Máster en Ciencias Actuariales y Financieras

TRABAJO FINAL DE MÁSTER

Título: Modelización de la probabilidad de no renovación para una cartera de salud a partir de Logit, Random Forest y SVM.

Autoría: Francisco Folk Delgado

Tutoría: Catalina Bolancé Losilla

Curso académico: 2021-2022

UNIVERSITAT DE BARCELONA
Facultat d'Economia i Empresa

Màster de Ciències Actuarials i Financeres

Facultad de Economía y Empresa

Universidad de Barcelona

Trabajo Final de Máster

Máster en Ciencias Actuariales y Financieras

Modelización de la probabilidad de no renovación para una cartera de salud a partir de Logit, Random Forest y SVM.

Autoría: Francisco Folk Delgado

Tutoría: Catalina Bolancé Losilla

“El contenido de este documento es de exclusiva responsabilidad del autor, quien declara que no ha incurrido en plagio y que la totalidad de referencias a otros autores han sido expresadas en el texto”.

“The content of this document is the sole responsibility of the author, who declares that he/she has not incurred plagiarism and that all references to other authors have been expressed in the text”.

Modelización de la probabilidad de no renovación para una cartera de salud a partir de Logit, Random Forest y SVM.

Francisco Folk Delgado
Tutor: Catalina Bolancé Losillla
Universidad de Barcelona, diciembre 2021

Resumen

Actualmente, el análisis de los datos se ha convertido en una herramienta crucial para el desarrollo de las empresas. La finalidad de este trabajo es modelizar una base de datos real a partir de la regresión logística y los algoritmos de clasificación Random Forest y Support Vector Machine con el objetivo de predecir la probabilidad de no renovación de una póliza. De esta manera, se podrá desarrollar un proyecto de estimación desde la base, comparar de forma empírica los resultados obtenidos bajo las distintas propuestas y analizar la calidad de los resultados obtenidos.

Palabras clave: Predicción, no renovación, Logit, Random Forest y SVM.

Abstract

Nowadays, data analysis has become a critical tool for business development. The aim of this study is to model a real database based on the logistic regression and the classification algorithms Random Forest and Support Vector Machine (SVM) in order to predict the probability of policy churn. As a result, a bottom-up estimation will be developed, compare the results obtained are empirically compared under the different proposal models and to analyze the quality of the results obtained.

Key words: Prediction, churn, Logit, Random Forest y SVM.

ÍNDICE

Sección 1

1. Introducción

Hoy en día, el tratamiento y el análisis de los datos se ha convertido en una herramienta crucial para el desarrollo de las empresas. A partir de la ciencia de datos, se puede transformar esta materia prima en nueva información de gran utilidad sobre los clientes, el sector, los productos, etc. En el sector asegurador, el cálculo y las estimaciones relacionados con siniestros y primas forma parte de la estructura y del buen funcionamiento de la empresa. Aun así, también cabe analizar la sección comercial que, al fin y al cabo, es la que reporta los ingresos principales a las aseguradoras.

Actualmente, la sociedad sigue lastrada por los efectos de la pandemia del COVID-19, los cuales han provocado gran incertidumbre sobre el futuro en todos los ámbitos, des del punto de vista personal, laboral y empresarial. Entre otros, uno de los temas que concierne al sector es el papel que tendrán las aseguradoras y su estado económico y de mercado una vez superada esta crisis. Para ello, una de las herramientas que pueden utilizar las empresas es el *Data Science* con el objetivo de obtener el mayor grado de información sobre los factores que repercuten directamente en la empresa. En este caso, enfocado más a un apartado comercial, es interesante que la empresa tenga conocimientos sobre sus clientes para poder predecir su comportamiento, cubrir sus necesidades y gestionar a los clientes de forma óptima. Este análisis puede comportar a la empresa una mejora de la calidad de su cartera, un afianzamiento de los clientes, gracias a la fidelidad, y un abaratamiento de los costes de adquisición, ya que el reclutamiento de nuevos clientes es siempre más costoso que el mantenimiento de los propios (Rodríguez, 2000).

Por ello, es interesante realizar una serie de análisis sobre la cartera de la empresa, a partir de modelos estadísticos como puede ser un *Costumer Lifetime Value*, con el que valorar la aportación de beneficio del cliente durante su vigencia, un *Cross Selling*, con el que poder anticipar las necesidades potenciales del cliente o un *Costumer Churn Model* o modelo de probabilidad de no renovación, con el que prever la pérdida de una póliza y poder realizar las acciones comerciales que la empresa crea necesarias. En este último se centra el presente trabajo.

A partir de una base de datos real sobre una cartera de salud para el año 2020, se intentará ajustar la probabilidad de no renovación de una póliza a partir del modelo de regresión logística (Logit) y los algoritmos *Random Forest* y *Support Vector Machine* (SVM), con el objetivo de que dicho modelo sea capaz de predecir el comportamiento futuro de los clientes. Para el cálculo de los distintos pasos de transformación de la base de datos y del cálculo de los modelos, así como la extracción de los datos de las tablas y las gráficas se ha utilizado el lenguaje de programación R.

Sección 2

2. Marco teórico

Para el desarrollo del proyecto de predicción de la probabilidad de no renovación de las pólizas se han utilizado tres modelos estadísticos o algoritmos de clasificación: Logit, Random Forest y SVM.

2.1. Logit

El Logit hace referencia al modelo de regresión logística que permite estimar la probabilidad de ocurrencia de un suceso en función de un conjunto de variables explicativas que pueden ser de naturaleza cualitativa y cuantitativa. La probabilidad se aproxima a partir de una transformación del predictor lineal. El caso más simple es que esta transformación sea igual a una identidad, lo que da lugar el modelo de probabilidad lineal. El modelo de regresión lineal es una de las herramientas clásicas del campo de la estadística y el análisis de datos. Se trata de un modelo que mide la relación entre una o más variables independientes frente a la dependiente a través de la siguiente función:

$$Y = \alpha + \beta * X + \varepsilon$$

Dónde Y representa la variable dependiente y X las variables independiente. En cuanto a las letras griegas, α representa el intercepto, el cual representa el valor de la variable dependiente cuando la variable independiente es 0. El parámetro β la relación lineal entre X e Y, midiendo la cantidad en la que variará Y en caso del aumento de X en una unidad. Por último, ε es el residuo aleatorio del modelo.

El modelo de regresión lineal simple es aquel en el que solo existe una variable independiente. Para asegurar el buen desarrollo del modelo se deben cumplir las siguientes hipótesis.

- Linealidad.
- Distribución normal de los residuos.
- Homocedasticidad.
- Revisión de *outliers*.
- Independencia de las observaciones.

En el caso de añadir un mayor número de variables independientes al modelo, se trataría de un modelo de regresión múltiple y la función sería la siguiente.

$$Y = \beta_0 + \beta_1 * X_1 + \ldots + \beta_i * X_i + \varepsilon$$

El intercepto se suele representar como β_0, aunque su definición e interpretación es la misma que para el modelo simple. Por otro lado, se añadirán las variables independientes a la función junto a su parámetro correspondiente. El método de estimación empleado de los parámetros es el de los Mínimos Cuadrados Ordinarios (MCO). Además de las hipótesis anteriores, el modelo de regresión múltiple debe hacer frente a otras condiciones, como la no colinealidad entre las variables predictoras, la relación lineal entre predictores y la variable independiente y el principio de parsimonia.

La principal dificultad del modelo lineal es que no está acotado entre 0 y 1 y que no cumple las hipótesis necesarias, como la de no homocedasticidad o la normalidad de los residuos.

Como se comentaba al principio del punto, el Logit es una variante del modelo de regresión lineal en el que, los valores siempre están acotados entre 0 y 1 gracias a una transformación del predictor lineal a partir de la función sigmoide. Para el Logit, al tratarse de un modelo no lineal, el método empleado para el cálculo de los parámetros es el Máxima Verosimilitud.

$$Función\ sigmoide = \frac{1}{1 + e^{-x}}$$

Si se substituye el valor de x por la función anterior y, posteriormente, se modifica la función y se obtiene la siguiente.

$$P(Y = 1) = \frac{e^{\beta_0 + \beta_1 * X_1 + \ldots + \beta_i * X_i}}{1 + e^{\beta_0 + \beta_1 * X_1 + \ldots + \beta_i * X_i}}$$

Con esta función se obtienen los valores de probabilidad de Y a partir de los valores de las variables y sus estimadores. Si se aplican logaritmos se obtiene la función definitiva.

$$\ln\left(\frac{P(Y = 1)}{1 - P(Y = 1)}\right) = \beta_0 + \beta_1 * X_1 + \ldots + \beta_i * X_i$$

Esta última función se ajusta al modelo de regresión múltiple a cambio de una transformación en la respuesta, originando la función del Logit o del Logaritmo de *odds*. Los *odds* hacen referencia a las ecuación $\frac{P(Y=1)}{1-P(Y=1)}$, la cual indica el número de eventos verdaderos esperados por cada evento falso. A diferencia del modelo de regresión, la interpretación de los coeficientes no es tan directa. El aumento de una unidad de la variable Xi, implica el aumento de β1 el logaritmo de *odds*.

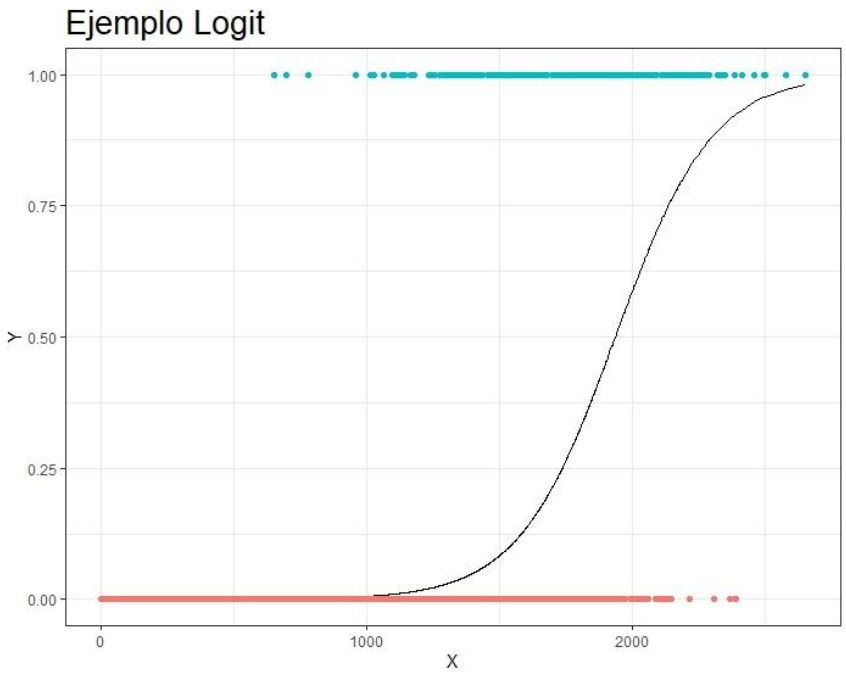


Ilustración 1: Ejemplo Logit (Elaboración propia).

2.2. Random Forest

El algoritmo *Random Forest* es un modelo estadístico de *ensemble* utilizado para la predicción y clasificación desarrollado por Leo Breiman en 2001. Este algoritmo es la combinación de los árboles de decisión y del algoritmo *Bagging*. El *Random Forest* funciona realizando un conjunto determinado de árboles de decisión, utilizando de forma aleatoria parte de los datos y variables introducidas en el modelo hasta llegar a un nodo terminal. Finalmente, se hace una media de todos los árboles predichos y se genera una predicción final.

El árbol de predicción es un modelo predictivo en el que los datos se van dividiendo de forma binaria según los valores de las variables independientes (Breiman, 1984). Estas divisiones se generan a partir de distintos nodos hasta llegar a un punto indivisible o especificado anteriormente. A consecuencia de las condiciones generadas por los nodos, se obtienen una valor numérico o categórico, dependiendo del objetivo y de la variable a predecir, formando un árbol de regresión para el primer caso o de clasificación en el segundo. Esta técnica estadística recibe el nombre de árbol ya que, si se representa gráficamente, los nodos que surgen se obtiene una imagen similar a la de un árbol. El problema principal de los árboles de decisión utilizados para predecir es el sesgo estadístico, ya que el método suele ajustarse notablemente a la muestra de entrenamiento haciendo que se produzcan errores considerables al aplicar el modelo a una muestra de test. En la ilustración, se muestra un ejemplo de árbol de regresión.

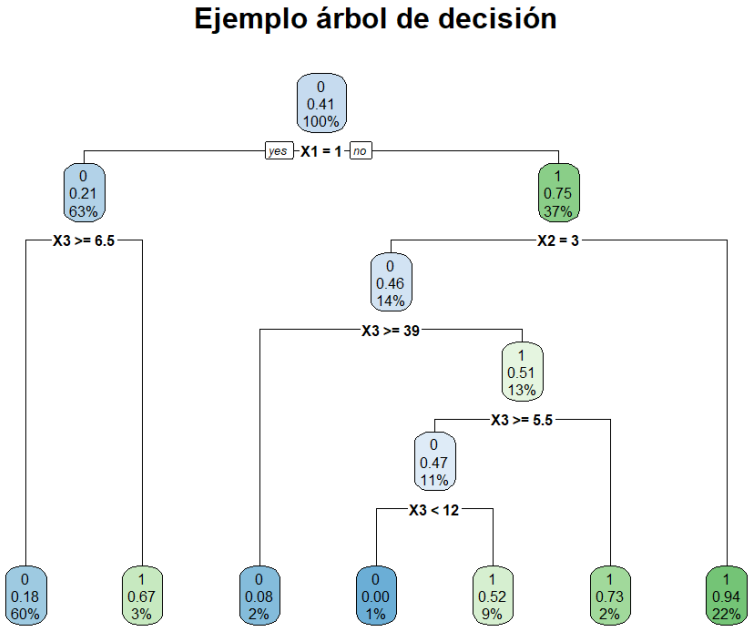


Ilustración 2:Ejemplo de árbol de regresión (Elaboración propia).

En base a la técnica de los árboles de predicción surgieron nuevos métodos y algoritmos, como el *Bagging*, el predecesor del *Random Forest*. La palabra *Bagging* es un acrónimo de *bootstrap aggregation*, ya que el modelo está formado a partir de la combinación del método de muestreo *Bootstrapping*. El método *Bagging* genera una partición aleatoria de los datos, a partir de los datos de la muestra original, y los utiliza para generar el modelo (Breiman, 1996). A diferencia del *bootsrapping*, este método utiliza el reemplazo de datos, es decir, las observaciones utilizadas en una muestra pueden volver a aparecer en la siguiente. De esta manera, cada modelo estará basado en una muestra distinta, proveniente de la original, con el objetivo de reducir la varianza del modelo.

Finalmente, con la combinación del modelo de árboles de decisión y el método *Bagging*, se obtiene el Random Forest, el cual es capaz de reducir las debilidades de las técnicas que lo componen. Como se ha comentado anteriormente, el modelo de árboles de decisión conlleva un problema de bias si se utiliza para la predicción, es decir, los valores obtenidos para los datos de predicción suelen tener errores significativos respecto del valor real. Además, también conlleva un problema con la varianza ligado a la debilidad anterior, ya que se producen cambios considerables dependiendo de los datos que se utilicen para el cálculo. De esta manera, al aplicar el algoritmo *Bagging*, utilizando de forma aleatoria las observaciones de la muestra original a través de *resampling* de los datos, y a partir de los parámetros del propio método *Random Forest* se puede obtener un modelo que encuentre un equilibrio entre el bias y la varianza.

Una vez claro el origen del método, el siguiente paso es conocer la estructura y el funcionamiento del propio algoritmo. Los parámetros principales utilizados para ajustar el modelo son el número de árboles utilizado en el cómputo global del modelo, el número de predictores utilizados en cada árbol y el tamaño mínimo de los nodos terminales. La adecuación de los parámetros es clave para generar un modelo sólido y capaz de predecir o clasificar los datos de forma idónea. Por contraste, un ajuste demasiado preciso puede conllevar caer en *overfitting* y, de la misma manera, el aumento de los parámetros puede comportar un incremento del coste de computación, de forma que suponga una menor calidad del modelo, alejándolo del óptimo. Para realizar este cálculo, se realizará varias pruebas mediante un *loop* en el cual se registrará el error de cada una de las variables. De esta manera, se escogerá el valor para cada parámetro que haya obtenido un error más bajo.

2.3. SVM

El *Support Vector Machine* es un algoritmo de *machine Learning*, desarrollado en la El *Support Vector Machine* es un algoritmo de *machine Learning*, desarrollado en la década de los 90 (Boser et al., 1992, Vapnik, 1999, Cristianini and Shawe-Taylor, 2000) basado en el algoritmo del *Perceptrón* (Rosenblatt, 1958). Estos algoritmos se basan en una misma idea, la de encontrar una función capaz de dividir correctamente dos grupos de una misma base de datos. Esta función es representada por un hiperplano, el cual se entiende como un subespacio de n dimensiones dentro de un espacio de n +1 dimensiones. Así pues, estos algoritmos son una buena alternativa para clasificaciones binarias.

El objetivo de Rosenblatt era encontrar cómo se transmite y se gestiona la información a través del sistema nervioso (Rosenblatt, 1958). El algoritmo *Perceptrón* se basa en la creación de una recta a través de productos escalares. Con tal de minimizar el error, Rosenblatt utilizó un sistema de pesos para ajustar el Perceptrón a los datos. El problema principal de este sistema es que requiere que los grupos pertenecientes a la base de datos sean linealmente separables, es decir, que exista al menos un hiperplano capaz de dividir a la perfección dichos grupos. Además, la única manera de mejorar la clasificación del *Perceptrón* es a partir de la modificación del vector de pesos, lo que implica que el hiperplano será distinto para cada cambio. Esto implica que el hiperplano calculado sea distinto cada vez. Y, aunque se obtengan buenos resultados, puede que uno de los hiperplanos calculados esté muy cerca de uno de los grupos y, a la hora de generalizar, clasifique erróneamente los nuevos datos (Kowalczyk, 2017).

Para reducir el error del *Perceptrón* se busca el hiperplano óptimo (Vapnik, 1982), también conocido como *Maximal Margin Hyperplane*. Esta idea consiste en encontrar el hiperplano con mayor distancia entre los puntos más cercanos de cada subgrupo, es decir, aquel que mantenga la separación más amplia entre grupos. El algoritmo sigue teniendo problemas de rigidez en el modelo ya que, el añadir nuevos datos implica el recálculo del hiperplano para encontrar el óptimo. Además, está mejora en la optimización del hiperplano hace que el modelo caiga en *overfitting*, haciendo que sea poco útil para la generalización. Como comentaba anteriormente, estos algoritmos se basan en la hipótesis de que los datos son linealmente separables, pero, en la realidad, pocos datos la cumplen.

Por lo tanto, en los casos en los que los datos no son linealmente separables se utilizan los *Soft Margin Hyperplane*. Esta modificación del algoritmo consiste en añadir un parámetro constante **C**, el cual implica el coste de la violación de las restricciones. De esta manera, el algoritmo gana flexibilidad y mejora la generalización de los datos en detrimento de la precisión de la clasificación. **C** es una constante que puede recibir valores entre 0 e ∞, siendo los valores más cercanos a 0 los que aportan mayor margen y menor penalización a los errores y, a medida que se aumenta el valor de la constante, se aumenta la severidad de las restricciones.

Hasta el momento, se ha supuesto que la relación entre las variables es lineal, limitando el alcance del algoritmo, pero la mayoría de problemas reales no siguen una relación estrictamente lineal. De esta idea surge la aplicación de los *kernels*. Un *kernel* se define como una función capaz de transformar el producto vectorial para llevar a los datos a una nueva dimensión, en la cual los datos no deben ser separables linealmente. Este método añade complejidad a los cálculos y permite un enfoque más eficiente, transformando una espacio de dos dimensiones en uno de n, a partir de una función especificada previamente. Los *kernels* más utilizados son los siguientes: Lineal, Polinómico y Radial o Gaussiano.

El kernel lineal es el más simple de todo y no implica una transformación en sí. Sería el equivalente a los vectores de clasificación vistos anteriormente. Este kernel se calcula a partir de la siguiente función:

$$K(x, x') = x \cdot x'$$

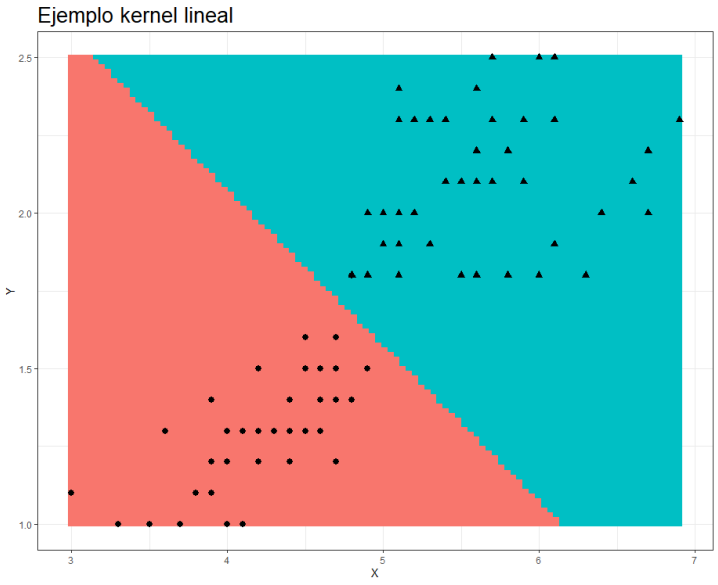


Ilustración 3: Ejemplo gráfico de SVM lineal (Elaboración propia).

El kernel polinómico transforma el producto vectorial en una función polinómica de grado d, siendo este el segundo parámetro necesario para la conversión polinómica y tiene la siguiente forma:

$$K(x, x') = (x \cdot x' + c)^d$$

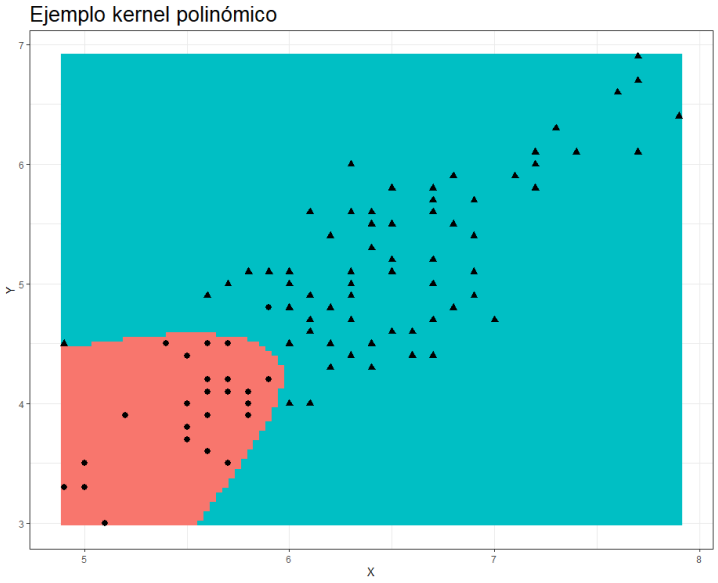


Ilustración 4: Ejemplo gráfico de SVM polinómico (Elaboración propia).

El kernel Radial o Gaussiano es el más sofisticado de la terna, ya que es capaz de ajustarse a los datos con una función radial del estilo:

$$K(x, x') = e^{\frac{-|x-y|^2}{2\sigma^2}}$$

Tras una simple transformación, se simplifica esta función obteniendo uno de los parámetros necesarios para ajustar este tipo de kernel. Se substituye $\frac{1}{2\sigma^2}$ por γ, obteniendo la siguiente función:

$$K(x, x') = e^{-\gamma|x-y|^2}$$

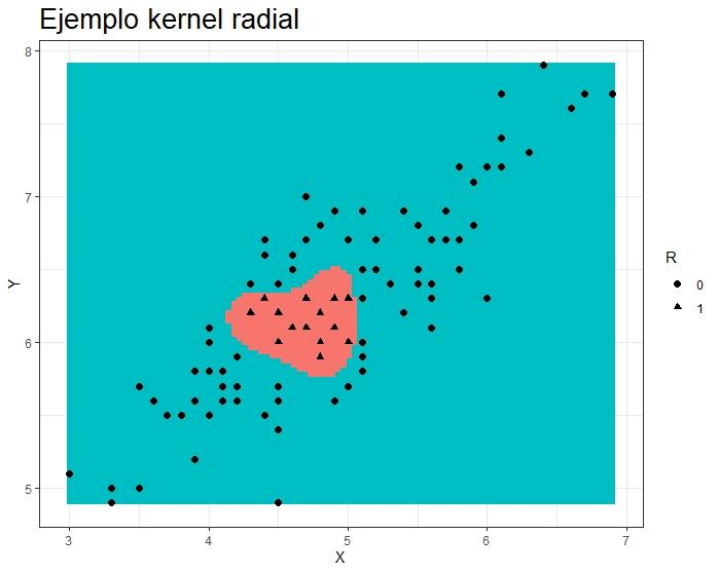


Ilustración 5: Ejemplo gráfico de SVM radial (Elaboración propia).

Sección 3

3. Análisis de los datos

Para el estudio se cuenta con un total de dieciséis variables y la variable objetivo. Los datos utilizados se han extraído de una base de datos real del año 2020, gracias a la colaboración de la empresa FIATC Seguros. La extracción, cálculo y manipulación inicial de los datos, ha sido elaborada específicamente para este proyecto, pero no se mostrarán en este documento, para conservar la privacidad de los mismos. En la siguiente tabla se muestran dichas variables junto a su origen, pudiendo ser una variables sobre el tomador, sobre la póliza o variables económicas.

Tipo	Variables
Variable objetivo	Estado (indica si la póliza está o no activa).
Tomador	Bajas anteriores distintas de salud, Bajas anteriores ramo de salud, Número de pólizas vigentes distintas de salud
Póliza	Antigüedad póliza, Autorizaciones, Código de pago, Edad media de la póliza, Familia, Número de asegurados vigentes, Uso
Económica	Prima actual, Prima anterior, Siniestralidad, Variación prima, Variación prima porcentual

Tabla 1: Variables iniciales.

El tratamiento de los datos es un paso clave dentro de cualquier proceso estadístico, ya que de éstos dependerá el resultado final del análisis y su calidad. Durante el proceso de extracción de la base de datos se han obtenido varias observaciones no disponibles (NA) y otras que no son coherentes con la realidad. Dada la extensión total de los datos y la poca aparición de este tipo de observaciones, se ha optado por eliminarlas.

3.1. Datos no balanceados

Uno de los problemas principales de este trabajo es que se trata de una base de datos no balanceados, es decir, que existe una clase mayoritaria que domina frente a la minoritaria. Previo al tratamiento de los datos, la base de datos está constituida por 60.631 observaciones, de las cuales, sólo 2.551 se corresponden con la clase minoritaria, en este caso, pólizas que se dieron de baja. Por lo tanto, la clase a predecir corresponde únicamente al 4.21% de las observaciones. Este hecho implica grandes dificultades a la hora de entrenar los modelos, ya que, al haber una clase mayoritaria tan dominante, los modelos de clasificación se ven fuertemente influenciados por estos, clasificando de forma destacable esta clase, pero, etiquetando de forma errónea a la clase minoritaria.

Para compensar este problema se va a utilizar un sistema de pesos que las propias funciones de R tienen incorporadas. De esta manera, dándole mayor importancia a la clase minoritaria, se espera encontrar un modelo que se ajuste a los datos.

3.2. Colinealidad entre variables

La colinealidad o correlación entre las variables es uno de las características principales a estudiar y, además, es una de las condiciones principales para la correcta aplicación del modelo de regresión logística. El fenómeno de la multicolinealidad, puede provocar errores en la estimación de coeficientes y en su significancia (Wang and Akabay, 1994). Existen gran cantidad de técnicas para determinar la colinealidad entre variables, en este caso, se utilizarán los coeficientes de correlación y el factor de incremento de la varianza (FIV) para las variables numéricas. Para el primer caso, se han obtenido las correlaciones mediante el coeficiente de correlación de Pearson, las cuales se muestran en la Tabla 18 de Anexos. Destacan las correlaciones entre las variables Prima actual y Prima anterior, Uso y Siniestralidad y la Variación de la prima con la Variación Porcentual. Este hecho era de esperar, las primeras dependen del mismo criterio para su cálculo ya que dependen de la misma tabla de tarifas y del crecimiento anual o el número de usos influye directamente en el cálculo de la siniestralidad, mientras que la última es una transformación de la misma información. Puesto a que tienen un valor superior o igual a 0,8, indican que tienen una correlación muy fuerte y que podría comportar problemas de colinealidad. Aun así, la ausencia de correlación no implica que no exista colinealidad (Mason and Perrault, 1991). Por lo tanto, otra método que contribuya a localizar la colinealidad entre variables es el FIV. Este se calcula a partir del coeficiente de determinación, el cual proviene de elevar al cuadrado el coeficiente de correlación (R), a partir de la siguiente fórmula.

$$FIV = \frac{1}{1 - R^2}$$

Un valor del FIV entre 5 y 10, muestra una dependencia ligera, mientras que, los valores superiores a 10 indican una dependencia fuerte entre las variables. De la misma manera que en el estudio anterior, las parejas de variables que destacan son las mismas, obteniendo un FIV de 11,7 para las primas y de 4 para las variaciones. Pese a que este último no entra dentro de los parámetros para este segundo estudio, se eliminará una de las variables como en el caso anterior. En cambio, a partir del FIV se observa que la dependencia entra las variables Uso y Siniestralidad no es relevante. Seguramente es debido a que los usos no son la única variable que influye en la siniestralidad, ya que también participan las autorizaciones y que el valor de cada uso o autorización no es el mismo. Los resultados del FIV se encuentran en la Tabla 19 del Anexo.

A partir del método de la Entropía, se determina que los variables Prima actual y Variación porcentual, aportan mayor información al modelo que sus respectivas análogas, por lo tanto, se mantendrán en el modelo en detrimento de estas. Mientras que, las variables Uso y Siniestralidad se mantendrán a la espera de realizar más análisis.

3.3. Eliminación de outliers

Otro de los problemas es el gran número de *outliers* encontrado en los datos. Los valores extremos pueden desvirtuar considerablemente el análisis de los datos y, por ende, los resultados obtenidos. Con tal de solucionar este problema y obtener un modelo más robusto, se ha optado por aplicar el Método de Tukey (Tukey, 1977). De forma simple, esté método utiliza el primer y el tercer cuartil de los datos para determinar que valores son extremos. El valor se obtiene a partir del cálculo del rango intercuantil (RIC), es decir, la diferencia entre el tercer y el primer cuartil, multiplicado por 1,5. Una vez calculado, se obtienen el límite superior e inferior. En el primer caso, se sumaría el tercer cuartil con el resultado anterior mientras que, para el segundo, se restaría al primer cuartil el valor obtenido en el primer cálculo. En caso de querer suavizar este método de limpieza de datos y eliminar solo los valores muy extremos, se podría multiplicar el rango intercuantil por 3.

$$RIC = Q_3 - Q_1$$

$$Límite\ Superior = Q_3 + 1.5 * RIC$$

$$Límite\ Inferior = Q_1 - 1.5 * RIC$$

A modo de ejemplo, en la Ilustración 1, se muestra un boxplot con los datos para la variable de la prima actual. El boxplot se caracteriza por ser un gráfico en el que se presentan los datos en tres capas distintas siguiendo los cálculos del Método de Tukey. Primero, se observa una caja en la que aparecen los datos entre el primer y el tercer cuartil, en este caso, de color verde. Seguidamente se muestran los datos del límite inferior y del límite superior con una línea de color negro. Por último, se señalan a partir de puntos los valores outliers, marcados en rojo. Así pues, queda representado como, para este variable hay valores por encima del límite superior que se alejan mucho de los valores de la muestra y que pueden distorsionar los resultados. A partir de la aplicación del Método de Tukey, los valores extremos marcados en rojo quedarán eliminados de la muestra.

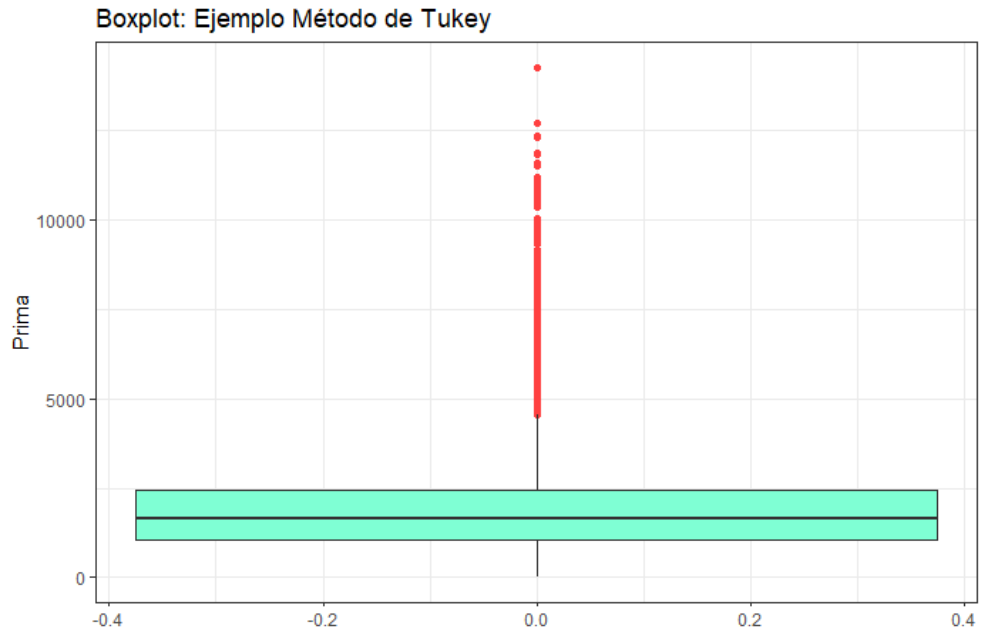


Ilustración 6:Ejemplo de valores extremos mediante un boxplot. (Elaboración propia).

Este método de exclusión de los *outliers* se ha utilizado para algunas de las variables numéricas, como la antigüedad de la póliza, el número de asegurados, el uso, las autorizaciones, la siniestralidad y las primas, tanto actuales como del año anterior. En cambio, para las variables de bajas anteriores en el ramo de salud, bajas anteriores en otros ramos y para el número de pólizas vigentes en otros ramos, no se ha aplicado porque, aunque haya valores extremos, la gran mayoría de valores se encuentran en 0. Por ello, y a partir de la valoración de las variables mediante la entropía, se ha decidido convertirlas en variables dicotómicas, en el que se le otorga el valor 0 en los casos en los que el valor es 0 y 1 para el resto, tal y como se observa en el siguiente apartado.

3.4. Ganancia de información

Una vez finalizado el proceso de limpieza de valores atípicos, queda una base de datos con 43.672 observaciones. De estas, solo 2.007 representan bajas y las 41.665 restantes representan las pólizas que se mantienen vigentes, es decir, las pólizas de baja de la base de datos se concentran en el 4,60% del total de las observaciones. Hay algunas variables numéricas que consideramos reconvertirlas a factor, en este caso, en una variable dicotómica, por los valores de la propia variable. Seguidamente, a partir de la siguiente técnica, se valorará si este cambio es necesario o si aporta mejoras a la importancia que tienen dentro del modelo.

El método utilizado para valorar la importancia de las variables es la teoría de la ganancia de información (Shannon, 1949). La entropía es un concepto de orden de una variable acotada entre 0 y 1, dónde 0 se refiere al orden total, es decir, que todas las observaciones de la propia variable se corresponden con la misma clase o valor, y 1 referido al desorden total, dónde las observaciones se encuentran equitativamente distribuidas. El cálculo de la entropía se realiza mediante la siguiente fórmula:

$$Entropia = -\sum p_i - log_2\,(p_i)$$

Dónde, pi se corresponde a la probabilidad de ocurrencia de la clase i para la variable X. Está función no es lineal, por lo que no sé puede extraer ninguna conclusión certera de los valores intermedios más que, los valores cercanos a los límites indican que las observaciones se distribuyen de forma cercana al orden o desorden total. En este caso, al ser la variable objetivo corresponde a unos datos no balanceados, es decir, hay una clase mayoritaria que domina por encima de la minoritaria de forma evidente, el valor de la entropía es cercano a 0. Tal y como se observa en la Tabla 2, la entropía correspondiente al estado de la póliza es de 0.269. Siguiendo la teoría de la Ganancia de información, se procede a calcular la entropía del resto de variables condicionado a la variable objetivo, mostrando si el comportamiento de éstas sigue una tendencia similar y si, por lo tanto, las variables explicativas están relacionadas con esta.

$$Ganancia\ de\ Información = Entropia_y - Entropia(X|Y)$$

De esta manera, si la entropía condicional de las variables independientes es cercana a 0 implica que hay una parte notable de información que explica ésta sobre la variable dependiente. En cuanto a la ganancia de información, cuanto más cercana sea ésta a la entropía de la variable objetivo mayor parte de información aportará la variable explicativa sobre esta, tal y como se observa en la siguiente tabla.

Variables	Entropía	Ganancia de información
Estado	0,269	-
Variación porcentual	0,004	0,265
Prima actual	0,017	0,252
Número de pólizas vigentes distintas de salud (Dicotómica)	0,025	0,244
Variación de prima	0,026	0,243
Prima año anterior	0,028	0,241
Bajas anteriores del ramo de salud (Dicotómica)	0,030	0,239
Siniestralidad	0,047	0,222
Bajas anteriores distintas de salud (Dicotómica)	0,050	0,219
Código de pago	0,145	0,124
Antigüedad de la póliza	0,173	0,096
Uso	0,209	0,061
Tipo de familia	0,212	0,057
Número de asegurados vigentes	0,245	0,024
Autorizaciones	0,249	0,020
Número de pólizas vigentes distintas de salud	0,259	0,010
Bajas anteriores del ramo de salud	0,263	0,006
Bajas anteriores distintas de salud	0,268	0,001

Tabla 2:Entropía de las variables.

La Tabla 2 muestra la ganancia de información que aporta cada una de las variables explicativas a la variable objetivo. Se observa que las variables económicas son las que mayor ganancia de aportación aportan a la variable objetivo, junto a la variable que indica si el tomador tiene alguna póliza vigente y si el tomador ha tenido bajas anteriores. También se concluye que la transformación comentada de las variables: Número de pólizas vigentes y de Bajas anteriores del ramo de salud y del resto de ramos, mejoran significativamente, pasando de no aportar prácticamente nada de información a estar bastante relacionadas con la variable dependiente.

En la Tabla 3 se encuentra un resumen de los datos finales tras las transformaciones ejecutadas en este último apartado.

Estado		Antigüedad de la póliza		Autorizaciones		Bajas anteriores del ramo de salud		Bajas anteriores distintas de salud		Código de pago		Edad media de la póliza	
0	41.665	Min	1	Min	0	Con bajas	8.588	Con bajas	4.383	Mensual	38.291	Min	18
1	2.007	1st Q	4	1st Q	0	Sin bajas	35.084	Sin bajas	39.289	Trimestral	2.358	1st Q	33
		Median	8	Median	0					Semestral	858	Median	46
		Mean	12,1	Mean	0,72					Anual	2.165	Mean	47,54
		3rd Q	20	3rd Q	1							3rd Q	60
		Max	34	Max	7							Max	100

Número de asegurados vigentes		Número de pólizas vigentes distintas de salud		Prima actual		Siniestralidad		Uso		Tipo de familia		Variación porcentual	
Min	1	Con pólizas	5.229	Min	26,08	Min	0,00	Min	0	Desconocido	18.527	Min	-0,9928
1st Q	1	Sin pólizas	38.443	1st Q	957,36	1st Q	227,20	1st Q	5	Con hijos mayores	7.131	1st Q	0,0150
Median	2			Median	1.432,14	Median	630,20	Median	13	Con hijos menores	11.604	Median	0,0152
Mean	1,87			Mean	1.592,54	Mean	845,00	Mean	17,2	Otros	778	Mean	0,0400
3rd Q	2			3rd Q	2.099,43	3rd Q	1.263,80	3rd Q	25	Pareja sin hijos	5.632	3rd Q	0,0561
Max	5			Max	4.019,40	Max	3.403,60	Max	77			Max	0,7722

Tabla 3: Resumen de las variables finales

Sección 4

4. Marco Práctico

4.1. Análisis de resultados

Para evaluar la capacidad predictiva de los modelos, se obtiene una matriz de confusión, en la cual se recogen de forma sencilla los resultados reales y los predichos. A partir de esta matriz se pueden obtener métricas importantes para el análisis del modelo. Como se observa en la tabla siguiente, los valores reales se sitúan verticalmente y los predichos de forma horizontal y al tener únicamente dos clases a predecir, queda una tabla con cuatro elementos.

Matriz de confusión		
	Real	
Predicción	0	1
0	a	b
1	c	d

Tabla 4: Ejemplo matriz de confusión.

- a indica los valores positivos reales (0) que han sido clasificados como positivos.
- b indica los valores negativos reales (1) que han sido clasificados como positivos.
- c indica los valores positivos reales que han sido clasificados como negativos.
- b indica los valores negativos reales que han sido clasificados como negativos.

Accuracy (Exactitud): Calcula el porcentaje de acierto global del modelo, sumando los verdaderos positivos y los verdaderos negativos y dividiéndolos entre el total de la muestra. En este caso, al ser una modelo de datos no balanceados, la exactitud no es una métrica tan importante, ya que, requiere que los datos empleados sean simétricos en cuanto a la distribución de las clases a predecir. Como se ha comentado anteriormente, en este tipo de modelos la clase positiva, en este caso mayoritaria, es tan dominante que una muy buena predicción de estos nos daría un *Accuracy* notable, aunque la clase minoritaria no esté bien clasificada.

$$Accuracy = \frac{a + d}{a + b + c + d}$$

Sensibilidad: Es una métrica que indica la cantidad de positivos clasificados correctamente, de la misma manera que para el exactitud, en este tipo de modelos es una métrica que suele dar buenos resultados gracias a que se centra en la clase mayoritaria.

$$Sensibilidad = \frac{a}{a + c}$$

Especificidad: Es una métrica que complementaria a la anterior, en la que se mide la tasa de verdaderos negativos. Es una de las métricas más importantes para el análisis de datos no balanceados, ya que es uno de los indicadores para saber si la clase minoritaria está siendo bien clasificada.

$$Especificidad = \frac{d}{b + d}$$

Verdaderos positivos predichos: Es una métrica que mide, dentro de los positivos reales, cuantos se han clasificado correctamente.

$$Verdaderos\ positivos\ predichos = \frac{a}{a + b}$$

Verdaderos negativos predichos: Es una métrica que mide, dentro de los negativos reales, cuantos se han clasificado correctamente. Esta métrica vuelve a ser relevante, ya que vuelve a poner el foco en la predicción de la clase minoritaria, esta vez, comparando los verdaderos negativos con los falsos negativos.

$$Verdaderos\ positivos\ predichos = \frac{d}{c + d}$$

Kappa: Mide la concordancia de los datos y el sesgo de los observadores teniendo en cuenta los valores esperados y los predichos, es decir, la concordancia observada. Esta métrica debe ser utilizada para dos observadores y es utilizado habitualmente cuando la variable a estimar es categórica (renovación o no renovación). El cálculo de ésta es más complejo, por lo que se dividirá en varias partes.

El primer paso es calcular el porcentaje de aciertos predichos, en este caso, se cuenta con la métrica Accuracy. Seguidamente se calcula la probabilidad de que el resultado se deba al azar a partir de calcular las probabilidades de acierto o error para los valores reales y para los valores predichos.

$$P(Predichos|0) = \frac{a+b}{a+b+c+d}$$

$$P(Predichos|1) = \frac{c+d}{a+b+c+d}$$

$$P(Reales|0) = \frac{a+c}{a+b+c+d}$$

$$P(Reales|1) = \frac{b+d}{a+b+c+d}$$

De esta manera se obtiene la probabilidad que otorga cada estado (Renovación o no renovación) condicionando a los valores reales y a los predichos.

A partir de aquí se calcula la probabilidad de concordancia esperada mediante la suma de las probabilidades condicionadas al estado, tal y como se observa en la siguiente fórmula.

$$P(Esperada) = \big(P(Predichos|0) * P(Reales|0)\big) + \big(P(Predichos|1) * P(Reales|1)\big)$$

Finalmente, se aplican estas probabilidades a la fórmula del cálculo de Kappa.

$$Kappa = \frac{P(Observada) - P(Esperada)}{1 - P(Esperada)}$$

Para todas la métricas, la valoración de los resultados se obtiene de la siguiente manera. Un valor inferior a 0.20 es bajo, entre 0.21 y 0.40 es débil, entre 0.41 y 0.60 es moderado, entre 0.61 y 0.8 es bueno y entre 0.81 y 1 es muy bueno.

4.2. *Cross Validation*

Para la validación del modelo, se utiliza el método de *Cross-Validation*. Este consiste en hacer k particiones de la base de datos y utilizando k-1 de estas como entrenamiento del modelo y la restante como test. Este método se aplica utilizando todas las particiones de manera que todas hayan sido parte del test y calculadas con el resto. De esta manera, se consigue una reducción tanto en el sesgo como en la varianza, mejorando así la predicción y reduciendo el error. El problema principal de este método es que incrementa el riesgo de caer en *overfitting*. En la siguiente imagen se observa un diagrama en el que se explica el método de *Cross Validation.*

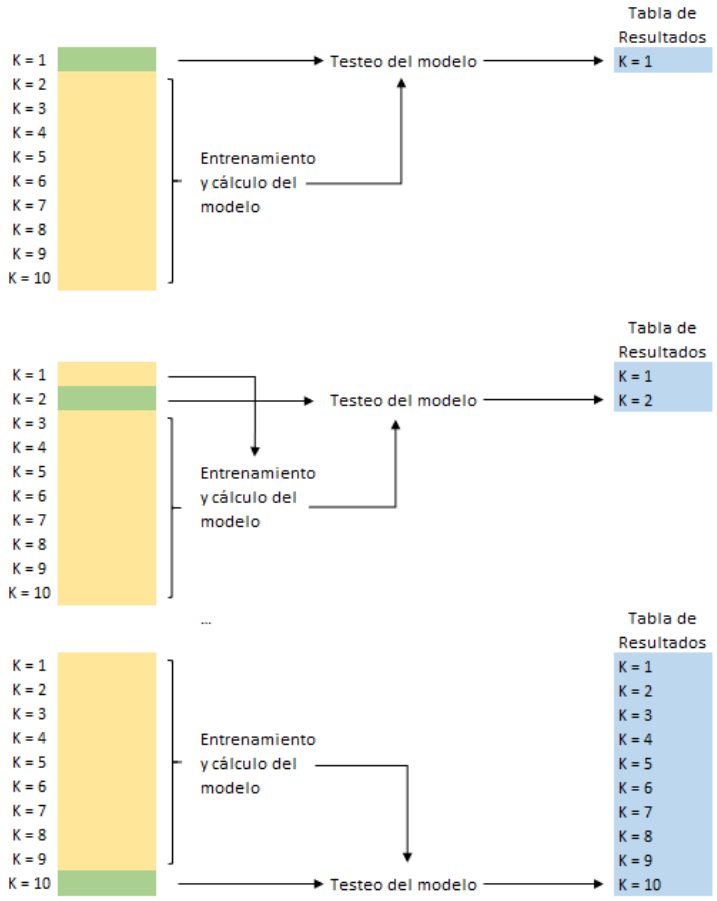


Ilustración 7: Diagrama de funcionamiento del método Cross Validation con k = 10 (Elaboración propia)..

4.3. Coeficientes e importancia de los modelos

Otra forma de obtener información sobre las variables y su importancia es a través de los modelos, en este caso, a partir del Logit y del Random Forest. Por un lado, los modelos de regresión logística funcionan a través del cálculo de los coeficientes, los cuales muestran la relación que hay entre las variables predictoras y la variable objetivo.

En el caso del modelo Logit, la relación no es directa al no ser lineal, aun así, el modelo muestra qué variables son significantes y cuáles no. Por otro lado, el Random Forest, es un modelo más complejo de interpretar, ya que es una combinación de distintos árboles, pero también puede aportar información sobre la importancia de las variables. Para ello calcula el error del modelo para cada uno de los árboles excluyendo una de las variables, de esta manera, si el error del modelo aumentar implica que se está perdiendo una información relevante que aportaba dicha variable. A través del cálculo del modelo en R, se puede extraer esta información sin necesidad de hacer ningún cálculo extra. La complejidad de interpretación del SVM para este modelo es tan elevada, que es muy complicado sacar una información sobre cómo afectan las variables al modelo.

A continuación, se muestran los cálculos realizados con las variables descritas en la Tabla 2 para los modelos comentados, con la finalidad de extraer mayor información sobre las variables a utilizar. De esta manera, localizando las variables que aportan una mayor información y las que son irrelevantes, se puede conseguir un cálculo más eficiente siguiendo el Principio de Parsimonia, dónde un modelo más simple es el óptimo.

Logit

Variables	**Coeficientes**	**Significancia**	
Intercepto	-1,90E+03	< 2e-16	***
Antigüedad de la póliza	-5,30E+01	< 2e-16	***
Número de pólizas vigentes distintas de salud (Dicotómica)	4,42E+02	2.76e-06	***
Número de asegurados vigentes	-3,09E+02	4.11e-12	***
Edad media de la póliza	-2,13E+01	3.92e-12	***
Tipo de familia (Con hijos mayores de edad)	-3,49E+02	0.000975	***
Tipo de familia (Con hijos menores de edad)	-4,33E+01	0.672122	
Tipo de familia (Sin hijos)	-3,45E+02	0.121680	
Tipo de familia (Otros)	-1,29E+02	0.233844	
Bajas anteriores distintas de salud (Dicotómica)	-1,83E+01	0.791280	
Bajas anteriores del ramo de salud (Dicotómica)	-1,03E+02	0.203302	
Uso	-1,94E+01	7.26e-09	***
Autorizaciones	-4,75E+01	0.054278	.
Siniestralidad	3,56E-01	1.29e-07	***
Código de pago (Mensual)	2,61E+02	0.060891	.
Código de pago (Semestral)	-7,05E+01	0.798843	
Código de pago (Trimestral)	2,20E+02	0.249503	
Prima actual	2,60E-01	0.000133	***
Variación porcentual	-6,41E+03	< 2e-16	***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Tabla 5: Resumen del cálculo del Logit para todas las variables.

En esta primera imagen se observa el resumen del modelo Logit. Para cada variable se muestra el valor del coeficiente (β) y su significancia. A la derecha se marca con asteriscos el tipo de significancia, siendo *** las que tienen mayor significancia (p-value menor que 0.001) y quedan sin marcar aquellas que no son significantes, siguiendo el patrón marcado en la parte inferior de la tabla. Siguiendo el principio de parsimonia, se procede a calcular el mismo modelo excluyendo las variables que no tienen un grado de significancia óptimo: Familia, Bajas anteriores distintas de salud, Bajas anteriores de salud, Autorizaciones y Código de pago.

Variables	Coeficientes	Significancia	
Intercepto	-1,77E+03	< 2e-16	***
Antigüedad de la póliza	-5,59E+01	< 2e-16	***
Número de pólizas vigentes distintas de salud (Dicotómica)	4,54E+02	7.68e-07	***
Número de asegurados vigentes	-3,26E+02	3.86e-14	***
Edad media de la póliza	-2,12E+01	< 2e-16	***
Uso	-1,75E+01	5.10e-08	***
Siniestralidad	2,79E-01	3.87e-07	***
Prima actual	2,36E-01	0.000182	***
Variación porcentual	-6,35E+03	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Tabla 6: Modelo Logit con selección de variables.

Si se compara con la Tabla 5, el valor de los coeficientes ha variado ligeramente, pero sigue siendo de un valor similar y del mismo signo, con lo que el modelo no se ha visto afectado por este cambio y los valores de significancia siguen siendo óptimos.

Matriz de confusión Todas las variables

Predicción	Real 0	Real 1
0	40815	850
1	1229	778

Accuracy	0,952
Kappa	0,404
Sensitivity	0,971
Specificity	0,478
Positive Pr.	0,980
Negative Pr.	0,388

Matriz de confusión *

Predicción	Real 0	Real 1
0	40831	834
1	1225	782

Accuracy	0,953
Kappa	0,407
Sensitivity	0,971
Specificity	0,484
Positive Pr.	0,980
Negative Pr.	0,390

Tabla 7: Matriz de confusión Logit.

A partir de la matriz de confusión, se puede realizar una comparativa entre los resultados de los modelos. Ambos han sido calculados de la misma manera, con un *cut off* de 0.10, valor óptimo obtenido a partir del código, con la única diferencia de las variables empleadas. Tal y como se observa, la simplificación del modelo a partir de la reducción de variables no ha modificado en exceso el resultado, de hecho, se observa una ligera mejora en las métricas de Accuracy, Kappa, Especificidad y Negativos predichos. Siguiendo el Principio de Parsimonia, la simplicidad del modelo aporta una mejora en los resultados y una optimización del cálculo, tanto en tiempo de cálculo como en resultados.

Random Forest

Como se comentaba al principio del apartado, el Random Forest es un modelo más complejo de interpretar, pero la función *randomForest* permite calcular la importancia de las variables. Está se calcula a partir de la *Mean Decrease Accuracy*, la cual permite ver como el error que se produce cuando se elimina una variable del modelo dejando el resto constante. De esta manera, en la siguiente gráfica se muestran, de mayor a menor y de color más claro a más oscuro, el error que se produce al excluir esa variable. Destacan como importantes las variables: Variación porcentual, Siniestralidad, Uso y Prima actual. Por otro lado, se observa que las variables de Código de pago, Bajas anteriores de salud y de otros ramos y Número de pólizas vigentes no aportan prácticamente información al modelo.

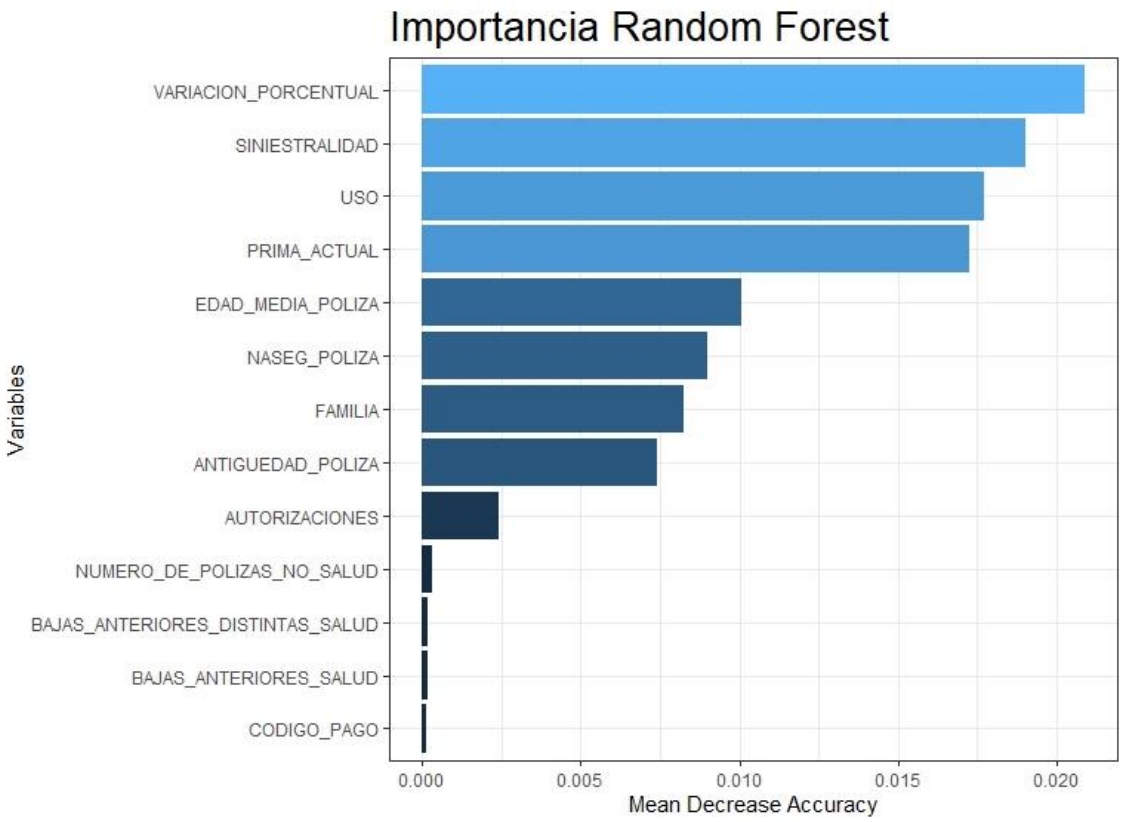


Ilustración 8: Importancia de las variables para el Random Forest.

De la misma manera que para el Logit, se ha realizado el cálculo de la matriz de confusión para todos las variables y para las variables con mayor importancia para realizar una comparativa.

Matriz de confusión Todas las variables		
	Real	
Predicción	0	1
0	41253	3
1	412	2004

Accuracy	0,990
Kappa	0,901
Sensitivity	0,990
Specificity	0,999
Positive Pr.	1,000
Negative Pr.	0,829

Matriz de confusión *		
	Real	
Predicción	0	1
0	41246	7
1	419	2000

Accuracy	0,990
Kappa	0,899
Sensitivity	0,990
Specificity	0,997
Positive Pr.	1,000
Negative Pr.	0,827

Tabla 8:Matriz de confusión Random Forest.

Se observa que, tras eliminar las cuatro variables antes comentadas, el resultado es bastante similar, siendo ligeramente peor en las métricas Kappa, Especificidad y Negativos predichos.

Variables	**Ganancia de información**	**Significancia Logit**	**Importancia RF**
Antigüedad de la póliza	0,0956	***	0,0074
Autorizaciones	0,0196	.	0,0024
Bajas anteriores del ramo de salud	0,2386		0,0002
Bajas anteriores distintas de salud	0,2186		0,0002
Código de pago	0,1239		0,0001
Edad media de la póliza	0,0988	***	0,0100
Número de asegurados vigentes	0,0243	***	0,0090
Número de pólizas vigentes distintas de salud	0,2444	***	0,0003
Prima actual	0,2519	***	0,0173
Siniestralidad	0,2220	***	0,0190
Tipo de familia	0,0574		0,0082
Uso	0,0605	***	0,0177
Variación porcentual	0,2654	***	0,0209

Tabla 9: Tabla resumen variables.

La importancia de estas para los modelos Logit y Random Forest son bastante similares, salvo para las variables Número de pólizas vigentes y Tipo de Familia. Las variables Antigüedad de la póliza, Edad media de la póliza, Número de asegurados vigentes y Uso, no tienen relevancia según la teoría de la Ganancia de información, pero si para los modelos. En cambio, las variables referentes a las bajas, sí que lo son para la Ganancia de información, pero no para los modelos. En este caso, siguiendo el Principio de parsimonia, se seleccionan para el cálculo de los modelos las variables importantes para los modelos y el Número de pólizas vigentes distintas de salud, ya que, según la teoría de la Ganancia de información y para el modelo Logit, sí que es relevante. Finalmente, las variables seleccionadas son: Antigüedad de la póliza, Edad media de la póliza, Número de asegurados vigentes, Número de pólizas vigentes distintas de salud, Prima actual, Siniestralidad, Uso y Variación porcentual de la prima.

Finalmente, se ha decidido estandarizar las variables de la base de datos, ya que, el tener distintas magnitudes los modelos, sobre todo los de *Machine Learning* pueden verse afectados y obtener resultados peores. Para la estandarización de las variables se han utilizado dos técnicas distintas según la naturaleza de los datos. Para las variables numéricas, se ha realizado la siguiente función de estandarización.

$$Z_i = \frac{X_i - \mu}{\sigma}$$

Siendo X_i el valor de las observaciones de cada variable, μ la media y σ la desviación estándar de las observaciones de la variable X_i.

Para las variables categóricas, se ha realizado un proceso de binarización previo a la estandarización. Para las variables categóricas de dos clases no hay problema ya que se encuentran en esta disposición, mientras que, para las variables con más de una categoría se ha binarizado de manera que para una variable con k categorías se obtienen k-1 variables binarias, ya que si todas las variables creadas tienen valor 0 implica que la restante tiene valor 1.

4.4. Resultados

Para la obtención de los resultados de los modelos se ha utilizado una partición de los datos del 70% para el entrenamiento, para el cálculo de los parámetros y del modelo. Una vez obtenido, se replicará en la base de test, del 30% respectivo, que no se ha visto involucrado en el entrenamiento del mismo. De modo que estos datos sean los mismos para todas las pruebas y modelos se ha establecido una semilla para el cálculo de las particiones. De esta manera, se comprobará si el modelo se ajusta bien y es capaz de predecir los valores de unos datos que no ha visto. Además, se utilizará el método de validación K-Fold Cross Validation para comprobar si realmente es un buen modelo para predecir en caso de añadir nuevas observaciones.

Logit

Para el modelo Logit se han realizado las pruebas sin pesos y para las distintas combinaciones entre 0.05 y 0.95 sumando 0.05 en cada prueba. El mejor resultado se ha obtenido para los pesos entre 0.05 y 0.20 de la clase minoritaria y entre 0.95 y 0.80 para la dominante respectivamente.

	Sin pesos	0,05 - 0,95	0,10 - 0,90	0,15 - 0,85	0,20 - 0,80
Accuracy	0,954	**0,965**	0,964	0,963	0,961
Kappa	0,393	0,391	0,414	**0,423**	0,421
Sensibilidad	**0,972**	0,968	0,969	0,971	0,971
Especificidad	0,459	**0,768**	0,698	0,632	0,575
Positivos Predichos	0,980	**0,996**	0,994	0,991	0,988
Negativos Predichos	**0,383**	0,275	0,310	0,339	0,356

Tabla 10: Resumen métricas Logit.

Si se observan las distintas pruebas, al aplicar los pesos el modelo mejora ligeramente en las métricas Accuracy, Kappa y Especificidad, mientras que pierde precisión en cuanto a sensibilidad y negativos predichos. Dado que el modelo está basado en una base de datos no balanceados y que las métricas de Accuracy, Sensibilidad y Positivos predichos son muy buenas en todas las pruebas, se focalizarán el estudio de los resultados en las métricas más importantes que son las relacionadas directamente con la predicción: Especificidad y Negativos predichos. Se aprecia que a medida que aumenta el valor del peso de la clase minoritaria, empeora la Especificidad, mientras que, contrariamente, aumenta el valor de negativos predichos. Esta variación no es constante, produciéndose una pérdida de Especificidad notable y un crecimiento ligero de los Negativos predichos. Se elige el modelo de pesos 0.05 – 0.95 y se aplica la validación del modelo.

Matriz de confusión test		
	Real	
Predicción	0	1
0	12454	47
1	411	156

Accuracy	0,965
Kappa	0,391
Sensitivity	0,968
Specificity	0,768
Positive Pr.	0,996
Negative Pr.	0,275

Matriz de confusión Cross Validation		
	Real	
Predicción	0	1
0	41511	154
1	1445	562

Accuracy	0,963
Kappa	0,398
Sensitivity	0,966
Specificity	0,785
Positive Pr.	0,996
Negative Pr.	0,280

Tabla 11: Validación Logit.

Comparando la matriz obtenida aplicando los resultados a los datos test con la obtenida a partir de aplicar Cross Validation a toda la base de datos, se alcanzan resultados muy similares, por lo que los resultados conseguidos son consistentes y se dan por válidos.

	Sin pesos	0,05 - 0,95	0,10 - 0,90	0,15 - 0,85	0,20 - 0,80
Accuracy	0,954	**0,965**	0,965	0,963	0,961
Kappa	0,394	0,397	0,423	**0,426**	0,416
Sensibilidad	**0,972**	0,968	0,970	0,971	0,971
Especificidad	0,462	**0,755**	0,709	0,642	0,577
Positivos Predichos	0,980	**0,996**	0,994	0,991	0,988
Negativos Predichos	**0,381**	0,282	0,317	0,339	0,349

Tabla 12: Resumen métricas con variables estandarizadas.

En esta última tabla, se muestra el cálculo del mismo modelo con las variables estandarizadas. Se observa que los resultados son muy similares y que la estandarización para este modelo no tiene relevancia.

Random Forest

Previo al cálculo del modelo, el Random Forest requiere de la optimización de los hiperparámetros. Para ello, se calcula el modelo probando los distintos parámetros mediante un *loop* y se almacenan el error de cada cálculo. Una vez obtenido el valor del error, se comparan los distintos valores y se elige el parámetro que consigue un menor error. Los parámetros obtenidos son los siguientes: 150 árboles, un nodo final y 2 predictores.

Una vez obtenidos los parámetros, se calcula el modelo con el sistema de pesos y se consiguen los siguientes resultados:

	Sin pesos	0,05 - 0,95	0,10 - 0,90	0,15 - 0,85	0,20 - 0,80
Accuracy	0,966	0,966	0,964	0,964	**0,966**
Kappa	0,417	0,438	0,440	0,440	**0,444**
Sensibilidad	0,969	0,970	**0,971**	0,971	0,970
Especificidad	**0,769**	0,740	0,679	0,677	0,759
Positivos Predichos	**0,996**	0,995	0,993	0,993	0,995
Negativos Predichos	0,300	0,326	**0,344**	**0,344**	0,328

Tabla 13: Resumen métricas Random Forest.

Las pruebas realizadas con los pesos no mejoran el modelo inicial, por lo que se mantiene éste como el mejor.

Al realizar el Cross Validation, se observa que este cae en *overfitting*. Este hecho indica que es posible que el modelo se ajuste demasiado bien a estos datos y que, si se aplicaran nuevas observaciones para una futura predicción, el modelo no funcionaría correctamente. En ese caso, se debería volver a ajustar el modelo y las variables utilizadas con la finalidad de encontrar uno que sea capaz de predecir nuevas observaciones.

Matriz de confusión test		
	Real	
Predicción	0	1
0	12442	59
1	381	186

Accuracy	0,966
Kappa	0,444
Sensitivity	0,970
Specificity	0,759
Positive Pr.	0,995
Negative Pr.	0,328

Matriz de confusión Cross Validation		
	Real	
Predicción	0	1
0	41619	7
1	46	2000

Accuracy	0,999
Kappa	0,986
Sensitivity	0,999
Specificity	0,997
Positive Pr.	1,000
Negative Pr.	0,978

Tabla 14: Validación Random Forest.

De la misma manera que para el Logit, la estandarización de los datos no ha aportado un cambio relevante, ya que los resultados son bastante similares. El mejor resultado para el modelo Random Forest lo encontramos, igualmente, para los datos no estandarizados.

	Sin pesos	0,05 - 0,95	0,10 - 0,90	0,15 - 0,85	0,20 - 0,80
Accuracy	0,964	0,963	0,965	**0,966**	0,963
Kappa	0,417	0,445	**0,446**	0,443	0,438
Sensibilidad	0,970	**0,972**	0,971	0,971	0,971
Especificidad	0,682	0,631	0,683	**0,722**	0,646
Positivos Predichos	0,993	0,990	0,993	**0,994**	0,991
Negativos Predichos	0,317	**0,365**	0,349	0,335	0,351

Tabla 15: Resumen métricas Random Forest con variables estandarizadas.

SVM

Para el modelo SVM se realizaron las mismas pruebas que para las anteriores, con unos resultados muy diferentes en comparación a los otros modelos. El primer paso es encontrar los parámetros que mejor se adecuen al modelo. Para ello, se ha programado un script con varios *loops*, que guardan los resultados obtenidos. Así, una vez calculado se observa la tendencia de cada uno de los parámetros. Posteriormente, se realiza otra prueba acotando valores para encontrar el que mejor se adapte a los datos. En el caso del kernel lineal, el coste que mejor resultados ha aportado es de 10, para el Polinómico se ha decidido un coste de 25 y 4 grados y para el Radial un coste de 75 y una gamma de 0.1. En la siguiente tabla se observan los resultados de estos modelos. En este caso no se han aplicado pesos, ya que no aportaban ningún cambio sobre los resultados.

	Lineal	Lineal *	Polinómico	Polinómico *	Radial	Radial *
Accuracy	**0,961**	**0,961**	0,958	0,953	0,956	0,951
Kappa	0,301	0,301	0,333	0,314	**0,341**	0,307
Sensibilidad	**0,999**	**0,999**	0,994	0,988	0,991	0,986
Especificidad	0,187	0,187	0,239	0,248	**0,261**	0,252
Positivos Predichos	0,961	0,961	0,963	0,963	**0,964**	0,964
Negativos Predichos	**0,935**	**0,935**	0,655	0,513	0,583	0,479

* Base de datos estandarizada

Tabla 16: Resumen métricas SVM.

Se observa que los resultados con el SVM son bastante peores que para los modelos Logit y Random Forest. Teniendo en cuenta que la métrica más importante en este modelo es la Especificidad, se elegiría el *Kernel* Radial como el que mejor se ha adaptado de los tres. Aun así, pese a todas las pruebas y los cambios producidos, los resultados son malos, por lo que no se ha podido obtener una hiperplano capaz de separar las clases con un buen grado de acierto.

5. Conclusiones

Pese a las dificultades inherentes a este tipo de base de datos, se han encontrado al menos dos modelos capaces de predecir con cierta validez la probabilidad de renovación o no renovación de una cartera. Teniendo en cuenta la variante menos técnica del trabajo, pese a que los valores Negativos predichos no tienen buenos resultados, si se ha conseguido un buen valor para la Especificidad. De cara a los objetivos de la hipotética empresa, los buenos resultados en la Especificidad son una buena noticia, ya que, si se realizaran acciones comerciales sobre los valores predichos, pese a que muchos de los tomadores se encuentran en el grupo que renovaría, se lograría localizar a gran parte de los que realmente se dan de baja. De esta manera, se podrían realizar acciones a esa gran parte de posibles bajas predichas, con el único inconveniente de que también se realizarían a un pequeño porcentaje de los que pensaban renovar.

Por lo tanto, se ha conseguido el objetivo de encontrar un modelo predictivo que se ajuste a las necesidades de la empresa y contribuya a mejorar la información que tiene esta sobre los clientes. En futuras investigaciones cabria mejorar el modelo para reducir errores del y aumentar la precisión. Para ello sería interesante obtener un mayor número de variables sobre características personales de los clientes o realizar combinaciones entre estas para poder obtener mayor información para trabajar. Además, también sería interesante probar técnicas para reducir el impacto de los datos no balanceados. Como se ha observado en el trabajo, el sistema de pesos ha sido capaz de mejorar los resultados en el caso del Logit, pero no ha sido relevante para el resto. En futuras investigaciones se podrían añadir técnicas de *Resampling* con las que equilibrar los datos y conseguir un modelo predictivo más sólido y preciso.

Referencias

Árboles de decisión, random forest, gradient boosting y C5.0 by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_C50.html

Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah?. ACM SIGKDD explorations newsletter, 2(2), 1-13.

Bolancé, C., Guillen, M., & AE, P. B. (2016). Predicting detection in non-life motor and home insurance. Lectures on Modeling and Simulation, 2, 107-120.

Bolancé, C., Guillen, M., & Padilla-Barreto, A. E. (2016, July). Predicting probability of customer churn in insurance. In International Conference on Modeling and Simulation in Engineering, Economics and Management (pp. 82-91). Springer, Cham.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152).

Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Cart. Classification and Regression Trees; Wadsworth and Brooks/Cole: Monterey, CA, USA.

Bustamante, F. C., Mora, J. R., Marino, C. M., & Torres, L. E. Selection at the AMSE Conferences-2016.

Charpentier, A. (Ed.). (2014). Computational actuarial science with R. CRC press.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.

Cristianini, N., & Shawe-Taylor, J. (2000). An introduction to support vector machines and other kernel-based learning methods. Cambridge university press.

Estany, M. G., Ayuso, M., & Carrillo, M. (2007). Econometria actuarial: material docent i casos pràctics (Vol. 57). Edicions Universitat Barcelona.

García, J. L., Chagolla, H., & Noriega, S. (2015). Modelos: efectos de la colinealidad en el modelado de regresión y su solución. Cultura Científica y Tecnológica, (17).

Gerber, G., Le Faou, Y., Lopez, O., & Trupin, M. (2018). The impact of churn on client value in health insurance, evaluation using a random forest under random censoring.

Kowalczyk, A. (2017). Support Vector Machine Succinctly.

Lantz, B. (2019). Machine learning with R: expert techniques for predictive modeling. Packt publishing ltd.

Máquinas de Vector Soporte (Support Vector Machines, SVMs) by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

Mason, C. H., & Perreault Jr, W. D. (1991). Collinearity, power, and interpretation of multiple regression analysis. Journal of marketing research, 28(3), 268-280.

Padilla Barreto, A. E., Bolancé Losilla, C., & Guillen, M. (2016). Cuantificación del riesgo para la tarificación en seguros de automóvil. Anales del Instituto de Actuarios Españoles, 2016, vol. 22, num. 3a. època, p. 1-24.

Padilla Barreto, A. E., Guillen, M., & Bolancé Losilla, C. (2017). Big-data Analytics en seguros. Anales del Instituto de Actuarios Españoles, 2017, vol. 23, p. 1-19.

Regresión logística simple y múltiple by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/27_regresion_logistica_simple_y_multiple.html

Rodríguez, M. J. G. (2000). La importancia del mantener la fidelidad de los clientes como un activo estratégico de gran valor para la marca. Esic Market, (107), 37-54.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

Shannon, C. E., & Weaver, W. (1949). The mathematical theory of communication, by CE Shannon (and recent contributions to the mathematical theory of communication), W. Weaver. University of illinois Press.

Spiteri, M., & Azzopardi, G. (2018, September). Customer churn prediction for a motor insurance company. In 2018 Thirteenth International Conference on Digital Information Management (ICDIM) (pp. 173-178). IEEE.

Tojo, J. F., & Sánchez, J. R. (2011). Orden, desorden y entropía en la construcción de la ciudad. Urban, (7), 8-15.

Tukey, J. W. (1977). Exploratory data analysis (Vol. 2, pp. 131-160).

Vapnik, V. (1999). The nature of statistical learning theory. Springer science & business media.

Vapnik, V.N. (1982). Estimation of Dependences Based on Empirical Data, Addendum 1, New York: Springer Verlag.

Wang, G. C., & Akabay, C. K. (1994). Autocorrelation: Problems and solutions in regression modeli. The Journal of Business Forecasting, 13(4), 18.

Yap, B. W., Abd Rani, K., Abd Rahman, H. A., Fong, S., Khairudin, Z., & Abdullah, N. N. (2014). An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In Proceedings of the first international conference on advanced data and information engineering (DaEng-2013) (pp. 13-22). Springer, Singapore.

Paquetes R

A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

Brian Ripley (2021). tree: Classification and Regression Trees. R package version 1.0-41. https://CRAN.R-project.org/package=tree

David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2021). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.7-6. https://CRAN.R-project.org/package=e1071

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

Hadley Wickham (2019). stringr: Simple, Consistent Wrappers for Common String Operations. R package version 1.4.0. https://CRAN.R-project.org/package=stringr

Hadley Wickham (2021). tidyr: Tidy Messy Data. R package version 1.1.3. https://CRAN.R-project.org/package=tidyr

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.5. https://CRAN.R-project.org/package=dplyr

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Max Kuhn and Hadley Wickham (2021). recipes: Preprocessing and Feature Engineering Steps for Modeling. R package version 0.1.17. https://CRAN.R-project.org/package=recipes

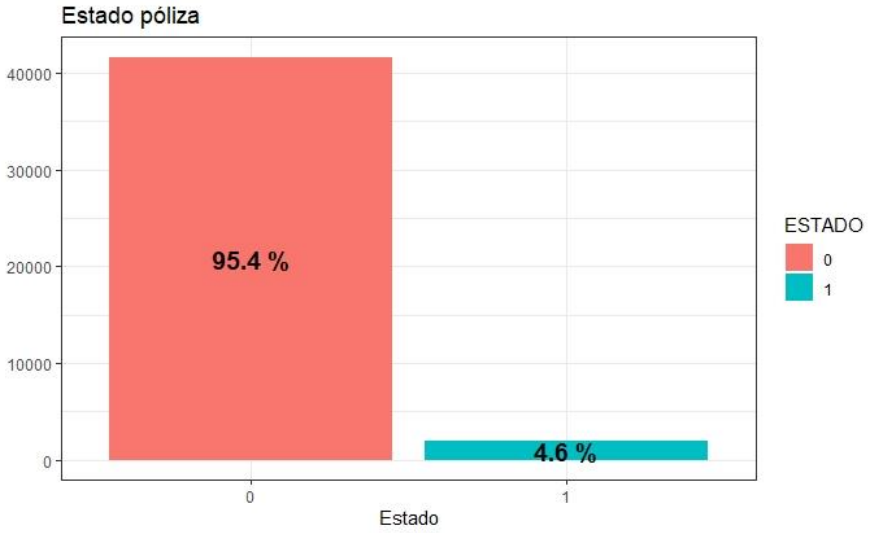
Max Kuhn (2021). caret: Classification and Regression Training. R package version 6.0-90. https://CRAN.R-project.org/package=caret

Stephen Milborrow (2021). rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'. R package version 3.1.0. https://CRAN.R-project.org/package=rpart.plot

Terry Therneau and Beth Atkinson (2019). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-15. https://CRAN.R-project.org/package=rpart

Anexo

Análisis de las variables



	0	1
1	41665	2007

Ilustración 9: Variable objetivo (Elaboración propia).

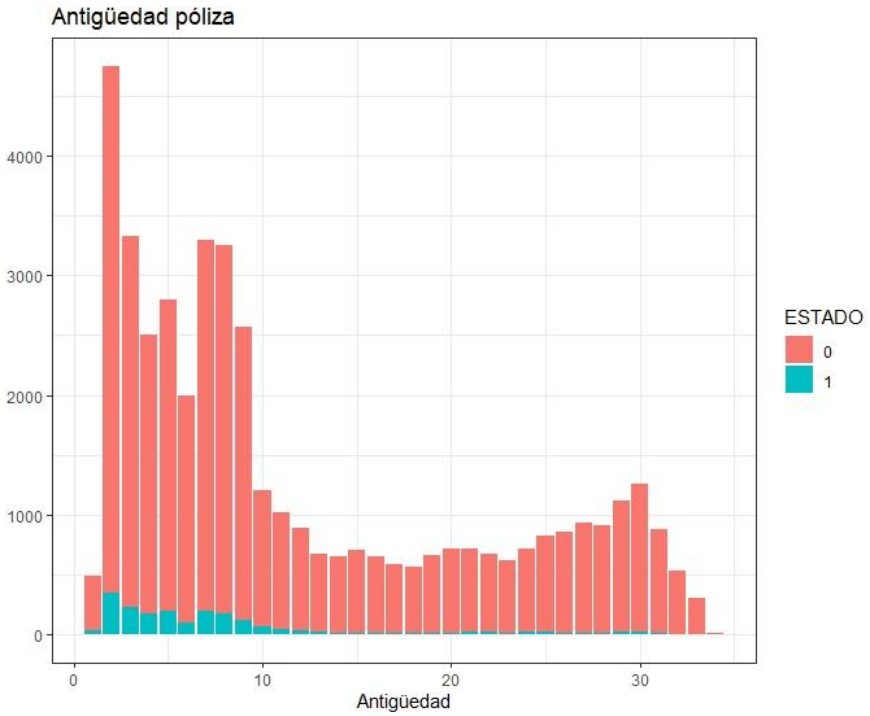


Ilustración 10: Variable Antigüedad de la póliza (Elaboración propia).

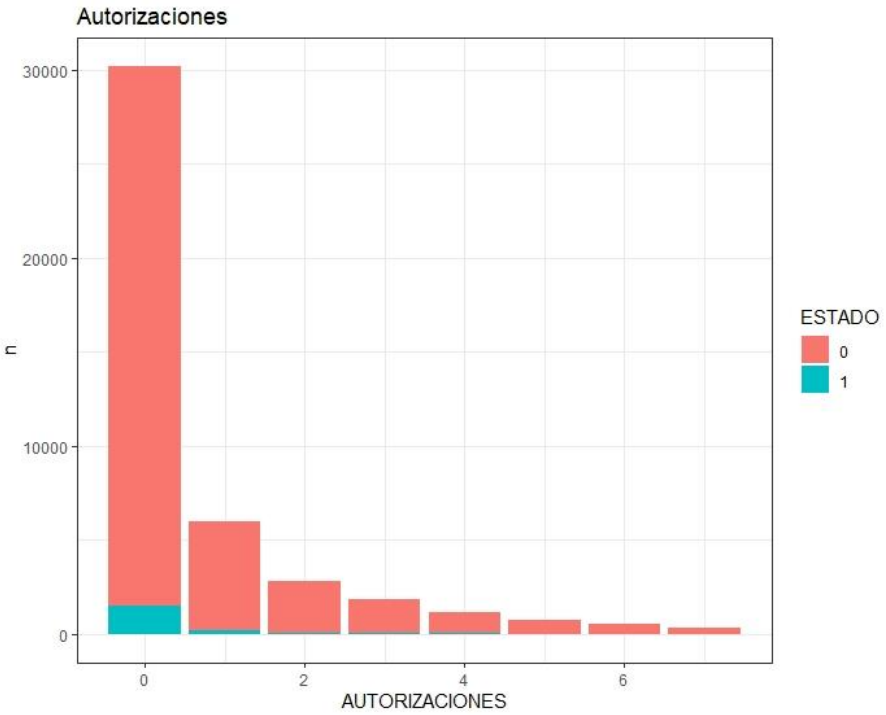


Ilustración 11:Autorizaciones (Elaboración propia).

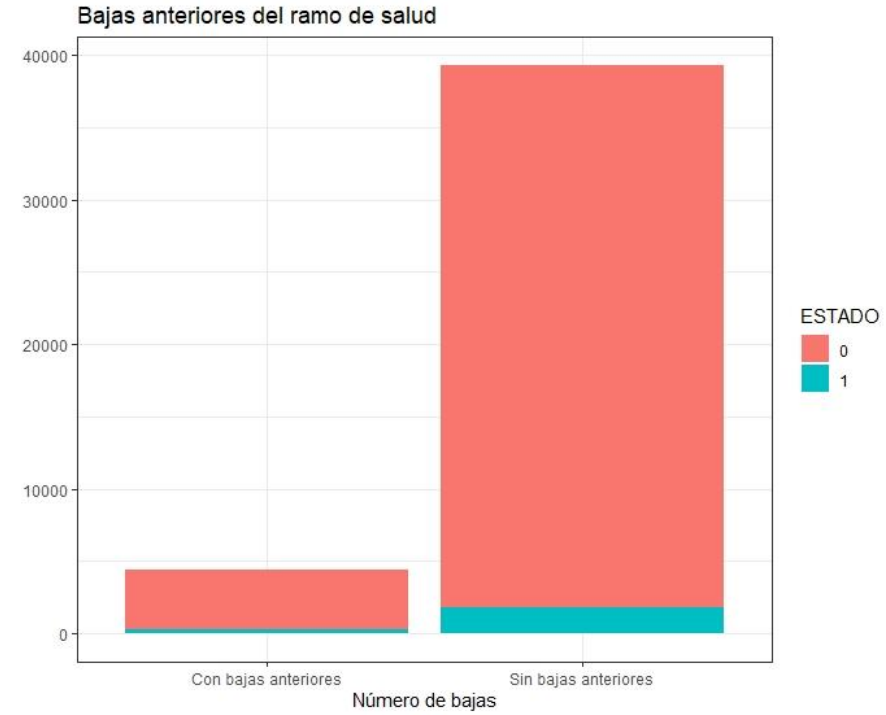


Ilustración 12: Bajas anteriores del ramo de salud (Elaboración propia).

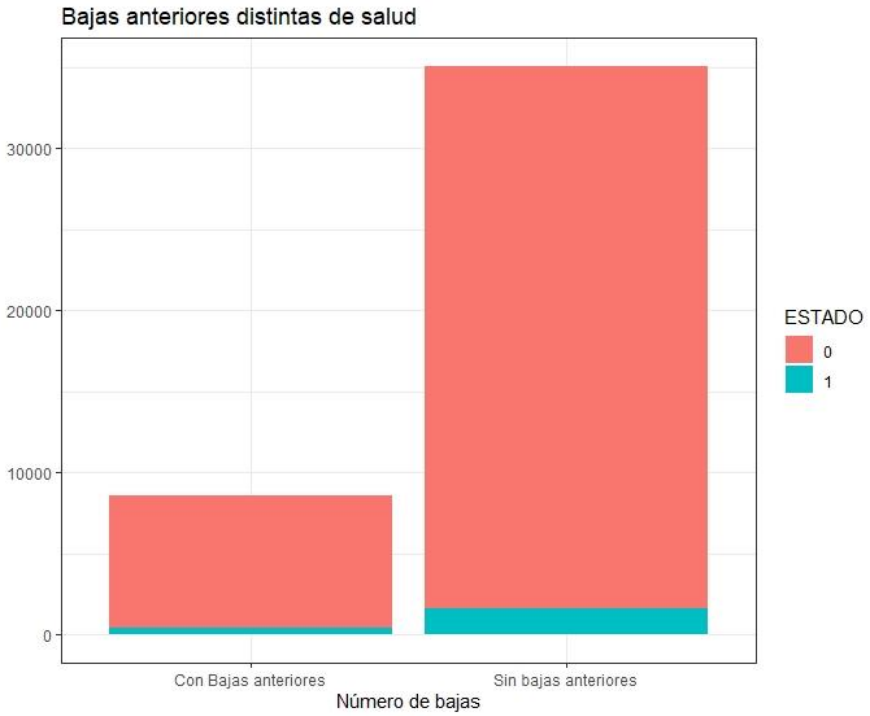


Ilustración 13:Bajas anteriores distintas del ramo de salud (Elaboración propia).

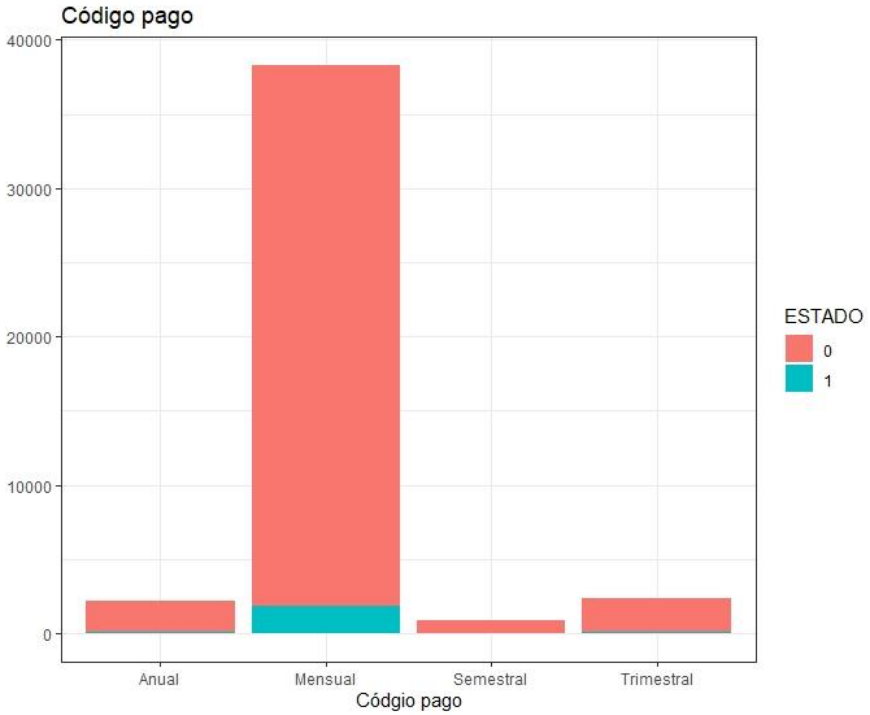


Ilustración 14: Código de pago (Elaboración propia).

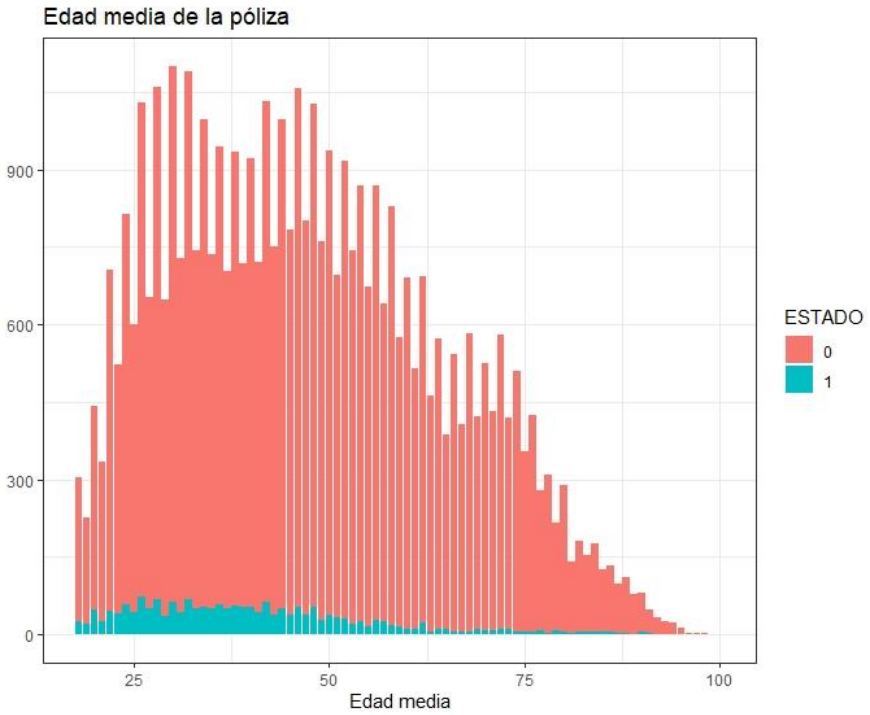


Ilustración 15: Edad media de la póliza (Elaboración propia).

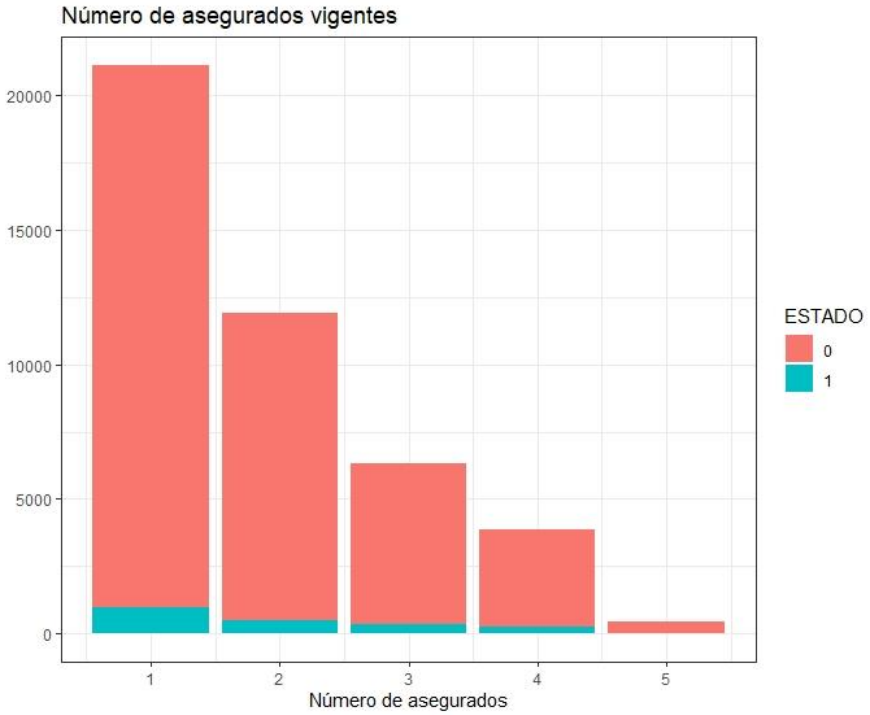


Ilustración 16: Número de asegurados vigentes (Elaboración propia).

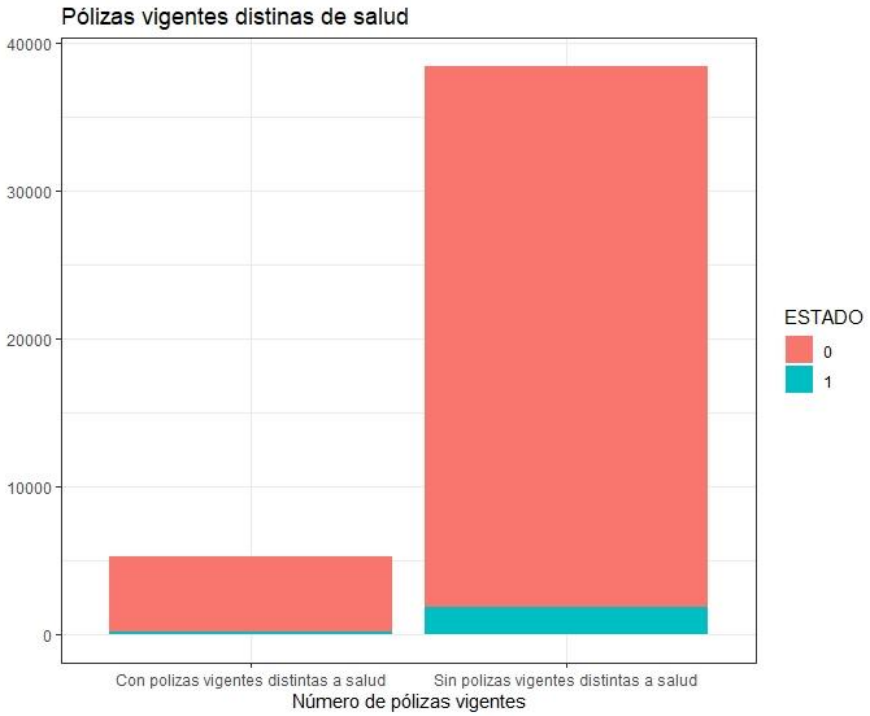


Ilustración 17: Pólizas vigentes distintas de salud (Elaboración propia).

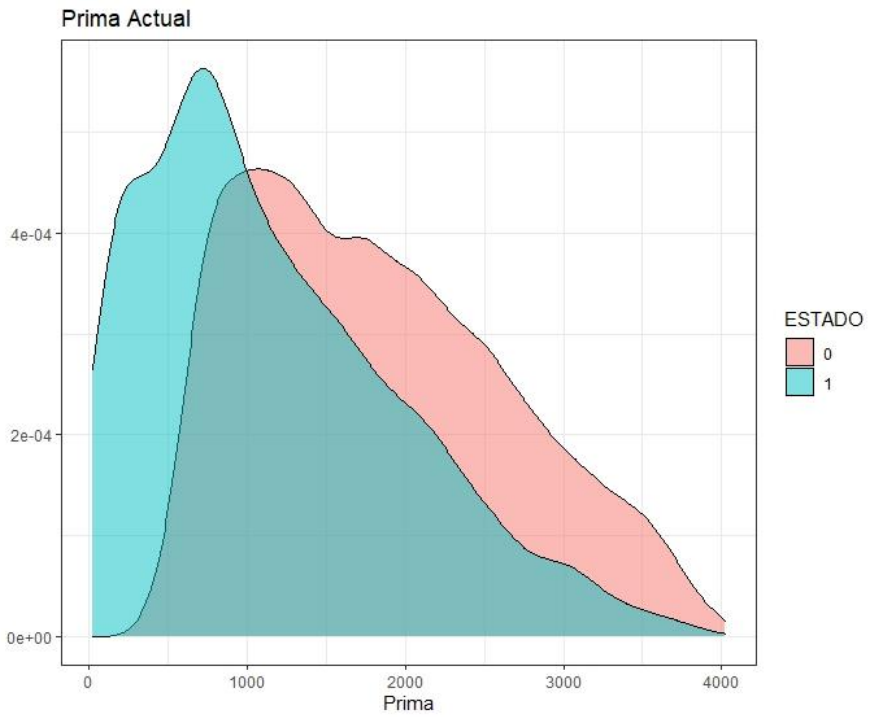


Ilustración 18: Prima actual (Elaboración propia).

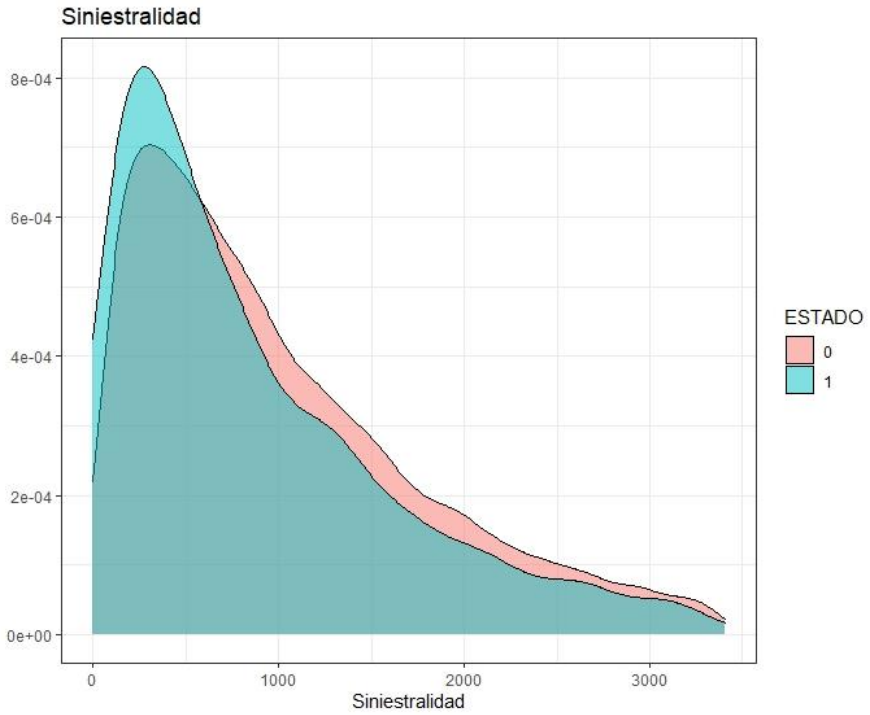


Ilustración 19: Siniestralidad (Elaboración propia).

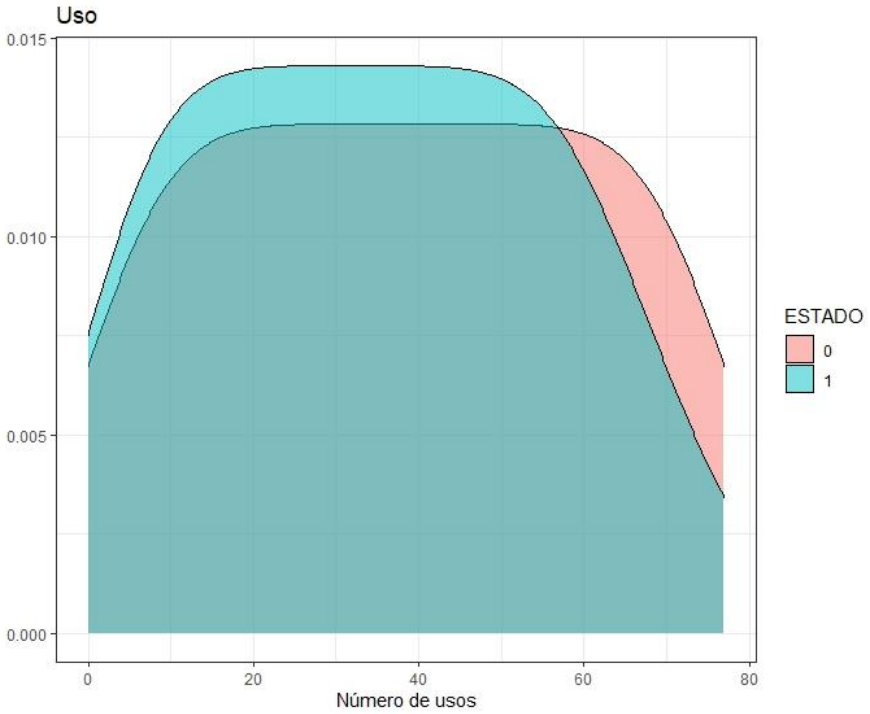


Ilustración 20: Número de usos (Elaboración propia).

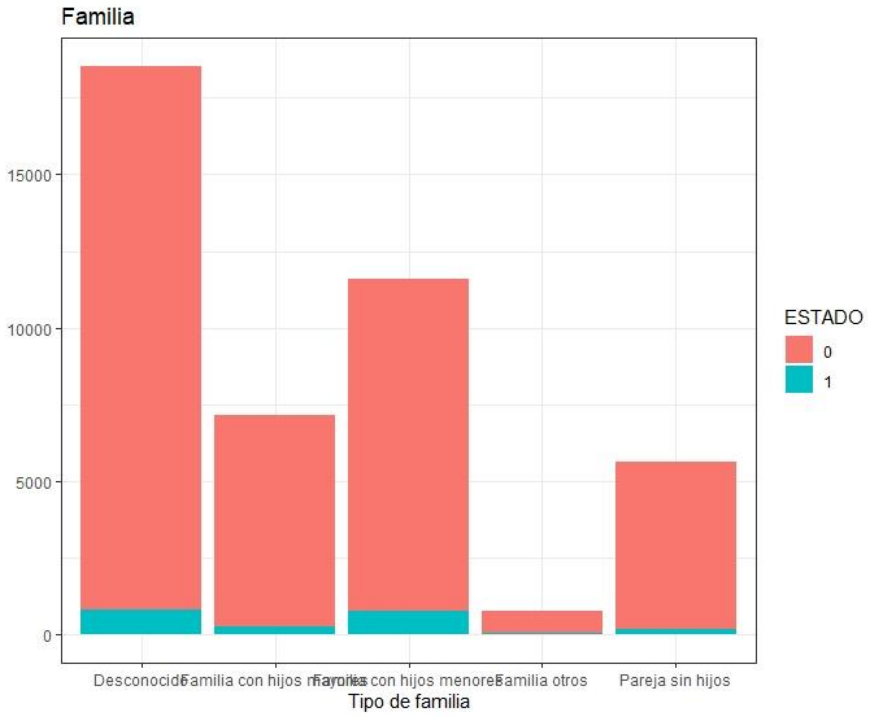


Ilustración 21: Tipo de Familia (Elaboración propia).

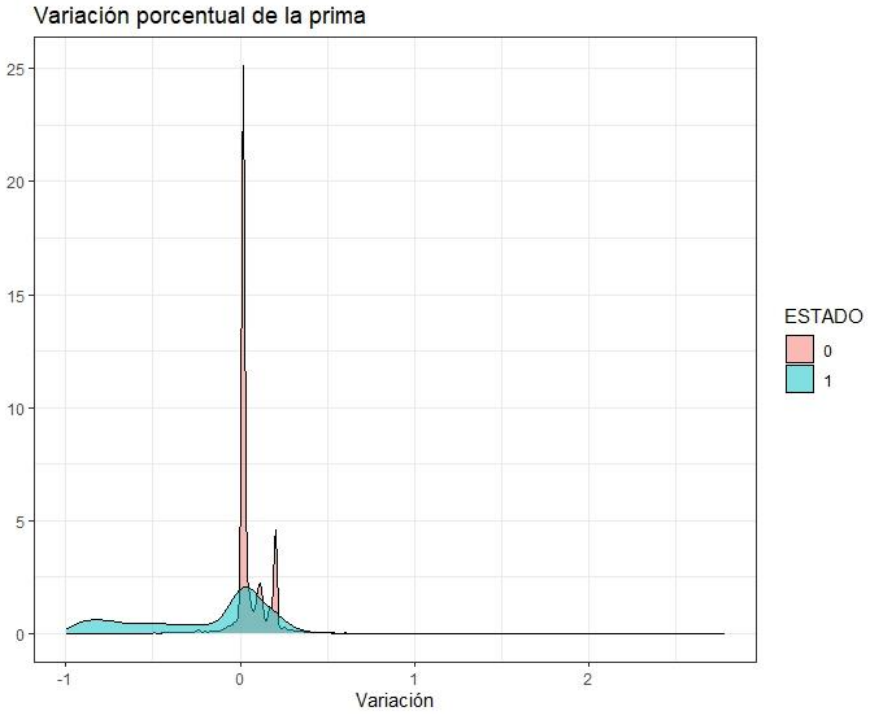


Ilustración 22: Variación porcentual de la prima (Elaboración propia).

Correlaciones

Correlación de Pearson	Antigüedad póliza	Autorizaciones	Bajas anteriores distintas de salud	Bajas anteriores salud	Edad media de la póliza	Número de asegurados vigentes	Número de pólizas vigentes distintas de salud	Prima actual	Prima anterior	Siniestralidad	Uso	Variación prima	Variación prima porcentual
Antigüedad póliza	1,00	0,00	0,04	-0,10	0,68	-0,23	0,00	0,28	0,30	0,03	0,07	-0,05	-0,01
Autorizaciones	0,00	1,00	0,02	0,01	0,01	0,11	0,01	0,15	0,13	0,55	0,30	0,06	0,07
Bajas anteriores distintas de salud	0,04	0,02	1,00	0,05	0,03	0,02	0,48	0,02	0,02	0,01	0,01	-0,01	0,00
Bajas anteriores salud	-0,10	0,01	0,05	1,00	-0,08	0,09	0,03	0,01	0,01	0,04	0,05	0,00	0,00
Edad media de la póliza	0,68	0,01	0,03	-0,08	1,00	-0,47	-0,02	0,12	0,12	-0,02	0,00	0,02	0,04
Número de asegurados vigentes	-0,23	0,11	0,02	0,09	-0,47	1,00	0,05	0,59	0,63	0,28	0,28	-0,08	-0,01
Número de pólizas vigentes distintas de salud	0,00	0,01	0,48	0,03	-0,02	0,05	1,00	0,01	0,01	0,01	0,01	0,00	0,00
Prima actual	0,28	0,15	0,02	0,01	0,12	0,59	0,01	1,00	0,96	0,39	0,42	0,17	0,31
Prima anterior	0,30	0,13	0,02	0,01	0,12	0,63	0,01	0,96	1,00	0,35	0,38	-0,09	0,01
Siniestralidad	0,03	0,55	0,01	0,04	-0,02	0,28	0,01	0,39	0,35	1,00	0,80	0,16	0,19
Uso	0,07	0,30	0,01	0,05	0,00	0,28	0,01	0,42	0,38	0,80	1,00	0,16	0,20
Variación prima porcentual	-0,05	0,06	-0,01	0,00	0,02	-0,08	0,00	0,17	-0,09	0,16	0,16	1,00	0,87
Variación prima	-0,01	0,07	0,00	0,00	0,04	-0,01	0,00	0,31	0,01	0,19	0,20	0,87	1,00

Tabla 17: Correlación de Pearson.

FIV

Factor de incremento de la varianza	Antigüedad póliza	Autorizaciones	Bajas anteriores distintas de salud	Bajas anteriores salud	Edad media de la póliza	Número de asegurados vigentes	Número de pólizas vigentes distintas de salud	Prima actual	Prima anterior	Siniestralidad	Uso	Variación prima	Variación prima porcentual
Antigüedad póliza	-	1,0	1,0	1,0	1,9	1,1	1,0	1,1	1,1	1,0	1,0	1,0	1,0
Autorizaciones	1,0	-	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,4	1,1	1,0	1,0
Bajas anteriores distintas de salud	1,0	1,0	-	1,0	1,0	1,0	1,3	1,0	1,0	1,0	1,0	1,0	1,0
Bajas anteriores salud	1,0	1,0	1,0	-	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Edad media de la póliza	1,9	1,0	1,0	1,0	-	1,3	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Número de asegurados vigentes	1,1	1,0	1,0	1,0	1,3	-	1,0	1,5	1,6	1,1	1,1	1,0	1,0
Número de pólizas vigentes distintas de salud	1,0	1,0	1,3	1,0	1,0	1,0	-	1,0	1,0	1,0	1,0	1,0	1,0
Prima actual	1,1	1,0	1,0	1,0	1,0	1,5	1,0	-	11,7	1,2	1,2	1,0	1,1
Prima anterior	1,1	1,0	1,0	1,0	1,0	1,6	1,0	11,7	-	1,1	1,2	1,0	1,0
Siniestralidad	1,0	1,4	1,0	1,0	1,0	1,1	1,0	1,2	1,1	-	1,1	1,0	1,0
Uso	1,0	1,1	1,0	1,0	1,0	1,1	1,0	1,2	1,2	1,1	-	1,0	1,0
Variación prima	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	-	4,0
Variación prima porcentual	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,1	1,0	1,0	1,0	4,0	-

Tabla 18: Factor de incremento de la varianza

Matrices de confusión Logit

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28595	569
1	876	564

Accuracy	0,953
Kappa	0,414
Sensitivity	0,970
Specificity	0,498
Positive Pr.	0,980
Negative Pr.	0,392

Matriz de confusión test		
	Real	
Predicción	0	1
0	12245	256
1	350	217

Accuracy	0,954
Kappa	0,393
Sensitivity	0,972
Specificity	0,459
Positive Pr.	0,980
Negative Pr.	0,383

Tabla 19: Matriz de confusión Logit sin pesos.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29050	114
1	1032	408

Accuracy	0,963
Kappa	0,401
Sensitivity	0,966
Specificity	0,782
Positive Pr.	0,996
Negative Pr.	0,283

Matriz de confusión test		
	Real	
Predicción	0	1
0	12454	47
1	411	156

Accuracy	0,965
Kappa	0,391
Sensitivity	0,968
Specificity	0,768
Positive Pr.	0,996
Negative Pr.	0,275

Tabla 20:Matriz de confusión Logit 0.05 - 0.95.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28992	172
1	980	460

Accuracy	0,962
Kappa	0,428
Sensitivity	0,967
Specificity	0,728
Positive Pr.	0,994
Negative Pr.	0,319

Matriz de confusión test		
	Real	
Predicción	0	1
0	12425	76
1	391	176

Accuracy	0,964
Kappa	0,414
Sensitivity	0,969
Specificity	0,698
Positive Pr.	0,994
Negative Pr.	0,310

Tabla 21:Matriz de confusión Logit 0.10 - 0.90.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28912	252
1	940	500

Accuracy	0,961
Kappa	0,438
Sensitivity	0,969
Specificity	0,665
Positive Pr.	0,991
Negative Pr.	0,347

Matriz de confusión test		
	Real	
Predicción	0	1
0	12389	112
1	375	192

Accuracy	0,963
Kappa	0,423
Sensitivity	0,971
Specificity	0,632
Positive Pr.	0,991
Negative Pr.	0,339

Tabla 22:Matriz de confusión Logit 0.15 - 0.85.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28828	336
1	915	525

Accuracy	0,959
Kappa	0,436
Sensitivity	0,969
Specificity	0,610
Positive Pr.	0,988
Negative Pr.	0,365

Matriz de confusión test		
	Real	
Predicción	0	1
0	12352	149
1	365	202

Accuracy	0,961
Kappa	0,421
Sensitivity	0,971
Specificity	0,575
Positive Pr.	0,988
Negative Pr.	0,356

Tabla 23:Matriz de confusión Logit 0.20 - 0.80.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28582	582
1	879	561

Accuracy	0,952
Kappa	0,410
Sensitivity	0,970
Specificity	0,491
Positive Pr.	0,980
Negative Pr.	0,390

Matriz de confusión test		
	Real	
Predicción	0	1
0	12249	252
1	351	216

Accuracy	0,954
Kappa	0,394
Sensitivity	0,972
Specificity	0,462
Positive Pr.	0,980
Negative Pr.	0,381

Tabla 24: Matriz de confusión Logit con variables estandarizadas sin pesos.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29073	91
1	1028	412

Accuracy	0,963
Kappa	0,410
Sensitivity	0,966
Specificity	0,819
Positive Pr.	0,997
Negative Pr.	0,286

Matriz de confusión test		
	Real	
Predicción	0	1
0	12449	52
1	407	160

Accuracy	0,965
Kappa	0,397
Sensitivity	0,968
Specificity	0,755
Positive Pr.	0,996
Negative Pr.	0,282

Tabla 25: Matriz de confusión Logit con variables estandarizadas 0.05-0.95.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28996	168
1	970	470

Accuracy	0,963
Kappa	0,436
Sensitivity	0,968
Specificity	0,737
Positive Pr.	0,994
Negative Pr.	0,326

Matriz de confusión test		
	Real	
Predicción	0	1
0	12427	74
1	387	180

Accuracy	0,965
Kappa	0,423
Sensitivity	0,970
Specificity	0,709
Positive Pr.	0,994
Negative Pr.	0,317

Tabla 26: Matriz de confusión Logit con variables estandarizadas 0.10 -0.90.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28910	254
1	941	499

Accuracy	0,961
Kappa	0,437
Sensitivity	0,968
Specificity	0,663
Positive Pr.	0,991
Negative Pr.	0,347

Matriz de confusión test		
	Real	
Predicción	0	1
0	12394	107
1	375	192

Accuracy	0,963
Kappa	0,426
Sensitivity	0,971
Specificity	0,642
Positive Pr.	0,991
Negative Pr.	0,339

Tabla 27: Matriz de confusión Logit con variables estandarizadas 0.15-0.85.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	28831	333
1	914	526

Accuracy	0,959
Kappa	0,438
Sensitivity	0,969
Specificity	0,612
Positive Pr.	0,989
Negative Pr.	0,365

Matriz de confusión test		
	Real	
Predicción	0	1
0	12356	145
1	369	198

Accuracy	0,961
Kappa	0,416
Sensitivity	0,971
Specificity	0,577
Positive Pr.	0,988
Negative Pr.	0,349

Tabla 28: Matriz de confusión Logit con variables estandarizadas 0.20-0.80.

Matrices de confusión Random Forest

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29097	67
1	1034	406

Accuracy	0,964
Kappa	0,411
Sensitivity	0,966
Specificity	0,858
Positive Pr.	0,998
Negative Pr.	0,282

Matriz de confusión test		
	Real	
Predicción	0	1
0	12450	51
1	397	170

Accuracy	0,966
Kappa	0,417
Sensitivity	0,969
Specificity	0,769
Positive Pr.	0,996
Negative Pr.	0,300

Tabla 29: Matriz de confusión Random Forest sin pesos.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29013	61
1	1024	416

Accuracy	0,964
Kappa	0,420
Sensitivity	0,966
Specificity	0,872
Positive Pr.	0,998
Negative Pr.	0,289

Matriz de confusión test		
	Real	
Predicción	0	1
0	12409	92
1	372	195

Accuracy	0,964
Kappa	0,440
Sensitivity	0,971
Specificity	0,679
Positive Pr.	0,993
Negative Pr.	0,344

Tabla 30:Matriz de confusión Random Forest 0.05 - 0.95.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29105	59
1	1034	406

Accuracy	0,964
Kappa	0,413
Sensitivity	0,966
Specificity	0,873
Positive Pr.	0,998
Negative Pr.	0,282

Matriz de confusión test		
	Real	
Predicción	0	1
0	12408	93
1	372	195

Accuracy	0,964
Kappa	0,440
Sensitivity	0,971
Specificity	0,677
Positive Pr.	0,993
Negative Pr.	0,344

Tabla 31:Matriz de confusión Random Forest 0.10 - 0.90.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29096	68
1	1027	413

Accuracy	0,964
Kappa	0,416
Sensitivity	0,966
Specificity	0,859
Positive Pr.	0,998
Negative Pr.	0,287

Matriz de confusión test		
	Real	
Predicción	0	1
0	12442	59
1	381	186

Accuracy	0,966
Kappa	0,444
Sensitivity	0,970
Specificity	0,759
Positive Pr.	0,995
Negative Pr.	0,328

Tabla 32:Matriz de confusión Random Forest 0.15 - 0.85.

Matriz de confusión test		
	Real	
Predicción	0	1
0	12442	59
1	381	186

Accuracy	0,966
Kappa	0,444
Sensitivity	0,970
Specificity	0,759
Positive Pr.	0,995
Negative Pr.	0,328

Matriz de confusión Cross Validation		
	Real	
Predicción	0	1
0	41619	7
1	46	2000

Accuracy	0,999
Kappa	0,986
Sensitivity	0,999
Specificity	0,997
Positive Pr.	1,000
Negative Pr.	0,978

Tabla 33:Matriz de confusión Random Forest 0.20 - 0.80.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29058	106
1	1018	422

Accuracy	0,963
Kappa	0,414
Sensitivity	0,966
Specificity	0,799
Positive Pr.	0,996
Negative Pr.	0,293

Matriz de confusión test		
	Real	
Predicción	0	1
0	12417	84
1	387	180

Accuracy	0,964
Kappa	0,417
Sensitivity	0,970
Specificity	0,682
Positive Pr.	0,993
Negative Pr.	0,317

Tabla 34: Matriz de confusión Random Forest con variables estandarizadas sin pesos.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29105	59
1	1029	411

Accuracy	0,964
Kappa	0,417
Sensitivity	0,966
Specificity	0,874
Positive Pr.	0,998
Negative Pr.	0,285

Matriz de confusión test		
	Real	
Predicción	0	1
0	12380	121
1	360	207

Accuracy	0,963
Kappa	0,445
Sensitivity	0,972
Specificity	0,631
Positive Pr.	0,990
Negative Pr.	0,365

Tabla 35: Matriz de confusión Random Forest con variables estandarizadas 0.05-0.95.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29097	67
1	1031	409

Accuracy	0,964
Kappa	0,413
Sensitivity	0,966
Specificity	0,859
Positive Pr.	0,998
Negative Pr.	0,284

Matriz de confusión test		
	Real	
Predicción	0	1
0	12409	92
1	369	198

Accuracy	0,965
Kappa	0,446
Sensitivity	0,971
Specificity	0,683
Positive Pr.	0,993
Negative Pr.	0,349

Tabla 36: Matriz de confusión Random Forest con variables estandarizadas 0.10 -0.90.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29090	74
1	1023	417

Accuracy	0,964
Kappa	0,418
Sensitivity	0,966
Specificity	0,849
Positive Pr.	0,997
Negative Pr.	0,290

Matriz de confusión test		
	Real	
Predicción	0	1
0	12428	73
1	377	190

Accuracy	0,966
Kappa	0,443
Sensitivity	0,971
Specificity	0,722
Positive Pr.	0,994
Negative Pr.	0,335

Tabla 37: Matriz de confusión Random Forest con variables estandarizadas 0.15-0.85

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29093	71
1	1022	418

Accuracy	0,964
Kappa	0,420
Sensitivity	0,966
Specificity	0,855
Positive Pr.	0,998
Negative Pr.	0,290

Matriz de confusión test		
	Real	
Predicción	0	1
0	12392	109
1	368	199

Accuracy	0,963
Kappa	0,438
Sensitivity	0,971
Specificity	0,646
Positive Pr.	0,991
Negative Pr.	0,351

Tabla 38: Matriz de confusión Random Forest con variables estandarizadas 0.20-0.80.

Matrices de confusión SVM

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29303	997
1	19	285

Accuracy	0,967
Kappa	0,349
Sensitivity	0,999
Specificity	0,222
Positive Pr.	0,967
Negative Pr.	0,939

Matriz de confusión test		
	Real	
Predicción	0	1
0	14397	584
1	9	134

Accuracy	0,961
Kappa	0,301
Sensitivity	0,999
Specificity	0,187
Positive Pr.	0,961
Negative Pr.	0,935

Tabla 39: Matriz de confusión SVM lineal con coste 10.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29303	997
1	19	285

Accuracy	0,967
Kappa	0,349
Sensitivity	0,999
Specificity	0,222
Positive Pr.	0,967
Negative Pr.	0,939

Matriz de confusión test		
	Real	
Predicción	0	1
0	14397	584
1	9	134

Accuracy	0,961
Kappa	0,301
Sensitivity	0,999
Specificity	0,187
Positive Pr.	0,961
Negative Pr.	0,935

Tabla 40: Matriz de confusión SVM lineal con coste 10 y datos estandarizados.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29317	797
1	5	485

Accuracy	0,974
Kappa	0,536
Sensitivity	1,000
Specificity	0,378
Positive Pr.	0,974
Negative Pr.	0,991

Matriz de confusión test		
	Real	
Predicción	0	1
0	14316	547
1	90	172

Accuracy	0,958
Kappa	0,333
Sensitivity	0,994
Specificity	0,239
Positive Pr.	0,963
Negative Pr.	0,655

Tabla 41: Matriz de confusión SVM Polinómico con coste 25 y polinomio de grado 4.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29317	698
1	5	584

Accuracy	0,977
Kappa	0,614
Sensitivity	1,000
Specificity	0,456
Positive Pr.	0,977
Negative Pr.	0,992

Matriz de confusión test		
	Real	
Predicción	0	1
0	14237	540
1	169	179

Accuracy	0,953
Kappa	0,314
Sensitivity	0,988
Specificity	0,248
Positive Pr.	0,963
Negative Pr.	0,513

Tabla 42: Matriz de confusión SVM Polinómico con coste 25 y polinomio de grado 4 con datos estandarizados.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29320	575
1	2	707

Accuracy	0,981
Kappa	0,701
Sensitivity	1,000
Specificity	0,552
Positive Pr.	0,981
Negative Pr.	0,997

Matriz de confusión test		
	Real	
Predicción	0	1
0	14272	531
1	134	188

Accuracy	0,956
Kappa	0,341
Sensitivity	0,991
Specificity	0,261
Positive Pr.	0,964
Negative Pr.	0,583

Tabla 43: Matriz de confusión SVM radial con coste 75 y gamma 0.1.

Matriz de confusión entrenamiento		
	Real	
Predicción	0	1
0	29322	542
1	0	740

Accuracy	0,982
Kappa	0,723
Sensitivity	1,000
Specificity	0,577
Positive Pr.	0,982
Negative Pr.	1,000

Matriz de confusión test		
	Real	
Predicción	0	1
0	14209	538
1	197	181

Accuracy	0,951
Kappa	0,307
Sensitivity	0,986
Specificity	0,252
Positive Pr.	0,964
Negative Pr.	0,479

Tabla 44: Matriz de confusión SVM radial con coste 75 y gamma 0.1 con datos estandarizados.

Importancia Variables Random Forest.

Variables	**Mean Decrease Accuracy**	**Mean Decrease Gini**
Variación porcentual	0,02089	113.035.436
Siniestralidad	0,01901	45.010.709
Uso	0,01771	29.285.579
Prima actual	0,01726	80.290.618
Edad media de la póliza	0,01005	31.715.871
Número de asegurados vigentes	0,00901	10.110.239
Tipo de familia	0,00825	8.027.058
Antigüedad de la póliza	0,00740	22.909.233
Autorizaciones	0,00239	9.362.494
Número de pólizas vigentes distintas de salud (Dicotómica)	0,00030	2.627.353
Bajas anteriores distintas de salud (Dicotómica)	0,00018	4.113.460
Bajas anteriores del ramo de salud (Dicotómica)	0,00016	3.353.152
Código de pago	0,00013	3.799.962

Tabla 45: Importancia de las variables Random Forest.

Parámetros Random Forest.

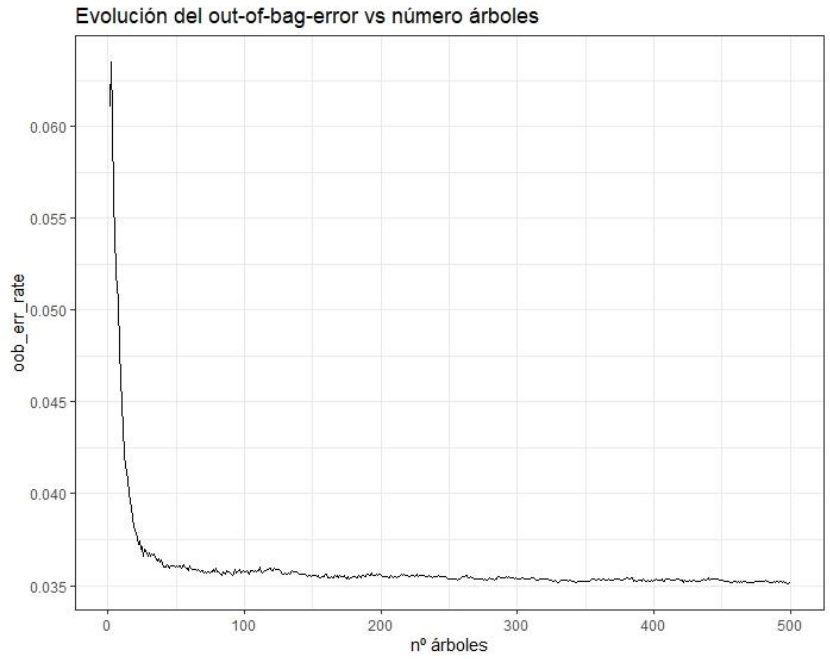


Ilustración 23: Cálculo parámetros número de árboles Random Forest.

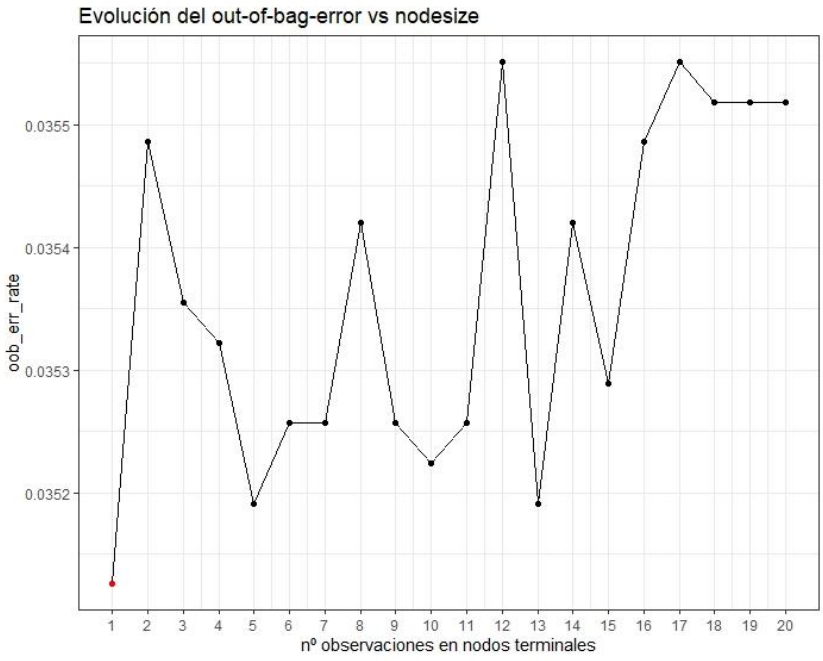


Ilustración 24: Cálculo parámetros número de nodos terminales Random Forest.

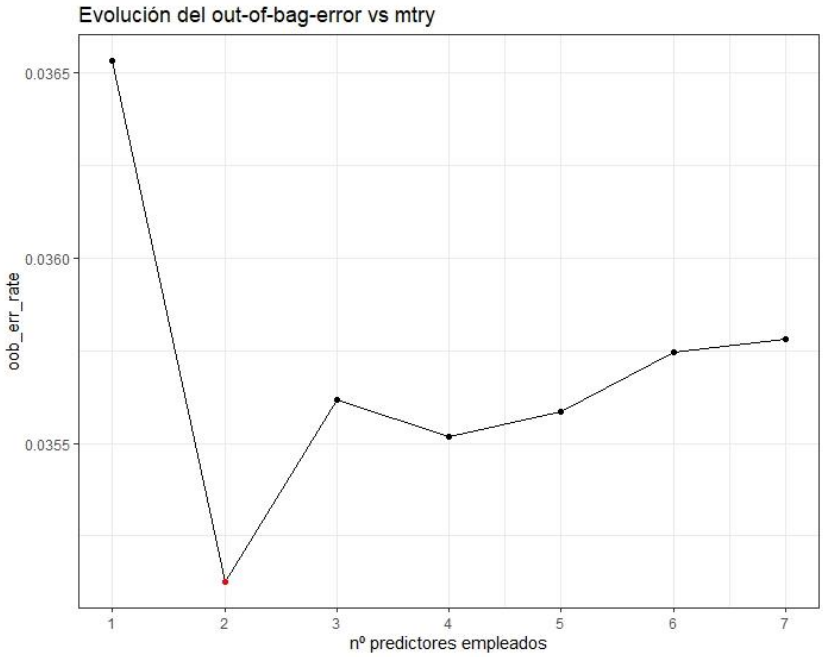


Ilustración 25:Cálculo parámetros número de predictores Random Forest.

Parámetros SVM lineal.

Kernel	Coste	Kappa train	Accuracy train	Sensibility train	Sepecificity train	Kappa test	Accuracy test	Sensibility test	Sepecificity test
linear	10	35,71	96,35	99,9	23,11	29,81	96,28	99,95	18,41
linear	0,1	35,59	96,35	99,9	22,98	29,17	96,27	99,96	17,9
linear	0,01	34,38	96,33	99,94	21,86	28,08	96,23	99,96	17,14
linear	0,001	12,9	95,71	100	7,2	10,22	95,75	100	5,63
linear	25	0	95,38	100	0	0	95,5	100	0
linear	50	0	95,38	100	0	0	95,5	100	0
linear	75	0	95,38	100	0	0	95,5	100	0
linear	100	0	95,38	100	0	0	95,5	100	0
linear	150	0	95,38	100	0	0	95,5	100	0

Tabla 46: Parámetros SVM lineal.

Parámetros SVM Polinómico.

Kernel	Coste	Degree	Kappa train	Accuracy train	Sensibility train	Sepecificity train	Kappa test	Accuracy test	Sensibility test	Sepecificity test
Polinómico	25	4	59,13	97,35	99,96	43,48	34,43	95,86	99,16	26,09
Polinómico	10	7	64,07	97,6	99,99	48,45	33,36	95,66	98,93	26,34
Polinómico	50	4	62,46	97,51	99,96	46,96	33,07	95,65	98,93	26,09
Polinómico	50	5	66,88	97,75	99,99	51,55	32,42	95,46	98,71	26,6
Polinómico	25	5	64,87	97,64	99,99	49,32	32,74	95,55	98,82	26,34
Polinómico	10	4	55,38	97,19	99,98	39,63	34,6	96,06	99,42	24,81
Polinómico	10	5	61,22	97,46	99,98	45,47	33,2	95,7	99	25,83
Polinómico	75	5	68,61	97,84	99,99	53,54	31,96	95,38	98,62	26,6
Polinómico	100	3	51,88	97,01	99,93	36,65	33,92	95,92	99,26	25,06
Polinómico	100	5	69,04	97,86	99,99	54,04	31,83	95,36	98,6	26,6
Polinómico	25	7	67,26	97,77	99,98	52,05	31,62	95,36	98,61	26,34
Polinómico	75	4	64,13	97,6	99,97	48,7	32,04	95,51	98,79	25,83
Polinómico	1	3	44,86	96,71	99,92	30,43	35,13	96,34	99,78	23,27
Polinómico	25	6	66,66	97,74	99,99	51,3	31,37	95,43	98,72	25,58
Polinómico	100	4	64,92	97,64	99,97	49,57	31,37	95,43	98,72	25,58
Polinómico	50	6	68,77	97,85	99,98	53,79	30,82	95,29	98,56	25,83
Polinómico	1	8	58,25	97,32	99,99	42,36	32,72	95,84	99,22	24,3
Polinómico	10	6	62,57	97,53	99,99	46,83	31,98	95,65	98,99	24,81
Polinómico	100	7	71,24	97,98	99,99	56,65	30,08	95,07	98,31	26,34
Polinómico	10	3	47,7	96,82	99,92	32,92	33,75	96,14	99,58	23,27
Polinómico	1	7	57,51	97,29	99,99	41,61	32,85	95,93	99,34	23,79
Polinómico	50	7	69,89	97,91	99,99	55,03	30,2	95,17	98,44	25,83
Polinómico	0,1	3	41,62	96,58	99,92	27,7	34,48	96,35	99,83	22,51
Polinómico	1	5	51,04	97	99,98	35,53	33,66	96,2	99,66	22,76
Polinómico	10	8	64,98	97,65	99,99	49,44	31,08	95,53	98,88	24,55

Polinómico	75	7	70,41	97,94	99,99	55,65	29,39	95,06	98,34	25,58
Polinómico	75	6	69,47	97,89	99,99	54,53	29,32	95,13	98,43	25,06
Polinómico	1	6	54,84	97,17	99,99	39,01	31,89	95,97	99,43	22,51
Polinómico	0,1	7	49,6	96,95	100	34,04	32,27	96,18	99,7	21,48
Polinómico	100	8	71,45	98	99,99	56,89	27,44	94,75	98,03	25,06
Polinómico	100	6	70,1	97,92	99,99	55,28	28,2	95,04	98,37	24,3
Polinómico	0,1	5	45,68	96,78	99,99	30,68	32,32	96,26	99,81	20,97
Polinómico	0,1	8	51,79	97,04	100	36,02	31,35	96,04	99,55	21,48
Polinómico	1	4	48,84	96,9	99,98	33,54	31,63	96,12	99,65	21,23
Polinómico	0,1	6	47,08	96,85	100	31,8	30,94	96,13	99,7	20,46
Polinómico	0,01	5	38,47	96,51	99,99	24,72	31,19	96,24	99,84	19,95
Polinómico	0,01	8	45,2	96,77	100	30,19	30,31	96,15	99,76	19,69
Polinómico	0,01	7	42,03	96,65	99,99	27,58	30,27	96,19	99,81	19,44
Polinómico	0,1	4	41,83	96,63	99,99	27,45	30,2	96,18	99,79	19,44
Polinómico	0,01	3	35,49	96,38	99,96	22,61	30,55	96,27	99,9	19,18
Polinómico	0,01	4	34,75	96,36	99,97	21,99	28,2	96,16	99,87	17,65
Polinómico	0,01	6	38,99	96,54	100	25,09	27,4	96,08	99,79	17,39

Tabla 47: Parámetros SVM Polinómico.

Parámetros SVM Radial.

Kernel	Coste	Gamma	Kappa train	Accuracy train	Sensibility train	Sepecificity train	Kappa test	Accuracy test	Sensibility test	Sepecificity test
Radial	75	0,1	70,92	97,96	99,98	56,4	34,35	95,78	99,05	26,6
Radial	25	0,1	65,26	97,66	99,97	49,94	35,27	96,03	99,34	25,83
Radial	25	0,05	53,64	97,1	99,97	38,01	36,2	96,29	99,66	24,81
Radial	100	0,1	71,84	98,01	99,98	57,52	33,64	95,67	98,93	26,6
Radial	150	0,05	65,43	97,67	99,98	50,06	34,28	95,84	99,13	26,09
Radial	10	0,05	50,07	96,94	99,96	34,78	36,31	96,34	99,72	24,55
Radial	10	0,1	58,39	97,32	99,98	42,61	35,79	96,23	99,6	24,81
Radial	50	0,05	57,52	97,28	99,98	41,74	35,38	96,14	99,49	25,06
Radial	50	0,1	69,77	97,9	99,98	55,03	34,21	95,86	99,17	25,83
Radial	100	0,05	63,21	97,55	99,97	47,7	34,44	95,93	99,25	25,58
Radial	150	0,1	73,52	98,11	99,98	59,5	32,76	95,48	98,72	26,85
Radial	75	0,05	60,69	97,43	99,98	44,97	34,44	96	99,35	25,06
Radial	50	0,01	44,09	96,67	99,92	29,81	35,29	96,36	99,81	23,27
Radial	75	0,01	44,52	96,69	99,92	30,19	34,39	96,3	99,77	22,76
Radial	100	0,01	44,95	96,72	99,93	30,43	34,15	96,27	99,73	22,76
Radial	150	0,01	46,1	96,77	99,93	31,43	33,74	96,24	99,72	22,51
Radial	10	0,01	42,17	96,6	99,92	28,2	33,9	96,34	99,84	21,99
Radial	25	0,01	42,71	96,63	99,93	28,57	33,9	96,34	99,84	21,99
Radial	10	0,01	42,17	96,6	99,92	28,2	33,9	96,34	99,84	21,99
Radial	1	0,1	44,95	96,73	99,95	30,31	32,81	96,29	99,83	21,23
Radial	1	0,01	38,7	96,48	99,94	25,22	32,88	96,34	99,89	20,97
Radial	0,1	0,01	35,39	96,35	99,92	22,73	31,17	96,28	99,89	19,69
Radial	10	0,5	80,47	98,54	99,99	68,45	25,52	95,62	99,28	18,16
Radial	25	0,5	85,17	98,85	99,99	75,16	23,58	95,2	98,82	18,41
Radial	75	0,5	89,36	99,14	100	81,49	21,35	94,53	98,08	19,18

Radial	100	0,5	90,15	99,2	99,99	82,86	20,88	94,41	97,96	19,18
Radial	50	0,5	88,04	99,05	99,99	79,63	21,25	94,68	98,28	18,41
Radial	150	0,5	90,92	99,26	99,99	84,1	20,17	94,23	97,77	19,18
Radial	0,1	0,1	32,26	96,26	99,95	20,25	24,47	96,08	99,92	14,83
Radial	100	1	94,25	99,52	99,99	89,69	16,89	94,47	98,23	14,83
Radial	10	1	87,79	99,03	99,99	79,13	17,73	95,24	99,13	12,79
Radial	1	1	62,57	97,53	99,99	46,83	18,22	95,96	99,99	10,49
Radial	100	10	99,21	99,93	100	98,51	2,7	94,95	99,32	2,3
Radial	10	10	98,35	99,86	100	96,89	2,66	95,12	99,51	2,05
Radial	0,1	1	5,99	95,53	100	3,23	2,89	95,57	100	1,53
Radial	1	10	88,01	99,05	100	79,38	1,43	95,52	99,99	0,77
Radial	0,01	0,01	0	95,38	100	0	0	95,5	100	0
Radial	0,01	0,1	0	95,38	100	0	0	95,5	100	0
Radial	0,01	100	0	95,38	100	0	0	95,5	100	0
Radial	0,01	10	0	95,38	100	0	0	95,5	100	0
Radial	0,01	1	0	95,38	100	0	0	95,5	100	0
Radial	0,1	100	0	95,38	100	0	0	95,5	100	0
Radial	0,1	10	0	95,38	100	0	0	95,5	100	0

Tabla 48: Parámetros SVM Radial.

Código en R

```
##############################################################################
##############################################################################
##                                                                          ##
##                    TRABAJO FINAL DE MÁSTER                               ##
##                                                                          ##
## Título: Modelización de la probabilidad de no renovación para una        ##
##          cartera de salud a partir de Logit, Random Forest y SVM.        ##
##                                                                          ##
## Autor: Francisco Folk Delgado                                            ##
##                                                                          ##
##############################################################################
##############################################################################

##############################################################################
##############################################################################
##                            SCRIPT 1                                      ##
## 1. Transformación de la base de datos                                    ##
##############################################################################
##############################################################################

##############################################################################
# 1.1. Carga de librerias
##############################################################################

if(!require(openxlsx)){
  install.packages(openxlsx)
  library(openxlsx)
}

if(!require(dplyr)){
  install.packages(dplyr)
  library(dplyr)
}

if(!require(ggplot2)){
  install.packages(ggplot2)
  library(ggplot2)
}

##############################################################################
# 1.2. Carga de datos
##############################################################################

load('.../BBDD TFM.rdata')

##############################################################################
# 1.3. Tratamiento de datos
##############################################################################

BBDD <- BBDD %>%
        mutate(NUMERO_DE_POLIZAS_NO_SALUD_dicotomica =
                     as.factor(ifelse(NUMERO_DE_POLIZAS_NO_SALUD == 0,
                               'Sin polizas vigentes distintas a salud',
                               'Con polizas vigentes distintas a salud')),
               BAJAS_ANTERIORES_DISTINTAS_SALUD_dicotomica =
                     as.factor(ifelse(BAJAS_ANTERIORES_DISTINTAS_SALUD == 0,
                               'Sin bajas anteriores',
                               'Con Bajas anteriores')),
```

```
                BAJAS_ANTERIORES_SALUD_dicotomica =
                     as.factor(ifelse(BAJAS_ANTERIORES_SALUD == 0,
                               'Sin bajas anteriores',
                               'Con bajas anteriores')))
summary(BBDD)
str(BBDD)

################################################################################
# 1.4. Entropia
################################################################################

Estado_table <- table(BBDD$ESTADO)

P_Estado <- Estado_table/sum(Estado_table)

Entropia_Estado <- (-sum(log2(P_Estado)*P_Estado))

for (i in 2:length(BBDD)) {

  BBDD_Entropia <- table(BBDD[, c(1, i)])

  for (j in 1:ncol(BBDD_Entropia)) {

    BBDD_Entropia_Individual <- BBDD_Entropia[, j]

    Probabilidad_individual <-
                  BBDD_Entropia_Individual/sum(BBDD_Entropia_Individual)

    Entropia_individual <-
            (-sum(log2(Probabilidad_individual)*Probabilidad_individual))

    if(is.na(Entropia_individual)){

      Entropia_individual <- 0

    } else {

      Entropia_individual <- Entropia_individual

    }

    Probabilidad_Colectivo <- BBDD_Entropia/sum(BBDD_Entropia)

    Entropia_Variable <- Entropia_individual * Probabilidad_Colectivo[j]

    if (exists('Entropia_Colectivo')) {

      Entropia_Colectivo <- c(Entropia_Colectivo, Entropia_Variable)

    } else {

      Entropia_Colectivo <- Entropia_Variable

    }

    Entropia_Colectivo <- sum(Entropia_Colectivo)
  }

  if (exists('Entropia_Global')) {
```

```
    Entropia_Global <- c(Entropia_Global, Entropia_Colectivo)

  } else {

    Entropia_Global <- Entropia_Colectivo

  }
  rm(Entropia_Colectivo)

}

GI_1 <- data.frame(Variable = colnames(BBDD)[-1],
                   GI = Entropia_Estado - Entropia_Global)

GI <- cbind(GI_1, Entropia_Global)  %>%
      arrange(desc(GI)) %>%
      mutate(VARIABLE = Variable,
             ENTROPIA = round(Entropia_Global, 4),
             GI = round(GI, 4)) %>%
      select(VARIABLE, ENTROPIA, GI)

GI_ESTADO <- c('ESTADO', round(Entropia_Estado, 4), '-')

GI_Final <- rbind(GI_ESTADO, GI)

#############################################################################
# 1.5. Colinealidad entre variables dependientes
#############################################################################
str(BBDD)

BBDD_colinealidad <- BBDD[,-c(1, 6, 12)]

BBDD_colinealidad <- BBDD_colinealidad[,sort(colnames(BBDD_colinealidad))]

str(BBDD_colinealidad)

Correlacion <- cor(BBDD_colinealidad)

VIF <- round(1/(1-cor(BBDD_colinealidad)^2),1)

#############################################################################
#############################################################################
##                           SCRIPT 2                                      ##
## 2. Base de datos final y partición                                      ##
#############################################################################
#############################################################################

#############################################################################
# 2.1. Carga de librerias
#############################################################################

if(!require(openxlsx)){
  install.packages("openxlsx")
  library(openxlsx)
}

if(!require(dplyr)){
  install.packages("dplyr")
  library(dplyr)
}
```

```
if(!require(recipes)){
  install.packages("recipes")
  library(recipes)
}

#############################################################################
# 2.2. Carga de datos
#############################################################################

load('.../BBDD TFM.rdata')

#############################################################################
# 2.3. Transformación de variables
#############################################################################

BBDD <- BBDD %>%
        mutate(NUMERO_DE_POLIZAS_NO_SALUD =
                     as.factor(ifelse(NUMERO_DE_POLIZAS_NO_SALUD == 0,
                                      'Sin polizas vigentes distintas a salud',
                                      'Con polizas vigentes distintas a salud')),
               BAJAS_ANTERIORES_DISTINTAS_SALUD =
                      as.factor(ifelse(BAJAS_ANTERIORES_DISTINTAS_SALUD ==
0,
                                       'Sin bajas anteriores',
                                       'Con Bajas anteriores')),
               BAJAS_ANTERIORES_SALUD =
                       as.factor(ifelse(BAJAS_ANTERIORES_SALUD == 0,
                                        'Sin bajas anteriores',
                                        'Con bajas anteriores')))

summary(BBDD)
str(BBDD)

#############################################################################
# 2.4. Estandarización de variables
#############################################################################

BBDD_2 <- BBDD %>%
          mutate(FAMILIA_HIJOS_MAYORES = ifelse(FAMILIA == 'Familia con
hijos mayores', 1, 0),
                 FAMILIA_HIJOS_MENORES = ifelse(FAMILIA == 'Familia con
hijos menores', 1, 0),
                 FAMILIA_OTROS = ifelse(FAMILIA == 'Familia otros', 1, 0),
                 FAMILIA_PAREJA_SIN_HIJOS = ifelse(FAMILIA == 'Pareja sin
hijos', 1, 0),
                 MENSUAL = ifelse(CODIGO_PAGO == 'Mensual', 1, 0),
                 ANUAL = ifelse(CODIGO_PAGO == 'Anual', 1, 0),
                 TRIMESTRAL = ifelse(CODIGO_PAGO == 'Trimestral', 1, 0))
%>%
          select(-c(FAMILIA, CODIGO_PAGO))

summary(BBDD_2)
str(BBDD_2)

BBDD_Estandar <- data.frame(ESTADO = as.factor(BBDD[,1]))

for (i in 2:length(BBDD_2)) {

  Estandar_aux <- (BBDD_2[,i] - mean(BBDD_2[,i]))/sd(BBDD_2[,i])
```

```
  BBDD_Estandar <- cbind(BBDD_Estandar, Estandar_aux)

}

colnames(BBDD_Estandar) <- colnames(BBDD_2)
summary(BBDD_Estandar)
str(BBDD_Estandar)

BBDD <- BBDD_Estandar
#############################################################################
# 2.5. Partición
#############################################################################

#Particion de los datos en train y test.

set.seed(9)

Index <- sample(2, nrow(BBDD), replace = T, prob = c(0.7, 0.3))

BBDD_train <- BBDD[which(Index == 1), ]
BBDD_test <- BBDD[which(Index == 2), ]

summary(BBDD_train)
summary(BBDD_test)

#############################################################################
#############################################################################
##                            SCRIPT 3                                     ##
## 3. Logit                                                                ##
#############################################################################
#############################################################################

#############################################################################
# 3.1. Carga de librerias
#############################################################################

if(!require(caret)){
  install.packages("caret")
  library(caret)
}

if(!require(tidyr)){
  install.packages("tidyr")
  library(tidyr)
}

if(!require(openxlsx)){
  install.packages("openxlsx")
  library(openxlsx)
}

if(!require(dplyr)){
  install.packages("dplyr")
  library(dplyr)
}

#############################################################################
# 3.2. Importancia de las variables
#############################################################################
```

```
# Todas las variables

BBDD$ESTADO <- as.numeric(as.character(BBDD$ESTADO))

str(BBDD)
summary(BBDD)

Logit <- glm(formula = ESTADO ~ .,
             data = BBDD,
             family = 'binomial')

summary(Logit)

Predict <- predict(Logit, BBDD, type = 'response')

table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0, 1,0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(Predict > co[i], 1, 0)))
  confumat <- confusionMatrix(data=predict_labels,
reference=as.factor(BBDD$ESTADO))

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
  arrange(desc(Kappa * Specificity))

best_cutoff <- table [1, 1]

BBDD$ESTADO <-  as.factor(BBDD$ESTADO)
Predict <- as.factor(ifelse(Predict > best_cutoff, 1, 0))

confusionMatrix(BBDD$ESTADO, Predict)

# Selección de variables

BBDD$ESTADO <- as.numeric(as.character(BBDD$ESTADO))

BBDD <- BBDD %>%
        select(-c(ANTIGUEDAD_POLIZA, NUMERO_DE_POLIZAS_NO_SALUD,
                  NASEG_POLIZA, EDAD_MEDIA_POLIZA, USO, SINIESTRALIDAD,
                  PRIMA_ACTUAL, VARIACION_PORCENTUAL))
str(BBDD)
summary(BBDD)

Logit <- glm(formula = ESTADO ~ .,
             data = BBDD,
```

```
             family = 'binomial')

summary(Logit)

Predict <- predict(Logit, BBDD, type = 'response')


table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0, 1, 0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(Predict > co[i], 1, 0)))
  confumat <- confusionMatrix(data=predict_labels,
reference=as.factor(BBDD$ESTADO))

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
         arrange(desc(Kappa * Specificity))

best_cutoff <- table [1, 1]

BBDD$ESTADO <-  as.factor(BBDD$ESTADO)
Predict <- as.factor(ifelse(Predict > best_cutoff, 1, 0))

confusionMatrix(BBDD$ESTADO, Predict)

#############################################################################
# 3.3. Prueba pesos
#############################################################################

CM_Train_list <- list()
CM_Test_list <- list()

w1 <- seq(0.05, 1, 0.05)
w0 <- 1 - w1
w0[length(w0)] <- 1

w <- rep(NA, length(BBDD_train))

for (j in 1:length(w0)) {

  w[which(BBDD_train$ESTADO == 1)] <- w1[j]
  w[which(BBDD_train$ESTADO == 0)] <- w0[j]

  summary(w)

  BBDD_train$ESTADO <- as.integer(as.character(BBDD_train$ESTADO))
```

```
  BBDD_test$ESTADO <-  as.integer(as.character(BBDD_test$ESTADO))

  str(BBDD_train)
  summary(BBDD_train)

  Logit <- glm(formula = ESTADO ~ .,
               data = BBDD_train,
               family = 'binomial',
               weights = w)

  Predict <- predict(Logit, BBDD_test, type = 'response')

  # Cut off

  table <- data.frame(cut.off=double(),
                      Accuracy=numeric(),
                      Sensitivity=numeric(),
                      Specificity=numeric(),
                      Kappa=numeric(),
                      stringsAsFactors = FALSE)

  co <- seq(0.05, 1, 0.05)

  for (i in 1:length(co)){

    predict_labels <- as.factor(as.character(ifelse(Predict > co[i], 1,
0)))
    confumat <- confusionMatrix(data=predict_labels,
reference=as.factor(BBDD_test$ESTADO))

    table[i,1] <- co[i]
    table[i,2] <- round(confumat$overall[1]*100,2)
    table[i,3] <- round(confumat$byClass[1]*100,2)
    table[i,4] <- round(confumat$byClass[2]*100,2)
    table[i,5] <- round(confumat$overall[2]*100,2)
  }

  table <- table %>%
           arrange(desc(Kappa * Specificity))

  best_cutoff <- table [1, 1]

  Predict_train <- as.factor(ifelse(Logit$fitted.values > best_cutoff, 1,
0))
  Predict_test <- as.factor(ifelse(Predict > best_cutoff, 1, 0))

  BBDD_train$ESTADO <- as.factor(BBDD_train$ESTADO)
  BBDD_test$ESTADO <-  as.factor(BBDD_test$ESTADO)

  CM_Train <- confusionMatrix(BBDD_train$ESTADO, Predict_train)
  CM_Test <- confusionMatrix(BBDD_test$ESTADO, Predict_test)

  CM_Train_list[[j]] <- CM_Train
  CM_Test_list[[j]] <- CM_Test

  print(paste0('Cálculo ', j, ' de ', length(w0), ' calculado con éxito'))

}

##############################################################################
```

```
# 3.4 Modelo elegido
##############################################################################

w <- rep(NA, length(BBDD_train))

w[which(BBDD_train$ESTADO == 1)] <- 0.05
w[which(BBDD_train$ESTADO == 0)] <- 0.95

BBDD_train$ESTADO <- as.integer(as.character(BBDD_train$ESTADO))
BBDD_test$ESTADO <-  as.integer(as.character(BBDD_test$ESTADO))

str(BBDD_train)
summary(BBDD_train)

Logit <- glm(formula = ESTADO ~ .,
             data = BBDD_train,
             family = 'binomial',
             weights = w)

Predict <- predict(Logit, BBDD_test, type = 'response')

# Cut off


table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0.05, 1, 0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(Predict > co[i], 1, 0)))
  confumat <- confusionMatrix(data=predict_labels,
reference=as.factor(BBDD_test$ESTADO))

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}


table <- table %>%
         arrange(desc(Kappa * Specificity));table

best_cutoff <- table [1, 1];best_cutoff

Predict_train <- as.factor(ifelse(Logit$fitted.values > best_cutoff, 1, 0))
Predict_test <- as.factor(ifelse(Predict > best_cutoff, 1, 0))
BBDD_train$ESTADO <- as.factor(BBDD_train$ESTADO)
BBDD_test$ESTADO <-  as.factor(BBDD_test$ESTADO)

summary(Logit)
confusionMatrix(BBDD_train$ESTADO, Predict_train)
confusionMatrix(BBDD_test$ESTADO, Predict_test)
```

```
################################################################################
# 3.5. Cross Validation
################################################################################

BBDD$ESTADO <- as.integer(as.character(BBDD$ESTADO))

BBDD_Resultado <- BBDD
BBDD_Resultado$pred <- NA

K <- 10

set.seed(9)
Index <- sample(K, nrow(BBDD), replace = T, prob = rep(1/K, K))

for(i in 1:K){

  testData <- BBDD[Index == i, ]
  trainData <- BBDD[-(Index == i), ]

  w <- rep(NA, length(trainData))

  w[which(trainData$ESTADO == 1)] <- 0.05
  w[which(trainData$ESTADO == 0)] <- 0.95

  summary(w)

  Logit <- glm(formula = ESTADO ~ .,
               data = trainData,
               family = 'binomial',
               weights = w)

  pred <- predict(Logit, testData, type = 'response')

  BBDD_Resultado[Index == i, ]$pred <- pred

}

#Matriz de confusión


# Cut off


table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0.05, 1, 0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(BBDD_Resultado$pred >
co[i], 1, 0)))
```

```
  confumat <- confusionMatrix(data=predict_labels,
reference=as.factor(BBDD_Resultado$ESTADO))

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
         arrange(desc(Kappa * Specificity))

best_cutoff <- table [1, 1]

BBDD_Resultado$pred_lab <- ifelse(BBDD_Resultado$pred > best_cutoff, 1, 0)
BBDD_Resultado$pred_lab <- as.factor(as.character(BBDD_Resultado$pred_lab))
BBDD_Resultado$ESTADO <- as.factor(BBDD_Resultado$ESTADO)

confumat <- confusionMatrix(data = BBDD_Resultado$ESTADO, reference =
BBDD_Resultado$pred_lab)
confumat

##############################################################################
##############################################################################
##                            SCRIPT 4                                      ##
## 4. Random Forest                                                         ##
##############################################################################
##############################################################################
##############################################################################
# 4.1. Carga de librerias
##############################################################################

if(!require(randomForest)){
  install.packages("randomForest")
  library(randomForest)
}

if(!require(caret)){
  install.packages("caret")
  library(caret)
}

if(!require(tidyr)){
  install.packages("tidyr")
  library(tidyr)
}

if(!require(ranger)){
  install.packages("ranger")
  library(ranger)
}

if(!require(openxlsx)){
  install.packages("openxlsx")
  library(openxlsx)
}

if(!require(dplyr)){
  install.packages("dplyr")
```

```
  library(dplyr)
}
#############################################################################
# 4.2. Importancia de las variables
#############################################################################

# Todas las variables

RF <- randomForest(x = BBDD[, -1],
                   y = BBDD[, 1],
                   ntree = 100,
                   importance = TRUE)

Predict <- predict(RF, BBDD, 'prob')

table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0,1,0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(Predict[,2] > co[i], 1,
0)))
  confumat <- confusionMatrix(data=predict_labels, reference=BBDD$ESTADO)

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
         arrange(desc(Kappa * Specificity))

best_cutoff <- table [1, 1]

Predict <- as.factor(ifelse(Predict[,2] > best_cutoff, 1, 0))

confusionMatrix(Predict, BBDD$ESTADO)

RF$importance

Plot_importancia <- data.frame(RF$importance) %>%
  mutate(Variable = as.factor(rownames(RF$importance))) %>%
  arrange(desc(MeanDecreaseAccuracy)) %>%
  select(Variable, MeanDecreaseAccuracy, MeanDecreaseGini)

ggplot(data = Plot_importancia,
       aes(x = MeanDecreaseAccuracy,
           y = reorder(Variable, MeanDecreaseAccuracy),
           fill = MeanDecreaseAccuracy)) +
  geom_col() +
  ggtitle('Importancia Random Forest') +
  scale_fill_continuous(guide='none') +
```

```
  labs(x= 'Mean Decrease Accuracy', y = 'Variables')+
  theme_bw() +
  theme(plot.title = element_text(size = 20))

# Selección de variables

BBDD <- BBDD %>%
        select(ESTADO, VARIACION_PORCENTUAL, PRIMA_ACTUAL, SINIESTRALIDAD,
               USO, EDAD_MEDIA_POLIZA, NASEG_POLIZA, FAMILIA,
ANTIGUEDAD_POLIZA)

RF <- randomForest(x = BBDD[, -1],
                   y = BBDD[, 1],
                   ntree = 100,
                   importance = TRUE)

Predict <- predict(RF, BBDD, 'prob')

table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0,1,0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(Predict[,2] > co[i], 1,
0)))
  confumat <- confusionMatrix(data=predict_labels, reference=BBDD$ESTADO)

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
  arrange(desc(Kappa * Specificity))

best_cutoff <- table [1, 1]

Predict <- as.factor(ifelse(Predict[,2] > best_cutoff, 1, 0))

confusionMatrix(Predict, BBDD$ESTADO)

RF$importance

Plot_importancia <- data.frame(RF$importance) %>%
                    mutate(Variable = as.factor(rownames(RF$importance)))
%>%
                    arrange(desc(MeanDecreaseAccuracy)) %>%
                    select(Variable, MeanDecreaseAccuracy,
MeanDecreaseGini)

ggplot(data = Plot_importancia,
       aes(x = MeanDecreaseAccuracy,
```

```
           y = reorder(Variable, MeanDecreaseAccuracy),
           fill = MeanDecreaseAccuracy)) +
  geom_col() +
  ggtitle('Importancia Random Forest') +
  scale_fill_continuous(guide='none') +
  labs(x= 'Mean Decrease Accuracy', y = 'Variables')+
  theme_bw() +
  theme(plot.title = element_text(size = 20),)


##############################################################################
# 4.3. Parametros Random Forest
##############################################################################

#Número de predictores

BBDD_train_tbl <- as_tibble(BBDD_train)

tuning_rf_mtry <- function(df, y, ntree){
  # Esta función devuelve el out-of-bag clasification error de un modelo
RandomForest
  # en función del número de predictores evaluados (mtry)

  # Argumentos:
  #   df = data frame con los predictores y variable respuesta
  #   y  = nombre de la variable respuesta
  #   ntree = número de árboles creados en el modelo randomForest

  max_predictores <- ncol(df) - 1
  n_predictores   <- rep(NA, max_predictores)
  oob_err_rate    <- rep(NA, max_predictores)

  for (i in 1:max_predictores) {

    set.seed(321)

    modelo_rf <- randomForest(x = df,
                              y = y ,
                              mtry = i,
                              ntree = ntree)

    n_predictores[i] <- i

    oob_err_rate[i] <- tail(modelo_rf$err.rate[, 1], n = 1)

    print(paste0("El cálculo de número de predictores para el predictor ",
i, " de ",  max_predictores, " se ha realizado con éxito. Calculado un ",
round(i*100/max_predictores, 2), " %."))
  }

  results <- data_frame(n_predictores, oob_err_rate) %>%
arrange(oob_err_rate)

  return(results)
}

hiperparametro_mtry <- tuning_rf_mtry(df = BBDD_train_tbl[, -1], y =
BBDD_train[, 1], ntree = 500)
hiperparametro_mtry
```

```
mtry_plot <- ggplot(data = hiperparametro_mtry, aes(x = n_predictores, y =
oob_err_rate)) +
  scale_x_continuous(breaks = hiperparametro_mtry$n_predictores) +
  geom_line() +
  geom_point() +
  geom_point(data = hiperparametro_mtry %>% arrange(oob_err_rate) %>%
head(1),
             color = "red") +
  labs(title = "Evolución del out-of-bag-error vs mtry",
       x = "nº predictores empleados") +
  theme_bw()

best.mtry <- as.integer(hiperparametro_mtry[1,1])

#Número de nodos

tuning_rf_nodesize <- function(df, y, size = NULL, ntree, mtry){
  # Esta función devuelve el out-of-bag clasification error de un modelo
RandomForest
  # en función del tamaño mínimo de los nodos terminales (nodesize).

  # Argumentos:
  #   df = data frame con los predictores y variable respuesta
  #   y  = nombre de la variable respuesta
  #   sizes = tamaños evaluados
  #   ntree = número de árboles creados en el modelo randomForest

  if (is.null(size)){
    size <- seq(from = 1, to = nrow(df), by = 5)
  }

  oob_err_rate <- rep(NA, length(size))

  for (i in 1:size) {

    set.seed(321)

    modelo_rf <- randomForest(x = df,
                              y = y ,
                              mtry = mtry,
                              ntree = ntree,
                              nodesize = i)

    oob_err_rate[i] <- tail(modelo_rf$err.rate[, 1], n = 1)

    print(paste0("El cálculo de número de nodos para el valor ", i, " del
tamaño total de ", size," se ha realizado con éxito. Calculado un ",
round(i*100/size, 2), " %." ))
  }

  results <- data_frame(Tamaño = 1:size, oob_err_rate )%>%
arrange(oob_err_rate)
  return(results)

}

hiperparametro_nodesize <- tuning_rf_nodesize(df = BBDD_train_tbl[, -1], y
= BBDD_train[, 1], size = 20 , ntree = 500, mtry = best.mtry)
```

```
Nodesplot <-  ggplot(data = hiperparametro_nodesize, aes(x = Tamaño, y =
oob_err_rate)) +
  scale_x_continuous(breaks = hiperparametro_nodesize$Tamaño) +
  geom_line() +
  geom_point() +
  geom_point(data = hiperparametro_nodesize %>% arrange(oob_err_rate) %>%
head(1),
             color = "red") +
  labs(title = "Evolución del out-of-bag-error vs nodesize",
       x = "n° observaciones en nodos terminales") +
  theme_bw()

best.nodesize <- as.integer(hiperparametro_nodesize[1,1])

#Número de Arboles

modelo_randomforest <- randomForest(x = BBDD_train_tbl[, -1],
                                    y =  BBDD_train[, 1],
                                    ntree = 500,
                                    importance = TRUE,
                                    mtry = best.mtry,
                                    nodesize = best.nodesize)


oob_err_rate <- data.frame(arboles =
seq_along(modelo_randomforest$err.rate[, 1]), oob_err_rate =
modelo_randomforest$err.rate[, 1])

Randomforestplot <- ggplot(data = oob_err_rate, aes(x = arboles, y =
oob_err_rate )) +
  geom_line() +
  labs(title = "Evolución del out-of-bag-error vs número árboles",
       x = "n° árboles") +
  theme_bw()

Hiperparametro_ntree <- data.frame(arboles =
seq_along(modelo_randomforest$err.rate[, 1]), oob_err_rate =
modelo_randomforest$err.rate[, 1]) %>% arrange(oob_err_rate)

best.ntree <- Hiperparametro_ntree[1, 1]


Hiperparametros <- data.frame("Núm. Nodo" = best.nodesize,
                              "Núm. Predictores" = best.mtry,
                              "Núm. Árboles" = best.ntree)

##############################################################################
# 4.4. Prueba pesos
##############################################################################

CM_Train_list <- list()
CM_Test_list <- list()

w1 <- seq(0.05, 1, 0.05)
w0 <- 1 - w1
w0[length(w0)] <- 1

for (j in 1:length(w0)) {

  w1_RF <- w1[j]
```

```
  w0_RF <- w0[j]

  RF <- randomForest(x = BBDD_train[, -1],
                     y = BBDD_train[, 1],
                     ntree = 150,
                     mtry = 2,
                     nodesize = 1,
                     classwt = c('0' = w0_RF, '1' = w1_RF))

  Predict <- predict(RF, BBDD_test, 'prob')

  table <- data.frame(cut.off=double(),
                      Accuracy=numeric(),
                      Sensitivity=numeric(),
                      Specificity=numeric(),
                      Kappa=numeric(),
                      stringsAsFactors = FALSE)

  co <- seq(0,1,0.05)

  for (i in 1:length(co)){

    predict_labels <- as.factor(as.character(ifelse(Predict[,2] > co[i], 1,
0)))
    confumat <- confusionMatrix(data=predict_labels,
reference=BBDD_test$ESTADO)

    table[i,1] <- co[i]
    table[i,2] <- round(confumat$overall[1]*100,2)
    table[i,3] <- round(confumat$byClass[1]*100,2)
    table[i,4] <- round(confumat$byClass[2]*100,2)
    table[i,5] <- round(confumat$overall[2]*100,2)
  }

  table <- table %>%
           arrange(desc(Kappa));table

  best_cutoff <- table[1, 1]

  Predict <- predict(RF, BBDD_test, 'prob')

  Predict_train <- as.factor(RF$predicted)
  Predict_test <- as.factor(ifelse(Predict[,2] > best_cutoff, 1, 0))


  CM_Train <- confusionMatrix(BBDD_train$ESTADO, RF$predicted)
  CM_Test <- confusionMatrix(BBDD_test$ESTADO, Predict_test)

  CM_Train_list[[j]] <- CM_Train
  CM_Test_list[[j]] <- CM_Test

  print(paste0('Cálculo ', j, ' de ', length(w0), ' calculado con éxito'))

}
#############################################################################
# 4.5 Modelo elegido
#############################################################################

RF <- randomForest(x = BBDD_train[, -1],
                   y = BBDD_train[, 1],
```

```
                    mtry = 2,
                    nodesize = 1,
                    ntree = 150)

Predict <- predict(RF, BBDD_test, 'prob')

table <- data.frame(cut.off=double(),
                     Accuracy=numeric(),
                     Sensitivity=numeric(),
                     Specificity=numeric(),
                     Kappa=numeric(),
                     stringsAsFactors = FALSE)

co <- seq(0,1,0.01)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(Predict[,2] > co[i], 1,
0)))
  confumat <- confusionMatrix(data=predict_labels,
reference=BBDD_test$ESTADO)

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
         arrange(desc(Kappa*Specificity))

best_cutoff <- table[1, 1]

Predict <- predict(RF, BBDD_test, 'prob')
Predict_test <- as.factor(ifelse(Predict[,2] > best_cutoff, 1, 0))

confusionMatrix(BBDD_train$ESTADO, RF$predicted)
confusionMatrix(BBDD_test$ESTADO, Predict_test)

##############################################################################
# 4.6. Cross Validation
##############################################################################

BBDD_Resultado <- BBDD
BBDD_Resultado$pred <- NA

K <- 10

set.seed(9)
Index <- sample(K, nrow(BBDD), replace = T, prob = rep(1/K, K))

for(j in 1:K){

  testData <- BBDD[Index == j, ]
  trainData <- BBDD[-(Index == j), ]

  forest <- randomForest(x = trainData[, -1],
                         y = trainData[, 1],
                         ntree = 150,
```

```
                          mtry = 2,
                          nodesize = 1)

  pred <- predict(forest, testData, 'prob')

  BBDD_Resultado[Index == j, ]$pred <- pred[,2]

  print(paste0("Cross Validation número ", j, " de ", K, " realizada con
éxito. Proceso al ", j*100/K, " %."))

}

#Matriz de confusión

table <- data.frame(cut.off=double(),
                    Accuracy=numeric(),
                    Sensitivity=numeric(),
                    Specificity=numeric(),
                    Kappa=numeric(),
                    stringsAsFactors = FALSE)

co <- seq(0.5,1,0.05)

for (i in 1:length(co)){

  predict_labels <- as.factor(as.character(ifelse(BBDD_Resultado$pred >
co[i], 1, 0)))
  confumat <- confusionMatrix(data=predict_labels, reference=BBDD$ESTADO)

  table[i,1] <- co[i]
  table[i,2] <- round(confumat$overall[1]*100,2)
  table[i,3] <- round(confumat$byClass[1]*100,2)
  table[i,4] <- round(confumat$byClass[2]*100,2)
  table[i,5] <- round(confumat$overall[2]*100,2)
}

table <- table %>%
  arrange(desc(Kappa));table

best_cutoff <- table[1, 1]

BBDD_Resultado$pred_lab <- ifelse(BBDD_Resultado$pred > best_cutoff, 1, 0)
BBDD_Resultado$pred_lab <- as.factor(as.character(BBDD_Resultado$pred_lab))

confumat <- confusionMatrix(data = BBDD_Resultado$pred_lab, reference =
BBDD_Resultado[,1])
confumat

##############################################################################
##                          SCRIPT 5                                        ##
## 5. SVM                                                                   ##
##############################################################################
##############################################################################
# 5.1. Carga de librerias
##############################################################################

if(!require(e1071)){
  install.packages("e1071")
  library(e1071)
}
```

```
if(!require(caret)){
  install.packages("caret")
  library(caret)
}

if(!require(openxlsx)){
  install.packages("openxlsx")
  library(openxlsx)
}

if(!require(dplyr)){
  install.packages("dplyr")
  library(dplyr)
}


#############################################################################
# 5.2. Parametros SVM
#############################################################################

# 5.2.1. Parametros Modelo

Kernel_Vector <- #c('linear', 'polynomial', 'radial'
Cost_Vector <- #c(0.001, 0.01, 0.1, 10, 25, 50, 75, 100, 150)
Gamma_Vector <- #c(0.05, 0.01, 0.1, 0.5)
degree_Vector <- #c(4, 5, 6, 7) # Polynomial

w1 <- seq(0.05, 1, 0.05)
w0 <- 1 - w1
w0[length(w0)] <- 1

w <- data.frame(w1 = w1,
                w0 = w0)

Pesos <- table(BBDD_train$ESTADO)

Combinaciones <- length(Kernel_Vector) * length(Cost_Vector) *
length(Gamma_Vector) * length(degree_Vector) * nrow(w)

# 5.2.2. Cálculo del modelo

Modelo <- data.frame(Kernel = NA,
                     Coste = NA,
                     Gamma = NA,
                     Degree = NA,
                     Kappa_train = NA,
                     Accuracy_train = NA,
                     Sensibility_train = NA,
                     Sepecificity_train = NA,
                     Kappa_test = NA,
                     Accuracy_test = NA,
                     Sensibility_test = NA,
                     Sepecificity_test = NA)

head(Modelo)

for (m in 1:length(Pesos)) {

  for (n in 1:length(degree_Vector)) {
```

```
    for (k in 1:length(Gamma_Vector)) {

      for (j in 1:length(Cost_Vector)) {

       for (i in 1:length(Kernel_Vector)) {

    #==================================
    # Hiperparametros loop
    #==================================

    Kernel_model <- Kernel_Vector[i] # linear, polynomial, radial
    cost_model <- Cost_Vector[j] # Coste de violaciÃ³n del margen.
Necesario en todos
    gamma_model <- Gamma_Vector[k] # Polynomial, radial
    degree_model <- degree_Vector[n] # Polynomial
    Pesos[1] <- w0[m]
    Pesos[2] <- w1[m]

    if (Kernel_model == 'linear'){

      Modelo_svm <- svm(formula = ESTADO ~ .,
                        data = BBDD_train,
                        kernel = Kernel_model,
                        cost = cost_model)

    } else if (Kernel_model == 'polynomial'){

      Modelo_svm <- svm(formula = ESTADO ~ .,
                        data = BBDD_train,
                        kernel = Kernel_model,
                        cost = cost_model,
                        degree = degree_model)

    } else if (Kernel_model == 'radial'){

      Modelo_svm <- svm(formula = ESTADO ~ .,
                        data = BBDD_train,
                        kernel = Kernel_model,
                        cost = cost_model,
                        gamma = gamma_model)

    } else {stop("Error en el cálculo del modelo (6). El parámetro Kernel
(Kernel_model) debe ser: 'linear', 'polynomial' o 'radial'. ")}

    Tabla_SVM <- table(Real = BBDD_train$ESTADO, Prediccion =
Modelo_svm$fitted)
    Tabla_SVM

    confumat_train <- confusionMatrix(data = Modelo_svm$fitted, reference =
BBDD_train$ESTADO)

    BBDD_test_x <- BBDD_test[, which(colnames(BBDD_test) != 'ESTADO')]

    Prediccion <- predict(Modelo_svm, BBDD_test)

    confumat_test <- confusionMatrix(data=Prediccion,
reference=BBDD_test$ESTADO)

# 5.2.3. Tabla de resultados
```

```
    Modelo_aux <- data.frame(Kernel = Kernel_model,
                             Coste = cost_model,
                             Gamma =  gamma_model,
                             Degree = degree_model,
                             Kappa_train =
round(confumat_train$overall[2]*100,2),
                             Accuracy_train =
round(confumat_train$overall[1]*100,2),
                             Sensibility_train =
round(confumat_train$byClass[1]*100,2),
                             Sepecificity_train =
round(confumat_train$byClass[2]*100,2),
                             Kappa_test =
round(confumat_test$overall[2]*100,2),
                             Accuracy_test =
round(confumat_test$overall[1]*100,2),
                             Sensibility_test =
round(confumat_test$byClass[1]*100,2),
                             Sepecificity_test =
round(confumat_test$byClass[2]*100,2),
                             row.names = nrow(Modelo) -1 )

    Modelo <- rbind(Modelo, Modelo_aux)

    names(Modelo_aux)

    print(paste0("Kappa test = ", round(confumat_test$overall[2]*100,2),
                 ", Accuracy test = ",
round(confumat_test$overall[1]*100,2) ,
                 ", Sensibility test = ",
round(confumat_test$byClass[1]*100,2) ,
                 ", Specificity test = ",
round(confumat_test$byClass[2]*100,2), ". ",
                 "Cálculo ", nrow(Modelo)-1, " de ", Combinaciones, ".
Calculado un ", round(((nrow(Modelo)-1) * 100) / Combinaciones, 2), " %.
"))

        } # loop Kernel

      } # loop Cost

    } # loop Gamma

  }# Degree

} # Pesos

Modelo <- Modelo[-1,]
Modelo
#############################################################################
# 5.3. Pruebas SVM
#############################################################################

CM_Train_list <- list()
CM_Test_list <- list()

w1 <- seq(0.05, 1, 0.05)
w0 <- 1 - w1
w0[length(w0)] <- 1
```

```
Pesos <- table(BBDD_train$ESTADO)

for (i in 1:length(w0)) {

  Pesos[1] <- w0[i]
  Pesos[2] <- w1[i]

SVM <- svm(formula = ESTADO ~ .,
           data = BBDD_train,
           kernel = 'radial', #c('linear', 'polynomial', 'radial',
'sigmoid'),
           cost = 1, # Todos los kernels
           gamma = 1, # Radial
           degree = 1, # Polinómico
           class.weigths = Pesos)

Predict <- predict(SVM, BBDD_test)

SVM
CM_Train <- confusionMatrix(SVM$fitted, BBDD_train$ESTADO)
CM_Test <- confusionMatrix(Predict, BBDD_test$ESTADO)

CM_Train_list[[i]] <- CM_Train
CM_Test_list[[i]] <- CM_Test

print(paste0('Cálculo ', i, ' de ', length(w0), ' calculado con éxito'))

}
##############################################################################
# 5.4. Modelo elegido
##############################################################################

# Lineal

Modelo_svm <- svm(formula = ESTADO ~ .,
                  data = BBDD_train,
                  kernel = 'linear',
                  cost = 10)

Predict <- predict(Modelo_svm, BBDD_test)

Modelo_svm
confusionMatrix(Modelo_svm$fitted, BBDD_train$ESTADO)
confusionMatrix(Predict, BBDD_test$ESTADO)

# Polinómico

Modelo_svm <- svm(formula = ESTADO ~ .,
                  data = BBDD_train,
                  kernel = 'polynomial',
                  cost = 25,
                  degree = 4)

Predict <- predict(Modelo_svm, BBDD_test)

Modelo_svm
confusionMatrix(Modelo_svm$fitted, BBDD_train$ESTADO)
confusionMatrix(Predict, BBDD_test$ESTADO)
```

```
# Radial

Modelo_svm <- svm(formula = ESTADO ~ .,
                  data = BBDD_train,
                  kernel = 'radial',
                  cost = 75,
                  gamma = 0.1)

Predict <- predict(Modelo_svm, BBDD_test)

Modelo_svm
confusionMatrix(Modelo_svm$fitted, BBDD_train$ESTADO)
confusionMatrix(Predict, BBDD_test$ESTADO)
#############################################################################
##                            SCRIPT 6                                     ##
## 6. Gráficos                                                             ##
#############################################################################
#############################################################################
# 6.1. Carga de librerias
#############################################################################

if(!require(ggplot2)){
  install.packages("ggplot2")
  library(ggplot2)
}

if(!require(dplyr)){
  install.packages("dplyr")
  library(dplyr)
}

if(!require(e1071)){
  install.packages("e1071")
  library(e1071)
}

if(!require(tree)){
  install.packages("tree")
  library(tree)
}

if(!require(rpart)){
  install.packages("rpart")
  library(rpart)
}

if(!require(rpart.plot)){
  install.packages("rpart.plot")
  library(rpart.plot)
}

if(!require(ISLR)){
  install.packages("ISLR")
  library(ISLR)
}

if(!require(GGally)){
  install.packages("GGally")
  library(GGally)
```

```
}

if(!require(gridExtra)){
  install.packages("gridExtra")
  library(gridExtra)
}

if(!require(tidyr)){
  install.packages("tidyr")
  library(tidyr)
}


##############################################################################
# 6.2. Análisis Variables
##############################################################################

load('.../BBDD TFM.rdata')

BBDD <- BBDD %>%
        mutate(NUMERO_DE_POLIZAS_NO_SALUD =
                       as.factor(ifelse(NUMERO_DE_POLIZAS_NO_SALUD == 0,
                                        'Sin polizas vigentes distintas a salud',
                                        'Con polizas vigentes distintas a salud')),
               BAJAS_ANTERIORES_DISTINTAS_SALUD =
                       as.factor(ifelse(BAJAS_ANTERIORES_DISTINTAS_SALUD ==
0,
                                        'Sin bajas anteriores',
                                        'Con Bajas anteriores')),
               BAJAS_ANTERIORES_SALUD =
                       as.factor(ifelse(BAJAS_ANTERIORES_SALUD == 0,
                                        'Sin bajas anteriores',
                                        'Con bajas anteriores')))

summary(BBDD)
str(BBDD)


# Estado


Plot_bbdd <- BBDD %>%
  arrange(ESTADO) %>%
  group_by(ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = ESTADO, value = n) %>%
  ungroup()

Tabla_grafico <- tableGrob(Tabla)

plot <- ggplot(Plot_bbdd, aes(x = ESTADO, y = n)) +
  geom_col(mapping = aes(x = ESTADO, fill = ESTADO)) +
  labs(title = paste0('Estado póliza'),
       x = 'Estado',
       y = '')+
  theme_bw() +
```

```
  geom_text(aes(label = paste0(round(n * 100 /sum(n), 2), ' %')),
            position = position_stack(vjust = 0.5),
            color = 'black',
            fontface = 'bold',
            size = 5)

plot_combinado <- grid.arrange(plot, Tabla_grafico, heights = c(3/4, 1/4))

rm(Plot_bbdd, plot, Tabla, datos,  Tabla_datos, plot_combinado,
Tabla_grafico)


# Antiguedad poliza


Plot_bbdd <- BBDD %>%
  arrange(ANTIGUEDAD_POLIZA) %>%
  group_by(ANTIGUEDAD_POLIZA, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = ANTIGUEDAD_POLIZA, value = n) %>%
  ungroup()

Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0
Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = ANTIGUEDAD_POLIZA, y = n)) +
  geom_col(mapping = aes(x = ANTIGUEDAD_POLIZA, fill = ESTADO)) +
  labs(title = paste0('Antigüedad póliza'),
       x = 'Antigüedad',
       y = '') +
  theme_bw()

rm(Plot_bbdd, plot, Tabla, plot_combinado, Tabla_datos, datos,
Tabla_grafico)


# Autorizaciones


Plot_bbdd <- BBDD %>%
  arrange(AUTORIZACIONES) %>%
  group_by(AUTORIZACIONES, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = AUTORIZACIONES, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = AUTORIZACIONES    , y = n)) +
  geom_col(mapping = aes(x = AUTORIZACIONES, fill = ESTADO)) +
  labs(title = paste0('Autorizaciones'))+
```

```
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Bajas anteriores distintas de salud


Plot_bbdd <- BBDD %>%
  arrange(BAJAS_ANTERIORES_DISTINTAS_SALUD) %>%
  group_by(BAJAS_ANTERIORES_DISTINTAS_SALUD, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = BAJAS_ANTERIORES_DISTINTAS_SALUD, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = BAJAS_ANTERIORES_DISTINTAS_SALUD, y = n))
+
  geom_col(mapping = aes(x = BAJAS_ANTERIORES_DISTINTAS_SALUD, fill =
ESTADO)) +
  labs(title = paste0('Bajas anteriores distintas de salud'),
       x = 'Número de bajas',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Bajas anteriores salud


Plot_bbdd <- BBDD %>%
  arrange(BAJAS_ANTERIORES_SALUD ) %>%
  group_by(BAJAS_ANTERIORES_SALUD, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = BAJAS_ANTERIORES_SALUD, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = BAJAS_ANTERIORES_SALUD , y = n)) +
  geom_col(mapping = aes(x = BAJAS_ANTERIORES_SALUD, fill = ESTADO)) +
  labs(title = paste0('Bajas anteriores del ramo de salud'),
       x = 'Número de bajas',
       y= '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)
```

```
# Tipo de pago


Plot_bbdd <- BBDD %>%
  arrange(CODIGO_PAGO) %>%
  group_by(CODIGO_PAGO, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = CODIGO_PAGO, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = CODIGO_PAGO, y = n)) +
  geom_col(mapping = aes(x = CODIGO_PAGO, fill = ESTADO)) +
  labs(title = paste0('Código pago'),
       x = 'Códgio pago',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Edad media póliza


Plot_bbdd <- BBDD %>%
  arrange(EDAD_MEDIA_POLIZA) %>%
  group_by(EDAD_MEDIA_POLIZA, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = EDAD_MEDIA_POLIZA, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0


plot <- ggplot(Plot_bbdd, aes(x = EDAD_MEDIA_POLIZA, y = n)) +
  geom_col(mapping = aes(x = EDAD_MEDIA_POLIZA, fill = ESTADO)) +
  labs(title = paste0('Edad media de la póliza'),
       x = 'Edad media',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)



# Asegurados
```

```
Plot_bbdd <- BBDD %>%
  arrange(NASEG_POLIZA) %>%
  group_by(NASEG_POLIZA, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = NASEG_POLIZA, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = NASEG_POLIZA, y = n)) +
  geom_col(mapping = aes(x = NASEG_POLIZA, fill = ESTADO)) +
  labs(title = paste0('Número de asegurados vigentes'),
       x = 'Número de asegurados',
       y = '')+
  theme_bw()


rm(Plot_bbdd, plot, Tabla)


# Polizas vigentes distintas salud


Plot_bbdd <- BBDD %>%
  arrange(NUMERO_DE_POLIZAS_NO_SALUD) %>%
  group_by(NUMERO_DE_POLIZAS_NO_SALUD, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = NUMERO_DE_POLIZAS_NO_SALUD, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0


plot <- ggplot(Plot_bbdd, aes(x = NUMERO_DE_POLIZAS_NO_SALUD, y = n)) +
  geom_col(mapping = aes(x = NUMERO_DE_POLIZAS_NO_SALUD, fill = ESTADO)) +
  labs(title = paste0('Pólizas vigentes distinas de salud'),
       x = 'Número de pólizas vigentes',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Prima actual


Plot_bbdd <- BBDD %>%
  arrange(PRIMA_ACTUAL) %>%
  group_by(PRIMA_ACTUAL, ESTADO) %>%
  summarise(n = n()) %>%
```

```
  ungroup()

plot <- ggplot(Plot_bbdd, aes(x = PRIMA_ACTUAL)) +
  geom_density(mapping = aes(x = PRIMA_ACTUAL, fill = ESTADO), alpha = 0.5)
+
  labs(title = paste0('Prima Actual'),
       x = 'Prima',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)



# Siniestralidad


Plot_bbdd <- BBDD %>%
  arrange(SINIESTRALIDAD) %>%
  group_by(SINIESTRALIDAD, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

plot <- ggplot(Plot_bbdd, aes(x = SINIESTRALIDAD)) +
  geom_density(mapping = aes(x = SINIESTRALIDAD, fill = ESTADO), alpha =
0.5) +
  labs(title = paste0('Siniestralidad'),
       x = 'Siniestralidad',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Uso


Plot_bbdd <- BBDD %>%
  arrange(USO) %>%
  group_by(USO, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

plot <- ggplot(Plot_bbdd, aes(x = USO)) +
  geom_density(mapping = aes(x = USO, fill = ESTADO), alpha = 0.5) +
  labs(title = paste0('Uso'),
       x = 'Número de usos',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Tipo de Familia


Plot_bbdd <- BBDD %>%
  arrange(FAMILIA) %>%
  group_by(FAMILIA, ESTADO) %>%
  summarise(n = n()) %>%
```

```
  ungroup()

Tabla <- Plot_bbdd %>%
  group_by(ESTADO) %>%
  spread(key = FAMILIA, value = n) %>%
  ungroup()

Tabla[1, which(is.na(Tabla[1,]) == T)] <- 0
Tabla[2, which(is.na(Tabla[2,]) == T)] <- 0

plot <- ggplot(Plot_bbdd, aes(x = FAMILIA, y = n)) +
  geom_col(mapping = aes(x = FAMILIA, fill = ESTADO)) +
  labs(title = paste0('Familia'),
       x = 'Tipo de familia',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)


# Porcentaje variacion prima


Plot_bbdd <- BBDD %>%
  arrange(VARIACION_PORCENTUAL) %>%
  group_by(VARIACION_PORCENTUAL, ESTADO) %>%
  summarise(n = n()) %>%
  ungroup()

plot <- ggplot(Plot_bbdd, aes(x = VARIACION_PORCENTUAL)) +
  geom_density(mapping = aes(x = VARIACION_PORCENTUAL, fill = ESTADO),
alpha = 0.5) +
  labs(title = paste0('Variación porcentual de la prima'),
       x = 'Variación',
       y = '')+
  theme_bw()

rm(Plot_bbdd, plot, Tabla)
##############################################################################
# 6.2. Boxplot
##############################################################################

ggplot(BBDD, aes( y= PRIMA_ACTUAL)) +
  geom_boxplot(fill = 'aquamarine', outlier.color = 'brown1' )+
  labs(title = 'Boxplot: Ejemplo Método de Tukey') +
  ylab('Prima')+
  theme_bw()
##############################################################################
# 6.4. Ejemplos modelos
##############################################################################

##Logit

BBDD_plot <- Default %>%
  mutate(Y = ifelse(default == 'No', 0, 1),
         X = balance) %>%
  select(Y, X)

Modelo <- lm(Y ~ .,
             data = BBDD_plot)
```

```
BBDD_plot$pred <- predict(Modelo, BBDD_plot)
BBDD_plot$Y_fct <- as.factor(BBDD_plot$Y)


ggplot(BBDD_plot, aes(x=X, y=Y)) +
  geom_line(aes(x=X, y=pred)) +
  geom_point(aes(x=X, y=Y, color = Y_fct), show.legend = FALSE) +
  ggtitle('Ejemplo regresión simple') +
  labs(x = 'X', y = 'Y')+
  scale_fill_discrete(guide='none') +
  theme_bw() +
  theme(plot.title = element_text(size = 20))


## Random Forest

BBDD_plot <- titanic_train %>%
  filter(is.na(Age) == F) %>%
  mutate(Y = as.factor(Survived),
         X1 = as.factor(ifelse(Sex == 'male', 1, 0)),
         X2 = as.factor(Pclass),
         X3 = Age) %>%
  select(Y, X1, X2, X3)

Modelo <- rpart(Y ~ .,
                data = BBDD_plot)

rpart.plot(Modelo,
           main = 'Ejemplo árbol de decisión',
           cex.main = 2)

## SVM

# Linear

BBDD_plot <- iris %>%
  filter(!Species %in% 'setosa') %>%
  mutate(Species = as.factor(as.character(Species)),
         A = ifelse(Species == 'virginica' & Petal.Width < 1.75, 1, 0),
         B = ifelse(Species == 'versicolor' & Petal.Length > 4.9, 1, 0))
%>%
  filter(A == 0) %>%
  filter(B == 0) %>%
  select(Species, Petal.Length, Petal.Width)

ggplot(BBDD_plot, aes(x = Petal.Length, y = Petal.Width)) +
  geom_point(aes(x = Petal.Length, y = Petal.Width, colour = Species))

summary(BBDD_plot)
str(BBDD_plot)

Modelo <- svm(formula = Species ~  Petal.Length + Petal.Width,
              data = BBDD_plot,
              kernel = 'linear',
              scale = F,
              cost = 100)

grid <- expand.grid(seq(min(BBDD_plot[, 2]), max(BBDD_plot[, 2]),
length.out = 100),
```

```
                    seq(min(BBDD_plot[, 3]), max(BBDD_plot[, 3]),
length.out = 100))

colnames(grid) <- c('Petal.Length', 'Petal.Width')

pred <- predict(Modelo, grid)

ggplot(grid, aes(x = Petal.Length, y =Petal.Width)) +
  geom_tile(aes(fill = pred)) +
  geom_point(BBDD_plot, mapping = aes(x = Petal.Length, y =Petal.Width,
shape = Species), size = 2, show.legend = FALSE) +
  ggtitle('Ejemplo kernel lineal') +
  scale_fill_discrete(guide='none') +
  labs(x = 'X', y = 'Y')+
  theme_bw() +
  theme(plot.title = element_text(size = 20),)

# Polynomial

BBDD_plot <- iris %>%
  filter(!Species == 'setosa') %>%
  mutate(Species = ifelse(Species == 'virginica', 'virginica',
                          ifelse( Sepal.Length < 6, 'versicolor',
'virginica')),
         Species = as.factor(as.character(Species))) %>%
  select(Species, Sepal.Length, Petal.Length)

ggplot(BBDD_plot, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point(aes(x = Sepal.Length, y = Petal.Length, colour = Species))

summary(BBDD_plot)
str(BBDD_plot)

Modelo <- svm(formula = Species ~  Sepal.Length + Petal.Length,
              data = BBDD_plot,
              kernel = 'polynomial',
              scale = F,
              cost = 100,
              degree = 3)

grid <- expand.grid(seq(min(BBDD_plot[, 2]), max(BBDD_plot[, 2]),
length.out = 100),
                    seq(min(BBDD_plot[, 3]), max(BBDD_plot[, 3]),
length.out = 100))

colnames(grid) <- c('Sepal.Length', 'Petal.Length')

pred <- predict(Modelo, grid)

ggplot(grid, aes(x = Sepal.Length, y =Petal.Length)) +
  geom_tile(aes(fill = pred)) +
  geom_point(BBDD_plot, mapping = aes(x = Sepal.Length, y =Petal.Length,
shape = Species), size = 2, show.legend = FALSE) +
  ggtitle('Ejemplo kernel polinómico') +
  labs(x = 'X', y = 'Y')+
  scale_fill_discrete(guide='none') +
  theme_bw() +
  theme(plot.title = element_text(size = 20))

# Radial
```

```
BBDD_plot <- iris %>%
  filter(!Species %in% 'setosa') %>%
  mutate(Species = as.factor(Species),
         R = as.factor(ifelse(Sepal.Length > 5.7 & Sepal.Length < 6.4
                              & Petal.Length > 4.2 & Petal.Length < 5.1, 1,
0))) %>%
  select(R, Petal.Length, Sepal.Length)


ggplot(BBDD_plot, aes(x = Petal.Length, y = Sepal.Length)) +
  geom_point(aes(x = Petal.Length, y = Sepal.Length,  colour = R))

Modelo <- svm(formula = R ~  Sepal.Length + Petal.Length,
              data = BBDD_plot,
              kernel = 'radial',
              scale = F,
              gamma = 10,
              cost = 100)

grid <- expand.grid(seq(min(BBDD_plot[, 2]), max(BBDD_plot[, 2]),
length.out = 100),
                    seq(min(BBDD_plot[, 3]), max(BBDD_plot[, 3]),
length.out = 100))

colnames(grid) <- c('Petal.Length', 'Sepal.Length')

pred <- predict(Modelo, grid)
pred_2 <- rep(NA, length(pred))
pred_2[which(pred == 1)] <- 0
pred_2[which(pred == 0)] <- 1

pred_2 <- as.factor(pred_2)

ggplot(grid, aes(x = Petal.Length, y =Sepal.Length)) +
  geom_tile(aes(fill = pred_2)) +
  geom_point(BBDD_plot, mapping = aes(x = Petal.Length, y =Sepal.Length,
shape = R), size = 2, show.legend = T) +
  ggtitle('Ejemplo kernel polinómico') +
  labs(x = 'X', y = 'Y')+
  scale_fill_discrete(guide='none') +
  theme_bw() +
  theme(plot.title = element_text(size = 20))
```