uc3m | Universidad Carlos III de Madrid

Máster Universitario en Ciencias Actuariales y Financieras
2020-2022

Trabajo Fin de Máster

"Predicción de la reserva total de siniestros de un seguro de no vida mediante modelos estocásticos versus algoritmos de machine learning"

Yesenia Canessa Poma

Tutor/es

José Miguel Rodríguez-Pardo

Jesús Ramón Simón del Potro

Madrid, 2022

ÍNDICE DE CONTENIDO

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación

En toda compañía de seguros la estimación de la reserva de siniestros es una de las tareas centrales que el actuario debe desempeñar, con la misión de obtener una estimación lo más confiable posible, puesto que este dinero representa las obligaciones que la compañía ha adquirido con sus asegurados. De no ser así, se pone en riesgo su solvencia y en consecuencia también el cumplimiento de sus obligaciones para con los asegurados, lo que puede acarrear incluso su intervención por parte de la autoridad supervisora.

Con este fin, en la literatura actuarial se describen diversas metodologías de estimación de reservas de siniestros, siendo quizás la más popular la metodología determinista Chain-Ladder para los seguros de no vida. Estas metodologías tienen en común que utilizan como input y punto de partida los siniestros agregados, en forma de un triángulo, ordenados por periodo de ocurrencia y desarrollo. Esta agregación de los datos genera a su vez una de las mayores críticas respecto a estos métodos, puesto que se dice que al agregarlos se está perdiendo información valiosa que puede ser útil para predecir de manera más precisa el costo final y por ende la reserva de siniestros.

Ante esta crítica, surge una alternativa natural, utilizar las técnicas de *machine learning* en la estimación de la reserva de siniestros. Estas técnicas tienen la ventaja de emplear todo la información disponible de manera desagregada, incluyendo otras variables distintas a los importes pagados y/o ya reservados, lo cual tiene el potencial de mejorar la predicción efectuada.

No obstante, si bien estos algoritmos se han popularizado en diversos campos, incluyendo el sector de seguros, principalmente en la detección de fraude y en las estrategias de venta cruzada identificando potenciales clientes o fidelizando a los ya existentes, puesto que han demostrado capturar interacciones complejas no lineales entre variables, usualmente estos modelos en la práctica no tienen como objetivo establecer la relación causal entre las variables lo cual puede dificultar la autorización de su utilización como modelo de estimación de reservas por parte de la autoridad supervisora[1].

[1] Blier-Wong, C.; Cossette, H.; Lamontagne, L.; Marceau, E. Machine Learning in P&C Insurance: AReview for Pricing and Reserving. Risks 2021, 9, 4.
Disponible en: *https://dx.doi.org/10.3390/risks9010004*

1.2. Objetivo

Este trabajo tiene como objetivo la comparación de las estimaciones de la reserva total de siniestros, incluyendo la reserva de siniestros pendientes de liquidación y de pago, y la reserva IBNR, obtenidas mediante la aplicación de los algoritmos de *machine learning* más utilizados actualmente como el Modelo lineal generalizado – GLM, k-Vecinos más cercanos – kNN, Bosque aleatorio, Gradient boosting machine – GBM y Extreme gradient boosting - XGB y dos de las metodologías estocásticas más clásicas basadas en Chain-Ladder, Mack con distribución libre y Bootstrap.

2. RESERVA TOTAL DE SINIESTROS

2.1. Proceso de la reserva de siniestros

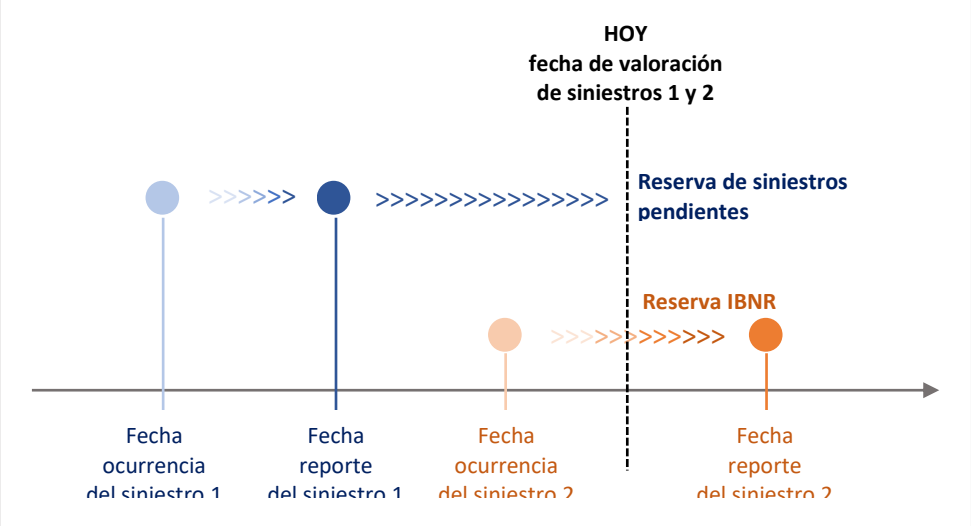
Entre las obligaciones que adquieren las empresas de seguros, se encuentran las indemnizaciones que deben pagar a sus asegurados por las coberturas contratadas, en los términos que se indiquen en las pólizas de seguros, cuando un siniestro ha ocurrido, por ejemplo, la detección de una enfermedad grave, la muerte de un asegurado, un accidente vehicular, el incendio de una fábrica, una indemnización por responsabilidad civil del asegurado, entre otros.

En particular para los seguros de no vida, la empresa de seguros tiene la obligación de valorar, constituir y mantener en todo momento el monto total, ya sea cierto o estimado, que pagará como indemnizaciones a los asegurados por los siniestros que han ocurrido hasta una fecha de valoración dada, incluyendo aquellos que a dicha fecha aún no han sido reportados a la aseguradora pero que si han ocurrido. A este monto total se le conoce como la reserva de siniestros.

Cuando se realizan pagos parciales del siniestros y/o este es liquidado completamente, esta reserva es disminuida en una cuantía equivalente al pago efectuado. Adicionalmente, el importe de la reserva de siniestros también puede variar porque han ocurrido siniestros adicionales y/o porque los importes estimados sufren variaciones con el paso del tiempo debido a que existe una mayor información del siniestro y por lo tanto se realiza una mejor estimación del importe final a liquidar o pagar; no obstante, se espera que desde su constitución inicial esta reserva sea lo más parecido al monto final a pagar a fin de conocer en todo momento las obligaciones reales (en términos monetarios) que la empresa de seguros mantiene con sus asegurados. Es por esto último que es importante disponer de métodos actuariales que estimen adecuadamente esta reserva de siniestros.

Los componentes de la reserva de siniestros son 2, la reserva de siniestros pendientes de liquidación o pago (en adelante RSPLP) y la reserva de siniestros ocurridos pero no reportados (en adelante IBNR, por sus siglas en inglés). Respecto a la primera, esta reserva corresponde a los siniestros que ocurrieron y que fueron reportados antes de la fecha de valoración y que a dicha fecha se encuentran pendientes de pago total o parcialmente. Por otro lado, la reserva IBNR corresponde a los siniestros que han ocurrido antes de la fecha de valoración pero que aún no han sido reportados o comunicados a la aseguradora, por lo cual su valor siempre es una estimación.

En el Gráfico 1 se muestra un diagrama del proceso descrito en los párrafos anteriores.



Elaboración propia

Gráfico 1. Proceso de la reserva de siniestros

2.2. Modelos clásicos de estimación de reservas de siniestros

Los métodos clásicos de valoración de la reserva de siniestros de los seguros de no vida pueden ser deterministas o estocásticos.

Los métodos deterministas son aquellos que no contemplan un componente de aleatoriedad en su desarrollo por lo que, al aplicar un método determinista sobre un conjunto de datos, el resultado siempre será el mismo. Por esta razón mediante estos métodos la volatilidad de la predicción de la reserva de siniestros es cero lo cual no proporciona información respecto al error asociado a la estimación efectuada. Los métodos más comunes son Chain-ladder, Bornhuetter-Ferguson y Loss ratio.

Por otro lado, los métodos estocásticos si contemplan un componente de aleatoriedad en su desarrollo por lo que, aunque la información de entrada sea la misma, al aplicar el método estocástico siempre se obtendrá un resultado diferente de la reserva de siniestros estimada. Esto permite medir la volatilidad de las estimaciones y obtener un intervalo de confianza. Los métodos más comunes son Mack y Bootstrap.

En la siguiente sección revisamos brevemente la teoría respecto a estos dos últimos métodos estocásticos, puesto que son lo que suelen presentar mejores estimaciones que los métodos deterministas y son ampliamente utilizados en la práctica actuarial.

3. MODELOS ESTOCÁSTICOS

Mack y Bootstrap, ambos métodos se basan en la técnica Chain-Ladder. Esta técnica utiliza como punto de partida la información de los importes pagados y/o reservados de los siniestros; esta información se ordena en forma de un triángulo llamado triángulo de desarrollo o triángulo *run-off*, donde cada fila agrupa los importes de los siniestros ocurridos en un mismo periodo (anual, trimestral, mensual, etc.); por otro lado, cada columna del triángulo representa el desarrollo del siniestro, es decir, el importe pagado y/o reservado durante cada periodo de desarrollo (anualmente, trimestralmente, mensualmente, etc.).

El triángulo de desarrollo puede ser incremental o acumulado. De acuerdo con el Gráfico 2, en un triángulo incremental cada $p_{i,j}$ representa el monto pagado de los siniestros ocurridos en el periodo i y desarrollo j, o bien, si nos fijamos en el triángulo de la derecha, cada r representa el monto pagado más el importe de las reservas de los siniestros ocurridos en el periodo i y el desarrollo j; a esto último se le conoce como incurridos.

Año de origen	Desarrollo anual				
	1	2	3	4	5
1	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$	$p_{1,5}$
2	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	
3	$p_{3,1}$	$p_{3,2}$	$p_{3,3}$		
4	$p_{4,1}$	$p_{4,2}$			
5	$p_{5,1}$				

Año de origen	Desarrollo anual				
	1	2	3	4	5
1	$r_{1,1}$	$r_{1,2}$	$r_{1,3}$	$r_{1,4}$	$r_{1,5}$
2	$r_{2,1}$	$r_{2,2}$	$r_{2,3}$	$r_{2,4}$	
3	$r_{3,1}$	$r_{3,2}$	$r_{3,3}$		
4	$r_{4,1}$	$r_{4,2}$			
5	$r_{5,1}$				

Elaboración propia

Gráfico 2. Triángulo incremental de pagos y de incurridos

Por otro lado, en un triángulo de desarrollo acumulado cada $P_{i,j}$ o $R_{i,j}$ representa el monto pagado o incurrido de los siniestros ocurridos en el periodo i hasta el desarrollo j.

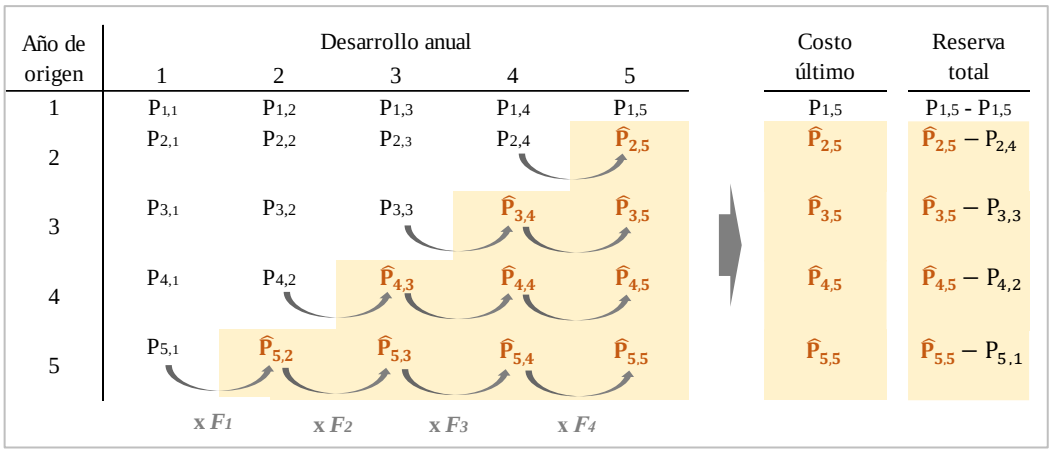
Año de origen	Desarrollo anual				
	1	2	3	4	5
1	$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	$P_{1,4}$	$P_{1,5}$
2	$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	$P_{2,4}$	
3	$P_{3,1}$	$P_{3,2}$	$P_{3,3}$		
4	$P_{4,1}$	$P_{4,2}$			
5	$P_{5,1}$				

Año de origen	Desarrollo anual				
	1	2	3	4	5
1	$R_{1,1}$	$R_{1,2}$	$R_{1,3}$	$R_{1,4}$	$R_{1,5}$
2	$R_{2,1}$	$R_{2,2}$	$R_{2,3}$	$R_{2,4}$	
3	$R_{3,1}$	$R_{3,2}$	$R_{3,3}$		
4	$R_{4,1}$	$R_{4,2}$			
5	$R_{5,1}$				

Elaboración propia

Gráfico 3. Triángulo acumulado de pagos y de incurridos

El objetivo de la técnica Chain-Ladder es estimar las obligaciones futuras de los siniestros, conocido también como costo último o *ultimate*, para ello se debe completar las celdas del triángulo acumulado inferior. Estas celdas se completarán multiplicando los valores de la última diagonal del triángulo acumulado superior (ejemplo, $P_{1,5}$, $P_{2,4}$, …, $P_{5,1}$ o $R_{1,5}$, $R_{2,4}$, …, $R_{5,1}$) por un factor de desarrollo, FD_j, que será el mismo para cada columna o desarrollo j, hasta llegar al último desarrollo o columna del triángulo que representará el costo último de los siniestros, tal como se muestra en el Gráfico 3 para el caso de un triángulo acumulado de pagos. Las celdas pintadas son los valores estimados.



Elaboración propia

Gráfico 4. Triángulo inferior, costo último y reserva total estimada

El factor de desarrollo F_j se calcula según la siguiente ecuación, donde n representa el número de periodos, que en el ejemplo dado es $n = 5$ años.

$$F_j = \frac{\sum_{i=1}^{i=n-j} P_{i,j+1}}{\sum_{i=1}^{i=n-j} P_{i,j}},$$

La razón por la cual se considera que el factor de desarrollo, F_j, es el mismo para cada fila o periodo de ocurrencia, se debe al supuesto principal del método Chain-Ladder, este es que, los pagos o montos incurridos de los siniestros en el futuro siguen el mismo patrón que en el pasado.

El costo último estimado o *ultimate*, representa el total de las obligaciones por los siniestros ocurridos en cada periodo del triángulo considerado, ya sea calculado con los importes de pagos o de incurridos. Si a este costo último le restamos todos los pagos efectuados hasta la fecha de cálculo (fecha de valoración) obtenemos la reserva total de siniestros; y si a esta reserva total de siniestros le restamos la reserva

de siniestros pendientes de liquidación o pago ($RSPLP$), obtenemos la reserva IBNR ($IBNR$).

$$Reserva\ total\ siniestros = Costo\ último - Pagos$$

$$Reserva\ total\ siniestros = RSPLP\ +\ IBNR$$

Por otro lado, también se pueden calcular factores de desarrollo individuales (f_j), que son útiles para el modelo de Mack. Estos factores también se obtienen a partir del triángulo de desarrollo acumulado representado en el Gráfico 3, a través de la siguiente relación:

$$f_j = \frac{P_{i,j}}{P_{i,j-1}} \quad \text{ó} \quad \frac{R_{i,j}}{R_{i,j-1}}$$

3.1. Modelo de Mack

El método de Mack utiliza el mismo triángulo de desarrollo que el método Chain-Ladder. Se calculan los factores de desarrollo, F_j, y se estiman sus errores estándar, $\hat{\sigma}_j$, sin asumir a priori una distribución de probabilidad en particular, según la siguiente fórmula dada por Mack en el año 1993.

$$\hat{\sigma}_j^2 = \frac{1}{n-j-2}\sum_{i=1}^{n-j} P_{i,j}\left(\frac{P_{i,j+1}}{P_{i,j}} - F_j\right)^2$$

Una vez estimado el error estándar para cada desarrollo j, mediante la siguiente ecuación se puede estimar el error cuadrático medio (MSE por su siglas en inglés) para cada valor estimado $\hat{P}_{i,J}$. El MSE es una medida de la incertidumbre asociada a una estimación efectuada. Cabe señalar que se trata de un MSE condicionado a la información conocida (del triángulo superior) $\mathcal{D}_I$.

$$\widehat{MSE}_{\hat{P}_{i,J}|\,\mathcal{D}_I} = \hat{P}_{i,J}^2 \sum_{j=J-i}^{J-1}\left[\frac{\hat{\sigma}_j^2}{F_j^2}\left(\frac{1}{\hat{P}_{i,j}} + \frac{1}{\sum_{k=1}^{I-j-1} P_{k,j}}\right)\right]$$

3.2. Modelo Bootstrap

Partiendo de la última diagonal del triángulo superior acumulado y con los factores de desarrollo calculados en el modelo Chain-Ladder, F_j, se reconstruye un triángulo superior, llamado triángulo ajustado acumulado.

$$\breve{P}_{i,j-1} = \frac{\breve{P}_{i,j}}{F_{j-1}}$$

Con base a estos últimos valores se obtienen los elementos de un triángulo ajustado incremental.

$$\breve{p}_{i,j} = \breve{P}_{i,j} - \breve{P}_{i,j-1}$$

Luego, se calculan los residuos de Pearson, $\hat{r}_{i,j}$, que es la diferencia entre los valores del triángulo incremental original y los valores del triángulo ajustado incremental, todo esto dividido entre los valores del triángulo ajustado incremental.

$$\hat{r}_{i,j} = \frac{p_{i,j} - \breve{p}_{i,j}}{\sqrt{\breve{p}_{i,j}}}$$

Una vez obtenidos los residuos de Pearson, se procede a remuestrearlos (con repetición) para así obtener un nuevo triángulo ajustado acumulado mediante el procedimiento explicado anteriormente, pero aplicado inversamente. Con el nuevo triángulo ajustado acumulado se calcula el costo último o *últimate* y por consiguiente la reserva total de siniestros y/o la reserva IBNR según el método Chain-Ladder.

El demuestra de los residuos de Pearson se repite un número suficiente de veces, por ejemplo 10.000 o 50.000, obteniendo de esta manera una distribución empírica del costo último o de la reserva total de siniestros. Con la distribución obtenida ya es posible calcular el error de estimación.

4. ALGORITMOS DE MACHINE LEARNING (ML)

4.1. Modelos lineales generalizados (GLM)

A diferencia del modelo de regresión lineal donde la variable respuesta debe distribuirse como una normal, los modelos GLM permiten otras distribuciones de la variable respuesta. Por otro lado, estos modelos también incluyen una función de enlace, $g(\ )$, que relaciona la media de la variable respuesta Y con una combinación lineal de los predictores X_i, como se muestra a continuación:

$$g(\mu) = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k$$

donde $\mu = E(Y|X)$ es la media condicional de la variable respuesta Y .

La distribución de la variable respuesta Y debe pertenecer a la familia de las distribuciones exponenciales como la Normal, Binomial, Poisson, Gamma, Gaussiana inversa y Binomial negativa. Las funciones de densidad de estas distribuciones comparten la siguiente expresión general:

$$f(y, \theta, \phi) = e^{\left[\frac{y\theta - b(\theta)}{\phi} + S(y,\phi)\right]}$$

donde:
θ es una función de la media de la variable respuesta μ.
ϕ es un parámetro llamado de escala.
$b(\theta)$ y $S(y, \phi)$ son funciones de θ e y y ϕ respectivamente.

Por otro lado, una vez definida la distribución de Y, se debe seleccionar la función de enlace $g(\ )$. Esta debe ser monotónica para poder invertirla y así despejar la media de la variable respuesta, μ, una vez que los parámetros β_i han sido estimados usualmente mediante el método de máxima verosimilitud.

Una función de enlace común es la función canónica; es decir, igual a una función de la media de la variable respuesta la cual ya habíamos definido anteriormente como θ.

$$g(\mu) = \theta$$

Algunas funciones de enlace canónicas más comunes son:

- Identidad: $g(\mu) = \theta = \mu$. Esto es cuando la distribución de Y es Normal.
- Logit: $g(\pi) = \theta = ln\left[\frac{\pi}{(1-\pi)}\right]$. Esto cuando la distribución de Y es Binomial.
- Logaritmo natural: $g(\mu) = \theta =$ ln (μ). Esto es cuando la distribución de Y es Poisson.

- Inversa: $g(\mu) = \theta = \frac{1}{\mu}$. Esto es cuando la distribución de Y es Gamma.
- Inversa cuadrada: $g(\mu) = \theta = \frac{1}{\mu^2}$. Esto es cuando la distribución de Y es Gaussiana inversa.

Finalmente:

$$g(\mu) = \theta = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_k X_k$$

Por ejemplo, si $\theta = \frac{1}{\mu}$, entonces:

$$\frac{1}{\mu} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_k X_k$$

$$\mu = (\hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_k X_k)^{-1}$$

4.2. k Vecinos más cercanos (k-NN)

Este método conocido en inglés como *k-Nearest Neigbors* pertenece al grupo de algoritmos no paramétricos denominado regresiones locales, puesto que la idea es predecir a partir de las observaciones más cercanas a la posición de la predicción. Este algoritmo k-NN asume que la media condicional, que es la predicción óptima, es constante localmente.

En particular, para clasificar una variable respuesta para un conjunto de predictores (x_0), se identifican los K puntos vecinos. Se considera como puntos vecinos a aquellas observaciones cuyos predictores x son más cercanos a x_0. Por ejemplo, serán $x_1, x_2, \ldots, x_k$. Se observan los *outputs* de dichos predictores, $y_1, y_2, \ldots, y_k$. Cada *output* pertenece a una clase, por ejemplo: sí y no; verdadero y falso; pequeño, mediano y grande; etc. Luego, se estima la proporción (probabilidad condicional) de cada clase en ese conjunto de K datos. La clase que tenga la proporción o probabilidad más grande (es decir, que sea la moda) será la clase que se le asignará al conjunto de predictores x_0 (y por la tanto esa será la predicción)

$$\mathbb{P}\,(Y = clase\ j \mid x_0) \approx \frac{\sum 1_{\{y_i = clase\ j\}}}{K}$$

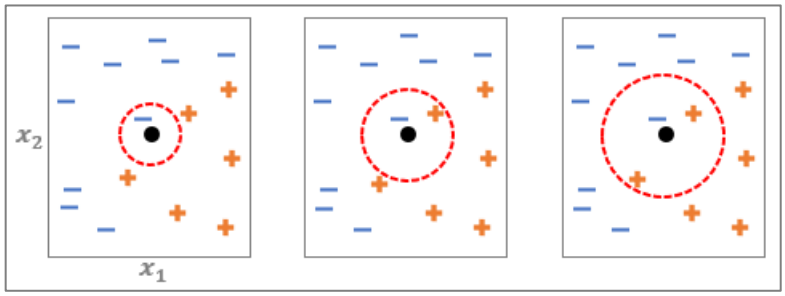
donde, K es un entero positivo que es preespecificado, que se puede calcular con validación cruzada, a fin de determinar el valor de K que genera el menor error de predicción.

Para determinar cuáles son los K puntos vecinos a x_0, se hace uso del concepto de "distancia Euclidiana". Los K puntos vecinos a x_0 serán aquellos que tienen la menor distancia Euclidiana a x_0:

$$dist(x_0, x_1) = \sqrt{(x_{01} - x_{11})^2 + (x_{02} - x_{12})^2 + \cdots + (x_{0p} - x_{1p})^2}$$

Donde, x_0, x_1 son 2 observaciones de las n observaciones del *training data*. Cada una de estas observaciones es el conjunto de los p predictores que estamos considerando, por ejemplo: $x_{11}, x_{12}, x_{13}, \ldots, x_{1p}$

Aquí se muestra un gráfico ilustrativo cuando $K = 1$ (izquierda), $K = 2$ (centro) y $K = 3$ (derecha). Cuando $K = 1$, solo hay un vecino con variable respuesta o clase (-) por lo tanto, la clase que se le asignará al conjunto de predictores objetivo x_0 será (-). En cambio, cuando $K = 3$, hay dos vecinos (+) y uno (-), entonces la clase asignada será (+). Cuando $K = 2$, hay un vecino de la clase (+) y otro vecino de la clase (-). En este caso, no hay una respuesta clara.



Fuente: Statistical risk modeling study manual by Actex.
Elaboración propia.

Gráfico 5. Ilustración del algoritmo k vecinos más cercanos

Esta técnica se puede usar tanto para clasificación como para regresiones. Cuando se trate de una variable respuesta continua, en lugar de estimar la clase mayoritaria entre el conjunto de vecinos, se calcula el promedio de las respuestas, siendo este valor el que se signará como respuesta para el conjunto de predictores objetivos x_0.

4.3. Árboles de decisión

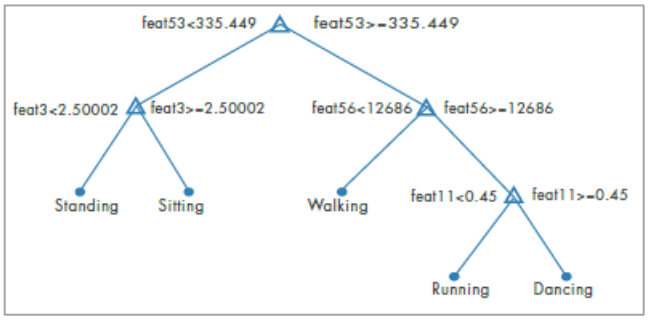
Este método no paramétrico fue desarrollado (en la década de los 70) como alternativa a los métodos clásicos de regresión lineal que están sujetos a hipótesis de linealidad y normalidad u otra distribución de probabilidad de la variable dependiente o respuesta en el caso de usar una técnica como GLM. Por lo que las ventajas principales de esta técnica son:

- Muy sencilla de utilizar e interpretar.
- No requiere supuestos.
- Es útil para relaciones no lineales.

No obstante, a menudo esta técnica tiene una baja calidad predictiva, por lo que en los últimos años se han desarrollado los métodos combinados o de ensamble con un alto poder predictivo (*Bagging* y *Boosting*), pero que tienen como base principal el árbol de decisión, por lo que se considera éste como un método básico y fundamental de *machine learning*.

Esta técnica consiste en segmentar el conjunto de predictores en regiones simples, mediante alguna regla que se aplica a uno o varios predictores. El árbol se inicia con un nodo que representa a toda la muestra o datos de entrenamiento, del que se originan 2 ramas que dividen la muestra en dos subconjuntos, cada uno representado por un nuevo nodo. Este proceso se repite un número finito de veces hasta obtener los nodos terminales (hojas del árbol), que son los que se emplean para la predicción. Si se trata de un problema de regresión, la predicción consiste en obtener el promedio de todos los elementos en cada nodo terminal; mientras que si se trata de un problema de clasificación la predicción será la moda.

A continuación, se muestra un ejemplo gráfico de cuando la muestra de aprendizaje o datos de entrenamiento contiene 3 predictores: $x_1 = feat53, x_2 = feat56$, $x_3 = feat3$ y $x_4 = feat11$



Fuente: Mathworks: machine learning with Matlab

Gráfico 6. Ilustración del algoritmo árbol de decisión

El árbol se construye dividiendo los datos de entrenamiento en J regiones: R_1, R_2, …, R_J, para cada una de las cuales se calcula la media de la variable respuesta y de las observaciones que caen en cada región: c_1, c_2, …, c_J. Esto se hace si estamos en un problema de regresión. Si es un problema de clasificación, c_j representará la moda.

La elección de las J regiones se hace con base al error cuadrado (SRR). Se escogerá el grupo de J regiones que tengan el mínimo SRR. Sin embargo, si se procede de esta manera, la cantidad de grupos de J regiones posibles es demasiado grande, lo

que lo hace inviable. En ese sentido, usualmente se utiliza el método CART que es una simplificación:

Se elige un predictor X_j y un punto de corte s, con los cuales se definen dos hiperplanos $R_1 = \{X \mid X_j \leq s\}$ y $R_2 = \{X \mid X_j > s\}$. La elección del predictor y el punto de corte será aquellos que minimicen el SRR:

$$SRR_{min} = \sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2 + \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2$$

Se repite el mismo proceso en cada una de las 2 regiones R_1 y R_2, así sucesivamente, hasta alcanzar un criterio de parada predefinido. Usualmente este criterio consiste en exigir un número mínimo de observaciones en los nodos finales.

Si el árbol que se obtiene es demasiado grande, sobreajustará los datos dotando de mucha variación al modelo, además que se tratará de un modelo complejo poco intuitivo. El árbol modelará las observaciones del conjunto de entrenamiento demasiado bien, muy ajustado a estos únicos datos. Esto genera un problema cuando ese árbol se utiliza con otros datos (por ejemplo, un conjunto de datos de entrenamiento ligeramente diferente al original), una ligera variación de los datos provocará un cambio importante en el valor de la predicción. Entonces, cada vez que ingresemos un conjunto de datos nuevos, no muy diferentes entre ellos, obtendremos predicciones muy diferentes entre sí; es decir, habremos elaborado un modelo inestable, con una varianza grande.

Si el árbol es demasiado pequeño, tendrá menor varianza y por lo tanto será un modelo más estable, de fácil interpretación, pero tendrá un mayor sesgo. El sesgo se refiere a que el modelo no se ajusta totalmente a los datos con los que fue construido (datos de entrenamiento), por lo que el modelo puede ser muy diferente al modelo real. Por ejemplo, podemos entenderlo de esta manera: si el modelo, en lugar de un árbol de decisión, fuera una función de regresión lineal. Esta regresión tiene unos parámetros reales, que son estimados mediante un método (mínimos cuadrados ordinarios). Si el modelo tiene mucho sesgo quiere decir que los parámetros estimados se alejan demasiado de los valores reales, lo cual también redundará en predicciones inexactas, pero más estables.

Todas las técnicas, algoritmos o modelos en *machine learning* o en general, se enfrentan a estos dos problemas: varianza versus sesgo (se le conoce como el *bias – variance trade-off*). Todo modelo con un sesgo pequeño tendrá una varianza grande, y viceversa. El sesgo disminuye cuando el modelo se hace más complejo o flexible; es decir, cuando utiliza demasiados predictores o en nuestro caso particular del árbol, cuando es demasiado profundo (tiene muchos nodos). Esto genera una baja capacidad predictiva porque los resultados variarán demasiado según el

conjunto de datos de entrenamiento que se use, entonces se convierte en un modelo poco fiable. Si elegimos un modelo con un sesgo grande, muy probablemente este modelo no refleje realmente el comportamiento de los datos, arrojando predicciones incorrectas, haciendo que el modelo también sea poco fiable. Por lo tanto, siempre se debe buscar un equilibrio entre sesgo y varianza.

4.4. Bagging

Bagging es un algoritmo combinado basado en la técnica *bootstrap*. Como se explicó en el apartado anterior, el tamaño del árbol de decisión implica una mayor o menor varianza en las predicciones, por lo tanto, en la efectividad del modelo. El *Bagging* surge (en la década de los 90) como respuesta a este problema; en particular, cuando el árbol es grande o muy profundo (muchos nodos), este tendrá una varianza grande con predicciones inestables. Una forma de estabilizar el modelo y por lo tanto reducir la varianza es, si contáramos con muchas muestras en lugar de tener una sola predicción, podríamos hacer un modelo con cada muestra y usar el promedio como predicción final. Esto se logra con el remuestreo o *bootstrap*. La muestra original se remuestrea (con repetición) muchas veces, dando origen a nuevas muestras que pueden contener o no todas las observaciones. Este proceso reduce la varianza significativamente.

Otra ventaja del *Bagging* es que, como se ha señalado una muestra *bootstrap* puede contener muchas observaciones repetidas y por ello, en promedio, sólo utiliza en promedio 2/3 de los datos. En consecuencia, las observaciones que no son utilizadas en la construcción del árbol se pueden utilizar para probar el modelo; es decir como datos de prueba. De esta manera no se requiere separar los datos en un conjunto de datos de entrenamiento y conjunto de datos de prueba, o utilizar el método de validación cruzada para probar la efectividad de las predicciones del modelo.

4.5. Bosque aleatorio

Esta técnica conocida en inglés como *Random forest* fue creada para mejorar la precisión de las predicciones resultantes del *Bagging*, puesto que con esta última estrategia es muy probable que algunos árboles contengan datos similares por lo que dichos árboles están correlacionados. *Random forest* busca construir árboles menos correlacionados entre sí; esto lo logra seleccionado una muestra de variables predictoras usualmente igual a la raíz cuadrada del total de variables predictoras si se trata de un problema de clasificación o un tercio si es un problema de regresión. Solo de esta muestra de predictores se seleccionará una variable predictora en cada corte (nodo) del árbol.

4.6. Boosting

El *Boosting* es una técnica que incrementa la varianza cuando el árbol es poco profundo (pequeño o pocos nodos) y por lo tanto las predicciones que genera son más estables (poca varianza) pero no necesariamente las más precisas (sesgo grande). Esta técnica fue desarrollada para impulsar métodos con poca capacidad predictiva, como los árboles de decisión.

La técnica inicial fue desarrollada para problemas de clasificación, a partir de 1984, pero solo en 1996 se desarrolló un algoritmo efectivo llamado *AdaBoost*, que se aplica a problemas de clasificación dicotómicos (solo dos categorías de la variable respuesta). Luego hubo muchos intentos para obtener un algoritmo que clasifique no solo problemas con dos categorías, además que pueda usarse para variables respuesta continuas. En el año 2001 se desarrolló el algoritmo *Gradient Boosting Machine* (GBM) que logra estos dos objetivos. El año siguiente, el mismo autor de GBM presentó una ligera mejora de su algoritmo, al cual llamó *Stochastic Gradient Boosting* (SGB). Posteriormente, en 2016 se desarrolló el algoritmo *Extreme Gradient Boosting* (XGB). Todos estos algoritmos actualmente son los más utilizados, junto con el *Bagging*.

- **AdaBoost**

Este algoritmo se aplica a problemas de clasificación con 2 únicas salidas utilizando un árbol de decisión pequeño o poco profundo.

1. A cada elemento del conjunto de datos de entrenamiento (n elementos) se le asigna un peso inicial igual a $w_i = \frac{1}{n}$.

2. Se construye un primer modelo b (árbol de decisión) con las observaciones iniciales y se calcula el error de clasificación.

$$error_b = \frac{\sum_{i=1}^{n} w_i \times 1_{\{y_i \neq \hat{y_i}\}}}{\sum_{i=1}^{N} w_i}$$

3. Luego se calcula $s_b = ln\,(\frac{1-error_b}{error_b})$. Mientras $error_b$ sea más grande, s_b será más pequeño.

4. Luego, los pesos iniciales w_i se actualizan a w_i^1. Los pesos asociados a los elementos del modelo b que fueron clasificados correctamente permanecen igual; mientras que los pesos de los elementos clasificados incorrectamente se incrementan, según la siguiente expresión:

$$w_i^1 = w_i\, e^{s_b \times 1_{\{y_i \neq \hat{y_i}\}}}$$

5. A continuación, se construye un segundo modelo (árbol de decisión) empleando las nuevas ponderaciones w_i^1. En este nuevo modelo los elementos que inicialmente no fueron clasificados correctamente; es decir con un w_i^1 mas grande, tendrán más relevancia para que esta vez, probablemente, sean clasificados correctamente.

6. Finalmente, se construye un tercer, cuarto, …, B-ésimo modelo repitiendo el mismo proceso anterior. La predicción final será el promedio ponderado de las predicciones de los B modelos, siendo el ponderador el peso de cada modelo, s_b.

4.6.1. Gradient Boosting Machine (GBM)

Como se ha señalado, este algoritmo se extiende para resolver problemas de regresión, por lo que es una generalización del algoritmo *AdaBoost.*

1. El algoritmo consiste en construir un modelo inicial b (árbol de decisión) con el conjunto de datos de entrenamiento. Luego se calcula la función de pérdida (también llamada función de costo o función de error). Esta función puede ser la suma de los cuadrados de los residuos (SRR_b)

$$Residuo_b = SRR_b = \sum_{i=1}^{n} \left[y_i - \hat{y}_i\right]^2$$

   Alternativamente, si existen valores atípicos en el conjunto de datos de entrenamiento, podría usarse como función de pérdida el error absoluto:

$$Residuo_b = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

2. Luego, se construye un nuevo modelo utilizando los residuos anteriores como respuesta; y se repite es proceso B veces.

3. Asimismo, como este modelo puede generar sobreajuste, se introduce un término de regularización conocido como *learning rate* ($\boldsymbol{\lambda}$), que limita la influencia de cada modelo en el modelo final.

4.6.2. Extreme Gradient Boosting (XGB)

Este es un algoritmo más complejo que GMB. Las modificaciones más resaltantes son:

- Utiliza una función de pérdida con una penalización por complejidad para evitar el sobreajuste. A este proceso se le conoce como regularización.
- Para la regularización se emplea la hessiana de la función de pérdida; es decir, las derivadas parciales de primer y de segundo orden.

5. EVALUACIÓN DE LA PRESICIÓN DE LOS ALGORITMOS ML

Como el objetivo del *machine learning* es la predicción, los modelos creados con base a los algoritmos descritos en la sección anterior, deben ser testeados para determinar su poder predictivo.

Esta prueba idealmente se debe realizar sobre una nueva muestra de observaciones; es decir, una muestra que no ha sido utilizada para el entrenamiento del algoritmo. A este conjunto de datos se le llama datos de prueba o de validación. No obstante, usualmente no se dispone de esta información; por ello, se aplican técnicas de remuestreo (*bootstrap*) a los datos de entrenamiento para generar este tipo de observaciones.

5.1. Conjuntos de validación

Es el método de remuestreo más sencillo, y también se le conoce en inglés como *validation set approach.* Consiste en dividir la muestra total en 2 grupos de manera aleatoria y de tamaños comparables entre sí: datos de entrenamiento y datos de prueba. Con las observaciones del conjunto de datos de validación calculamos el *test error* o también llamado *test mean squared error* (*test MSE*) como:

$$Test\ MSE = promedio\ [y_0 - \widehat{y_0}]^2$$

- **Desventaja**

Una desventaja de este método es que el *test error* dependerá en gran medida de cómo se han formado el conjunto de datos de entrenamiento y el de validación. Si se hace por ejemplo una vez más la división aleatoria de la muestra en los dos conjuntos de datos, muy probablemente se obtendrán conjuntos muy distintos a los primeros y por lo tanto se obtendrá un modelo distinto empleando los datos de entrenamiento; asimismo, también se obtendrá un *test error* muy distinto cada vez.

5.2. Validación cruzada leave-one-out

Los métodos de validación cruzada son los más populares por la ventaja que presentan respecto al método anterior: no divide aleatoriamente la muestra original.

Esta técnica LOOCV (por sus siglas en inglés) consiste en definir el conjunto de datos de entrenamiento como un subconjunto compuesto por todas las observaciones de la muestra menos una observación, por ejemplo (x_1, y_1) Dado que esta observación, (x_1, y_1), no se utiliza para entrenar el algoritmo, puede ser utilizada para testear o validar el modelo. En consecuencia, se calculará un primer error de predicción con esta observación y el modelo generado con el resto de las observaciones de la muestra:

$$MSE_1 = (y_1 - \widehat{y_1})^2$$

Seguidamente, se vuelve a formar un nuevo conjunto de datos de entrenamiento y por lo tanto se creará un nuevo modelo, pero esta vez dejando fuera la segunda observación de la muestra total; es decir, (x_2, y_2). Con esta observación se volverá a testear el nuevo modelo, obteniendo así un segundo error de predicción:

$$MSE_2 = (y_2 - \widehat{y_2})^2$$

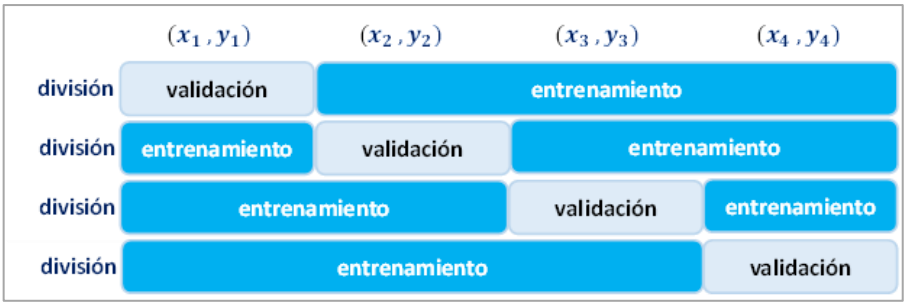
Se repetirá este procedimiento hasta dejar fuera del nuevo conjunto de datos de entrenamiento la última observación de la muestra disponible: (x_n, y_n) y con ella estimar el último error de predicción:

$$MSE_n = (y_n - \widehat{y_n})^2$$

Finalmente, el *test error* se estimará como el promedio de los n errores de predicción:

$$test\ error = CV_n = \frac{1}{n}\sum_{i=1}^{n} MSE_i = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

A continuación, a modo ilustrativo se muestra un gráfico elaborado que representa el funcionamiento de esta técnica LOOCV, asumiendo que la muestra total disponible contiene solo 4 observaciones.



Elaboración propia

Gráfico 7. Ilustración de la técnica validación cruzada leave-one-out

- **Ventaja**

Esta técnica no necesita dividir en dos grupos la muestra disponible, y los datos de entrenamiento que se van formando son similares porque solo difieren en una observación (x_i, y_i) entre submuestra y submuestra. Esto hace que en general los modelos sean prácticamente los mismos, por lo que el *test error* que se estima con base a los errores de predicción presenta poca variabilidad.

Como este método utiliza casi todas las observaciones ($(n-1)$ observaciones) de la muestra disponible para entrenar el modelo, el *test error* que se estima tiene poco sesgo, en contraste con el método anterior, *validation set approach*, que solo utiliza aproximadamente la mitad de la muestra total como *training data*, lo cual hace que el *test error* estimado tenga mucho sesgo.

- **Desventaja**

Una desventaja es que esta técnica es intensiva computacionalmente, dado que se tiene que ejecutar n modelos que de por si consisten en algoritmos que usualmente también son demandantes computacionalmente.

5.3. Validación cruzada k-fold

De manera alternativa a la validación cruzada leave-one-out, en lugar de dejar fuera una sola observación cada vez, se elige una cantidad mayor de observaciones que se dejarán fuera. Esto se logra dividiendo aleatoriamente en k grupos la muestra total disponible. Por consiguiente, se calculan k errores de predicción: MSE_1, MSE_2, …, MSE_k, y el *test error* será igual a:

$$test\ error = \ CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

Como se puede apreciar, si $k = 2$ entonces *k-fold CV* será igual que la primera técnica descrita, conjuntos de validación; mientras que si $k = n$, será igual que LOOCV. Por lo tanto, ambos métodos son un caso particular de validación cruzada *k-fold*.

6. DESCRIPCIÓN DE LA BASE DE DATOS DE SINIESTROS

6.1. Obtención de la base de datos

Para el análisis efectuado en el presente documento se recurrió a datos individuales de siniestros simulados en el software R, ante la dificultad de encontrar datos reales de siniestros de uso libre.

Los datos de siniestros simulados se obtuvieron mediante la aplicación de un algoritmo de simulación desarrollado por M. Wang[2] y M.V. Wüthrich[3], llamado "*data simulation.R*[4]". Este algoritmo es descrito en el documento *Individual claims generator for claims reserving studies: data simulation.R* publicado por los mismos autores en junio de 2022, y elaborado para el grupo de trabajo *Data Science* de la Asociación Suiza de Actuarios (SAV).

El algoritmo de simulación de siniestros "*data simulation.R*" a la vez está basado en el paquete "*SynthETIC*[5]*"* del software R desarrollarlo por B. Avanzi[6] et all en 2021 y explicado en el documento *SynthETIC: An individual insurance claim simulator with feature control* de los mismos autores.

Estos algoritmos fueron desarrollados para ofrecer al público información suficiente de siniestros individuales de un seguro de no vida y ser empleados en la evaluación de las distintas metodologías de *reserving*, puesto que los siniestros simulados contienen las características más comunes observadas en datos reales.

De acuerdo con dichos documentos, el paquete *SynthETIC* por default está calibrado para recrear siniestros con 10 años de antigüedad de una cartera anónima de seguros de responsabilidad civil de vehículos por lesiones corporales. En particular, el número de siniestros es una variable aleatoria que sigue por default una distribución de Poisson, mientras que el importe de los siniestros es otra variable aleatoria con distribución Power-normal y también son descontados con una tasa de inflación de 2% anual. El resto de los parámetros, como son: la fecha de ocurrencia, la fecha de notificación, la fecha de cierre, el número de pagos parciales, el importe de los pagos parciales y la distribución de los pagos parciales en el tiempo siguen la forma funcional que se muestran en la Tabla 1 del Anexo 1.

Así también, es importante señalar que la simulación de los siniestros incluye la recreación de dependencias naturales que se dan en un proceso real de generación de siniestros de la cartera de referencia, y además incorpora efectos de cambios normativos (u equivalentes) que impactan en el proceso de reclamación de los siniestros.

---

[2] School of risk and actuarial studies, UNSW Sydney,

[3] RiskLab, Department of Mathematics, ETH Zurich.

[4] Disponible en: https://github.com/JSchelldorfer/IndividualClaimsSimulator

[5] Disponible en: https://github.com/fstpackage/synthetic

[6] Centre for actuarial studies, Department of economics, University of Melbourne VIC 3010, Australia.

Con respecto al algoritmo *data simulation.R*, este mejora los siniestros simulados mediante *SynthETIC*, a fin de que estos reflejen de la mejor manera posible un portafolio real de siniestros del seguro de responsabilidad civil de vehículos por lesiones corporales, agregando las variables tipo de siniestro y edad del lesionado, siendo estas características relevantes en la determinación de la velocidad de reporte y de pagos de los siniestros así como, del importe de los siniestros.

Una de las ventajas de trabajar con esta base de datos simulada es que contiene información incluso de los siniestros ocurridos y no reportados, a los que se les conoce como siniestros IBNR. Esto permite que se pueda medir la precisión de los métodos de estimación de reserva de siniestros, puesto que existirá una variable con la que comparar las predicciones. En la vida real, esta información no está disponible, lo que supone una restricción al momento del análisis de las distintas metodologías.

6.2. Descripción de la base de datos

Como se ha señalado en la sección precedente, la cartera de siniestros simulados corresponde a un seguro de responsabilidad civil por lesiones corporales de vehículos. En particular se simularon 61.843 siniestros con fechas de ocurrencia entre el 01/01/2012 y el 31/12/2021. La función *data simulation.R*, genera 2 bases de datos.

En la primera base de datos, a la que llamamos siniestros.csv, cada fila representa un siniestro con 15 campos, los cuales se detallan a continuación:

- **Id:** es el código del siniestro que lo identifica de manera única.
- **Type:** es el tipo de siniestro que ha ocurrido. Existen 6 tipos de siniestros (del 1 al 6)
- **Age:** es la edad de la persona lesionada que está en el intervalo de 18 y 65 años.
- **Status:** es el status del siniestro a la fecha de corte de los datos (31/12/2021): Closed (cerrado), RBNS (pendiente o abierto) o IBNR (ocurrido y no reportado)
- **AccDate:** es la fecha de ocurrencia del siniestro, en formato aaaa/mm/dd.
- **AccMonth:** son los meses transcurridos desde la fecha de origen de los datos (enero de 2012) hasta la fecha de ocurrencia del siniestro, por lo que los valores están entre 1 y 120 meses.
- **AccWeekday:** es el día de la semana en el que ocurrió el siniestro.
- **RepDate:** es la fecha de reporte del siniestro, en formato aaaa/mm/dd. Para los siniestros con status IBNR, este campo no presenta valores (se denota como NA)
- **RepMonth:** son los meses transcurridos desde la fecha de origen de los datos (enero de 2012) hasta la fecha de reporte del siniestro, por lo que los valores están entre 1 y 120 meses. Para los siniestros con status IBNR, este campo no presenta valores (se denota como NA)

- **RepDelDays:** son los días de retraso en el reporte del siniestro contados desde la fecha de ocurrencia. Para los siniestros con status IBNR, este campo no presenta valores (se denota como NA)
- **SetMonth:** son los meses transcurridos desde la fecha de origen de los datos (enero de 2012) hasta la fecha de liquidación del siniestro, por lo que los valores están entre 1 y 120 meses. Para los siniestros con status IBNR, este campo no presenta valores (se denota como NA)
- **SetDelMonths:** son los meses de retraso en la liquidación del siniestro contados desde la fecha de reporte. Para los siniestros con status IBNR, este campo no presenta valores (se denota como NA)
- **Cumpaid:** es el monto del pago acumulado del siniestro hasta la fecha de corte de los datos (31/12/2021), en unidades de euro.
- **PayCount:** es el número de pagos parciales que se han efectuado hasta la fecha de corte de los datos (31712/2021).
- **Ultimate:** es el costo final del siniestro, en unidades de euro.

Por otro lado, la segunda base de datos, a la que llamamos "pagos.csv", contiene el detalle de los pagos parciales de los siniestros reportados, según se detalla a continuación:

- **Id:** es el código del siniestro que lo identifica de manera única.
- **EventId:** es el código del evento. Un evento puede representar un pago parcial o un cambio en el estatus del siniestro.
- **EventMonth:** son los meses transcurridos desde la fecha de origen de los datos (enero de 2012) hasta la fecha en la que ha ocurrido el evento.
- **Paid:** es el monto del pago parcial correspondiente al siniestro, en unidades de euro.
- **OpenInd:** indica si el siniestro continúa abierto (valor 1) o cerrado (valor 0) a la fecha de corte de los datos (31/12/2021).

Esta segunda base de datos, "pagos.csv", permite el cálculo de la reserva de siniestros mediante los métodos estocásticos, puesto que estos requieren el desarrollo de los siniestros que son los pagos parciales.

7. APLICACIÓN DE LOS ALGORITMOS ML

Para la aplicación de los algoritmos de *machine learning* se trabajó con la base de datos "siniestros.csv", donde cada fila representa un siniestro. Como se ha explicado anteriormente, los algoritmos deben entrenarse con un conjunto de datos y luego ser probados o testeados con otro grupo de datos.

Como se detallará en las secciones posteriores, para estimar la reserva total de siniestros (reserva de siniestros pendientes y reserva IBNR), primero se debe predecir el costo final del siniestro, al que llamamos *ultimate*; por ello, los datos de entrenamiento y de prueba se circunscriben únicamente a los siniestros cerrados (con estatus Closed) puesto que en la vida real solo se conoce el costo final del siniestro cuando este ha sido liquidado y cerrado.

Así también, las variables predictoras son aquellas variables que caracterizan a los siniestros y que se conocen a la fecha de evaluación o de corte de los datos (31/12/2021), sin incluir aquellas referidas a la liquidación del siniestro (como es el mes de liquidación del siniestro, los pagos parciales, etc.) Esto debido a que, como se estimará el *ultimate* de los siniestros reportados pendientes de liquidación (status RBNS) y no reportados (status IBNR), pues no se cuenta con información respecto a su liquidación ya que aún están pendientes.

De acuerdo con lo explicado, a continuación, se muestran los campos que se utilizan para la aplicación de las técnicas de *machine learning*.

Variable objetivo o variable respuesta:

- **Ultimate:** costo final del siniestro cerrado, en unidades de euro.

Variables predictoras:

- **Type:** tipo de siniestro cerrado. Existen 6 tipos de siniestros (del 1 al 6)
- **Age:** edad de la persona lesionada que está en el intervalo de 18 y 65 años.
- **AccYearMonth:** mes de ocurrencia del siniestro cerrado, valores entre 1 y 12.
- **AccWeekday:** es el día de la semana en el que ocurrió el siniestro.
- **RepYearMonth:** mes de reporte del siniestro cerrado, valores entre 1 y 12.
- **RepDelDays:** son los días de retraso en el reporte del siniestro cerrado contados desde la fecha de ocurrencia.

7.1. Análisis exploratorio de las variables

En esta sección se muestra las estadísticas descriptivas básicas de las variables descritas en el apartado anterior, incluyendo la interacción entre la variable respuesta u objetivo (*ultimate* o costo final del siniestro) y las variables predictoras.

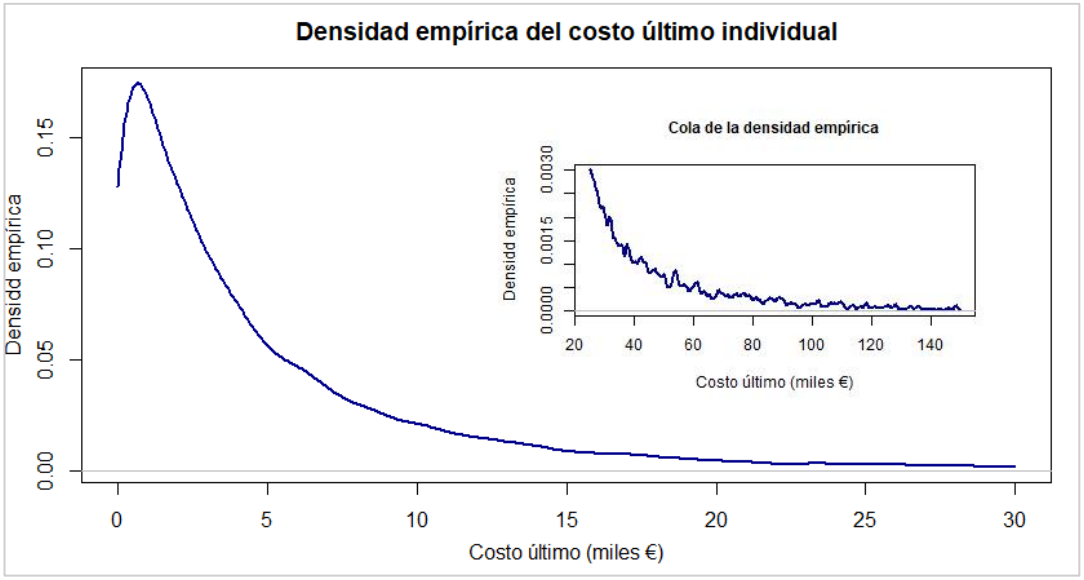
- Variable respuesta: Ultimate (costo final individual)

Tabla 1. Costo final agregado de los siniestros cerrados por año de accidente

Año de origen	Número de siniestros	Ultimate (miles €)	Promedio (miles €)	Mínimo (miles €)	Máximo (miles €)
2012	6,319	49,058	7.76	-	256
2013	6,190	47,893	7.74	-	312
2014	6,313	50,012	7.92	-	377
2015	6,091	49,362	8.10	-	419
2016	6,148	51,690	8.41	-	741
2017	6,099	51,256	8.40	-	526
2018	5,705	45,715	8.01	-	555
2019	4,950	35,288	7.13	-	374
2020	3,251	18,030	5.55	-	325
2021	794	3,448	4.34	-	149
Total	**51,860**	**401,751**	**7.75**	**-**	**741**

Elaboración propia

De acuerdo con el Tabla 1, se observa que para los años de accidente más recientes (2019, 2020 y 2021) existió una disminución importante de los siniestros cerrados respecto a los años de ocurrencia anteriores; siendo el cambio más pronunciado para el año de ocurrencia 2021. Para este año, el costo promedio disminuyó a 4.340 €, importe por debajo del costo promedio total que es 7.800 €. Esto nos indica que ha habido un cambio en la velocidad de liquidación de los siniestros, ya sea debido a cambios normativos, disminución en el reporte de siniestros debido a factores externos, o a cambios internos en la política de gestión de siniestros, por ejemplo, puede haberse implementado nuevos criterios para la aceptación de siniestros, o puede haber existido un cambio en la tramitación de los siniestros que impactan en la velocidad del pago de estos.



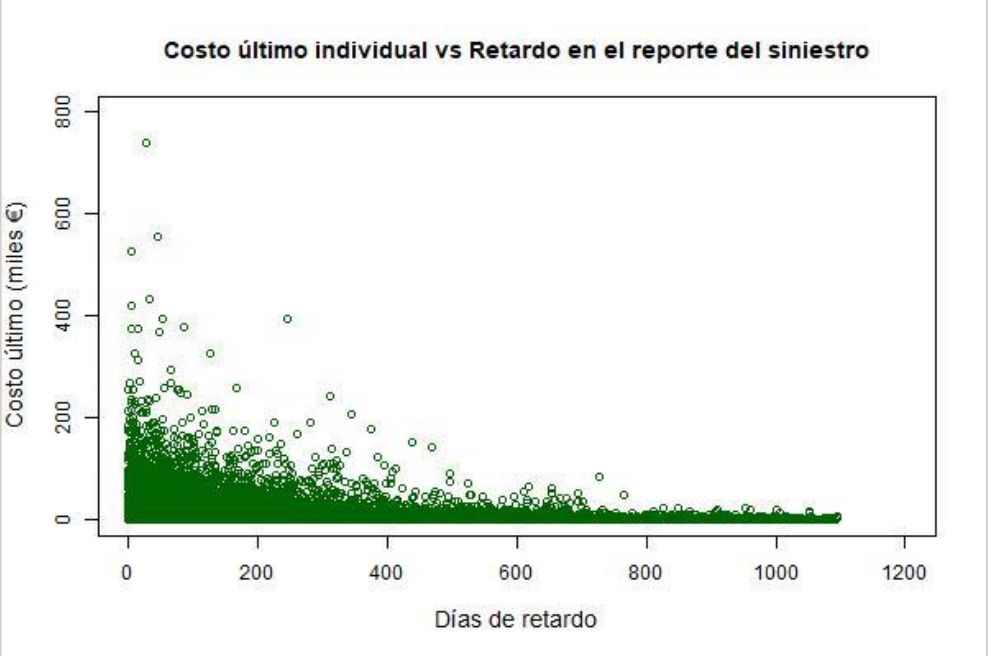
Elaboración propia

Gráfico 8. Densidad empírica del costo último individual de los siniestros cerrados

Por otro lado, en el Gráfico 8 se muestra la densidad empírica de los siniestros cerrados a nivel individual. Tal como se espera, existe una mayor frecuencia de siniestros cuyas severidades no superan los 5.000 €. No obstante, también existen siniestros con severidades elevadas, llegando hasta los 741.000 €. Asimismo, se observa que estos siniestros tienen una cola pesada puesto que existe una probabilidad significativa de ocurrencia de siniestros con severidades altas.

- Predictor: RepDelDays (retardo en el reporte del siniestro cerrado)

Según el Gráfico 9, no se observa una relación lineal entre el retraso en el reporte de los siniestros y el costo final de los mismos. Se puede ver de manera general que existe una mayor concentración de siniestros reportados en menos de 200 días (casi 7 meses) desde su fecha de ocurrencia cuyos costos finales están entre 0 € y 200.000 €. Los días máximo de retraso en reporte es 1.095 días.

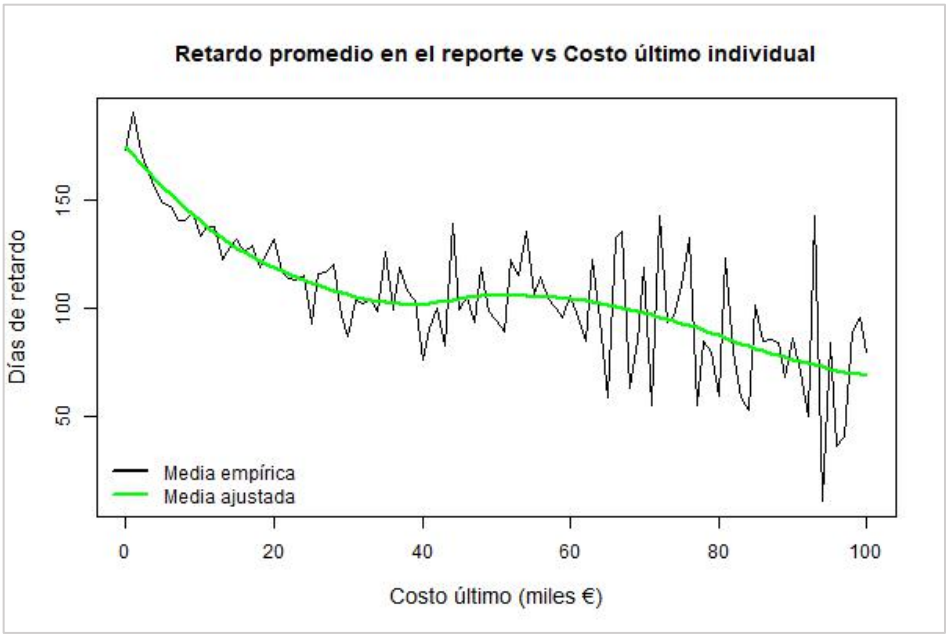


Elaboración propia

Gráfico 9. Dispersión del costo último individual versus Días de retardo en el reporte

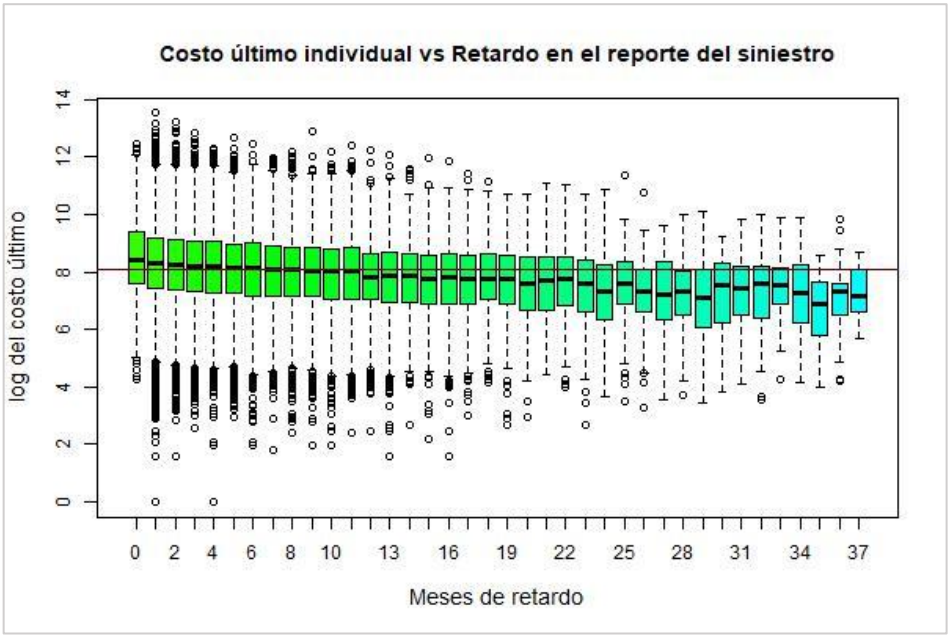
En consecuencia, a fin de tener un mejor entendimiento de la relación entre costo último individual y el retraso en reporte, se realizaron 2 gráficos adicionales. Según el Gráfico 10, se observa claramente que mayor retardo (menor retardo) en el reporte del siniestro, es menor (mayor) su costo último. Esto se puede interpretar de la siguiente manera, ante la ocurrencia de un siniestro con severidad alta es más probable que el asegurado comunique con prontitud el accidente a fin de cubrir los costos derivados del mismo; mientras que, si el siniestro ocurrido no presenta una severidad significativa, el asegurado no necesariamente tiene un incentivo para reportarlo inmediatamente.

Así también, del Gráfico 11 se observa que, para los primeros meses de retardo, el costo último presenta una mayor variabilidad, es decir, existen severidades más extremas.



Elaboración propia

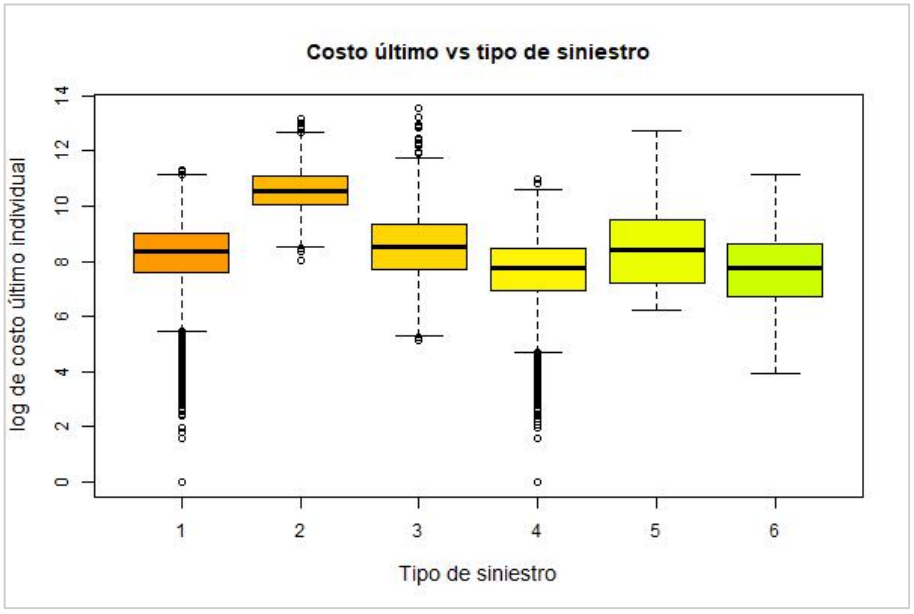
Gráfico 10. Retardo promedio en el reporte versus Costo último individual



Elaboración propia

Gráfico 11. Diagrama de cajas del log. costo último individual Vs Meses de retardo en el reporte

- Predictor: Type (tipo de siniestro cerrado)



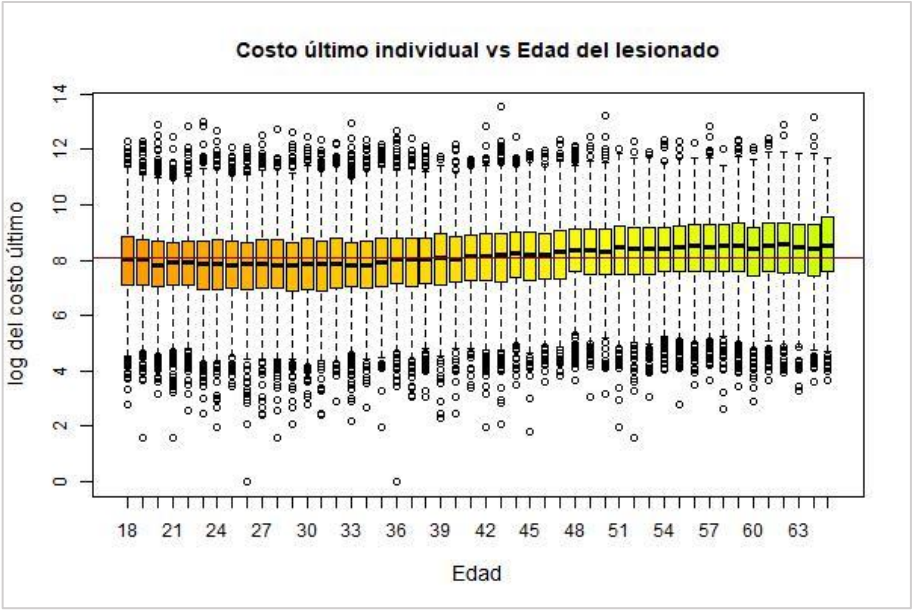
Elaboración propia

Gráfico 12. Diagrama de cajas del log. costo último individual Vs Tipo de siniestro

Con respecto al predictor Type (tipo de siniestro), del Gráfico 12 se observa que los siniestros de tipo 2 tienen la mediana más elevada del costo último y un rango intercuartílico más angosto; es decir, la severidad de dichos siniestros es más grande si lo comparamos con el resto de los siniestros y más homogéneos. Asimismo, los siniestros de tipo 4 y 6 son los que presentan severidades más bajas; mientras que los siniestros de tipo 5 presenta un mayor rango intercuartílico; es decir, los costos últimos de los siniestros cerrados son más heterogéneos.

- Predictor: Age (edad del lesionado)

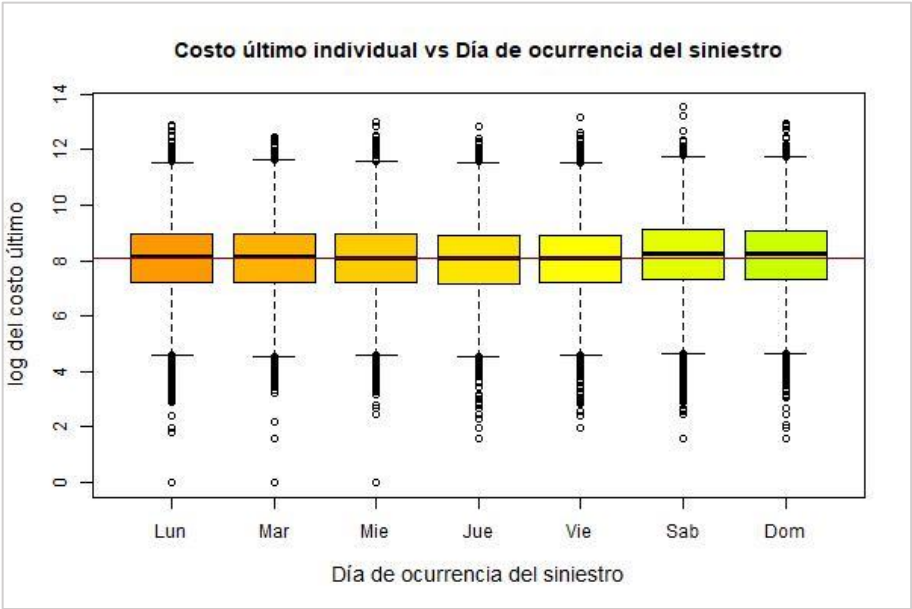
Con relación al predictor Age, observamos que a medida que la edad se incrementa desde los 18 años hasta los 40 años, el costo final del siniestro disminuye ligeramente. Por otro lado, para edades superiores a 40 años el costo final del siniestro se va incrementando. Este comportamiento se puede explicar por las características del seguro en evaluación, responsabilidad civil por lesiones corporales. Mientras mayor sea la persona que ha sufrido la lesión es más probable que estas lesiones sean más graves y que los costos médicos para su recuperación sen más elevados.



Elaboración propia

Gráfico 13. Diagrama de cajas del log. costo último individual Vs Edad de lesionado

- Predictor: AccWeekday (día de ocurrencia del siniestro)



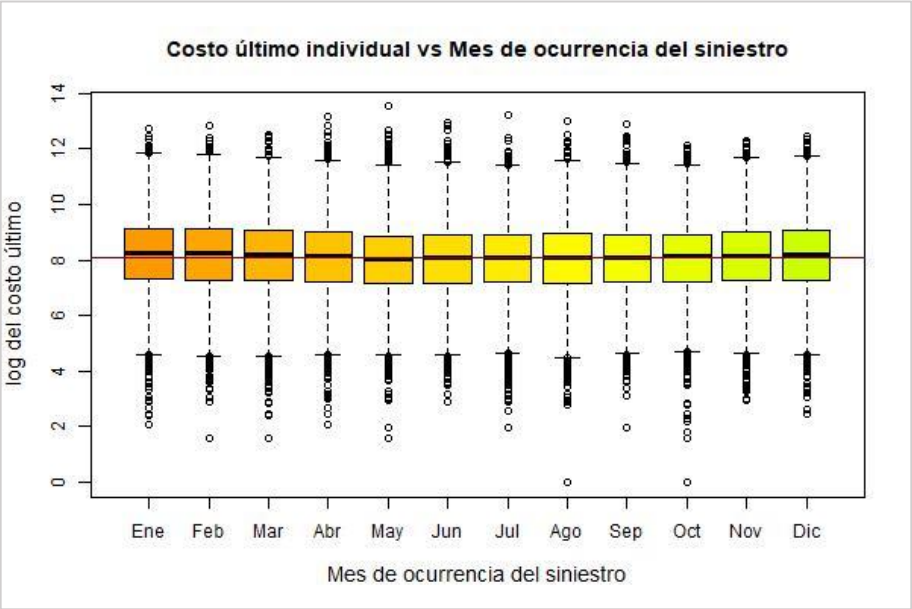
Elaboración propia

Gráfico 14. Diagrama de cajas del log. costo último individual Vs Día de ocurrencia

Del Gráfico 14, se puede observar que existe una relación positiva entre los siniestros ocurridos durante el fin de semana (sábado y domingo) y el costo final de estos; es decir, los accidentes ocurridos en el fin de semana presentan ligeramente una mayor severidad que aquellos ocurridos el resto de los días. Este comportamiento se puede explicar debido a que es más probable que durante los

fines de semana las personas hagan un mayor uso de sus vehículos y además quizás algunas personas conduzcan en estado de ebriedad lo que puede generar una mayor cantidad de accidentes con una severidad más alta en esos días.

- Predictor: AccYearMonth (mes de ocurrencia del siniestro)



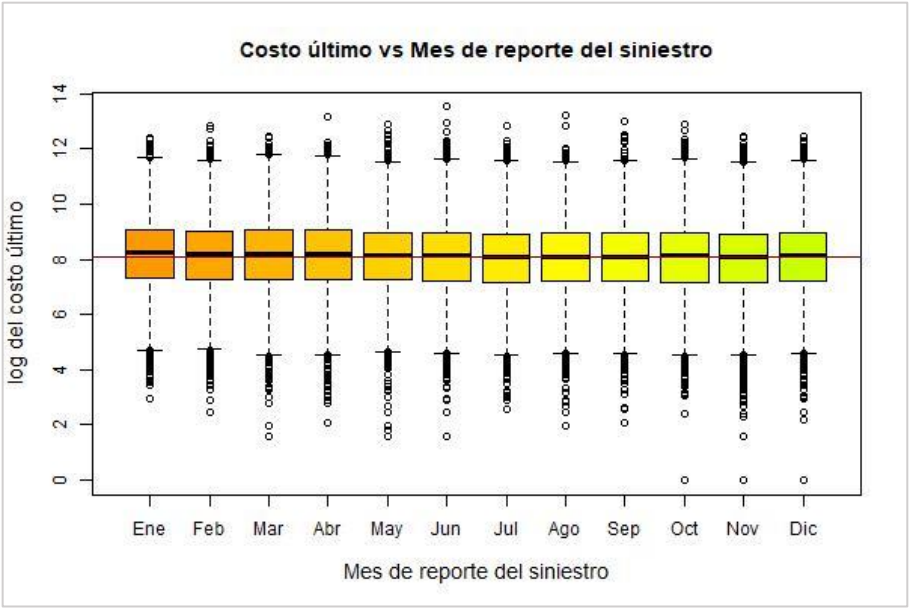
Elaboración propia

Gráfico 15. Diagrama de cajas del log. costo último individual Vs Mes de ocurrencia

Con respecto a la relación entre el mes de ocurrencia del siniestro y su costo final, según el Gráfico 15, si bien el costo final no presenta una heterogeneidad marcada entre meses de ocurrencia, si se observa que durante los meses de enero, febrero, noviembre y diciembre los costos últimos (en media) se elevan ligeramente respecto al resto de los meses.

El comportamiento descrito coincide con los meses de invierno en Europa, lo cual hace sentido, debido que los datos de siniestros simulados replican el comportamiento de una cartera anónima pero real de siniestros de responsabilidad civil de vehículos. Existe una mayor probabilidad que ocurran siniestros más severos en los meses de invierno debido al clima característico de la zona, la nieve, que dificulta la conducción de los vehículos y están más propensos a sufrir accidentes que pueden involucrar la colisión entre más de 2 vehículos.

- Predictor: RepYearMonth (mes de reporte del siniestro)



Elaboración propia

Gráfico 16. Diagrama de cajas del log. costo último individual Vs Mes de reporte

Finalmente, en el Gráfico 16 se presenta la relación entre el mes de reporte del siniestro y su costo final. Se observa un mayor costo medio para los siniestros ocurridos en los meses del primer cuatrimestre de cada año.

7.2. Entrenamiento de los algoritmos ML

En este apartado se describe el proceso de entrenamiento de los siniestros con los algoritmos de *machine learning* elegidos para predecir el costo final de cada siniestro reportado, al que nos referimos como *ultimate* (individual).

Para obtener el conjunto de datos de entrenamiento y de prueba se decidió aplicar el método de validación cruzada k-fold, explicado en un capítulo anterior de este documento, con el objetivo de realizar estimaciones más precisas; en lugar de dividir los datos en una única muestra de entrenamiento y otra de prueba como usualmente se hace, en la proporción de 80% y 20%.

Para aplicar esta metodología, validación cruzada k-fold, se han dividido aleatoriamente los siniestros (cerrados) individuales en $k = 10$ grupos (o folds). El primer grupo es utilizado como el conjunto de datos de prueba, mientras que los 9 grupos restantes son empleados en conjunto como datos de entrenamiento. Cada algoritmo de *machine learning* se entrena con este último conjunto de datos.

Una vez entrenado el algoritmo, se aplica al primer conjunto que llamamos datos de prueba para calcular la bondad de ajuste del modelo o error de predicción

conocido como el error cuadrático medio normalizado (*normalized root mean square error*), $NRMSE_1$, correspondiente al primer conjunto de datos de prueba, de acuerdo con las siguientes ecuaciones:

$$RMSE_1 = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n_1}}$$

$$NRMSE_1 = \frac{RMSE_1}{y_{max,1} - y_{min,1}}$$

donde:

- $RMSE_{k=1}$ es el error cuadrático medio del primer conjunto de datos de prueba.
- y_i es el costo último observado del $i-$esimo siniestro (cerrado) del primer conjunto de datos de prueba.
- $\hat{y}_i$ es el costo último estimado del $i-$esimo siniestro del primer conjunto de datos de prueba.
- n_1 es el número de siniestros del primer conjunto de datos de prueba.
- $y_{max,1}$ e $y_{min,1}$ son el costo último máximo y mínimo, respectivamente, de los siniestros del primer conjunto de datos de prueba.
- $NRMSE_1$ es el error cuadrático medio normalizado del primer conjunto de datos de prueba.

A continuación, se repite el mismo proceso descrito en los párrafos anteriores; es decir, se selecciona el segundo grupo como el conjunto de datos de prueba, y los 8 grupos restantes constituyen el conjunto de datos de entrenamiento. Este último conjunto de datos es empleado para el entrenamiento de los algoritmos de *machine learning*. Una vez entrenados los algoritmos, se vuelve a calcular el error cuadrático medio normalizado correspondiente al segundo grupo, $NRMSE_2$, de acuerdo con las 2 ecuaciones anteriores.

Se repite el mismo proceso 8 veces más, obteniendo en total 10 errores cuadráticos medios normalizados, $NRMSE_1$, $NRMSE_2$, $NRMSE_3$, …, $NRMSE_{10}$, correspondientes a los $k = 10$ grupos en los cuales la muestra inicial de datos fue dividida.

Finalmente, se obtiene un error de predicción total para cada algoritmo de *machine learning*, calculado como el promedio simple de los 10 errores cuadráticos medios normalizados.

$$NRMSE = \sqrt{\frac{\sum_{j=1}^{10} {RMSE_j}^2}{10}}$$

donde:

- $NRMSE$ es el error cuadrático medio normalizado del algoritmo de *machine learning.*

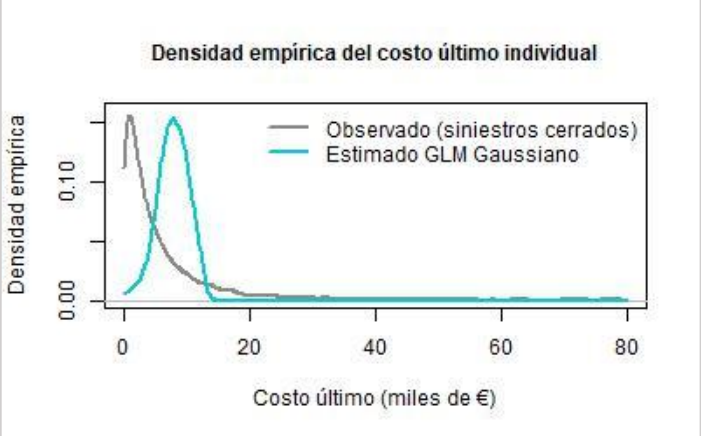
7.3. Selección de los algoritmos ML

Con base a los explicado en la sección anterior, se procedió a calcular el $NRMSE$ de 5 algoritmos de *machine learning*, así como a graficar las densidades empíricas estimadas comparadas con las densidades observadas, a fin de corroborar visualmente cuales algoritmos se ajustan mejor a los datos de la variable objetivo, el costo final del siniestro.

Tabla 2. Error cuadrático medio normalizado según algoritmo ML

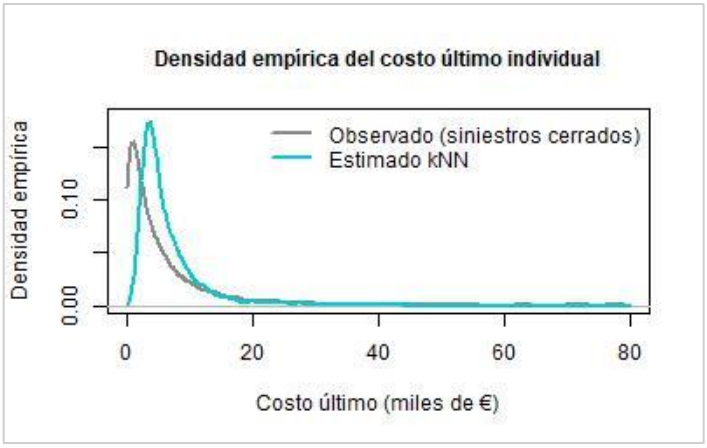
	GLM	k-NN	RandomForest	GBM	XGB
NRMSE	0.0231	0.0208	0.0200	0.0233	0.0202

Elaboración propia



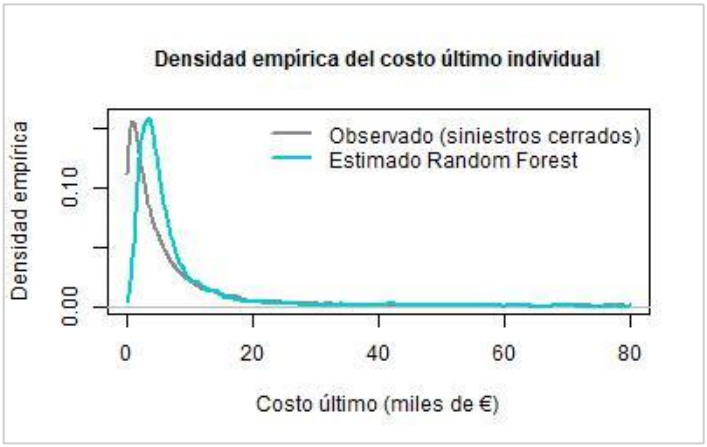
Elaboración propia

Gráfico 17. Densidad empírica del costo último aplicando GLM



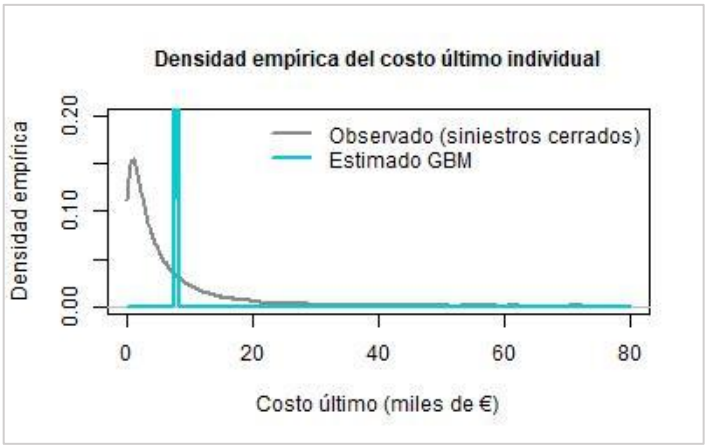
Elaboración propia

Gráfico 18. Densidad empírica del costo último aplicando k-NN



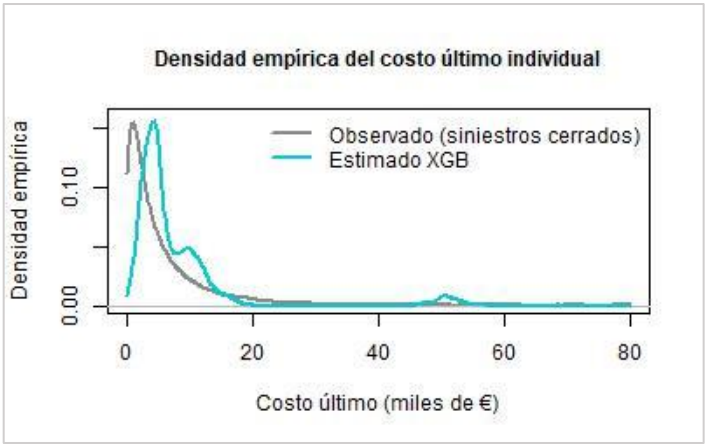
Elaboración propia

Gráfico 19. Densidad empírica del costo último aplicando Random Forest



Elaboración propia

Gráfico 20. Densidad empírica del costo último aplicando GBM



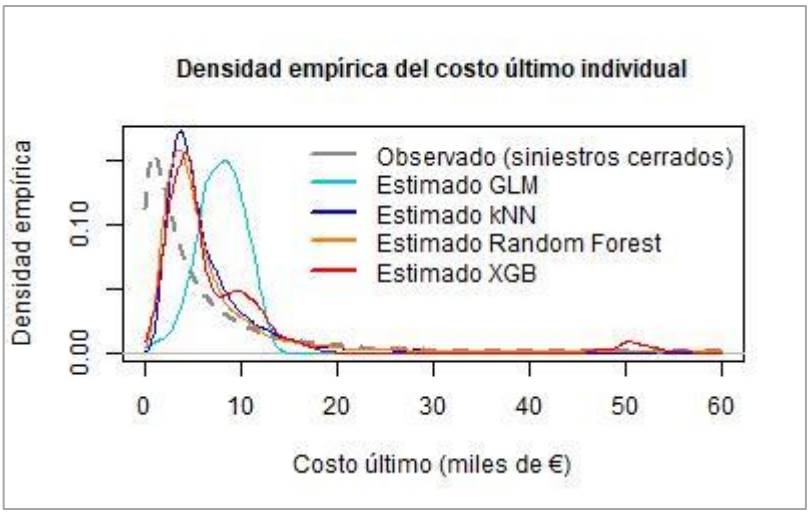
Elaboración propia

Gráfico 21. Densidad empírica del costo último aplicando XGB

Según se puede apreciar en la Tabla 2, algoritmo Random Forest (Bosque Aleatorio) presenta el menor error cuadrático medio (0,2000), seguido del algoritmo Extreme Gradient Boosting - XGB (0,2002) y de k-Nearest Neighbors – k-NN (k Vecinos más cercanos) (0,0208). Por otro lado, los algoritmos que presentan peores performances son Gradient Boosting Machine - GBM (0,0233) y Generalized Linear Model – GLM (0,0231).

Si analizamos las distribuciones de densidad estimadas mostradas en los gráficos del 17 al 21 o el Gráfico 22, observamos que en realidad los siniestros con costos últimos menores de 15,000 € no fueron estimados de manera precisa, puesto que se observan desviaciones importantes a lo largo del histograma, principalmente los algoritmos GBM y GLM. Por otro lado, sí se observa un buen ajuste para los siniestros más graves (en la cola de la distribución).

Este resultado quiere decir que las variables características utilizadas como predictores (días de retraso en el reporte, tipo de siniestro, edad del lesionado, día de ocurrencia del siniestro, mes de ocurrencia y mes de reporte del siniestro) tienen poco poder predictivo sobre la variable respuesta (el costo último del siniestro).



Elaboración propia

Gráfico 22. Densidad empírica del costo último para todos los algoritmos ML

Por lo tanto, como una primera conclusión, diremos que las variables características disponibles y utilizadas como predictores (días de retraso en el reporte, tipo de siniestro, edad del lesionado, día de ocurrencia del siniestro, mes de ocurrencia y mes de reporte del siniestro) en este estudio tienen poco poder predictivo sobre la variable respuesta (el costo último del siniestro).

Por consiguiente, para mejorar los resultados de las técnicas de *machine learning*, es necesario contar con otras características o variables de la cartera de siniestros, como: el costo inicial del siniestro reportado, el tipo o modelo de vehículo asegurado, el grado alcohólico del conductor del vehículo, el género del asegurado, la ciudad o zona de ocurrencia del siniestro, si el asegurado es nuevo o no en la compañía, entre otras características. Estas variables típicamente si están disponibles en una aseguradora, por lo que en un ejercicio con datos reales si es posible aplicar y obtener resultados más precisos mediante estos algoritmos.

No obstante, el objetivo de este documento es comparar las predicciones para la reserva total de siniestros con los campos que típicamente conforman las base de datos de siniestros que se emplean en los modelos clásicos de estimación de reservas de siniestros. Una variable típica que está presente en estas bases de datos es la reserva inicial reportada del siniestro, la cual podría ser un predictor importante del costo final del siniestro. Desafortunadamente, la base de datos simulada, con la que se está trabajando, no contiene esta información.

7.4. Estimación de la reserva de siniestros pendientes (RSP)

Una vez entrenados los algoritmos, se procede a aplicarlos a datos nuevos para predecir la variable respuesta, cuyo valor real no se conoce.

En nuestro caso, aplicaremos los algoritmos a los siniestros pendientes de liquidación o también llamados siniestros abiertos; es decir, aquellos que han sido reportados a la compañía pero que su valor final o costo último permanece desconocido. Una vez que estimamos de manera individual el costo final de cada siniestro reportado, se le resta los pagos parciales ya efectuados, obteniendo como resultado la reserva del siniestro pendiente de liquidación. Si sumamos cada una de estas reservas, se conocerá la reserva de siniestros pendientes de la cartera en evaluación, el seguro de responsabilidad civil vehicular por daños corporales.

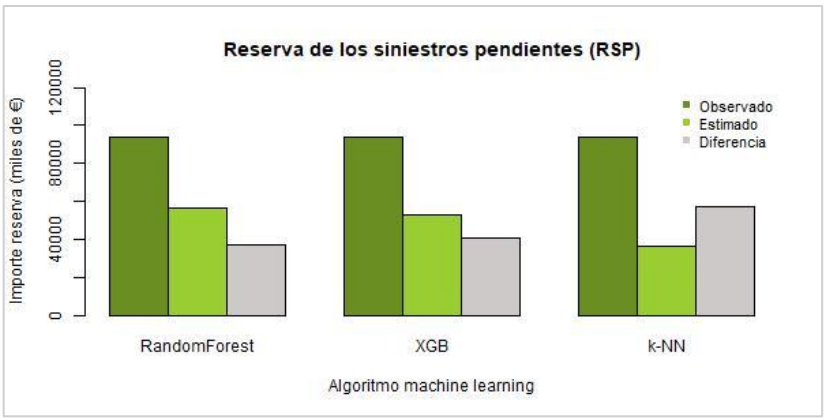
Además, puesto que la base de datos con la que estamos trabajando, al ser simulada si contiene los costos finales o *ultimates* de cada siniestro, esté este siniestro cerrado o abierto, es posible contrastar las predicciones efectuadas con los algoritmos de *machine learning* con los importes reales u observados de la reserva de siniestros pendientes.

A continuación, se muestra los resultados, por año de ocurrencia, obtenidos siguiendo el proceso descrito en el primer párrafo de esta sección. Cabe señalar, que se descartaron los algoritmos GBM y GLM por presentar las peores capacidades predictivas (los NRMSE más bajos) como se mostró anteriormente.

Tabla 3. RSP observada y estimada por año de ocurrencia (en miles de €)

Año de origen	Observado	Algoritmo Random Forest			Algoritmo XGB			Algoritmo k-NN		
		Estimada	Dif. en €	**Dif. %**	Estimada	Dif. en €	**Dif. %**	Estimada	Dif. en €	**Dif. %**
2012	0	0	0	**0%**	0	0	**0%**	0	0	**0%**
2013	113	20	-93	**-82%**	44	-69	**-61%**	5	-108	**-95%**
2014	114	-63	-177	**-155%**	-57	-171	**-150%**	-87	-201	**-176%**
2015	87	66	-21	**-24%**	29	-58	**-67%**	-21	-108	**-124%**
2016	749	36	-713	**-95%**	31	-718	**-96%**	-251	-1,000	**-133%**
2017	2,144	480	-1,664	**-78%**	506	-1,638	**-76%**	-145	-2,289	**-107%**
2018	8,287	1,965	-6,322	**-76%**	2,028	-6,259	**-76%**	-737	-9,024	**-109%**
2019	15,132	6,155	-8,978	**-59%**	5,285	-9,848	**-65%**	2,040	-13,093	**-87%**
2020	31,225	20,012	-11,213	**-36%**	18,812	-12,413	**-40%**	13,582	-17,643	**-57%**
2021	35,898	28,051	-7,848	**-22%**	26,543	-9,356	**-26%**	22,070	-13,829	**-39%**
Total	93,752	56,722	-37,030	**-39%**	53,221	-40,531	**-43%**	36,456	-57,296	**-61%**

Elaboración propia



Elaboración propia

Gráfico 23. RSP observada Vs RSP estimada según algoritmo machine learning

Según los resultados presentados en la Tabla 3 y Gráfico 23, todos los algoritmos al tener poco poder predictivo del costo final del siniestro también subestiman de manera significativa la reserva pendiente tanto por año de ocurrencia como también a nivel total. La diferencia mínima de la reserva de siniestros estimada respecto a su valor observado corresponde al algoritmo Random Forest (-39%), mientras que la mayor diferencia se obtiene con el algoritmo k-NN (-61%).

Para una mejor visualización, se muestran los gráficos de barras por año de ocurrencia.

Elaboración propia

Gráfico 24. RSP observada y estimada mediante algoritmo Random Forest

Elaboración propia

Gráfico 25. RSP observada y estimada mediante algoritmo XGB

Elaboración propia

Gráfico 26. RSP observada y estimada mediante algoritmo k-NN

7.5. Estimación de la reserva IBNR

Para la estimación de la reserva IBNR se replicó un procedimiento descrito brevemente en el documento "*Estimation of Individual Claim Liabilities*" de M. De Virgilis y P. Cerqueti.

Este procedimiento consiste en agrupar los costos finales de los siniestros cerrados por año de ocurrencia y según el retardo en el año de reporte:

- El primer grupo se llamará siniestros sin retardo y son los siniestros reportados en el mismo año en que ocurrieron.
- El segundo grupo se llamará siniestros con retardo y son los siniestros reportados uno o más años después de sus respectivas fechas de ocurrencia.

Luego hallamos una ratio equivalente al cociente entre el costo total de los siniestros con retardo y los siniestros sin retardo, para cada año de ocurrencia. Este ratio es un estimador de la proporción de siniestros que son reportados en una fecha posterior a su ocurrencia respecto a aquellos reportados en la misma fecha de ocurrencia. Para el último año de ocurrencia, 2021, no se puede calcular el ratio puesto que no se han desarrollado los siniestros, por lo que se debe estimar, con base a la serie de los valores de los años anteriores.

Una vez que ya se ha obtenido el ratio para el año de ocurrencia 2021, este se multiplica por el costo total de los siniestros ocurridos y reportados en el año 2021 (esto incluye a los siniestros cerrados y pendientes). El resultado del producto efectuado será una estimación de la reserva IBNR del año 2021 (que es la fecha de corte y análisis de los datos), puesto que representa el importe estimado de los siniestros ocurridos en el 2021 pero que serán reportados después de ese año (siniestros con retardo). Como a la fecha de corte, se conoce el costo final de los siniestros cerrados, estos importes son ciertos y no deben estimarse; no obstante, no se conoce el costo final de los siniestros pendientes o abiertos por lo que en su lugar se emplean los importes estimados mediante las técnicas de *machine learning*.

A continuación, se muestran los ratios observados, obtenidos según el procedimiento descrito en los párrafo anteriores:

Tabla 4. Ratio siniestros con retardo / siniestros sin retardo

Año de origen	Costo final de los siniestros cerrados (miles €)		Ratio
	Sin retardo	Con retardo	
2012	34,604	14,454	42%
2013	32,876	15,016	46%
2014	34,815	15,197	44%
2015	35,229	14,133	40%
2016	35,195	16,494	47%
2017	36,356	14,901	41%
2018	32,806	12,909	39%
2019	26,724	8,564	32%
2020	15,081	2,948	20%

Elaboración propia

Según lo mostrado en la Tabla 4, los ratios permanecen más o menos estables entre los años 2021 y 2018; sin embargo, para los años 2019 y 20220 se observa una disminución importante; si bien estos nivel pueden ser los correctos y mantenerse en el tiempo, es más probable que la disminución de los ratios obedece a que para los años más recientes los siniestros no están totalmente desarrollados, por lo que los ratios de los años 2019 y 2010 están subestimados.

En este caso, como se trata de siniestros cerrados, al decir que no están totalmente desarrollados, nos referimos a que aún existe una cantidad importante de siniestros ocurridos en los años 2019 y 2020 que serán reportados a partir del año 2022; es decir aún no se conocen, por lo que también se debe calcular la reserva IBNR para estos años de ocurrencia. Esto es coherente con el retardo en el reporte de los siniestros que se analizó anteriormente, donde se obtuvo que el retardo máximo de los siniestros cerrados es de 1054 días o 36 meses; por lo que se debe considerar que para los 3 años de ocurrencia más recientes (2019, 2020 y 2021) existen siniestros no reportados aún.

Por lo explicado en el párrafo anterior, se procedió a estimar los ratios de los años 2019, 2020 y 2021 en función de los ratios observados desde el año 2012 hasta el año 2018, ajustando dichos ratios a una regresión lineal, como se muestra en el Gráfico 27. También se probó otras regresiones, por ejemplo, una regresión con polinomios de grado 2 y 3; sin embargo, fue la regresión lineal (polinomio de grado1) que mejor se ajustaba a los datos observados.

Elaboración propia

Gráfico 27. Ajuste del ratio siniestros con retardo / siniestros sin retardo

Con los ratios ya estimados, se procedió a la obtención de la predicción de la reserva IBNR. Para el año 2021, la reserva estimada IBNR será equivalente al producto entre el ratio estimado correspondiente y el costo de los siniestros ocurridos en el año 2021 y reportados en esa misma fecha (siniestros sin retardo). Este costo está compuesto tanto por los siniestros cerrados como por los siniestros pendientes o abiertos cuyo costo último ha sido estimado con los algoritmos de *machine learning*.

Para el año 2019, la reserva IBNR a la fecha de corte 2021 es el producto entre el ratio estimado y el costo de los siniestros sin retardo, igual que para el año 2021, pero adicionalmente se le debe restar a dicho producto el costo de los siniestros que

ocurrieron en el año 2019 y que ya fueron reportados en los años 2020 y 2021 (esto incluye los siniestros ya cerrados como los siniestros abiertos. El costo final estimado de estos siniestros pendientes ya fue incluido en la reserva de siniestros pendientes)

De manera similar se calcula el IBNR para el año 2020 a la fecha de corte 2021, como el producto del ratio estimado y costo de los siniestros sin retardo, menos el costo de los siniestros que ocurrieron en el año 2020 y que ya fueron reportados durante el 2021.

En la Tabla 5 se presenta el detalle de los resultados de la reserva IBNR estimada.

Tabla 5. Reserva IBNR estimada según tipo de algoritmo machine learning

Año de origen	Ratio estimado (a)	Costo de los siniestros reportados sin retardo (b)			Menos siniestros ya reportados (c)			Costo estimado de los siniestros reportados con retardo (a x b - c = IBNR)		
		Random Forest	XGB	k-NN	Random Forest	XGB	k-NN	Random Forest	XGB	k-NN
2019	41%	34,023	33,454	31,725	14,492	14,190	12,675	-640	-571	241
2020	40%	31,449	30,542	27,391	13,714	13,421	11,342	-1,062	-1,133	-323
2021	40%	34,986	33,479	29,005	0	0	0	13,908	13,309	11,530
Total								12,206	11,605	11,448

Elaboración propia

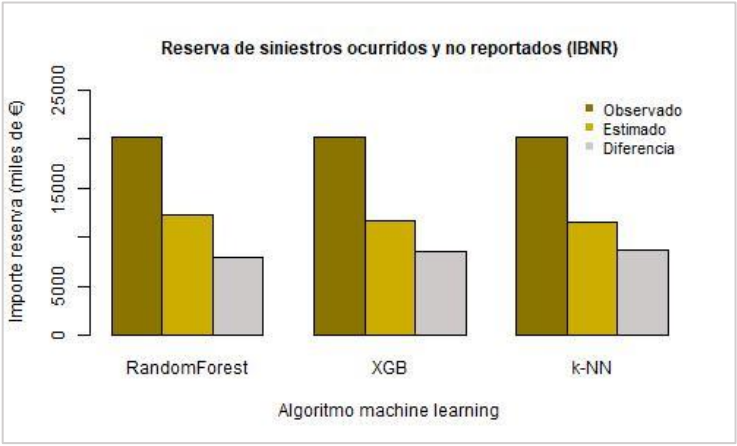
Finalmente, las estimaciones efectuadas se comparan con la reserva IBNR observada[7], a fin de determinar la precisión de las estimaciones mediante los algoritmos de *machine learning*. En la Tabla 6 y Gráfico 28, se presentan dichas comparaciones. Debido a que la estimación de la reserva IBNR depende del costo final estimado de los siniestros pendientes, costo que ha sido subestimado con las técnicas de *machine learning* como se mostró en la sección anterior, también se obtiene una reserva IBNR significativamente subestimada. La reserva IBNR con base a las predicciones del algoritmo Random Forest presenta la menor diferencia (39%); sin embargo, este grado de diferencia no es aceptable para ningún método de predicción.

[7] Como se señaló en la sección de descripción de la base de datos, esta contiene además de los costos finales de los siniestros cerrados y pendientes, también los costos finales o ultimate de los siniestros ocurridos y no reportados.

Tabla 6. Reserva IBNR observada y estimada según algoritmo machine learning

Algoritmo	Reserva IBNR (en miles €)	Diferencia en €	Diferencia %
Obserrvada	20,104	0	0%
Random Forest	12,206	-7,898	-39%
XGB	11,605	-8,499	-42%
k-NN	11,448	-8,656	-43%

Elaboración propia



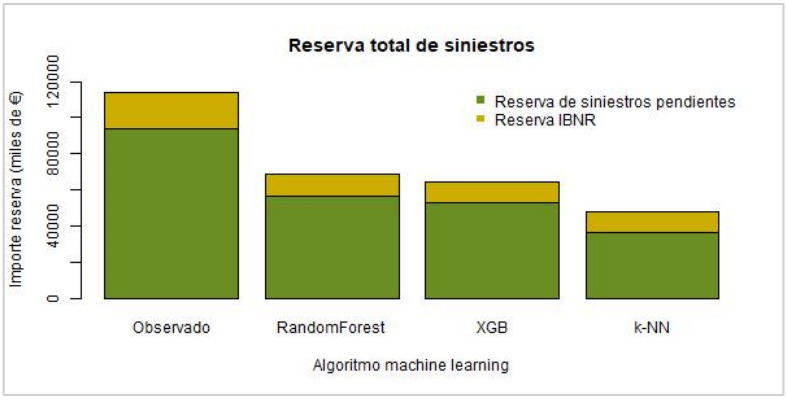
Elaboración propia

Gráfico 28. IBNR observado Vs IBNR estimado según algoritmos ML

7.6. Reserva total de siniestros

La reserva total de siniestros está compuesta por la suma de la reserva de siniestros pendientes (RSP) y la reserva IBNR. De acuerdo con los resultados de las 2 secciones anteriores, los algoritmos de *machine learning* subestiman de manera significativa la reserva total de siniestros observada, encontrando una diferencia de hasta -58% con el método k-NN; mientras que la menor diferencia se reporta con el método Random Forest, siendo esta -39%.

Como también se ha señalado, la falta de poder predictivo de los algoritmos es consecuencia de la escasez de predictores que ofrece la base de datos simulada (tipo de siniestro, retraso en el reporte del siniestro, edad del lesionado, día de ocurrencia, mes de ocurrencia y mes de reporte del siniestro).



Elaboración propia

Gráfico 29. Reserva total observada y estimada según tipo de reserva

Tabla 7. Reserva total observada y estimada según algoritmo machine learning

Año de origen	Observado	Algoritmo Random Forest			Algoritmo XGB			Algoritmo k-NN		
		Estimada	Dif. en €	**Dif. %**	Estimada	Dif. en €	**Dif. %**	Estimada	Dif. en €	**Dif. %**
2012	0	0	0	**0%**	0	0	**0%**	0	0	**0%**
2013	113	20	-93	**-82%**	44	-69	**-61%**	5	-108	**-95%**
2014	114	-63	-177	**-155%**	-57	-171	**-150%**	-87	-201	**-176%**
2015	87	66	-21	**-24%**	29	-58	**-67%**	-21	-108	**-124%**
2016	749	36	-713	**-95%**	31	-718	**-96%**	-251	-1,000	**-133%**
2017	2,144	480	-1,664	**-78%**	506	-1,638	**-76%**	-145	-2,289	**-107%**
2018	8,287	1,965	-6,322	**-76%**	2,028	-6,259	**-76%**	-737	-9,024	**-109%**
2019	15,245	5,515	-9,731	**-64%**	4,714	-10,531	**-69%**	2,280	-12,965	**-85%**
2020	32,694	18,951	-13,744	**-42%**	17,679	-15,015	**-46%**	13,260	-19,435	**-59%**
2021	54,420	41,959	-12,462	**-23%**	39,851	-14,569	**-27%**	33,600	-20,821	**-38%**
Total	113,856	68,929	-44,927	**-39%**	64,827	-49,030	**-43%**	47,905	-65,951	**-58%**

Elaboración propia

8. APLICACIÓN DE MODELOS ESTOCÁSTICOS

En esta sección se estima la reserva total de siniestros de acuerdo con los métodos estocásticos Mack de distribución libre (en adelante, Mack) y Bootstrap chain-ladder (en adelante, Boot) explicados en la primera sección de este documento.

El triángulo formado con los datos disponibles corresponde a un triángulo de pagos parciales acumulados tanto de los siniestros cerrados como de los siniestros pendientes o abiertos, sobre el cual posteriormente se aplican los modelos estocásticos.

Como se observa en las Tabla 8 y 9, el último año de desarrollo presenta un incremento de los pagos respecto a los desarrollos anteriores, para los siniestros ocurridos en los últimos 4 años. Esto se puede deber a un cambio en la velocidad de liquidación de los siniestros, sea este derivado por cambios regulatorios, cambios en la política interna de gestión de siniestros (en la aceptación o tramitación de estos) o que la cartera del seguro de responsabilidad civil vehicular por lesiones corporales se ha incrementado y naturalmente con ello el pago de los siniestros reportados.

Por otro lado, se observa que no existe la necesidad de incorporar un factor cola puesto que el triángulo de pagos se encuentra completamente desarrollado dado que para los años de ocurrencia 2015 hacia atrás los factores de desarrollo son iguales a la unidad.

Tabla 8. Triángulo de pagos acumulados en miles de €

Año de origen	1	2	3	4	5	6	7	8	9	10
2012	4,232	19,705	33,406	40,988	45,806	47,845	48,520	48,820	48,961	49,047
2013	4,664	22,789	36,199	42,788	45,901	46,773	47,518	47,912	47,899	
2014	5,321	23,336	37,810	45,172	48,526	49,643	50,025	50,118		
2015	4,948	23,146	37,668	44,976	48,065	49,250	49,437			
2016	5,135	24,443	40,552	47,797	50,994	52,249				
2017	5,457	24,259	40,791	48,701	52,436					
2018	5,237	25,632	41,964	50,460						
2019	6,023	25,533	42,276							
2020	5,563	25,141								
2021	6,919									

Elaboración propia

Tabla 9. Factores de desarrollo individuales

Año de origen	2/1	3/2	4/3	5/4	6/5	7/6	8/7	9/8	10/9
2012	4.66	1.70	1.23	1.12	1.04	1.01	1.01	1.00	1.00
2013	4.89	1.59	1.18	1.07	1.02	1.02	1.01	1.00	
2014	4.39	1.62	1.19	1.07	1.02	1.01	1.00		
2015	4.68	1.63	1.19	1.07	1.02	1.00			
2016	4.76	1.66	1.18	1.07	1.02				
2017	4.45	1.68	1.19	1.08					
2018	4.89	1.64	1.20						
2019	4.24	1.66							
2020	4.52								

Elaboración propia

8.1. Estimación de la reserva total de siniestros

Los métodos estocásticos Mack y Boot estiman el costo último de los siniestros por año de ocurrencia, y dado que únicamente se cuenta con el triángulo de pagos, la diferencia entre el costo último estimado y la última diagonal del triángulo de pagos es la reserva total de siniestros que incluye la reserva de siniestros pendientes como la reserva IBNR. Desafortunadamente no se puede hacer una desagregación entre estos 2 tipos de reservas, debido a que no se cuenta información de las reservas contables de siniestros pendientes.

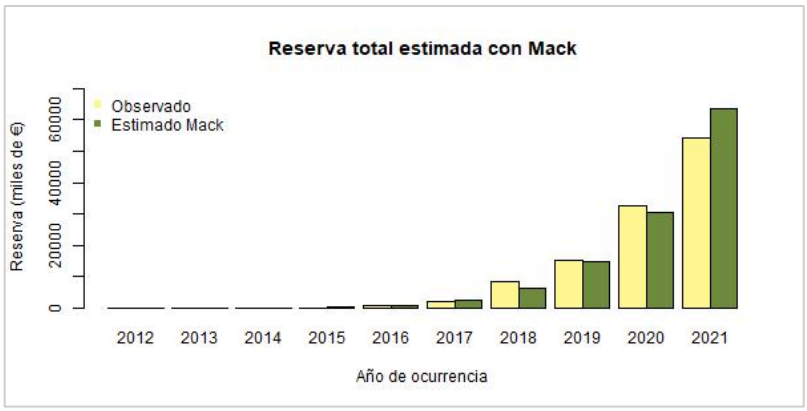
- Reserva total estimada mediante Mack

  La reserva total de siniestros estimada mediante este método a nivel agregado sobreestima en 4,9% la reserva total observada; aunque si analizamos las estimaciones por año de ocurrencia, vemos que existe una mayor variabilidad en los resultados.

Tabla 10. Reserva total (RSP + IBNR) observada y estimada mediante Mack

Año de origen	Observado (miles de €)	Estimado (miles de €)	Diferencia en €	Diferencia %
2012	11	0	-11	-100%
2013	113	84	-29	-26%
2014	114	155	41	36%
2015	87	420	333	383%
2016	812	985	173	21%
2017	2,240	2,433	193	9%
2018	8,305	6,502	-1,803	-22%
2019	15,329	14,781	-548	-4%
2020	32,705	30,678	-2,027	-6%
2021	54,437	63,655	9,218	17%
Total	**114,153**	**119,693**	**5,540**	**4.9%**

Elaboración propia



Elaboración propia

Gráfico 30. Reserva total (RSP + IBNR) observada y estimada mediante Mack

- Reserva total estimada mediante Bootstrap

  La reserva total de siniestros estimada mediante este método a nivel agregado sobreestima en 5,0% la reserva total observada; así también, de manera similar que Mack, las estimaciones por año de ocurrencia presentan una mayor variabilidad.

Tabla 11. Reserva total estimada (RSP + IBNR) mediante Bootstrap

Año de origen	Observado (miles de €)	Estimado (miles de €)	Diferencia en €	Diferencia %
2012	11	0	-11	-100%
2013	113	85	-28	-25%
2014	114	156	42	37%
2015	87	422	335	385%
2016	812	987	175	22%
2017	2,240	2,434	194	9%
2018	8,305	6,501	-1,804	-22%
2019	15,329	14,789	-540	-4%
2020	32,705	30,696	-2,009	-6%
2021	54,437	63,791	9,354	17%
Total	**114,153**	**119,861**	**5,708**	**5.0%**

Elaboración propia

Elaboración propia

Gráfico 31. Reserva total (RSP + IBNR) observada y estimada mediante Bootstrap

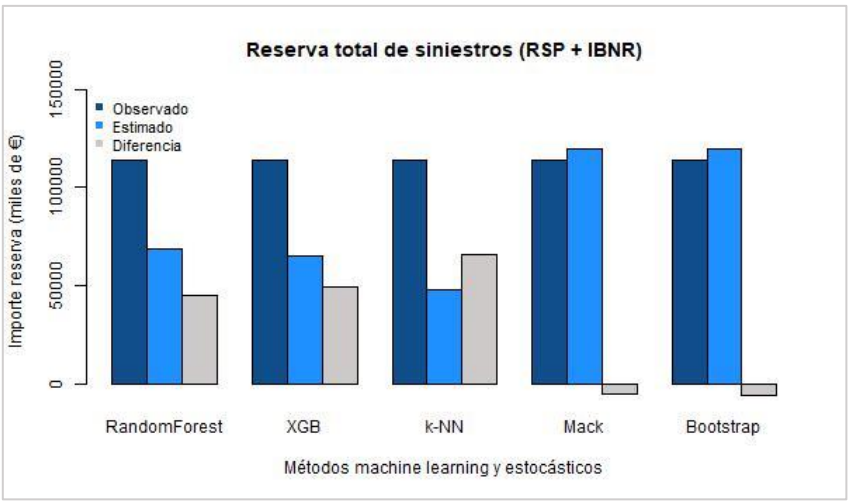
9. RESUMEN DE RESULTADOS

A continuación, en la Tabla 12 y Gráfico 32 se muestran los resultados de la reserva total de siniestros estimada mediante los algoritmos de *machine learning* seleccionados y los métodos estocásticos clásicos.

Tabla 12. Comparativo de la reserva total estimada (RSP + IBNR) (en miles de €)

Método	Estimado	Diferencia	Diferencia %
Observado	114,153	0	0%
Random Forest	68,929	-44,927	-65%
XGB	64,827	-49,030	-76%
k-NN	47,905	-65,951	-138%
Mack	119,693	5,540	5%
Bootstrap	119,861	5,708	5%

Elaboración propia



Elaboración propia

Gráfico 32. Comparativo de la reserva total estimada (RSP + IBNR)

10. CONCLUSIONES

La reserva total de siniestros (reserva de siniestros pendientes de liquidación y de pago, y la reserva IBNR) se estimó para una cartera de siniestros simulados de un seguro de responsabilidad civil por lesiones corporales de vehículos, consistente en 61.843 siniestros ocurridos durante un periodo de 10 años entre los años 2012 y 2021.

Con relación a los algoritmos de *machine learning*, las reservas totales estimadas mediante los 3 algoritmos de seleccionados (Random Forest, XGB y k-NN) no fueron estimaciones razonables puesto que difieren de las reservas totales observadas (reales) de manera significativa, llegando incluso a presentar una diferencia superior al -100% en el caso del algoritmo k-Nearest Neighbors (k-NN) y como mínimo una diferencia de -65% obtenida con el algoritmo Random Forest, como se puede apreciar en la Tabla 12 y Gráfico 32, donde se resumen los resultados. En consecuencia, las reservas fueron subestimadas, lo cual genera una importante insuficiencia de reservas, que es el peor escenario que puede presentarse.

El poco poder predictivo que presentaron estos algoritmos puede ser una consecuencia de la escasez de predictores que ofrece la base de datos simulada empleada (donde los predictores fueron: tipo de siniestro, retraso en el reporte del siniestro, edad del lesionado, día de ocurrencia, mes de ocurrencia y mes de reporte del siniestro).

Por otro lado, los métodos estocásticos clásicos, Mack y Bootstrap, sí lograron estimar de manera razonable las reservas totales agregadas, al presentar una diferencia (positiva) con respecto a las reservas totales observadas solo del 5% para ambos métodos. No obstante, si observamos las estimaciones de la reserva por año de ocurrencia, ambos métodos no fueron precisos.

Por lo tanto, en el caso particular de este estudio, los métodos clásicos estocásticos presentaron una mejor performance que las técnicas de *machine learning* en la estimación de la reserva total de siniestros de un seguro de responsabilidad civil por lesiones corporales de vehículos, lo que significa que el poder predictivo de estos algoritmos como es de suponer depende de manera importante de las estructura de los datos, estos deben contener una cantidad suficiente de variables explicativas o características para obtener resultados óptimos; mientras que los métodos estocásticos clásicos pueden ser muy útiles cuando se tiene información limitada como únicamente la fecha de ocurrencia, la fecha de reporte y los pagos parciales de los siniestros.

11. REFERENCIAS

- Blier-Wong C.; Cossette H.; Lamontagne L.; Marceau E. Machine learning in P&C insurance: A review for pricing and reserving. Risks 2021, 9, 4. Disponible en: https://dx.doi.org/10.3390/risks9010004

- Gabrielli A.; Wüthrich M. Individual claims history simulation machine. February 26, 2018. Risks 2018 https://www.mdpi.com/2227-9091/6/2/29. Disponible en: https://ssrn.com/abstract=3130560

- Wang M.; Wüthrich M. Individual claims generator for claims reserving studies: data simulation.R. June 3, 2022. Disponible en: https://ssrn.com/abstract=4127073

- Avanzi B.; Taylor G.; Wang M.; Wong B. SynthETIC: An individual insurance claim simulator with feature control. June 29, 2021. Insurance: Mathematics and Economics, 100, 296-308. ISSN 01676687, doi: 10.1016/j.insmatheco.2021.06.04. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/S0167668721001013412707 3

- Wüthrich M.; Buser C. Data analytics for non-life insurance pricing. October 27, 2021. Swiss Finance Institute Research Paper Nº 16-68. Disponible en: https://ssrn.com/abstract=2870308

- Wüthrich M. Non-Life Insurance: Mathematics & Statistics. February 7, 2022. Disponible en: https://ssrn.com/abstract=2319328

- De Virgilis M.; Cerqueti P. Estimation of individual claim liabilities: A comparison of traditional and machine learning methodologies. 2020.

- Loss data analytics: An open text authored by the actuarial community. July 7, 2022. Disponible en: https://openacttexts.github.io/Loss-Data-Analytics/

- Fernández R., Costa J., Oviedo M. Aprendizaje Estadístico. Octubre 13, 2021. Disponible en: https://rubenfcasal.github.io/aprendizaje_estadistico/index.html

- Finan M. An introductory guide in the construction of actuarial models: A preparation for the actuarial exam C/4. November 4, 2017.

- Álvarez J.; Coll V. Estimación de reservas en una compañía aseguradora. Una aplicación en Excel del método Chain-Ladder y Bootstrap. Diciembre, 2012.

ANEXO 1: PARÁMETROS DE LOS SINIESTROS SIMULADOS CON SynthETIC[8]

Parameter type	Functional form	Numerical parameters
Global		Time unit $= 1/4$ (calendar quarter) Reference claim size $= 200,000$[1]
Claim occurrence[2]		$I = 40$ $E_i = 12000$ $\lambda_i = 0.03$
Claim size[1]	Power-normal	$S_{ir}^{0.2} \sim \mathrm{N}(9.5, 3)$
Claim notification[1]	Weibull	Mean = $\min(3, \max(1, 2 - \frac{1}{3}\ln(s_{ir}/100000)))$ Coefficient of variation = 70%
Claim closure[1]	Weibull	Mean = $a(i)\min(25, \max(1, 6 + 4\ln(s_{ir}/20000)))$ Coefficient of variation = 60% where $a(i) = \max(0.85, 1 - 0.0075i)$, subject to the over-riding condition that, for $s_{ir} < 20000$ and $i \geq 21$, $a(i) = \min(0.85, 0.65 + 0.02(i - 21))$
Partial payments: Number[1]		For $s_{ir} \leq 7500$, $\mathrm{Prob}(M_{ir} = 1) = \mathrm{Prob}(M_{ir} = 2) = 1/2$ For $7500 \leq s_{ir} \leq 15000$, $\mathrm{Prob}(M_{ir} = 2) = 1/3$, $\mathrm{Prob}(M_{ir} = 3) = 2/3$ For $s_{ir} > 15000$, distribution of M_{ir} is geometric, with minimum 4 and mean = $\min(8, 4 + \ln(s_{ir}/15000))$
Partial payments: Amounts[1]	$F_{P\|s} =$ Beta	Mean = $1 - \min(0.95, 0.75 + 0.04\ln(s_{ir}/20000))$ Coefficient of variation = 20%
	$F_{\overline{Q}\|s} =$ Beta	Mean = 0.9 Coefficient of variation = 3%
	$F_{\hat{P}\|m} =$ Beta	Mean = $\left(1 - \left(p_{ir}^{(m_{ir}-1)} + p_{ir}^{(m_{ir})}\right)\right) / (m_{ir} - 2)$ Coefficient of variation = 10%
Distribution of payments over time	$F_{\hat{D}\|m} =$ Weibull	For $m_{ir} \geq 4$ and $m = m_{ir}$, Mean = 3 months (converted to the relevant time units) Coefficient of variation = 20% For $m_{ir} \geq 4$ and $m < m_{ir}$, or $m_{ir} < 4$, Mean = mean claim closure delay (from notification) / m_{ir} Coefficient of variation = 35%
Base inflation	$f(\bar{t})$	$= (1 + \alpha)^{\bar{t}}$, where α is equivalent to an increase of 2% p.a., allowing for the relevant time units
Superimposed inflation[1]	$g_O(u\|s)$	$= 1$ for $u \leq 20 = 1 - 0.4\max(0, 1 - s/50000)$ for $u > 20$
	$g_C(\bar{t}\|s)$	$= (1 + \beta(s))^{\bar{t}}$ with $\beta = \gamma\max(0, 1 - s/200000)$ and γ is equivalent to a 30% p.a. inflation rate, allowing for the relevant time units

Notes:
[1] This component is defined in terms of claim size. The definition here displays claim size in raw (uninflated) units. The inputs to the example application of `SynthETIC`, on the other hand, express claim sizes as multiples of a reference claim size equal to 200,000. For example, the amount of 100,000 that appears in the definition of claim notification delay is expressed in `SynthETIC` as $0.5 \times 200,000$.
[2] Exposure and claim frequency are annual effective parameters.

[8] Avanzi B.; Taylor G.; Wang M.; Wong B. SynthETIC: An individual insurance claim simulator with feature control. June 29, 2021. Insurance: Mathematics and Economics, 100, 296-308. ISSN 01676687, doi: 10.1016/j.insmatheco.2021.06.04. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/S0167668721001013412707 3

ANEXO 2: CÓDIGO EN R STUDIO

```
#==============================================================================
#                        PREVIO
#==============================================================================

# Limpieza de espacio de trabajo
rm(list=ls())

# Establecimiento de directorio de trabajo
setwd("../EstimacionReservaSiniestros")
getwd()

## Carga de paquetes a utilizar
install.packages("SynthETIC")  # -> Para simulación de siniestros
install.packages("plyr")      # -> Para round_any
install.packages("locfit")
install.packages("dplyr")
install.packages("actuar")
install.packages("tidyr")
install.packages("reshape2")
install.packages("ChainLadder", dependencies = TRUE)
install.packages('Rcpp')
install.packages("dplyr")     # -> library(xgboost)
install.packages("ggplot2")   # -> library(caret)
install.packages("lattice")   # -> library(caret)
install.packages("caTools")   # -> library(CaTools)  # para usar "sample.split"
install.packages('openxlsx')  # -> pra exportar en formato xlsx

#==============================================================================
#            SIMULACIÓN DE LA BASE DE DATOS DE SINIESTROS
#==============================================================================
library(SynthETIC)  # -> para función "data.generation"
library(dplyr)      # -> para función "set_parameters"

ref_claim = 1           # currency is 1
time_unit = 1/12        # 12 we consider monthly claims development
set_parameters(ref_claim = ref_claim, time_unit = time_unit)

years = 10              # number of occurrence years
I    = years/time_unit  # number of development periods

# Ruta donde se encuentran los códigos complementarios para la simulación de la
# base de datos.
source("./Tools/functions simulation.R")

# Ruta donde se guardan los datos simulados, resultados y gráficos
PathData    = "./Data/"
PathResults = "./Results/"
PathPlot    = "./Plots/"

# GENERACIÓN de siniestros individuales: tarda al rededor de 1 minuto
claims_list = data.generation(seed = 145, future_info = FALSE)  # 10 años

# Siniestros simulados
siniestros = claims_list[[1]]
pagos      = claims_list[[2]] # Contiene detalle de pagos parciales para triángulo agregado
```

```
# Tratamiento adicional de la base de datos simulada
siniestros$AccYearMonth = as.integer(x = format(x = siniestros$AccDate,'%m'))
siniestros$RepYearMonth = as.integer(x = format(x = siniestros$RepDate,'%m'))
siniestros$SetDelDays   = as.numeric(x = siniestros$SetDelMonths * 30)
siniestros$RepDelMonths = as.numeric(x = siniestros$RepDelDays / 30)
siniestros$AccYear      = as.integer(x = format(x = siniestros$AccDate,'%Y'))
siniestros$RepYear      = as.integer(x = format(x = siniestros$RepDate,'%Y'))

# Guardando las bases de datos en el directorio de trabajo
write.csv(x = siniestros, file = paste(PathData, "siniestros.csv", sep = ""))
write.csv(x = pagos, file = paste(PathData, "pagos.csv", sep = ""))

#==============================================================================
#             SINIESTROS OBSERVADOS POR AÑO DE ACCIDENTE
#==============================================================================
library(dplyr)

# Pagos y costo final observado por año de accidente de todos los siniestros
totales_aggr = siniestros %>%
  dplyr::group_by(AccYear) %>%
  dplyr::summarise(num = n(), ult = sum(Ultimate)/1000, pag = sum(CumPaid)/1000,
              res = (sum(Ultimate) - sum(CumPaid))/1000)

# Costo final observado por año de accidente de los siniestros cerrados
cerrados_aggr = siniestros[siniestros$Status == "Closed", ] %>%
  dplyr::group_by(AccYear) %>%
  dplyr::summarise(num = n(), ult = sum(Ultimate)/1000, pag = sum(CumPaid)/1000,
              res = (sum(Ultimate) - sum(CumPaid))/1000, min = min(Ultimate), max = max(Ultimate))


# Costo final observado por año de accidente de los siniestros pendientes
pendientes_aggr = siniestros[siniestros$Status == "RBNS", ] %>%
  dplyr::group_by(AccYear) %>%
  dplyr::summarise(num = n(), ult = sum(Ultimate)/1000, pag = sum(CumPaid)/1000,
              res = (sum(Ultimate) - sum(CumPaid))/1000)

### Hay siniestros pendientes desde el 2013. Debemos agregar el año 2012 para
### tener la misma cantidad de años que el resto de tipos de siniestros (cerrados)

adicional      = data.frame("AccYear" = 2012, "num" = 0 ,"ult" = 0, "pag" = 0,
                    "res" = 0)
pendientes_aggr = rbind(adicional, pendientes_aggr)
pendientes_aggr

# Costo final observado por año de accidente de los siniestros ibnr
num_ibnr = totales_aggr$num - cerrados_aggr$num - pendientes_aggr$num
ult_ibnr = totales_aggr$ult - cerrados_aggr$ult - pendientes_aggr$ult
ult_ibnr = round(x = ult_ibnr, digits = 0)
pag_ibnr = totales_aggr$pag - cerrados_aggr$pag - pendientes_aggr$pag
pag_ibnr = round(x = pag_ibnr, digits = 0)
res_ibnr = totales_aggr$res - cerrados_aggr$res - pendientes_aggr$res
res_ibnr = round(x = res_ibnr, digits = 0)

# Guardando los ultimates observados por año de accidente, en miles de euros
siniestros_aggr_bystatus = cbind(totales_aggr$num, totales_aggr$ult, totales_aggr$pag,
                         totales_aggr$res, cerrados_aggr$num, cerrados_aggr$ult,
                         cerrados_aggr$pag, cerrados_aggr$res, pendientes_aggr$num,
```

```
                    pendientes_aggr$ult, pendientes_aggr$pag, pendientes_aggr$res,
                    num_ibnr, ult_ibnr, pag_ibnr, res_ibnr)

colnames(x = siniestros_aggr_bystatus) = c("num_tot","ult_tot", "pag_tot","res_tot",
                            "num_cerr", "ult_cerr", "pag_cerr", "res_cerr",
                            "num_pend", "ult_pend", "pag_pend", "res_pend",
                            "num_ibnr", "ult_ibnr", "pag_ibnr", "res_ibnr")

write.xlsx(x = as.data.frame(siniestros_aggr_bystatus), file = paste(PathResults,
"siniestros_aggr_bystatus.xlsx",
                                        sep = ""))

#=============================================================================
#                CONSTRUCCIÓN DEL TRIÁNGULO DE PAGOS
#=============================================================================
library(reshape2)    # -> para función "dcast"
library(openxlsx)    # -> para función "write.xlsx"
library(ChainLadder) # -> para función "as.triangle"

# Datos con los pagos parciales
pagos_parc = merge(x = pagos, y = siniestros, by = "Id")

# Periodos de desarrollo
pagos_parc$AccYear  = ceiling((pagos_parc$AccMonth - 1/2)/12)
pagos_parc$RepYear  = ceiling((pagos_parc$RepMonth - 1/2)/12)
pagos_parc$PayYear  = ceiling((pagos_parc$EventMonth - 1/2)/12)
pagos_parc$PayDelay = pagos_parc$PayYear - pagos_parc$AccYear

# Triángulo de pagos incrementales en miles de euros
triang_pagos = dcast(pagos_parc, AccYear ~ PayDelay, sum, value.var = "Paid")
triang_pagos = triang_pagos[ , -1] / 1000
J = ncol(x = triang_pagos)

# Triángulo pagos acumulados en miles de euros
for (j in 2:J){
  triang_pagos[ , j] = triang_pagos[ , j] + triang_pagos[, j-1]
  triang_pagos[(J-j+2):J, j] = NA
}

triang_pagos = as.triangle(Triangle = as.matrix(x = triang_pagos))
triang_pagos

# Guardando el triángulo
write.csv(x = triang_pagos, file = paste(PathData, "triang_pagos.csv", sep = ""))

#=============================================================================
#       ESTIMACIÓN DE LA RESERVA DE SINIESTROS CON MÉTODOS AGREGADOS
# La reserva de siniestros total incluye la reserva de siniestros pendientes
# de liquidación y la reserva de siniestros ocurridos y no reportados (IBNR)
#=============================================================================

#::::::::::::::::::::::::::::::::::::::::::: MÉTODO MACK CHAIN-LADDER ::::::::::
library(ChainLadder) # -> para función "MackChainLadder"

unidades = 1
mack = MackChainLadder(Triangle = triang_pagos/unidades, est.sigma = "Mack")
```

```
results_mack = round(x = cbind(totales_aggr$ult, mack$FullTriangle[, J],
                        mack$FullTriangle[, J] - totales_aggr$ult,
                        mack$Mack.S.E[, J]))
colnames(x = results_mack) = c("ult_obs", "ult_est", "ult_dif", "S.E.")

# Añadiendo columna con el % de error de la estimación: RMSE
RMSE_mack = round(x = abs(results_mack[,3]/results_mack[,4]), digits = 2)
results_mack = cbind(results_mack, RMSE_mack)

# Añadiendo columnas con las reservas totales
res_obs = round(x = totales_aggr$res)
res_est = round(x = mack$FullTriangle[, J] - totales_aggr$pag)
res_dif = res_est - res_obs

results_mack = cbind(results_mack, res_obs, res_est, res_dif)
results_mack


#### Guardando resultados
write.xlsx(x = as.data.frame(results_mack), file = paste(PathResults, "results_mack.xlsx", sep = ""))

#### qq plot de los ultimates
jpeg(filename = paste(PathResults, "mack_ult_qq.jpeg", sep = ""),
    width = 400, height = 400)

plot(x = results_mack[1:10, "ult_obs"], y = results_mack[1:10, "ult_est"],
    xlim = c(45000,71000), ylim= c(45000,71000),
    xlab = "Costo observado (miles de €)",  ylab = "Costo estimado (miles de €)",
    pch = 20, cex = 2, col = "cyan4",
    main = list("Costo último estimado con Mack", cex = 1.2))
abline(a = 0, b = 1, lty = 1, col = "red")
axis(side = 1, tck = 1, lty = 2, col = "gray")  # grid
axis(side = 2, tck = 1, lty = 2, col = "gray")  # grid
box()

dev.off()

# Gráfico de barras de los ultimates
modified_data = results_mack[c(1:10), c("ult_obs","ult_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)


jpeg(filename = paste(PathResults, "mack_ult_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
      ylim = c(0,71000), col = c("snow3","cyan3"),
      xlab = "Año de ocurrencia", ylab = "Costo último (miles de €)",
      main = "Costo último estimado con Mack")
legend(x = "topleft", legend = c("Observado","Estimado Mack"),  bty = "n",
     pch = 15, cex = 1, col = c("snow3","cyan3"))

dev.off()

# Gráfico de barras de las reservas totales
modified_data = results_mack[c(1:10), c("res_obs","res_est")]
```

```
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "mack_res_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
       ylim = c(0,71000), col = c("khaki1","darkolivegreen4"),
       xlab = "Año de ocurrencia", ylab = "Reserva (miles de €)",
       main = "Reserva total estimada con Mack")
legend(x = "topleft", legend = c("Observado","Estimado Mack"),  bty = "n",
      pch = 15, cex = 1, col = c("khaki1","darkolivegreen4"))

dev.off()

#:::::::::::::::::::::::::::::::::::::: MÉTODO BOOTSTRAP CHAIN-LADDER ::::::::::
library(ChainLadder)  # -> para función "BootChainLadder"
library(openxlsx)     # -> para función "write.xlsx"

unidades = 1
boot = BootChainLadder(Triangle = triang_pagos/unidades, R = 100000,
                  process.distr = c("gamma"))

boot_byorig = summary(object = boot)$ByOrigin
boot_tot    = summary(object = boot)$Totals

results_boot = round(x = cbind(totales_aggr$ult, boot_byorig$`Mean Ultimate`,
                         boot_byorig$`Mean Ultimate` - totales_aggr$ult,
                         boot_byorig$`SD IBNR`))
colnames(x = results_boot) = c("ult_obs", "ult_est", "dif", "S.E.")


# Añadiendo columnas con las reservas totales
res_obs = round(x = totales_aggr$res)
res_est = round(x = boot_byorig$`Mean Ultimate` - totales_aggr$pag)
res_dif = res_est - res_obs

results_boot = cbind(results_boot, res_obs, res_est, res_dif)
results_boot

#### Guardando resultados
write.xlsx(x = as.data.frame(results_boot), file = paste(PathResults, "results_boot.xlsx", sep = ""))

#### qq plot de los ultimates
jpeg(filename = paste(PathResults, "boot_ult_qq.jpeg", sep = ""),
    width = 400, height = 400)

plot(x = results_boot[1:10, "ult_obs"], y = results_boot[1:10, "ult_est"],
    xlim = c(45000,71000), ylim= c(45000,71000),
    xlab = "Costo observado (miles de €)",  ylab = "Costo estimado (miles de €)",
    pch = 20, cex = 2, col = "cyan4",
    main = list("Costo último estimado con Bootstrap", cex = 1.2))
abline(a = 0, b = 1, lty = 1, col = "red")
axis(side = 1, tck = 1, lty = 2, col = "gray")  # grid
axis(side = 2, tck = 1, lty = 2, col = "gray")  # grid
box()
```

```
dev.off()

#### Gráfico de barras de los ultimates
modified_data = results_boot[c(1:10), c("ult_obs","ult_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)


jpeg(filename = paste(PathResults, "boot_ult_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
       ylim = c(0,71000), col = c("snow3","cyan3"),
       xlab = "Año de ocurrencia", ylab = "Costo último (miles de €)",
       main = "Costo último estimado por Bootstrap")
legend(x = "topleft", legend = c("Observado","Estimado Bootstrap"),  bty = "n",
      pch = 15, cex = 1, col = c("snow3","cyan3"))

dev.off()

# Gráfico de barras de las reservas totales
modified_data = results_boot[c(1:10), c("res_obs","res_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)


jpeg(filename = paste(PathResults, "boot_res_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
       ylim = c(0,71000), col = c("khaki1","darkolivegreen4"),
       xlab = "Año de ocurrencia", ylab = "Reserva (miles de €)",
       main = "Reserva total estimada con Bootstrap")
legend(x = "topleft", legend = c("Observado","Estimado Bootstrap"),  bty = "n",
      pch = 15, cex = 1, col = c("khaki1","darkolivegreen4"))

dev.off()

#=============================================================================
#            ESTIMACIÓN DEL COSTO ÚLTIMO CON MÉTODOS INDIVIDUALES
#=============================================================================
library(caret)   # -> para función "createFolds"

# Datos con los que se entrenan los modelos
BD = siniestros[siniestros$Status == "Closed", ]
BD = BD[ , c("Id","AccYear","RepYear","Type","Age","AccYearMonth","RepYearMonth",
          "AccWeekday","RepDelDays","Ultimate")]

# Escalando los predictores numéricos
scaledBD = BD
scaledBD[ , c("Age","RepDelDays")] = scale(x = scaledBD[ , c("Age","RepDelDays")])

# Dividiendo los datos en k = 10 grupos para aplicar validación cruzada y
# calcular el RMSE de cada modelo.
folds = createFolds(y = scaledBD$Ultimate, k = 10)  # Folds identifica los índices
str(object = folds)
```

```
max = max(scaledBD$Ultimate)
min = min(scaledBD$Ultimate)

#:::::::::::::: ALGORITMO MACHINE LEARNING 1: GENERALIZED LINEAR MODEL ::::::::::

# RMSE hallado con Cross Validation  (k = 10)
cvMSE_glm = lapply(X = folds, FUN = function(x){
  train_fold = scaledBD[-x, 4:10]
  test_fold  = scaledBD[x, 4:10]

  model     = glm(formula = train_fold$Ultimate ~ ., data = train_fold,
                  family = gaussian(link = "identity"))
  pred_y    = predict(object = model, newdata = test_fold)
  MSE       = mean(x = (test_fold[ , 7] - pred_y)^2)

})

cvMSE_glm   = as.numeric(x = cvMSE_glm)
cvRMSE_glm  = sqrt(x = mean(x = cvMSE_glm))
cvNRMSE_glm = cvRMSE_glm / (max - min)
cvNRMSE_glm

# Eligiendo el modelo con el menor RMSE o MSE
num_model_glm = which(cvMSE_glm == min(cvMSE_glm))
num_model_glm

# Obteniendo el modelo elegido
ind_glm = unlist(x = folds[num_model_glm])

train     = scaledBD[-ind_glm, 4:10]
model_glm = glm(formula = train$Ultimate ~ ., data = train,
                family = gaussian(link = "identity"))

# Aplicando el modelo a los datos de prueba
test      = scaledBD[ind_glm, 4:10]
pred_glm_y = predict(object = model_glm, newdata = test)

# Comparando las densidades empíricas de los datos de prueba
jpeg(filename = paste(PathResults, "densidad_glm.jpeg", sep = ""),
    width = 400, height = 250)

plot(x = density(x = test[ , 7]/1000, from = 0, to = 80),
    col  = "snow4", lwd = 2, lty = 1, ylim = c(0, 0.16),
    main = list("Densidad empírica del costo último individual", cex = 1),
    cex.lab = 1,
    ylab = "Densidad empírica", xlab = "Costo último (miles de €)")

lines(x = density(x = pred_glm_y/1000, from = 0, to = 80), col="cyan3",
     lwd = 2, lty = 1)

legend(x = "topright", cex = 1,  lty = rep(x = 1, 2), lwd = c(2,2), bty = "n",
      col = c("snow4","cyan3"),
      legend = c("Observado (siniestros cerrados)", "Estimado GLM Gaussiano"))

dev.off()
```

```
#:::::::::::::::::: ALGORITMO MACHINE LEARNING 2: K NEAREST NEIGHBORS ::::::::::
library(class)  # -> para función "knnreg"
library(caret)

# RMSE hallado con Cross Validation  (k = 10)
cvMSE_knn = lapply(X = folds, FUN = function(x){
  train_fold = scaledBD[-x, 4:10]
  test_fold  = scaledBD[x, 4:10]

  model     = knnreg(formula = train_fold$Ultimate ~ ., data = train_fold, k = 16)
  pred_y    = predict(object = model, newdata = test_fold)
  MSE       = mean(x = (test_fold[ , 7] - pred_y)^2)

})

cvMSE_knn   = as.numeric(x = cvMSE_knn)
cvRMSE_knn  = sqrt(x = mean(x = cvMSE_knn))
cvNRMSE_knn = cvRMSE_knn / (max - min)
cvNRMSE_knn

# Eligiendo el modelo con el menor RMSE o MSE
num_model_knn = which(cvMSE_knn == min(cvMSE_knn))
num_model_knn

# Obteniendo el modelo elegido
ind_knn = unlist(x = folds[num_model_knn])

train     = scaledBD[-ind_knn, 4:10]
model_knn = knnreg(formula = train$Ultimate ~ ., data = train, k = 16)

# Aplicando el modelo a los datos de prueba
test      = scaledBD[ind_knn, 4:10]
pred_knn_y = predict(object = model_knn, newdata = test)

# Comparando las densidades empíricas de los datos de prueba
jpeg(filename = paste(PathResults, "densidad_knn.jpeg", sep = ""),
    width = 400, height = 250)

plot(x = density(x = test[ , 7]/1000, from = 0, to = 80),
    col  = "snow4", lwd = 2, lty = 1, ylim = c(0, 0.18),
    main = list("Densidad empírica del costo último individual", cex = 1),
    cex.lab = 1,
    ylab = "Densidad empírica", xlab = "Costo último (miles de €)")

lines(x = density(x = pred_knn_y/1000, from = 0, to = 80), col="cyan3",
     lwd = 2, lty = 1)

legend(x = "topright", cex = 1,  lty = rep(x = 1, 2), lwd = c(2,2), bty = "n",
      col = c("snow4","cyan3"),
      legend = c("Observado (siniestros cerrados)", "Estimado kNN"))

dev.off()


#:::::::::::::::::::::::: ALGORITMO MACHINE LEARNING 3: RANDOM FOREST ::::::::::
library(randomForest)  # -> para función "randomForest"
library(caret)
```

```
# RMSE hallado con Cross Validation  (k = 10)
cvMSE_rf = lapply(X = folds, FUN = function(x){
  train_fold = scaledBD[-x, 4:10]
  test_fold  = scaledBD[x, 4:10]

  model     = randomForest(formula = train_fold$Ultimate ~ ., data = train_fold,
                           ntree = 10)
  pred_y    = predict(object = model, newdata = test_fold)
  MSE       = mean(x = (test_fold[ , 7] - pred_y)^2)

})

cvMSE_rf   = as.numeric(x = cvMSE_rf)
cvRMSE_rf  = sqrt(x = mean(x = cvMSE_rf))
cvNRMSE_rf = cvRMSE_rf / (max - min)
cvNRMSE_rf

# Eligiendo el modelo con el menor RMSE o MSE
num_model_rf = which(cvMSE_rf == min(cvMSE_rf))
num_model_rf

# Obteniendo el modelo elegido
ind_rf = unlist(x = folds[num_model_rf])

train    = scaledBD[-ind_rf, 4:10]
model_rf = randomForest(formula = train$Ultimate ~ ., data = train, ntree = 10)

# Aplicando el modelo a los datos de prueba
test     = scaledBD[ind_rf, 4:10]
pred_rf_y = predict(object = model_rf, newdata = test)

# Comparando las densidades empíricas de los datos de prueba
jpeg(filename = paste(PathResults, "densidad_rf.jpeg", sep = ""),
    width = 400, height = 250)

plot(x = density(x = test[ , 7]/1000, from = 0, to = 80),
    col  = "snow4", lwd = 2, lty = 1, ylim = c(0, 0.16),
    main = list("Densidad empírica del costo último individual", cex = 1),
    cex.lab = 1,
    ylab = "Densidad empírica", xlab = "Costo último (miles de €)")

lines(x = density(x = pred_rf_y/1000, from = 0, to = 80), col="cyan3",
     lwd = 2, lty = 1)

legend(x = "topright", cex = 1,  lty = rep(x = 1, 2), lwd = c(2,2), bty = "n",
      col = c("snow4","cyan3"),
      legend = c("Observado (siniestros cerrados)", "Estimado Random Forest"))

dev.off()

#:::::::::::: ALGORITMO MACHINE LEARNING 4: GRADIENT BOOSTING MACHINE ::::::::::
library(gbm)    # -> para función "gbm"
library(caret)

# RMSE hallado con Cross Validation  (k = 10)
cvMSE_gbm = lapply(X = folds, FUN = function(x){
  train_fold = scaledBD[-x, 4:10]
```

```
  test_fold  = scaledBD[x, 4:10]

  model      = gbm(formula = train_fold$Ultimate ~ ., data = train_fold,
               distribution = "gaussian", cv.folds = 10, shrinkage = .01,
               n.minobsinnode = 10, n.trees = 10)
  pred_y     = predict(object = model, newdata = test_fold)
  MSE        = mean(x = (test_fold[ , 7] - pred_y)^2)

})

cvMSE_gbm   = as.numeric(x = cvMSE_gbm)
cvRMSE_gbm  = sqrt(x = mean(x = cvMSE_gbm))
cvNRMSE_gbm = cvRMSE_gbm / (max - min)
cvNRMSE_gbm

# Eligiendo el modelo con el menor RMSE o MSE
num_model_gbm = which(cvMSE_gbm == min(cvMSE_gbm))
num_model_gbm

# Obteniendo el modelo elegido
ind_gbm = unlist(x = folds[num_model_gbm])

train     = scaledBD[-ind_gbm, 4:10]
model_gbm = gbm(formula = train$Ultimate ~ ., data = train,
              distribution = "gaussian", cv.folds = 10, shrinkage = .01,
              n.minobsinnode = 10, n.trees = 10)

# Aplicando el modelo a los datos de prueba
test      = scaledBD[ind_gbm, 4:10]
pred_gbm_y = predict(object = model_gbm, newdata = test)

# Comparando las densidades empíricas de los datos de prueba
jpeg(filename = paste(PathResults, "densidad_gbm.jpeg", sep = ""),
    width = 400, height = 250)

plot(x = density(x = test[ , 7]/1000, from = 0, to = 80),
    col  = "snow4", lwd = 2, lty = 1, ylim = c(0, 0.2),
    main = list("Densidad empírica del costo último individual", cex = 1),
    cex.lab = 1,
    ylab = "Densidad empírica", xlab = "Costo último (miles de €)")

lines(x = density(x = pred_gbm_y/1000, from = 0, to = 80), col="cyan3",
     lwd = 2, lty = 1)

legend(x = "topright", cex = 1,  lty = rep(x = 1, 2), lwd = c(2,2), bty = "n",
      col = c("snow4","cyan3"),
      legend = c("Observado (siniestros cerrados)", "Estimado GBM"))

dev.off()


#:::::::::::: ALGORITMO MACHINE LEARNING 5: EXTREME GRADIENT BOOSTING ::::::::::
library(xgboost)  # -> para funciones "xgb.DMatrix" y "xgboost"
library(caret)
# RMSE hallado con Cross Validation  (k = 10)
cvMSE_xgb = lapply(X = folds, FUN = function(x){
  train_fold_x = scaledBD[-x, 4:9]
```

```
  train_fold_y = scaledBD[-x, 10]

  test_fold_x  = scaledBD[x, 4:9]
  test_fold_y  = scaledBD[x, 10]

  train_fold_xgb = xgb.DMatrix(data = data.matrix(train_fold_x), label = train_fold_y)
  test_fold_xgb  = xgb.DMatrix(data = data.matrix(test_fold_x), label = test_fold_y)

  model      = xgboost(data = train_fold_xgb, max.depth = 2, nrounds = 50)
  pred_y     = predict(object = model, newdata = test_fold_xgb)
  MSE        = mean(x = (test_fold_y - pred_y)^2)

})

cvMSE_xgb   = as.numeric(x = cvMSE_xgb)
cvRMSE_xgb  = sqrt(x = mean(x = cvMSE_xgb))
cvNRMSE_xgb = cvRMSE_xgb / (max - min)
cvNRMSE_xgb

# Eligiendo el modelo con el menor RMSE o MSE
num_model_xgb = which(cvMSE_xgb == min(cvMSE_xgb))
num_model_xgb

# Obteniendo el modelo elegido
ind_xgb = unlist(x = folds[num_model_xgb])

train_x = scaledBD[-ind_xgb, 4:9]
train_y = scaledBD[-ind_xgb, 10]

train_xgb  = xgb.DMatrix(data = data.matrix(train_x), label = train_y)
model_xgb  = xgboost(data = train_xgb, max.depth = 2, nrounds = 50)

# Aplicando el modelo a los datos de prueba
test_x = scaledBD[ind_xgb, 4:9]
test_y = scaledBD[ind_xgb, 10]

test_xgb   = xgb.DMatrix(data = data.matrix(test_x), label = test_y)
pred_xgb_y = predict(object = model_xgb, newdata = test_xgb)

# Comparando las densidades empíricas de los datos de prueba
jpeg(filename = paste(PathResults, "densidad_xgb.jpeg", sep = ""),
    width = 400, height = 250)

plot(x = density(x = test_y/1000, from = 0, to = 80),
    col  = "snow4", lwd = 2, lty = 1, ylim = c(0, 0.16),
    main = list("Densidad empírica del costo último individual", cex = 1),
    cex.lab = 1,
    ylab = "Densidad empírica", xlab = "Costo último (miles de €)")

lines(x = density(x = pred_xgb_y/1000, from = 0, to = 80), col="cyan3",
     lwd = 2, lty = 1)

legend(x = "topright", cex = 1,  lty = rep(x = 1, 2), lwd = c(2,2), bty = "n",
      col = c("snow4","cyan3"),
      legend = c("Observado (siniestros cerrados)", "Estimado XGB"))

dev.off()
```

```
#=============================================================================
#   GRÁFICO CONJUNTO DE LAS DENSIDADES EMPÍRICAS DE LOS MODELOS INDIVIDUALES
#=============================================================================
num_model = 1
ind      = unlist(x = folds[num_model])
test     = scaledBD[ind, 4:10]


jpeg(filename = paste(PathResults, "densidad_modelosML.jpeg", sep = ""),
    width = 400, height = 250)

plot(x = density(x = test$Ultimate/1000, from = 0, to = 60),
    col  = "snow4", lwd = 2, lty = "dashed", ylim = c(0, 0.17),
    main = list("Densidad empírica del costo último individual", cex = 1),
    cex.lab = 1,
    ylab = "Densidad empírica", xlab = "Costo último (miles de €)")

lines(x = density(x = pred_glm_y/1000, from = 0, to = 60), col="cyan3",
     lwd = 1.5, lty = 1)

lines(x = density(x = pred_knn_y/1000, from = 0, to = 60), col="blue",
     lwd = 1.5, lty = 1)

lines(x = density(x = pred_rf_y/1000, from = 0, to = 60), col="darkorange1",
     lwd = 1.5, lty = 1)

lines(x = density(x = pred_xgb_y/1000, from = 0, to = 60), col="red",
     lwd = 1.5, lty = 1)

legend(x = "topright", cex = 1,  lty = rep(x = 1, 5), lwd = rep(x = 2, 5), bty = "n",
      col = c("snow4","cyan3","blue","darkorange1","red"),
      legend = c("Observado (siniestros cerrados)","Estimado GLM","Estimado kNN",
             "Estimado Random Forest", "Estimado XGB"))

dev.off()

#=============================================================================
# ESTIMACIÓN DEL COSTO ÚLTIMO DE LOS SINIESTROS PENDIENTES CON MÉTODOS INDIVIDUALES
#=============================================================================
# Siniestros pendientes
BD_pend = siniestros[siniestros$Status == "RBNS", ]
BD_pend = BD_pend[ , c("Id","AccYear","RepYear","Type","Age","AccYearMonth",
                 "RepYearMonth", "AccWeekday","RepDelDays","Ultimate")]


# Escalando los predictores numéricos
scaledBD_pend = BD_pend
scaledBD_pend[ , c("Age","RepDelDays")] = scale(x = scaledBD_pend[ , c("Age","RepDelDays")])


#::::::::::::::::::: ALGORITMO MACHINE LEARNING: K NEAREST NEIGHBORS ::::::::::

# Aplicando el modelo
pred_knn_pend = predict(object = model_knn, newdata = scaledBD_pend[ , 4:10])

# Agregando las estimaciones del costo último por año de ocurrencia
modified_data = cbind(scaledBD_pend$AccYear, scaledBD_pend$Ultimate/1000,
```

```
            pred_knn_pend/1000,
            (pred_knn_pend - scaledBD_pend$Ultimate)/1000)
colnames(x = modified_data) = c("AccYear", "ult_obs", "ult_est", "ult_dif")
modified_data = as.data.frame(x = modified_data)

results_knn_pend = aggregate(x = modified_data, by = list(modified_data$AccYear), FUN = sum)
results_knn_pend = round(x = results_knn_pend)
results_knn_pend = results_knn_pend[ , -c(2:2)]
colnames(x = results_knn_pend) = c("AccYear", "ult_obs", "ult_est", "ult_dif")
results_knn_pend

### Agregando datos para el año 2012
adicional = data.frame("AccYear" = 2012, "ult_obs" = 0, "ult_est" = 0, "ult_dif" = 0)

results_knn_pend = rbind(adicional, results_knn_pend)
results_knn_pend

# Restando al costo último estimado los pagos ya realizados, para obtener la
# reserva estimada de dichos siniestros pendientes

res_est = results_knn_pend$ult_est - as.data.frame(x = siniestros_aggr_bystatus)$pag_pend
res_obs = as.data.frame(x = siniestros_aggr_bystatus)$res_pend
res_dif = res_est - res_obs

results_knn_pend = cbind(results_knn_pend, res_obs, res_est, res_dif)
results_knn_pend

# Guardando resultados
write.xlsx(x = results_knn_pend, file = paste(PathResults, "results_knn_pend.xlsx", sep = ""),
        rowNames = FALSE)

# qq plot de los ultimates agregados por año de ocurrencia
jpeg(filename = paste(PathResults, "knn_pend_ult_qq.jpeg", sep = ""),
    width = 400, height = 400)

plot(x = results_knn_pend$ult_obs, y = results_knn_pend$ult_est,
    xlim = c(0,40000), ylim= c(0,40000),
    xlab = "Costo observado (miles de €)",  ylab = "Costo estimado (miles de €)",
    pch = 20, cex = 2, col = "cyan4",
    main = list("Costo último de los siniestros pendientes estimado con kNN", cex = 1))
abline(a = 0, b = 1, lty = 1, col = "red")
axis(side = 1, tck = 1, lty = 2, col = "gray")  # grid
axis(side = 2, tck = 1, lty = 2, col = "gray")  # grid
box()

dev.off()

# Gráfico de barras de los ultimates agregados por año de ocurrencia
modified_data = results_knn_pend[c(1:10), c("ult_obs","ult_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "knn_pend_ult_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
      ylim = c(0, 40000), col = c("snow3","cyan3"),
```

```
      xlab = "Año de ocurrencia", ylab = "Costo último (miles de €)",
      main = "Costo último de los siniestros pendientes estimado con kNN")
legend(x = "topleft", legend = c("Observado (siniestros pendientes)",
                         "Estimado k-Nearest-Neighbors"), bty = "n",
     pch = 15, cex = 1, col = c("snow3","cyan3"))

dev.off()

# Gráfico de barras de las reservas agregadas por año de ocurrencia
modified_data = results_knn_pend[c(1:10), c("res_obs","res_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "knn_pend_res_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
      ylim = c(0, 40000), col = c("khaki1","darkolivegreen4"),
      xlab = "Año de ocurrencia", ylab = "Importe reserva (miles de €)",
      main = "Reserva de los siniestros pendientes estimada con kNN")
legend(x = "topleft", legend = c("Observado (siniestros pendientes)",
                         "Estimado k-Nearest-Neighbors"), bty = "n",
     pch = 15, cex = 1, col = c("khaki1","darkolivegreen4"))

dev.off()

#:::::::::::::::::::::::::: ALGORITMO MACHINE LEARNING: RANDOM FOREST ::::::::::

# Aplicando el modelo
pred_rf_pend = predict(object = model_rf, newdata = scaledBD_pend[ , 4:10])

# Agregando las estimaciones del costo último por año de ocurrencia
modified_data = cbind(scaledBD_pend$AccYear, scaledBD_pend$Ultimate/1000,
                pred_rf_pend/1000,
                (pred_rf_pend - scaledBD_pend$Ultimate)/1000)
colnames(x = modified_data) = c("AccYear", "ult_obs", "ult_est", "ult_dif")
modified_data = as.data.frame(x = modified_data)

results_rf_pend = aggregate(x = modified_data, by = list(modified_data$AccYear), FUN = sum)
results_rf_pend = round(x = results_rf_pend)
results_rf_pend = results_rf_pend[ , -c(2:2)]
colnames(x = results_rf_pend) = c("AccYear", "ult_obs", "ult_est", "ult_dif")
results_rf_pend

### Agregando datos para el año 2012
adicional = data.frame("AccYear" = 2012, "ult_obs" = 0, "ult_est" = 0, "ult_dif" = 0)

results_rf_pend = rbind(adicional, results_rf_pend)
results_rf_pend

# Restando al costo último estimado los pagos ya realizados, para obtener la
# reserva estimada de dichos siniestros pendientes

res_est = results_rf_pend$ult_est - as.data.frame(x = siniestros_aggr_bystatus)$pag_pend
res_obs = as.data.frame(x = siniestros_aggr_bystatus)$res_pend
res_dif = res_est - res_obs
```

```
results_rf_pend = cbind(results_rf_pend, res_obs, res_est, res_dif)
results_rf_pend

# Guardando resultados
write.xlsx(x = results_rf_pend, file = paste(PathResults, "results_rf_pend.xlsx", sep = ""),
        rowNames = FALSE)

# qq plot de los ultimates agregados por año de ocurrencia
jpeg(filename = paste(PathResults, "rf_pend_ult_qq.jpeg", sep = ""),
    width = 400, height = 400)

plot(x = results_rf_pend$ult_obs, y = results_rf_pend$ult_est,
    xlim = c(0,40000), ylim= c(0,40000),
    xlab = "Costo observado (miles de €)",  ylab = "Costo estimado (miles de €)",
    pch = 20, cex = 2, col = "cyan4",
    main = list("Costo último de los siniestros pendientes estimado con Random Forest",
            cex = 1))
abline(a = 0, b = 1, lty = 1, col = "red")
axis(side = 1, tck = 1, lty = 2, col = "gray")  # grid
axis(side = 2, tck = 1, lty = 2, col = "gray")  # grid
box()

dev.off()

# Gráfico de barras de los ultimates agregados por año de ocurrencia
modified_data = results_rf_pend[c(1:10), c("ult_obs","ult_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "rf_pend_ult_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
      ylim = c(0, 40000), col = c("snow3","cyan3"),
      xlab = "Año de ocurrencia", ylab = "Costo último (miles de €)",
      main = "Costo último de los siniestros pendientes estimado con Random Forest")
legend(x = "topleft", legend = c("Observado (siniestros pendientes)",
                        "Estimado Random Forest"), bty = "n",
     pch = 15, cex = 1, col = c("snow3","cyan3"))

dev.off()

# Gráfico de barras de las reservas agregadas por año de ocurrencia
modified_data = results_rf_pend[c(1:10), c("res_obs","res_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "rf_pend_res_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
      ylim = c(0, 40000), col = c("khaki1","darkolivegreen4"),
      xlab = "Año de ocurrencia", ylab = "Importe reserva (miles de €)",
      main = "Reserva de los siniestros pendientes estimada con Random Forest")
legend(x = "topleft", legend = c("Observado (siniestros pendientes)",
                        "Estimado Random Forest"), bty = "n",
     pch = 15, cex = 1, col = c("khaki1","darkolivegreen4"))
```

```
dev.off()

#:::::::::::::: ALGORITMO MACHINE LEARNING: EXTREME GRADIENT BOOSTING ::::::::::

# Aplicando el modelo
scaledBD_pend_x = scaledBD_pend[ , 4:9]
scaledBD_pend_y = scaledBD_pend[ , 10]

scaledBD_pend_xgb = xgb.DMatrix(data = data.matrix(scaledBD_pend_x),
                      label = scaledBD_pend_y)
pred_xgb_pend     = predict(object = model_xgb, newdata = scaledBD_pend_xgb)

# Agregando las estimaciones del costo último por año de ocurrencia
modified_data = cbind(scaledBD_pend$AccYear, scaledBD_pend$Ultimate/1000,
                pred_xgb_pend/1000,
                (pred_xgb_pend - scaledBD_pend$Ultimate)/1000)
colnames(x = modified_data) = c("AccYear", "ult_obs", "ult_est", "ult_dif")
modified_data = as.data.frame(x = modified_data)

results_xgb_pend = aggregate(x = modified_data, by = list(modified_data$AccYear), FUN = sum)
results_xgb_pend = round(x = results_xgb_pend)
results_xgb_pend = results_xgb_pend[ , -c(2:2)]
colnames(x = results_xgb_pend) = c("AccYear", "ult_obs", "ult_est", "ult_dif")
results_xgb_pend

### Agregando datos para el año 2012
adicional = data.frame("AccYear" = 2012, "ult_obs" = 0, "ult_est" = 0, "ult_dif" = 0)

results_xgb_pend = rbind(adicional, results_xgb_pend)
results_xgb_pend

# Restando al costo último estimado los pagos ya realizados, para obtener la
# reserva estimada de dichos siniestros pendientes

res_est = results_xgb_pend$ult_est - as.data.frame(x = siniestros_aggr_bystatus)$pag_pend
res_obs = as.data.frame(x = siniestros_aggr_bystatus)$res_pend
res_dif = res_est - res_obs

results_xgb_pend = cbind(results_xgb_pend, res_obs, res_est, res_dif)
results_xgb_pend

# Guardando resultados
write.xlsx(x = results_xgb_pend, file = paste(PathResults, "results_xgb_pend.xlsx", sep = ""),
        rowNames = FALSE)

# qq plot de los ultimates agregados por año de ocurrencia
jpeg(filename = paste(PathResults, "xgb_pend_ult_qq.jpeg", sep = ""),
    width = 400, height = 400)

plot(x = results_xgb_pend$ult_obs, y = results_xgb_pend$ult_est,
    xlim = c(0,40000), ylim= c(0,40000),
    xlab = "Costo observado (miles de €)",  ylab = "Costo estimado (miles de €)",
    pch = 20, cex = 2, col = "cyan4",
    main = list("Costo último de los siniestros pendientes estimado con XGB",
            cex = 1))
abline(a = 0, b = 1, lty = 1, col = "red")
axis(side = 1, tck = 1, lty = 2, col = "gray")  # grid
```

```
axis(side = 2, tck = 1, lty = 2, col = "gray")  # grid
box()

dev.off()

# Gráfico de barras de los ultimates agregados por año de ocurrencia
modified_data = results_xgb_pend[c(1:10), c("ult_obs","ult_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "xgb_pend_ult_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
       ylim = c(0, 40000), col = c("snow3","cyan3"),
       xlab = "Año de ocurrencia", ylab = "Costo último (miles de €)",
       main = "Costo último de los siniestros pendientes estimado con XGB")
legend(x = "topleft", legend = c("Observado (siniestros pendientes)",
                         "Estimado Extreme Gradient Boosting"), bty = "n",
      pch = 15, cex = 1, col = c("snow3","cyan3"))

dev.off()

# Gráfico de barras de las reservas agregadas por año de ocurrencia
modified_data = results_xgb_pend[c(1:10), c("res_obs","res_est")]
modified_data = t(x = modified_data)
colnames(x = modified_data) = c(2012:2021)

jpeg(filename = paste(PathResults, "xgb_pend_res_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = modified_data, beside = TRUE, space = rep(x = 1:0, times = 10)/3,
       ylim = c(0, 40000), col = c("khaki1","darkolivegreen4"),
       xlab = "Año de ocurrencia", ylab = "Importe reserva (miles de €)",
       main = "Reserva de los siniestros pendientes estimada con XGB")
legend(x = "topleft", legend = c("Observado (siniestros pendientes)",
                         "Estimado Extreme Gradient Boosting"), bty = "n",
      pch = 15, cex = 1, col = c("khaki1","darkolivegreen4"))

dev.off()

#=============================================================================
#   GRÁFICO CONJUNTO DEL COSTO ÚLTIMO Y RESERVA DE LOS SINIESTROS PENDIENTES
#               ESTIMADOS CON LOS MÉTODOS INDIVIDUALES
#=============================================================================

# Preparando información para gráfico
tot_rf_pend  = colSums(x = results_rf_pend[ , 2:7])
tot_xgb_pend = colSums(x = results_xgb_pend[ , 2:7])
tot_knn_pend = colSums(x = results_knn_pend[ , 2:7])
tot_glm_pend = colSums(x = results_glm_pend[ , 2:7])

results_aggr_pend = cbind(tot_rf_pend, tot_xgb_pend, tot_knn_pend, tot_glm_pend)
results_aggr_pend[c("ult_dif","res_dif"), ] = - results_aggr_pend[c("ult_dif","res_dif"), ]
colnames(x = results_aggr_pend) = c("RandomForest","XGB","k-NN","GLM")

ult_obs = tot_glm_pend["ult_obs"]
```

```
ult_est = ult_obs
ult_dif = 0
res_obs = tot_glm_pend["res_obs"]
res_est = res_obs
res_dif = 0

obs = rbind(ult_obs, ult_est, ult_dif, res_obs, res_est, res_dif)
colnames(x = obs) = c("Observado")

results_aggr_pend = cbind(obs, results_aggr_pend)
results_aggr_pend

# Guardando resultados
write.csv(x = results_aggr_pend, file = paste(PathResults, "results_aggr_pend.csv", sep = ""))

# Gráfico de barras del costo último
jpeg(filename = paste(PathResults, "modelosML_pend_ult_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = results_aggr_pend[ c("ult_obs","ult_est","ult_dif"), 2:5], beside = TRUE,
       ylim = c(0, 150000), col = c("dodgerblue4","dodgerblue1","snow3"),
       xlab = "Algoritmo machine learning", ylab = "Costo último (miles de €)",
       main = "Costo último agregado de los siniestros pendientes")
legend(x = "topright", bty = "n",
      legend = c("Observado","Estimado", "Diferencia"),
      pch = 15, cex = 0.9, col = c("dodgerblue4","dodgerblue1","snow3"))

dev.off()

# Gráfico de barras de la reserva
jpeg(filename = paste(PathResults, "modelosML_pend_res_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = results_aggr_pend[ c("res_obs","res_est","res_dif"), 2:5], beside = TRUE,
       ylim = c(0, 120000), col = c("olivedrab","olivedrab3","snow3"),
       xlab = "Algoritmo machine learning", ylab = "Importe reserva (miles de €)",
       main = "Reserva de los siniestros pendientes (RSP)")
legend(x = "topright", bty = "n",
      legend = c("Observado","Estimado", "Diferencia"),
      pch = 15, cex = 0.9, col = c("olivedrab","olivedrab3","snow3"))

dev.off()

#=============================================================================
#    ESTIMACIÓN DEL COSTO ÚLTIMO DE LOS SINIESTROS NO REPORTADOS (IBNR)
#                    CON MÉTODOS INDIVIDUALES
#=============================================================================

#::::::::::::::::::::::::::::::::::::::::::: ESTIMACIÓN DEL RATIO IBNR ::::::::::
library(dplyr)

# Siniestros cerrados
BD_cerr = siniestros[siniestros$Status == "Closed", c("Id","AccYear","RepYear","Ultimate")]
BD_cerr$RepDelay = BD_cerr$RepYear - BD_cerr$AccYear
str(BD_cerr)

# Agrupamos los siniestros por año de ocurrencia y según el retardo en el reporte
```

```
# cerr_sin_retardo: Siniestros reportados en el mismo año que ocurrieron ("RepDelay" = 0)
# cerr_con_retardo: Siniestros reportados en un año distinto al del ocurrido ("RepDelay" >0)

cerr_sin_retardo = BD_cerr[(BD_cerr$RepDelay == 0 & BD_cerr$AccYear < 2021), ] %>%
  dplyr::group_by(AccYear) %>%
  dplyr::summarise(ult = sum(Ultimate))

cerr_con_retardo = BD_cerr[BD_cerr$RepDelay > 0, ] %>%
  dplyr::group_by(AccYear) %>%
  dplyr::summarise(ult = sum(Ultimate))

# Obtenemos los ratios para cada año de ocurrencia. Esto representa el porcentaje
# de siniestros reportados despúes del año en el que ocurrieron, aproximación a un ratio IBNR.
ratio_ibnr = cerr_con_retardo$ult / cerr_sin_retardo$ult
ratio_ibnr = data.frame("AccYear" = 2012:2018, "ratio" = ratio_ibnr[1:7])
ratio_ibnr

# Modelo para estimar el ratio_ibnr para los años que no se tiene información
model_ratio = lm(formula = ratio ~ poly(x = AccYear, degree = 1), data = ratio_ibnr)
summary(object = model_ratio)

# Error cuadrático medio del modelo
RMSE_ratio = RMSE(pred = model_ratio$fit, obs = ratio_ibnr$ratio)
RMSE_ratio

# Gráfico de estimaciones
est_ratio = predict(object = model_ratio, newdata = ratio_ibnr, se.fit = TRUE, level = 0.95)
IC_inf    = est_ratio$fit - 1.96*est_ratio$se.fit
IC_sup    = est_ratio$fit + 1.96*est_ratio$se.fit

jpeg(filename = paste(PathResults, "ratio_ibnr_ic.jpeg", sep = ""),
    width = 600, height = 300)

plot(x = ratio_ibnr$AccYear, y = ratio_ibnr$ratio, pch = 20, lwd = 3,
    ylim = c(0.3,0.5), col = "snow4", main = "Ajuste del ratio IBNR",
    xlab = "Año de ocurrencia", ylab = "Ratio")
lines(x = ratio_ibnr$AccYear, est_ratio$fit, col = "red", lwd = 1.5)
lines(x = ratio_ibnr$AccYear, IC_inf, col = "firebrick", lwd = 1, lty = "dashed")
lines(x = ratio_ibnr$AccYear, IC_sup, col = "firebrick", lwd = 1, lty = "dashed")
legend(x = "bottom", bty = "n", cex = 1, lty = c(1,1,3), lwd = c(2,2,2),
      col = c("snow4","red","firebrick"),
      legend = c("Observado", "Estimado: regresión lineal",
            "Intervalo de confianza"))

dev.off()

# Predicciones de los ratios para los años 2019, 2020 y 2021
newAccYear = data.frame("AccYear" = c(2019:2021))
pred_ratio_ibnr = predict(object = model_ratio, newdata = newAccYear)
names(x = pred_ratio_ibnr) = c("2019","2020","2021")
pred_ratio_ibnr

#::::::::::::::::::::::::::::::::::::::::: ESTIMACIÓN DEL IBNR ::::::::::::::::

# Preparación de la información, en miles de euros
BD_cerr_ibnr = siniestros[siniestros$Status == "Closed", c("Id","AccYear","RepYear","Ultimate")]
BD_cerr_ibnr = BD_cerr_ibnr[BD_cerr_ibnr$AccYear >= 2019, ]
```

```
cerr_aggr_ibnr = BD_cerr_ibnr  %>%
  dplyr::group_by(AccYear, RepYear) %>%
  dplyr::summarise(ult_obs = sum(Ultimate)/1000)
cerr_aggr_ibnr = as.data.frame(x = cerr_aggr_ibnr)
cerr_aggr_ibnr


BD_pend_ibnr = siniestros[siniestros$Status == "RBNS", c("Id","AccYear","RepYear")]
BD_pend_ibnr = cbind(BD_pend_ibnr,
                pred_rf_pend, pred_xgb_pend, pred_knn_pend, pred_glm_pend)
BD_pend_ibnr = BD_pend_ibnr[BD_pend_ibnr$AccYear >= 2019, ]

pend_aggr_ibnr = BD_pend_ibnr  %>%
  dplyr::group_by(AccYear, RepYear) %>%
  dplyr::summarise(ult_est_rf  = sum(pred_rf_pend)/1000,
                ult_est_xgb = sum(pred_xgb_pend)/1000,
                ult_est_knn = sum(pred_knn_pend)/1000,
                ult_est_glm = sum(pred_glm_pend)/1000,)
pend_aggr_ibnr = as.data.frame(x = pend_aggr_ibnr)
pend_aggr_ibnr


# Estimación de IBNR (en miles de euros) por modelo machine learning y año de ocurrencia

# Modelo: RANDOM FOREST
rf_ibnr_2019 = (cerr_aggr_ibnr[1,"ult_obs"] +
pend_aggr_ibnr[1,"ult_est_rf"])*pred_ratio_ibnr["2019"]
rf_ibnr_2019 = rf_ibnr_2019 - (cerr_aggr_ibnr[2,"ult_obs"] + cerr_aggr_ibnr[3,"ult_obs"]) -
  (pend_aggr_ibnr[2,"ult_est_rf"] + pend_aggr_ibnr[3,"ult_est_rf"])


rf_ibnr_2020 = (cerr_aggr_ibnr[4,"ult_obs"] +
pend_aggr_ibnr[4,"ult_est_rf"])*pred_ratio_ibnr["2020"]
rf_ibnr_2020 = rf_ibnr_2020 - cerr_aggr_ibnr[5,"ult_obs"] - pend_aggr_ibnr[5,"ult_est_rf"]

rf_ibnr_2021 = (cerr_aggr_ibnr[6,"ult_obs"] +
pend_aggr_ibnr[6,"ult_est_rf"])*pred_ratio_ibnr["2021"]

# Modelo: EXTREME GRADIENT BOOSTING
xgb_ibnr_2019 = (cerr_aggr_ibnr[1,"ult_obs"] +
pend_aggr_ibnr[1,"ult_est_xgb"])*pred_ratio_ibnr["2019"]
xgb_ibnr_2019 = xgb_ibnr_2019 - (cerr_aggr_ibnr[2,"ult_obs"] + cerr_aggr_ibnr[3,"ult_obs"]) -
  (pend_aggr_ibnr[2,"ult_est_xgb"] + pend_aggr_ibnr[3,"ult_est_xgb"])

xgb_ibnr_2020 = (cerr_aggr_ibnr[4,"ult_obs"] +
pend_aggr_ibnr[4,"ult_est_xgb"])*pred_ratio_ibnr["2020"]
xgb_ibnr_2020 = xgb_ibnr_2020 - cerr_aggr_ibnr[5,"ult_obs"] - pend_aggr_ibnr[5,"ult_est_xgb"]

xgb_ibnr_2021 = (cerr_aggr_ibnr[6,"ult_obs"] +
pend_aggr_ibnr[6,"ult_est_xgb"])*pred_ratio_ibnr["2021"]

# Modelo: K NEAREST NEIGHBORS
knn_ibnr_2019 = (cerr_aggr_ibnr[1,"ult_obs"] +
pend_aggr_ibnr[1,"ult_est_knn"])*pred_ratio_ibnr["2019"]
knn_ibnr_2019 = knn_ibnr_2019 - (cerr_aggr_ibnr[2,"ult_obs"] + cerr_aggr_ibnr[3,"ult_obs"]) -
  (pend_aggr_ibnr[2,"ult_est_knn"] + pend_aggr_ibnr[3,"ult_est_knn"])
```

```
knn_ibnr_2020 = (cerr_aggr_ibnr[4,"ult_obs"] +
pend_aggr_ibnr[4,"ult_est_knn"])*pred_ratio_ibnr["2020"]
knn_ibnr_2020 = knn_ibnr_2020 - cerr_aggr_ibnr[5,"ult_obs"] - pend_aggr_ibnr[5,"ult_est_knn"]

knn_ibnr_2021 = (cerr_aggr_ibnr[6,"ult_obs"] +
pend_aggr_ibnr[6,"ult_est_knn"])*pred_ratio_ibnr["2021"]


# Resumen de resultados ibnr en miles de euros
tot_obs_ibnr = sum(siniestros_aggr_bystatus[ ,"ult_ibnr"])
tot_rf_ibnr  = rf_ibnr_2019 + rf_ibnr_2020 + rf_ibnr_2021
tot_xgb_ibnr = xgb_ibnr_2019 + xgb_ibnr_2020 + xgb_ibnr_2021
tot_knn_ibnr = knn_ibnr_2019 + knn_ibnr_2020 + knn_ibnr_2021

ibnr_obs = rep(x = tot_obs_ibnr, times = 5)
ibnr_est = cbind(tot_obs_ibnr, tot_rf_ibnr, tot_xgb_ibnr, tot_knn_ibnr)
ibnr_dif = ibnr_obs - ibnr_est

results_aggr_ibnr = rbind(ibnr_obs, ibnr_est, ibnr_dif)
colnames(x = results_aggr_ibnr) = c("Observado","RandomForest","XGB","k-NN")
rownames(x = results_aggr_ibnr) = c("ibnr_obs","ibnr_est","ibnr_dif")
results_aggr_ibnr

# Guardando resultados
write.csv(x = results_aggr_ibnr, file = paste(PathResults, "results_aggr_ibnr.csv", sep = ""))

# Gráfico de barras de la reserva IBNR
jpeg(filename = paste(PathResults, "modelosML_ibnr_res_bar.jpeg", sep = ""),
    width = 500, height = 300)

barplot(height = results_aggr_ibnr[ c("ibnr_obs","ibnr_est","ibnr_dif"), 2:4], beside = TRUE,
       ylim = c(0, 25000), col = c("gold4","gold3","snow3"),
       xlab = "Algoritmo machine learning", ylab = "Importe reserva (miles de €)",
       main = list("Reserva de siniestros ocurridos y no reportados (IBNR)", cex = 1))
legend(x = "topright", bty = "n",
      legend = c("Observado","Estimado", "Diferencia"),
      pch = 15, cex = 0.9, col = c("gold4","gold3","snow3"))

dev.off()

#=============================================================================
#    GRÁFICO DE BARRAS DE LAS RESERVAS TOTALES ESTIMADAS POR MACHINE LEARNING
#=============================================================================

results_aggr_reservas_ml = rbind(results_aggr_pend, results_aggr_ibnr)
results_aggr_reservas_ml

jpeg(filename = paste(PathResults, "modelosML_res_tot_bar.jpeg", sep = ""),
    width = 600, height = 300)

barplot(height = results_aggr_reservas_ml[ c("res_est","ibnr_est"), 1:4], beside = FALSE,
       ylim = c(0, 120000), col = c("olivedrab","gold3"), #width = rep(x = 0.1, times = 5),
       xlab = "Algoritmo machine learning", ylab = "Importe reserva (miles de €)",
       main = list("Reserva total de siniestros", cex = 1.2))
legend(x = "topright", bty = "n",
      legend = c("Reserva de siniestros pendientes","Reserva IBNR"),
      pch = 15, cex = 1, col = c("olivedrab","gold3"))
```

```
dev.off()
#==============================================================================
#   GRÁFICO DE BARRAS DE LAS RESERVAS TOTALES ESTIMADAS POR TODOS LOS MÉTODOS
#               MACHINE LEARNING Y ESTOCÁSTICOS
#==============================================================================

tot_res_boot = rbind(sum(results_boot[ , "res_obs"]), sum(results_boot[ , "res_est"]),
               -sum(results_boot[ , "res_dif"]))
tot_res_mack = rbind(sum(results_mack[ , "res_obs"]), sum(results_mack[ , "res_est"]),
               -sum(results_mack[ , "res_dif"]))

tot_res_obs_ml = results_aggr_reservas_ml["res_obs", ] + results_aggr_reservas_ml["ibnr_obs", ]
tot_res_est_ml = results_aggr_reservas_ml["res_est", ] + results_aggr_reservas_ml["ibnr_est", ]
tot_res_dif_ml = results_aggr_reservas_ml["res_dif", ] + results_aggr_reservas_ml["ibnr_dif", ]

results_aggr_reservas_metodos = rbind(tot_res_obs_ml, tot_res_est_ml, tot_res_dif_ml)
results_aggr_reservas_metodos = cbind(results_aggr_reservas_metodos, tot_res_mack,
tot_res_boot)

colnames(x = results_aggr_reservas_metodos)[6:7] = c("Mack", "Bootstrap")
rownames(x = results_aggr_reservas_metodos)[1:3] = c("tot_res_obs", "tot_res_est","tot_res_dif")
results_aggr_reservas_metodos

# Guardando resultados
write.csv(x = results_aggr_reservas_metodos, file = paste(PathResults,
"results_aggr_reservas_metodos.csv", sep = ""))

jpeg(filename = paste(PathResults, "modelos_res_tot_bar.jpeg", sep = ""),
    width = 600, height = 350)

barplot(height = results_aggr_reservas_metodos[ c("tot_res_obs","tot_res_est","tot_res_dif"),
c(2:4,6:7)],
       beside = TRUE, ylim = c(-6000, 150000), col = c("dodgerblue4","dodgerblue1","snow3"),
       xlab = "Métodos machine learning y estocásticos", ylab = "Importe reserva (miles de €)",
       main = list("Reserva total de siniestros (RSP + IBNR)", cex = 1.2))
legend(x = "topleft", bty = "n",
      legend = c("Observado","Estimado","Diferencia"),
      pch = 15, cex = 0.9, col = c("dodgerblue4","dodgerblue1","snow3"))

dev.off()

#==============================================================================
#               ANÁLISIS GRÁFICO DEL COSTO ÚLTIMO DE LOS
#               SINIESTROS CERRADOS POR CADA PREDICTOR
#==============================================================================

# Datos
BD_cerr = siniestros[siniestros$Status == "Closed", ]
BD_cerr = BD[ , c("Id","AccYear","RepYear","Type","Age","AccYearMonth","RepYearMonth",
             "AccWeekday","RepDelDays","Ultimate")]

# Gráfico de densidad empírica del costo final de los siniestros (ultimate)

jpeg(filename = paste(PathPlot, "UltimateDensidad.jpeg", sep = ""),
    width = 600, height = 400)

plot(x = density(x = BD_cerr$Ultimate/1000, from = 0, to = 30),
```

```
    col  = "darkblue", lwd = 2,
    main = list("Densidad empírica del costo último individual", cex = 1.2),
    cex.lab = 1,
    ylab = "Densidd empírica", xlab = "Costo último (miles €)")

dev.off()

jpeg(filename = paste(PathPlot, "UltimateDensidad_cola.jpeg", sep = ""),
    width = 400, height = 250)
plot(x = density(x = BD_cerr$Ultimate/1000, from = 25, to = 150),
    col  = "darkblue", lwd = 2,
    main = list("Cola de la densidad empírica", cex = 1),
    cex.lab = 1,
    ylab = "Densidd empírica", xlab = "Costo último (miles €)")

dev.off()

# Gráfico de dispersión del costo final según días de retardo en el
# reporte del siniestro

jpeg(filename = paste(PathPlot, "UltimateRetReporte_disp.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
plot(formula = Ultimate/1000 ~ RepDelDays, data = filtered_data,
    xlim = c(0,1200), ylim = c(0,800), col = "darkgreen",
    main = list("Costo último individual vs Retardo en el reporte del siniestro", cex = 1.2),
    ylab = "Costo último (miles €)", xlab = "Días de retardo", cex.lab = 1.2)

dev.off()

# Diagrama de caja del costo final según meses de retardo en el reporte del siniestro

jpeg(filename = paste(PathPlot, "UltimateRetReporte_caj.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
filtered_data$RepDelMonths = filtered_data$RepDelDays / 30
filtered_data$RepDelMonths = ceiling(x = filtered_data$RepDelMonths)
col = rainbow(n = length(unique(filtered_data$RepDelMonths)), start = 0.3, end = 0.5)
boxplot(formula = log(filtered_data$Ultimate) ~ RepDelMonths, data = filtered_data,
      col = col,
      main = list("Costo último individual vs Retardo en el reporte del siniestro", cex = 1.2),
      ylab = "log del costo último", xlab = "Meses de retardo", cex.lab = 1.2)
abline(h = mean(log(filtered_data$Ultimate)), col="darkred", lwd = 1.5)

dev.off()

# Gráfico de días de retraso promedio en el reporte del siniestro versus el costo final
library(plyr)    # para función "rpund_any"
library(dplyr)
library(locfit)  # para hacer el ajuste

modified_data_ulti = BD_cerr
modified_data_ulti$Ultimate = pmin(round_any(modified_data_ulti$Ultimate, 1000, f = ceiling),
100000)
```

```
plot_data = modified_data_ulti %>%
 dplyr::group_by(Ultimate) %>%
 dplyr::summarise(mean_RepDel = mean(RepDelDays))

jpeg(filename = paste(PathPlot, "UltimateRetReporte_line.jpeg", sep = ""),
    width = 600, height = 400)

with(
 plot_data, {
  plot(x = Ultimate/1000, y = mean_RepDel, type = 'l', col = "black",
      ylab = "Días de retardo", xlab = "Costo último (miles €)", cex.lab = 1.2,
      main = list("Retardo promedio en el reporte vs Costo último individual", cex = 1.2))
  lines(x = Ultimate/1000, y = predict(locfit(mean_RepDel ~ Ultimate, alpha = .5, deg = 2),
                              newdata = Ultimate), col = "green", lwd = 2)
  legend(x = "bottomleft", cex = 1, lty = rep(1,2), lwd = c(2,2), bty = "n",
       col = c("black","green"), legend=c("Media empírica", "Media ajustada"))
 }
)

dev.off()

# Diagrama de caja del costo final por tipo de siniestro (tipo de accidente)

jpeg(filename = paste(PathPlot, "UltimateTipo_caj.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
col = rainbow(n = length(unique(filtered_data$Type)), start = 0.1, end = 0.2)
boxplot(formula = log(filtered_data$Ultimate) ~ Type, data = filtered_data,
      col = col, ylab = "log de costo último individual", xlab = "Tipo de siniestro", cex.lab = 1.2,
      main = list("Costo último vs tipo de siniestro", cex = 1.2))

dev.off()

# Diagrama de caja del costo final por edad del accidentado

jpeg(filename = paste(PathPlot, "UltimateEdad_caj.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
col = rainbow(n = length(unique(filtered_data$Age)), start = 0.1, end = 0.2)
boxplot(formula = log(filtered_data$Ultimate) ~ Age, data = filtered_data,
      col = col,
      main = list("Costo último individual vs Edad del lesionado", cex = 1.2),
      ylab = "log del costo último", xlab = "Edad", cex.lab = 1.2)
abline(h = mean(log(filtered_data$Ultimate)), col="darkred", lwd = 1.5)

dev.off()

# Diagrama de caja del costo final por edad, para cada tipo de siniestro

## Función previa para gráfico, tomada de ....
UltimatePerClaimType = function(data, type, save.yes, path) {
 col = rainbow(n = length(unique(data$Age)), start = 0.1, end = 0.2)
 if (save.yes == 1) {
  jpeg(filename = path)
 }
```

```
  filtered_data = filter(data, Type == type)
  # boxplot of log ultimate vs age, for a given claim type
  boxplot(log(Ultimate) ~ Age, data = filtered_data, col = col,
        main = paste("Costo último por tipo de siniestro ", type, sep=""),
        ylab = "log del costo último", xlab = "Edad", cex.lab = 1.2)
  # plot average log claim size as a horizontal line
  abline(h = mean(log(filtered_data$Ultimate)), col="darkred", lwd = 1.5)

  if (save.yes == 1) {
    dev.off()
  }
}

## Diagramas
for (type in c(1:6)) {
  UltimatePerClaimType(data = filter(BD_cerr, Ultimate > 0), type = type,
                save.yes = 0,
                path = paste(PathPlot, "UltimateEdadTipo_caj", type, ".jpeg", sep=""))
}


# Diagrama de caja del costo final según día del accidente

jpeg(filename = paste(PathPlot, "UltimateDiaAcc_caj.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
col = rainbow(n = length(unique(filtered_data$AccWeekday)), start = 0.1, end = 0.2)
boxplot(formula = log(filtered_data$Ultimate) ~ AccWeekday, data = filtered_data,
      col = col, xaxt = "n", # para borrar las etiquetas del eje x
      main = list("Costo último individual vs Día de ocurrencia del siniestro", cex = 1.2),
      ylab = "log del costo último", xlab = "Día de ocurrencia del siniestro", cex.lab = 1.2)
axis(side = 1, at = 1:7,
    labels = c("Lun","Mar","Mie","Jue","Vie","Sab","Dom"))
abline(h = mean(log(filtered_data$Ultimate)), col="darkred", lwd = 1.5)

dev.off()

# Diagrama de caja del costo final según mes del accidente

jpeg(filename = paste(PathPlot, "UltimateMesAcc_caj.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
col = rainbow(n = length(unique(filtered_data$AccYearMonth)), start = 0.1, end = 0.2)
boxplot(formula = log(filtered_data$Ultimate) ~ AccYearMonth, data = filtered_data,
      col = col, xaxt = "n", # para borrar las etiquetas del eje x
      main = list("Costo último individual vs Mes de ocurrencia del siniestro", cex = 1.2),
      ylab = "log del costo último", xlab = "Mes de ocurrencia del siniestro", cex.lab = 1.2)
axis(side = 1, at = 1:12,
    labels = c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic"))
abline(h = mean(log(filtered_data$Ultimate)), col="darkred", lwd = 1.5)

dev.off()

# Diagrama de caja del costo final según mes de reporte del siniestro
```

```
jpeg(filename = paste(PathPlot, "UltimateMesRep_caj.jpeg", sep = ""),
    width = 600, height = 400)

filtered_data = filter(BD_cerr, Ultimate > 0)
col = rainbow(n = length(unique(filtered_data$RepYearMonth)), start = 0.1, end = 0.2)
boxplot(formula = log(filtered_data$Ultimate) ~ RepYearMonth, data = filtered_data,
       col = col, xaxt = "n", # para borrar las etiquetas del eje x
       main = list("Costo último vs Mes de reporte del siniestro", cex = 1.2),
       ylab = "log del costo último", xlab = "Mes de reporte del siniestro", cex.lab = 1.2)
axis(side = 1, at = 1:12,
    labels = c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic"))
abline(h = mean(log(filtered_data$Ultimate)), col="darkred", lwd = 1.5)

dev.off()

#==============================================================================
```