Maestría en Ciencias Actuariales

Exploración de Modelos para Determinar el Incurrido en Seguros de Autos

Camilo Esteban Posada Aguilar

Bogotá D.C. 20 de Mayo del 2022

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO
VIGILADA MINEDUCACIÓN
UNIVERSIDAD

Maestría en Ciencias Actuariales

Trabajo de Grado para optar por el Título de Magíster en Ciencias Actuariales

Catalina Lozano Murcia

Directora

July Andrea Salazar Gómez

Jurado 1

Wilmer Darío Pineda Ríos

Jurado 2

Bogotá D.C., 20 de Mayo del 2022

El Trabajo de grado de maestría titulada “Exploración de Modelos para Determinar el Incurrido en Seguros de Autos”, presentada por Camilo Esteban Posada Aguilar cumple con los requisitos establecidos para optar al título de Magíster en Ciencias Actuariales

Directora

Catalina Lozano Murcia

Jurado 1

July Andrea Salazar Gomez

Jurado 2

Wilmer Darío Pineda Ríos

Bogotá D.C. 27 de Mayo del 2022

Dedico este trabajo a Dios, mi mamá, mi abuela y mi hermano por ayudarme a llegar hasta acá.

Lo dedico también a mis amigos.

Agradezco a la profesora Catalina Lozano por su tutoría en el trabajo, a la compañía de seguros en la cual trabajo y a la Escuela Colombiana de Ingeniería Julio Garavito.

Resumen

En los seguros generales es particularmente complicado hallar la forma de pronosticar el incurrido del ramo de autos en los amparos de pérdida y hurto parcial. Pronosticar esta cifra es vital para la compañía puesto que se lograría entender el comportamiento del negocio a través de las reservas, la siniestralidad, el índice combinado y la tarifación en caso de que el portafolio sea muy pequeño para realizar un GLM.

El propósito de este trabajo es proveer una metodología para pronosticar el incurrido, de tal forma que la reserva de siniestros avisados esté mucho más alineada con el valor de las autopartes en el mercado. Para ello se propone entrenar algoritmos de series de tiempo cuyo *output* sea un índice de precios de las piezas creado a partir del portafolio de la compañía y su *input* sean índices económicos y de desempeño del país.

Palabras clave: SVR, LSTM, XGBoost, ARIMA, VAR, Reserva de Siniestros Avisados, TRM, IPC, seguro de autos, repuestos.

Abstract

In Property and Casualty Insurance is particularly tough to find a way to forecast the reported losses from the car's insurance coverages of partial and thief loss. Forecast this figure is key for the company, because the business understanding will increase through the reserves, loss ratio, combined ratio and the pricing component if the company has a small portfolio to perform a GLM.

The main purpose of the work is provide a methodology for forecasting the reported losses, in such a way that the case reserve will be in line with the auto part's market. These algorithms were trained from time series whose output is a price index from cars part's market created through the company's portfolio and whose outputs would be economic and country's performance indexes.

Key words: SVR, LSTM, XGBoost, ARIMA, VAR , Case Reserve, TRM, IPC, car insurance, car parts.

Contenido

Índice de Tablas

Índice de Ilustraciones

Índice de Ecuaciones

1. Introducción

Hace más de 30 años (Cummins & Griepentrog, 1985) crearon una metodología para pronosticar el incurrido y el *loss ratio* (ver sección 3.2) tratando de estimar un factor de severidad usando modelos econométricos clásicos, modelos ARIMA y tomando como *input* el IPC y el valor de la mano de obra. Con el avance de la tecnología nuevos y más poderosos modelos han surgido y estos se pueden utilizar para hallar el factor de siniestralidad por lo que el presente trabajo se puede ver como la continuación del de (Cummins & Griepentrog, 1985), con la diferencia de que se construyó un índice de precios propios basados en una base de datos con más de 100 mil siniestros. La variación porcentual del índice hará las veces de factor de severidad, los *inputs* ahora son varias series de tiempo del contexto colombiano y entran en escena nuevos modelos más robustos como el VAR, LSTM, XGBoost y SVR.

El problema que motiva el presente trabajo recae en que si el incurrido esperado no está correctamente estimado se puede llegar a cometer errores de suscripción que afecten gravemente a la compañía, tener reservas insuficientes o caer en una mala tarifación en el caso de una compañía pequeña sin la suficiente cantidad de siniestros. El horizonte temporal, los datos y los precios del mercado de autopartes van del primero de enero del 2018 al 31 de diciembre del 2021.

El objetivo general de este trabajo, consiste en "proponer una nueva metodología para el cálculo del incurrido esperado y la reserva de siniestros avisados de los amparos de perdida y hurto parcial a través de la elección del modelo con mejor capacidad predictiva". La hipótesis de la que se parte es si un índice de precios creado a partir del portafolio de la empresa es eficaz para formular una metodología para el incurrido esperado.

La implementación de metodologías y modelos para determinar el incurrido y particularmente usarlo en la reserva de siniestros avisados es muy importante para conocer las proyecciones en siniestralidad, dar unas reservas más precisas y darle una mejor experiencia al cliente al tener un mejor estimador del daño que tiene su vehículo.

A lo largo de este trabajo se tomará la frecuencia como un input dado por la compañía, el valor objeto de estudio será el costo esperado del siniestro; en adelante, aunque se hable de incurrido se entenderá que el objeto modelado dentro del incurrido será el costo esperado de las auto partes de un siniestro según sus características debido a la falta de datos de la mano de obra en los talleres. Por último, en lo que respecta a este trabajo el valor de significancia estadístico será de 5%.

Este trabajo consta de 5 secciones que empiezan por la sección 2., en la que se muestran los objetivos; en la sección 3. se presenta el marco teórico, este se divide principalmente en una explicación de los términos de seguros que se van a considerar, la metodología con la que se construyó el índice de precios, cuáles son las hipótesis y teoría tras el funcionamiento de los modelos que se van a implementar y por último la definición de la medida de error con la que se evaluarán los modelos; en la sección 4. se muestra cómo se implementó la metodología CRISP-DM en el desarrollo de este proyecto; la sección 5. muestra la construcción, parámetros y resultados de los modelos, así como la metodología para calcular el incurrido esperado y la reserva de siniestros avisados. Finalmente, la sección 6. Muestra las conclusiones y recomendaciones.

2. Objetivos

2.1. Objetivo General

Proponer una nueva metodología para el cálculo del incurrido esperado y la reserva de siniestros avisados en los amparos de pérdida y hurto parcial a través de la elección del modelo con mejor capacidad predictiva.

2.2. Objetivos específicos

- Programar algoritmos de inteligencia artificial que predigan el incurrido del ramo de autos en los amparos de pérdida y hurto parcial.
- Comparar el desempeño predictivo de cada uno de los algoritmos para identificar el que tenga mejor capacidad.
- Proponer el uso de una metodología para el incurrido esperado y la reserva de siniestros avisados.

2. Marco Teórico

En esta sección se hablará de los principales conceptos tratados a lo largo del trabajo y que permitirán lograr los objetivos. La sección 3.1. dará los conceptos de seguros necesarios para construir el índice de precios y dar un entendimiento del contexto de los seguros. La sección 3.2. define el objeto β_s que se quiere construir. La sección 3.3. mostrará cómo se construyó el índice mensual que creará la serie de tiempo que se quiere pronosticar y permitirá estimar β_s. Las secciones 3.4. a la 3.8. expondrán los modelos que se usaron en el desarrollo del trabajo, sus hipótesis, parámetros y pruebas estadísticas para su medición y por último la sección 3.9 define el error que se usará para comparar los modelos.

3.1. Pérdidas

Según (Werner & Claudine, 2016) una pérdida en el mundo de los seguros, está definida como la cantidad de dinero pagada a un beneficiario bajo los términos y condiciones de la póliza de seguros.

El valor pagado es el dinero que se le debe transferir al beneficiario de la póliza en compensación por un siniestro, sin embargo, existe otro valor que debe ser considerado. Cuando se le reporta a una aseguradora un siniestro y se espera que el pago se haga en el futuro, se debe establecer una reserva de siniestros avisados, la cual es un estimador de la cantidad de dinero que se pagará. Esta reserva, es monitoreada y ajustada a medida que el siniestro se va desarrollando o se conoce información más precisa sobre los daños o se excluyen valores ya pagados. Del valor pagado y la reserva de siniestros avisados se desprende el incurrido que se define como:

Ecuación 1

$$\text{Incurrido} = \text{Reserva de siniestros avisados} + \text{Valor Pagado}$$

El incurrido será tomado como un estimador de la pérdida hasta que se cierre el siniestro (Werner & Claudine, 2016)

3.2. Incurrido esperado

Con el fin de estimar el desempeño del incurrido a futuro, analizar el *loss ratio* estimado, etc. Es necesario definir el incurrido esperado como (Cummins & Griepentrog, 1985):

Ecuación 2: *Incurrido Esperado*

$$L_a = L_0\left(1+\beta_f\right)^k(1+\beta_s)^{\mathrm{f}}$$

Donde:

L_a: Incurrido esperado

L_0 : Incurrido observado durante el periodo de estudio

β_s : La tasa pronosticada de cambio de la severidad

β_f : La tasa pronosticada de cambio de la frecuencia

k: Número de periodos al futuro

La tasa de severidad es más usada en análisis y generalmente es mayor que la tasa de la frecuencia, por ende, se omitirá la tasa de frecuencia. La tasa de severidad es más cercana a índices económicos y debería poderse modelar a través de modelos econométricos. En un ramo como autos, donde el valor pagado y los daños dependen de las autopartes y la mano de obra de las reparaciones, es razonable pensar que estén sujetos a variaciones de un índice económico subyacente como el IPC.

3.3. Cálculo del IPC

El índice de precios al consumidor está diseñado para medir cambios en los precios, a través de la construcción de una gran canasta familiar (cesta básica en el seguro en el contexto de este trabajo) que contiene los productos más consumidos por los hogares o para el objetivo de este trabajo, las piezas más siniestradas. Si el precio de los productos cambia, el valor de la canasta también y por ende el índice que debería reflejar el movimiento (Office for National Statistics, 2014).

Sea $I_{t,0}$ el índice de precios en el tiempo 0, entonces $I_{t,0}$ es igual a:

Ecuación 3: *IPC*

$$I_{t,0} = 100 \times \frac{\sum_i P_{it}\, Q_{ib}}{\sum_i P_{i0}\, Q_{ib}}$$

Donde:

P_{it} ≔ El precio del i-esimo ítem de la cesta básica en el tiempo t.

P_{i0} ≔ El precio del i-esimo ítem de la cesta básica en el tiempo 0.

Q_{ib} ≔ Cantidad del i-esimo ítem en el año base.

Donde los precios se toman de una muestra en cada parte del país (en este caso la base de siniestros de la compañía) en el que se va a medir el IPC.

Otra forma de escribir el índice es:

Ecuación 4: *Simplificación IPC*

$$I_{t,0} = 100 \times \frac{\sum_i (P_{it} / P_{i0})\, W_i}{\sum_i W_i}$$

Donde $W_i = P_{i0} Q_{ib}$.

Algunos costos son más altos en algunas regiones que en otras (o para este escrito para algunas referencias de autos), por lo que hay que ponerles ponderaciones a los precios. Estas ponderaciones son iguales a $w_i = \frac{P_{it}^k}{\sum_j P_{jt}}$

Donde $\sum_j P_{jt}$ es la suma de los precios totales del i-esimo producto de la canasta básica para todas las referencias y P_{it}^k, es el precio del i-esimo producto para la autoparte k (Office for National Statistics, 2014).

Este índice es muy importante en el desarrollo del trabajo, ya que da una metodología para hallar un número que sólo tome en cuenta el precio puro de las autopartes y elimine los ruidos dados por la frecuencia de los precios. En lo que se consultó del estado del arte no se encontraron antecedentes de esta metodología en actuaría.

3.4. Modelos Autorregresivos, Integrados y de Media Móvil

La idea tras los modelos autorregresivos es predecir el valor x_n de la serie de tiempo $\{x_t\}$ en función de los p valores pasados $\{x_{n-1}, x_{n-2}, \ldots, x_{n-p}\} \subseteq \{x_t\}$, donde p debe es un número natural distinto de 0;

sin embargo, estas condiciones no son suficientes para poder definir un modelo adecuado, es necesario que las serie $\{x_t\}$ sea estacionaria con w_t un ruido aleatorio Gaussiano con media 0 y varianza σ_w^2 (Shumway & Stoffer, 2006). Así las cosas, un modelo *AR(p)* es aquel que se puede escribir como:

Ecuación 5: Proceso AR

$$x_t = \alpha + \varphi_1 x_{t-1} + \varphi_2 x_{t-2} + \cdots + \varphi_p x_{t-p} + w_t$$

Donde p se conoce como el orden, φ_i ($\varphi_p \neq 0$) son los parámetros, α es una constante que se define como $\alpha = \mu(1 - \varphi_1 - \cdots - \varphi_p)$ considerando μ como la media de $\{x_t\}$ (en adelante se considerara que los modelos tienen media 0).

Una de las formas más fáciles de construir una serie de tiempo estacionaria es a través de tomar $\{Z_i\}_{i=0}^{q}$ variables aleatorias i.i.d y definir $X_n = g(Z_{n-1}, Z_{n-2}, \dots, Z_{n-q})$ donde g es una función de variable real, esto quiere decir que si se toma $g(x) = x$ entonces se puede predecir el valor x_n a través de una combinación lineal de las variables aleatorias $\{Z_i\}_{i=0}^{q}$, en particular si estas son los ruidos aleatorios w_{n-1}, w_{n-2},…, w_{n-q} de x_{n-1}, x_{n-2},…, x_{n-q} (Brockwell & Davis, Introduction to Time Series and Forecasting, 2002), en consecuencia, se definen los modelos $MA(q)$ como aquel que se puede escribir como:

Ecuación 6: *Proceso MA*

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + \cdots + \theta_q w_{t-q}$$

Donde $q \neq 0$ es el orden del modelo de media móvil, θ_1, θ_2, …,θ_q ($\theta_q \neq 0$) son los parámetros y $\{w_{t-k}\}_{k=0}^{q} \sim N(0, \sigma_w^2)$.

Según (Brockwell & Davis, Introduction to Time Series and Forecasting, 2002) un proceso $ARMA(p, q)$ es aquel que se genera al integrar los procesos $AR(p)$ y $MA(q)$, cumple que $\{x_t\}$ sea una serie de tiempo estacionaria y que para todo n, x_n se pueda escribir como:

Ecuación 7: *Proceso ARMA*

$$x_n - \varphi_1 x_{n-1} - \varphi_2 x_{n-2} - \cdots - \varphi_p x_{n-p} = w_n + \theta_1 w_{n-1} + \theta_2 w_{n-2} + \cdots + \theta_{n-q} w_q$$

Adicionalmente debe cumplir las siguientes condiciones:

1. Los polinomios $\varphi(z) = 1 - \varphi_1 z - \varphi_2 z^2 - \cdots - \varphi_p z^p$ y $\theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \cdots + \theta_p z^p$; $z \in \mathbb{C}$, con $\theta_p \neq 0$ y $\varphi \neq 0$ no tienen términos en común.
2. El polinomio $\varphi(z) = 1 - \varphi_1 z - \varphi_2 z^2 - \cdots - \varphi_p z^p \neq 0$ para todo $|z| \leq 1$ (esto es equivalente a decir que las raíces del polinomio estén fuera del círculo unitario). Esta condición se denomina **causalidad**.
3. El polinomio $\theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \cdots + \theta_p z^p \neq 0$ para todo $|z| \leq 1$, es decir, las raíces del polinomio $\theta(z)$ deben estar por fuera del círculo unitario. Esta condición se denomina **invertibilidad**.

Si la serie de tiempo $\{x_n\}$ resulta ser no estacionaria se puede calcular $\{\nabla^i x_n\}$ $i = 1,2,3, \ldots d$, donde d será mínimo número de iteraciones donde la prueba de Dickey-Fuller sea significativa, es decir, cuando halla evidencia suficiente para afirmar que la serie $\{\nabla^d x_n\}$ es estacionaria (Hernandez, 2015). En este punto se puede aplicar un modelo $ARMA(p, q)$ a la serie $\{\nabla^d x_n\}$. Este modelo es conocido como $ARIMA(p, d, q)$ y se escribe como (Brockwell & Richard, Time Series: Theory and Methods, 2006):

Ecuación 8: *Proceso ARIMA*

$$\nabla^d x_n - \varphi_1 \nabla^d x_{n-1} - \varphi_2 \nabla^d x_{t-2} - \cdots - \varphi_p \nabla^d x_{n-p} = w_t + \theta_1 w_{n-1} + \theta_2 w_{n-2} + \cdots + \theta_{n-q} w_q. \quad (8)$$

Dependiendo de los valores de p, d y q el proceso puede ser diferente como lo muestra la siguiente tabla:

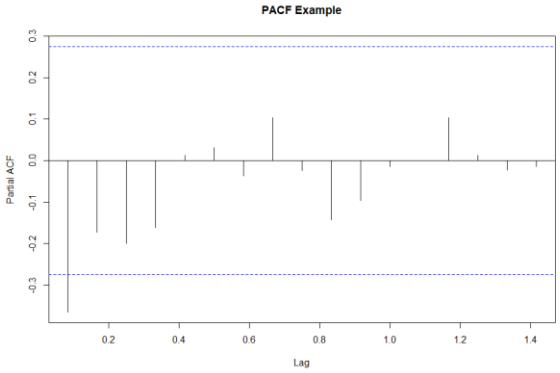
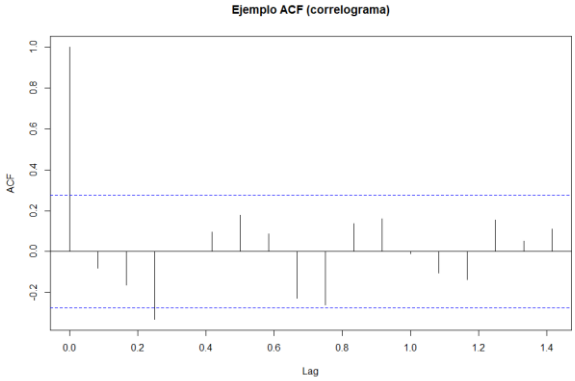
Tabla 1: *Clasificación de los Modelos ARIMA*

Modelo	Descripción
$ARIMA(0,1,0)$	Caminata Aleatoria
$ARIMA(0,0,0)$	Ruido Blanco
$ARIMA(p,0,0)$	Modelo Autorregresivo de orden p
$ARIMA(0,0,q)$	Modelo de Media Móvil de orden q
$ARIMA(p,0,q)$	$ARMA$ de ordenes p y q
$ARIMA(p,d,q)$	Modelo $ARIMA$

Nota: Esta tabla muestra que tipo de proceso ARIMA se tiene dependiendo de p, d y q.

Para estimar los órdenes (Lazerri, 2021) basta con hacer los gráficos del ACF para la parte del MA y el PACF para la parte del proceso AR. Para ello se toma el *lag* más alto que supere las franjas de los intervalos de confianza como se muestra en las siguientes imágenes:

Tabla 2: *Elección del Orden*

Estimación del Orden p del Proceso AR	**Estimación del Orden q del Proceso MA**
Ilustración 1: *Diagrama PACF* 	**Ilustración 2:** *Diagrama ACF* 
El *lag* ideal es 0	**Se toma el *lag* 4 o el 0**

Estos valores serán una aproximación inicial, después se elegirán como ordenes p y q aquellos que minimicen el AIC, BIC o el FPE.

De acuerdo con el trabajo de (Cummins & Griepentrog, 1985) el modelo $ARIMA$ ya fue utilizado para pronosticar el β_s nombrado en la sección 1.1 para el ramo de automóviles usando datos que contenían siniestros de daños al vehículo, donde compararon el modelo $ARIMA$ contra modelos lineales para evaluar su efectividad usando como métricas la raíz del error cuadrático medio RMSE y el TPCE. La conclusión fue que el modelo $ARIMA$ fue peor por casi un 5% que los modelos lineales econométricos tradicionales de ese entonces.

3.5. Vectores Autorregresivos (VAR).

Con el modelo $ARIMA$ de la sección anterior queda el interrogante de ¿Qué pasa si hay más de una serie de tiempo influyendo en la predicción de la serie $\{x_t\}$ o que influyan entre ellas? La respuesta está en que

hay series de tiempo endógenas que explican el comportamiento de $\{x_t\}$ y puede ser que $\{x_t\}$ influya en la explicación de las otras series. Estos modelos son conocidos como Vectores Autorregresivos de orden p (en adelante conocido como $VAR(p)$) y estos son aquellos que se escribe como:

Ecuación 9: *Proceso VAR*

$$\boldsymbol{x_t} = \boldsymbol{v} + A_1\boldsymbol{x_{t-1}} + A_2\boldsymbol{x_{t-2}} + \cdots + A_p\boldsymbol{x_{t-p}} + \boldsymbol{u_t} \qquad t = \pm1, \pm2, \pm3 \ldots,$$

Donde $\boldsymbol{x_t} = < \boldsymbol{x_{1t}}, \boldsymbol{x_{2t}}, \ldots, \boldsymbol{x_{Kt}} >$ es un vector aleatorio de $(K \times 1)$ de elementos a pronosticar provenientes de K series de tiempo estacionarias diferentes, A_i son matrices $(K \times K)$ con coeficientes que acompañan a cada *lag* del proceso VAR, si $k > p$ entonces A_k será una matriz con todas sus componentes iguales a 0, $\boldsymbol{v}$ es el vector de interceptos y $\boldsymbol{u_t} = < \boldsymbol{u_{1t}}, \boldsymbol{u_{2t}}, \ldots, \mathbf{u_{Kt}} >$ es un vector de ruidos blancos $(K \times 1)$. Los ruidos blancos deben cumplir las siguientes condiciones: $\mathbb{E}(\boldsymbol{u_t}) = 0$, $\mathbb{E}(\boldsymbol{u_t}\boldsymbol{u_t^T}) = 0$ y $\mathbb{E}(\boldsymbol{u_t}\boldsymbol{u_s^T}) = \Sigma_u$, con Σ_u no singular, $\boldsymbol{u_t} \sim N(0, \Sigma_u)$ para todo t y u_s debe ser independiente de u_t para todo $s \neq t$ (Lütkepohl, 2005), para evaluar que los residuales $\boldsymbol{u_i}$ no están correlacionados se usa la prueba de *Portmanteau*.

Las condiciones adicionales para que el proceso VAR funcione son (Lütkepohl, 2005):

1. El $det(I_K - A_1 z - A_2 z^2 - \cdots - A_p z^p) \neq 0$ para todo $|z| \leq 1$, $z \in \mathbb{C}$, es decir, las raíces del polinomio característico deben estar fuera de la circunferencia unitaria, esta condición se conoce como **estabilidad**.
2. La serie endógena $\{x_{it}\}$ debe ser **causal** $\{x_{1t}\}$, esto quiere decir que para la regresión del valor $x_n = \alpha + \sum_{i=1}^{q} \gamma_i \, x_{n-i} + \sum_{i=1}^{p} \beta_i y_{n-i} + u_{1t}$ se debe cumplir que $\sum_{i=1}^{p} \beta_i \neq 0$, para evaluar esto se usa la prueba de causalidad de *Granger*.

Para estimar el orden p del VAR se elige aquel que minimice el AIC que para el caso del VAR es igual a $AIC(m) = \ln\left|\widetilde{\boldsymbol{\Sigma}_{\boldsymbol{u}}}(m)\right| + \frac{2mK^2}{T}$, donde m es el orden estimado y T el tamaño de la muestra tomada de $\boldsymbol{x_t}$.

3.6. Extreme Gardient Boost Regressor (XGBoost Regressor):

Es un algoritmo de Machine Learning basado en árboles de decisión específicamente con los "*gradient boosting*" el cual mejora significativamente la velocidad de procesamiento con respecto a los otros algoritmos basados en árboles. La idea básica tras el *XGBoost* es combinar clasificadores o regresores débiles con diferentes pesos al conjunto de árboles en uno fuerte de forma lineal (Zhang, 2021).

El *learning objective* (la capacidad que tiene el modelo de ajustarse de la data) del *XGBoost* se define como (Wade, 2020):

Ecuación 10: *Learning Object XGBoost*

$$obj(\theta) = l(\theta) + \Omega(\theta)$$

Donde $l(\theta)$ es el Error cuadrático Medio MSE.

Los árboles de la iteración actual deben ser lo más parecidos a los residuales de la iteración anterior, la función que determina la $i - esima$ fila es la suma de las predicciones anteriores por lo tanto será:

Ecuación 11: *i-esima fila predicciones*

$$\widehat{y_i} = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F}$$

Donde T es el número de árboles mejorados, $\widehat{y_i}$ es el valor predicho, f_t es el t-esimo árbol regresor, $\mathcal{F}$ es el conjunto de todos los árboles regresores. Quiere decir que el valor de la predicción será la suma de todos los árboles, tanto los anteriores como los actuales.

La función de regularización es igual a:

Ecuación 12: *Función de Regularización*

$$\Omega(f) = \gamma \mathrm{T} + 0.5\, \lambda \sum_{\mathrm{j}=1}^{\mathrm{T}} \mathrm{w}_{\mathrm{j}}^2$$

Con γ y λ constantes para prevenir el sobre entrenamiento.

Por otro lado, la componente del error se puede escribir a través del polinomio de Taylor de segundo grado como:

Ecuación 13: *Función del Error*

$$l(f) = \sum_{i=1}^{n} g_i f_t(x_i) + 0.5 h_i f_t(x_i)^2$$

Donde g_i y h_i son las primera y segundas derivadas de l con respecto a $\widehat{y_{(i)}}$

La función f es una función de la raíz del árbol a las hojas, por lo tanto, se escribirá como $w_{q(x)}$ donde q es una función que asigna puntos de los datos a las hojas.

Así las cosas, el *learning objective* para el t-esimo árbol se expresa como:

Ecuación 14: *Learning Objective*

$$obj^{(t)} = \sum_{i=1}^{T} g_i w_{q(x_i)} + 0.5 h_i w_{q(x_i)}^2 + \gamma \mathrm{T} + 0.5\lambda \sum_{\mathrm{j}=1}^{\mathrm{T}} \mathrm{w_j^2}$$

Por último, minimizando la función $obj^{(t)}$ respecto a w se llega a que (Wade, 2020)

$obj^{(t)} = -0.5 \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma \mathrm{T}$ con $G = \sum g_i$ y $H = \sum h_i$

3.7 Support Vector Regression (SVR):

Un problema de regresión puede ser visto como un problema de clasificación continuo, este es justo el salto que se da entre el modelo tradicional del *Support Vector Machine* (SVM) (diseñado para clasificación) al *Support Vector Regressor* (SVR), para ello al modelo SVM se le agrega una función llamada el ε-tubo que formará una región que se conocerá como ε-insensible y se entenderá que si la distancia entre $f(x)$ (la función de regresión) y $\boldsymbol{y}$ (el vector con los valores reales) es menor que ε entonces no existe error entre los dos. Esto con el fin de que el modelo se acerque al valor de la función-continua real a través de la optimización del ε-tubo, balanceando la optimización del error de predicción. Así las cosas, se construye el "tubo" más delgado alrededor de un hiperplano que contenga la mayor cantidad de puntos del entrenamiento (Awad & Khanna, 2015).

Supóngase que $\boldsymbol{x}$ contiene los M valores input y y los outputs, con en el SVR se busca hallar la función:

Ecuación 15: *Modelo SVR*

$$f(\boldsymbol{x}) = \boldsymbol{\omega} \cdot \boldsymbol{x} + \boldsymbol{b} \;\; con \;\; \boldsymbol{\omega}, \boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^{\boldsymbol{M}}$$

Dado que se busca encontrar el tubo más estrecho alrededor de la superficie con los valores del hiperplano se encuentra que la función objetivo es igual a:

Ecuación 16: *SVR Learning Objective*

$$min_w \frac{1}{2} \|\mathrm{w}\|^2$$

Se debe elegir una función de costo *L* que garantice que el error sea cero entre $\boldsymbol{y}$ y $f(x)$, para ello se toma:

Ecuación 17: *SVR Función de Costo*

$$L_\varepsilon(f(x), y) = \begin{cases} |f(x) - y| - \varepsilon \;\; si \;\; |f(x) - y| > \varepsilon \\ 0 \;\; en\ otro\ caso \end{cases}$$

Para añadir un margen se incluyen las variables de holgura ζ y ζ^* con el fin de darle un margen contra los atípicos, se incluye un hiperparámetro C para darle más peso a la minimización del aplanamiento. Dado que se busca optimizar ω con multiplicadores de *Lag*range aparecen términos como λ, λ^*, α y α^* no negativos así la función objetivo se convierte en

Ecuación 18: *Función a Minimizar SVR*

$$min_w \frac{1}{2} \|\mathrm{w}\|^2 + C \sum_{i=1}^{M} \zeta + \zeta^*$$

Con las condiciones iniciales:

$y_i - \omega^T x \leq \varepsilon + \zeta^*$

$\omega^T x - y_i \leq \varepsilon +$

Al aplicar multiplicadores de *Lag*range se llega a que $\omega = \sum_{i=1}^{M} (\alpha_i{}^* - \alpha_i) x_i$ con la condición de que $\sum_{i=1}^{M} (\alpha_i{}^* - \alpha_i) = 0$, si el modelo es no lineal se requiere un *kernel* $k(x_i, x)$ que es una función que mapea a espacios de dimensión más alta en problemas no lineales (Awad & Khanna, 2015), entonces la

solución no lineal es $\omega = f(x) = \sum_{i=1}^{M}(\alpha_i{}^* - \alpha_i)k(x_i, x)$. El *kernel* que se tomará para este trabajo será el Radial Basis Function (RBF) que está definido como:

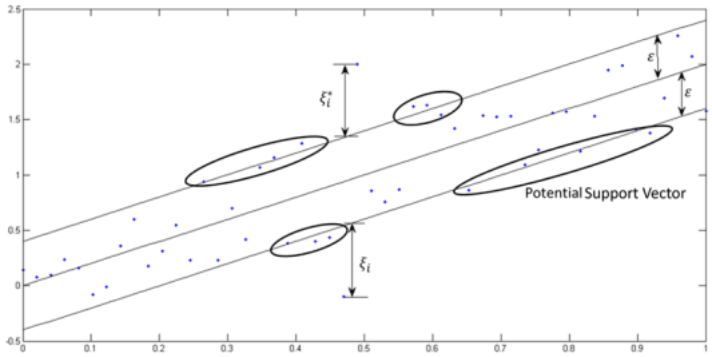
Ecuación 19: *Kernel*

$$k(X_1, X_2) = exp(\frac{\|X_1 - X_2\|}{2\sigma^2})$$

Donde σ^2 es el hiperparametro de la varianza.

A continuación, se muestra una figura para ilustrar el SVR:

Ilustración 3: *SVR en una dimensión*



Nota: Tomado de Efficient Learning Machines (p-68) por Awad ,Mariette y Khanna Rahul

En el estado del arte, no se encontraron proyectos que usaran el SVR actuarialmente, sin embargo, si hay artículos relacionados con el pronóstico del IPC usando este algoritmo, (Wang, Wang, & Xinyang, 2012) usaron el SVR comparándolo con el VAR, usando la hipótesis que el IPC no se comporta linealmente y por lo tanto no debe ser medido con un modelo como el VAR y efectivamente si se ven los resultados del VAR vs el SVR, el SVR tiene un mejor R^2 y RMSE, haciéndolo un mejor modelo para los *lags* que se usaron en comparación.

(Rohmah, 2021) hace un análisis del IPC en 3 ciudades de Indonesia y usan el SVR para predecirlo, usan como medida de error el MAPE "*Mean Absoluto Percentage Error*" para medir sus pronósticos y concluyen que es un muy buen método para el pronóstico de datos como el IPC.

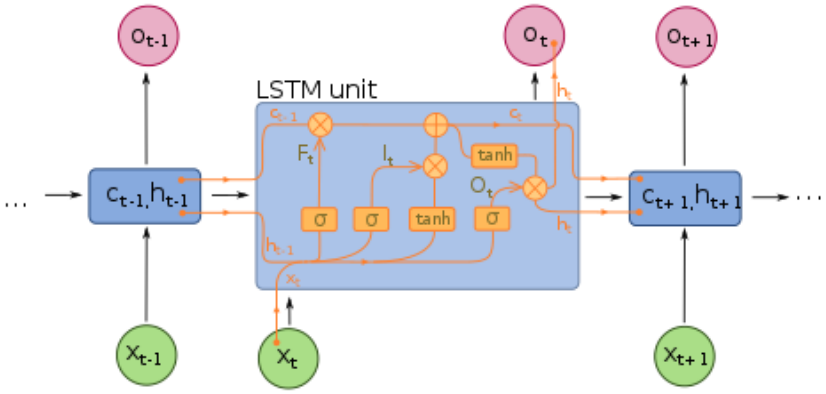
3.8 Long Short Term Memory Networks (LSTM):

Son un tipo de red neuronal recurrente (RNN) capaz de aprender a largo plazo, además ayuda a que el error de *backpropagation* pase de una capa a otra sin perder información, son capaces de olvidar información irrelevante para hacer espacio a datos útiles que ayuden al entrenamiento de la red. Dado que tiene mecanismos internos que le permiten olvidar, procesar y transferir información denominados puertas (por ello hacen parte de las redes neuronales llamadas "*Gated Neural Networks*"). Tienen células de estados que le permiten regular el flujo de información (Lazerri, 2021).

Las celdas toman información de qué, cuánto y cómo, almacenar información, además de qué datos deben dejar entrar, salir o borrar. Usan el algoritmo de "*backpropagation*" para medir el error y usan *gradient descent* (Lazerri, 2021).

Una neurona de una red LSTM luce así:

Ilustración 4: *Neurona de una LSTM*



Nota: Tomado Deep Learning Quick Reference por Bernico, Mike

Donde según (Lazerri, 2021) y (Bernico, 2018) los pasos se definen como:

- $\boldsymbol{x_t}$ es el *input* para el tiempo t.
- o_t es el *output*, C es el bus de memoria que pasa la información del paso anterior C_{t-1} al siguiente paso C_t.
- La puerta de olvido se denomina $F_t = \sigma\big(W_f . [H_{t-1}, x_t]\big)$, esta función determina qué valores se deben olvidar, que valores se deben recordar y qué valores se deben transferir; el *output* de esta función será un número entre 0 y 1 que luego es multiplicado puntualmente con C_{t-1}.
- La puerta *input* $I_t = \sigma(W_i . [H_{t-1}, x_t])$ es usada junto a un candidato de C_t y aprende que puede ser añadido a la memoria de la información del estado oculto anterior h_{t-1} que pasa por la función sigmoide. El valor de I_t está entre 0 y 1. Si está más cerca este a 0 significa que la red no debe añadir el estado y si está más cerca de 1 quiere decir que si debe añadirlo.
- La puerta output $O_t = \sigma(W_o . [H_{t-1}, x_t])$, es la que decide cual debe ser la siguiente capa oculta h_t
- $h_t = O * tanh(C_t)$, cuando h_{t-1} y C_t pasan por la función sigmoide, el resultado es después pasado por $tanh$ y luego es multiplicado por el *output* de la sigmoide para decidir qué información del estado oculto debe quedarse.
- $\sigma(x) = \frac{1}{1+e^{-x}}$ es la función de activación sigmoide.

El IPC ha sido proyectado usando LSTM como se puede observar en (Riofrio, Chang, Fuelagan, & Peluffo, 2020) donde en el caso de Ecuador supera a todos los modelos incluidos el SVR y ARIMA al tener un mejor desempeño en el MAPE.

3.8 Error Cuadrático Medio (MSE)

El error cuadrático medio (Wikipedia, 2014), es un estimador que mide el promedio de errores al cuadrado y es igual a:

Ecuación 20: *Error Cuadrático Medio*

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{Y} - Y\right)^2$$

Donde Y es el valor real, $\hat{Y}$ el valor predicho y N el número de datos

4. Metodología

La metodología utilizada para este proyecto es la CRISP-DM, a continuación, se describirán cada uno de los pasos:

4.1. Entendimiento del Negocio:

Se analizará la dinámica del mercado de autopartes junto con la siniestralidad, el comportamiento del dólar, el costo de ciertas autopartes y el IPC.

Dentro del entendimiento del negocio de las autopartes, se analizará el comportamiento del costo de piezas en función de si son genuinas u originales, por taller, por pieza de la canasta básica, concesionario, marca, modelo y referencia.

Se consideró necesario construir un índice de precios con la metodología del IPC para evitar ruidos inducidos por el propio portafolio de la compañía y que la frecuencia de los siniestros no sesgara la estimación de los datos del incurrido.

4.2 Entendimiento de la Data:

El entendimiento de la data se dividirá en las siguientes fases:

4.2.1. Recolección de los datos:

Se adquirieron las bases de datos históricas de siniestros avisados a marzo del 2022 con una ventana de tiempo que abarca del 1 de enero del 2018 al 31 de diciembre del 2021, la base fue recogida a través del área de siniestros de una compañía de seguros con un portafolio lo suficientemente robusto como para contar con la información de siniestros de los amparos de pérdida y hurto parcial. Cabe resaltar que el ramo de autos se desarrolla muy rápido y los siniestros que aún tenían reservas de siniestros avisados pendientes eran menos del 0.8% de los datos. No se consideró IBNR, por lo tanto, los siniestros no están en el *ultimate* ya que esto sesgaría el valor de mercado de las autopartes.

Se construyó un índice de precios (en adelante índice de la cesta básica o ICB) con la metodología expuesta en la sección 3.3. con unos ligeros ajustes. Se tomó como mes base el 1 de enero del 2018, en lugar de darle un peso *w* a la zona geográfica, se le dio a la llave (“Marca”-“Referencia”) para saber qué

referencias de autos tenían más peso en el portafolio y darles más importancia a los costos de las piezas de estos autos en el índice de precios; al final el ICB contó con 48 puntos de información.

La "canasta familiar" con la que se construyó el índice (en adelante cesta básica) fue definida por el área de siniestros de acuerdo con su experiencia y está compuesta por: bómper delantero y trasero, capó, costados derecho e izquierdo, parte eléctrica, farola derecha e izquierda, frontal, guardafangos, panel trasero, puerta delantera y trasera, stop derecho e izquierdo y baúl.

La serie de tiempo del incurrido de las autopartes fue estimada a través de la misma base de siniestros a través del valor nominal (dado que se piensa calcular una inflación no se usaron valores reales) del incurrido. La reserva de siniestros avisados está basada en cotizaciones por lo que no mete ruidos aleatorios a la base. El mejor estimador de los precios de las autopartes fue la mediana segmentada por mes, marca, referencia y pieza de la cesta básica, con este valor, se construyó el ICB.

Para el dólar se tomó como estimador mensual el último valor de cada mes de la TRM extraído de la SFC (Super Intendencia Financiera de Colombia, 2022) con una ventana de tiempo del 31 de enero del 2018 al 31 de diciembre de 2021. El IPC mensual expedido por el DANE se extrajo de (Banco de la República, 2022) con la misma ventana temporal que la TRM.

4.2.3. Categorizar las variables según el tipo más adecuado:

Se dividirán las variables en categóricas ordinales, categóricas cualitativas, continuas, discretas, fecha, booleanas, etc.

La base de datos contaba con datos categóricos como la marca o la referencia, para la referencia se optó por homologar los modelos de vehículos de acuerdo con una referencia general, por ejemplo, *Chevrolet Aveo Emotion* y el *Chevrolet Aveo Family* se consideraron los dos como "*Chevrolet Aveo*" al momento del cálculo del ICB.

Para los datos numéricos se encontró un estimador por fecha del precio de las autopartes; el cual fue la mediana por cada referencia y luego se calculó el índice acuerdo con la metodología de la parte 3.3.

En variables booleanas estaban si la autoparte era de taller o no, si era genuina o no y si ya había sido entregada.

4.2.4. Entendimiento e imputación de los atípicos y los valores perdidos:

Los atípicos se definieron como aquel dato que superara el percentil 90 de la distribución de siniestros por autoparte (este percentil se definió así por experiencia del área de siniestros y su objetivo final es darle simplicidad al cálculo); luego fueron eliminados de la base de datos.

Se evaluó la base de datos y no se encontraron *missing values* que fuera necesario imputar, sin embargo, si se da el caso algunas técnicas recomendadas para lidiar con ello son la imputación a través de una regresión, la sustitución por la media o la eliminación de los datos vacíos si no son un porcentaje significativo (Kang, 2013).

4.2.5. Análisis descriptivo de la data: Primero se realizó un análisis de la descomposición de las series de tiempo en tendencia, estacionalidad y aleatoriedad para evaluar qué tan similares pueden llegar a ser, después se ejecutó un análisis de correlaciones y multicolinealidad a través de VIF (Variance Inflation Factor).

Se visualizaron las series de tiempo a través de diagramas ACF, CCF y PACF, además de realizar pruebas estadísticas entre las series y sus *lags*. A la serie del ICB se le hizo un diagrama de coordenadas polares para hallar patrones de estacionalidad.

4.3 Modelación

Una vez la data estaba lista, se procedió con el entrenamiento de los modelos. La base de datos se dividió en *train* 80% y *test* 20% sin revolver los datos aleatoriamente para no dañar la parte del orden temporal. Se consideraron dos clases de modelos, los primeros son modelos univariados donde el *input* del modelo fue el primer *lag* del ICB o de la diferencia de éste (dependiendo el modelo) y los multivariados que cuentan con el primer *lag* del ICB, IPC y TRM, además de ponerle variables como el número de la semana (XGBoost), o la transformación $y_n = ln\left(\frac{x_{n-1}}{x_n}\right)$ donde $\{x_t\} \in \{IPC, ICB, TRM\}$ (XGBoost, SVR, LSTM).

Para la ejecución de los modelos se hizo uso de diferentes técnicas dependiendo el modelo, para los LSTM y SVR se recurrió a la transformación *MinMax* de la librería *sklearn* de *Python*. Se seleccionaron

aquellos hiperparámetros que minimizaron la función de costo que para el LSTM, SVR y XGBoost fue el MSE.

Para el ARIMA y el VAR se tuvo que calcular la primera diferencia de las series (excepto por el IPC que fue descartado por no ser estacionario después de hacer 2 diferencias) antes de correr el modelo para volverlas estacionarias. En este contexto el modelo ARIMA será el modelo univariado y el VAR hará las veces de modelo multivariado.

Los modelos de *Machine Learning* se ejecutaron en *Python* y los modelos econométricos en *RStudio*.

4.4 Evaluación

Para evaluar los resultados se usaron técnicas diferentes dependiendo el modelo; para el ARIMA se usó la prueba de *Ljung Box* para ver si los residuales estaban auto correlacionados, la prueba de *Jarque Bera* para la normalidad de los residuales y el *Dickey Fuller* para la significancia de los parámetros.

Para el VAR se hizo la prueba de respuesta-impulso, la prueba de *Portmanteau* para ver las correlaciones entre los residuales y el *Jarque Bera* para revisar la normalidad de los residuales.

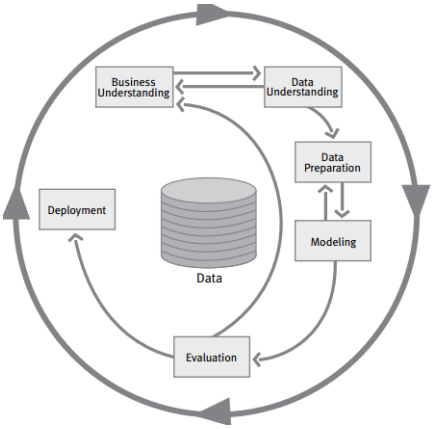
Los modelos de *Deep Learning* se validaron a través del el MSE y en el caso del XGBoost también el MAE. Se revisó que no hubiera *overfiting* observando que el error en el entrenamiento disminuyera junto con el error de *test.* Para el LSTM se validó que el MSE cayera a medida que aumentaba el número de épocas (tanto en *train* como en *test*). Por último, no hubo base de validación debido a la falta de data.

4.5 Implementación:

Es poner en práctica el modelo seleccionado a la compañía o sector requerido, esta fase se llevará a cabo en el futuro para estimar *loss ratios*, reservas y demás.

En la Figura 5 se muestra el proceso detallado de la metodología CRISP-DM:

Ilustración 5: *Metodología CRIP DM*



Nota. Imagen extraída de Opiniones Sobre Ciencia, 2016,

https://opinionessobreciencia.wordpress.com/2016/07/06/metodologia-crisp-dm-entendimiento-del-negocio-2/

5. Resultados y Contribución

En esta sección se hará una descripción, análisis y entendimiento de las características de las series de tiempo trabajadas (Sección 5.1), los modelos univariados ARIMA, XGBoost, LSTM y SVR (Sección 5.2); de los multivariados VAR, XGBoost, LSTM y SVR; de los resultados que dieron en función del MSE (Sección 5.3) y por último una introducción a la metodología para el cálculo de la reserva de siniestros esperados y proyección del incurrido (Sección 5.4). Todos los modelos predicen el mes siguiente, es decir, el tiempo IBC_{n+1}.

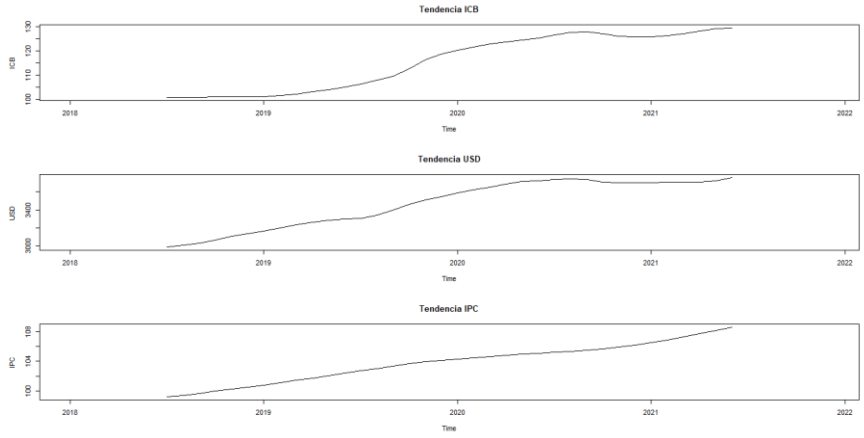
5.1. Análisis Exploratorio de las Series de Tiempo

Se trabajó básicamente con 3 series de tiempo el índice de la cesta básica (ICB), el IPC mensual expedido por el DANE y la serie de tiempo compuesta por la TRM del último día de cada mes. Cada serie cuenta con 48 puntos de información que van de enero del 2018 a diciembre del 2021.

Primero se hizo la descomposición de la serie de tiempo de la TRM y el IPC para observar qué tan parecidas son al ICB; el análisis es el siguiente:

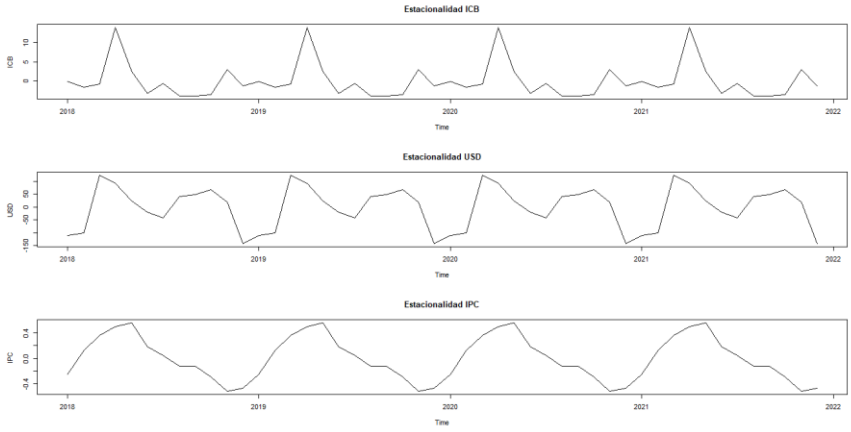
1. **Tendencia:** La tendencia de las tres series de tiempo son crecientes, mostrando que la media de las tres no es constante a medida que avanza el tiempo (se espera que sean claramente no estacionarias), las tres tendencias son similares especialmente la del ICB con la del dolar. La tendencia de las tres series se puede ver en la siguiente imagen:

Ilustración 6: *Tendencia de las series de tiempo*



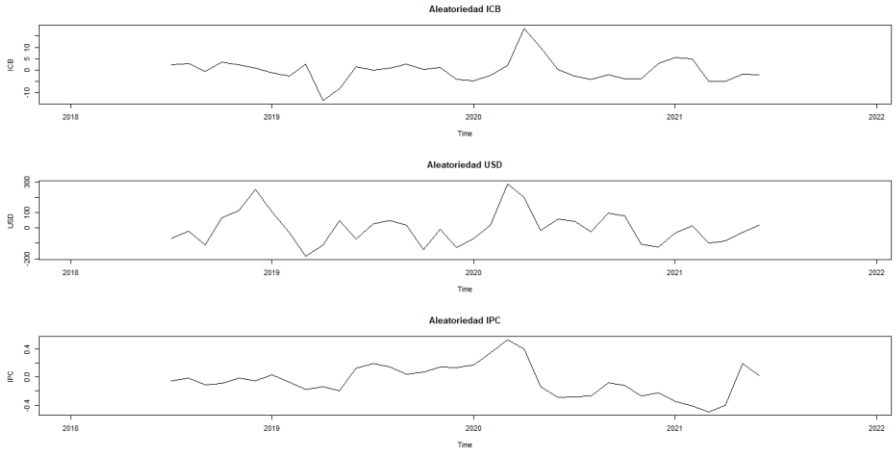
2. **Estacionalidad:** Las tres muestran una estacionalidad anual que se repite en enero de cada año, la TRM parece la serie con más volatilidad en su estacionalidad, seguido por el ICB y por último el IPC. Las series de la estacionalidad se pueden observar a continuación.

   Ilustración 7: *Estacionalidad de las series*



3. **Componente Aleatoria:** La componente aleatoria de la TRM y el ICB son muy similares, como se muestra a continuación.

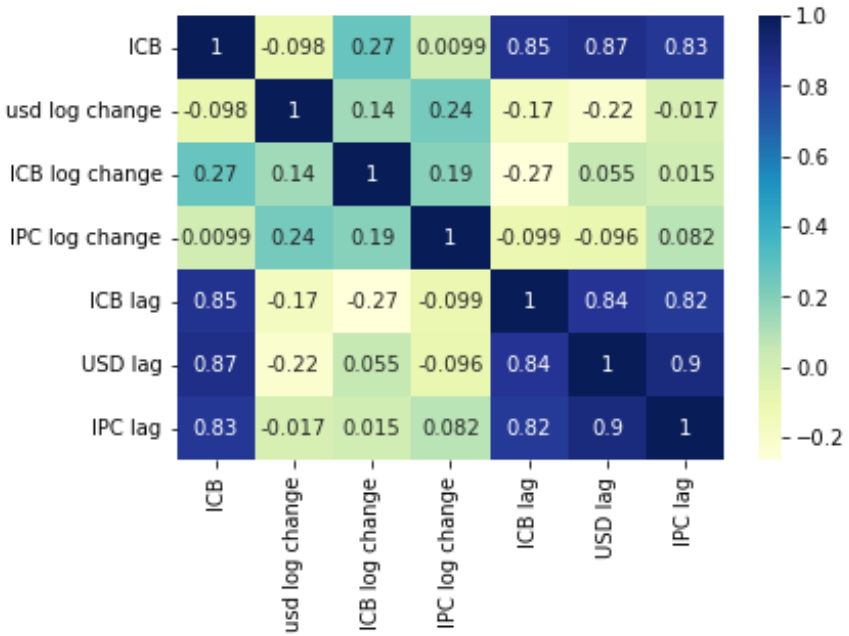
   Ilustración 8: *Aleatoriedad de las series de tiempo*



Antes de ejecutar el modelo es necesario realizar un análisis de multicolinealidad para asegurarse que el IPC y la TRM sean idóneos para la ejecución del modelo. Se hace uso del VIF para medir la multicolinealidad y se obtiene que es igual a 5.3243, esto indica que el IPC y la TRM tiene multicolinealidad, además si se corre un VIF con el ICB en función de los rezagos de las dos variables se obtiene un valor de 5.024; esto indica que alguna de las dos series de tiempo se debe rechazar antes de correr el VAR y esta será el IPC (más adelante se mostrará que el IPC no es estacionario lo que hace aún más razonable su rechazo), no obstante, se utilizará en los modelos de *Machine Learning*.

En los modelos de *Machine Learning* se incluirán las series $y_n(x_n) = ln\left(\frac{x_{n-1}}{x_n}\right)$ donde $\{x_t\} \in \{IPC, ICB, TRM\}$. Estas series son muy comunes en finanzas para medir las volatilidades de títulos valores, es por ello por lo que se incluyeron en el trabajo. Con estas tres nuevas series se mostrará la matriz de correlación de las 6 series de tiempo (las otras 3 son la TRM, el IPC y el ICB), el resultado fue:

Ilustración 9: *Correlación entre las variables*



Claramente se observa una gran correlación entre el ICB y sus primeros rezagos, por otro lado, el hecho de que las funciones $y_n(x_n) = ln\left(\frac{x_{n-1}}{x_n}\right)$ no obtengan una correlación lineal significativa, claramente se debe a la transformación aplicada que cambia la escala de la variable.

Se hizo el análisis de estacionariedad de las 3 series de tiempo antes de ejecutar el primer algoritmo a través de la prueba de *Dickey Fuller* los resultados fueron los siguientes:

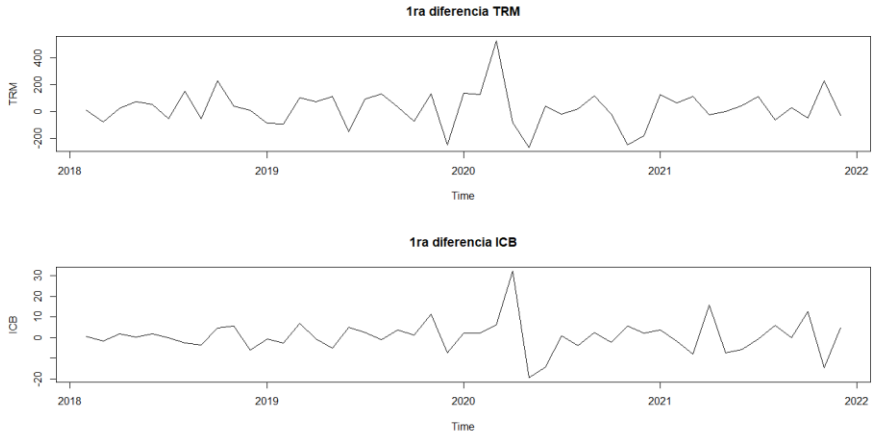
Tabla 3: *Prueba de Dickey Fuller de las 3 series de tiempo*

Serie de Tiempo	***Dickey Fuller* (p-valor)**	***Dickey Fuller* ∇^1 (p-valor)**
ICB	0.36	0.01
IPC	0.63	0.22
TRM	0.56	0.01

Nota. Si el p-valor es menor de 0.05 se considera que la serie es estacionaria

Con esto se concluye que $\nabla^1 ICB$ y $\nabla^1 TRM$ son estacionarias, por lo tanto, son útiles para los modelos ARIMA y el VAR; con los valores de la Tabla 3 se deduce que el valor d del modelo ARIMA es igual 1. A continuación, se muestran las gráficas de las series estacionarias:

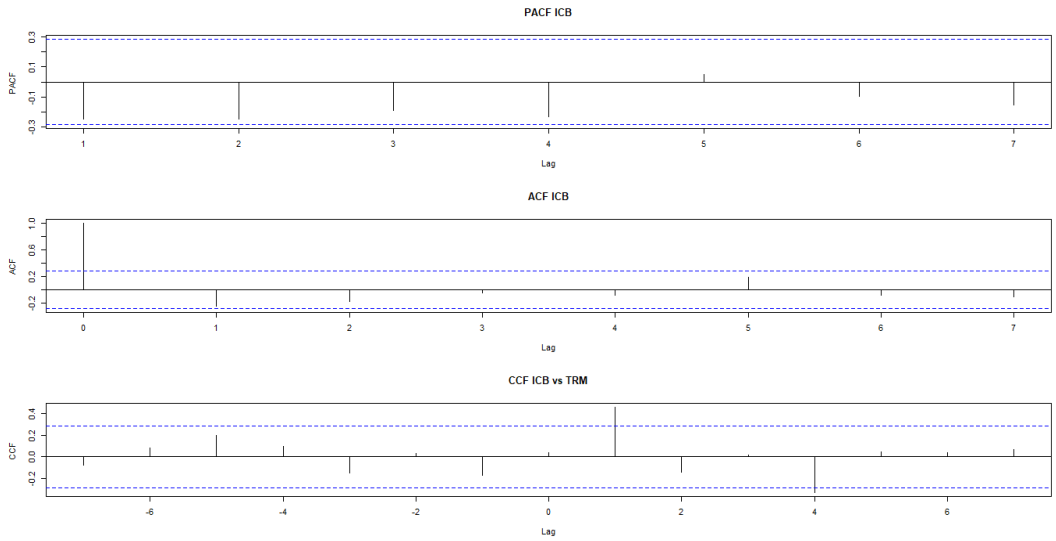
Ilustración 10: *Gráficas de* $\nabla^1 TRM$ *y el* $\nabla^1 ICB$



Una vez que se sabe qué series de tiempo son estacionarias, se procederá a hacer los gráficos del ACF (*Auto Correlation Function*), PACF (*Partial Autocorrelación Function*) y CCF (*Cross Correlation*

Function) para saber la correlación de $\nabla^1 ICB$ con los *lags* de $\nabla^1 TRM$. Con estos tres diagramas se pueden determinar los posibles órdenes p y q para el modelo ARIMA y el orden p del modelo VAR:

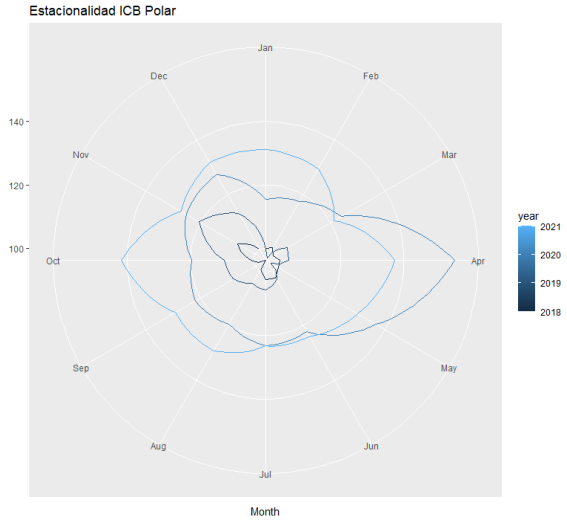
Ilustración 11: *PACF y ACF del ICB; CCF ICB vs TRM*



Con estos resultados se deduce que p y q podrían ser 0, es decir, se espera que el resultado del ARIMA para la serie del ICB sea ruido blanco (recordar que d=1). Para el modelo VAR se estima que p sea igual a 1, porque en el diagrama CCF es el primer *lag* el que supera el intervalo de confianza (en el gráfico son las líneas horizontales punteadas en azul).

Otra parte importante del análisis exploratorio de datos es indagar acerca de la estacionalidad de las series de tiempo pero para decidir si se trata de un ARIMA o un SARIMA (*Seasonal ARIMA*), sin embargo, el ICB no mostró tener estacionalidad fuerte (esto es diferente a la componente estacional); esto puede deberse a que los datos fueron tomados en medio de una pandemia que causó cambios en el crecimiento económico, la TRM y la inflación. La falta de estacionalidad se puede apreciar mejor en el siguiente diagrama:

Ilustración 12: *Diagrama de Estacionalidad*



Por último, se revisó la causalidad entre el ICB con el IPC y la TRM a través de la prueba de Granger, el resultado fue el siguiente:

Tabla 4: *Resumen de la prueba de Causalidad de Granger*

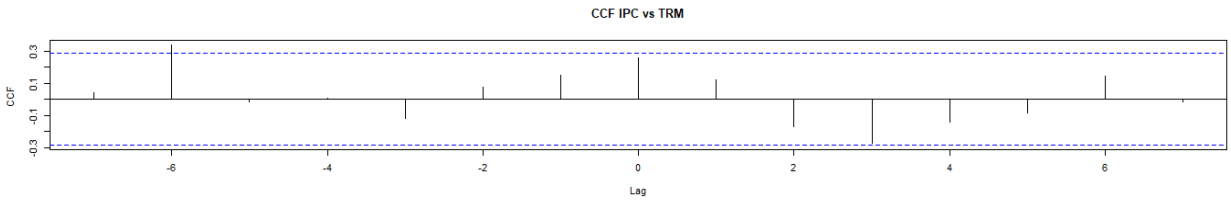
Serie de Datos	p-valor
TRM vs ICB	4.962e-05
$\nabla^1 ICB$ vs $\nabla^1 TRM$	7.777e-04
IPC vs ICB	1.118e-04

Nota. Si el p-valor es menor que 0.05 se dirá que la serie causa el ICB o el $\nabla^1 ICB$

Debido a que $\nabla^1 TRM$ y $\nabla^1 ICB$ son estacionarias y la $\nabla^1 TRM$ causa el $\nabla^1 ICB$, entonces se concluye que serán las variables que se usen para el VAR (el IPC se descarta porque no es estacionaria).

Por último se evaluó si habían autocorrelaciones entre el IPC y la TRM a través de la función CCF y se concluyó que sí en un plazo de 6 meses, lo que ratifica aún más la decisión de excluir el IPC del VAR, en la siguiente imagen se muestra el resultado:

Ilustración 13: *CCF IPC vs TRM*



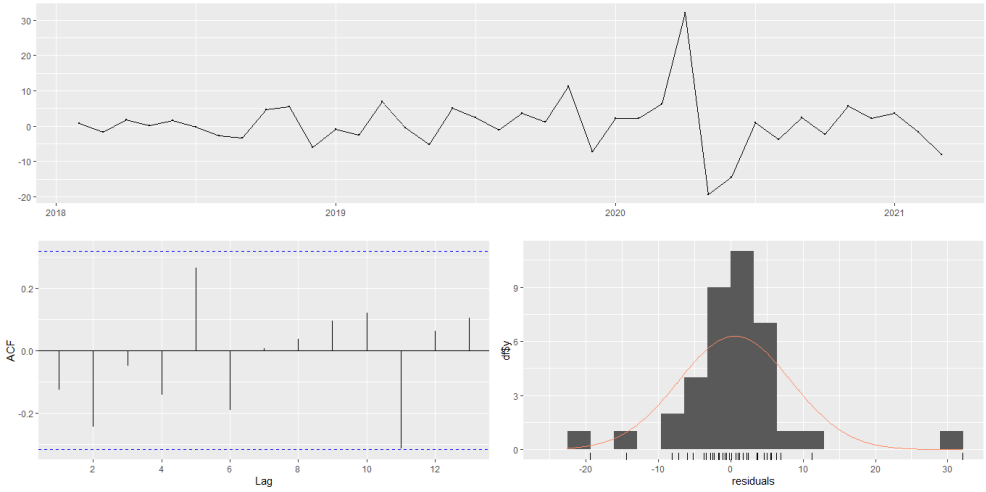
5.2. Modelo $ARIMA(p, d, q)$.

Para ejecutar el modelo se dividió la base en *train* y *test*, donde *train* tiene 39 datos y test 9 datos, luego se calculó la primera diferencia de la serie de *test*, se le aplicó la pueba de Dickey Fuller para saber si era estacionaria y se calculó el p-valor el cual es menor que 0.01, en consecuencia, la base es estacionaria.

Al correr el modelo (ver Anexo 1 para ver el código) se llega a que el modelo que mejor se ajusta a la serie es un $ARIMA(0,1,0)$, es decir el ICB es una caminata aleatoria, esta situación no es necesariamente mala, sino que es la motivación para usar modelos más poderosos y robustos del *Machine Learning*.

Ahora se debe comprobar que efectivamente es una caminata, para ello se verificar que los residuales no son normales y no están auto correlacionados. La primera aproximación para pensar que esto es verdad es el siguiente gráfico.

Ilustración 14: *Residuales del Modelo ARIMA*



Como se observa en el ACF parece ser que los residuales no están auto correlacionados, sin embargo, para tener mayor certeza de este hecho se aplicó la prueba de Ljung-Box y se obtuvo un p-valor de 0.964 (Si el p-valor es mayor que 0.05 hay evidencia suficiente para firmar que los residuales no están correlacionados).

Para las pruebas de normalidad se aplicó una prueba Jarque-Bera y se obtuvo un p-valor de 2.2e-16, por lo que hay evidencia suficiente para afirmar que los residuales no son normales ($p < 0.05$). Por lo tanto, el ICB es una caminata aleatoria.

5.3. Modelo $VAR(p)$.

Dado el análisis exploratorio de la sección 5.1. y por las razones expuestas en el último párrafo de esta sección, se corrió el VAR entre $\nabla^1 ICB$ y $\nabla^1 TRM$.

Para desarrollar el modelo primero se ejecutó un código (ver Anexo 2) para determinar el orden p tal que minimice los modelos de bondad y ajuste, el resultado se encuentra en la siguiente imagen:

Ilustración 15: Órdenes del VAR

```
AIC(n)  HQ(n)  SC(n) FPE(n)
     3      1      1      3

$criteria
                1            2            3            4            5
AIC(n) 1.407804e+01 1.421898e+01 1.404115e+01 1.419075e+01 1.421988e+01
HQ(n)  1.416903e+01 1.437063e+01 1.425346e+01 1.446372e+01 1.455351e+01
SC(n)  1.432628e+01 1.463271e+01 1.462037e+01 1.493547e+01 1.513009e+01
FPE(n) 1.300854e+06 1.500408e+06 1.261015e+06 1.475115e+06 1.536280e+06
```

Nota. Usando esta tabla como referencia y el CCF de la sección 5.1 en el código de R se ejecutará el VAR con un *lag.max*=4

Después se corrió el modelo y se obtuvo el siguiente resultado:

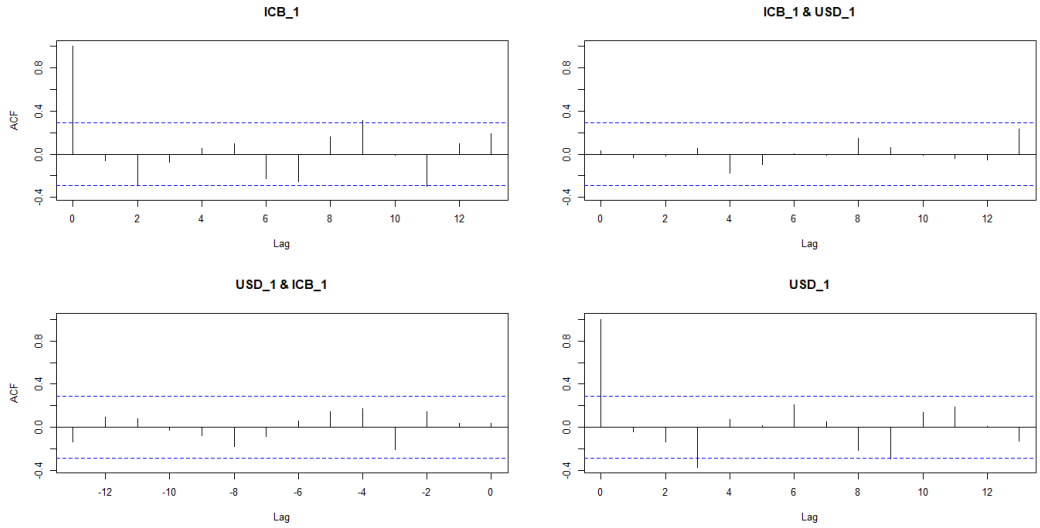
Ecuación 21: *VAR Estimado por R*

$$\nabla^1 ICB_n = -0.2688\nabla^1 ICB_{n-1} + 0.02796\nabla^1 TRM_{n-1} + w_1$$

Donde el coeficiente que acompaña a $\nabla^1 ICB_{n-1}$ tuvo una significancia del 0.0416 y el coeficiente que acompaña a $\nabla^1 TRM_{n-1}$ tuvo una significancia de 0.022796. El orden que optimizó el modelo e hizo que los coeficientes fueran significativos fue $p = 1$.

Ahora se realizó la validación de que los residuales no estuvieran correlacionados, la primera evaluación gráfica se observa a continuación:

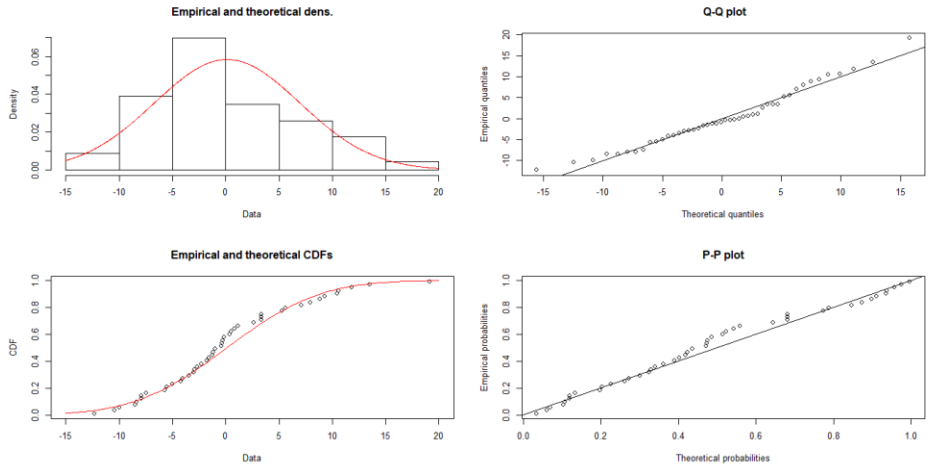
Ilustración 16: ACF Y CCF de los Residuales



Nota. ICB_1 y USD_1 se refieren a $\nabla^1 ICB$ y $\nabla^1 TRM$

Como se puede observar los residuales no parecen correlacionados, sin embargo, se validó usando la prueba de Portmanteau y se obtuvo un p-valor de 0.496 reafirmando la hipótesis de no autocorrelación. Otro punto importante es revisar que los residuales son normales, primero se ejecutó un histograma, un *qq-plot* y *pp-plot*, los resultados fueron los siguientes:

Ilustración 17 *Normalidad de los residuales*



Como se puede apreciar los residuales parecen normales, así que se hizo la verificación con la prueba de Jarque-Bera y se obtuvo un p-valor de *0.3051* comprobando que los residuales si son normales.

La función de impulso respuesta es muy útil para saber qué tanto afecta una unidad de innovación en el futuro de la serie de tiempo (Buteikis, 2020), para el modelo VAR del ICB vs TRM es claro que el dólar no muestra un gran impacto en el ICB, pero el ICB si muestra una perturbación en el modelo VAR del ICB (el VAR del USD en función del ICB y el VAR fue ignorado para este trabajo ya que no está en el alcanza pronosticar la TRM) como se muestra a continuación:

Ilustración 18: *Función de Impulso Respuesta USD*

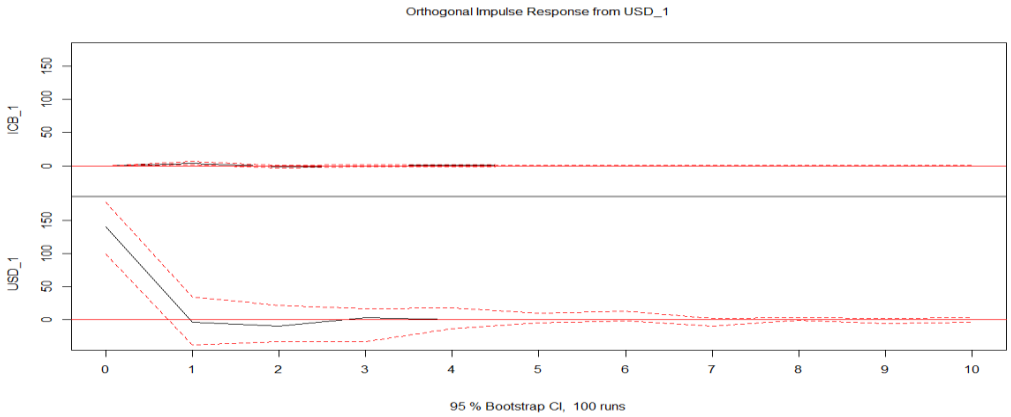
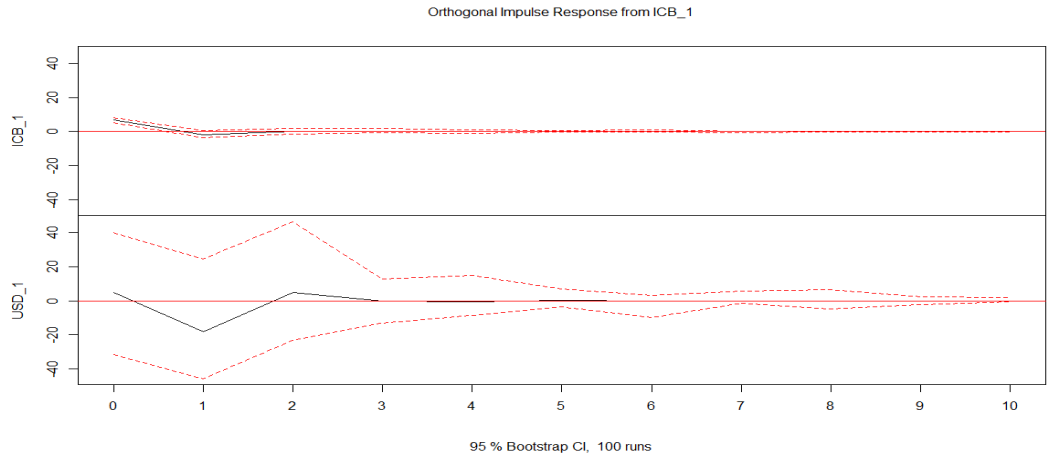


Ilustración 19: *Función Impulso Respuesta ICB*



5.4. Modelos Univariados.

Los modelos Univariados consisten en usar el ICB con un rezago de un 1 mes y usarlo para predecir el siguiente mes, la base de datos utilizada estaba dividida en 80% train y 20% test.

5.4.1. XGBoost univariado

El XGBoost univariado seleccionado fue aquel que tenía un *learning rate* de 0.0015 y tuvo 250 árboles involucrados; se usaron como validadores de overfiting el MAE y el MSE comparándolos con la base de *test* (Para ver el código remitirse a los anexos).

5.4.2. LSTM Univariada:

La LSTM univariada se entrenó con un *batch size* de 5 y con 10 épocas. La arquitectura usada consistió en 8 neuronas LSTM seguidas por una capa densa, a continuación, se muestran los parámetros y el resultado del entrenamiento (Para ver el código remitirse a los anexos):

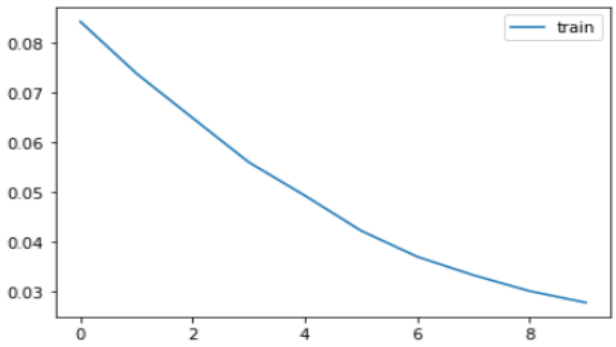
Ilustración 20: Parámetros del LSTM Univariado

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 8)                 320

 dense (Dense)               (None, 1)                 9

=================================================================
Total params: 329
Trainable params: 329
Non-trainable params: 0
_________________________________________________________________
```

Ilustración 21: *Gráfico del error vs el número de épocas Durante el Entrenamiento*



Nota. Se observa que el algoritmo se entrenó correctamente debido a que a medida que aumentan las épocas disminuye el MSE.

5.4.3. SVR.

Para el SVR se utilizó un *kernel* de tipo RBF el cual es explicado en detalle en la sección 3.8. (Para ver el código remitirse a los anexos)

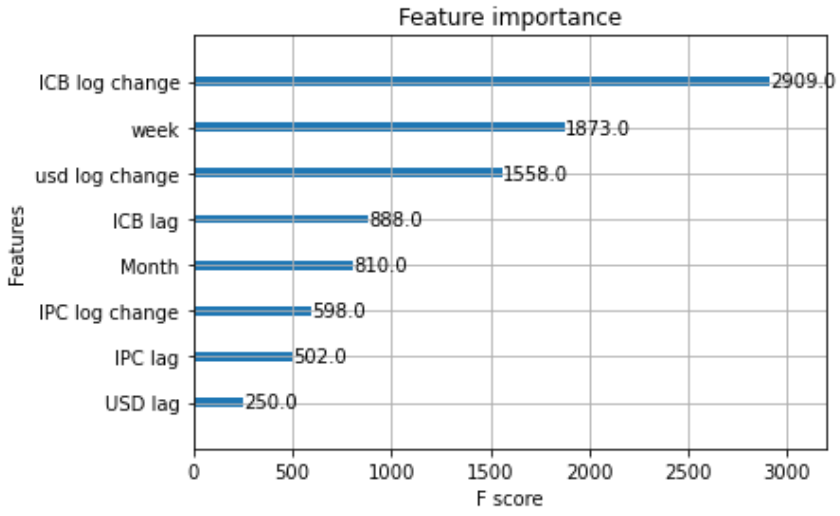
5.5. Modelos Multivariados.

En el caso de los modelos Multivariados se incluyeron los primeros rezagos la TRM, el IPC y además se incluyó la serie de tiempo $y_n(x_n) = ln\left(\frac{x_{n-1}}{x_n}\right)$ donde $\{x_t\} \in \{IPC, ICB, TRM\}$ la adición de estas 3 series se hizo con el fin de captar la volatilidad y los cambios porcentuales de un mes a otro de las series de tiempo, finalmente en el XGBoost se incluyó la variable *semana* que le dio un plus de poder predictivo.

5.5.1. XGBoost Multivariado.

Para programar este modelo se usó un *learning rate* de 0.01 y un total de 1100 árboles de decisión; al terminar el entrenamiento estas fueron las variables que tuvieron mayor importancia (Para ver el código remitirse a los anexos):

Ilustración 22: *Importancia de las Variables*



Aquí se puede apreciar que la inclusión de las funciones $y_n(x_n)$ le aporta más poder predictivo al modelo, así como la semana.

5.5.2. LSTM Multivariado.

El LSTM multivariado se ejecutó con una capa de 5 neuronas LSTM seguidas por una capa densa. Se tomó un *batch size* de 5, con 18 épocas. En las siguientes figuras se muestran los parámetros y gráficos de entrenamiento (Para ver el código remitirse a los anexos)

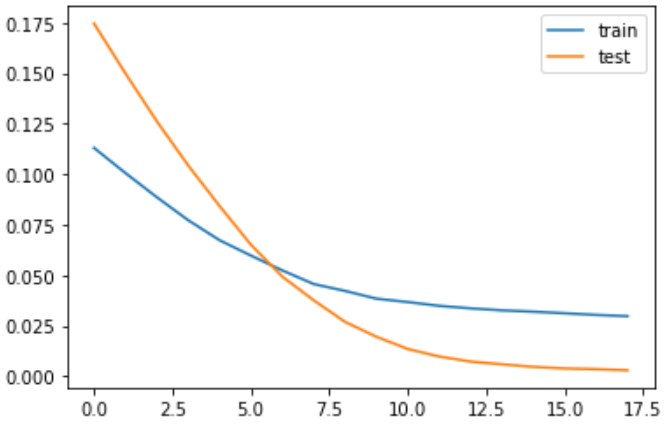
Ilustración 23: *Parámetros de la red neuronal Multivariada*

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 5)                 260

 dense (Dense)               (None, 1)                 6

=================================================================
Total params: 266
Trainable params: 266
Non-trainable params: 0
_________________________________________________________________
```

Ilustración 24: *Entrenamiento de la red neuronal*



Para esta red se intentó poner más capas ocultas de LSTM con diversas funciones de activación, sin embargo, el error se disparaba por sólo poner una segunda capa así que se descartó.

5.5.3. SVR Multivariado.

El SVR fue el que tuvo mejor desempeño con respecto a los demás modelos multivariados, uso al igual que su contraparte univariada un *kernel* RBF (Para ver el código remitirse a los anexos)

5.5. Resumen de los Modelos.

A continuación, se muestran los resultados de los modelos evaluados para el pronóstico del IBC:

Tabla 5: *Evaluación del MSE de por cada Modelo*

	ARIMA/VAR	**SVR**	**LSTM**	**XGBoost**
Univariado	N/A	1.9675	3.508	6.663
Multivariado	9.8372	2.8780	3.255	3.984

Nota. Se observa que el mejor modelo es el SVR.

5.5. Metodología para el Cálculo de Siniestros Avisados e Incurrido Esperado.

La metodología propuesta por este trabajo consiste en estimar β_s mostrado en la sección 3.2. a través del cálculo de una "inflación" usando el ICB calculado con la metodología de la sección 3.3 para esta compañía aseguradora. La inflación $i_{n,n+k}$ entre los periodos n y $n+k$ de la Canasta Básica será definida como:

Ecuación 22: *Inflación del Portafolio*

$$i_{n,n+k} = \frac{IBC_n}{IBC_{n+k}} - 1$$

Por lo tanto, la tasa β_s debe ser igual a $i_{n,n+k}$. Si el IBC no es conocido debido a que es un valor que se encuentra en el futuro, basta estimar $\widehat{IBC} = Model(\{IBC_t\})$, donde $Model \in \{SVR, XGBoost, LSTM, VAR\}$, tomando $\widehat{\beta_s} = \frac{IBC_n}{\widehat{IBC_{n+k}}} - 1$ se tiene que el incurrido esperado es igual a:

Ecuación 23: *Incurrido Esperado*

$$\widehat{I_{n+1}} = I_n\left(1 + \widehat{\beta_s}\right)$$

con $\widehat{I_{n+1}} :=$ Incurrido esperado y $I_n :=$ Incurrido actual

Tal como lo propone (Cummins & Griepentrog, 1985), con el factor diferenciador de que el índice es construido con el mismo portafolio de la compañía.

Como este índice ICB es propio de cada compañía, en el momento en el que se reporte un siniestro la reserva de siniestros avisados RSA será igual a:

Ecuación 24: *Reserva de Siniestros Avisados*

$$RSA = \left(\sum_i VPCB_i\right)\left(1 + \widehat{\beta_s}\right) * p_j \quad (22)$$

Donde $(\sum_i VPCB_i)$ es el valor de las i autopartes dañadas durante el siniestro y p_j la probabilidad de que un juzgado falle en contra de la aseguradora generalmente es una distribución binomial ajustada con los datos legales de la compañía, esta probabilidad es opcional y sugerida por (Fasecolda). Puede darse el caso en el que la pieza no necesite ser necesariamente cambiada si no reparada, por lo tanto, el RSA actuará como una cota superior de la pérdida, a medida que el siniestro se desarrolle el estimador deberá aplicarse sobre sobre las piezas faltantes por repararse. Cabe recalcar que si la pieza afectada no está incluida en la cesta básica el factor $\widehat{\beta_s}$ se debería aplicar igual tal como se hace habitualmente con la inflación usual.

6. Conclusiones y Recomendaciones

6.1. Conclusiones

Se pudo obtener una metodología para el cálculo del incurrido esperado y la reserva de siniestros avisados en los amparaos de perdida y hurto parcial, la cual fue usar el SVR para estimar $\widehat{\beta_s}$ de tal manera que fuera posible capitalizar los valores de las autopartes; a su vez se validó la pertenencia de desarrollar algoritmos de inteligencia artificial mejorando la capacidad predictiva para la resolución del problema, atendiendo así el objetivo general y los específicos planteados al inicio del presente trabajo.

La segunda conclusión es que la TRM pesa mucho más en la explicación de los siniestros (a través del ICB) que el mismo IPC, lo que indica que el mercado de autopartes es mucho más sensible a movimientos del dólar y a la importación de piezas que a cambios en la canasta familiar y el aumento en el costo de vida.

Se concluye que pese a estar en pandemia y a la caída en la frecuencia en el año 2020, las piezas se siguieron comportando de acuerdo con el movimiento del dólar, por lo que no fue un factor decisivo en el cálculo del índice.

En la ejecución de los modelos se encontró que el que tiene mejor capacidad predictiva tanto en modelos univariados como en multivariados es el SVR, que sorprendentemente logró una mayor explicabilidad en su forma univariada que en su forma multivariada, esto puede deberse a que el LSTM, SVR multivariado y el XGBoost requieran una mayor cantidad de datos para tomar decisiones.

El modelo ARIMA fue una caminata aleatoria, por lo que se concluye que necesariamente para el análisis de series de tiempo se requieren modelos más robustos y potentes que los modelos econométricos tradicionales y demuestra que la dinámica de los siniestros es mucho más compleja y difícil de explicar que la linealidad.

Con este modelo deberían mejorar las estimaciones del *loss ratio* proyectado y usarlo de cara a la construcción de planes de negocio y control del gasto, además de darle explicabilidad a las variables externas que afectan el portafolio de autos y crear planes de mitigación de riesgos para controlarlos.

6.2. Recomendaciones y Pasos Siguientes.

Usar nuevas variables en el entrenamiento de los modelos como el salario mínimo, la DTF, los tipos de interés, otras *commodities* como el acero o el plástico, incluir más divisas como el euro, etc.

Probar con otros modelos diferentes para estimar β_s, entre algunos modelos sugeridos están las redes neuronales CNN, las GRU o las RNN. En los modelos econométricos tradicionales usar el suavizamiento exponencial y dado el caso y si las variables lo permiten probar más modelos como los SARIMAX, los VECM o SVAR.

Incluir modelos de pérdida y hurto total con esta misma metodología, pero en lugar de modelar las autopiezas usar los carros por marca y referencia.

Encontrar la forma de incluir el índice en los modelos de IBNR con corrección de inflación para aumentar la "personalización" en las reservas de las compañías.

Refinar la construcción del índice y hallar metodologías para incluir productos nuevos en la cesta básica, así como la llegada de nuevos modelos de autos que se hagan populares en el mercado o la salida de referencias antiguas que ya no sean tan significativas como lo fueron en el año base.

Seguir recolectando datos para hacer más robusto el ICB y mejorar los *inputs* para el entrenamiento de modelos en el futuro.

Construir un "índice" para la frecuencia que tenga en cuenta series de tiempo climáticas, geográficas y demográficas para tener en cuenta la parte de la exposición en el modelo y poder construir una prima pura de riesgo en una compañía en la que los datos no alcancen para crear un GLM.

Referencias y Bibliografía

Asteriou, D., & Stephen, H. (2011). *Applied Econometrics.* New York: Palgrave MacMillan.

Awad, M., & Khanna, R. (2015). *Efficient Learning Machines.* New York: Springer.

Banco de la República. (Marzo de 2022). *Banco de la República*. Obtenido de https://www.banrep.gov.co/es/estadisticas/indice-precios-consumidor-ipc

Bernico, M. (2018). *Deep Learning Quick Reference.* Mumbai: Packt Publishing.

Brockwell, P., & Davis, R. (2002). *Introduction to Time Series and Forecasting.* Nueva York: Springer.

Brockwell, P., & Richard, D. (2006). *Time Series: Theory and Methods.* New York: Springer.

Buteikis, A. (2020). Multivariate models: Granger causality, VAR and VECM models. *Multivariate models: Granger causality, VAR and VECM models*. Vilna, Lituania.

Cummins, D., & Griepentrog, G. (1985). Forecasting Automobile Insurance and Paid Claim Costs using Economoetric and ARIMA Models. *International Journal of Forecasting 1*, 203-215.

Eshel, G. (s.f.). The Yule Walker Equations for the AR Coefficients.

Fasecolda. (s.f.). *Fasecolda.* Obtenido de Facecolda: https://fasecolda.com/cms/wp-content/uploads/2019/08/15_efectos_en_el_pg_en_subestimar_la_reserva_de_ibnr.pdf

Hernandez, S. (2015). Análisis de Series de Tiempo. *Curso Regional Sobre Hoja de Balance de Alimentos, Series de Tiempo y Análisis de Política*. Ciudad de México: CEPAL.

Hyndman, R., & Khandakar, Y. (2008). Automatic Time Series Forecasting: Th forecast Package for R. *Journal of Statistical Software*, 8-12.

Kang, H. (2013). The Prevention and Handling of the Misssing Data. *Korean Journal Anesthesiology*, 402-406.

Kwiatkowski, D., Phillips, P., Schmidt, P., & Shin, Y. (1991). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics*, 159-178.

Lazerri, F. (2021). *Machine Learning for Time Series with Python.* Indianapolis: Wiley.

Lütkepohl, H. (2005). *New Introduction to Multiple Time Series Analysis.* Berlin: Springer.

Office for National Statistics. (2014). *Consumer Price Index Technical Manual.* Londres: Office for National Statistics.

Riofrio, J., Chang, O., Fuelagan, R., & Peluffo, D. (2020). Forecasting the Consumer Price Index of Ecuador a Comparative Study of Predictive Models. *International Journal on Advanced Science Engineering Information Technology*.

Rohmah, F. (2021). Comparison Four Kernels of SVR to Predict Consumer Price Index. *Journal of Physics: Conference Series*.

Shumway, R., & Stoffer, D. (2006). *Time Series Analysis and its Applications.* New York: Springer.

Super Intendencia Financiera de Colombia. (Marzo de 2022). *Super Intendencia Financiera de Colombia*. Obtenido de https://www.superfinanciera.gov.co/inicio/informes-y-cifras/cifras/establecimientos-de-credito/informacion-periodica/diaria/tasa-de-cambio-representativa-del-mercado-trm-60819

Wade, C. (2020). *Hands-On Gradient Boosting with XGBoost and scikit-learn.* Packt Publishing.

Wang, Y., Wang, B., & Xinyang, Z. (2012). A new application of the support vector regression on the construction of financial conditions index to CPI prediction . *International Conference on Computational Science*, 1263-1272.

Werner, G., & Claudine, M. (2016). *Basic Ratemakingx.* CAS.

Wikipedia. (Mayo de 2014). *Wikipedia*. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio

Zhang, L. (2021). Time series forecast of sales volume based on XGBoost. *Journal of Physics*.

Anexos

Anexo 1 Tabla de la TRM y el IPC

Fecha	TRM	IPC
2018-01-01	2,844.14	97.53
2018-02-01	2,855.93	98.22
2018-03-01	2,780.47	98.45
2018-04-01	2,806.28	98.91
2018-05-01	2,879.32	99.16
2018-06-01	2,930.80	99.31
2018-07-01	2,875.72	99.18
2018-08-01	3,027.39	99.30
2018-09-01	2,972.18	99.47
2018-10-01	3,202.44	99.59
2018-11-01	3,240.02	99.70
2018-12-01	3,249.75	100.00
2019-01-01	3,163.46	100.60
2019-02-01	3,072.01	101.18
2019-03-01	3,174.79	101.62
2019-04-01	3,247.72	102.12
2019-05-01	3,357.82	102.44
2019-06-01	3,205.67	102.71
2019-07-01	3,296.85	102.94
2019-08-01	3,427.29	103.03
2019-09-01	3,462.01	103.26
2019-10-01	3,389.94	103.43
2019-11-01	3,522.48	103.54
2019-12-01	3,277.14	103.80
2020-01-01	3,411.45	104.24
2020-02-01	3,539.86	104.94
2020-03-01	4,064.81	105.53
2020-04-01	3,983.29	105.70
2020-05-01	3,718.82	105.36
2020-06-01	3,758.91	104.97
2020-07-01	3,739.49	104.97
2020-08-01	3,760.38	104.96
2020-09-01	3,878.94	105.29
2020-10-01	3,858.56	105.23
2020-11-01	3,611.44	105.08
2020-12-01	3,432.50	105.48
2021-01-01	3,559.46	105.91
2021-02-01	3,624.39	106.58

Fecha	TRM	IPC
2021-03-01	3,736.91	107.12
2021-04-01	3,712.89	107.76
2021-05-01	3,715.28	108.84
2021-06-01	3,756.67	108.78
2021-07-01	3,867.88	109.14
2021-08-01	3,806.87	109.62
2021-09-01	3,834.68	110.04
2021-10-01	3,784.44	110.06
2021-11-01	4,010.98	110.60
2021-12-01	3,981.16	111.41

Anexo 2

Construcción del índice (Python)

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib as plt
import datetime as dt
import scipy
from scipy import stats
Siniest_lvn_Precio['Año_cotizacion']=pd.DatetimeIndex(Siniest_lvn_Precio["Fec
ha Cotización"]).year
Siniest_lvn_Precio['Mes_cotizacion']=pd.DatetimeIndex(Siniest_lvn_Precio["Fec
ha Cotización"]).month
Sin_lvn_media=Siniest_lvn_Precio.groupby(by=["Armadora -LIVIANO",
                                             "Linea - LIVIANO",
                                             "PIEZA CESTA BASICA - LIV",
                                             'Año_cotizacion',
                                             'Mes_cotizacion']).median()
Sin_lvn_count=Siniest_lvn_Precio.groupby(by=["Armadora -LIVIANO",
                                             "Linea - LIVIANO",
                                             "PIEZA CESTA BASICA - LIV",
                                             'Año_cotizacion',
                                             'Mes_cotizacion']).count()
Sin_lvn_count=Sin_lvn_count.reset_index()
Sin_lvn_count=Sin_lvn_count.loc[(Sin_lvn_count["Año_cotizacion"]==2018) &
(Sin_lvn_count["Mes_cotizacion"]==1)]
#Sin_lvn_count=Sin_lvn_count.drop("Fecha Cotización",axis=1)
#Sin_lvn_count=Sin_lvn_count.drop(["level_0","index"],axis=1)
Sin_lvn_count
Sin_lvn_media=Sin_lvn_media.merge(Sin_lvn_count,how="left",left_on=["Linea -
LIVIANO","PIEZA CESTA BASICA - LIV"],
                                  right_on=["Linea - LIVIANO","PIEZA CESTA
BASICA - LIV"]).drop(["Año_cotizacion_y",

"Mes_cotizacion_y",

"Armadora -LIVIANO_y"
```

```
],axis=1)

#Sin_lvn_media=Sin_lvn_media.drop("Fecha Cotización",axis=1)

Sin_lvn_media=Sin_lvn_media.loc[Sin_lvn_media["Linea - LIVIANO"] !="OTROS"]

Sin_lvn_P_Base=Sin_lvn_media.loc[(Sin_lvn_media["Año_cotizacion"] ==2018) &
(Sin_lvn_media["Mes_cotizacion"] ==1),
                                 ["Linea - LIVIANO","PIEZA CESTA BASICA -
LIV","Precio"]]

Sin_lvn_P_Base=Sin_lvn_P_Base.rename(columns={"Precio":"Precio_Base"})

Sin_lvn_media=Sin_lvn_media.merge(Sin_lvn_P_Base,how="left",left_on=["Linea -
LIVIANO","PIEZA CESTA BASICA - LIV"],
                                  right_on=["Linea - LIVIANO","PIEZA CESTA
BASICA - LIV"])

Sin_lvn_media["Weight"]=Sin_lvn_media["Cantidad_Mes_Base"]*Sin_lvn_media["Pre
cio_Base"]

Sin_lvn_media["CPI
Factor"]=Sin_lvn_media["Weight"]*Sin_lvn_media["Precio"]/Sin_lvn_media["Preci
o_Base"]

Sin_lvn_CPI=Sin_lvn_media[["Armadora -LIVIANO",
              "Linea - LIVIANO",
              "Año_cotizacion",
              "Mes_cotizacion",
              "Weight",
              "CPI Factor"]].groupby(["Armadora -LIVIANO",
                                      "Linea - LIVIANO",
                                      "Año_cotizacion",

"Mes_cotizacion"]).sum().reset_index()

Sin_lvn_CPI["CPI_Reference"]=100*Sin_lvn_CPI["CPI
Factor"]/Sin_lvn_CPI["Weight"]

Sin_lvn_CPI=Sin_lvn_CPI.loc[Sin_lvn_CPI["Año_cotizacion"]>2017]

Sin_lvn_CPI=Sin_lvn_CPI.merge(Sin_Precio_Mes[["Linea -
LIVIANO","Año_cotizacion","Mes_cotizacion","Precio"]],
                    how="left",left_on=["Linea -
LIVIANO","Año_cotizacion","Mes_cotizacion"],
                    right_on=["Linea -
LIVIANO","Año_cotizacion","Mes_cotizacion"])

Sin_Precio=Sin_lvn_CPI[["Año_cotizacion","Mes_cotizacion","Precio"]].groupby(
["Año_cotizacion","Mes_cotizacion"]).sum().reset_index()


Sin_Precio=Sin_Precio.rename(columns={"Precio":"Precio_Mes"})

Sin_Precio
```

```
Sin_lvn_CPI=Sin_lvn_CPI.merge(Sin_Precio,
                    how="left",left_on=["Año_cotizacion","Mes_cotizacion"],
                    right_on=["Año_cotizacion","Mes_cotizacion"])
Sin_lvn_CPI["Weight_by_reference"]=Sin_lvn_CPI["Precio"]/Sin_lvn_CPI["Precio_
Mes"]
Sin_lvn_CPI=Sin_lvn_CPI.drop(["Precio","Precio_Mes"],axis=1)
Sin_lvn_CPI["CPI_Factor_2"]=Sin_lvn_CPI["CPI_Reference"]*Sin_lvn_CPI["Weight_
by_reference"]
CPI=Sin_lvn_CPI[["Año_cotizacion","Mes_cotizacion","CPI_Factor_2"]].groupby([
"Año_cotizacion","Mes_cotizacion"]).sum().reset_index()
CPI["day"]=1
CPI["Fecha"]=pd.to_datetime(dict(year=CPI.Año_cotizacion,month=CPI.Mes_cotiza
cion,day=CPI.day))
```

ARIMA (RStudio)

```
#1. Se divide la serie de tiempo en train y test:
train<-head(CPI_ts,39)
test<-tail(CPI_ts,9)
CPI_ts_1<-diff(CPI_ts,lag=1)
#2. Medición de estacionaridad de la serie del IPC de autos (train):
adf.test(train)
#2.1.No es estacionaria, se procede a hacer la primer diferencia:
train_diff_1=diff(train,lag=1)
adf.test(train_diff_1) #La serie es estacionaria
kpss.test(train_diff_1, null="Trend") #La serie es estacionaria
#2.2. Observar normalidad en la serie
qqnorm(train_diff_1)
qqline(train_diff_1)
#3. Diagramas ACF y PACF:
acf(train_diff_1,main="ACF Train Data")
pacf(train_diff_1,main="PACF Train Data")
#Parece que no hay lags.
#4. Se corre el modelo ARIMA
ARIMA_IPC<-auto.arima(train_diff_1) #Es un modelo ARIMA(0,1,0) Es ruido
blanco
ARIMA_IPC
#5. Los residuales no están autocorrelacionados y son normales
checkresiduals(ARIMA_IPC,main="Residuales ARIMA")
#5.1 No autocorrelación de los residuales
Box.test(ARIMA_IPC$residuals, lag=37, type="Ljung-Box")
#p>0.05 los residuales no están autocorrelacionados
#5.2 Los residuales son normales, por lo tanto son ruido blanco
jarque.bera.test(ARIMA_IPC$residuals)
#Por Jarque Bera los residuos tienen distribución normal p<0.05
```

VAR (RStudio)

```
library(vars)
#1. Variables económicas
CPI_ts=ts(CPI_[,2],c(2018,1),frequency = 12)
USD_ts=ts(CPI_[,3],c(2018,1),frequency = 12) #USD
IPC_ts=ts(CPI_[,4],c(2018,1),frequency = 12) #IPC Se descarta el IPC para el modelo VAR no es estacionaria en 2 diferencias
#Convertir las series en estacionarias
adf.test(CPI_ts) #La serie no es estacionaria
CPI_1=diff(CPI_ts,lag=1)
adf.test(CPI_1) #Dif 1. estacionaria
adf.test(USD_ts) #La serie no es estacionaria
USD_1=diff(USD_ts,lag=1)
adf.test(USD_1) #Dif 1. no estacionaria
adf.test(IPC_ts) #La serie no es estacionaria
IPC_1=diff(IPC_ts,lag=1)
adf.test(IPC_1) #Dif 1. estacionaria
#4. Sólo se dejan las series que causen el IPC de autos a través de la prueba de Granger
grangertest(USD_ts,CPI_ts,order = 1)
grangertest(USD_1,CPI_1,order = 1)
grangertest(IPC_ts,CPI_ts,order = 1)
data<-cbind(CPI_1,USD_1)
ts<-ts(data,start=c(2018,2),frequency = 12)
train<-head(ts,39)
test<-tail(ts,9)
plot.ts(train)
#5. Elección del Orden se elige p=3
VARselect(ts,lag.max = 5)
#6. Modelo
VarCPI<-VAR(ts, lag.max =4 , type="none")
summary(VarCPI)
#7. Validación del modelo
residuales<-resid(VarCPI)
#7.1 Residuales no autocorrelacionados
acf(residuales)
```

```
#7.3 Raices del polinomio caracteristico
roots(VarCPI)

plot(irf(VarCPI, impulse = "USD_1"))
#7.4 Test de Portmanteau
serial.test(VarCPI, lags.pt = 16)
#Los residuales están correlacionados
#Predicción de valores
predict(VarCPI,test)
```

XGBOOST Univariado (https://machinelearningmastery.com/xgboost-for-time-series-forecasting/#:~:text=XGBoost%20is%20an%20implementation%20of,model%20for%20time%20series%20forecasting) (Python)

```
#Importar Librerias
import xgboost as xg
from pandas import concat
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from xgboost import plot_importance
from matplotlib import pyplot
model = xg.XGBRegressor()
#Arreglo de los datos
CPI=CPI[["CPI"]]
CPI["Fecha"]=CPI["Fecha"].str[:10]
CPI["Fecha"]=pd.to_datetime(CPI["Fecha"],format='%Y/%m/%d')
#Función Sliding Window
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
	n_vars = 1 if type(data) is list else data.shape[1]
	df = data
	cols = list()
	for i in range(n_in, 0, -1):
		cols.append(df.shift(i))
	for i in range(0, n_out):
		cols.append(df.shift(-i))
	agg = concat(cols, axis=1)
	if dropnan:
		agg.dropna(inplace=True)
	return agg.values
data=series_to_supervised(CPI)
#Dividir la data en train y test
def train_test_split(data, n_test):
	return data[:-n_test, :], data[-n_test:, :]
train,test=train_test_split(data, 5)
train = np.asarray(train)
trainX, trainy = train[:, :-1], train[:, -1]
test = np.asarray(test)
```

```
testX, testy = test[:, :-1], test[:, -1]

#Entrenamientos del Modelo

reg=xg.XGBRegressor(n_estimators=400,learning_rate=0.015)

reg.fit(trainX,trainy,eval_set=[(trainX,trainy),(testX,testy)],eval_metric=["
mae","rmse"])

per=[202107,202108,202109,202110,202111]

df=pd.DataFrame({"Month":per,"y_pred":predictions,"y_test":testy.tolist()})

# MSE

from sklearn.metrics import mean_squared_error

rms = mean_squared_error(df["y_test"], df["y_pred"], squared=False)
```

XGBOOST Multivariado (Python)

```
#Importar Librerias
import xgboost as xg
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from xgboost import plot_importance
from matplotlib import pyplot
model = xg.XGBRegressor()
#Tratamiento de la Base
CPI=CPI[["Fecha","CPI","usd","IPC"]]
CPI["week"]=pd.DatetimeIndex(CPI["Fecha"]).week
CPI["Month"]=pd.DatetimeIndex(CPI["Fecha"]).year*100+pd.DatetimeIndex(
CPI["Fecha"]).month
CPI["usd change"]=np.log(CPI.usd/CPI.usd.shift(1))
CPI["CPI change"]=np.log(CPI.CPI/CPI.CPI.shift(1))
CPI["IPC change"]=np.log(CPI.IPC/CPI.IPC.shift(1))
CPI["CPI lag"]=CPI.CPI.shift(1)
CPI["USD lag"]=CPI.usd.shift(1)
CPI["IPC lag"]=CPI.IPC.shift(1)
CPI=CPI.drop(["usd","IPC"],axis=1)
#Construcción train y test
Car=CPI.columns.values.tolist()
Car=Car[1:len(Car)]
pred="CPI"
test=CPI.loc[41:48]
train=CPI.loc[0:40]
X_train,y_train=train[Car],train[pred]
X_test,y_test=test[Car],test[pred]
#Entrenamiento del Modelo
reg=xg.XGBRegressor(n_estimators=1100,learning_rate=0.01)
reg.fit(X_train,y_train,eval_set=[(X_train,y_train),(X_test,y_test)],e
val_metric=["mae","rmse"])
```

```
#Resultados
per=[202106,202107,202108,202109,202110,202111,202112]
df=pd.DataFrame({"Month":per,"y_pred":predictions,"y_test":y_test.toli
st()})
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(df["y_test"], df["y_pred"], squared=False)
rmse=np.sqrt(mse)
mse
```

LSTM Univariado (Lazerri, 2021) **(Python)**

```
#Librerias
import os
import warnings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import datetime as dt
from collections import UserDict
from sklearn.preprocessing import MinMaxScaler
from IPython.display import Image
import datetime
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
#Tratamiento de datos
CPI_USD["Fecha"]=CPI_USD["Fecha"].str[:10]
CPI_USD["Fecha"]=pd.to_datetime(CPI_USD["Fecha"],format='%Y/%m/%d')
CPI_USD=CPI_USD.set_index("Fecha")
#Construcción de la Ventana
T=2
lag=1
train = CPI_USD.copy()[CPI_USD.index < test_st_data_load][['CPI']]
scaler = MinMaxScaler()
train['CPI'] = scaler.fit_transform(train)
train_shifted = train.copy()
train_shifted['y_t+1'] = train_shifted['CPI'].shift(-1)
for t in range(1, T+1):
    train_shifted[str(T-t)] = train_shifted['CPI'].shift(T-t)
y_col = 'y_t+1'
X_cols = ['CPI_t-1',
          'CPI_t']
```

```
train_shifted.columns = ['CPI_original']+[y_col]+X_cols
train_shifted = train_shifted.dropna(how='any')
y_train = train_shifted[y_col].to_numpy()
X_train=train_shifted[X_cols].to_numpy()
y_train.shape
X_train.shape
#Entrenamiento del Modelo
model = Sequential()
model.add(LSTM(8, input_shape=(T, 1)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.summary()
model_history = model.fit(X_train,
                          y_train,
                          batch_size=5,
                          epochs=10,
                          verbose=1)
plt.plot(model_history.history['loss'], label='train')
plt.legend()
plt.show()
#Procesamiento Test
test = CPI_USD.copy()[CPI_USD.index >= test_st_data_load][['CPI']]
scaler = MinMaxScaler()
test['CPI'] = scaler.fit_transform(test)
test_shifted = test.copy()
test_shifted['y_t+1'] = test_shifted['CPI'].shift(-1)
for t in range(1, T+1):
    test_shifted[str(T-t)] = test_shifted['CPI'].shift(T-t)
y_col = 'y_t+1'
X_cols = ['CPI_t-1',
          'CPI_t']

test_shifted.columns = ['CPI_original']+[y_col]+X_cols
test_shifted = test_shifted.dropna(how='any')
test_shifted.head(10)
```

```
y_test =test_shifted[y_col].to_numpy()

X_test=test_shifted[X_cols].to_numpy()

#Resultados

ev_ts_data = pd.DataFrame(ts_predictions, columns=['t+'+str(t) for t in
range(1, lag+1)])

ev_ts_data['timestamp'] = test_shifted.index

ev_ts_data = pd.melt(ev_ts_data, id_vars='timestamp',value_name='prediction',
var_name='h')

ev_ts_data['actual'] = np.transpose(y_test).ravel()

ev_ts_data[['prediction', 'actual']] =
scaler.inverse_transform(ev_ts_data[['prediction', 'actual']])

ev_ts_data.head()
```

LSTM Multivariado (Python)

```
#Librerias
import os
import warnings
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import concat
from keras.layers import concatenate
import datetime as dt
from collections import UserDict
from sklearn.preprocessing import MinMaxScaler
from IPython.display import Image
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import datetime
#Tratamiento de datos
CPI_USD["Fecha"]=CPI_USD["Fecha"].str[:10]
CPI_USD["Fecha"]=pd.to_datetime(CPI_USD["Fecha"],format='%Y/%m/%d')
CPI_USD=CPI_USD.set_index("Fecha")
CPI_USD=CPI_USD[["CPI","usd","IPC"]]
CPI_USD["week"]=pd.DatetimeIndex(CPI_USD.index).week
CPI_USD["usd change"]=np.log(CPI_USD.usd/CPI_USD.usd.shift(1))
CPI_USD["CPI change"]=np.log(CPI_USD.CPI/CPI_USD.CPI.shift(1))
CPI_USD["IPC change"]=np.log(CPI_USD.IPC/CPI_USD.IPC.shift(1))
#Ventana de datos
# prepare data for lstm
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

```
# convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
	n_vars = 1 if type(data) is list else data.shape[1]
	df = DataFrame(data)
	cols, names = list(), list()
	# input sequence (t-n, ... t-1)
	for i in range(n_in, 0, -1):
		cols.append(df.shift(i))
		names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
	# forecast sequence (t, t+1, ... t+n)
	for i in range(0, n_out):
		cols.append(df.shift(-i))
		if i == 0:
			names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
		else:
			names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
	# put it all together
	agg = concat(cols, axis=1)
	agg.columns = names
	# drop rows with NaN values
	if dropnan:
		agg.dropna(inplace=True)
	return agg

# load dataset
dataset = CPI_USD
values = dataset.values
# integer encode direction
encoder = LabelEncoder()
values[:,4] = encoder.fit_transform(values[:,4])
# ensure all data is float
values = values.astype('float32')
# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
```

```
scaled = scaler.fit_transform(values)
# frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
# drop columns we don't want to predict
reframed.drop(reframed.columns[[4,5,6,8,9,10]], axis=1, inplace=True)
print(reframed.head())

#Train, test y conversion en Tensores
train_X,train_Y=reframed.values[:, :-1], reframed.values[:, -1]
values = reframed.values
n_train = 40
train = values[:n_train, :]
test = values[n_train:, :]

train_X, train_y = train[:, [0,1,2,3,5,6,7]], train[:, -4]
test_X, test_y = test[:, [0,1,2,3,5,6,7]], test[:, -4]

train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))

print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)
test_X
#Modelo
model = Sequential()
model.add(LSTM(5, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model_history = model.fit(train_X,
                          train_y,
                          batch_size=5,
                          epochs=18,
                          validation_data=(test_X, test_y),
                          verbose=1)
odel_history.history['loss'], label='train')
plt.plot(model_history.history['val_loss'], label='test')
```

```
plt.legend()
plt.show()
yhat=model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(inv_yhat,inv_y, squared=False)
rmse=np.sqrt(mse)
mse
```

SVR Univariado (Python)

```
#Librerias
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
from sklearn.svm import SVR
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
from sklearn.preprocessing import MinMaxScaler
#Arreglo de la base
CPI=CPI[["CPI"]]
#Slide Window
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
	n_vars = 1 if type(data) is list else data.shape[1]
	df = data
	cols = list()
	# input sequence (t-n, ... t-1)
	for i in range(n_in, 0, -1):
		cols.append(df.shift(i))
	# forecast sequence (t, t+1, ... t+n)
	for i in range(0, n_out):
		cols.append(df.shift(-i))
	# put it all together
	agg = concat(cols, axis=1)
	# drop rows with NaN values
	if dropnan:
		agg.dropna(inplace=True)
	return agg.values
```

```
from pandas import concat
data=series_to_supervised(CPI)
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(data)
#Data de train y test
# split a univariate dataset into train/test sets
def train_test_split(data, n_test):
        return data[:-n_test, :], data[-n_test:, :]
train,test=train_test_split(scaled, 5)
train = np.asarray(train)
trainX, trainy = train[:, :-1], train[:, -1]
test = np.asarray(test)
testX, testy = test[:, :-1], test[:, -1]
regressor = SVR(kernel='rbf')
regressor.fit(trainX,trainy)
y_pred = regressor.predict(testX)
pred=np.reshape(y_pred,(5,1))
scaler.inverse_transform(np.concatenate((pred,testX),axis=1))
from sklearn.metrics import mean_squared_error

rms = mean_squared_error(df[0], df[1], squared=False)
rms
```

SVR Multivariado (Python)

```
#Librerias
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
from sklearn.svm import SVR
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split
from matplotlib import pyplot
from sklearn.preprocessing import MinMaxScaler
#Transformación de datos

CPI_USD["Fecha"]=CPI_USD["Fecha"].str[:10]
CPI_USD["Fecha"]=pd.to_datetime(CPI_USD["Fecha"],format='%Y/%m/%d')
CPI_USD=CPI_USD.set_index("Fecha")
CPI_USD=CPI_USD[["CPI","usd","IPC"]]
CPI_USD["week"]=pd.DatetimeIndex(CPI_USD.index).week
CPI_USD["usd change"]=np.log(CPI_USD.usd/CPI_USD.usd.shift(1))
CPI_USD["CPI change"]=np.log(CPI_USD.CPI/CPI_USD.CPI.shift(1))
CPI_USD["IPC change"]=np.log(CPI_USD.IPC/CPI_USD.IPC.shift(1))
#Slide Window

# convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
      n_vars = 1 if type(data) is list else data.shape[1]
      df = DataFrame(data)
      cols, names = list(), list()
      # input sequence (t-n, ... t-1)
      for i in range(n_in, 0, -1):
            cols.append(df.shift(i))
            names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
```

```
	# forecast sequence (t, t+1, ... t+n)
	for i in range(0, n_out):
		cols.append(df.shift(-i))
		if i == 0:
			names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
		else:
			names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
	# put it all together
	agg = concat(cols, axis=1)
	agg.columns = names
	# drop rows with NaN values
	if dropnan:
		agg.dropna(inplace=True)
	return agg

# load dataset
dataset = CPI_USD
values = dataset.values
# integer encode direction
encoder = LabelEncoder()
values[:,4] = encoder.fit_transform(values[:,4])
# ensure all data is float
values = values.astype('float32')
# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)
# frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
# drop columns we don't want to predict
reframed.drop(reframed.columns[[4,5,6,8,9,10]], axis=1, inplace=True)
print(reframed.head())
#Train y Test
train_X,train_Y=reframed.values[:, :-1], reframed.values[:, -1]
values = reframed.values
n_train = 40
```

```
train = values[:n_train, :]
test = values[n_train:, :]

train_X, train_y = train[:, [0,1,2,3,5,6,7]], train[:, -4]
test_X, test_y = test[:, [0,1,2,3,5,6,7]], test[:, -4]
#Modelo
regressor = SVR(kernel='rbf')
regressor.fit(train_X,train_y)
#Resultados
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
#MSE
from sklearn.metrics import mean_squared_error

mse = mean_squared_error(inv_yhat,inv_y, squared=False)
rmse=np.sqrt(mse)
mse
```