Modelado basado en datos para la clasificación semiautomática de correspondencia electrónica: caso de estudio para la Administración Pública Colombiana

Edwin Alberto Vargas Antolínez

Escuela Colombiana de Ingeniería Julio Garavito
Maestría en Gestión de Información
Mayo 2018

Modelado basado en datos para la clasificación semiautomática de correspondencia electrónica: caso de estudio para la Administración Pública Colombiana

Edwin Alberto Vargas Antolínez

Trabajo de grado presentado para optar al grado de
Maestría en Gestión de Información

Asesores:
Victoria Eugenia Ospina, PhD.
Dante Conti, PhD.

Escuela Colombiana de Ingeniería Julio Garavito
Maestría en Gestión de Información
Mayo 2018

Resumen

El uso de correo electrónico a nivel de las organizaciones, como canal de comunicación en procesos de servicio y atención al cliente, ha crecido en los últimos tiempos. Por tanto, las organizaciones han debido implementar procesos focalizados en organizar dichos correos de acuerdo con la temática esencial transmitida en ellos, para así dar una respuesta eficiente ante las solicitudes de los clientes. Una manera de abordar el problema es a través de la categorización de la correspondencia electrónica mediante la extracción del contenido textual en corpus de palabras determinantes (Minería de Texto) que se usan para una posterior clasificación de los correos con técnicas de aprendizaje automático de máquinas (Machine Learning). Este proyecto presenta un enfoque metodológico que evalúa diversos algoritmos de clasificación con técnicas de muestreo aleatorio simple sobre una población de documentos (correos) del registro de correspondencia del Departamento Administrativo de la Función Pública en Colombia, como caso de estudio. La investigación se detalla en un sistema paso a paso, desde el preprocesamiento de la información, reducción de la dimensionalidad, selección de diversas muestras hasta la aplicación de algoritmos de clasificación. El modelado incluye un benchmarking entre diversos algoritmos: clasificadores de tipo Naive Bayesianos, máquinas de soporte vectorial (SVM) y Boosting. Se propone, además, una arquitectura funcional semiautomática que puede escalarse en futuro en un sistema productivo de gran manejo de datos en tiempo real (streaming) basada en R, Spark y MapReduce. El modelo se pone a prueba logrando valores de “accuracy” superiores al 90% que soportan una buena Línea Base para soluciones en producción para el enfoque empleado en esta investigación.

Tabla de Contenidos

Lista de tablas

Lista de Figuras

Listado de Anexos

I. Introducción

El correo electrónico ha estado presente desde los inicios del internet y sigue siendo un popular medio de comunicación. El número de cuentas de correo electrónico de acuerdo al estudio del Radicati Group en el mundo durante el 2016 fue de 2672 millones y las proyecciones indican que para el 2020 cerca de la mitad de la población mundial utilizará este servicio (Radicati Group, Inc., 2016). Mientras que en Colombia de acuerdo a los indicadores reportados por del DANE sobre tenencia y uso de Tecnologías de Información durante el 2015, el 59.6% de la población usó internet y de estos el 55% la utiliza para comunicarse a través de correo electrónico (Departamento Administrativo Nacional de Estadística (DANE), 2016).

Este uso masivo del correo electrónico ha llevado a que típicamente un usuario reciba docenas de correos al día, los cuales requieren ser organizados a menudo en carpetas relacionados por tema (Meek & Brutlag, 2000), al nivel de las organizaciones, muchas empresas basan su proceso de servicio al cliente en este tipo de comunicaciones de manera que han tenido que enfrentar el problema de clasificar los correos de acuerdo a su contenido textual. En el ámbito mundial los problemas de clasificación de correos generalizado a clasificación de documentos, sigue siendo un tema de investigación prominente, que utiliza diversas técnicas y sus combinaciones en sistemas aún más complejos, una muestra del mapa de investigaciones en este tópico se pueden consultar en estudios empíricos que comparan la efectividad y en algunos casos la eficiencia de los modelos de clasificación de texto (Dumais, Platt, & Heckerman, 1998) (Sebastiani, 2002) (Mironczuk & Protasewikz, 2018).

En el contexto del Gobierno Nacional de Colombia y en lo referente al uso que los ciudadanos hacen del correo electrónico para solicitar y recibir información en el caso particular del Departamento Administrativo de la Función Pública entidad de la Rama Ejecutiva del Gobierno de Colombia, se ha evidenciado desde 2010 que la correspondencia electrónica se viene casi duplicando año a año de manera que por ejemplo en 2015 se recibieron 23 mil comunicaciones y en el 2016 se recibieron más de 40 mil. De otro lado, las entidades del gobierno tienen un plazo en el que deben dar respuesta conforme a lo establecido en el Código de Procedimiento Administrativo y de lo Contencioso Administrativo (Congreso de Colombia, 2011) y no atender la solicitud o el retardo injustificado acarrea sanciones para los funcionarios de las dependencias encargados de dar respuestas y el representante legal.

Este incremento en el número de correspondencia electrónica recibida por la entidad ha llevado a gastar más y más tiempo clasificando la correspondencia por su contenido textual para establecer cual Dependencia (categoría) tiene la competencia para dar respuesta al ciudadano. Esto ha traído congestión en la tarea de clasificación realizada por un grupo de expertos, siendo uno de los factores que ha llevado a que el tiempo promedio en dar respuesta a las peticiones de consulta de información se haya incrementado de 21 días en el 2013 a casi 27 días en el 2016 ubicándose cada vez más cerca del límite de 30 días para dar respuesta a este tipo de consulta. En tanto que las peticiones que no son competencia del departamento cuyo límite fijado por la ley es de 5 días tienen un tiempo promedio de respuesta de 3.5 días (Superintendencia de Servicios Públicos Domiciliarios, 2006), siendo

que al no requerir elaborar una respuesta para este tipo de la solicitud de información podrían ser resulta apenas se recibe.

Por lo ya expuesto, el presente trabajo tiene el objetivo diseñar un modelo predictivo adecuado para la tarea de clasificación de correos electrónicos que contribuya a disminuir el tiempo en dar respuesta a la correspondencia recibida por el Departamento Administrativo de la Función Pública.

En adelante el lector podrá de manera breve conocer la entidad Función Pública, profundizar en el proceso de Administración de la Correspondencia y sus factores preponderantes, lo que lleva al planteamiento de los objetivos. Además, se presenta el sustento teórico sobre el que se construye el modelo, la metodología propuesta para evaluación y selección del modelo y finalmente se consignan los resultados obtenidos con el caso de estudio y las conclusiones derivadas de la investigación.

Enfoque de resolución del problema

El Departamento Administrativo de la Función Pública es una entidad de la Rama Ejecutiva del Gobierno Nacional de Colombia que contribuye al cumplimiento de los compromisos del gobierno con el ciudadano asesorando a la presidencia de la república en toma de decisiones de creación, reestructuración o eliminación de entidades del gobierno, el diseño de instituciones públicas eficientes y eficaces, y con la simplificación de trámites del ciudadano frente al Estado. Con el ánimo de gestionar los requerimientos formulados por los usuarios de la entidad se cuenta con un modelo de servicio al ciudadano compuesto por los canales de atención al usuario; escrito mediante el formulario Web de Peticiones Quejas y Reclamos PQR, correo electrónico y postal, presencial en la ventanilla de la entidad,

telefónico a las líneas de atención al usuario y virtual mediante una herramienta de chat. Además, el modelo contempla un proceso de Administración de Correspondencia cuyo objetivo es la recepción y radicación de los documentos físicos o electrónicos que ingresan a través de los diferentes canales.

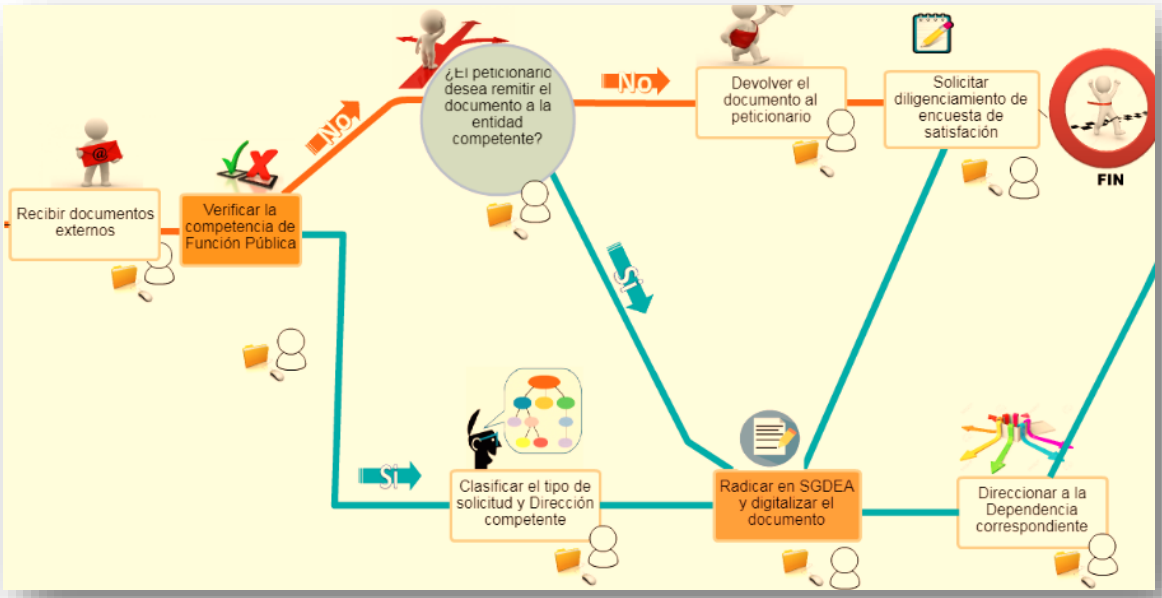


Figura 1. Actividades del proceso de administración de correspondencia, tomado de la intranet de Función Pública (Función Pública, 2017).

En las primeras etapas del proceso, ver la figura 1, el requerimiento plasmado en un documento es recibido e ingresado al sistema de información ORFEO el cual le asigna un número de radicado y se deja constancia de la fecha y hora de recepción con el propósito de oficializar su trámite y cumplir con los términos establecidos por la ley (Código de Procedimiento Administrativo y de lo Contencioso Administrativo, 2011). Posteriormente

es clasificado por un funcionario experto quien asigna la categoría a la que corresponde el documento para ser direccionado a la Dependencia con competencia para dar respuesta al requerimiento, como apoyo el experto cuenta con un documento guía que relaciona los temas y subtemas con las 17 dependencias de la entidad, ver anexo A; extracto del Acta de direccionamiento de correspondencia.

Para clasificar correctamente la correspondencia, el funcionario debe dar lectura completa al documento y entender la solicitud del peticionario o ciudadano y asignar la Dependencia que maneja el tema, en realidad para hacerlo no basta con la guía, se requiere de la experiencia, dado que hay cuestiones no contemplados en la guía y que por su complejidad no pueden ser contrastados directamente con los temas en el Acta, sino que dependiendo del Remitente, Asunto u otros elementos de la comunicación, son resueltos comúnmente por una u otra Dependencia. En fin, hay una serie de reglas tácitas que están en cabeza de las personas, con lo cual el conocimiento de estos expertos es crítico para la entidad, que además se les pide hacer su trabajo con celeridad y siempre estar disponibles.

En respuesta a lo anterior, se presenta este trabajo de grado con el propósito de codificar el conocimiento en clasificación de correspondencia por lo que se traza el objetivo de diseñar un modelo predictivo, de tal manera que el conocimiento pueda ser trasmitido a una computadora para que esta realice la tarea con un rendimiento similar al obtenido por el experto, de modo que pueda sugerir rápidamente como se debe clasificar la correspondencia, que se adapte al crecimiento de la correspondencia y que además tenga la cualidad de estar disponible cuando se le necesite.

La correspondencia electrónica que hace parte de este estudio tiene diferentes limitaciones las cuales serán detalladas cuando se haga referencia a la población y muestra del objeto de estudio.

Objetivo general

Diseñar un modelo de clasificación automática de correspondencia electrónica en el Departamento Administrativo de la Función Pública.

Objetivos específicos

- Caracterizar la situación actual del manejo de la correspondencia electrónica en el Departamento Administrativo de la Función Pública.
- Definir las variables para representar el corpus de documentos constitutivos del modelo.
- Diseñar el modelo predictivo para clasificación automática de documentos.
- Diseñar estrategias para conseguir que el modelo propuesto tenga mayor grado de escalabilidad y disponibilidad.

Hipótesis

Como se ha expuesto, el objetivo planteado es diseñar un modelo predictivo de clasificación de correspondencia electrónica, escalable y disponible. De manera que la hipótesis sobre la que más adelante se construye la metodología, se centra en la cualidad

de exactitud del modelo; por lo tanto, la **hipótesis** es: El grado de efectividad[1] del modelo de clasificación de correspondencia está determinado por los términos que hacen parte del corpus de documentos y el modelo de clasificación.

Variables que intervienen en la formulación de hipótesis:

- Variable independiente: Documentos X_i representados por un vector de términos que hacen parte del corpus de documentos $D = \{X_1, X_2, X_3, X_4 \dots X_n\}$.
- Variable dependiente: Variable categórica C_i que identifica la exactitud con la que es clasificado determinado documento, donde i representa la dependencia (clase o categoría) como una variable categórica $i = \{1,2,3, \dots 9\}$
- Variable interviniente: Los modelos de clasificación utilizados durante la experimentación.

Prueba estadística para verificar la Hipótesis: Teniendo en cuenta las características de la hipótesis se usa la exactitud obtenida con la técnica de comprobación basada en la matriz de confusión (Kohavi, Fawcett, & Provost, 1998) .

Vistazo al diseño metodológico

El diseño metodológico es ajustado a la validación de la hipótesis, consiste en una serie de iteraciones en cada uno de los cuales se realiza el proceso de *Machine Learning (ML)* con varios algoritmos y se evalúa el resultado, al final se selecciona el modelo con la mejor exactitud, modelo en el cual se verifica la hipótesis.

[1] La efectividad es la habilidad de tomar la correcta decisión de clasificación, y su medida es la exactitud o "accuracy" en inglés.

II. Marco Teórico

Clasificar o categorizar[2] documentos de textos ha sido un problema abordado desde principios de los años 80 mediante técnicas de ingeniería de conocimiento *(Knowledge Engineering)* usualmente utilizando expertos que crean un conjunto de reglas con los cuales se podía clasificar los documentos que hacen parte del dominio del problema, a partir de los 90's hacen aparición métodos con enfoques basados en Machine Learning(ML) que se destacan por obtener buenos resultados en términos de correcta clasificación y con la ventaja de no depender de expertos.

Durante este capítulo se revisan las técnicas de minería que convierten al texto en un input o representación para que el algoritmo de aprendizaje lo pueda manejar, así como los principales métodos de ML para analítica de texto. Por tanto, se describen los algoritmos de Boosting y Support Vector Machines (SVM). Al final del capítulo se analiza la arquitectura funcional que permita ejecutar los modelos de forma distribuida, lo que le confiere al modelo escalabilidad, adaptabilidad al crecimiento y aumento en la disponibilidad, propiedades requeridas para llevarlo a un sistema productivo en su posible implementación dentro de la organización.

[2] Clasificar es el proceso por el cual se trata de establecer la clase correcta para un objeto (Guus T. Schreiber, 2000),

Machine Learning

El Aprendizaje de máquina o Machine Learning[3] surge a partir de la Inteligencia Artificial con la idea de que el mejor camino para hacer una computadora inteligente es mediante el uso de algoritmos de aprendizaje que imiten el mecanismo de aprendizaje, esto es, que la máquina aprenda a realizar una nueva tarea a medida que ensaya y se equivoca o acierta, mejorando así con la experiencia.

Fuer Arthur Samuel quien tuvo la idea de hacer que una computadora participara en un concurso de "juego de damas chinas" y se empeñó en ello durante varios años mucho antes de que las computadoras estuvieran en capacidad de participar en el juego, no fue sino hasta la década 50 y parte de los 60 cuando trabajando para IBM, se le permitió instalar el programa con el juego de damas chinas en varias computadoras IBM y Arthur las colocó a jugar entre sí miles de veces, de manera que se registraran cuales jugadas eran mejores que otros por su tendencia a ganar el juego y así jugando miles de veces para ganar experiencia mediante la repetición y observación de los mejores resultados, al punto que el programa jugó mejor que lo que su autor podía hacerlo.

Pues bien, formalmente, y utilizando la definición del propio Arthur, aprendizaje de maquina es el campo de estudio que da a las computadoras la habilidad de aprender sin ser programadas explícitamente". Más recientemente Tom Mitchell autoridad mundial de esta materia, actualizó esta definición la cual describe más concretamente lo realizado en esta investigación; Aprendizaje automático es un campo cuyo objetivo es construir programas

[3] Machine Learning es comúnmente traducido al español como aprendizaje de máquina, también es usado aprendizaje automático, aunque en menor proporción.

que mejoren su rendimiento en alguna tarea a través de la experiencia (Mitchell, 1997). Cabe anotar que Machine Learning (ML) es multidisciplinario basado en resultados de estadísticos, probabilidades, teoría de información y otros. Además, comparte algoritmos y otros elementos comunes con disciplinas como la minería de datos, el procesamiento de lenguaje natural y es usado en aplicaciones industriales como el piloto automático de Tesla Motors o las herramientas de auto recomendación de Amazon y Netflix.

Ya en clasificación de correos electrónicos, problema generalizado a la clasificación de texto, el paradigma de ML, según el cual a partir de un conjunto de documentos preclasificados (experiencia) y mediante un método inductivo se construye un clasificador automático de texto, tiene significativas ventajas dado que se ha logrado obtener una exactitud comparable a la alcanzada por expertos.

Proceso de Machine Learning

Típicamente ML es un proceso que tiene varias fases, ver figura 2, como si dijo el algoritmo es restrictivo con los datos a la entrada, de modo que deben tener la "forma correcta", entonces ¿Cuál es la forma correcta? En primer lugar, depende del algoritmo, siendo diferente por ejemplo para un "árbol de decisión" que para una "regresión", es por ello que los datos a la entrada pueden ser listas de símbolos, vectores alfabéticos, o matrices numéricas discretas, continuas, binarias o decimales. En segundo lugar, los datos con los que se trabaja en aplicaciones de ML suelen ser no estructurados, con ruido, inconsistentes, incompletos, de manera que la etapa de pre-procesamiento se encarga de extraer los datos, limpiarlos y transformarlos en la forma que sean requeridos por la siguiente fase. Ahora bien, esta fase no se hace una sola vez y termina su utilidad, por el contrario, es normal que

se vuelva a ella buscando mejorar la exactitud en la fase de aprendizaje, ya que buena parte de su efectividad dependerá de que tan “limpios” estén los datos que le damos para aprender.

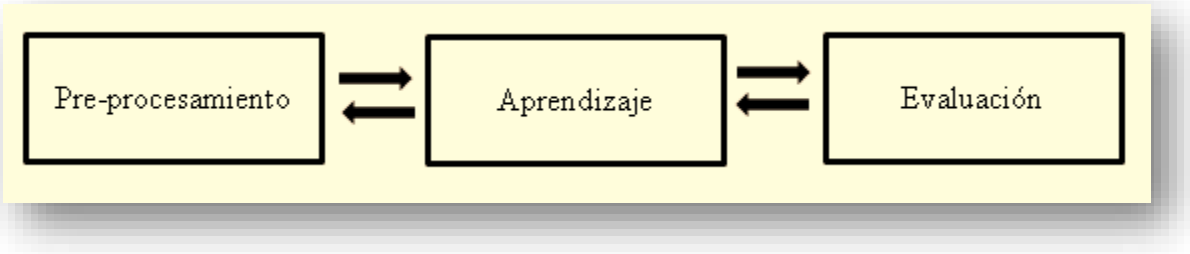


Figura 2. Proceso de machine Learning

En la fase de aprendizaje, es donde se ejecuta el algoritmo y se establecen los parámetros del modelo que va realizar la predicción o descubrir los patrones.

En la etapa final, la de evaluación se mide el rendimiento del modelo, lo cual se hace en un conjunto de datos diferente al utilizado en la fase de aprendizaje y tiene que ver con tareas como estimación del error y pruebas estadísticas.

Usualmente cuando en analista requiere ejecutar el proceso de ML recurre a una plataforma la cual ya tiene implementados los algoritmos y presenta un entorno de desarrollo que da facilidades para ejecutar las fases de las que venimos hablando con funcionalidades que permitan importar, exportar datos y visualizarlos gráficamente, entre otras características. En este sentido, las plataformas de ML más usadas de código abierto son Python y R (R Core, 2014), en tanto que comercialmente Alteryx, KNIME, RapidMiner son algunas de las más valoradas.

Algoritmos de Clasificación de Texto

A continuación, se describen algunos de los más efectivos algoritmos de aprendizaje utilizados en tareas de clasificación de texto de acuerdo al estudio comparativo de Sebastiani (Sebastiani, 2002), ver figura 3, el cual, utilizando el conjunto de documentos de Reuters[4] concluye que los métodos SVM y ADAboost(Adaptative Boosting) tienen los mejores resultados en términos de exactitud, estos resultados son consistentes con los obtenidos por múltiples investigaciones comparativas (Cardoso-Cachopo & Limede Oliveira, 2003) (Rousu, Saunders, Szedmak, & Shawe-Taylor, 2005) (Dumais, Platt, & Heckerman, 1998).

Table VI. Comparative Results Among Different Classifiers Obtained on Five Different Versions of Reuters. (Unless otherwise noted, entries indicate the microaveraged breakeven point; within parentheses, "M" indicates macroaveraging and "F_1" indicates use of the F_1 measure; **boldface** indicates the best performer on the collection)

			#1	#2	#3	#4	#5
		# of documents	21,450	14,347	13,272	12,902	12,902
		# of training documents	14,704	10,667	9,610	9,603	9,603
		# of test documents	6,746	3,680	3,662	3,299	3,299
		# of categories	135	93	92	90	10
System	**Type**	**Results reported by**					
WORD	(non-learning)	Yang [1999]	.150	.310	.290		
	probabilistic	[Dumais et al. 1998]				.752	.815
	probabilistic	[Joachims 1998]				.720	
	probabilistic	[Lam et al. 1997]	.443 (MF_1)				
PROPBAYES	probabilistic	[Lewis 1992a]	.650				
BIM	probabilistic	[Li and Yamanishi 1999]				.747	
	probabilistic	[Li and Yamanishi 1999]				.773	
NB	probabilistic	[Yang and Liu 1999]				.795	
CLASSI	neural network	[Ng et al. 1997]		.802			
NNET	neural network	Yang and Liu 1999]				.838	
	neural network	[Wiener et al. 1995]			**.820**		
GIS-W	example-based	[Lam and Ho 1998]				.860	
k-NN	example-based	[Joachims 1998]				.823	
k-NN	example-based	[Lam and Ho 1998]				.820	
k-NN	example-based	[Yang 1999]	.690	.852	**.820**		
k-NN	example-based	[Yang and Liu 1999]				.856	
	SVM	[Dumais et al. 1998]				.870	**.920**
SVMLIGHT	SVM	[Joachims 1998]				.864	
SVMLIGHT	SVM	[Li Yamanishi 1999]				.841	
SVMLIGHT	SVM	[Yang and Liu 1999]				.859	
ADABOOST.MH	committee	[Schapire and Singer 2000]		**.860**			
	committee	[Weiss et al. 1999]				**.878**	
	Bayesian net	[Dumais et al. 1998]				.800	.850
	Bayesian net	[Lam et al. 1997]	.542 (MF_1)				

Figura 3. Exactitud comparada de varios modelos de clasificación de texto, recuperado del estudio comparativo de clasificadores de texto (Sebastiani, 2002)

[4] Colección de noticias de propósito experimental liberadas por la agencia de Noticias Reuters

Máquina de Soporte Vectorial

Máquina de soporte Vectorial es la traducción de Support Virtual Machine(SVM) el cual es un algoritmo de minimización del error de estimación en clasificación basado en el trabajo de Vladimir Vapnik, el algoritmo construye una línea recta(en el plano bidimensional) que generalizado a espacios multidimensionales es llamado hiperplano, el cual separa las diferentes clases del conjunto de datos de entrenamiento de manera que se establezca el plano cuyo margen o separación entre las clases sea más amplio y por lo tanto es el hiperplano óptimo. (Vapnik , 1982)

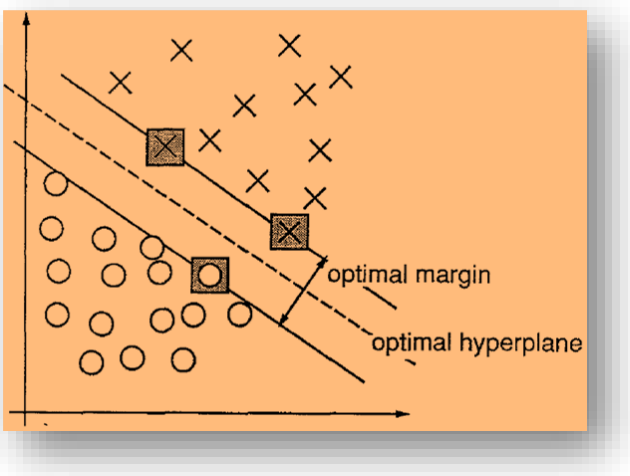


Figura 4. Margen de separación de dos clases linealmente separables (Vapnik et al. 1995)

En la figura 4, se ha modelado un problema de clasificación con dos variables de entrada y dos clases (círculos y cruces), lo cual permite representarlo gráficamente, la línea recta

discontinúa representa el plano optimo y la distancia entre este plano y las líneas paralelas es el límite de decisión, cuyo objetivo es encontrar el hiperplano que tenga el margen más amplio. El mismo Vapnik hizo la generalización para cuando los elementos no son linealmente separables utilizando una transformación denominada kernels, cuyo truco consciente en aumentar la dimensión del espacio cartesiano hasta que los elementos sean separables por un hiperplano.

SVN ha sido ha sido diseñado principalmente como clasificador binario, de manera que para SVN y otros clasificadores binarios hay varias vías para combinar varios de ellos en un solo clasificador multiclase. El esquema de uno contra el resto *one-versus-rest*, consiste en entrenar k clasificadores binarios, el primero de los cuales separa una clase del resto, el segundo separa la segunda clase del resto, y así sucesivamente.

Ya en clasificación de textos, la introducción de SVM fue hecha por Joachims (JOACHIMS, 1998), quien contrastó algunos de los principales métodos conocidos al momento de su investigación, encontrando que con SVM se obtiene mejor exactitud en una variedad de tareas de clasificación de texto.

Boosting

Boosting es la idea básica tras varios métodos simples, pero eso sí, altamente efectivos que construye un modelo de predicción combinando las fortalezas de un grupo de modelos más sencillos, para reducir la varianza y el sesgo, inicialmente desarrollado para problemas de clasificación.

Suponiendo que se entrena un clasificador con alguno de los algoritmos que se reconocieron anteriormente; SVM o ANN, en varios subconjuntos de datos tomados aleatoriamente del conjunto de datos de entrenamiento original, En la figura 5 se e ilustra un posible resultado que se obtendría con tres modelos A, B y C al separar dos clases. Cada número se puede imaginar como un subconjunto de observaciones azules o negras, entonces; la línea A, es buen clasificador para los miembros del subconjunto 3, 4, 5, 6,7 azules y 1, 2,3 negros, no así con en el subconjunto 4 y en algunos miembros del subconjunto 5 negro, en particular para el 4 y 5 negro quizás el modelo B sea el mejor, de manera que el modelo B obtendrá más exactitud que el modelo A en estos subconjuntos. Para aumentar la diversidad usualmente también se hace un muestreo de las variables o parámetros de cada subconjunto de entrenamiento, de manera que se obtienen otros subconjuntos de entrenamiento derivados de las primeros, esta serie de subconjuntos/entrenamientos forma un árbol de modelos. Al final se eligen la clasificación de cada ejemplo de la predicción de la mayoría de modelos o promediando los resultados de los modelos, en consecuencia, el modelo final ya no es lineal, este método toscamente ilustrado es el denominado *Random Forest.*

Pues bien, basado en el *Random Forest* se creó el *Adaptative Boosting* (Freund & Schapire, 1997), pero con una forma más sofisticada de crear los diversos subconjuntos.

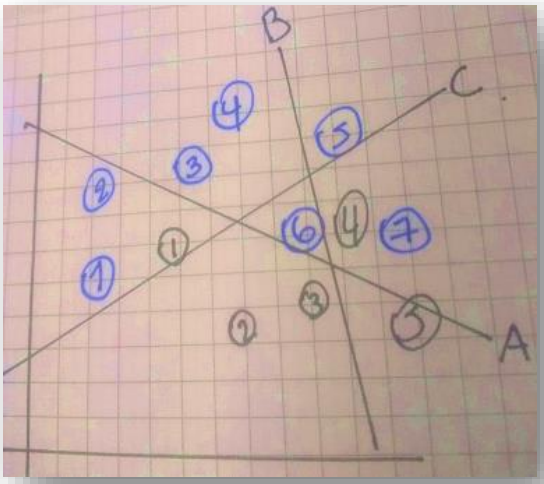


Figura 5. Tres Clasificadores lineales A, B, C

En ADAboost se calcula el error de entrenamiento en un primer modelo y cada observación mal clasificada es actualizada multiplicándola por el peso con relación el error, lo que deriva en un nuevo conjunto de datos y es ejecutado nuevamente el algoritmo que se fijará más en aquellas observaciones con mayor peso, con lo cual el nuevo modelo reduce el error, este proceso puede repetirse muchas veces inclusive hasta que el error de entrenamiento sea muy cercano a cero. Aunque suena bien, el error cero conduce al dilema sesgo y varianza, consiste que entre menor sea el error de entrenamiento seguramente la línea(ya sería una curva) que separa las dos clases hipotéticas de que se venían hablando, sigue perfectamente cada observación del conjunto de entrenamiento, el resultado es alta dependencia del conjunto de entrenamiento con lo cual resultado del modelo en el conjunto de datos de validación puede no ser tan bueno aumentando la varianza, en general se dirá

que un modelo más complejo aumenta la varianza en tanto que entre más simple el modelo, puede tender a estar sesgado.

Una variante de ADAboost es el Gradient Boosting Tree (GBT) (Friedman, 2001) el cual se diferencia del primero en que para nueva ejecución del algoritmo, esta se hace sobre un subconjunto compuesto únicamente por aquellas observaciones que tuvieron error en la clasificación anterior, es otra manera de darle más importancia a las observaciones con error.

Pre-procesamiento

En esta fase se utilizan algunas técnicas de minería de textos, de manera que se obtenga la representación de los textos de correspondencia electrónica que permita más adelante hacer el análisis de la información. Usualmente los datos en texto son representados como "bolsa de palabras" aunque también pueden ser tratados de forma más sofisticada, se dirá "semánticamente", esto es haciendo distinciones entre los sustantivos, como personas, organizaciones y lugares y sus asociaciones. No obstante, el estado de arte en métodos de procesamiento de lenguaje natural mediante una representación semántica no obtiene aún resultados suficientemente buenos contrastados con la dificultad de obtención de la representación semántica, por lo que "bolsa de palabras" se mantiene como la más sencilla y efectiva técnica de representación del texto, así las cosas, es la utilizada en el trascurso de este trabajo.

Bolsa de Palabras

La representación más simple del texto del documento es a través de una lista de palabras, en la cual dentro de un documento D, no importa cuántas veces se repita una palabra, es representada una única vez, ahora entonces denomina *término*:

$$D = (término1, término2, terminoN)$$

Para obtener este vector de términos en primer lugar se debe tokenizar el texto, es decir separar las palabras como instancias, por ejemplo, Santa-Fe es dividida en dos términos, Santa y Fe. Una vez hecho esto se puede obtener el vector numérico del documento en donde a cada termino se le asigna un numero escalar de acuerdo a la relevancia o peso que este tenga dentro del texto, a la representación numérica del corpus de documentos[5], es decir el conjunto de vectores y la matriz resultante se le denomina Bolsa de Palabras o Bag of Words (BoW). Pues bien, en esta matriz de términos puede ser binaria, según si el término está o no en el documento (no importa la frecuencia), también puede tener una representación con un número racional cuando se utiliza alguna función con el fin de estimar el peso de la palabra dentro del documento, como por ejemplo TFIDF[6].

Finalmente, una de las representaciones más sencillas, pero con mayor efectividad es mediante un numero natural que representa la frecuencia de aparición de determinada palabra en el texto (Buckley, 1988). En este caso, para determinar la relevancia de cada

[5] Conjunto de documentos que hacen parte de la tarea de clasificación.
[6] Función en que asigna pesos con el criterio según el cual cuanto más a menudo un término aparece en el documento más representa su contenido, y de otro lado, disminuye su importancia si tiene alta frecuencia de aparición en el corpus de documentos.

término en el documento se le asigna un peso a cada término de acuerdo a la frecuencia que tienen dentro del documento (ver tabla 1).

	empleo	**desarrollo**	**mérito**	**municipio**
Doc1	7	5	2	1
Doc2	4	7	0	0
Doc3	7	3	9	0
Doc4	13	0	14	1

Tabla 1: Bolsa de palabras con la frecuencia de aparición de cada término.

Antes de proseguir, se deben eliminar aquellas palabras que contribuyen a la sintaxis de las oraciones, pero sin significado propio, también llamadas *stop words* o palabras vacías, tales como artículos, preposiciones, conjunciones y se limpia el texto removiendo número, puntación, y en general cualquier tipo de carácter aislado. También se cambian las vocales con tilde por su igual sin tilde, de modo que errores ortográficos al escribir una palabra no produzcan varios términos de la misma palabra, por la misma razón se unifica todo el texto a minúsculas. No tan frecuente es el uso de la técnica *stemming*, para descomponer los términos y trabajar únicamente con la raíz (lexema), dado que hay estudios que muestran reducción en la efectividad del clasificador.

Reducción de la dimensión.

Un factor a tener en cuenta en procesamiento de texto es la dimensión de los términos, es decir el vocabulario o léxico(*lexicon*), que pueden llegar a miles y hasta cientos de miles si se tiene en cuenta que el diccionario español tiene alrededor de 100 mil palabras y se estima que el lenguaje español tiene cerca de 300 mil palabras, esta dimensión conlleva a problemas computacionales tales que el espacio de memoria para almacenar tal número de

variables o la cantidad de cómputo para procesarlas exceden la capacidad de las computadoras personales.

De otro lado una alta dimensión en los términos con los que se construye el clasificador tiende a producir sobreajuste que es el fenómeno por el cual el algoritmo de aprendizaje queda demasiado adaptado a las características del grupo de entrenamiento con lo cual no funciona tan bien tratando de predecir las clases del grupo de comprobación.

Hay dos caminos para reducción de la dimensión o *Dimensionality Reduction* DR de los términos, mediante la selección términos por el cual el resultado es un subconjunto de los términos originales y mediante extracción de términos en donde los términos resultantes no son los mismos términos del conjunto original.

La técnica de extracción de términos *Latent Semantic Indexing* o LSI aborda los problemas derivados por el uso de sinónimos y palabras polisémicas[7], esta técnica reduce la dimensión del vector original combinando los términos de acuerdo a patrones de ocurrencia, de manera que la representación final estará compuesta por nuevos términos compuestos. Una aproximación semejante es la de lematización[8] en donde las palabras se reducen por su raíz en común.

En tanto que en el enfoque de selección de términos la técnica más usada es la *Document Frequency* en la que sólo se utilizan aquellos términos con mayor frecuencia de aparición. En este caso Yan y Pedersen en un serie de experimentos (YANG, 1997) mostraron que es posible reducir la cantidad de términos por un factor de 100 sin pérdida de efectividad

[7] Se dice de una palabra con varias acepciones.
[8] Técnica de supresión de sufijos

manteniendo sólo aquellos términos con mayor frecuencia de aparición y desechando aquellos que por contraste aparecen menos veces en el corpus.

Evaluación y selección del modelo

Lo que se propone en este título es caracterizar algunos de los experimentos que son realizados para evaluar un algoritmo de aprendizaje en uno más conjuntos de datos de modo que se obtengan las medidas que se requieren para dar las respuestas adecuadas al objetivo de aprendizaje. Entonces, en primero lugar hay que preguntarse ¿Qué medir?, Para decidir que medir primero se requiere conocer el objetivo experimental y el contexto para el que se desarrolla. En ML la respuesta usualmente es el porcentaje de los datos que fueron clasificados correctamente denominado exactitud (en inglés accuracy), no obstante, pese a ser la medida de valuación más común no es la única, en la figura 6 se ilustra algunas de las medidas utilizadas para evaluar los modelos de ML.

		Predicción	
		Positivo	Negativo
Valor Real	Positivos	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos(VN)

Figura 6. Matriz de Confusión, reproducido de (Fawcett, 2006)

En la anterior imagen, la matriz de confusión, se puede observar el desempeño del modelo en el caso de clasificación entre positivos y negativos, cada columna de la matriz representa el número de predicciones para cada clase obtenidas con el modelo, mientras que al sumar las observaciones de cada fila representa el valor real en esa categoría. En este caso binario las medidas son Verdaderos Positivos (VP), Verdaderos Negativos (VN), Falsos Negativos (FN), Falsos Positivos (FP). La exactitud es una medida compuesta y se expresa en la fórmula (de nuevo para el caso binario):

$$exactitud = \frac{\mathrm{VP} + \mathrm{VN}}{Total}$$

Ahora bien, la exactitud como medida de evaluación del modelo se basa en la presunción que la distribución en el conjunto de comprobación es similar a la distribución de la población de observaciones. Lo anterior es debido a un factor que está presente en los problemas de clasificación con ML, la Distribución de clases. Con un ejemplo es más fácil ilustrar el efecto de la distribución de clases en la exactitud del modelo, suponga una clasificación ente positivos y negativos cuando el número de negativos es mucho más grande que las observaciones positivas, si el modelo tiene sesgo por los negativos (tendencia a clasificar las observaciones como negativas) entonces tendrá una buena exactitud, solo por el hecho de que hay más observaciones negativas. Razón por la cual en el escenario anterior no sería la exactitud la mejor manera de evaluar el modelo, siendo quizá la tasa de verdaderos positivos VP (en inglés *recall*) le medida adecuada para este ejemplo en particular.

En resumen, la matriz de confusión contiene las mediciones, de manera que lo restante es evaluar la medición o mediciones escogidas para seleccionar el modelo. Hay varios

métodos de medición, algunos analíticos de estimación de error como AIC(Akaike information criterion) y el BIC(Bayesian information criterion) detallados ampliamente en el libro de Elementos estadístico de Machine Le (Hastie, Tibshirani, & Friedman, 2009) , este documento se enfoca en el método de validación cruzada de K iteraciones o *k-fold cross-validation*, esta es una técnica que consiste en dividir la muestra en k subconjuntos, convencionalmente k=10 (Bouckaer & Frank, 2004), y cada subconjunto se divide en dos partes, una parte de los datos para entrenar el modelo, usualmente con el 80% de las observaciones, una segunda parte para evaluar el mejor modelo(validación) con el 20% de los ejemplos. En escenarios donde hay suficientes observaciones, una vez se ha terminado el proceso de selección, se utiliza otra muestra para evaluar le generalización del modelo en observaciones que no jugaron ningún papel en la selección del modelo, esta muestra comúnmente es denominada conjunto de datos de comprobación o de pruebas (*test set*). Utilizar observaciones diferentes para entrenar y validar el modelo es condición crítica para darle carácter científico al experimento (Mitchell, 1997), (Hastie, Tibshirani, & Friedman, 2009).

Ahora bien, con el método de validación cruzada de K iteraciones se puede evaluar cada iteración, pero si el experimento tiene un k=10, por ejemplo, se obtendrán 10 subconjuntos con distribuciones diferentes, por tanto, es de esperar que la exactitud en cada iteración sea diferente y entonces se tendrá un rango de mediciones. Al igual que los métodos de evaluación, aquí nuevamente hay muchos métodos estadísticos para lidiar con esta incertidumbre, así que para comparar algoritmos de aprendizaje en múltiples conjuntos de datos hay una prueba específicamente diseñada para hacerlo, la prueba según Friedman

cuya idea es hacer un ranking con la *exactitud* de todos los algoritmos por cada subconjunto de datos, asignándoles un rango desde el mejor desempeño hasta el de peor desempeño. En el siguiente cuadro de resultado del *Friedman test* se compara la exactitud para un dado conjunto de datos de los algoritmos; Ingenuo Bayes NB, Árbol de Decisión AD y Vecino más Cercano VC[9] donde las posiciones en el ranking de exactitud son colocados en paréntesis y finalmente se calcula el promedio del desempeño que obtuvo cada algoritmo para las 5 iteraciones, la interpretación del resultado es que el algoritmo AD obtuvo mejor promedio del ranking en las pruebas por tanto es el mejor candidato.

Iteración	NB	AD	VC
1	0.6809 (3)	0.7524 (1)	0.7164 (2)
2	0.7017 (3)	0.8964 (1)	0.8883 (2)
3	0.7012 (2)	0.6803 (3)	0.8410 (1)
4	0.6913 (2)	0.9102 (1)	0.6825 (3)
5	0.6333 (3)	0.7758 (1)	0.7599 (2)
Promedio	2.6	**1.4**	2

Tabla 2: Experimento según Friedman, reproducción del libro Machine Learning (Flach, 2012)

[9] Traducción libre de los algoritmos; Naive Bayes, Decision tree y Nearest neighbour

En evaluación de algoritmos de clasificación de texto, estudios comparativos de varios métodos de clasificación para los cuales se ha medido de la exactitud sirven como marco guía para selección del algoritmo, no obstante, la validez de estos experimentos es interna. Finalmente, el diseño de experimentos en ambientes Big Data agrega otro nivel de dificultad derivados de lidiar con la heterogeneidad, variaciones en los experimentos, sesgos estadísticos, entre otros, que tienen su origen en la combinación de diferentes fuentes de información y de tecnologías subyacentes. De manera que, la discusión sobre el diseño de experimentos en escenarios Big Data está abierta sin que aún haya consensos entre investigadores sobre cuáles son los mejores métodos, en Principios de diseño experimental para Big Data (Drovandi, 2017) el autor propone un algoritmo de experimentos secuenciales repartidos entre varios CPU utilizando el framework Hadoop, es una aproximación experimental antes que analítica, en donde se hacen experimentos iterativos a fin de en cada iteración refinar la exactitud o la medida con que se evalúa el algoritmo, tiene similitudes con el método de validación cruzada de K iteraciones, pero agrega mayor complejidad debido la variedad subyacente propia del Big Data.

Machine Learning en Big Data

Big Data le ha permitido a machine Learning (ML) encontrar patrones y hacer predicciones de forma distribuida, es así como en el contexto de este trabajo utilizar computación distribuida le permite al modelo tener la propiedad de escalabilidad, la cual le confiere al modelo la habilidad de ser ejecutado en múltiples máquinas trabajando en paralelo para adaptar el rendimiento al crecimiento de datos sin perder la efectividad en clasificación de

documentos, y de paso, aumenta la disponibilidad. Factores que como se dijo en el planteamiento del problema son críticos para el éxito del proceso de Administración de Correspondencia. A continuación, se describen MapReduce y Spark, componentes utilizados durante el diseño del modelo.

El análisis de grandes volúmenes de datos o Big Data[10], demanda partir los problemas en piezas independientes para ser trabajadas en paralelo, y la solución para esto fue una contribución de Jeffrey Dean y Sanjay Ghemawat quienes trabajando para Google, Inc. Desarrollaron el MapReduce MR, como un modelo y una implementación para procesar grandes conjuntos de datos (Dean & Ghemawat, 2004).

MapReduce es un framework de software que puede ejecutar programas escritos en varios lenguajes como Java, Python, Ruby, Scala. Diferentes implementaciones de MR han sido desarrolladas desde su aparición, en la actualidad las implementaciones más populares son MapReduce2 y YARN, ambas publicadas bajo licencia de código abierto. Funciona al dividir los trabajos que se le encomiendan en dos tipos de tareas: las tareas de mapeo, *map task* y las tareas de reducción, *reduce task*. Estas tareas de mapeo son ejecutadas en el nodo principal del clúster denominado ResourceManager el cual mapea el trabajo en varias tareas llave, valor y se las encarga para su ejecución a los nodos esclavos denominados NodeManager. Cuando los NodeManager terminan la tarea que se les encomendó, envían

[10] Comunmente definido por sus características, las 5Vs; Volumen, Variedad y Velocidad (Zikopoulos, Eaton, deRoos, Deutsch, & Lapis, 2011), a las cuales posteriormente se sumaron 2Vs mas, la Veracidad y el Valor (Demchenko, Grosso, de Laat, & Membrey, 2013).

el resultado de regreso en un nuevo par llave, valor, al ResourceManager quien ahora reduce las entradas y entrega el resultado del trabajo (White, 2009).

Por otro lado, en materia de algoritmos de Machine Learning hay dos grandes implementaciones de algoritmos de ML escalables, el Apache Mahout y Apache Spark ambos pueden funcionar con YARN para manejo de flujos de trabajo. Siendo su principal diferencia la manera como se manejen los conjuntos de datos sobre los que se trabaja, en el caso de Mahout estos son mantenidos en disco en tanto que en el caso de Spark se privilegia la memoria, lo que le confiere a Spark mayor rapidez.

El corazón de Spark su módulo de ML es MLlib, tiene implementaciones para una gran cantidad de algoritmos de ML, ver algunos en la tabla 3, a los cuales se puede acceder con los lenguajes Scala, Java, y Python.

Algoritmo	Implementación en SPARK
Redes Neuronales	ml_multilayer_perceptron_classifier ml_multilayer_perceptron
Regresión	ml_linear_regression
Naive Bayes	ml_naive_bayes
Máquina de Soporte Vectorial	ml_linear_svc
Gradient Boosting Tree	ml_gbt_classifier ml_gradient_boosted_trees ml_gbt_regressor

Tabla 3: Algoritmos de ML en SPARK

Aquellos analistas y estadísticos que usan la herramienta R (R Core, 2014), para aprovechar la escala de procesamiento del clúster en Hadoop, no necesitan adquirir conocimiento de programación adicional; hay muchos paquetes y bibliotecas que se han desarrollando para integrarse con R, entre las cuales se han destacado RHadoop, Rhipe y Sparklyr. El paquete sparklyr utilizado en el caso de estudio, proporciona una interfaz entre R y los algoritmos de aprendizaje automático de Spark, el cual admite la conexión a clústeres Apache Spark tanto locales como remotos. A continuación, se describe como estos componentes trabajan en conjunto para entregar servicios de ML.

Estado del Arte

Revisando la literatura se encuentran aplicaciones con Apache Spark para análisis predictivos y búsqueda de patrones, que han llevado ML a solucionar problemas complejos que requieren más agilidad como análisis de trasmisiones de videos y voz, problemas que hace apenas un par de años no eran posibles de solucionar en ambientes standalone por la capacidad de computo requerida, ni en despliegues distribuidos, especialmente por la lentitud en los accesos a discos.

Los autores (Naika & Purohitb, 2017) , en su artículo, han hecho un intento por comparar la clasificación binaria mediante algoritmos de ML en Apache Spark con grandes volúmenes de datos, en el cual se utiliza procesamiento distribuido con Hadoop desplegado en máquinas virtuales en la nube Amazon Web Services. El propósito de este estudio es comparar la eficiencia de tres variantes del algoritmo árbol de decisión; Decision Tree,

Random Forest Tree y Gradient Boosted Tree. El estudio demuestra que el Random Forest es el más eficiente árbol ya que tomó el menor tiempo en construir el árbol de clasificación. Pero Spark no se ha limitado a los algoritmos de ML, Spark y MapReduce también ha sido usado en problema de reducción de dimensionalidad, permitiendo la ejecución paralela de estos algoritmos (Peralta , S, S, & I, 2015).

Por otra parte específicamente en analítica de texto en el libro Mining Text Data (Aggarwal & Zhai, 2012) se hace una exhaustiva revisión de los principales elementos, técnicas y herramientas en analítica de texto, sin embargo este trabajo, pese a ser reciente está desactualizado en el sentido de no incluir los novísimos adelantos en materia de manejo de grandes volúmenes de datos. Siendo el análisis de sentimiento u opinión, motivado por el creciente uso de las redes sociales uno de los tópicos que más han generado investigación en los últimos dos años de ML sobre Big Data, en el artículo (Ortigosa, Martín, & Carro, 2014) los autores clasifican textos recogidos de twitter por su sentimiento negativo o positivo respecto a un producto, también se ha realzado la detección de ironía en twitter sobre temas de política en Grecia (Charalampakis, Spathis, Kouslis, & Kermanidis, 2016). Finalmente, en el campo de la medicina la clasificación de texto también ha sido de intereses como parte del proceso de selección de artículos de investigaciones relevantes al dominio de la medicina, encontrando el resultado sorprendente de un 84% de exactitud comparado con los expertos médicos que evalúan que realizan están revisiones (García Adeva, Pikatza Atxa, Ubeda Carrillo, & Ansuategi Zengotitabengoa, 2014).

III. Propuesta metodológica

Este capítulo explica la metodología utilizada para la experimentación de Machine Learning en clasificación de correspondencia electrónica. La principal motivación es contar con un diseño experimental óptimo de manera que permita establecer el modelo de clasificación que tenga mayor exactitud, que pueda operar en diferentes contextos con diferentes distribuciones de clases. Ahora bien, el número de potenciales diseños es ilimitado debido a que existen un gran número de configuraciones del algoritmo y características como la calidad y cantidad de observaciones que hacen parte del conjunto de entrenamiento (training set) tiene un alto impacto en la exactitud final del modelo.

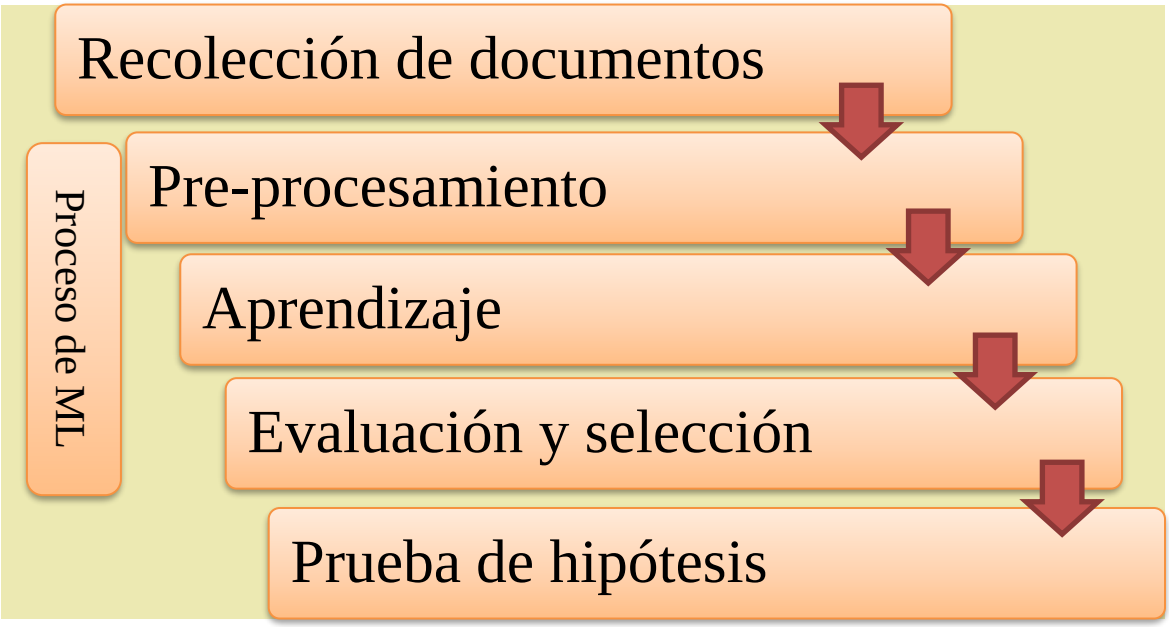


Figura 7. Etapas de la metodología

Esta metodología está estructurada en 5 etapas, ver figura 7, cada una se compone de un conjunto de métodos organizados lógicamente por su propósito. El experimento es ajustado específicamente al conjunto de datos utilizados, por lo que su validez es interna.

Recolección de documentos

La fuente de información es una combinación entre documentos almacenados en el sistema de archivos y el registro histórico de correspondencias electrónicas en el Sistema de Información de gestión documental ORFEO, ver figura 8. Cada correspondencia electrónica es un documento que es almacenado como archivo HTML, por lo que debe ser transformado retirando las etiquetas HTML y dejando únicamente el contenido textual.

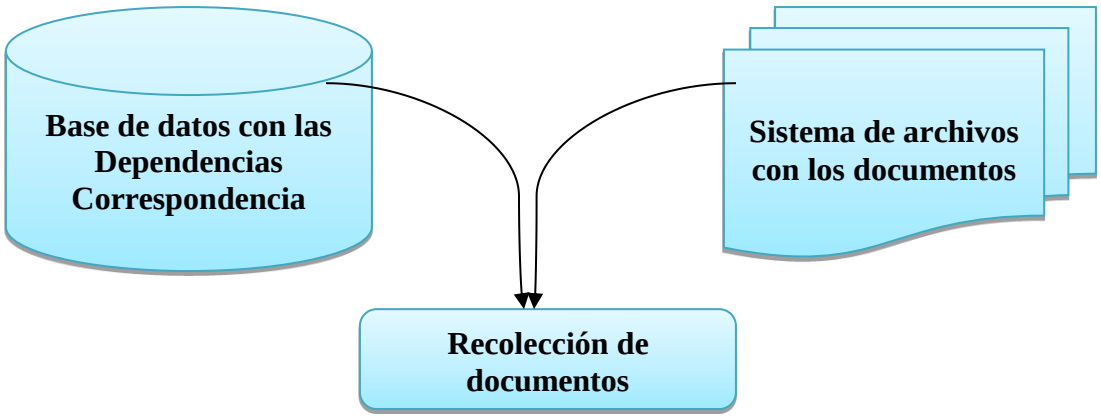


Figura 8. Etapa de recolección de documentos

El corpus[11] N son los documentos almacenados sistema de archivos. En tanto que la Base de Datos del S.I contiene los registros de la Dependencia en la que fueron clasificados, en total 17 Clases, C.

$$N = 572.000$$

$$C = 17$$

De esta población de documentos se toma una muestra n de 6000 documentos clasificados en C_i=9 Dependencias dado que estas han tenido continuidad desde que se inició el registro de correspondencia en el S.I.

$$n = \{X_1, X_2, X_3 \dots X_{6000}\}$$

$$C_i = 9$$

La correspondencia cuyo contenido esta en archivos anexos como, PDF, Word, imágenes u otro está fuera de los límites de esta investigación, por lo que estos archivos deben ser retirados inspeccionando en su contenido y eliminando aquellos que no tienen texto.

Etapas del Proceso de Machine Learning

Como se dijo en el capítulo de marco teórico los algoritmos de ML requieren que el texto sea representado en una forma que pueda ser computable, hacerlo corresponde a la etapa de pre-procesamiento, en donde se realizan técnicas para limpiar los datos, de transformación y se evita el sobreajuste mediante le reducción de la dimensión. El resultado es una base de datos utilizada como insumo por la siguiente etapa.

[11] En el contexto de esta investigación a la población de estudio se le denomina corpus

Para la etapa de Aprendizaje, se utiliza la herramienta estadística R y los algoritmos ML implementados en Spark; Máquina de soporte Vectorial, Gradient Boosting Tree y como referencia también se incluye en los experimentos el método estadístico Naïve Bayes. La interfaz R con Spark se realiza mediante la herramienta sparklyr la cual permite utilizar los algoritmos Spark con la convención de lenguaje de R, en tanto que la distribución de las tareas es ejecutada por MR desplegado en clúster.

La etapa de Evaluación y Selección del modelo es secuencial adaptado de los métodos de validación cruzada de K iteraciones y el diseño experimental para Big Data propuesto por Drovandi (Drovandi, 2017) . En donde se realizan $K = 5$ experimentos de aprendizaje con los tres algoritmos, de cada experimento toman parte un subconjunto de documentos $n_i = 1000$. En cada iteración se registra la distribución de clases $P(x)$ del subconjunto que toma parte del experimento y la matriz de confusión con el resultado de la clasificación de las C_i Dependencias.

$$exactitud = \frac{\sum VC_i}{Total}$$

Siendo X el número de documentos que toman parte de cada experimento i, cada subconjunto formalmente puede expresarse:

$$n_i = \{X_1, X_2, X_3 \dots X_{1000}\}$$

Cada subconjunto n_i es dividido aleatoriamente en conjunto de datos de entrenamiento con el 80% de los documentos y un conjunto de datos de validación con el 20%. El conjunto de datos de entrenamiento es utilizado para entrenar el modelo, mientras que el conjunto de validación es utilizado para medir la exactitud de cada modelo seleccionado.

Conjunto de datos de entrenamiento con 80% de n_i^t

Conjunto de datos de validación con 20% de n_i^v

Se registra la distribución de frecuencia de clases para los K experimentos, y al final mediante la prueba utilizada por Friedman se estima el mejor modelo posible.

Etapa de prueba de hipótesis

Finalmente, el modelo seleccionado es ejecutado con el conjunto de datos de comprobación, una muestra de datos que no tomó parte para entrenar ni para validar los modelos en ninguna de las K iteraciones, y con dicho conjunto se prueba la hipótesis H_A, utilizando la matriz de confusion para medir la exactitud y nuevamente se registra la distribución de frecuencias.

Conjunto de datos de comprobación $n_h = \{X_1, X_2, X_3 \dots X_{1000}\}$

Herramientas y Arquitectura de ejecución.

Cada etapa de la metodología precisa herramientas específicas para su cometido; los archivos que hacen parte de la primera etapa se seleccionan aleatoriamente con la herramienta shuf del S.O Linux y trasferidos mediante la herramienta de FTP Seguro WinSCP hacia el equipo con la herramienta de análisis. La base de datos del S.I ORFEO es Oracle Data Base, la cual contiene las Dependencias en que fue clasificada la correspondencia, para la extracción de estas categorías se utilizó el cliente SQL par Oracle denominado TOAD.

La etapa de pre-procesamiento se realiza con RStudio 1.1 (RStudio Team, 2016) con las

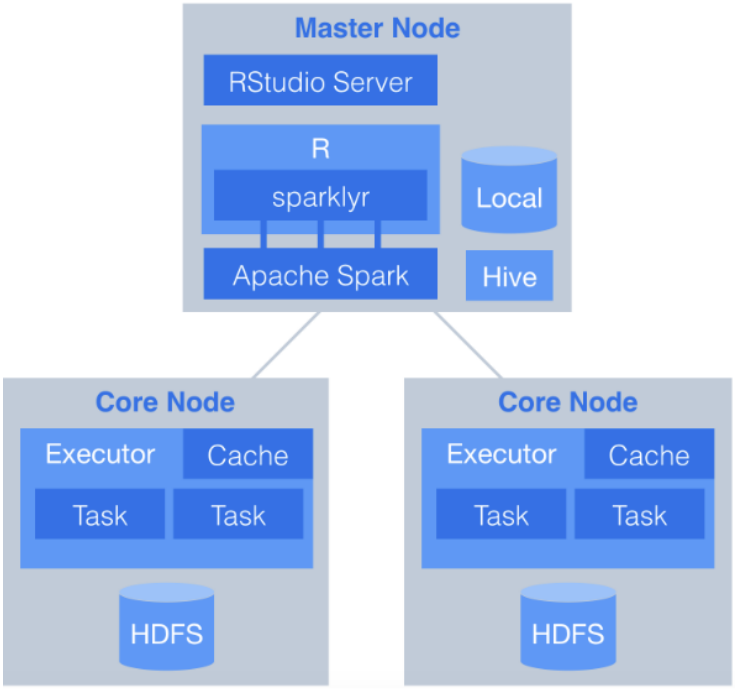


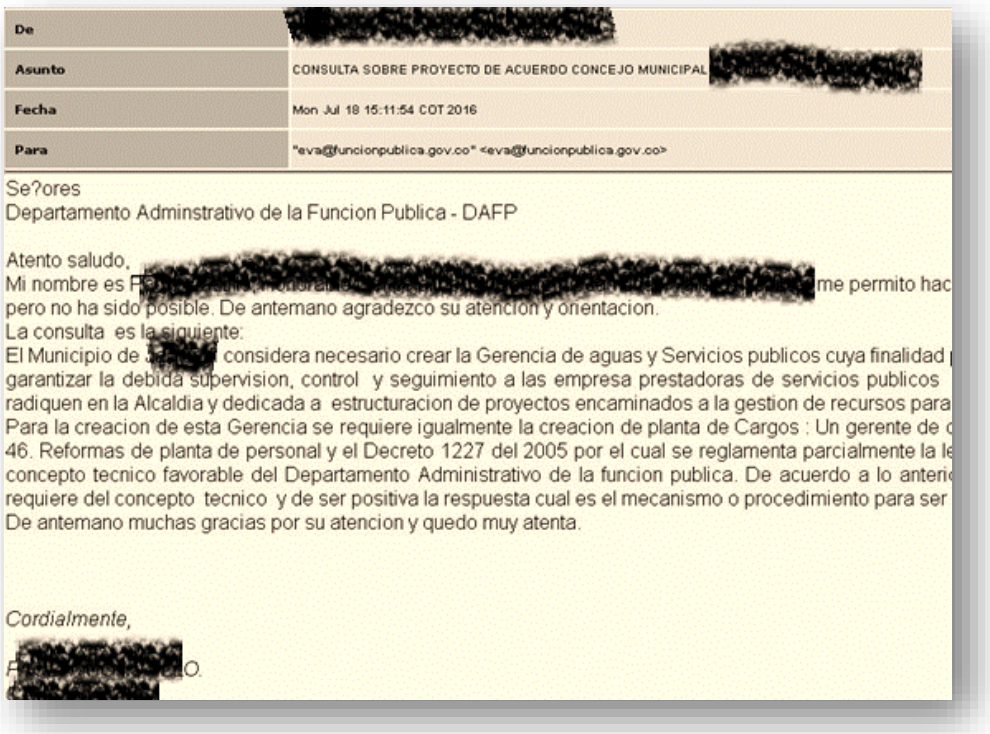
Figura 9: Arquitectura de despliegue en clúster con sparklyr, reproducido de http://spark.rstudio.com

librerías especializadas para el manejo de texto; tm, wordcloud, ggplot2, dplyr, readr, cluster RCurl, XML descargadas desde la red de librerías para R CRAN. En tanto que las etapas correspondientes al Proceso de ML y prueba de hipótesis, son realizadas con la herramienta RStudio Server 1.0, integrado con Spark y el clúster Big Data mediante el paquete sparklyr 2.1. La Arquitectura del clúster corresponde al modo clúster con YARN, en la figura 9, se muestra cómo se despliega sparklyr en un clúster Apache Spark. En esta arquitectura los datos se descargan de la web y se almacenan en HDFS en los nodos del cluster, los cuales ejecutan las tareas. RStudio Server está instalado en el nodo maestro y orquesta el análisis con Spark.

IV. Caso de Estudio análisis y predicción de categorías en correspondencia electrónica

Recolección de documentos

Los documentos que conforman el Corpus, en total 572.220, están almacenados en un sistema de archivos, organizados en directorios nombrados por el año en que ingresaron al sistema, entre el 2009 al 2017. Cada correo electrónico es almacenado como un archivo con formato HTML, en la figura 10, se puede ver una correspondencia recibida por la entidad del caso.



De	[redacted]
Asunto	CONSULTA SOBRE PROYECTO DE ACUERDO CONCEJO MUNICIPAL [redacted]
Fecha	Mon Jul 18 15:11:54 COT 2016
Para	"eva@funcionpublica.gov.co" <eva@funcionpublica.gov.co>

Se?ores
Departamento Adminstrativo de la Funcion Publica - DAFP

Atento saludo,
Mi nombre es [redacted] me permito hac
pero no ha sido posible. De antemano agradezco su atencion y orientacion.
La consulta es la siguiente:
El Municipio de [redacted] considera necesario crear la Gerencia de aguas y Servicios publicos cuya finalidad
garantizar la debida supervision, control y seguimiento a las empresa prestadoras de servicios publicos
radiquen en la Alcaldia y dedicada a estructuracion de proyectos encaminados a la gestion de recursos para
Para la creacion de esta Gerencia se requiere igualmente la creacion de planta de Cargos : Un gerente de
46. Reformas de planta de personal y el Decreto 1227 del 2005 por el cual se reglamenta parcialmente la le
concepto tecnico favorable del Departamento Administrativo de la funcion publica. De acuerdo a lo anteri
requiere del concepto tecnico y de ser positiva la respuesta cual es el mecanismo o procedimiento para ser
De antemano muchas gracias por su atencion y quedo muy atenta.

Cordialmente,
[redacted]

Figura 10. Documento de correspondencia electrónica

En primer lugar, los documentos fueron cargados aleatoriamente como un vector en RStudio desde el directorio que contiene los archivos .HTML y luego los archivos se transformaron de HTML a texto. Dada la variedad de los clientes de correo electrónico,

Conjunto de caracteres	Resultado
windows-1252	Cordial saludo Mi nombre es HENRY JESUS SOLARTE FAJARDO identificado con Cdula de Ci sta en el ao 2012 desde el mes de agosto hasta el mes de diciembre del mismo ao con el para solicitar una constancia que necesito para mi hoja de vida. Adjunto copia del con
ASCII//TRANSLIT	Cordial saludo Mi nombre es HENRY JESUS SOLARTE FAJARDO identificado con Cedula de Ciudadania sta en el ano 2012 desde el mes de agosto hasta el mes de diciembre del mismo ano con el contr n para solicitar una constancia que necesito para mi hoja de vida. Adjunto copia del contrato.
iso_8859_15	Cordial saludo Mi nombre es HENRY JESUS SOLARTE FAJARDO identificado con CÃ©dula de Ciudada ista en el aÃ±o 2012 desde el mes de agosto hasta el mes de diciembre del mismo aÃ±o con el nten para solicitar una constancia que necesito para mi hoja de vida. Adjunto copia del cont
UTF-8	Cordial saludo Mi nombre es HENRY JESUS SOLARTE FAJARDO identificado con Cédula de Ciudadanía N sta en el año 2012 desde el mes de agosto hasta el mes de diciembre del mismo año con el contrat n para solicitar una constancia que necesito para mi hoja de vida. Adjunto copia del contrato. P

Tabla 4: Efecto del conjunto de caracteres en la lectura del texto

el conjunto de caracteres con que se escriben no es uniforme por lo que al leerlo debe asignársele el conjunto de caracteres dinámicamente probado para cada archivo cual codificación es la que más se ajusta al texto conforme se codifique correctamente las tildes y las eñes. El efecto de las diferentes lecturas para un mismo archivo con diferentes conjuntos de caracteres, se muestra en la tabla 4.

Una vez fue seleccionada la codificación de forma automática, se creó un vector con el listado de documentos, ahora denominado Corpus y se consultó dentro de la base datos del S.I ORFEO a que Dependencia(clase) fue asignada cada correspondencia, un extracto de la consulta puede consultarse en el Anexo G. se tuvo en cuenta que el S.I no revisa la integridad del Sistema de archivos con la con la Base de datos, por esto los registros no

cruzan en todos los casos, por lo que aquellos documentos sin clase tuvieron que ser retirados de la muestra.

El resultado son 7206 documentos de los cuales se utilizaron 6000, clasificados en 9 Dependencias, la tabla 5 contiene las características de la muestra utilizada.

Población	**572.220** archivos .HTML recibidos y almacenados entre el 2009 al 2017. Clasificados en 17 dependencias de la entidad del gobierno de Colombia: Función Pública
Base de la muestra	Repositorio documental de correspondencia Base de datos de correspondencia
Tamaño y distribución de la muestra	6000 archivos distribuidos en las siguientes 9 clases: Dirección jurídica, Grupo de servicio al ciudadano institucional Grupo de gestión meritocracia Dirección de participación, transparencia y servicio al ciudadano Oficina de tecnologías de la información y las comunicaciones Grupo de gestión financiera Despacho del director Grupo de gestión contractual Grupo de gestión administrativa

Tabla 5: Ficha de la muestra

Con la construcción del Corpus de documentos se concluye esta etapa, el script en lenguaje R puede ser consultado en el Anexo C.

Pre-procesamiento de datos

Los textos que conforman el corpus de documentos deben ser limpiados de caracteres que son de poca utilidad en la minería de textos para entender el sentido de este, tales como palabras vacías, puntuación, números y fechas, y deben ser formateados conforme se requiere para la siguiente fase, por lo que se utilizaron las técnicas de minería de texto que se esquematizan en la figura 11.

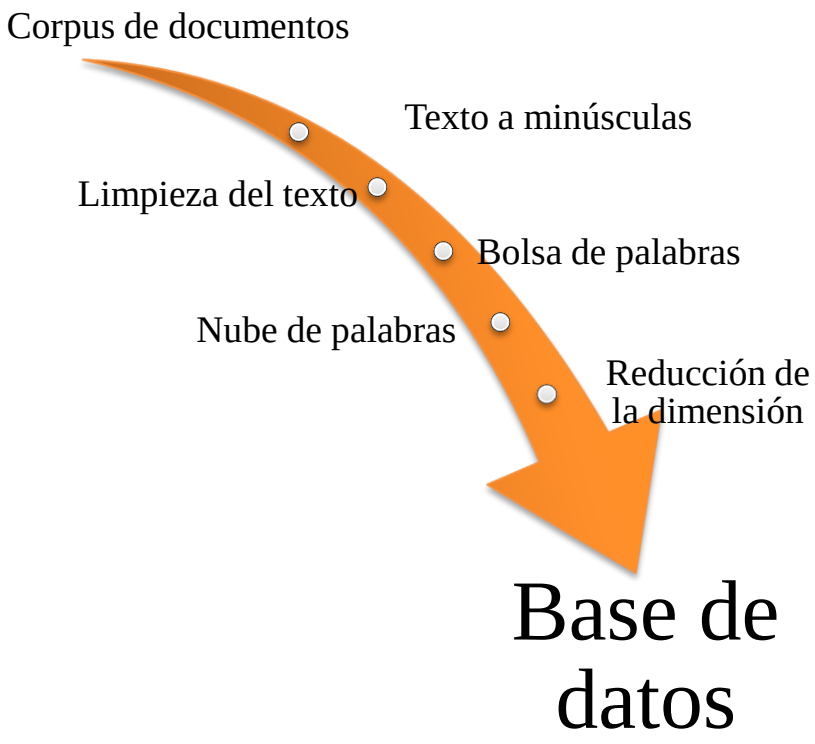


Figura 11: Técnicas de minería de texto

Así las cosas, los contenidos de documentos que hacen parte del corpus son convertidos a minúsculas para que una palabra con contenido en mayúscula y su igual en minúscula sean un único término. Luego se removieron las *palabras vacías* y se realizó la limpieza de tildes, caracteres, números, espacios y símbolos, a continuación, se ejemplifica el estado inicial y final, luego de la aplicación de las técnicas:

Texto Original	"\t\t\t\t\t\t\t *20142060000812* Radicado No. 20142060000812 Fecha : 2014/01/03 \t\t\t\t\t\t De (Contenido Omitido)< (Contenido Omitido)@ (Contenido Omitido) Buenos días, por favor solicito las claves para ingresar la información de la vigencia 2013 sobre la evaluación de control interno y las fechas de ingreso de la informacion. Remitir lo solicitado a este correo y al correo … (Contenido Omitido)"
Texto después de la limpieza	" radicado fecha (Contenido Omitido) asunto solciitud fecha buenos dias favor solicito claves ingresar informacion vigencia evaluacion control interno fechas ingreso informacion remitir solicitado correo correo …(Contenido Omitido)"

Luego de aplicar estas técnicas el corpus es transformado en una matriz de números enteros conformando una bolsa de palabras clasificadas en C clases, en donde las columnas son los términos t y las filas los documentos X, la última columna es la clase a que pertenece el documento:

Bolsa de palabras:

	t_0	t_i	C
X_1	2	1	1
X_i	0	0	9

Los identificadores en la columna de clases C son la variable categórica correspondiente a cada una de las 9 dependencias que hacen parte de la muestra como se puede ver en la siguiente tabla:

$\#C_i$	**Dependencia**
1	Dirección jurídica
2	Grupo de servicio al ciudadano institucional
7	Grupo de gestión meritocracia
8	Dirección de participación, transparencia y servicio al ciudadano
9	Oficina de tecnologías de la información y las comunicaciones
10	Grupo de gestión financiera
11	Despacho del director
12	Grupo de gestión contractual
13	Grupo de gestión administrativa

Tabla 6: Tabla de correspondencia identificador de Clase / Dependencia

Una vez se ha mapeado el corpus, se construyó una nube de palabras *wordcloud* disponible en la librearía del mismo nombre, la cual permite visualmente darse cuenta de los términos más frecuentes en el corpus de documentos, ver figura 13.

Figura 12. Nube de palabras

Se pudo inspeccionar visualmente mediante la nube de palabras, que aún quedan palabras con mucha frecuencia pero que no sirven para entender el sentido del texto como correos electrónicos webmasterdafpgovco (al cual se le quitó el @) o algunas palabras vacías en inglés, por lo que estas también fueron retiradas.

En la nube de palabras en la figura 3 se observa que se removieron algunos de los términos con problemas, detectados en su anterior versión.

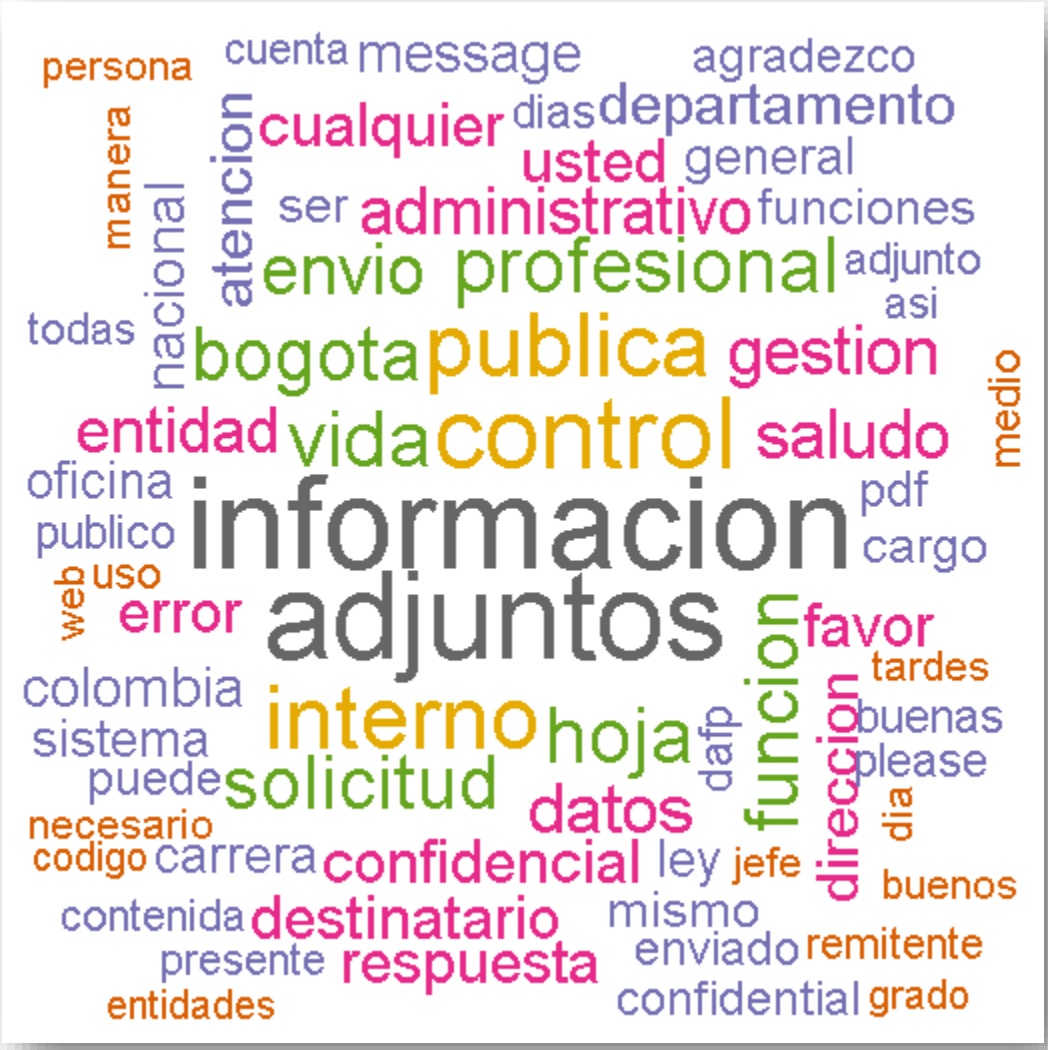


Figura 13. Nube de palabras para análisis

En la figura 14 se presentan las veinte palabras más frecuentes en la correspondencia recibida por la entidad del caso, y con esto se puede dar por terminadas las actividades de limpieza de texto. Recapitulado, se ha unificado la ortografía eliminando tildes y sustituyendo mayúsculas, se han eliminado números y caracteres aislados, palabras vacías

en inglés y español, así como otros términos que, aunque tienen alta frecuencia de aparición, no sirven para discriminar la temática del correo.

Archivo	acuerdo	administrativo	Anterior	así	atenta	buenas	cargo	...	clases
1	1	1	1	1	1	1	2	...	11
2	0	0	0	0	0	0	0	...	7
3	0	0	0	0	0	0	0	...	12
4	0	0	0	0	0	0	0	...	12
5	0	0	1	0	0	1	0	...	12
6	0	0	0	0	0	0	0	...	12
7	2	0	1	0	0	0	0	...	11
8	1	1	0	0	0	0	0	...	11
9	0	1	0	0	0	0	0	...	12
10	0	0	0	0	0	0	0	...	12
11	0	0	0	0	1	0	0	...	12
12	0	0	0	0	0	0	0	...	12
13	0	0	1	0	0	0	2	...	12
14	0	0	0	0	0	0	0	...	12
15	0	0	1	0	1	0	6	...	11
16	0	0	0	0	0	0	0	...	12
17	0	0	0	0	0	0	0	...	12
18	0	0	0	0	0	0	0	...	12
19	0	0	0	0	0	0	0	...	1
20	0	0	0	1	0	0	0	...	12
..	..	..	..	..	..	..	..	...	..
7206	0	0	0	0	0	0	0	...	12

Tabla 7: Base de datos resultado del pre-procesamiento

Sólo resta preparar el insumo de la siguiente etapa y reducir la dimensión para evitar el sobreajuste mediante el método de *Document Frequency* con una reducción de los términos en un factor de 100, que conforme se ha demostrado no afecta la efectividad de la clasificación (YANG, 1997) formalmente:

$$\#términos = \frac{\sum t_i}{100} = \frac{54.488}{100}$$

Como resultado de 54.488 términos en la matriz original se pasa a 573 términos, correspondientes a aproximadamente dividir la dimensión original en 100.

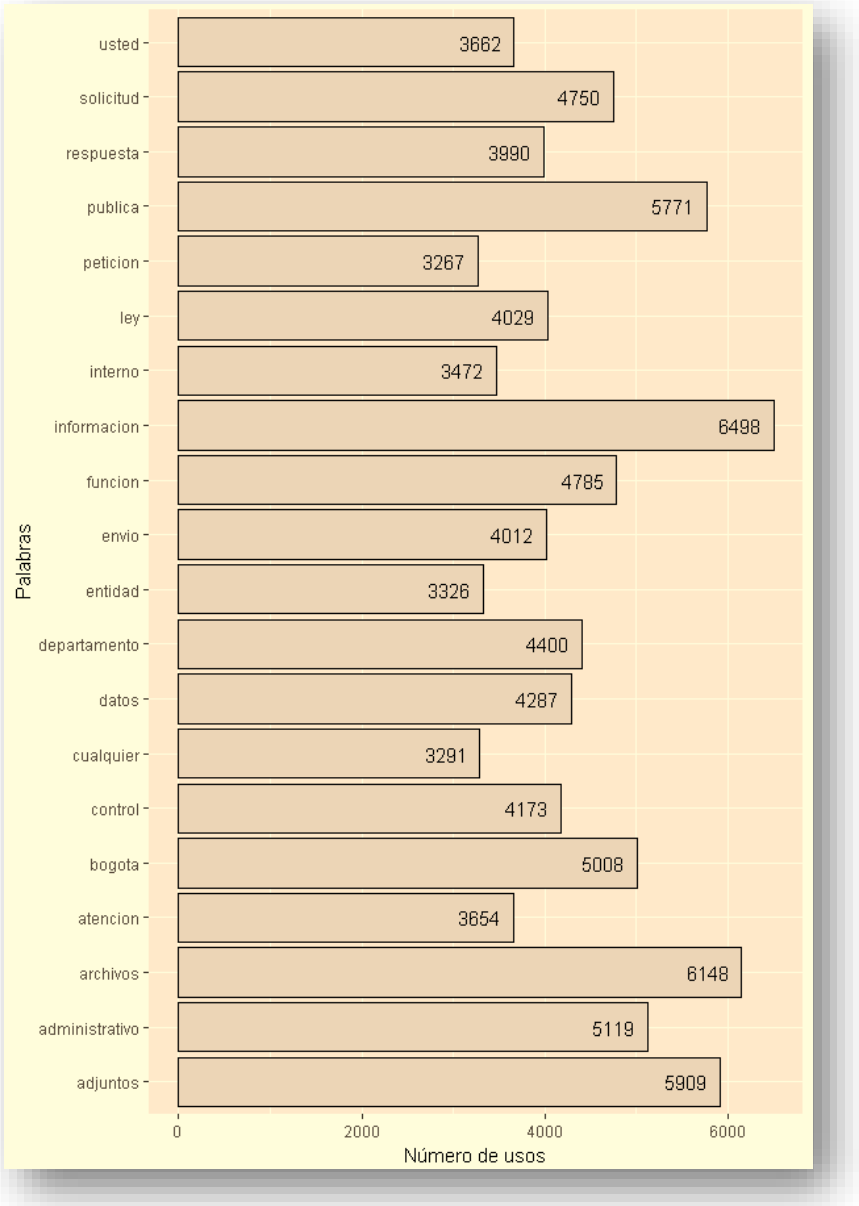


Figura 14: Palabras usadas más frecuentemente

La base de datos obtenida, ver tabla 7 y tabla 8, se utilizó como insumo para crear los modelos de pronóstico en la siguiente etapa, consta de una tabla compuesta por 7206 documentos y 573 términos y una columna más con la clase real.

Clases		
Campo	*Tipo*	*Descripción*
Archivo	Número	Identificador por orden del documento
Palabras, columna 1 a 573	Número	Cantidad de Ocurrencia de la palabra en el archivo
Clase, columna 574	Número	Numero con Id de la clase a que pertenece.

Tabla 8: Significado de los campos en la base de datos de documentos

Aprendizaje

Las operaciones de acá en adelante se ejecutaron en R-Server y los modelos fueron diseñados para operar en clúster el clúster como a continuación se describe:

Arquitectura tecnológica para los experimentos con ML

La arquitectura de despliegue con sparklyr corresponde al Modo clúster con YARN. El clúster es conformado por un conjunto de cinco maquinas que comparten los servicios de ML, con lo cual, si se produce una falla en una máquina, YARN está en capacidad de cambiar el flujo de trabajo a los nodos disponibles lo que le confiere tolerancia a fallas y

alta disponibilidad. Y por otro lado el clúster está diseñado para adaptar su rendimiento al volumen del problema por cuanto los cálculos son realizados en paralelo por todos los

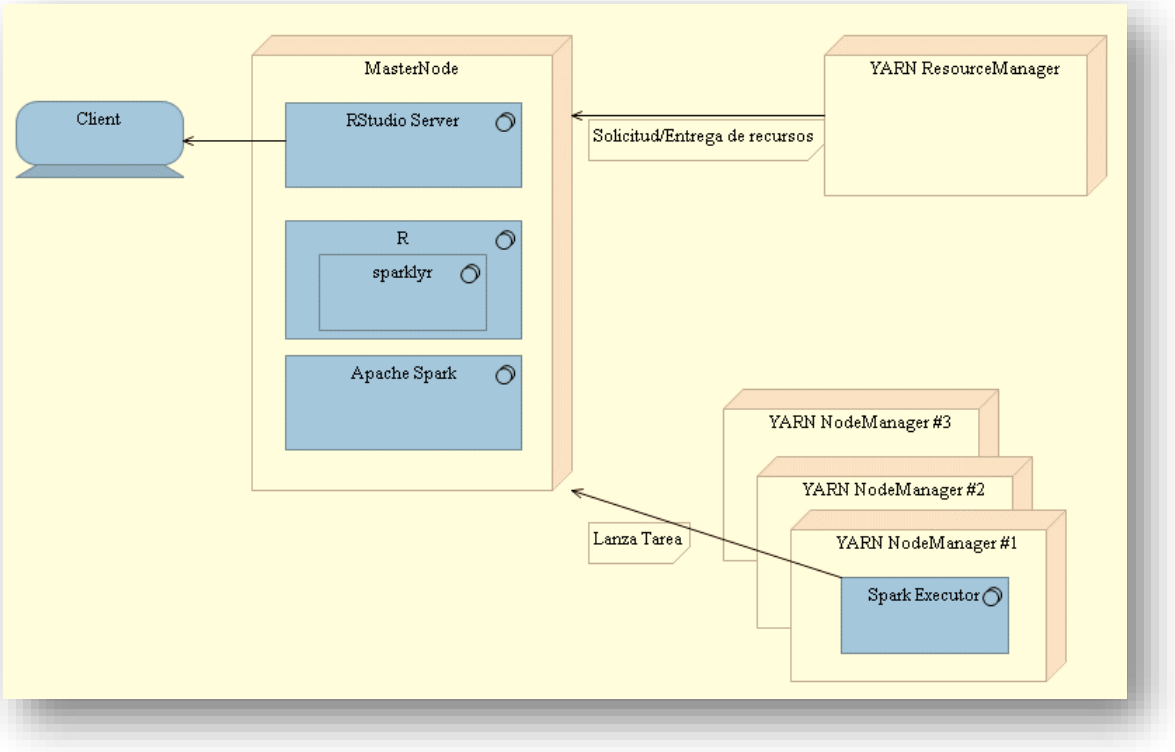


Figura 15: Arquitectura de despliegue de sparklyr: modo YARN-clúster

miembros de este, de manera que si se requiere hacer la tarea en menor tiempo esto se puede conseguir añadiendo nuevos nodos al cluster. En la figura 15 se ilustra con la notación en lenguaje Archimate (The Open Group, 2013) la arquitectura tecnológica de RServer (RStudio Team, 2016) con el motor de ML Spark, desplegado en clúster mediante YARN, el detalle de la arquitectura se describe en el Anexo K.

En este despliegue los datos se cargaron en RStudio Server y luego son transferidos a los nodos del clúster utilizando la librería sparklyr quien usando sintaxis de R permite acceder al clúster Hadoop. R Server está instalado en el nodo maestro o MasterNode (ver figura 15), el cual orquesta el análisis en ML con Spark, los recursos de hardware (nodos ejecutores) son provistos por el nodo maestro de YARN, es decir el ResourceManager, quien divide el trabajo en tareas y entrega para su ejecución en los NodeManager, estos nodos a su vez utilizando ejecutores de Spark para en análisis con ML (ver "Spark Executor" en la figura 15). Una vez se han resuelto las tareas el flujo es contrario, los NodeManager informan al ResourceManager, esta colecta los resultados y notifica el resultado al nodo maestro quien recupera los datos y muestra en la consola de RStudio Server los resultados al cliente. En el Anexo H se registran los 4 nodos que conforman el clúster, y en el Anexo L se registra los objetos en Spark vistos desde la consola de R-Server

Validación cruzada con cinco iteraciones

Se realizaron $K = 5$ iteraciones, de cada iteración tomaron parte 1000 diferentes documentos conforme se describe en la metodología, a continuación, se describen los resultados del primer experimento para los 3 modelos SVM, GBT y NB:

La bolsa de palabras utilizada en la primera iteración, es una matriz de 1000 documentos el cual se dividió en dos subconjuntos, el subconjunto de entrenamiento con 801 documentos, aproximadamente el 80 % de los elementos y el conjunto de validación con los restantes 199 documentos, cerca del 20% de los elementos.

Los modelos se construyeron con el subconjunto de entrenamiento y luego el modelo se probó en el subconjunto de validación, las tareas que se ejecutaron en el clúster se evidencian en la herramienta de métricas Hadoop Anexo J.

- Primera iteración para Máquina de Soporte Vectorial

		Dependencia									
		1	**2**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Predicción	**1**	3	0	0	0	0	0	0	0	0	
	2	0	3	0	0	0	0	0	0	0	
	7	0	0	4	0	0	1	0	0	0	
	8	0	0	1	12	0	2	1	1	1	
	9	0	0	0	0	2	0	0	1	0	
	10	0	1	0	1	0	18	8	8	0	
	11	0	1	0	0	0	4	15	0	1	
	12	0	0	0	3	1	10	5	89	1	
	13	0	0	0	0	0	0	0	0	1	
Distribución de Clases		**3**	**5**	**5**	**16**	**3**	**35**	**29**	**99**	**4**	**Exactitud=0.7387**
Total documentos		**199**									

Tabla 9: Matriz de confusión para el modelo SVM

La implementación en Spark de SVM es binaria por lo que se utilizó el algoritmo SVM junto con el método de clasificación multiclase para llegar a las 9 Dependencias que tomaron parte del experimento, la equivalencia entre el identificador numérico de la clase y la dependencia puede ser consultada en la Anexo I. En la matriz de confusión del modelo SVM generado para los documentos que hacen parte del subconjunto validación (ver tabla 9) se pude establecer que el modelo predijo correctamente la clase de los 3 documentos que pertenecen la Dependencia “1”, de los 5 documentos que hacen parte de la Dependencia “2” el modelo predijo correctamente 3, 2 más fueron clasificadas erróneamente. De los 5 documentos que hacen parte de la clase “7” el modelo predijo incorrectamente 1, continuando así sucesivamente se obtiene una exactitud sobre todas las clases de 0.7, se observa que para la clase “13” el rendimiento fue de apenas 1 de 4 clasificados correctamente.

- Primera iteración para Naive Bayes

El modelo NB es implementado en Spark como multiclase por lo que no se utilizaron técnicas adicionales para entrenar el modelo. En la matriz de confusión del modelo generado (ver tabla 10), se describen las predicciones del modelo por Dependencia y la exactitud general de 0.68.

		Dependencia									
		1	**2**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Predicción	**1**	2	0	0	0	0	0	0	0	0	
	2	0	1	0	0	0	0	0	0	0	
	7	0	1	4	0	0	3	1	0	1	
	8	0	0	1	11	0	1	1	2	1	
	9	0	1	0	0	2	1	0	0	0	
	10	0	1	0	0	0	16	3	3	0	
	11	0	1	0	0	0	4	15	3	1	
	12	1	0	0	5	1	9	7	85	0	
	13	0	0	0	0	0	1	2	6	1	
Distribución de clases		**3**	**5**	**5**	**16**	**3**	**35**	**29**	**99**	**4**	**Exactitud= 0.6884**
Total documentos		**199**									

Tabla 10: Matriz de confusión para el modelo NB

- Primera iteración para Gradient Boosting Tree

Esta implementación es binaria por lo que nuevamente se utilizó un método multiclase de Spark para clasificar los documentos en las 9 Dependencias, el algoritmo GBT realiza 20 iteraciones, luego el árbol de decisión final se compone de 20 modelos y la decisión de predicción se obtiene al promediar los resultados de los modelos individuales. La exactitud en la matriz de confusión para el modelo GBT está consignada en la tabla 11, en la cual se

observa que en el subconjunto de validación únicamente se clasificó mal un documento que pertenecía a la Dependencia “12” y la predicción lo clasificó en la “7”.

La sintaxis en R de la primera iteración con el entrenamiento de los 3 modelos, incluido el método multiclase cuando fue requerido, así como las pruebas con el subconjunto de validación y los resultados obtenido se consignaron en el Anexo E.

		Dependencia									
		1	**2**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Predicción	**1**	3	0	0	0	0	0	0	0	0	
	2	0	5	0	0	0	0	0	0	0	
	7	0	0	5	0	0	0	0	1	0	
	8	0	0	0	16	0	0	0	0	0	
	9	0	0	0	0	3	0	0	0	0	
	10	0	0	0	0	0	35	0	0	0	
	11	0	0	0	0	0	0	29	0	0	
	12	0	0	0	0	0	0	0	98	0	
	13	0	0	0	0	0	0	0	0	4	
Distribución de clases		**3**	**5**	**5**	**16**	**3**	**35**	**29**	**99**	**4**	**Exactitud= 0.995**
Total documentos		**199**									

Tabla 11: Matriz de confusión para el modelo GBT

Las demás 4 iteraciones son repetición del primero con diferentes documentos conformando los subconjuntos de entrenamiento y validación. Al final de estas 5 iteraciones tomaron parte del experimento 5000 documentos, la distribución de las

$\#C_i$	$n_{1\,iteración}^{entrenam}$	$n_{1\,iteración}^{validación}$	n_2^e	n_2^v	n_3^e	n_3^v	n_4^e	n_4^v	n_5^e	n_5^v
1	3	12	6	23	8	33	4	15	14	58
2	5	18	7	28	3	14	1	6	1	4
7	5	21	11	44	8	32	6	26	2	6
8	16	66	12	48	42	170	82	327	82	330
9	3	13	7	26	3	10	0	1	1	4
10	35	141	44	178	28	112	23	93	24	96
11	29	116	40	160	52	209	27	109	21	84
12	99	398	70	278	50	198	50	202	46	182
13	4	16	4	14	6	2	6	22	9	36
Sub-conjuntos	**199**	**801**	**201**	**799**	**200**	**780**	**199**	**801**	**200**	**800**
Documentos por iteración	**1000**		**100**		**1000**		**1000**		**1000**	
Muestras para aprendizaje	**5000**									

Tabla 12: Tabla de distribución de frecuencias para los experimentos

Clases(C_i) en de la serie de experimentos esta registrada en la Tabla 12, y todo el script de las iteraciones puede ser consultado en el Anexo S.

Evaluación y Selección.

El resultado de la serie de experimentos se consigna en la tabla 13 utilizando la prueba según Friedman, en paréntesis se le asignó un ranking a cada exactitud obtenida, siendo (1) el algoritmo con más alta exactitud y (3) el que obtuvo la peor. Y al final se promedian los resultados del ranking.

Iteración	NB	SVN	GBT
1	0.6884 **(3)**	0.7387 **(2)**	0.9950 **(1)**
2	0.5970 **(3)**	0.7065 **(2)**	1.000 **(1)**
3	0.7050 **(3)**	0.7600 **(2)**	0.995 **(1)**
4	0. 6795 **(3)**	0. 8492 **(2)**	1.000 **(1)**
5	0. 7550 **(3)**	0.8200 **(2)**	0.975 **(1)**
Promedio del ranking	**3**	**2**	**1**

Tabla 13: Ranking con el resultado de los experimentos

De la anterior evaluación se concluye que el mejor promedio del ranking con valor de 1, fue el obtenido por el modelo GBT por lo que, el candidato seleccionado es el modelo Gradient Boosting Tree.

Prueba de hipótesis

Con el modelo GBT obtenido durante de la serie de iteraciones, se probó la exactitud utilizando el conjunto de datos de comprobación, el cual conforme se explica en la metodología se compone de 1000 documentos no vistos anteriormente, en la siguiente matriz de confusión se muestran las predicciones por Dependencia.

		Dependencia									
		1	**2**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	
Predicción	**1**	32	0	0	0	0	0	0	0	0	
	2	0	1	0	0	2	0	0	0	0	
	7	0	0	4	0	2	0	0	0	0	
	8	0	7	4	448	22	0	0	0	0	
	9	1	0	1	0	1	0	0	0	0	
	10	0	0	1	0	0	127	0	0	0	
	11	0	0	0	0	0	0	107	0	0	
	12	0	0	0	0	0	0	0	203	0	
	13	0	0	0	0	0	0	0	0	37	
Distribución de clases		**33**	**8**	**10**	**448**	**27**	**127**	**107**	**203**	**37**	**Exactitud= 0.96**
Total documentos		**1000**									

Tabla 14: Matriz de confusión para el modelo seleccionado

La exactitud del modelo es de .96 en porcentaje equivale al 96%, resultado por el cual se acepta la hipótesis planteada por cuanto la calidad y cantidad de términos junto con el modelo con mayor efectividad, permitió obtener el mejor modelo posible con exactitud comparable con el estado del arte en clasificación de texto (Sebastiani, 2002) (Cardoso-Cachopo & Limede Oliveira, 2003) (Rousu, Saunders, Szedmak, & Shawe-Taylor, 2005) (Dumais, Platt, & Heckerman, 1998). En la siguiente tabla se puede ver la distribución de frecuencias utilizadas durante la prueba de hipótesis.

$\#C_i$	**Dependencia**	n_h
1	Dirección jurídica	33
2	Grupo de servicio al ciudadano institucional	8
7	Grupo de gestión meritocracia	10
8	Dirección de participación, transparencia y servicio al ciudadano	448
9	Oficina de tecnologías de la información y las comunicaciones	27
10	Grupo de gestión financiera	127
11	Despacho del director	107
12	Grupo de gestión contractual	203
13	Grupo de gestión administrativa	37
Total		**1000**

Tabla 15: Distribución de clases en el conjunto de comprobación

Se puede observar que las distribuciones de las frecuencias de categorías en los conjuntos de entrenamiento, validación para las 5 iteraciones registradas en la tabla 12 y de comprobación en la tabla 15, son diferentes y el modelo consistentemente obtiene una exactitud, en todos los casos superior a la expectativa. La sintaxis completa de prueba se registra en el Anexo S.

Con la etapa de prueba se concluye este capítulo por lo que a continuación se presentan las conclusiones relativas a los resultados del caso de estudio.

V. Conclusiones

Este estudio exploró como la entidad del caso puede usar aprendizaje de máquina para clasificar la correspondencia electrónica con el propósito de codificar el conocimiento.

En consecuencia, y en concordancia directa con los objetivos del proyecto se tiene:

- Se elaboró y presentó una propuesta metodológica en un sistema de diferentes etapas lógicamente organizadas; (i) Recolección de documentos, (ii) Pre-procesamiento, (iii) Aprendizaje, (iv) Evaluación y selección, (v) Prueba de hipótesis, las cuales permitieron la construcción de un modelo capaz de clasificar la correspondencia electrónica en 9 clases con una precisión del 96%, probado para operar con diferentes distribuciones de clases, superando la expectativa planteada en la hipótesis.
- Se levantó la información de entrevistas con los funcionarios que hacen parte del proceso de administración de correspondencia y del análisis de los documentos que hacen parte del mencionado proceso. Esto permitió realizar una semblanza de los rasgos característicos del manejo de la correspondencia en la entidad del caso. Al respecto de las variables para representar el corpus de documentos se realizó una implementación para recolección de documentos y pre-procesamiento de texto utilizando técnicas de uso común con herramientas para analítica de texto en R-Studio y de consulta a bases de datos transaccionales, lo cual por cierto, significó el mayor empleo de tiempo y esfuerzo, dado que los correos electrónicos al provenir de distintas fuentes usan codificación de texto diferentes lo que dificultaba la extracción y limpieza del texto. Finalmente se formatearon los datos para la capa

de aprendizaje y se realizó una reducción de dimensión, creando así un motor de preprocesamiento robusto y fiable en el caso de estudio.

- La parte de innovación de este trabajo viene del diseño ajustado casi personalizado al caso de estudio con una metodología y una posible arquitectura de producción para el modelo, de forma que permita satisfacer en un futuro la escalabilidad de la solución dentro de la organización.
- Finalmente, los algoritmos de aprendizaje de máquina en Big Data pueden potencialmente realizar tareas de clasificación de textos con grandes cantidades de documentos, con la velocidad que se requiera y con un número elevado de categorías, lo que amplía el alcance de aplicación a chats, redes sociales, formularios Web, en fin, casi cualquier medio de comunicación escrito. En el contexto del caso estudiado el modelo obtenido podría llegarse a convertir en el cerebro de un *hub* de interacción del estado con el ciudadano.

Este estudio tiene limitaciones por lo que hay margen para futuras investigaciones. Primero, el estudio no exploró los métodos para recuperar información de los archivos adjuntos con contenido en diversos formatos de texto, como tampoco audio, imagen y video *(Multimedia information retrieval MIR)*. En segundo lugar, este trabajo de grado se centró en la exactitud por considerarla el factor más relevante del modelo y aunque éste incluye las características de escalabilidad y disponibilidad, estas no fueron analizadas de forma sistemática lo que podría dar lugar a un estudio comparativo de la eficiencia de los algoritmos para análisis de texto en clúster. Por último hay espacio para estudios de

comparación de herramientas de clasificación a través de ML, como Spark (utilizado en este trabajo de grado), Python, TensorFlow, Etc.

Si bien la validez de esta investigación es interna, por cuanto el corpus de documento es privado y no puede ser publicado para otros trabajos, creo que ayudará a profesionales interesados con ML en Big Data, como base para realizar sus estudios en el área de clasificación de texto e inclusive otros problemas que puedan ser resueltos con modelos predictivos.

VI. Bibliografía

Freund , Y., & Schapire, R. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*.

Aggarwal, C., & Zhai, C. (2012). A SURVEY OF TEXT CLASSIFICATION ALGORITHMS. In *Mining Text Data* (pp. 163-222).

Bouckaer, R., & Frank, E. (2004). Evaluating the replicability of significance tests for comparing learning algorithms. *Advances in Knowledge Discovery and Data Mining*, volume 3056, pp. 3–12.

Boyd, D., & Crawford, K. (2012). Critical Questions For Big Data. *Information, Communication & Society*, 662-679.

Buckley, G. S. (1988). Term-weighting approaches in automatic text retrieval.

Cardoso-Cachopo, A., & Limede Oliveira, A. (2003). An Empirical Comparison of Text. *String Processing and Information Retrieval*.

Charalampakis, D., Spathis, E., Kouslis, & Kermanidis, K. (2016). A comparison between semisupervised and supervised text mining techniques on detecting irony in greek political tweets. *Engineeringing Applications of Artificial Intelligence*.

Código de Procedimiento Administrativo y de lo Contencioso Administrativo, Ley Nº 1437 de 2011 (enero 18, 2011).

Congreso de Colombia. (2011). *Código de Procedimiento Administrativo y de lo Contencioso Administrativo, Ley Nº 1437 de 2011 (18 de enero de 2011).*

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 273–297.

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. *Sixth Symposium on Operating System Design and Implementation.* San Francisco, CA .

Dehaene S, M. J. (1992). Cross-linguistic regularities in the frequency of number words. *Cognition. Apr*, 1-29.

Demchenko, Y., Grosso, P., de Laat, C., & Membrey, P. (2013). Addressing big data issues in Scientific Data Infrastructure. *2013 International Conference on Collaboration Technologies and Systems (CTS).* San Diego: IEEE.

Departamento Administrativo Nacional de Estadística (DANE). (2016). *Indicadores Básicos de Tenencia y Uso de Tecnologías de la Información y Comunicación – TIC en Hogares y Personas de 5 y más años de edad 2015.* Bogotá.

Dieng, R. C. (1999). Methods and tools for corporate knowledge management. *Internation Journal of Human Computer Studies*, 567-598.

Drovandi, C. C. (2017). Principles of Experimental Design for Big Data Analysis. *Statistical Science : A Review Journal of the Institute of Mathematical Statistics*, 385–404.

Dumais, S., Platt, J., & Heckerman, D. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of CIKM '98 - 7th International Conference on Information and Knowledge Management. Bethesda, MD, USA, 19981103* (pp. 148-55). New York: Bethesda, MD USA: ACM Country of Publication: USA.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 861–874.

Flach, P. (2012). *Machine Learning The Art and Science of Algorithms that Make Sense of Data.* Cambridge University Press.

Fortuna, B., Mladeni, D., & Grobelnik, M. (2005). Visualization of Text Document Corpus. 2-5.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 1189-1232.

Función Pública. (2017, 6 30). *Intranet Función Pública*. Retrieved from Procedimiento Correspondencia: https://intranet.funcionpublica.gov.co/procedimiento-administracion-de-correspondencia

García Adeva, Pikatza Atxa, Ubeda Carrillo, M., & Ansuategi Zengotitabengoa, E. (2014). Automatic text classification to support systematic reviews in medicine. *Expert Systems with Ap- plications*, 1498–1508.

Guus T. Schreiber, H. A. (2000). *Knowledge engineering and management: the CommonKADS methodology.* Cambridge, MA, USA: MIT Pres.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: data mining, inference and prediction.* Springer.

JOACHIMS, T. (1998). Text categorization with support vector machines: learning with many relevant features. *10th European Conference on Machine Learning*, (pp. 137–142). Germany.

Kohavi, R., Fawcett, T., & Provost, F. (1998). *Proceeding ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning*, 445-453.

Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*.

Luhn, H. P. (1960). Keyword-in-context index for technical literature. *American Documentation*, 288–295.

Maron, M. E. (1961). Automatic Indexing: An Experimental Inquiry. *Maron, M. E.*, 1961.

McCulloch, W., & Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. Chicago: University of Chigcago Press.

Meek, C., & Brutlag, J. (2000). Challenges of the email domain for text classification. *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 103–110). San Francisco: Morgan Kaufmann Publishers Inc.

Mironczuk, M., & Protasewikz, J. (2018). A recent overview of the state-of-the-art elements of text. *Expert System with Applications*.

Mitchell, M. (1997). *Machine learning.* McGrawHill.

Naika, N., & Purohitb, S. (2017). Comparative Study of Binary Classification Methods to Analyze a Massive Dataset on Virtual Machine. *Procedia Computer Science*, 1863-1870.

Ortigosa, A., Martín, J., & Carro, R. (2014). Sentiment analysis in Facebook and its application to e-learning. *Computers in human behavior*.

Peralta , D., S, R., S, R.-G., & I, T. (2015). Evolutionary Feature Selection for Big Data Classification: A MapReduce Approach. *Mathematical Problems in Engineering*, 11.

Piantadosi. (2012). *Approximate number from first principles.*

R Core, T. (2014). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing.

Radicati Group, Inc. (2016). Email Market, 2016-2020.

Rousu, J., Saunders, C., Szedmak, S., & Shawe-Taylor, J. (2005). Learning hierarchical multi-category text classification models. *Proceedings of the 22nd international conference on Machine learning* (pp. 744-751). Bonn, Germany: ACM.

RStudio Team. (2016). RStudio: Integrated Development Environment for R. Massachusetts, Boston: RStudio, Inc.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 1-47.

Seguer, R. G. (2007). Propuesta para el Modelado del Conocimiento Empresarial.

Superintendencia de Servicios Públicos Domiciliarios. (2006). *www.orfeoGPL.org*. Retrieved from www.orfeoGPL.org

The Open Group. (2013). *ArchiMate 2.1 Specification.* Van Haren Publishing.

Thomas H. Davenport, L. P. (1998). *Working Knowledge: How Organizations Manage What They Know*. Harvard Business Press.

Vapnik , V. (1982). *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics.* Springer-Verlag New York, Inc. Secaucus, NJ, USA.

White, T. (2009). *Hadoop: The Definitive Guide.* O'Reilly.

YANG, Y. A. (1997). A comparative study on feature selection in text categorization. *Proceedings of ICML-97, 14th International*, (pp. 412–420.). Nashville.

Yang, Y., & Pedersen, J. (1997). A comparative study on feature selection in text categorization.

Zikopoulos, P., Eaton, C., deRoos, D., Deutsch, T., & Lapis, G. (2011). *Under-standing big data: Analytics for enterprise class hadoop and streaming data.* Mc Graw Hill.

Zipf, G. (1936). *The Psychobiology of Language.* London.

VII. Anexos

ACTA DIRECCIONAMIENTO DE CORRESPONDENCIA

CIUDAD Y FECHA: Bogotá, 03 de septiembre de 2015

(.....)

Los temas de cada una de las Direcciones Técnicas y el Grupo de Atención al Ciudadano son una herramienta fundamental para el direccionamiento de las peticiones que ingresan a la Función Pública por los diferentes canales (ventanilla de correspondencia, correo web master, fax institucional y PQR's Página Web),

A continuación se relacionan los temas a cargo de cada Dirección Técnica, y del Grupo de Atención al Ciudadano, así mismo se mencionarán las conclusiones:

(.....)

DIRECCION DE EMPLEO PÚBLICO

- Trámites
  - Ley 962/2005 y/o Decreto 019/2012
  - Defensa de trámites (según Decreto Antitramites)
  - Planes sectoriales de trámites
  - Inscripción y racionalización de trámites
  - Aprobación de nuevos trámites
  - Eliminación de trámites
  - SUIT – Sistema Único de Información de Trámites
  - Grupo de racionalización y Automatización de trámites – GRAT (Mesas Sectoriales)
  - Cadena de trámites
  - FURAG – Componente trámites
  - Plan anticorrupción – Componente trámites

GRUPO DE SERVICIO AL CIUDADANO

- Reclamos
- Reiteración de peticiones
- Denuncias por actos de corrupción
- Quejas
- Traslados por competencia
  - Instrumento evaluación del desempeño (CNSC)
  - Inscripciones carrera administrativa (CNSC)
  - Certificaciones sobre inscripción a carrera administrativa (CNSC)
  - Listas de legibles (CNSC)
  - Concursos (CNSC)
  - Evaluaciones de desempeño funcionarios de carrera y libre nombramiento (CNSC)
  - Convocatorias públicas (CNSC)
  - Creación, conformación e implementación de la comisión de personal (CNSC)
  - Interpretación del código contencioso administrativo (Min. Interior)
  - Supervisión de contratos estatales (Colombia Compra Eficiente)

(.....)

Anexo A: Extracto de acta de Direccionamiento 2015

Anexo B: Código fuente de un documento de correspondencia

```
> html.files <- list.files("./archivos",include.dirs =TRUE)
> source("./htmlToText.R")
> html.to.txt <- lapply(html.files, htmlToText)
> strOptions(html.files)
$strict.width
   [1] "20142060000032.html" "20142060000122.html" "20142060000542.html" "20142060000572.html"
"20142060000592.html" [6] (…)
 [996] "20142060057582.html" "20142060057602.html" "20142060057612.html" "20142060057622.html"
"20142060057712.html"
 [ reached getOption("max.print") -- omitted 15821 entries ]
```

Anexo C: Script de la fase de recolección de Documentos en R

```
> txt.clean <- sapply(html.to.txt, function(x) iconv(x, "UTF-8", "ASCII/
/TRANSLIT", sub=""))
> txt.clean <- gsub("[[:cntrl:]]", " ", txt.clean)
> txt.clean <- tolower(txt.clean)
> txt.clean <- removeWords(txt.clean, words = stopwords("spanish"))
> txt.clean <- removePunctuation(txt.clean)
> txt.clean <- removeNumbers(txt.clean)

> txt.corpus <- Corpus(VectorSource(txt.clean))
> txt.corpus
<<SimpleCorpus>>
Metadata:  corpus specific: 1, document level (indexed): 0
Content:  documents: 7206
> library(wordcloud)
> wordcloud(txt. corpus, max.words = 80, random.order = F, colors = brew
er.pal(name = "Dark2", n = 8))

> txt.clean <- removeWords(txt.clean, words = stopwords("english"))

> txt.clean <- removeWords(txt.clean, words = c("evafuncionpublicagovco"
, "webmasterdafpgovco", "webmasterfuncionpublicagovco", "fecha", "asunto
", "radicado","archivos", "mensaje", "correo", "ext", "mail", "from", "w
ed","tue","thu","tue","cot","and","cordial","gracias","cordialmente","fr
i","mon","for","fin","tel"))

> matrix.terminos <- TermDocumentMatrix(txt.corpus)
<<TermDocumentMatrix (terms: 54488, documents: 7206)>>
Non-/sparse entries: 626438/392014090
Sparsity           : 100%
Maximal term length: 586
Weighting          : term frequency (tf)
matrix.terminos.reddimension <- removeSparseTerms(matrix.terminos, spars
e = .972)
> matrix.terminos.reddimension
<<DocumentTermMatrix (documents: 7206, terms: 573)>>
Non-/sparse entries: 315243/3813795
Sparsity           : 92%
Maximal term length: 22
Weighting          : term frequency (tf)
```

Anexo D: Script de la fase de pre-procesamiento de Documentos en R

```
> library(dplyr)
> training.subset.tbl <- copy_to(sc,training.subset,overwrite = TRUE)
> test.subset.tbl <- copy_to(sc,test.subset,overwrite = TRUE)
> src_tbls(sc)
[1] "sparklyr_6f4b4c7187fb" "sparklyr_6f4bd037ad0"

> feature_colnames <- training.subset.tbl %>%
+   select(1:574)%>%
+   # obtienen los nombres de las columnas
+   colnames()
> modelo <- ml_linear_svc(sc)
> one_vs_rest <- ml_one_vs_rest(training.subset.tbl,classifier=modelo,re
sponse = "clases", features = c(feature_colnames))
> responses <- test.subset.tbl  %>%
+   # Selecciona la columna de respuesta
+   select(clases) %>%
+   # Recoge los resultados
+   collect() %>%
+   # Agrega la columna de predicciones
+   mutate(
+     predicted_class = predict(
+       one_vs_rest,
+       test.subset.tbl
+     )
+   )
> head(responses)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     12 12
2     11 11
3     11 12
4     12 12
5     12 12
6     12 12


> confusionMatrix(responses$predicted_class,responses$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   3  0  0  0  0  0  0  0  0
        2   0  3  0  0  0  0  0  0  0
        7   0  0  4  0  0  1  0  0  0
        8   0  0  1 12  0  2  1  1  1
        9   0  0  0  0  2  0  0  1  0
        10  0  1  0  1  0 18  8  8  0
        11  0  1  0  0  0  4 15  0  1
        12  0  0  0  3  1 10  5 89  1
```

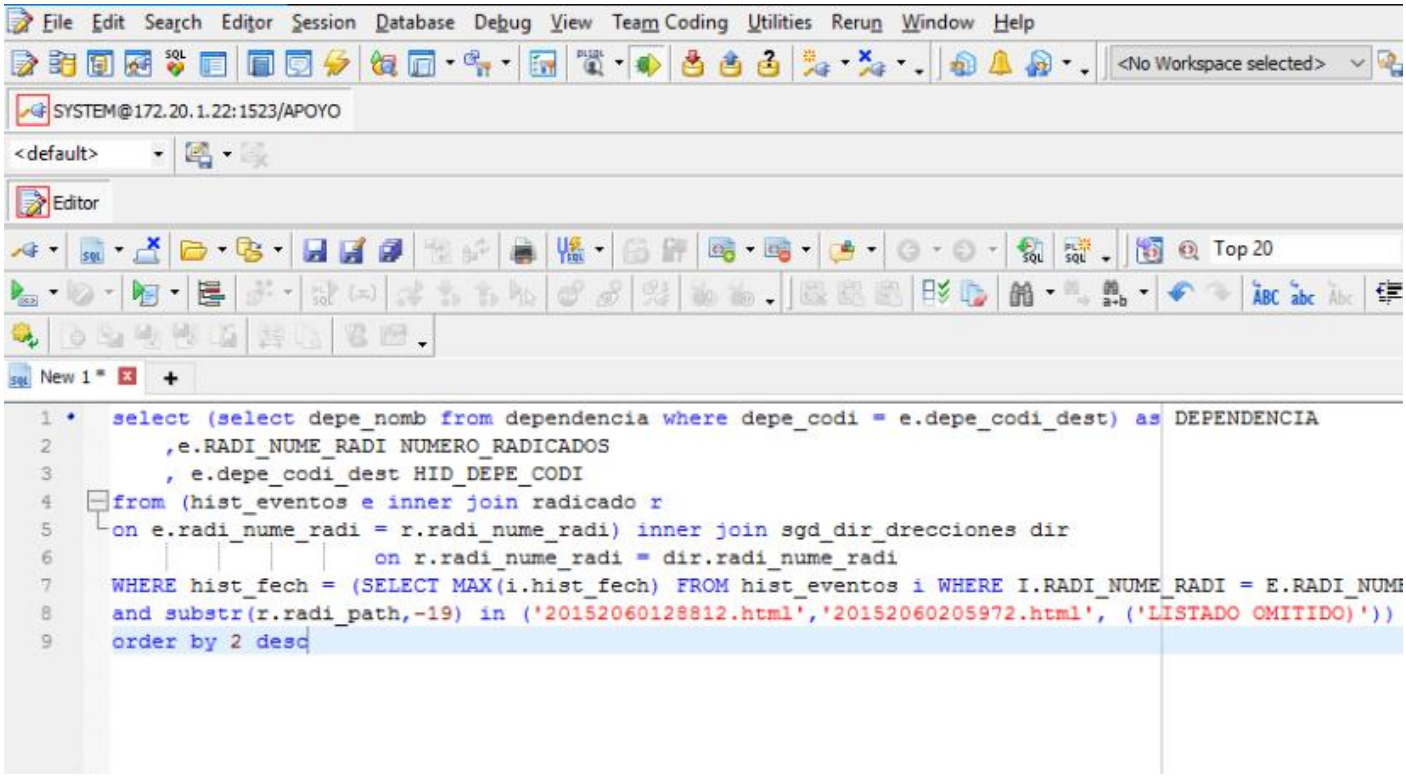
```
        13  0  0  0  0  0  0  0  0  1

Overall Statistics

               Accuracy : 0.7387
> modelo_bt <- ml_gbt_classifier(sc)
> one_vs_rest_bt <- ml_one_vs_rest(training.subset.tbl,classifier=modelo
_bt,response = "clases", features = c(feature_colnames))
> responses_bt <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       one_vs_rest_bt,
+       test.subset.tbl
+     )
+   )
> head(responses_bt)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     12 12
2     11 11
3     11 11
4     12 12
5     12 12
6     12 12
> confusionMatrix(responses_bt$predicted_class,responses_bt$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   3  0  0  0  0  0  0  0  0
        2   0  5  0  0  0  0  0  0  0
        7   0  0  5  0  0  0  0  1  0
        8   0  0  0 16  0  0  0  0  0
        9   0  0  0  0  3  0  0  0  0
        10  0  0  0  0  0 35  0  0  0
        11  0  0  0  0  0  0 29  0  0
        12  0  0  0  0  0  0  0 98  0
        13  0  0  0  0  0  0  0  0  4

Overall Statistics

               Accuracy : 0.995
> modelo_nb=ml_naive_bayes(training.subset.tbl,response = "clases", feat
ures = c(feature_colnames))
> responses_nb <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
```

```
+   mutate(
+     predicted_class = predict(
+       modelo_nb,
+       test.subset.tbl
+     )
+   )
> head(responses_nb)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     12 12
2     11 8
3     11 11
4     12 13
5     12 12
6     12 12
> confusionMatrix(responses_nb$predicted_class,responses_nb$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   2  0  0  0  0  0  0  0  0
        2   0  1  0  0  0  0  0  0  0
        7   0  1  4  0  0  3  1  0  1
        8   0  0  1 11  0  1  1  2  1
        9   0  1  0  0  2  1  0  0  0
        10  0  1  0  0  0 16  3  3  0
        11  0  1  0  0  0  4 15  3  1
        12  1  0  0  5  1  9  7 85  0
        13  0  0  0  0  0  1  2  6  1

Overall Statistics

               Accuracy : 0.6884
```

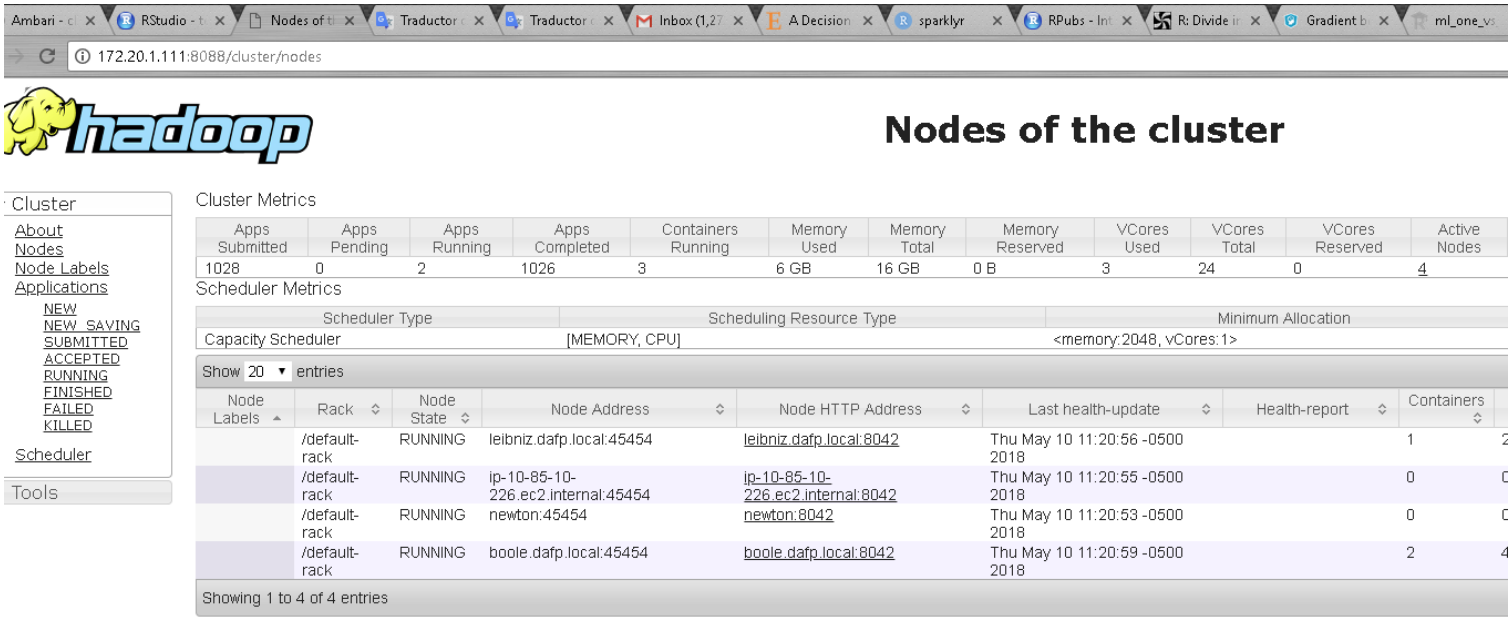
Anexo E: Script de la fase del experimento inicial en R

```
> table(matrix.terminos.reddimension6$clases)
  1   2   7   8   9  10  11  12  13
 33   8  10 448  27 127 107 203  37
> Comprobacion.subset.tbl <- copy_to(sc,matrix.terminos.reddimension6,ov
erwrite = TRUE)
> responses_GBT_win <- Comprobacion.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       one_vs_rest_bt,
+       Comprobacion.subset.tbl
+     )
+   )
> confusionMatrix(responses_GBT_win$predicted_class,responses_GBT_win$cl
ases)
Confusion Matrix and Statistics

          Reference
Prediction   1   2   7   8   9  10  11  12  13
        1   32   0   0   0   0   0   0   0   0
        2    0   1   0   0   2   0   0   0   0
        7    0   0   4   0   2   0   0   0   0
        8    0   7   4 448  22   0   0   0   0
        9    1   0   1   0   1   0   0   0   0
        10   0   0   1   0   0 127   0   0   0
        11   0   0   0   0   0   0 107   0   0
        12   0   0   0   0   0   0   0 203   0
        13   0   0   0   0   0   0   0   0  37

Overall Statistics

               Accuracy : 0.96
```

Anexo F: Script de la prueba en el conjunto de datos de comprobación



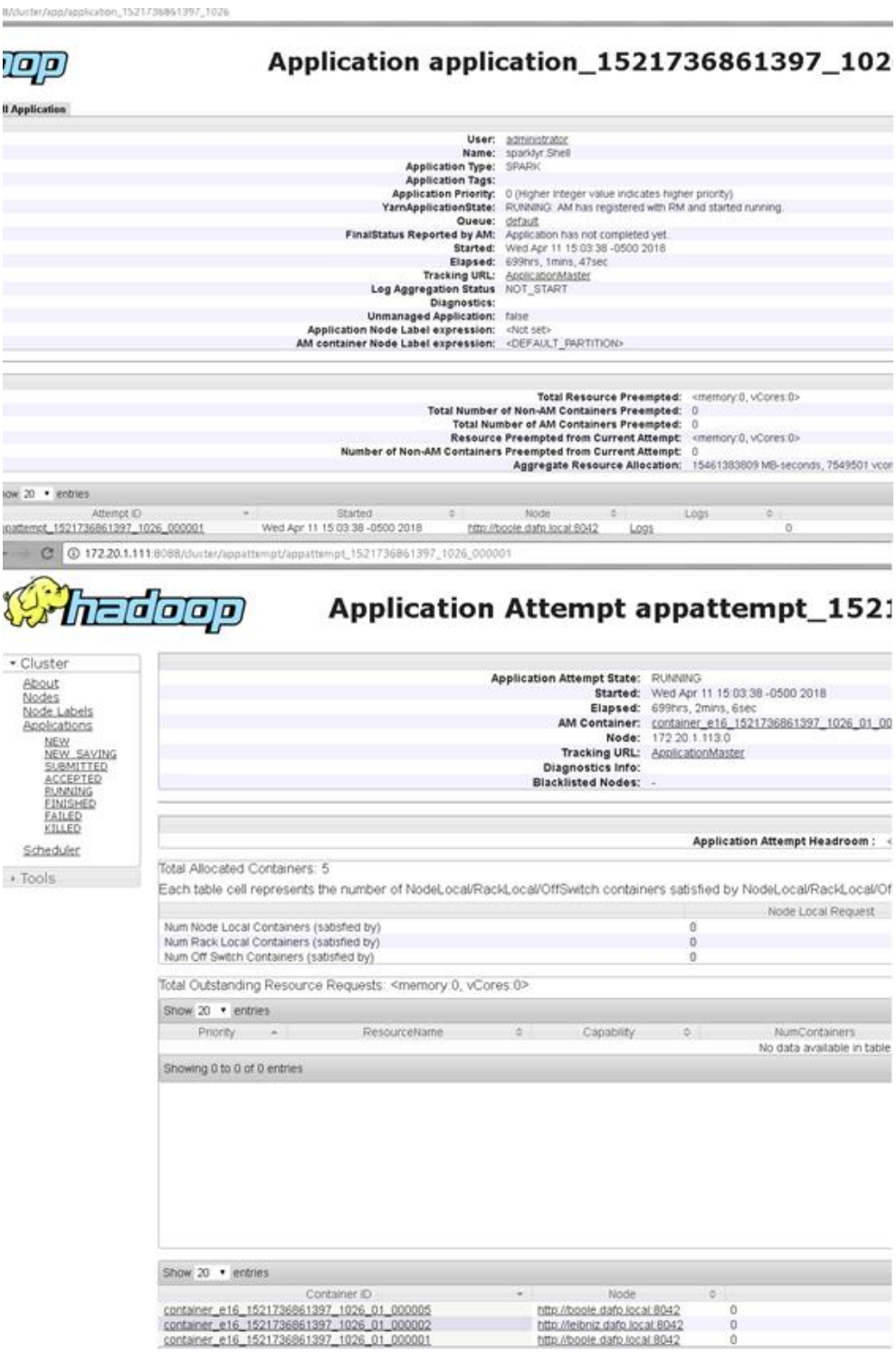
Anexo G: Parte de la consulta de dependencias



Anexo H: Nodos del clúster

$\#C_i$	**Clase**
1	Dirección jurídica
2	Grupo de servicio al ciudadano institucional
7	Grupo de gestión meritocracia
8	Dirección de participación, transparencia y servicio al ciudadano
9	Oficina de tecnologías de la información y las comunicaciones
10	Grupo de gestión financiera
11	Despacho del director
12	Grupo de gestión contractual
13	Grupo de gestión administrativa

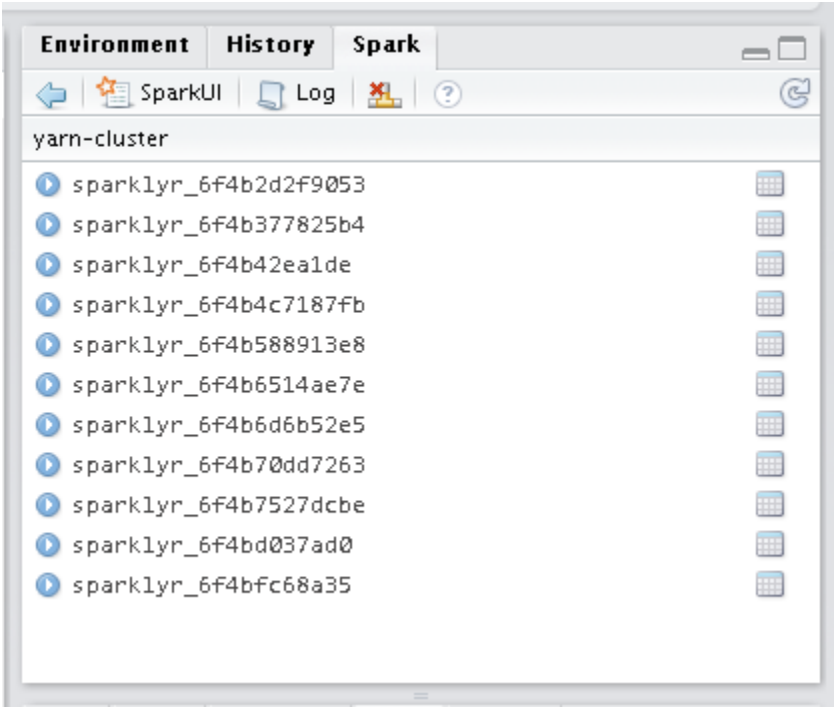
Anexo I: Tabla de correspondencia Clase / Dependencia



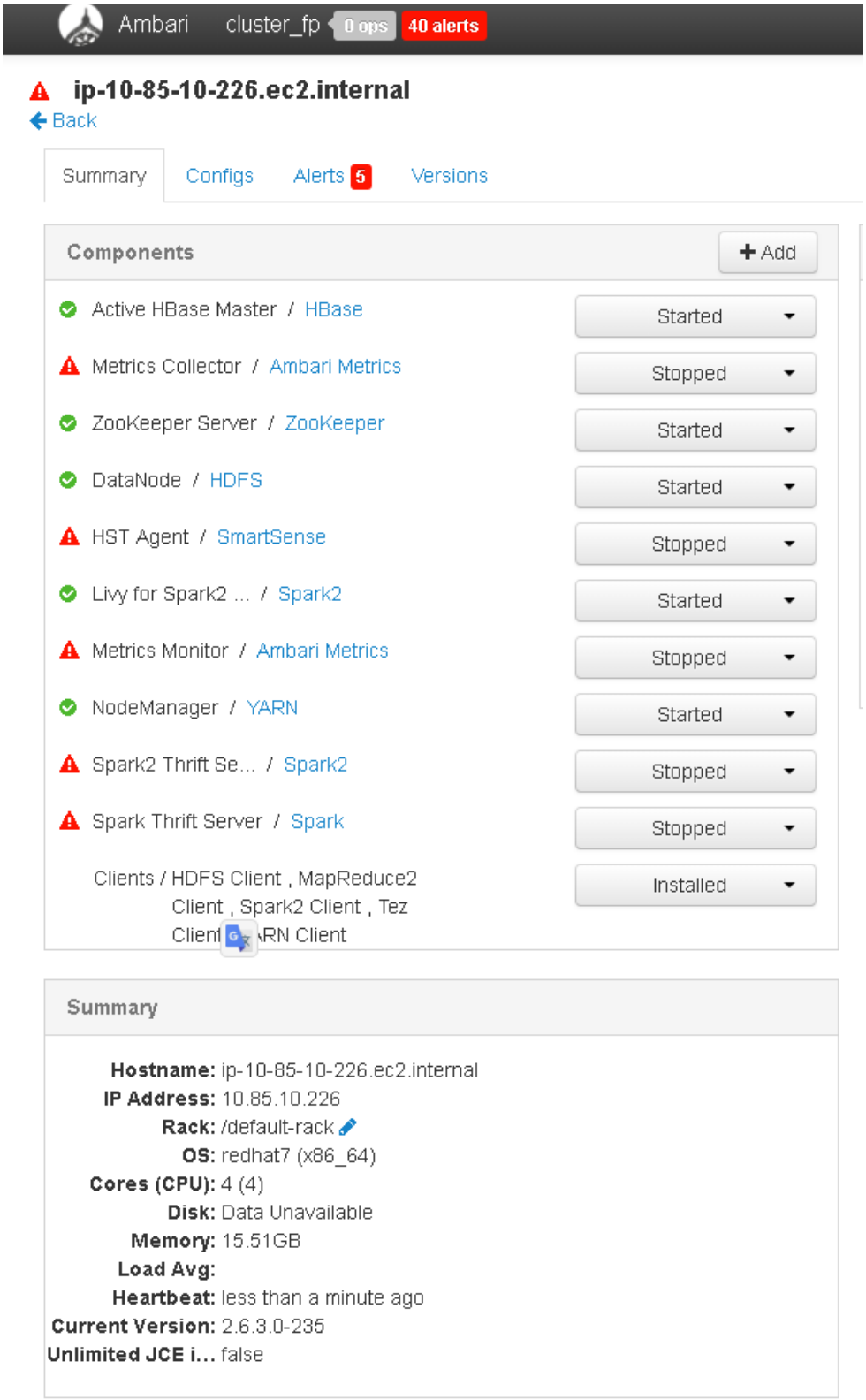
Anexo J: Tarea de SPARK en ejecución en 5 contenedores, 1 ResourceManager y 4 NodeManagers.

Nombre	Hardware	Software para ML	Rol
leibniz	Virtual en nube privada con: 8 VCPU y 16 Gbytes RAM, ver Anexo I	RServer, Spark, HDFS y NodeManager/YARN	MasterNode
newton	Nube privada: 4 VCPU y 8 Gbytes RAM, ver Anexo J	ResourceManager/YARN, HDFS y Spark2 Client	ResourceManager
ip-10-85-10-226.ec2.internal	Virtual en nube publica AWS con: 4 VCPU y 16 Gbytes RAM, ver Anexo H	NodeManager/YARN, HDFS y Spark2 Client	NodeManager
boole	Nube privada: 8 VCPU y 8 Gbytes RAM, ver Anexo K	NodeManager/YARN, HDFS, Ambari Server y Spark2 Client	NodeManager
dafppr02	Nube privada: 2 VCPU y 8 Gbytes RAM, ver Anexo L	NodeManager/YARN, HDFS y Spark2 Client	NodeManager

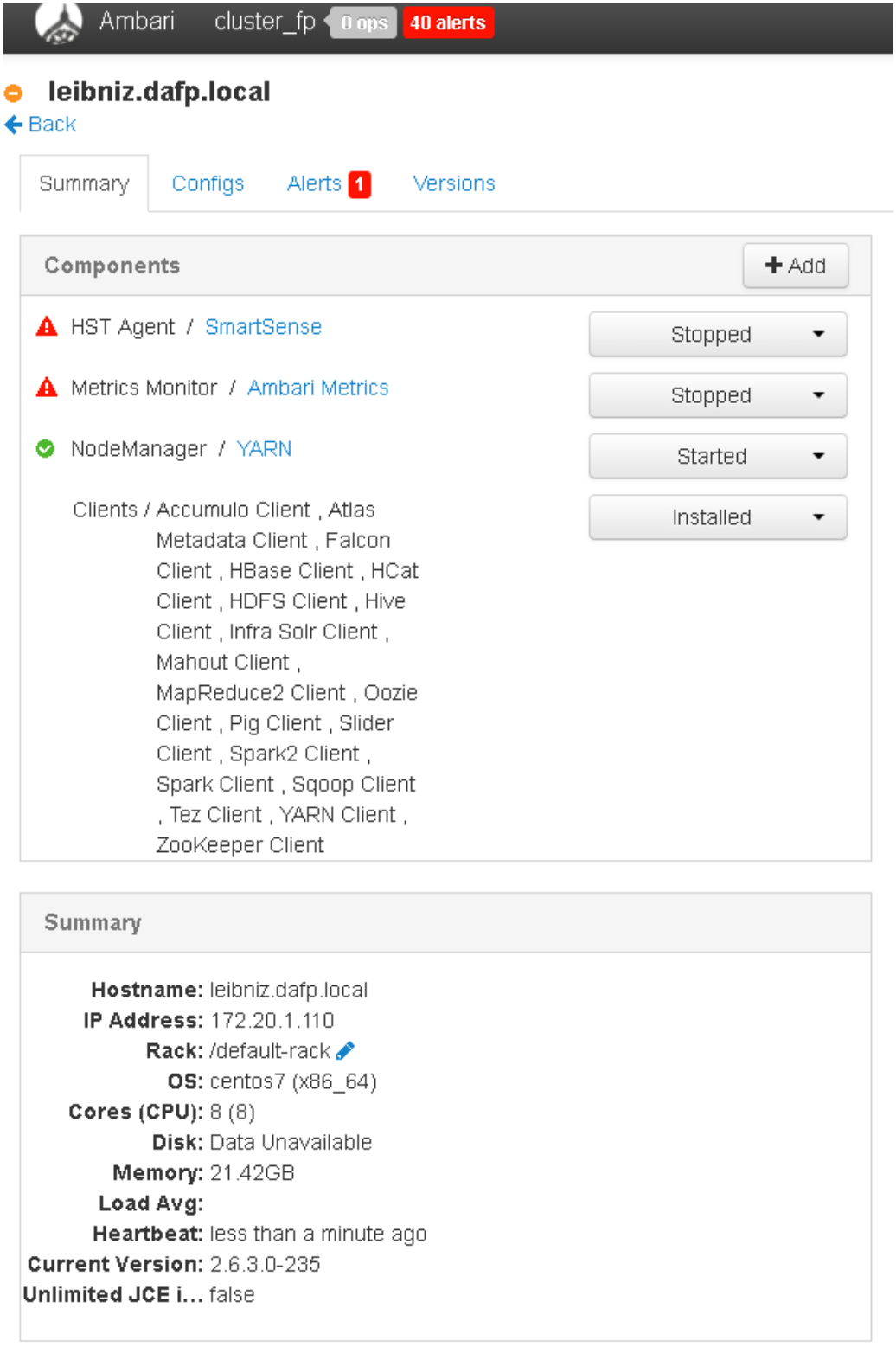
Anexo K: Detalle de la arquitectura tecnológica en clúster



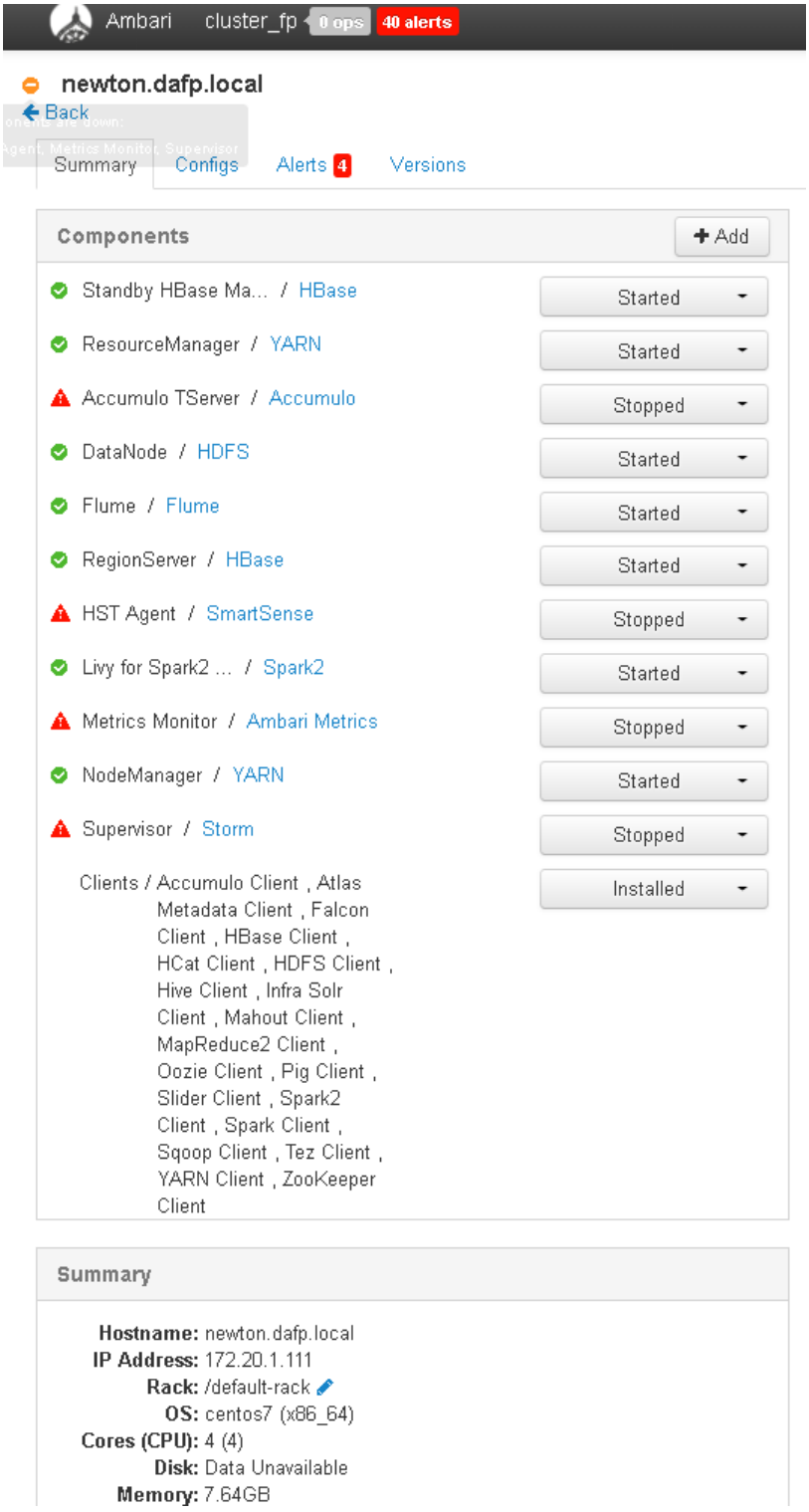
Anexo L: Conexión "yarn-cluster" y objetos en el cluster



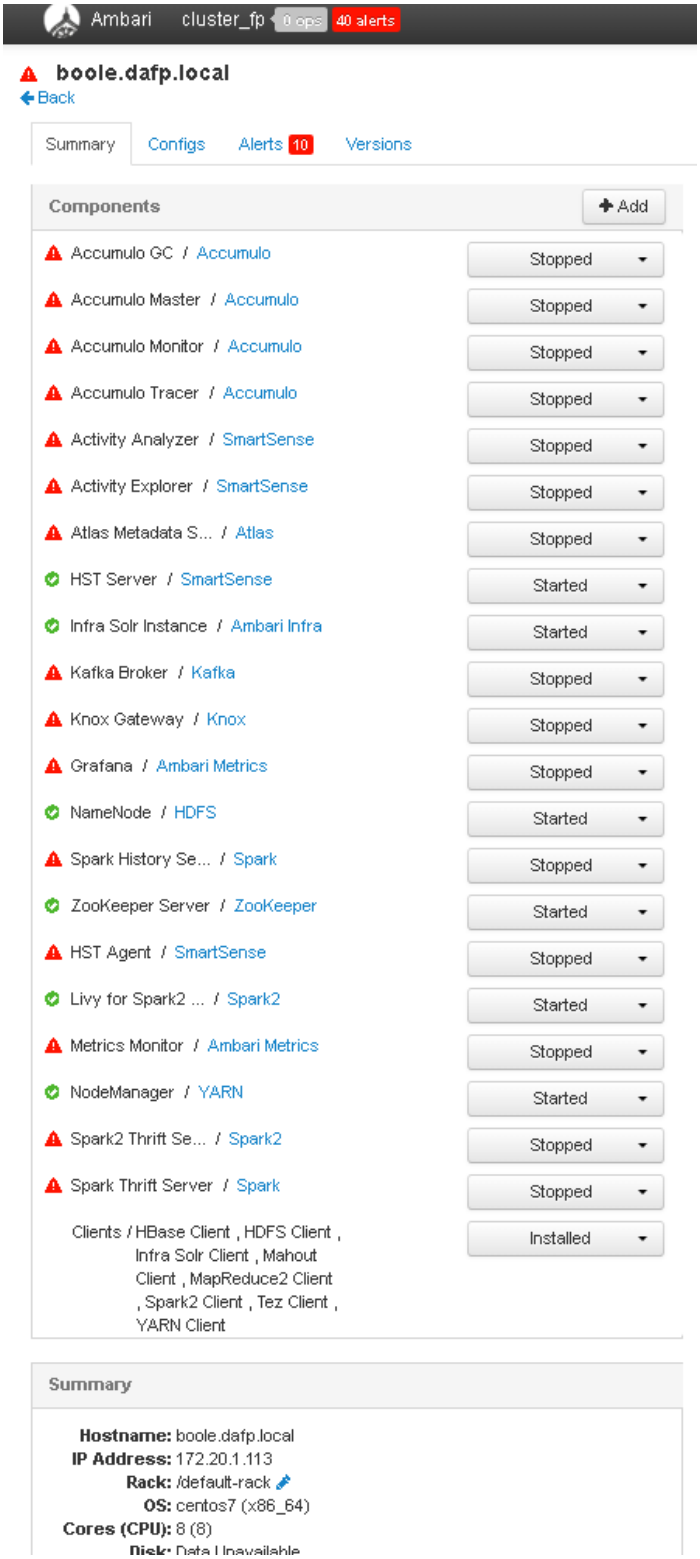
Anexo M: Miembro del clúster en AWS: NodeManager 3



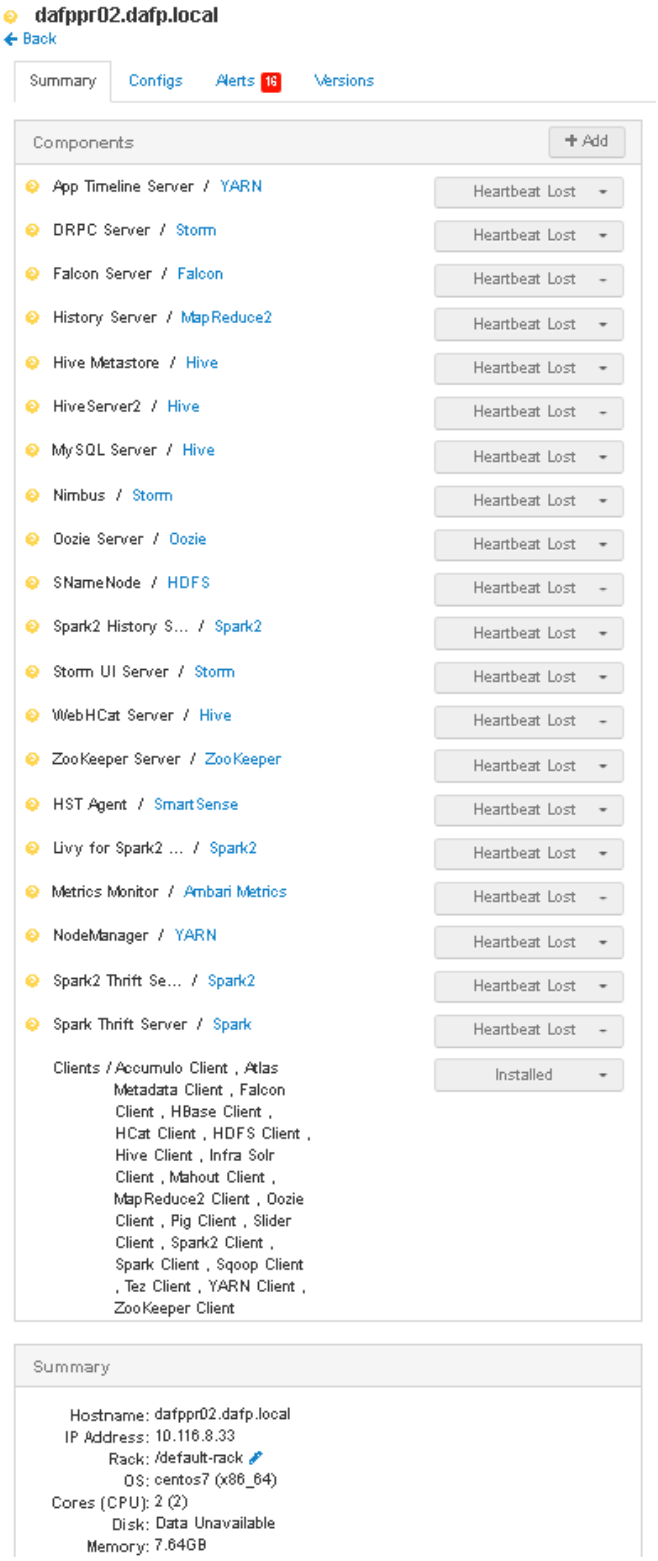
Anexo N: Miembro del clúster local: MasterNode



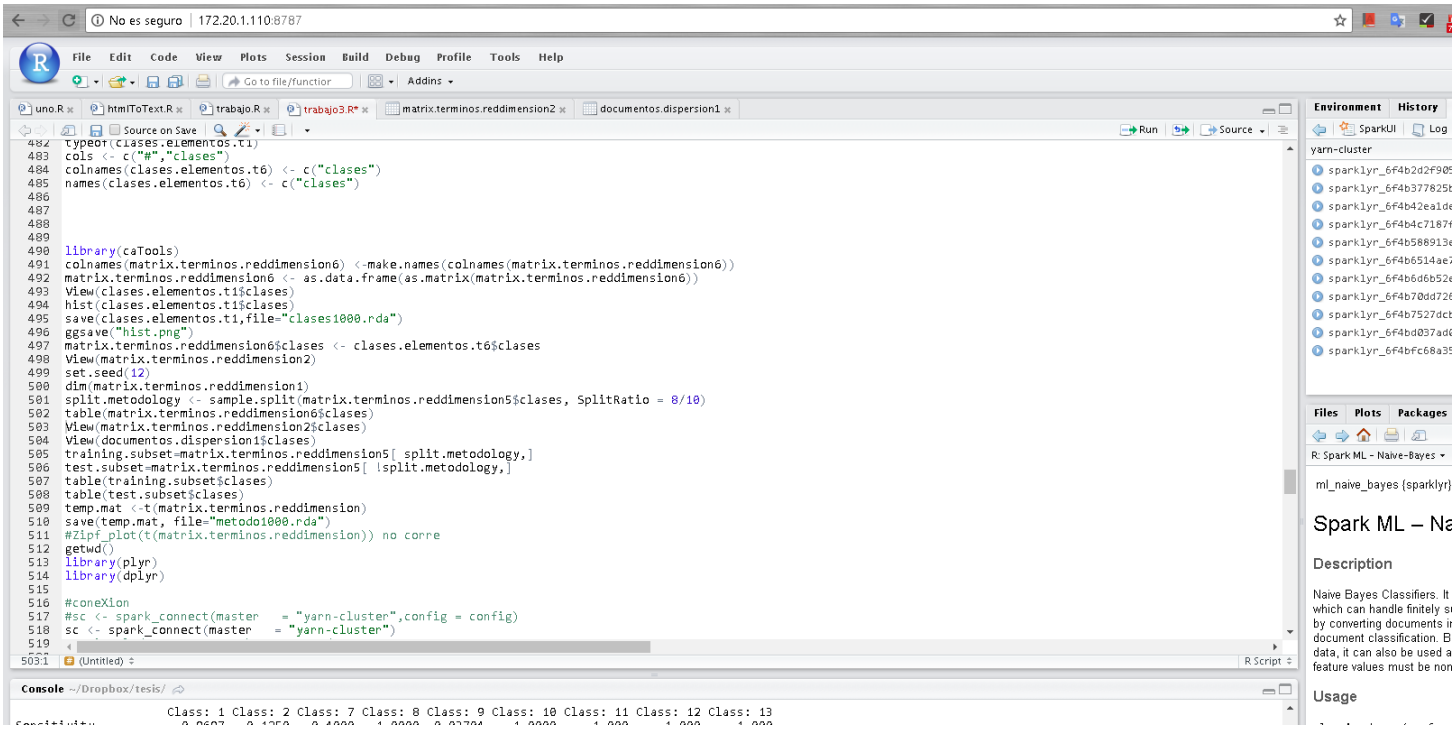
Anexo O: Miembro del clúster local: ResourceManager



Anexo P: Miembro del clúster local: NodeManager 1



Anexo Q: Miembro del clúster local: NodeManager 2



Anexo R: RStudio

```
> load(file= "~/tesis2/matrix.terminos.rda")
> matrix.terminos.trans <- t(matrix.terminos)
> dim(matrix.terminos.trans)
[1]  7206 54488
> matrix.terminos.reddimension <- removeSparseTerms(matrix.terminos.trans, sparse = .972)
> dim(matrix.terminos.reddimension)
[1] 7206  573
> matrix.terminos.reddimension1 <- (matrix.terminos.reddimension[1:1000,1:573])
> matrix.terminos.reddimension2 <- (matrix.terminos.reddimension[1001:2000,1:573])
> matrix.terminos.reddimension3 <- (matrix.terminos.reddimension[2001:3000,1:573])
> matrix.terminos.reddimension4 <- (matrix.terminos.reddimension[3001:4000,1:573])
> matrix.terminos.reddimension5 <- (matrix.terminos.reddimension[4001:5000,1:573])
> matrix.terminos.reddimension6 <- (matrix.terminos.reddimension[5001:6000,1:573])
> matrix.terminos.reddimension7 <- (matrix.terminos.reddimension[6001:7000,1:573])
> clases.elementos.t1 <-as.data.frame((clases[1:1000]))
> clases.elementos.t2 <-as.data.frame((clases[1001:2000]))
> clases.elementos.t3 <-as.data.frame((clases[2001:3000]))
> clases.elementos.t4 <-as.data.frame((clases[3001:4000]))
> clases.elementos.t5 <-as.data.frame((clases[4001:5000]))
> clases.elementos.t6 <-as.data.frame((clases[5001:6000]))
> clases.elementos.t7 <-as.data.frame((clases[6001:7000]))
> cols <- c("#","clases")
> colnames(clases.elementos.t2) <- c("clases")
> names(clases.elementos.t2) <- c("clases")
> colnames(matrix.terminos.reddimension2) <-make.names(colnames(matrix.terminos.reddimension2))
> matrix.terminos.reddimension2 <- as.data.frame(as.matrix(matrix.terminos.reddimension2))
> matrix.terminos.reddimension2$clases <- clases.elementos.t2$clases
> split.metodology <- sample.split(matrix.terminos.reddimension2$clases, SplitRatio = 8/10)
> table(matrix.terminos.reddimension2$clases,split.metodology)
    split.metodology
     FALSE TRUE
  1      6   23
  2      7   28
  7     11   44
  8     12   48
  9      7   26
  10    44  178
  11    40  160
  12    70  278
  13     4   14
> training.subset=matrix.terminos.reddimension2[ split.metodology,]
> test.subset=matrix.terminos.reddimension2[ !split.metodology,]
> training.subset.tbl <- copy_to(sc,training.subset,overwrite = TRUE)
> test.subset.tbl <- copy_to(sc,test.subset,overwrite = TRUE)
> src_tbls(sc)
[1] "sparklyr_6f4b377825b4" "sparklyr_6f4b4c7187fb" "sparklyr_6f4b6d6b52e5" "sparklyr_6f4bd037a
d0
> feature_colnames <- training.subset.tbl %>%
+   select(1:574)%>%
+   colnames()
> modelo  <- ml_linear_svc(sc)
> one_vs_rest <- ml_one_vs_rest(training.subset.tbl,classifier=modelo ,response = "clases", fea
tures = c(feature_colnames))
> responses <- test.subset.tbl  %>%
+   # Select the response column
+   select(clases) %>%
+   # Collect the results
+   collect() %>%
+   # Add in the predictions
+   mutate(
+     predicted_class = predict(
+       one_vs_rest,
+       test.subset.tbl
```

```
+     )
+   )
> head(responses)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1      2 2
2     12 12
3     11 12
4      9 11
5     12 12
6     12 12
> confusionMatrix(responses$predicted_class,responses$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   5  0  0  0  0  0  0  0  0
        2   1  6  0  0  0  0  0  1  0
        7   0  0 10  0  0  0  0  0  0
        8   0  0  0  4  0  1  1  0  0
        9   0  0  0  2  5  0  1  3  0
        10  0  0  1  4  1 29  4  4  0
        11  0  1  0  2  1  7 25  2  1
        12  0  0  0  0  0  7  9 58  3
        13  0  0  0  0  0  0  0  2  0

Overall Statistics

               Accuracy : 0.7065
                 95% CI : (0.6383, 0.7684)
    No Information Rate : 0.3483
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.62
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 12
Class: 13
Sensitivity           0.83333  0.85714  0.90909  0.33333  0.71429    0.6591    0.6250    0.8286
0.00000
Specificity           1.00000  0.98969  1.00000  0.98942  0.96907    0.9108    0.9130    0.8550
0.98985
Pos Pred Value        1.00000  0.75000  1.00000  0.66667  0.45455    0.6744    0.6410    0.7532
0.00000
Neg Pred Value        0.99490  0.99482  0.99476  0.95897  0.98947    0.9051    0.9074    0.9032
0.97990
Prevalence            0.02985  0.03483  0.05473  0.05970  0.03483    0.2189    0.1990    0.3483
0.01990
Detection Rate        0.02488  0.02985  0.04975  0.01990  0.02488    0.1443    0.1244    0.2886
0.00000
Detection Prevalence  0.02488  0.03980  0.04975  0.02985  0.05473    0.2139    0.1940    0.3831
0.00995
Balanced Accuracy     0.91667  0.92342  0.95455  0.66138  0.84168    0.7850    0.7690    0.8418
0.49492
Warning message:
In confusionMatrix.default(responses$predicted_class, responses$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
> modelo_bt <- ml_gbt_classifier(sc)
> one_vs_rest_bt <- ml_one_vs_rest(training.subset.tbl,classifier=modelo_bt,response = "clases"
, features = c(feature_colnames)

 > responses_bt <- test.subset.tbl  %>%
```

```
+   # Select the response column
+   select(clases) %>%
+   # Collect the results
+   collect() %>%
+   # Add in the predictions
+   mutate(
+     predicted_class = predict(
+       one_vs_rest_bt,
+       test.subset.tbl
+     )
+   )
> head(responses_bt)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1      2 2
2     12 12
3     11 11
4      9 9
5     12 12
6     12 12
> confusionMatrix(responses_bt$predicted_class,responses_bt$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   6  0  0  0  0  0  0  0  0
        2   0  7  0  0  0  0  0  0  0
        7   0  0 11  0  0  0  0  0  0
        8   0  0  0 12  0  0  0  0  0
        9   0  0  0  0  7  0  0  0  0
        10  0  0  0  0  0 44  0  0  0
        11  0  0  0  0  0  0 40  0  0
        12  0  0  0  0  0  0  0 70  0
        13  0  0  0  0  0  0  0  0  4

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9818, 1)
    No Information Rate : 0.3483
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 1
2 Class: 13
Sensitivity           1.00000  1.00000  1.00000   1.0000  1.00000    1.0000     1.000    1.000
0    1.0000
Specificity           1.00000  1.00000  1.00000   1.0000  1.00000    1.0000     1.000    1.000
0    1.0000
Pos Pred Value        1.00000  1.00000  1.00000   1.0000  1.00000    1.0000     1.000    1.000
0    1.0000
Neg Pred Value        1.00000  1.00000  1.00000   1.0000  1.00000    1.0000     1.000    1.000
0    1.0000
Prevalence            0.02985  0.03483  0.05473   0.0597  0.03483    0.2189     0.199    0.348
3    0.0199
Detection Rate        0.02985  0.03483  0.05473   0.0597  0.03483    0.2189     0.199    0.348
3    0.0199
Detection Prevalence  0.02985  0.03483  0.05473   0.0597  0.03483    0.2189     0.199    0.348
3    0.0199
```

```
Balanced Accuracy     1.00000  1.00000  1.00000   1.0000  1.00000    1.0000     1.000    1.000
0    1.0000
Warning message:
In confusionMatrix.default(responses_bt$predicted_class, responses_bt$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
 > modelo_nb=ml_naive_bayes(training.subset.tbl,response = "clases", features = c(feature_colna
 > responses_nb <- test.subset.tbl  %>%
 +   select(clases) %>%
 +   collect() %>%
 +   mutate(
 +     predicted_class = predict(
 +       modelo_nb,
 +       test.subset.tbl
 +     )
 +   )
 > head(responses_nb)
 # A tibble: 6 x 2
   clases predicted_class
    <int> <chr>
 1      2 2
 2     12 12
 3     11 12
 4      9 12
 5     12 12
 6     12 12
 > confusionMatrix(responses_nb$predicted_class,responses_nb$clases)
 Confusion Matrix and Statistics

           Reference
 Prediction  1  2  7  8  9 10 11 12 13
         1   1  0  0  1  0  0  0  0  0
         2   0  4  0  0  0  0  0  0  0
         7   1  1  9  1  0  2  2  5  0
         8   0  0  1  2  0  3  0  1  0
         9   0  0  0  1  6  1  0  0  2
         10  2  2  0  5  0 24  7  2  1
         11  1  0  0  1  0  2 17  2  0
         12  1  0  1  1  1 12 14 57  1
         13  0  0  0  0  0  0  0  3  0

 Overall Statistics

                Accuracy : 0.597
                  95% CI : (0.5257, 0.6654)
     No Information Rate : 0.3483
     P-Value [Acc > NIR] : 5.881e-13

                   Kappa : 0.4747
  Mcnemar's Test P-Value : NA

 Statistics by Class:

                      Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class: 1
 Sensitivity          0.166667  0.57143  0.81818  0.16667  0.85714    0.5455   0.42500    0.814
 Specificity          0.994872  1.00000  0.93684  0.97354  0.97938    0.8790   0.96273    0.763
 Pos Pred Value       0.500000  1.00000  0.42857  0.28571  0.60000    0.5581   0.73913    0.647
 Neg Pred Value       0.974874  0.98477  0.98889  0.94845  0.99476    0.8734   0.87079    0.885
 Prevalence           0.029851  0.03483  0.05473  0.05970  0.03483    0.2189   0.19900    0.348
 Detection Rate       0.004975  0.01990  0.04478  0.00995  0.02985    0.1194   0.08458    0.283
 Detection Prevalence 0.009950  0.01990  0.10448  0.03483  0.04975    0.2139   0.11443    0.437
 Balanced Accuracy    0.580769  0.78571  0.87751  0.57011  0.91826    0.7122   0.69387    0.788
 Warning message:
 In confusionMatrix.default(responses_nb$predicted_class, responses_nb$clases) :
   Levels are not in the same order for reference and data. Refactoring data to match.
```

```
> cols <- c("#","clases")
> colnames(clases.elementos.t3) <- c("clases")
> names(clases.elementos.t3) <- c("clases")
> colnames(matrix.terminos.reddimension3) <-make.names(colnames(matrix.terminos.reddimension3)
)
> matrix.terminos.reddimension3 <- as.data.frame(as.matrix(matrix.terminos.reddimension3))
> matrix.terminos.reddimension3$clases <- clases.elementos.t3$clases
> split.metodology <- sample.split(matrix.terminos.reddimension3$clases, SplitRatio = 8/10)
> table(matrix.terminos.reddimension3$clases,split.metodology)
    split.metodology
     FALSE TRUE
  1      8   33
  2      3   14
  7      8   32
  8     42  170
  9      3   10
  10    28  112
  11    52  209
  12    50  198
  13     6   2
> training.subset=matrix.terminos.reddimension3[ split.metodology,]
> test.subset=matrix.terminos.reddimension3[ !split.metodology,]
 > training.subset=matrix.terminos.reddimension3[ split.metodology,]
 > test.subset=matrix.terminos.reddimension3[ !split.metodology,]
 > training.subset.tbl <- copy_to(sc,training.subset,overwrite = TRUE)
 > test.subset.tbl <- copy_to(sc,test.subset,overwrite = TRUE)
 > src_tbls(sc)
 [1] "sparklyr_6f4b377825b4" "sparklyr_6f4b42ea1de"  "sparklyr_6f4b4c7187fb" "sparklyr_6f4b6d6
 b52e5" "sparklyr_6f4b70dd7263" "sparklyr_6f4bd037ad0"
  > feature_colnames <- training.subset.tbl %>%
  +   select(1:574)%>%
  +   # Get the column names
  +   colnames()
  > modelo  <- ml_linear_svc(sc)
  > one_vs_rest <- ml_one_vs_rest(training.subset.tbl,classifier=modelo ,response = "clases",
  features = c(feature_colnames))
  > responses <- test.subset.tbl  %>%
  +   # Select the response column
  +   select(clases) %>%
  +   # Collect the results
  +   collect() %>%
  +   # Add in the predictions
  +   mutate(
  +     predicted_class = predict(
  +       one_vs_rest,
  +       test.subset.tbl
  +     )
  +   )
   > head(responses)
   # A tibble: 6 x 2
     clases predicted_class
      <int> <chr>
   1     10 11
   2     11 12
   3     12 12
   4     12 12
   5      7 7
   6     12 11
   > confusionMatrix(responses$predicted_class,responses$clases)
   Confusion Matrix and Statistics
```

```
          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   8  0  0  0  0  0  0  0  0
        2   0  1  0  0  0  0  0  0  0
        7   0  0  7  1  0  0  0  0  0
        8   0  2  1 36  0  1  1  1  1
        9   0  0  0  0  2  1  0  0  0
        10  0  0  0  1  1 20  6  3  0
        11  0  0  0  1  0  3 40 10  0
        12  0  0  0  3  0  3  5 36  3
        13  0  0  0  0  0  0  0  0  2

Overall Statistics

               Accuracy : 0.76
                 95% CI : (0.6947, 0.8174)
    No Information Rate : 0.26
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6987
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class
: 12 Class: 13
Sensitivity              1.00   0.3333   0.8750   0.8571   0.6667    0.7143    0.7692    0.
7200    0.3333
Specificity              1.00   1.0000   0.9948   0.9557   0.9949    0.9360    0.9054    0.
9067    1.0000
Pos Pred Value           1.00   1.0000   0.8750   0.8372   0.6667    0.6452    0.7407    0.
7200    1.0000
Neg Pred Value           1.00   0.9899   0.9948   0.9618   0.9949    0.9527    0.9178    0.
9067    0.9798
Prevalence               0.04   0.0150   0.0400   0.2100   0.0150    0.1400    0.2600    0.
2500    0.0300
Detection Rate           0.04   0.0050   0.0350   0.1800   0.0100    0.1000    0.2000    0.
1800    0.0100
Detection Prevalence     0.04   0.0050   0.0400   0.2150   0.0150    0.1550    0.2700    0.
2500    0.0100
Balanced Accuracy        1.00   0.6667   0.9349   0.9064   0.8308    0.8252    0.8373    0.
8133    0.6667
Warning message:
In confusionMatrix.default(responses$predicted_class, responses$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
> modelo_bt <- ml_gbt_classifier(sc)
> one_vs_rest_bt <- ml_one_vs_rest(training.subset.tbl,classifier=modelo_bt,response = "cla
ses", features = c(feature_colnames))
> responses_bt <- test.subset.tbl  %>%
+   # Select the response column
+   select(clases) %>%
+   # Collect the results
+   collect() %>%
+   # Add in the predictions
+   mutate(
+     predicted_class = predict(
+       one_vs_rest_bt,
+       test.subset.tbl
+     )
+   )
> head(responses_bt)
# A tibble: 6 x 2
  clases predicted_class
```

```
   <int> <chr>
1     10 10
2     11 11
3     12 12
4     12 12
5      7 7
6     12 12
> confusionMatrix(responses_bt$predicted_class,responses_bt$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   8  0  0  0  0  0  0  0  0
        2   0  3  0  0  0  0  0  0  0
        7   0  0  7  0  0  0  0  0  0
        8   0  0  0 42  0  0  0  0  0
        9   0  0  1  0  3  0  0  0  0
        10  0  0  0  0  0 28  0  0  0
        11  0  0  0  0  0  0 52  0  0
        12  0  0  0  0  0  0  0 50  0
        13  0  0  0  0  0  0  0  0  6

Overall Statistics

               Accuracy : 0.995
                 95% CI : (0.9725, 0.9999)
    No Information Rate : 0.26
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9938
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class
: 12 Class: 13
Sensitivity              1.00    1.000   0.8750     1.00   1.0000      1.00      1.00
1.00      1.00
Specificity              1.00    1.000   1.0000     1.00   0.9949      1.00      1.00
1.00      1.00
Pos Pred Value           1.00    1.000   1.0000     1.00   0.7500      1.00      1.00
1.00      1.00
Neg Pred Value           1.00    1.000   0.9948     1.00   1.0000      1.00      1.00
1.00      1.00
Prevalence               0.04    0.015   0.0400     0.21   0.0150      0.14      0.26
0.25      0.03
Detection Rate           0.04    0.015   0.0350     0.21   0.0150      0.14      0.26
0.25      0.03
Detection Prevalence     0.04    0.015   0.0350     0.21   0.0200      0.14      0.26
0.25      0.03
Balanced Accuracy        1.00    1.000   0.9375     1.00   0.9975      1.00      1.00
1.00      1.00
Warning message:
In confusionMatrix.default(responses_bt$predicted_class, responses_bt$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
> modelo_nb=ml_naive_bayes(training.subset.tbl,response = "clases", features = c(feature_co
lnames))
> responses_nb <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       modelo_nb,
```

```
+       test.subset.tbl
+     )
+   )
> head(responses_nb)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     10 11
2     11 10
3     12 12
4     12 12
5      7 7
6     12 12
> confusionMatrix(responses_nb$predicted_class,responses_nb$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1   5  0  0  0  0  0  0  0  0
        2   0  1  0  0  0  0  0  0  0
        7   0  0  7  2  0  0  4  2  0
        8   0  1  0 29  0  3  0  1  2
        9   0  0  0  0  2  2  0  0  0
        10  1  0  0  2  1 15  7  1  0
        11  0  0  1  7  0  7 37  2  0
        12  2  1  0  2  0  1  3 44  3
        13  0  0  0  0  0  0  1  0  1

Overall Statistics

               Accuracy : 0.705
                 95% CI : (0.6366, 0.7672)
    No Information Rate : 0.26
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6305
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class
: 12 Class: 13
Sensitivity            0.6250   0.3333   0.8750   0.6905   0.6667    0.5357    0.7115    0.
8800    0.1667
Specificity            1.0000   1.0000   0.9583   0.9557   0.9898    0.9302    0.8851    0.
9200    0.9948
Pos Pred Value         1.0000   1.0000   0.4667   0.8056   0.5000    0.5556    0.6852    0.
7857    0.5000
Neg Pred Value         0.9846   0.9899   0.9946   0.9207   0.9949    0.9249    0.8973    0.
9583    0.9747
Prevalence             0.0400   0.0150   0.0400   0.2100   0.0150    0.1400    0.2600    0.
2500    0.0300
Detection Rate         0.0250   0.0050   0.0350   0.1450   0.0100    0.0750    0.1850    0.
2200    0.0050
Detection Prevalence   0.0250   0.0050   0.0750   0.1800   0.0200    0.1350    0.2700    0.
2800    0.0100
Balanced Accuracy      0.8125   0.6667   0.9167   0.8231   0.8283    0.7330    0.7983    0.
9000    0.5808
Warning message:
In confusionMatrix.default(responses_nb$predicted_class, responses_nb$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
> cols <- c("#","clases")
> colnames(clases.elementos.t4) <- c("clases")
```

```
> names(clases.elementos.t4) <- c("clases")
> colnames(matrix.terminos.reddimension4) <-make.names(colnames(matrix.terminos.reddimensio
n4))
> matrix.terminos.reddimension4 <- as.data.frame(as.matrix(matrix.terminos.reddimension4))
> matrix.terminos.reddimension4$clases <- clases.elementos.t4$clases
> split.metodology <- sample.split(matrix.terminos.reddimension4$clases, SplitRatio = 8/10)
> table(matrix.terminos.reddimension4$clases,split.metodology)
    split.metodology
     FALSE TRUE
  1      4   15
  2      1    6
  7      6   26
  8     82  327
  9      0    1
  10    23   93
  11    27  109
  12    50  202
  13     6   22
> training.subset=matrix.terminos.reddimension4[ split.metodology,]
> test.subset=matrix.terminos.reddimension4[ !split.metodology,]
> training.subset.tbl <- copy_to(sc,training.subset,overwrite = TRUE)
> test.subset.tbl <- copy_to(sc,test.subset,overwrite = TRUE)
> src_tbls(sc)
[1] "sparklyr_6f4b377825b4" "sparklyr_6f4b42ea1de"  "sparklyr_6f4b4c7187fb" "sparklyr_6f4b5
88913e8" "sparklyr_6f4b6514ae7e" "sparklyr_6f4b6d6b52e5" "sparklyr_6f4b70dd7263"
[8] "sparklyr_6f4bd037ad0"
> feature_colnames <- training.subset.tbl %>%
+   select(1:574)%>%
+   # Get the column names
+   colnames(


> modelo  <- ml_linear_svc(sc)
> one_vs_rest <- ml_one_vs_rest(training.subset.tbl,classifier=modelo ,response = "clases",
features = c(feature_colnames))
> responses <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       one_vs_rest,
+       test.subset.tbl
+     )
+   )
> head(responses)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     11 11
2     10 10
3      8 8
4      8 8
5     11 7
6     12 12
> confusionMatrix(responses$predicted_class,responses$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8 10 11 12 13
        1   3  0  0  0  0  0  0  0
        2   1  1  0  0  0  0  0  0
        7   0  0  6  1  1  1  0  0
        8   0  0  0 79  1  2  3  0
```

```
         10  0  0  0  1 14  1  2  0
         11  0  0  0  0  4 20  1  1
         12  0  0  0  1  3  2 42  1
         13  0  0  0  0  0  1  2  4

Overall Statistics

               Accuracy : 0.8492
                 95% CI : (0.7918, 0.8959)
    No Information Rate : 0.4121
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7938
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 10 Class: 11 Class: 12 Clas
s: 13
Sensitivity           0.75000 1.000000  1.00000   0.9634   0.60870    0.7407    0.8400   0.
66667
Specificity           1.00000 0.994949  0.98446   0.9487   0.97727    0.9651    0.9530   0.
98446
Pos Pred Value        1.00000 0.500000  0.66667   0.9294   0.77778    0.7692    0.8571   0.
57143
Neg Pred Value        0.99490 1.000000  1.00000   0.9737   0.95028    0.9595    0.9467   0.
98958
Prevalence            0.02010 0.005025  0.03015   0.4121   0.11558    0.1357    0.2513   0.
03015
Detection Rate        0.01508 0.005025  0.03015   0.3970   0.07035    0.1005    0.2111   0.
02010
Detection Prevalence  0.01508 0.010050  0.04523   0.4271   0.09045    0.1307    0.2462   0.
03518
Balanced Accuracy     0.87500 0.997475  0.99223   0.9561   0.79298    0.8529    0.8965   0.
82556
Warning message:
In confusionMatrix.default(responses$predicted_class, responses$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match


> modelo_bt <- ml_gbt_classifier(sc)
> one_vs_rest_bt <- ml_one_vs_rest(training.subset.tbl,classifier=modelo_bt,response = "cla
ses", features = c(feature_colnames))
> responses_bt <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       one_vs_rest_bt,
+       test.subset.tbl
+     )
+   )
> head(responses_bt)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     11 11
2     10 10
3      8 8
4      8 8
5     11 11
6     12 12
> confusionMatrix(responses_bt$predicted_class,responses_bt$clases)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8 10 11 12 13
        1   4  0  0  0  0  0  0  0
        2   0  1  0  0  0  0  0  0
        7   0  0  6  0  0  0  0  0
        8   0  0  0 82  0  0  0  0
        10  0  0  0  0 23  0  0  0
        11  0  0  0  0  0 27  0  0
        12  0  0  0  0  0  0 50  0
        13  0  0  0  0  0  0  0  6

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9816, 1)
    No Information Rate : 0.4121
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 10 Class: 11 Class: 12 Clas
s: 13
Sensitivity            1.0000 1.000000  1.00000   1.0000    1.0000    1.0000    1.0000   1.
00000
Specificity            1.0000 1.000000  1.00000   1.0000    1.0000    1.0000    1.0000   1.
00000
Pos Pred Value         1.0000 1.000000  1.00000   1.0000    1.0000    1.0000    1.0000   1.
00000
Neg Pred Value         1.0000 1.000000  1.00000   1.0000    1.0000    1.0000    1.0000   1.
00000
Prevalence             0.0201 0.005025  0.03015   0.4121    0.1156    0.1357    0.2513   0.
03015
Detection Rate         0.0201 0.005025  0.03015   0.4121    0.1156    0.1357    0.2513   0.
03015
Detection Prevalence   0.0201 0.005025  0.03015   0.4121    0.1156    0.1357    0.2513   0.
03015
Balanced Accuracy      1.0000 1.000000  1.00000   1.0000    1.0000    1.0000    1.0000   1.
00000
Warning message:
In confusionMatrix.default(responses_bt$predicted_class, responses_bt$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.

> modelo_nb=ml_naive_bayes(training.subset.tbl,response = "clases", features = c(feature_co
lnames))
> responses_nb <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       modelo_nb,
+       test.subset.tbl
+     )
+   )
> head(responses_nb)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1     11 11
```

```
2     10 12
3      8 8
4      8 8
5     11 7
6     12 10
> confusionMatrix(responses_nb$predicted_class,responses_nb$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8 10 11 12 13
        1   2  0  0  0  0  0  0  0
        2   1  1  0  0  0  0  0  0
        7   0  0  6  1  1  3  1  0
        8   0  0  0 76  1  2  0  0
        10  0  0  0  2 10  4  6  0
        11  0  0  0  2  3 15  2  1
        12  1  0  0  1  8  3 39  2
        13  0  0  0  0  0  0  2  3

Overall Statistics

               Accuracy : 0.7638
                 95% CI : (0.6986, 0.821)
    No Information Rate : 0.4121
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6795
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 10 Class: 11 Class: 12 Clas
s: 13
Sensitivity           0.50000 1.000000  1.00000   0.9268   0.43478   0.55556    0.7800   0.
50000
Specificity           1.00000 0.994949  0.96891   0.9744   0.93182   0.95349    0.8993   0.
98964
Pos Pred Value        1.00000 0.500000  0.50000   0.9620   0.45455   0.65217    0.7222   0.
60000
Neg Pred Value        0.98985 1.000000  1.00000   0.9500   0.92655   0.93182    0.9241   0.
98454
Prevalence            0.02010 0.005025  0.03015   0.4121   0.11558   0.13568    0.2513   0.
03015
Detection Rate        0.01005 0.005025  0.03015   0.3819   0.05025   0.07538    0.1960   0.
01508
Detection Prevalence  0.01005 0.010050  0.06030   0.3970   0.11055   0.11558    0.2714   0.
02513
Balanced Accuracy     0.75000 0.997475  0.98446   0.9506   0.68330   0.75452    0.8397   0.
74482
Warning message:
In confusionMatrix.default(responses_nb$predicted_class, responses_nb$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
> cols <- c("#","clases")
> colnames(clases.elementos.t5) <- c("clases")
> names(clases.elementos.t5) <- c("clases")
> colnames(matrix.terminos.reddimension5) <-make.names(colnames(matrix.terminos.reddimensio
n5))
> matrix.terminos.reddimension5 <- as.data.frame(as.matrix(matrix.terminos.reddimension5))
> matrix.terminos.reddimension5$clases <- clases.elementos.t5$clases
> split.metodology <- sample.split(matrix.terminos.reddimension5$clases, SplitRatio = 8/10)
> table(matrix.terminos.reddimension5$clases,split.metodology)
    split.metodology
     FALSE TRUE
```

```
  1     14   58
  2      1    4
  7      2    6
  8     82  330
  9      1    4
  10    24   96
  11    21   84
  12    46  182
  13     9   36
> training.subset=matrix.terminos.reddimension5[ split.metodology,]
> test.subset=matrix.terminos.reddimension5[ !split.metodology,]
> training.subset.tbl <- copy_to(sc,training.subset,overwrite = TRUE)
> test.subset.tbl <- copy_to(sc,test.subset,overwrite = TRUE)
> src_tbls(sc)
 [1] "sparklyr_6f4b2d2f9053" "sparklyr_6f4b377825b4" "sparklyr_6f4b42ea1de"  "sparklyr_6f4b
4c7187fb" "sparklyr_6f4b588913e8" "sparklyr_6f4b6514ae7e"
 [7] "sparklyr_6f4b6d6b52e5" "sparklyr_6f4b70dd7263" "sparklyr_6f4bd037ad0"  "sparklyr_6f4b
fc68a35"
> dim(training.subset.tbl)
[1]  NA 574
> feature_colnames <- training.subset.tbl %>%
+   select(1:574)%>%
+   # Get the column names
+   colnames()
> modelo  <- ml_linear_svc(sc)
> one_vs_rest <- ml_one_vs_rest(training.subset.tbl,classifier=modelo ,response = "clases",
features = c(feature_colnames))
> responses <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       one_vs_rest,
+       test.subset.tbl
+     )
+   )
> head(responses)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1      8 8
2     12 12
3     11 11
4     12 12
5      8 8
6     12 12
> confusionMatrix(responses$predicted_class,responses$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1  13  0  0  0  1  0  1  0  0
        2   1  0  0  0  0  0  0  0  0
        7   0  0  0  0  0  1  0  0  0
        8   0  1  0 76  0  2  1  0  0
        9   0  0  0  0  0  0  0  1  0
        10  0  0  1  3  0 19  4  1  4
        11  0  0  0  1  0  0 11  1  2
        12  0  0  1  1  0  2  2 43  1
        13  0  0  0  1  0  0  2  0  2

Overall Statistics
```

```
               Accuracy : 0.82
                 95% CI : (0.7596, 0.8706)
    No Information Rate : 0.41
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7584
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class
: 12 Class: 13
Sensitivity            0.9286   0.0000   0.0000   0.9268   0.0000    0.7917    0.5238    0.
9348    0.2222
Specificity            0.9892   0.9950   0.9949   0.9661   0.9950    0.9261    0.9777    0.
9545    0.9843
Pos Pred Value         0.8667   0.0000   0.0000   0.9500   0.0000    0.5938    0.7333    0.
8600    0.4000
Neg Pred Value         0.9946   0.9950   0.9899   0.9500   0.9950    0.9702    0.9459    0.
9800    0.9641
Prevalence             0.0700   0.0050   0.0100   0.4100   0.0050    0.1200    0.1050    0.
2300    0.0450
Detection Rate         0.0650   0.0000   0.0000   0.3800   0.0000    0.0950    0.0550    0.
2150    0.0100
Detection Prevalence   0.0750   0.0050   0.0050   0.4000   0.0050    0.1600    0.0750    0.
2500    0.0250
Balanced Accuracy      0.9589   0.4975   0.4975   0.9465   0.4975    0.8589    0.7507    0.
9447    0.6033
Warning message:
In confusionMatrix.default(responses$predicted_class, responses$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match
> modelo_bt <- ml_gbt_classifier(sc)
> one_vs_rest_bt <- ml_one_vs_rest(training.subset.tbl,classifier=modelo_bt,response = "cla
ses", features = c(feature_colnames))
> responses_bt <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       one_vs_rest_bt,
+       test.subset.tbl
+     )
+   )
> head(responses_bt)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1      8 8
2     12 12
3     11 11
4     12 12
5      8 8
6     12 12
> confusionMatrix(responses_bt$predicted_class,responses_bt$clases)
Confusion Matrix and Statistics

          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1  14  0  0  0  0  0  0  0  0
        2   0  0  0  0  0  0  0  0  0
        7   0  0  0  0  0  0  0  0  0
        8   0  1  1 82  1  0  0  0  0
        9   0  0  1  0  0  1  0  0  0
```

```
         10  0  0  0  0  0 23  0  0  0
         11  0  0  0  0  0  0 21  0  0
         12  0  0  0  0  0  0  0 46  0
         13  0  0  0  0  0  0  0  0  9

Overall Statistics

               Accuracy : 0.975
                 95% CI : (0.9426, 0.9918)
    No Information Rate : 0.41
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9663
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class
: 12 Class: 13
Sensitivity             1.00    0.000     0.00   1.0000   0.0000    0.9583     1.000
1.00     1.000
Specificity             1.00    1.000     1.00   0.9746   0.9899    1.0000     1.000
1.00     1.000
Pos Pred Value          1.00      NaN      NaN   0.9647   0.0000    1.0000     1.000
1.00     1.000
Neg Pred Value          1.00    0.995     0.99   1.0000   0.9949    0.9944     1.000
1.00     1.000
Prevalence              0.07    0.005     0.01   0.4100   0.0050    0.1200     0.105
0.23     0.045
Detection Rate          0.07    0.000     0.00   0.4100   0.0000    0.1150     0.105
0.23     0.045
Detection Prevalence    0.07    0.000     0.00   0.4250   0.0100    0.1150     0.105
0.23     0.045
Balanced Accuracy       1.00    0.500     0.50   0.9873   0.4950    0.9792     1.000
1.00     1.000
Warning messages:
1: In levels(reference) != levels(data) :
  longitud de objeto mayor no es múltiplo de la longitud de uno menor
2: In confusionMatrix.default(responses_bt$predicted_class, responses_bt$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match.
> modelo_nb=ml_naive_bayes(training.subset.tbl,response = "clases", features = c(feature_co
lnames))
> responses_nb <- test.subset.tbl  %>%
+   select(clases) %>%
+   collect() %>%
+   mutate(
+     predicted_class = predict(
+       modelo_nb,
+       test.subset.tbl
+     )
+   )
> head(responses_nb)
# A tibble: 6 x 2
  clases predicted_class
   <int> <chr>
1      8 8
2     12 12
3     11 11
4     12 12
5      8 8
6     12 12
> confusionMatrix(responses_nb$predicted_class,responses_nb$clases)
Confusion Matrix and Statistics
```

```
          Reference
Prediction  1  2  7  8  9 10 11 12 13
        1  10  0  0  1  1  2  1  0  0
        2   1  0  0  0  0  0  0  0  0
        7   1  0  1  1  0  0  1  0  0
        8   0  0  0 71  0  4  1  0  0
        9   0  1  0  1  0  0  0  0  0
        10  1  0  0  2  0 13  2  6  1
        11  0  0  0  3  0  1 12  0  0
        12  1  0  1  1  0  4  2 39  3
        13  0  0  0  2  0  0  2  1  5

Overall Statistics

               Accuracy : 0.755
                 95% CI : (0.6894, 0.8129)
    No Information Rate : 0.41
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6752
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 7 Class: 8 Class: 9 Class: 10 Class: 11 Class
: 12 Class: 13
Sensitivity            0.7143   0.0000   0.5000   0.8659   0.0000    0.5417    0.5714    0.
8478    0.5556
Specificity            0.9731   0.9950   0.9848   0.9576   0.9899    0.9318    0.9777    0.
9221    0.9738
Pos Pred Value         0.6667   0.0000   0.2500   0.9342   0.0000    0.5200    0.7500    0.
7647    0.5000
Neg Pred Value         0.9784   0.9950   0.9949   0.9113   0.9949    0.9371    0.9511    0.
9530    0.9789
Prevalence             0.0700   0.0050   0.0100   0.4100   0.0050    0.1200    0.1050    0.
2300    0.0450
Detection Rate         0.0500   0.0000   0.0050   0.3550   0.0000    0.0650    0.0600    0.
1950    0.0250
Detection Prevalence   0.0750   0.0050   0.0200   0.3800   0.0100    0.1250    0.0800    0.
2550    0.0500
Balanced Accuracy      0.8437   0.4975   0.7424   0.9117   0.4950    0.7367    0.7745    0.
8850    0.7647
Warning message:
In confusionMatrix.default(responses_nb$predicted_class, responses_nb$clases) :
  Levels are not in the same order for reference and data. Refactoring data to match
```

Anexo S: Script del ciclo de experimentos en R