *EXCELERATE*, `https://elixir-europe.org/about-us/how-funded/eu-projects/excelerate`, [Online; accessed 18-March-2021], 2014.

[ELI19] ELIXIR, *European Research Infrastructure for Data in Life Sciences*, `https://elixir-europe.org`, [Online; accessed 18-March-2021], 2019.

[Eng21] Genomics England, *Department of Health and Social Care*, `https://www.genomicsengland.co.uk/information-for-participants/participant-forms/`, [Online; accessed 21-July-2021], 2021.

[Era+19] Gökcen Eraslan et al., «Deep learning: new computational modelling techniques for genomics», *in*: *Nature Reviews Genetics* 20.*7* (2019), pp. 389–403.

[Est+19] Andre Esteva et al., «A guide to deep learning in healthcare», *in*: *Nature medicine* 25.*1* (2019), pp. 24–29.

[EU20] GDPR EU, *What is the LGPD? Brazil's version of the GDPR*, https://gdpr.eu/gdpr-vs-lgpd/, [Online; accessed 21-July-2021], 2020.

[Eur14] I European Union, «Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions», *in*: *A newskillsagendaforeurope. Brussels* (2014).

[Eur20] EUR Lex Europa, *Implementing Decision (EU) 2019/419*, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2019.076.01.0001.01.ENG&toc=OJ:L:2019:076:TOC, [Online; accessed 06-August-2021], 2020.

[FBS12] Juliana Freire, Philippe Bonnet, and Dennis Shasha, «Computational reproducibility: state-of-the-art, challenges, and database research opportunities», *in*: *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, 2012, pp. 593–596.

[Fec+17] Benedikt Fecher et al., «A reputation economy: how individual reward considerations trump systemic arguments for open access to data», *in*: *Palgrave Communications* 3.*1* (2017), pp. 1–10.

[Fed+15] Lisa M Federer et al., «Biomedical data sharing and reuse: attitudes and practices of clinical and scientific research staff», *in*: *PloS one* 10.*6* (2015), e0129506.

[Fen+19] Zhi Feng et al., «Securegbm: Secure multi-party gradient boosting», *in*: *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, 2019, pp. 1312–1321.

[Fer+12] óscar Ferrández et al., «Generalizability and comparison of automatic clinical text de-identification methods and resources», *in*: *AMIA Annual Symposium Proceedings*, vol. 2012, American Medical Informatics Association, 2012, p. 199.

[Fer+18] Alberto Fernández et al., *Learning from imbalanced data sets*, vol. 10, Springer, 2018.

[Fer89] David Fernández-Baca, «Allocating modules to processors in a distributed system», *in*: *IEEE Transactions on Software Engineering* 15.*11* (1989), pp. 1427–1436.

[FHL14] Jianqing Fan, Fang Han, and Han Liu, «Challenges of big data analysis», *in*: *National science review* 1.*2* (2014), pp. 293–314.

[FHT01] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, «The elements of statistical learning. vol. 1 Springer series in statistics», *in*: *New York* (2001).

[Fos+08] Ian Foster et al., «Cloud computing and grid computing 360-degree compared», *in*: *2008 grid computing environments workshop*, Ieee, 2008, pp. 1–10.

[Fou14] National Science Foundation, *Core Techniques and Technologies for Advancing Big Data Science and Engineering*, 2014, URL: `http://www.nsf.gov/pubs/2012/nsf12499/nsf12499.htm` (visited on 04/03/2019).

[Fou17] New America Foundation, *Translation: Cybersecurity Law of the People's Republic of China*, `https://www.newamerica.org/cybersecurity-initiative/digichina/blog/translation-cybersecurity-law-peoples-republic-china/`, [Online; accessed 06-August-2021], 2017.

[Fou20] New America Foundation, *Translation: China's Draft Personal Information Protection Law*, `https://www.newamerica.org/cybersecurity-initiative/digichina/blog/chinas-draft-personal-information-protection-law-full-translation/`, [Online; accessed 06-August-2021], 2020.

[Fre14] Alex A Freitas, «Comprehensible classification models: a position paper», *in*: *ACM SIGKDD explorations newsletter* 15.*1* (2014), pp. 1–10.

[Fri+18] Kyle Fritchman et al., «Privacy-preserving scoring of tree ensembles: A novel framework for AI in healthcare», *in*: *2018 IEEE international conference on big data (Big Data)*, Ieee, 2018, pp. 2413–2422.

[Fri01] Jerome H Friedman, «Greedy function approximation: a gradient boosting machine», *in*: *Annals of statistics* (2001), pp. 1189–1232.

[Fry+15] Stephen V Frye et al., «Tackling reproducibility in academic preclinical drug discovery», *in*: *Nature Reviews Drug Discovery* 14.*11* (2015), pp. 733–734.

[FS+96] Yoav Freund, Robert E Schapire, et al., «Experiments with a new boosting algorithm», *in*: *icml*, vol. 96, Citeseer, 1996, pp. 148–156.

[FS97] Yoav Freund and Robert E Schapire, «A decision-theoretic generalization of on-line learning and an application to boosting», *in*: *Journal of computer and system sciences* 55.*1* (1997), pp. 119–139.

[GA4] GA4GH, *The Global Alliance for Genomics and Health*, `https://www.ga4gh.org/`.

[Gar+22] Wilmer Garzón et al., «A taxonomy of tools and approaches for distributed genomic analyses», *in*: *Informatics in Medicine Unlocked* (2022), p. 101024.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.

[GEW06] Pierre Geurts, Damien Ernst, and Louis Wehenkel, «Extremely randomized trees», *in*: *Machine learning* 63 (2006), pp. 3–42.

[GFI16] Steven N Goodman, Daniele Fanelli, and John PA Ioannidis, «What does research reproducibility mean?», *in*: *Science translational medicine* 8.*341* (2016), 341ps12–341ps12.

[GG20] The Global Alliance for Genomics and Health (GA4GH), *GDPR Brief: Japan obtains the first adequacy agreement under the GDPR*, `https://www.ga4gh.org/news/gdpr-brief-japan-obtains-the-first-adequacy-agreement-under-the-gdpr/`, [Online; accessed 06-August-2021], 2020.

[GH] The Global Alliance for Genomics and Health, *Genomics England Implements GA4GH*, `https://www.ga4gh.org/news/genomics-england-implements-ga4gh-api-to-provide-secure-access-to-genomic-data-for-the-nhs/`.

[Gia+19] Irene Giacomelli et al., «Privacy-preserving collaborative prediction using random forests», *in*: *AMIA summits on translational science proceedings* 2019 (2019), p. 248.

[Gil+10] Yolanda Gil et al., «Wings: Intelligent workflow-based design of computational experiments», *in*: *IEEE Intelligent Systems* 26.*1* (2010), pp. 62–72.

[GL02] Seth Gilbert and Nancy Lynch, «Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services», *in*: *Acm Sigact News* 33.*2* (2002), pp. 51–59.

[GLN13] Thore Graepel, Kristin Lauter, and Michael Naehrig, «ML confidential: Machine learning on encrypted data», *in*: *Information Security and Cryptology–ICISC 2012: 15th International Conference, Seoul, Korea, November 28-30, 2012, Revised Selected Papers 15*, Springer, 2013, pp. 1–21.

[Glo] Privacy Laws Around Globe, *Privacy Laws Around Globe*, `https://piwik.pro/privacy-laws-around-globe/`, Accessed: 2021-09-30.

[Goe+10] Jeremy Goecks et al., «Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences», *in*: *Genome biology* 11 (2010), pp. 1–13.

[Gol77] Barry Goldman, *Deadlock detection in computer networks*, tech. rep., Massachusetts Institute of Tech Cambridge Lab for Computer Science, 1977.

[Gov] North Carolina Gov, *North Carolina Public Records Law*, `https://www.nccourts.gov/services/request-a-public-record/about-north-carolina-public-records-law`, [Online; accessed 21-July-2021].

[GPB11] Benjamin A Goldstein, Eric C Polley, and Farren BS Briggs, «Random forests for genetic association studies», *in*: *Statistical applications in genetics and molecular biology* 10.*1* (2011).

[GR18] Julija Golosova and Andrejs Romanovs, «The advantages and disadvantages of the blockchain technology», *in*: *2018 IEEE 6th workshop on advances in information, electronic and electrical engineering (AIEEE)*, IEEE, 2018, pp. 1–6.

[GTA06] Michael I Gordon, William Thies, and Saman Amarasinghe, «Exploiting coarse-grained task, data, and pipeline parallelism in stream programs», *in*: *ACM SIGPLAN Notices* 41.*11* (2006), pp. 151–162.

[Guo+18] Runxin Guo et al., «Bioinformatics applications on apache spark», *in*: *GigaScience* 7.*8* (2018), giy098.

[GX09] James Gardner and Li Xiong, «An integrated framework for de-identifying unstructured medical data», *in*: *Data & Knowledge Engineering* 68.*12* (2009), pp. 1441–1451.

[GX15] Slawomir Goryczka and Li Xiong, «A comprehensive comparison of multiparty secure additions with differential privacy», *in*: *IEEE transactions on dependable and secure computing* 14.*5* (2015), pp. 463–477.

[Hao+19] Meng Hao et al., «Towards efficient and privacy-preserving federated deep learning», *in*: *ICC 2019-2019 IEEE international conference on communications (ICC)*, IEEE, 2019, pp. 1–6.

[Has+09] Trevor Hastie et al., *The elements of statistical learning: data mining, inference, and prediction*, vol. 2, Springer, 2009.

[Has+18] Tatsunori Hashimoto et al., «Fairness without demographics in repeated loss minimization», *in*: *International Conference on Machine Learning*, PMLR, 2018, pp. 1929–1938.

[Hea+18] National Institutes of Health et al., «NIH strategic plan for data science», *in*: *NIH, June* (2018).

[Hea+21] National Institutes of Health et al., «Guidance: rigor and reproducibility in grant applications», *in*: *Online document at: https://grants. nih. gov/policy/reproducibility/guidance. htm, accessed March* 24 (2021).

[Hea+98] Marti A. Hearst et al., «Support vector machines», *in*: *IEEE Intelligent Systems and their applications* 13.*4* (1998), pp. 18–28.

[Hea21] National Institutes of Health (NIH), *NIH Genomic Data Sharing*, `https://osp.od.nih.gov/scientific-sharing/genomic-data-sharing/`, [Online; accessed 18-July-2021], 2021.

[HG13a] Yunda Huang and Raphael Gottardo, «Comparability and reproducibility of biomedical data», *in*: *Briefings in bioinformatics* 14.*4* (2013), pp. 391–401.

[HG13b] Yunda Huang and Raphael Gottardo, «Comparability and reproducibility of biomedical data», *in*: *Briefings in bioinformatics* 14.*4* (2013), pp. 391–401.

[Hin13] Pieter Hintjens, *ZeroMQ: messaging for many applications*, " O'Reilly Media, Inc.", 2013.

[HKS18] Liren Huang, Jan Krüger, and Alexander Sczyrba, «Analyzing large scale genomic data on the cloud with Sparkhit», *in*: *Bioinformatics* 34.*9* (2018), pp. 1457–1465.

[Hom+08] Nils Homer et al., «Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays», *in*: *PLoS genetics* 4.*8* (2008), e1000167.

[Hon+12] Dongwan Hong et al., «FX: an RNA-Seq analysis tool on the cloud», *in*: *Bioinformatics* 28.*5* (2012), pp. 721–723.

[Hor14] Horizon, *H2020 Programme*, `https://ec.europa.eu/easme/en/section/horizon-2020-energy-efficiency/h2020-programme`, [Online; accessed 18-March-2021], 2014.

[HPS16] Moritz Hardt, Eric Price, and Nati Srebro, «Equality of opportunity in supervised learning», *in*: *Advances in neural information processing systems* 29 (2016).

[HR82] Gary S. Ho and CV Ramamoorthy, «Protocols for deadlock detection in distributed database systems», *in*: *IEEE Transactions on Software Engineering* *6* (1982), pp. 554–557.

[HSW93] Martin Henz, Gert Smolka, and Jörg Würtz, «Oz-a programming language for multi-agent systems», *in*: *IJCAI*, Citeseer, 1993, pp. 404–409.

[HTP13] Hailiang Huang, Sandeep Tata, and Robert J Prill, «BlueSNP: R package for highly scalable genome-wide association studies using Hadoop clusters», *in*: *Bioinformatics* 29.*1* (2013), pp. 135–136.

[Hun+11] Che-Lun Hung et al., «CloudTSS: a TagSNP selection approach on cloud computing», *in*: *Grid and Distributed Computing: International Conference, GDC 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings*, Springer, 2011, pp. 525–534.

[Hut10] Stu Hutson, «Data handling errors spur debate over clinical trial», *in*: *Nature medicine* 16.*6* (2010), p. 618.

[Im+12] Hae Kyung Im et al., «On sharing quantitative trait GWAS results in an era of multiple-omics data and the limits of genomic privacy», *in*: *The American Journal of Human Genetics* 90.*4* (2012), pp. 591–598.

[Ize13] Alan Julian Izenman, «Linear discriminant analysis», *in*: *Modern multivariate statistical techniques*, New York: Springer, 2013, pp. 237–280.

[JAP20] Personal Information Protection Commission JAPAN, *Act on the Protection of Personal Information*, `https://www.ppc.go.jp/en/legal/`, [Online; accessed 06-August-2021], 2020.

[Jia+09] Rui Jiang et al., «A random forest approach to the detection of epistatic interactions in case-control studies», *in*: *BMC bioinformatics* 10.*1* (2009), pp. 1–12.

[JMM96] Anil K Jain, Jianchang Mao, and K Moidin Mohiuddin, «Artificial neural networks: A tutorial», *in*: *Computer* 29.*3* (1996), pp. 31–44.

[JT00] Gillian CL Johnson and John A Todd, «Strategies in complex disease mapping», *in*: *Current opinion in genetics & development* 10.*3* (2000), pp. 330–334.

[Ju+11] Young Seok Ju et al., «Extensive genomic and transcriptional diversity identified through massively parallel DNA and RNA sequencing of eighteen Korean individuals», *in*: *Nature genetics* 43.*8* (2011), pp. 745–752.

[Kai+20] Georgios A Kaissis et al., «Secure, privacy-preserving and federated machine learning in medical imaging», *in*: *Nature Machine Intelligence* 2.*6* (2020), pp. 305–311.

[Kai+21] Peter Kairouz et al., «Advances and open problems in federated learning», *in*: *Foundations and Trends® in Machine Learning* 14.*1–2* (2021), pp. 1–210.

[Kan+08] Murat Kantarcioglu et al., «A cryptographic approach to securely share and query genomic sequences», *in*: *IEEE Transactions on information technology in biomedicine* 12.*5* (2008), pp. 606–617.

[Kar+18] Md Rezaul Karim et al., «Improving data workflow systems with cloud services and use of open data for bioinformatics research», *in*: *Briefings in bioinformatics* 19.*5* (2018), pp. 1035–1050.

[Kha+19] Farah Zaib Khan et al., «Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv», *in*: *GigaScience* 8.*11* (2019), giz095.

[Kie+14] Peter Kieseberg et al., «Protecting anonymity in data-driven biomedical science», *in*: *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics: State-of-the-Art and Future Challenges* (2014), pp. 301–316.

[Kie03] Bartosz Kiepusewski, *Expressiveness and suitability of languages for control flow modelling in workflows*, Queensland University of Technology, Brisbane, 2003.

[Kim19] Ju Han Kim, *Genome Data Analysis*, Springer Singapore, 2019, URL: https://www.springer.com/gp/book/9789811319419.

[Kit14] Rob Kitchin, *The data revolution: Big data, open data, data infrastructures and their consequences*, Sage, 2014.

[Kno14] Bartha Maria Knoppers, «Framework for responsible sharing of genomic and health-related data», *in*: *The HUGO journal* 8.*1* (2014), p. 3.

[Kon+16a] Jakub Konečný et al., «Federated learning: Strategies for improving communication efficiency», *in*: *arXiv preprint arXiv:1610.05492* (2016).

[Kon+16b] Jakub Konečný et al., «Federated optimization: Distributed machine learning for on-device intelligence», *in*: *arXiv preprint arXiv:1610.02527* (2016).

[Koo+17] Thijs Kooi et al., «Large scale deep learning for computer aided detection of mammographic lesions», *in*: *Medical image analysis* 35 (2017), pp. 303–312.

[Kor+14] Eija Korpelainen et al., *RNA-seq data analysis: a practical approach*, CRC press, 2014.

[Kot13] Sotiris B Kotsiantis, «Decision trees: a recent overview», *in*: *Artificial Intelligence Review* 39 (2013), pp. 261–283.

[Kow+19] Kamran Kowsari et al., «Text classification algorithms: A survey», *in*: *Information* 10.*4* (2019), p. 150.

[KR12] Johannes Köster and Sven Rahmann, «Snakemake—a scalable bioinformatics workflow engine», *in*: *Bioinformatics* 28.*19* (2012), pp. 2520–2522.

[KS11] Ajay D Kshemkalyani and Mukesh Singhal, *Distributed computing: principles, algorithms, and systems*, Cambridge University Press, 2011.

[KTB00] Bartek Kiepuszewski, Arthur Harry Maria Ter Hofstede, and Christoph J Bussler, «On structured workflow modelling», *in*: *Advanced Information Systems Engineering: 12th International Conference, CAiSE 2000 Stockholm, Sweden, June 5–9, 2000 Proceedings 12*, Springer, 2000, pp. 431–445.

[Kuh+07] K Kuhn et al., «The cancer biomedical informatics grid (caBIG™): Infrastructure and applications for a worldwide research community», *in*: *Medinfo* 1 (2007), p. 330.

[KV16] Donghoon Kim and Mladen A Vouk, «Assessing run-time overhead of Securing Kepler», *in*: *Procedia Computer Science* 80 (2016), pp. 2281–2286.

[lab19] Palumbi lab, *An Introduction to High-Throughput Sequencing Data Analysis*, `http://sfg.stanford.edu/`, [Online; accessed 20-June-2019], 2019.

[Lab21] U.S. Deparment of Labor, *Guidance on the Protection of Personal Identifiable Information*, `https://www.dol.gov/general/ppii`, [Online; accessed 15-July-2021], 2021.

[Lan+09] Ben Langmead et al., «Searching for SNPs with cloud computing», *in*: *Genome biology* 10 (2009), pp. 1–10.

[Lar+18] Elise Larsonneur et al., «Evaluating workflow management systems: A bioinformatics use case», *in*: *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, 2018, pp. 2773–2775.

[LD09] Heng Li and Richard Durbin, «Fast and accurate short read alignment with Burrows–Wheeler transform», *in*: *bioinformatics* 25.*14* (2009), pp. 1754–1760.

[LD11] Xin Liu and Anwitaman Datta, «Towards intelligent data placement for scientific workflows in collaborative cloud environment», *in*: *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*, IEEE, 2011, pp. 1052–1061.

[Leg] California State Legislature, *The California Consumer Privacy Act of 2018*, `https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB1121`.

[Leg20a] New Zealand Legislation, *New Zealand Privacy Act 2020*, `https://www.legislation.govt.nz/act/public/2020/0031/latest/LMS23223.html`, [Online; accessed 21-July-2021], 2020.

[Leg20b] Virginia's Legislative, *Consumer Data Protection Act*, `https://lis.virginia.gov/cgi-bin/legp604.exe?212+sum+SB1392`, [Online; accessed 06-August-2021], 2020.

[Leu+15] Michael KK Leung et al., «Machine learning in genomic medicine: a review of computational problems and data sets», *in*: *Proceedings of the IEEE* 104.*1* (2015), pp. 176–197.

[LH05] Gregor von Laszewski and Mike Hategan, «Workflow concepts of the java cog kit», *in*: *Journal of Grid Computing* 3 (2005), pp. 239–258.

[LHL10] Ben Langmead, Kasper D Hansen, and Jeffrey T Leek, «Cloud-scale RNA-sequencing differential expression analysis with Myrna», *in*: *Genome biology* 11 (2010), pp. 1–11.

[Li+09] Ruiqiang Li et al., «SNP detection for massively parallel whole-genome re-sequencing», *in*: *Genome research* 19.*6* (2009), pp. 1124–1132.

[Li+16] Xuejun Li et al., «A novel workflow-level data placement strategy for data-sharing scientific cloud workflows», *in*: *IEEE Transactions on Services Computing* 12.*3* (2016), pp. 370–383.

[Li+19] Qinbin Li et al., «A survey on federated learning systems: vision, hype and reality for data privacy and protection», *in*: *Hype and Reality for Data Privacy and Protection* (2019).

[Li+20a] Jin Li et al., «A multicenter random forest model for effective prognosis prediction in collaborative clinical research network», *in*: *Artificial intelligence in medicine* 103 (2020), p. 101814.

[Li+20b] Tian Li et al., «Federated learning: Challenges, methods, and future directions», *in*: *IEEE signal processing magazine* 37.*3* (2020), pp. 50–60.

[Lie+16] Chee Sun Liew et al., «Scientific workflows: moving across paradigms», *in*: *ACM Computing Surveys (CSUR)* 49.*4* (2016), pp. 1–39.

[Liñ+19] Jose Liñares Blanco et al., «Differential gene expression analysis of RNA-seq data using machine learning for Cancer research», *in*: *Machine learning paradigms: applications of learning and analytics in intelligent systems* (2019), pp. 27–65.

[Liu+14] Ji Liu et al., «Parallelization of scientific workflows in the cloud», PhD thesis, INRIA, 2014.

[Liu+15] Ji Liu et al., «A survey of data-intensive scientific workflow management», *in*: *Journal of Grid Computing* 13 (2015), pp. 457–493.

[Liu+17] Ji Liu et al., «Scientific workflow scheduling with provenance data in a multisite cloud», *in*: *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXXIII* (2017), pp. 80–112.

[Liu+18] Ji Liu et al., «Efficient scheduling of scientific workflows using hot metadata in a multisite cloud», *in*: *IEEE Transactions on Knowledge and Data Engineering* 31.*10* (2018), pp. 1940–1953.

[Liu+20a] Yang Liu et al., «Federated forest», *in*: *IEEE Transactions on Big Data* 8.*3* (2020), pp. 843–854.

[Liu+20b] Yi Liu et al., «A systematic literature review on federated learning: From a model quality perspective», *in*: *arXiv preprint arXiv:2012.01973* (2020).

[Liu+22] Ji Liu et al., «From distributed machine learning to federated learning: A survey», *in*: *Knowledge and Information Systems* 64.*4* (2022), pp. 885–917.

[Liu14] Ji Liu, «Multisite Management of Data-intensive Scientific Workflows in the Cloud», *in*: *BDA: Gestion de Données—Principes, Technologies et Applications*, 2014, pp. 28–30.

[LN15] Maxwell W Libbrecht and William Stafford Noble, «Machine learning applications in genetics and genomics», *in*: *Nature Reviews Genetics* 16.*6* (2015), pp. 321–332.

[Loh11] Wei-Yin Loh, «Classification and regression trees», *in*: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1.*1* (2011), pp. 14–23.

[Lov+12] Jakob Lovén et al., «Revisiting global gene expression analysis», *in*: *Cell* 151.*3* (2012), pp. 476–482.

[LSZ09] Simone Leo, Federico Santoni, and Gianluigi Zanetti, «Biodoop: bioinformatics on hadoop», *in*: *2009 International Conference on Parallel Processing Workshops*, IEEE, 2009, pp. 415–422.

[Lu+17] Yang Young Lu et al., «CAFE: a C celerated A lignment-F r E e sequence analysis», *in*: *Nucleic acids research* 45.*W1* (2017), W554–W559.

[LWH] Q Li, Z Wen, and B He, «Federated learning systems: Vision, hype and reality for data privacy and protection. arXiv 2019», *in*: *arXiv preprint arXiv:1907.09693* ().

[LYS15] Wen-Jie Lu, Yoshiji Yamada, and Jun Sakuma, «Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption», *in*: *BMC medical informatics and decision making*, vol. 15, Springer, 2015, pp. 1–8.

[LZ11] Shiyong Lu and Jia Zhang, «Collaborative scientific workflows supporting collaborative science», *in*: *International Journal of Business Process Integration and Management* 5.*2* (2011), pp. 185–199.

[Mad03] Tom Madden, «The BLAST sequence analysis tool», *in*: *The NCBI handbook* (2003).

[Map] Global Privacy Map, *Global Privacy Map*, `https://wfanet.org/tools/global-privacy-map`, Accessed: 2021-09-30.

[Mar+04] Fernando Martin-Sanchez et al., «Synergy between medical informatics and bioinformatics: facilitating genomic medicine for future health care», *in*: *Journal of biomedical informatics* 37.*1* (2004), pp. 30–42.

[Mar+14] Ronald Margolis et al., «The National Institutes of Health's Big Data to Knowledge (BD2K) initiative: capitalizing on biomedical big data», *in*: *Journal of the American Medical Informatics Association* 21.*6* (2014), pp. 957–958.

[McK+10] Aaron McKenna et al., «The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data», *in*: *Genome research* 20.*9* (2010), pp. 1297–1303.

[McM+17] Brendan McMahan et al., «Communication-efficient learning of deep networks from decentralized data», *in*: *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[Meh+21] Ninareh Mehrabi et al., «A survey on bias and fairness in machine learning», *in*: *ACM Computing Surveys (CSUR)* 54.*6* (2021), pp. 1–35.

[MEO13] Bradley A Malin, Khaled El Emam, and Christine M O'Keefe, «Biomedical data privacy: problems, perspectives, and recent advances», *in*: *Journal of the American medical informatics association* 20.*1* (2013), pp. 2–6.

[Mey+14] Stéphane M Meystre et al., «Text de-identification for privacy protection: a study of its impact on clinical text information content», *in*: *Journal of biomedical informatics* 50 (2014), pp. 142–150.

[MG19] Kalikinkar Mandal and Guang Gong, «PrivFL: Practical privacy-preserving federated regressions on high-dimensional data over mobile networks», *in*: *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2019, pp. 57–68.

[MGG10] Matthew Meyerson, Stacey Gabriel, and Gad Getz, «Advances in understanding cancer genomes through second-generation sequencing», *in*: *Nature Reviews Genetics* 11.*10* (2010), pp. 685–696.

[MHS05] Marjan Mernik, Jan Heering, and Anthony M Sloane, «When and how to develop domain-specific languages», *in*: *ACM computing surveys (CSUR)* 37.*4* (2005), pp. 316–344.

[Mil00] Perry L Miller, «Opportunities at the intersection of bioinformatics and health informatics: a case study», *in*: *Journal of the American Medical Informatics Association* 7.*5* (2000), pp. 431–438.

[Mil94] Randolph A Miller, «Medical diagnostic decision support systems—past, present, and future: a threaded bibliography and brief commentary», *in*: *Journal of the American Medical Informatics Association* 1.*1* (1994), pp. 8–27.

[Mit80] Tom M Mitchell, *The need for biases in learning generalizations*, Citeseer, 1980.

[MK03] Victor Maojo and Casimir A Kulikowski, «Bioinformatics and medical informatics: collaborations on the road to genomic medicine?», *in*: *Journal of the American Medical Informatics Association* 10.*6* (2003), pp. 515–522.

[MK20] Fruzsina Molnár-Gábor and Jan O Korbel, «Genomic data sharing in Europe is stumbling—Could a code of conduct prevent its fall?», *in*: *EMBO molecular medicine* 12.*3* (2020), e11421.

[ML17] Gunasekaran Manogaran and Daphne Lopez, «A survey of big data architectures and machine learning algorithms in healthcare», *in*: *International Journal of Biomedical Engineering and Technology* 25.*2-4* (2017), pp. 182–211.

[MMP11] James D Malley, Karen G Malley, and Sinisa Pajevic, *Statistical learning for biomedical data*, Cambridge University Press, 2011.

[MO14] Mehdi Mirza and Simon Osindero, «Conditional generative adversarial nets», *in*: *arXiv preprint arXiv:1411.1784* (2014).

[Mor+13] Luc Moreau et al., «PROV-N: The provenance notation», *in*: *W3C Recommendation* (2013).

[Mou+11] George P Moustris et al., «Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature», *in*: *The international journal of medical robotics and computer assisted surgery* 7.*4* (2011), pp. 375–392.

[MR14] Oded Z Maimon and Lior Rokach, *Data mining with decision trees: theory and applications*, vol. 81, World scientific, 2014.

[MSS19] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh, «Agnostic federated learning», *in*: *International Conference on Machine Learning*, PMLR, 2019, pp. 4615–4625.

[MTF08] Andréa Matsunaga, Maurício Tsugawa, and José Fortes, «Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications», *in*: *2008 IEEE Fourth International Conference on eScience*, IEEE, 2008, pp. 222–229.

[NB18] Vivek Navale and Philip E Bourne, «Cloud computing applications for biomedical science: A perspective», *in*: *PLoS computational biology* 14.*6* (2018), e1006144.

[Neu+19] Geoffrey K Neumann et al., «Pseudonymization risk analysis in distributed systems», *in*: *Journal of Internet Services and Applications* 10.*1* (2019), pp. 1–16.

[Nor+13] Henrik Nordberg et al., «BioPig: a Hadoop-based analytic toolkit for large-scale sequence data», *in*: *Bioinformatics* 29.*23* (2013), pp. 3014–3019.

[NSF19] NSF, *Chapter XI - Other Post Award Requirements and Consideration*, `https://www.nsf.gov/pubs/policydocs/pappg19_1/pappg_11.jsp#XID4`, [Online; accessed 20-June-2019], 2019.

[NSR11] Tung Nguyen, Weisong Shi, and Douglas Ruden, «CloudAligner: a fast and full-featured MapReduce based tool for sequence mapping», *in*: *BMC research notes* 4.*1* (2011), pp. 1–7.

[NV08] Meiyappan Nagappan and Mladen A Vouk, «A model for sharing of confidential provenance information in a query based system», *in*: *Provenance and Annotation of Data and Processes: Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected Papers 2*, Springer, 2008, pp. 62–69.

[Obe82] Ron Obermarck, «Distributed deadlock detection algorithm», *in*: *ACM Transactions on Database Systems (TODS)* 7.*2* (1982), pp. 187–208.

[OBr+15] Aidan R O'Brien et al., «VariantSpark: population scale clustering of genotype information», *in*: *BMC genomics* 16 (2015), pp. 1–9.

[Oca+15] Kary ACS Ocaña et al., «Data analytics in bioinformatics: data science in practice for genomics analysis workflows», *in*: *2015 IEEE 11th International Conference on e-Science*, IEEE, 2015, pp. 322–331.

[Ohn+12] Lucila Ohno-Machado et al., «iDASH: integrating data for analysis, anonymization, and sharing», *in*: *Journal of the American Medical Informatics Association* 19.*2* (2012), pp. 196–201.

[Ohr+16] Olga Ohrimenko et al., «Oblivious multi-party machine learning on trusted processors.», *in*: *USENIX Security Symposium*, vol. 16, 2016, pp. 10–12.

[Ovt+15] Kalin Ovtcharov et al., «Accelerating deep convolutional neural networks using specialized hardware», *in*: *Microsoft Research Whitepaper* 2.*11* (2015), pp. 1–4.

[Pan+09] Biswanath Panda et al., «Planet: massively parallel learning of tree ensembles with mapreduce», *in*: (2009).

[Pap+18a] Louis Papageorgiou et al., «Genomic big data hitting the storage bottleneck», *in*: *EMBnet. journal* 24 (2018).

[Pap+18b] Nicolas Papernot et al., «Scalable private learning with pate», *in*: *arXiv preprint arXiv:1802.08908* (2018).

[Par21] European Parliament, *Data Protection Working Party*, `https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf`, [Online; accessed 21-July-2021], 2021.

[PC95] European Parliament and Council, *Data Protection Directive*, `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:31995L0046`, [Online; accessed 21-July-2021], 1995.

[PCA15] Luis Pineda-Morales, Alexandru Costan, and Gabriel Antoniu, «Towards multi-site metadata management for geographically distributed cloud workflows», *in*: *2015 IEEE International Conference on Cluster Computing*, IEEE, 2015, pp. 294–303.

[PCX11] Rachida Parks, Chao-Hsien Chu, and Heng Xu, «Healthcare information privacy research: Iusses, gaps and what next?», *in*: (2011).

[PDK15] Rob Patro, Geet Duggal, and Carl Kingsford, «Salmon: accurate, versatile and ultrafast quantification from RNA-seq data using lightweight-alignment», *in*: (2015).

[Pel+20] David Peloquin et al., «Disruptive and avoidable: GDPR challenges to secondary research uses of data», *in*: *European Journal of Human Genetics* 28.*6* (2020), pp. 697–705.

[PG13] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas, «A survey of methods for distributed machine learning», *in*: *Progress in Artificial Intelligence* 2 (2013), pp. 1–11.

[Pin+16] Luis Pineda-Morales et al., «Managing hot metadata for scientific workflows on multisite clouds», *in*: *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 390–397.

[PLZ11] Luca Pireddu, Simone Leo, and Gianluigi Zanetti, «SEAL: a distributed short read mapping and duplicate removal tool», *in*: *Bioinformatics* 27.*15* (2011), pp. 2159–2160.

[Por] ICGC Data Portal, *The Pan-Cancer Analysis of Whole Genomes (PCAWG) study*, `https://dcc.icgc.org/pcawg`, [Online; accessed 10-July-2020].

[PS13] Ram Vinay Pandey and Christian Schlötterer, «DistMap: a toolkit for distributed short read mapping on a Hadoop cluster», *in*: *PloS one* 8.*8* (2013), e72614.

[PST11] Chandra Shekhar Pareek, Rafal Smoczynski, and Andrzej Tretyn, «Sequencing technologies and genome sequencing», *in*: *Journal of applied genetics* 52 (2011), pp. 413–435.

[Puc11] Mario Eduardo Sánchez Puccini, «Executable models for extensible workflow engines», PhD thesis, Citeseer, 2011.

[Qi12] Yanjun Qi, «Random forest for bioinformatics», *in*: *Ensemble machine learning: Methods and applications*, Springer, 2012, pp. 307–323.

[Qui14] J Ross Quinlan, *C4. 5: programs for machine learning*, Elsevier, 2014.

[Qui86] J. Ross Quinlan, «Induction of decision trees», *in*: *Machine learning* 1 (1986), pp. 81–106.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al., «On data banks and privacy homomorphisms», *in*: *Foundations of secure computation* 4.*11* (1978), pp. 169–180.

[Rai+18] Jean Louis Raisaro et al., «Protecting privacy and security of genomic data in i2b2 with homomorphic encryption and differential privacy», *in*: *IEEE/ACM transactions on computational biology and bioinformatics* 15.*5* (2018), pp. 1413–1426.

[RB17] Maria A Rodriguez and Rajkumar Buyya, «Scientific workflow management system for clouds», *in*: *Software architecture for big data and the cloud*, Elsevier, 2017, pp. 367–387.

[RB89] Marina Roesler and Walter A. Burkhard, «Resolution of deadlocks in object-oriented distributed systems», *in*: *IEEE Transactions on Computers* 38.*8* (1989), pp. 1212–1224.

[Ren+18] Jie Ren et al., «Alignment-free sequence analysis and applications», *in*: *Annual Review of Biomedical Data Science* 1 (2018), pp. 93–114.

[Rep18] Presidência da República du Brasil, *Lei Geral de Proteção de Dados Pessoais (LGPD)*, `http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`, [Online; accessed 06-August-2021], 2018.

[Rep19] NIH Data Sharing Repositorie, *NIH Data Sharing Repositories*, `https://www.nlm.nih.gov/NIHbmic/nih_data_sharing_repositories.html`, [Online; accessed 20-Sept-2019], 2019.

[Rep20] Constitution of the Republic of South Africa, *Protection of Personal Information Act (POPI Act)*, `https://popia.co.za/`, [Online; accessed 06-August-2021], 2020.

[RHW19] Yuji Roh, Geon Heo, and Steven Euijong Whang, «A survey on data collection for machine learning: a big data-ai integration perspective», *in*: *IEEE Transactions on Knowledge and Data Engineering* 33.*4* (2019), pp. 1328–1347.

[Ria+18] M Sadegh Riazi et al., «Chameleon: A hybrid secure computation framework for machine learning applications», *in*: *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, pp. 707–721.

[Ris00] Neil J Risch, «Searching for genetic determinants in the new millennium», *in*: *Nature* 405.*6788* (2000), pp. 847–856.

[Rok09] Lior Rokach, «Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography», *in*: *Computational statistics & data analysis* 53.*12* (2009), pp. 4046–4072.

[Rok10] Lior Rokach, «Ensemble-based classifiers», *in*: *Artificial intelligence review* 33 (2010), pp. 1–39.

[Ros61] Frank Rosenblatt, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*, tech. rep., Cornell Aeronautical Lab Inc Buffalo NY, 1961.

[RR13] Zeehasham Rasheed and Huzefa Rangwala, «A map-reduce framework for clustering metagenomes», *in*: *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, IEEE, 2013, pp. 549–558.

[RS04] Laura Elena Raileanu and Kilian Stoffel, «Theoretical comparison between the gini index and information gain criteria», *in*: *Annals of Mathematics and Artificial Intelligence* 41 (2004), pp. 77–93.

[RT+00] Robert B Ross, Rajeev Thakur, et al., «PVFS: A parallel file system for Linux clusters», *in*: *Proceedings of the 4th annual Linux showcase and conference*, 2000, pp. 391–430.

[Ryn+19] Mats Rynge et al., «Integrity protection for scientific workflow data: Motivation and initial experiences», *in*: *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, 2019, pp. 1–8.

[SAG16] Stian Soiland-Reyes, Pinar Alper, and Carole Goble, «Tracking workflow execution with TavernaProv», *in*: *PROV Three Years Later* (2016).

[Sal+16] Salman Salloum et al., «Big data analytics on Apache Spark», *in*: *International Journal of Data Science and Analytics* 1 (2016), pp. 145–164.

[Sana] Troutman Sanders, *Data Privacy: The Current Legal Landscape 2018*, `https://www.troutman.com/images/content/1/9/v2/198033/Data-Privacy-Newsletter-Jan2019.pdf`, Report, Accessed: 2021-09-30.

[Sanb] Jessica Santos, *International landscape of privacy legislation, past, present and future*, `https://www.kantar.com/inspiration/research-services/global-privacy-landscape-for-2019-and-beyond`, Report, Accessed: 2021-09-30.

[Sav+20] Sinem Sav et al., «Poseidon: Privacy-preserving federated neural network learning», *in*: *arXiv preprint arXiv:2009.00349* (2020).

[SB18] Richard S Sutton and Andrew G Barto, *Reinforcement learning: An introduction*, MIT press, 2018.

[Sch+04] Thomas G Schulze et al., «Defining haplotype blocks and tag single-nucleotide polymorphisms in the human genome», *in*: *Human Molecular Genetics* 13.*3* (2004), pp. 335–342.

[Sch+10a] Eric E Schadt et al., «Computational solutions to large-scale data management and analysis», *in*: *Nature reviews genetics* 11.*9* (2010), pp. 647–657.

[Sch+10b] Michael C Schatz et al., «De novo assembly of large genomes using cloud computing», *in*: *Proceedings of the Cold Spring Harbor Biology of Genomes Conference*, 2010.

[Sch08] Michael C Schatz, «BlastReduce: high performance short read mapping with MapReduce», *in*: *University of Maryland, http://cgis. cs. umd. edu/-Grad/scholarlypapers/papers/MichaelSchatz. pdf* (2008).

[Sch09] Michael C Schatz, «CloudBurst: highly sensitive read mapping with MapReduce», *in*: *Bioinformatics* 25.*11* (2009), pp. 1363–1369.

[SCJ17] Disha Shrivastava, Santanu Chaudhury, and Dr Jayadeva, «A data and model-parallel, distributed and scalable framework for training of deep networks in apache spark», *in*: *arXiv preprint arXiv:1708.05840* (2017).

[Sco00] Michael Lee Scott, *Programming language pragmatics*, Morgan Kaufmann, 2000.

[SD98] Marina Skurichina and Robert PW Duin, «Bagging for linear classifiers», *in*: *Pattern Recognition* 31.*7* (1998), pp. 909–930.

[Sen+18] Izzet F Senturk et al., «A resource provisioning framework for bioinformatics applications in multi-cloud environments», *in*: *Future Generation Computer Systems* 78 (2018), pp. 379–391.

[SH02] Frank B Schmuck and Roger L Haskin, «GPFS: A Shared-Disk File System for Large Computing Clusters.», *in*: *FAST*, vol. 2, 19, 2002.

[SH89] Beverly A Sanders and Philipp A Heuberger, «Distributed deadlock detection and resolution with probes», *in*: *Distributed Algorithms: 3rd International Workshop Nice, France, September 26–28, 1989 Proceedings 3*, Springer, 1989, pp. 207–218.

[Shi+17] Benjamin Shickel et al., «Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis», *in*: *IEEE journal of biomedical and health informatics* 22.*5* (2017), pp. 1589–1604.

[Shi99] Y-S Shih, «Families of splitting criteria for classification trees», *in*: *Statistics and Computing* 9.*4* (1999), pp. 309–315.

[Sho+14] Edward H Shortliffe et al., *Biomedical informatics: computer applications in health care and biomedicine*, Springer, 2014.

[Sho12] Edward Shortliffe, *Computer-based medical consultations: MYCIN*, vol. 2, Elsevier, 2012.

[Sil+17] Rafael Ferreira da Silva et al., «A characterization of workflow management systems for extreme-scale applications», *in*: *Future Generation Computer Systems* 75 (2017), pp. 228–238.

[Sin89] Mukesh Singhal, «Deadlock detection in distributed systems», *in*: *Computer* 22.*11* (1989), pp. 37–48.

[SL09] Marina Sokolova and Guy Lapalme, «A systematic analysis of performance measures for classification tasks», *in*: *Information processing & management* 45.*4* (2009), pp. 427–437.

[SL91] S Rasoul Safavian and David Landgrebe, «A survey of decision tree classifier methodology», *in*: *IEEE transactions on systems, man, and cybernetics* 21.*3* (1991), pp. 660–674.

[Sou+20] Lucas Airam C de Souza et al., «DFedForest: Decentralized federated forest», *in*: *2020 IEEE International conference on blockchain (blockchain)*, IEEE, 2020, pp. 90–97.

[SP15] Idafen Santana-Perez and María S Pérez-Hernández, «Towards reproducibility in scientific workflows: An infrastructure-based approach», *in*: *Scientific Programming* 2015 (2015).

[Spj+15] Ola Spjuth et al., «Experiences with workflows for automating data-intensive bioinformatics», *in*: *Biology direct* 10.*1* (2015), pp. 1–12.

[SPS88] Peter Szolovits, Ramesh S Patil, and William B Schwartz, «Artificial intelligence in medical diagnosis», *in*: *Annals of internal medicine* 108.*1* (1988), pp. 80–87.

[SR18] Omer Sagi and Lior Rokach, «Ensemble learning: A survey», *in*: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.*4* (2018), e1249.

[SS16] Ignacio San Segundo-Val and Catalina S Sanz-Lozano, «Introduction to the gene expression analysis», *in*: *Molecular genetics of asthma* (2016), pp. 29–43.

[SSK15] Alexei A Sharov, David Schlessinger, and Minoru SH Ko, «ExAtlas: An interactive online tool for meta-analysis of gene expression data», *in*: *Journal of bioinformatics and computational biology* 13.*06* (2015), p. 1550019.

[Sta+19] Zornitza Stark et al., «Australian genomics: a federated model for integrating genomics into healthcare», *in*: *The American Journal of Human Genetics* 105.*1* (2019), pp. 7–14.

[Ste+15] Zachary D Stephens et al., «Big data: astronomical or genomical?», *in*: *PLoS biology* 13.*7* (2015), e1002195.

[Sub+05] Aravind Subramanian et al., «Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles», *in*: *Proceedings of the National Academy of Sciences* 102.*43* (2005), pp. 15545–15550.

[Swe00] Latanya Sweeney, «Simple demographics often identify people uniquely», *in*: *Health (San Francisco)* 671.*2000* (2000), pp. 1–34.

[SYP17] Kaz Sato, Cliff Young, and David Patterson, «An in-depth look at Google's first Tensor Processing Unit (TPU)», *in*: *Google Cloud Big Data and Machine Learning Blog* 12 (2017).

[Taf+17] Ahmad P Tafti et al., «Machine learning-as-a-service and its application to medical informatics», *in*: *Machine Learning and Data Mining in Pattern Recognition: 13th International Conference, MLDM 2017, New York, NY, USA, July 15-20, 2017, Proceedings 13*, Springer, 2017, pp. 206–219.

[Tan+01] Todd Tannenbaum et al., «Condor: a distributed job scheduler», *in*: *Beowulf cluster computing with windows*, 2001, pp. 307–350.

[Tan+10] Wei Tan et al., «CaGrid Workflow Toolkit: A taverna based workflow tool for cancer grid», *in*: *BMC bioinformatics* 11 (2010), pp. 1–12.

[Tan+16] Haixu Tang et al., «Protecting genomic data analytics in the cloud: state of the art and opportunities», *in*: *BMC medical genomics* 9 (2016), pp. 1–9.

[Tay+07a] Ian Taylor et al., «The triana workflow environment: Architecture and applications», *in*: *Workflows for e-science: Scientific workflows for grids* (2007), pp. 320–339.

[Tay+07b] Ian J Taylor et al., *Workflows for e-Science: scientific workflows for grids*, vol. 1, Springer, 2007.

[Ter14] Sharon F Terry, «The global alliance for genomics & health», *in*: *Genet Test Mol Biomarkers* 18.*6* (2014), pp. 375–6.

[TGL18] Cheng Tang, Damien Garreau, and Ulrike von Luxburg, «When do random forests fail?», *in*: *Advances in neural information processing systems* 31 (2018).

[TTL05] Douglas Thain, Todd Tannenbaum, and Miron Livny, «Distributed computing in practice: the Condor experience», *in*: *Concurrency and computation: practice and experience* 17.*2-4* (2005), pp. 323–356.

[TZL18] Ngoc Hieu Tran, Xianglilan Zhang, and Ming Li, «Deep omics», *in*: *Proteomics* 18.*2* (2018), p. 1700319.

[Uff+21] Emil Uffelmann et al., «Genome-wide association studies», *in*: *Nature Reviews Methods Primers* 1.*1* (2021), p. 59.

[Val+18] Patrick Valduriez et al., «Scientific data analysis using data-intensive scalable computing: The scidisc project», *in*: *LADaS: Latin America Data Science Workshop*, vol. 2170, CEUR-WS. org, 2018.

[Van+10] Marten Van Dijk et al., «Fully homomorphic encryption over the integers», *in*: *Advances in Cryptology–EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30–June 3, 2010. Proceedings 29*, Springer, 2010, pp. 24–43.

[VC11] Tran Van Hung and Huang Chuanhe, «An effective data placement strategy in main-memory database cluster», *in*: *2011 Second International Conference on Networking and Distributed Computing*, IEEE, 2011, pp. 93–98.

[Ver+20] Joost Verbraeken et al., «A survey on distributed machine learning», *in*: *Acm computing surveys (csur)* 53.*2* (2020), pp. 1–33.

[VH04] Peter Van Roy and Seif Haridi, *Concepts, techniques, and models of computer programming*, MIT press, 2004.

[Vig+02] Alain Vignal et al., «A review on SNP and other types of molecular markers and their use in animal genetics», *in*: *Genetics selection evolution* 34.*3* (2002), pp. 275–305.

[Vis+12] Peter M Visscher et al., «Five years of GWAS discovery», *in*: *The American Journal of Human Genetics* 90.*1* (2012), pp. 7–24.

[Vis+17] Peter M Visscher et al., «10 years of GWAS discovery: biology, function, and translation», *in*: *The American Journal of Human Genetics* 101.*1* (2017), pp. 5–22.

[Wan06] Yanyan Wang, «Automating experimentation with distributed systems using generative techniques», PhD thesis, University of Colorado at Boulder, 2006.

[WCA09] Jianwu Wang, Daniel Crawl, and Ilkay Altintas, «Kepler+ Hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems», *in*: *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, 2009, pp. 1–8.

[WCW08] Yanyan Wang, Antonio Carzaniga, and Alexander L Wolf, «Four enhancements to automated distributed system experimentation methods», *in*: *Proceedings of the 30th international conference on Software engineering*, 2008, pp. 491–500.

[WFB13] Stacey J Winham, Robert R Freimuth, and Joanna M Biernacka, «A weighted random forests approach to improve predictive performance», *in*: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 6.*6* (2013), pp. 496–505.

[WG06] Charles B Weinstock and John B Goodenough, *On system scalability*, tech. rep., carnegie-mellon univ pittsburgh pa software engineering inst, 2006.

[WGS09] Zhong Wang, Mark Gerstein, and Michael Snyder, «RNA-Seq: a revolutionary tool for transcriptomics», *in*: *Nature reviews genetics* 10.*1* (2009), pp. 57–63.

[Who20] The ICGC/TCGA Pan-Cancer Analysis of Whole Genomes Consortium, «Pan-cancer analysis of whole genomes», *in*: *Nature* 578.*7793* (2020), pp. 82–93.

[Wie+14] Marek S Wiewiórka et al., «SparkSeq: fast, scalable and cloud-ready tool for the interactive genomic data analysis with nucleotide precision», *in*: *Bioinformatics* 30.*18* (2014), pp. 2652–2653.

[Wil+09] Michael Wilde et al., «Parallel scripting for applications at the petascale and beyond», *in*: *Computer* 42.*11* (2009), pp. 50–60.

[Wil+11] Michael Wilde et al., «Swift: A language for distributed parallel scripting», *in*: *Parallel Computing* 37.*9* (2011), pp. 633–652.

[Wil+16] Mark D Wilkinson et al., «The FAIR Guiding Principles for scientific data management and stewardship», *in*: *Scientific data* 3.*1* (2016), pp. 1–9.

[Wol+13] Katherine Wolstencroft et al., «The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud», *in*: *Nucleic acids research* 41.*W1* (2013), W557–W561.

[WWG21] Laura Wratten, Andreas Wilm, and Jonathan Göke, «Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers», *in*: *Nature methods* 18.*10* (2021), pp. 1161–1168.

[XGL12] Baomin Xu, Jin Gao, and Chunyan Li, «An efficient algorithm for DNA fragment assembly in MapReduce», *in*: *Biochemical and biophysical research communications* 426.*3* (2012), pp. 395–398.

[Xie+10] Jiong Xie et al., «Improving mapreduce performance through data placement in heterogeneous hadoop clusters», *in*: *2010 IEEE international symposium on parallel & distributed processing, workshops and Phd forum*, IEEE, 2010, pp. 1–9.

[Xie08] Tao Xie, «Sea: A striping-based energy-aware strategy for data placement in raid-structured storage systems», *in*: *IEEE Transactions on Computers* 57.*6* (2008), pp. 748–761.

[Xin+15] Eric P Xing et al., «Petuum: A new platform for distributed machine learning on big data», *in*: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 1335–1344.

[Xu+17a] Bo Xu et al., «DSA: scalable distributed sequence alignment system using SIMD instructions», *in*: *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, 2017, pp. 758–761.

[Xu+17b] Bo Xu et al., «Efficient distributed smith-waterman algorithm based on apache spark», *in*: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, IEEE, 2017, pp. 608–615.

[Xu+21] Jie Xu et al., «Federated learning for healthcare informatics», *in*: *Journal of Healthcare Informatics Research* 5 (2021), pp. 1–19.

[Yan+10] Pengyi Yang et al., «A review of ensemble methods in bioinformatics», *in*: *Current Bioinformatics* 5.*4* (2010), pp. 296–308.

[Yan+19] Qiang Yang et al., «Federated learning», *in*: *Learning* 13.*3* (2019), pp. 1–207.

[Yao86] Andrew Chi-Chih Yao, «How to generate and exchange secrets», *in*: *27th annual symposium on foundations of computer science (Sfcs 1986)*, IEEE, 1986, pp. 162–167.

[YB05] Jia Yu and Rajkumar Buyya, «A taxonomy of workflow management systems for grid computing», *in*: *Journal of grid computing* 3 (2005), pp. 171–200.

[YBK18] Kun-Hsing Yu, Andrew L Beam, and Isaac S Kohane, «Artificial intelligence in healthcare», *in*: *Nature biomedical engineering* 2.*10* (2018), pp. 719–731.

[Yin+11] Yong Yin et al., «Data mining: Concepts, methods and applications in management and engineering design», *in*: Springer Science & Business Media, 2011.

[Yu+12] Hsiang-Fu Yu et al., «Large linear classification when data cannot fit in memory», *in*: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5.*4* (2012), pp. 1–23.

[Yua+10] Dong Yuan et al., «A data placement strategy in scientific cloud workflows», *in*: *Future Generation Computer Systems* 26.*8* (2010), pp. 1200–1214.

[ZG09] Xiaojin Zhu and Andrew B Goldberg, «Introduction to semi-supervised learning», *in*: *Synthesis lectures on artificial intelligence and machine learning* 3.*1* (2009), pp. 1–130.

[Zha+11] Junjun Zhang et al., «International Cancer Genome Consortium Data Portal—a one-stop shop for cancer genomics data», *in*: *Database* 2011 (2011).

[Zha+12a] Lu Zhang et al., «Gene set analysis in the cloud», *in*: *Bioinformatics* 28.*2* (2012), pp. 294–295.

[Zha+12b] Jun Zhao et al., «Why workflows break—Understanding and combating decay in Taverna workflows», *in*: *2012 IEEE 8th International Conference on e-Science*, IEEE, 2012, pp. 1–9.

[Zha+15a] Chen Zhang et al., «Optimizing FPGA-based accelerator design for deep convolutional neural networks», *in*: *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, 2015, pp. 161–170.

[Zha+15b] Yuchen Zhang et al., «Foresee: Fully outsourced secure genome study based on homomorphic encryption», *in*: *BMC medical informatics and decision making*, vol. 15, 5, BioMed Central, 2015, pp. 1–11.

[Zha+15c] Yong Zhao et al., «Enabling scalable scientific workflow management in the Cloud», *in*: *Future Generation Computer Systems* 46 (2015), pp. 3–16.

[Zha+16] Qing Zhao et al., «A new energy-aware task scheduling method for data-intensive applications in the cloud», *in*: *Journal of Network and Computer Applications* 59 (2016), pp. 14–27.

[Zha+17] Di Zhang et al., «SEQSpark: a complete analysis tool for large-scale rare variant association studies using whole-genome and exome sequence data», *in*: *The American Journal of Human Genetics* 101.*1* (2017), pp. 115–122.

[Zha+20] Chengliang Zhang et al., «Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning», *in*: *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC 2020)*, 2020.

[Zho+16] Yu Zhou et al., «Gene Expression and Profiling», *in*: *Application of Clinical Bioinformatics* (2016), pp. 59–82.

[Zho+17] Wei Zhou et al., «MetaSpark: a spark-based distributed processing tool to recruit metagenomic reads to reference genomes», *in*: *Bioinformatics* 33.*7* (2017), pp. 1090–1092.

[Zho21] Zhi-Hua Zhou, «Ensemble Learning», *in*: *Machine Learning*, Springer Singapore, 2021, pp. 181–210, ISBN: 978-981-15-1967-3, DOI: 10.1007/978-981-15-1967-3_8.

[Zhu+18] Min Zhu et al., «Class weights random forest algorithm for processing class imbalanced medical data», *in*: *IEEE Access* 6 (2018), pp. 4641–4652.

[Zie+17] Andrzej Zielezinski et al., «Alignment-free sequence comparison: benefits, applications, and tools», *in*: *Genome biology* 18 (2017), pp. 1–17.

[ZLS15] Guoguang Zhao, Cheng Ling, and Donghong Sun, «SparkSW: scalable distributed computing system for large-scale biological sequence alignment», *in*: *2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing*, IEEE, 2015, pp. 845–852.

[ZM12] Cha Zhang and Yunqian Ma, *Ensemble machine learning: methods and applications*, Springer, 2012.

[ZQ11] Matthias Zytnicki and Hadi Quesneville, «S-MART, a software toolbox to aid RNA-Seq data analysis», *in*: *PloS one* 6.*10* (2011), e25988.

[ZZK16] Lifang Zhang, Yan Zheng, and Raimo Kantoa, «A review of homomorphic encryption and its applications», *in*: *Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications*, 2016, pp. 97–106.