uc3m | Universidad Carlos III de Madrid

Máster Universitario de Ciencias Actuariales y Financieras
2021-2022

Trabajo Fin de Máster

"Inteligencia Artificial aplicada al *cross-selling* en seguros"

Miriam Arribas García

Tutores

José Miguel Rodríguez-Pardo del Castillo

Jesús Ramón Simón del Potro

Madrid, junio de 2022

RESUMEN

La transformación imparable que supone la Inteligencia Artificial no deja indiferente al sector asegurador. Una de las estrategias de venta más rentables e innovadoras actualmente es el c*ross-selling*, que combinada con la Inteligencia Artificial en las compañías de seguros genera un enorme interés. En este trabajo se evalúan y discuten distintas metodologías modernas para predecir el interés de los actuales clientes de una compañía de seguros sobre una oferta de un seguro de automóviles. Este análisis constituye una propuesta que abarca numerosos beneficios debido a la exploración y manipulación de una gran cantidad de datos de la que se extrae información muy relevante.

Palabras clave: Conversión, *cross-selling*, Inteligencia Artificial, predicción, seguros.

ABSTRACT

The unstoppable transformation that Artificial Intelligence entails does not leave the insurance sector indifferent. Nowadays, one of the most profitable and innovative sales strategies is *cross-selling*, which combined with Artificial Intelligence in insurance companies generates enormous interest. In this work, different modern methodologies are evaluated and discussed to predict the interest of the current customers of an insurance company in an automobile insurance offer. This analysis constitutes a proposal that encompasses numerous benefits due to the exploration and manipulation of a large amount of data from which highly relevant information is extracted.

Keywords: Artificial Intelligence, conversion, *cross-selling*, insurance, prediction.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este trabajo se combina la ciencia de datos con una de las tácticas de venta más rentables actualmente. Gracias a esta combinación no solo es posible identificar a los futuros asegurados que estén más interesados en el producto de venta, sino obtener un profundo conocimiento sobre las necesidades de los clientes y sus deseos, detectar patrones de comportamiento, etc. Además, los datos reflejan una realidad que cambia constantemente por lo que es necesario actualizar este análisis para ser capaces de ofrecer a los clientes los productos más adecuados en el momento concreto.

Para llevar a cabo este estudio, primero se explicará la estrategia de *cross-selling* en el capítulo 3. Se describirán sus principales beneficios tales como la mejora en la relación con el cliente y en los ingresos de la compañía, así como sus principales riesgos que pueden llevar a una pérdida de los clientes o al diseño de una propuesta inadecuada.

En el capítulo 4 se realizará un breve análisis sobre las características de la base de datos que se usará para predecir la tasa de *cross-selling* en este estudio. Este capítulo constituye la base para identificar el problema de estudio en función a los datos disponibles y es clave para decidir el tipo de modelos de Inteligencia Artificial que se pueden aplicar para modelizar el objetivo del estudio.

El capítulo 5 describe el funcionamiento de los modelos que han ofrecido resultados muy buenos posteriormente. Se han introducido de manera que se entiendan fácilmente sin poseer conocimientos previos sobre Inteligencia Artificial. Por tanto, este capítulo persigue la idea de quede claro cómo predice cada modelo, así como conocer las distintas métricas de evaluación de los modelos.

El capítulo 6 propone una metodología para garantizar que los datos posean los requerimientos necesarios en la mayoría de modelos de Inteligencia Artificial. Se discuten diferentes técnicas y se escogen las más apropiadas para realizar el análisis de correlaciones, la codificación de las variables categóricas, la estandarización, la técnica de balanceado y la selección de las variables predictoras para los modelos.

En el capítulo 7 se muestran los resultados del entrenamiento de los modelos de Inteligencia Artificial con mejor capacidad predictiva. Así, se utiliza la base de datos preprocesada en el capítulo anterior, y se ponen en práctica los conceptos desarrollados en el capítulo 5 para encontrar un modelo que prediga lo mejor posible la tasa de *cross-selling*.

Por último, el capítulo 8 recoge las conclusiones de este Trabajo Fin de Máster.

2. OBJETIVOS

Los principales objetivos están ligados a maximizar el éxito de que se produzca el *cross-selling* mediante la predicción de los clientes interesados en ello. Así, los mediadores podrían realizar la oferta comercial directamente a los clientes que se conoce de antemano que estarían más interesados o se podría ajustar la oferta de *cross-selling* de manera que resulte más atractiva a un mayor número de clientes.

Junto a este fin, el procedimiento científico de los datos que se seguirá se explicará detalladamente de manera que constituya una propuesta metodológica aplicable a las compañías de seguros. No solo este proceso servirá para predecir la tasa de *cross-selling* o de conversión hacia el nuevo producto, sino que los mismos modelos y procedimientos se podrían utilizar para predecir cualquier otra variable dicotómica como la tasa de fraude o la tasa de abandono[1].

Como se describe en el capítulo 3, los beneficios del *cross-selling* van más allá de una venta tradicional. El propósito de predecir esta tasa también conlleva el cumplimiento de numerosos objetivos como: la obtención de información valiosa sobre los datos analizados trasladable a la mejora de otros proyectos de la compañía, identificación de las peculiaridades de las personas más interesadas en el *cross-selling* y con ello la detección de sus necesidades para poder ofrecerles mejores productos, aumento notable de la rentabilidad de la compañía de seguros, mejora de la relación con el cliente y sus probabilidades de fidelización…

[1] También conocida como *churn rate* en inglés.

3. CROSS-SELLING EN LAS COMPAÑÍAS ASEGURADORAS

Cross-selling[2] es una táctica de marketing mediante la cual se intenta la venta de productos adicionales tan pronto como el cliente esté interesado en comprar el producto o ya lo haya adquirido. Esta estrategia deriva en un **aumento de la rentabilidad** de las empresas que logran llevarla a cabo.

Las compañías de seguros se diferencian del resto de empresas en que el producto que venden es la seguridad. Es crucial transmitir al cliente plena confianza con la compañía aseguradora. En consecuencia, en el sector asegurador cobra aún más importancia para que el *cross-selling* se produzca y sea duradero el hecho de **mejorar la relación** con el cliente.

Para poder aplicarlo, es necesario comprender de antemano los momentos idóneos e identificar las características del cliente que propician la contratación de ciertos seguros. Con ello también se conseguirán mejorar las relaciones con el cliente y sus probabilidades de **fidelización**.

Por tanto, se debe realizar un **estudio para conocer mejor a los clientes** y ofrecerles productos adecuados a sus necesidades. Así, se les ofrecerán seguros con las coberturas de riesgo más necesitadas en el mercado.

Sin embargo, en el análisis del *cross-selling* se deben tener en cuenta también los precios de los contratos junto con sus coberturas ofrecidos por otras compañías de seguros que forman parte de la competencia. Es probable que el cliente también busque información externa y la compare, por lo que los **precios** de los nuevos contratos que se ofrecen deben ser **competitivos**.

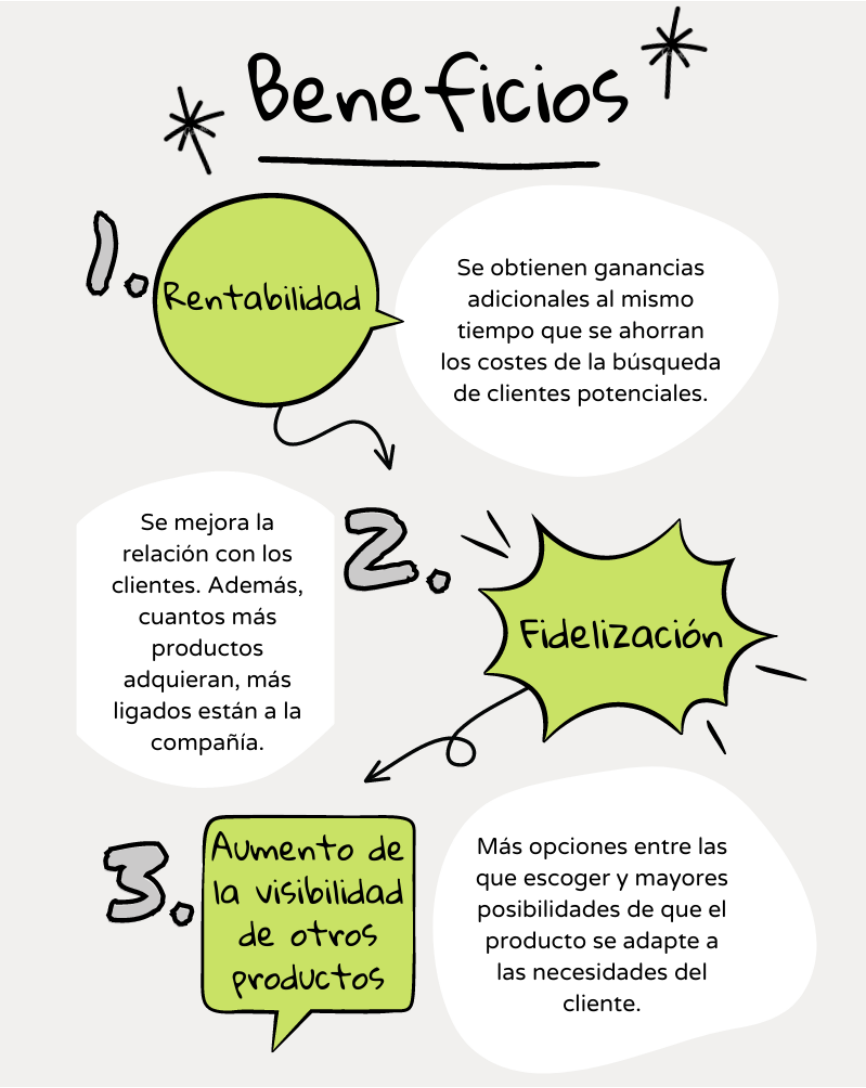
Las compañías de seguros pertenecen a un sector muy consolidado donde su bienestar en el mercado reside en la lealtad de sus clientes. Al promover esta fidelización, las compañías son conscientes de la importancia de estrategias como *cross-selling* para garantizar su supervivencia. Haciendo uso de un **CRM** (del inglés, *Customer Relationship Management*) se comprenden mejor las características principales del cliente, los productos que la compañía posee, así como los momentos más apropiados para ofrecer al cliente otro producto (Salazar et al., 2004). Las mejores oportunidades suelen ser cerca del momento de vencimiento de las pólizas o en determinados momentos del año como Navidad o verano, dependiendo del cliente.

[2] También conocido como “venta cruzada”.

Una vez que se ha analizado la estrategia del *cross-selling* y se ha señalado la importancia de mejorar el CRM y realizar un estudio previo para conocer de antemano las necesidades de los clientes más propensos a aumentar el contrato actual o crear uno nuevo con la compañía de seguros, aumentan notablemente las probabilidades de éxito del *cross-selling*.

A continuación, se exponen los principales beneficios y riesgos que conlleva el *cross-selling*.

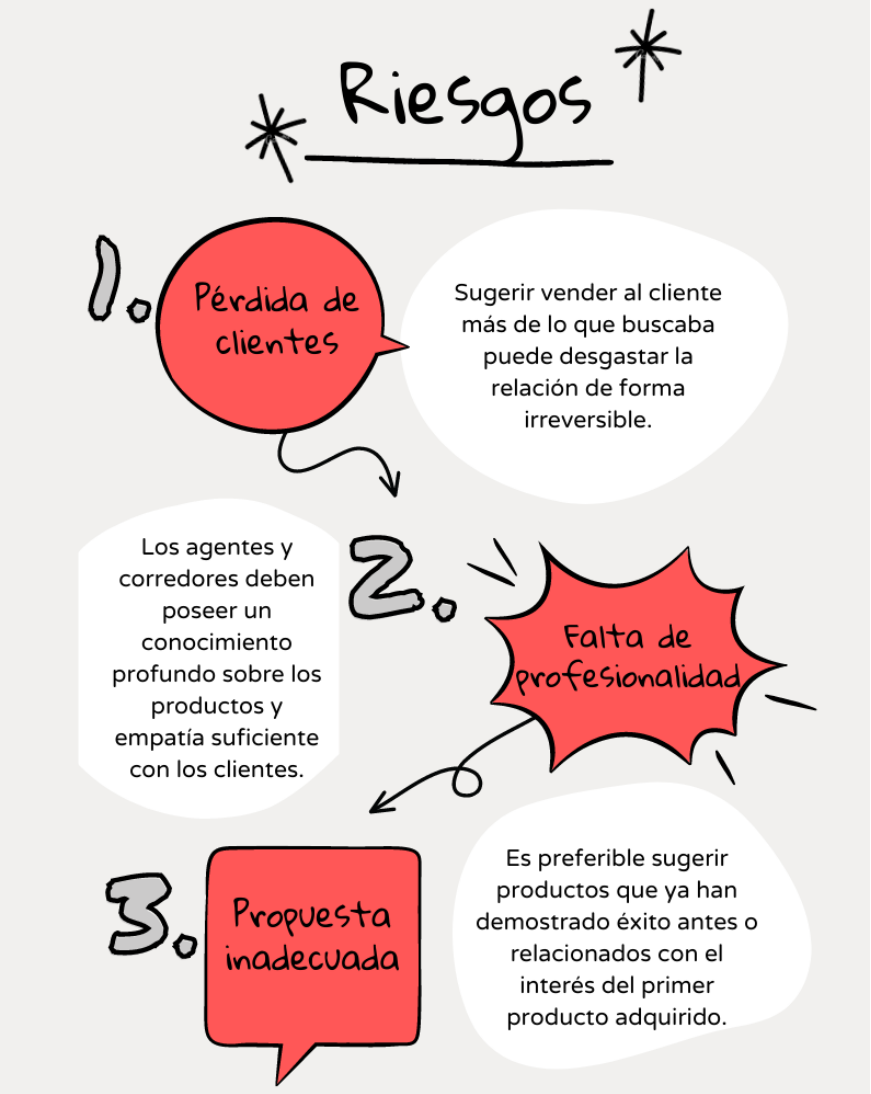
Figura 1. *Principales beneficios del cross-selling*



Fuente: Elaboración propia mediante Canva[3]

[3] https://www.canva.com/

Figura 2. *Principales riesgos del cross-selling*



Fuente: Elaboración propia mediante Canva[4]

Teniendo en cuenta lo anterior, la compañía debe ofrecer información actualizada con frecuencia a los agentes y corredores de seguros con el fin de que conozcan los productos lanzados, así como recordarles que revisen previamente los seguros y las coberturas que tiene contratado cada cliente. Además, la compañía de seguros debe disponer de un proyecto que ayude a entender en profundidad las necesidades de sus clientes y proporcionarles la propuesta más interesante.

Para ello es muy recomendable emplear algoritmos de Inteligencia Artificial para llevar a cabo una exploración de las peculiaridades de cada cliente e identificar la oferta que más se ajuste a sus necesidades.

[4] https://www.canva.com/

En este trabajo se realizará un estudio sobre los clientes que ya tienen contratado un seguro de salud con una compañía de seguros y se predecirá la tasa de conversión de estos clientes a un seguro de automóviles. Se propondrá una metodología para identificar las características de los clientes más propensos a la contratación del nuevo seguro con el fin de obtener un plan de *cross-selling* con garantías de éxito.

Para ello, se utilizarán numerosos modelos de Inteligencia Artificial y se comparará la eficacia de cada uno de ellos. No obstante, es necesario destacar que, aunque el plan de *cross-selling* sea el mejor que se pueda obtener, es imprescindible ofrecer un plan de formación constante a los comerciales de seguros con el fin de que adquieran las habilidades precisas para hacer el mejor uso posible de las conclusiones y combinarlo con buenas dotes persuasivas para cerrar el contrato.

4. ANÁLISIS DESCRIPTIVO DE LA BASE DE DATOS

La calidad de los datos es crucial para obtener buenos resultados en cualquier estudio. En este capítulo se realizará un análisis descriptivo a fondo de las variables de tarificación más significativas para nuestro caso de estudio. Se observará cómo se distribuyen, se analizarán las relaciones entre dichas variables y la variable objetivo, y se buscarán grupos homogéneos que presenten niveles similares en la variable objetivo.

Origen de los datos

Para predecir la tasa de venta adicional que se produce en cross-selling, se emplearán datos provenientes de Analytics Vidhya y obtenidos mediante hackathons en internet por Anmol Kuma. Estos datos han sido publicados en Kaggle[5], una plataforma pública de científicos de datos en la que también se comparten bases de datos.

La base de datos que se va a utilizar contiene información sobre clientes actuales que poseen un seguro de salud con una compañía de seguros americana. A estos clientes que ya tienen un seguro de salud, la misma compañía les ha intentado vender un nuevo producto: un seguro de automóviles. Como se ha explicado en el capítulo tercero, a esta táctica se la conoce como *cross-selling*.

El objetivo principal será predecir la tasa de conversión al seguro de automóviles e identificar las características del cliente y de su vehículo que lo suelen contratar. De esta manera, se obtiene información muy valiosa con la que se pueden incrementar los ingresos de la compañía de seguros.

También habría que tener en cuenta que la contratación de este nuevo seguro de automóviles depende de los precios de las primas que se ofrecen en la competencia. Por lo que habría que hacer un estudio de mercado para comparar la prima que ofrece nuestra compañía con las primas que ofrecen otras compañías de seguros. Sin embargo, se desconoce dicha información.

Esta predicción se realizará en base a los datos del cliente, del vehículo y de la nueva póliza de automóviles. Los datos constan de 381.109 registros recogidos durante un periodo de observación de 300 días. A continuación, se muestran las variables de estudio y una definición de las mismas:

[5] https://www.kaggle.com/datasets/anmolkumar/health-insurance-cross-sell-prediction

Tabla I. *Variables de estudio*

Nombre de la variable	Descripción
id	**Número identificador** del cliente
sexo	**Género** del cliente
edad	**Edad** del cliente
carnet_conducir	**0**: El cliente **no tiene licencia de conducir**, **1**: El cliente **sí** tiene licencia de conducir
codigo_postal	**Código de la región** donde vive el cliente
asegurado_antes	**1**: El cliente **tiene seguro de automóvil**, **0**: El cliente **no** tiene seguro de automóvil
edad_vehiculo	**Antigüedad del vehículo**
daños_vehiculo	**Yes**: El cliente tuvo **daños en su vehículo en el pasado**, **No**: El cliente **no tuvo daños** en su vehículo en el pasado,
prima	**Prima anual** que pagaría el cliente
canal_contacto	**Código del canal de contacto** con el cliente: agentes, por correo, por teléfono, en persona, etc.
dia	**Día** en el que se ha recogido los datos sobre el cliente interesado
conversion	**1**: El cliente **ha contratado** el seguro de coche **0**: El cliente **no ha contratado** el seguro de coche

Fuente: Elaboración propia

Además de las características nombradas previamente, también se tiene información sobre la variable respuesta u objetivo: *conversion*. Esta variable es dicotómica, es decir, solamente puede tomar los valores 0 o 1; por tanto, es indicadora de si el cliente ha convertido.

Se analizarán brevemente las variables objeto de estudio de la tabla anterior para tener una idea de la distribución que presentan. Como se ha indicado anteriormente, esta evaluación previa de los datos es necesaria para posteriormente preprocesar cada una de las variables que se van a utilizar en los modelos de manera adecuada. Además, en algunas variables también se pueden encontrar datos faltantes, inconsistentes o anómalos que es necesario tratar de la manera correcta. Por eso es necesario evaluar la calidad de los datos previamente, así nos aseguramos de obtener resultados consistentes en los modelos.

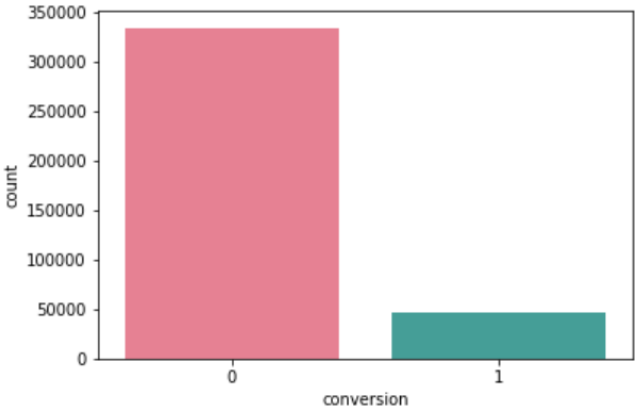
A continuación, se muestra la distribución de las variables más relevantes de acuerdo a su naturaleza categórica o numérica.

4.1. Análisis previo de variables categóricas

conversion

La variable *conversion* es una variable indicadora de si el cliente ha contratado el seguro de su vehículo. Esta variable se distribuye como una Bernoulli cuya media es 0.12256, y varianza 0.10754.

Figura I. *Diagrama de barras de conversion*



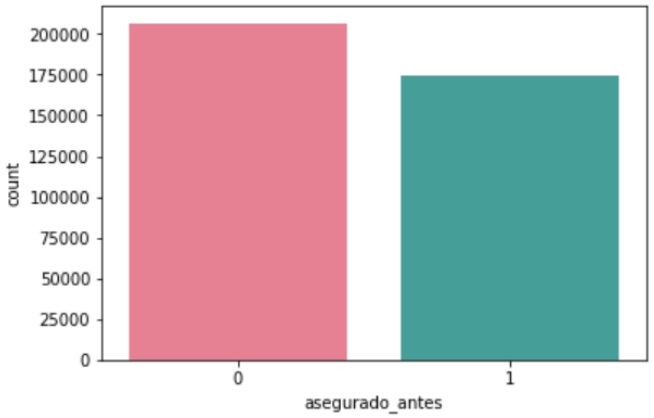
Fuente: Elaboración propia

En la figura anterior se observa una tendencia hacia la no contratación del seguro (valor 0), como es habitual en los modelos de conversión. Esto presenta un problema para los modelos predictivos ya que puede que tiendan a suponer que lo usual es que el cliente no convierta, y por tanto, no se estarían captando los patrones que suelen estar ligados a la conversión correctamente.

asegurado_antes

La variable *asegurado_antes* indica si el cliente posee un seguro de su vehículo actualmente. A partir de la figura II se ve que la mayoría de los vehículos no están asegurados (probablemente porque son nuevos, como se comprobará posteriormente con la variable *vehiculo_edad*).

Figura II. *Diagrama de barras de asegurado_antes*



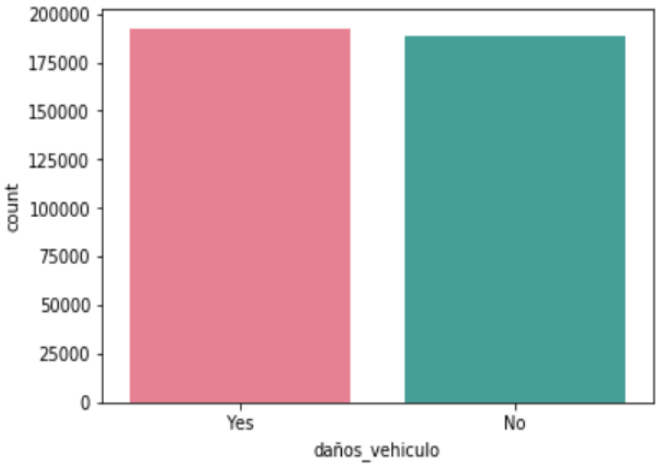
Fuente: Elaboración propia

Al igual que *conversion*, la variable *asegurado_antes* se distribuye como una Bernoulli pero con media 0.45821, lo que indica que está más balanceada.

daños_vehiculo

Una de las variables más significativas para estimar la prima, que como posteriormente se comprobará, está correlacionada con la conversión, es *daños_vehiculo*. La razón es que las compañías de seguros estiman que los clientes que han tenido siniestralidad en el pasado son más propensos a que la vuelvan a presentar. Por ende, la prima que la compañía de seguros ofrece será mayor, y la consecuente conversión más baja.

Figura III. *Diagrama de barras de daños_vehiculo*



Fuente: Elaboración propia

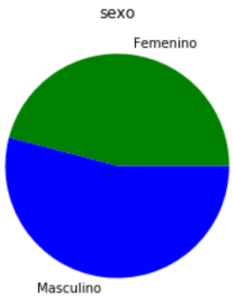
Si los valores que toma la variable se codifican como 0 (*No*) y 1 (*Yes*), se observa que la distribución de *daños_vehiculo* es Bernoulli con media 0.50487, y varianza 0.24997, lo que demuestra que está bastante nivelada en sus dos categorías.

Figura IV. *Diagrama de sectores de sexo*

sexo

En la variable que representa el sexo del cliente que contrata el seguro se observa un 56.88% de género masculino y un 46.66% de género femenino.

Por tanto, la moda, que es una de las medidas estadísticas más relevantes en esta variable, es el sexo masculino.



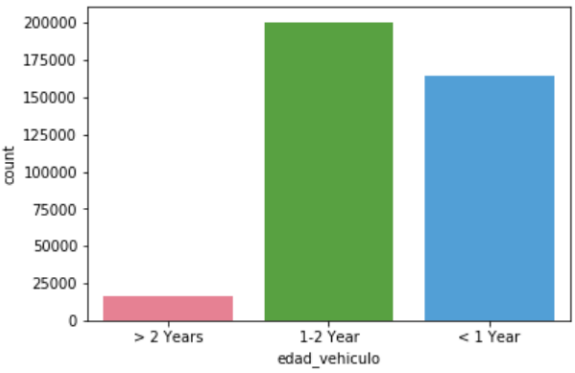
Fuente: Elaboración propia

edad_vehiculo

La variable *edad_vehiculo* refleja que en los registros hay una gran proporción de vehículos que poseen antigüedades inferiores a dos años.

Como se puede observar en la figura de la derecha, los valores que presenta la variable ya están agrupados en 3 categorías; por lo que se desconoce la antigüedad exacta de la mayoría de vehículos.

Figura V. *Diagrama de barras de edad_vehiculo*



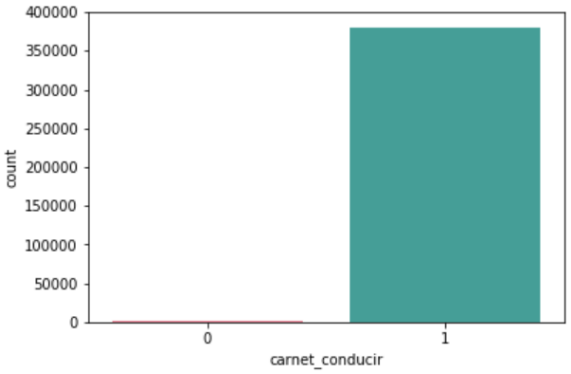
Fuente: Elaboración propia

carnet_conducir

La variable *carnet_conducir* indica si el cliente que contrata el seguro tiene la licencia necesaria para conducir (valor *1*).

En algunas compañías de seguros de España se puede ser el tomador del seguro sin tener el carnet de conducir. Sin embargo, es necesario especificar bien el conductor habitual.

Figura VI. *Diagrama de barras de carnet_conducir*



Fuente: Elaboración propia

En Estados Unidos la situación es similar. Está permitido pero algunas compañías de seguros para la contratación del seguro indican que se debe incluir al menos un conductor con licencia en la póliza.

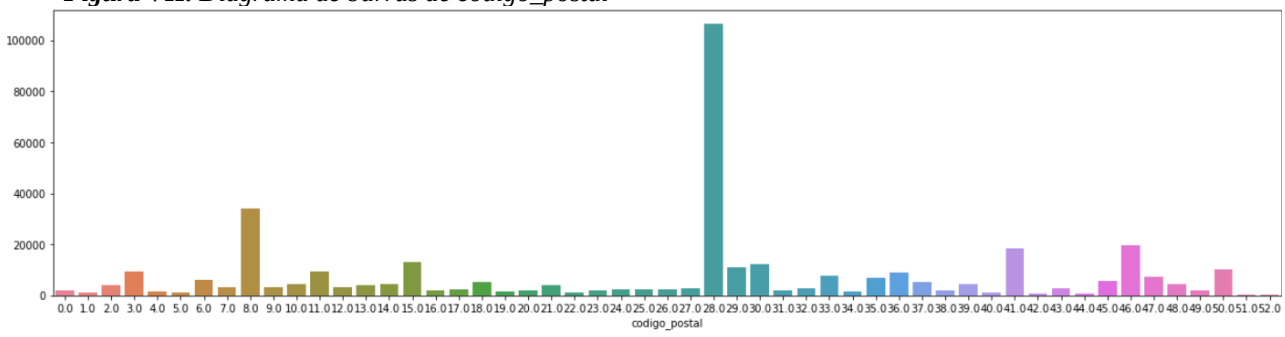
Por tanto, en el trabajo se permitirán pólizas cuyo tomador del seguro no tenga la licencia de conducción. Sin embargo, como se puede ver en la figura superior, en prácticamente todos los registros se posee la licencia de conducir, lo que indica que es una variable redundante con poco carácter de predicción, como se demostrará al ejecutar el mejor algoritmo para seleccionar las variables predictoras (*BorutaShap*).

codigo_postal

En la base de datos se tiene información sobre 52 códigos postales anonimizados. Cabe destacar la categoría 28.0 con un porcentaje cercano al 28%, representando el código postal con mayor frecuencia en esta variable. Puede que esta categoría corresponda a una ciudad grande, y el resto de categorías a municipios menos poblados.

Como hay demasiadas categorías en *codigo_postal*, es necesario agruparlas. Como los códigos postales están anonimizados, no se puede suponer que los códigos postales identificados por números próximos entre sí se localizan próximos entre ellos. Es decir, a partir de la figura inferior puede que los códigos postales representados por un color similar no se localicen cercanos en un mapa. En el capítulo 6 se abordará este problema.

Figura VII. *Diagrama de barras de codigo_postal*



Fuente: Elaboración propia

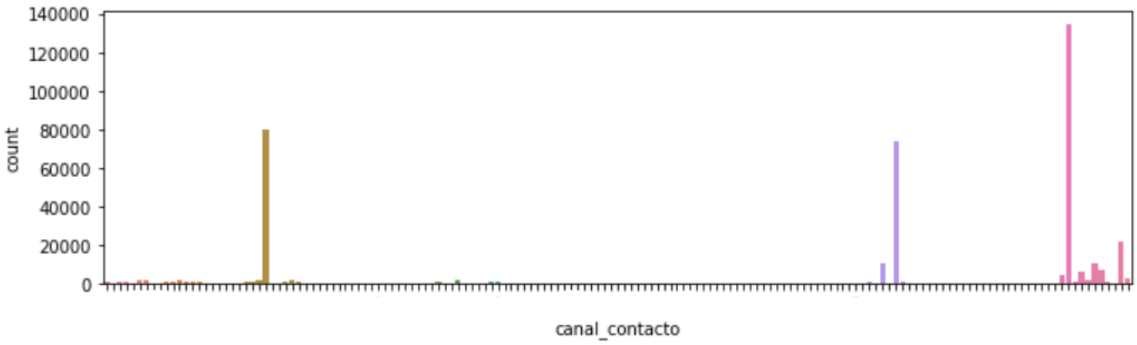
canal_contacto

Un comportamiento similar a la variable *codigo_postal* lo presenta la variable que indica el canal de venta elegido por el cliente, ya que en ella existe un número notable de categorías. Sin embargo, en *canal_contacto* se encuentran muchas más. En concreto, 163 canales anonimizados.

Lo ideal sería conocer una descripción de cada categoría que representa un canal de venta distinto y agrupar los distintos canales de manera que tengan en común las principales características. Por ejemplo, todos los clientes que se hayan puesto en contacto con distintos mediadores de seguros se podrían agrupar en una única categoría que se denomine “mediadores”.

En base a la figura inferior, cabe destacar 3 categorías con una alta frecuencia en la base de datos. Una de ellas podría ser el canal directo, en el que el cliente se pone en contacto directamente con la compañía de seguros mediante una llamada telefónica (por ejemplo). Otros posibles canales de contacto según la documentación publicada con la base de datos son: mediante correo, en persona, a partir de diferentes agentes, etc. También cabría añadir mediante la página web de la compañía de seguros, y mediante comparadores de seguros online como es el caso de *Rastreator* en España.

Figura VIII. *Diagrama de barras de canal_contacto*



Fuente: Elaboración propia

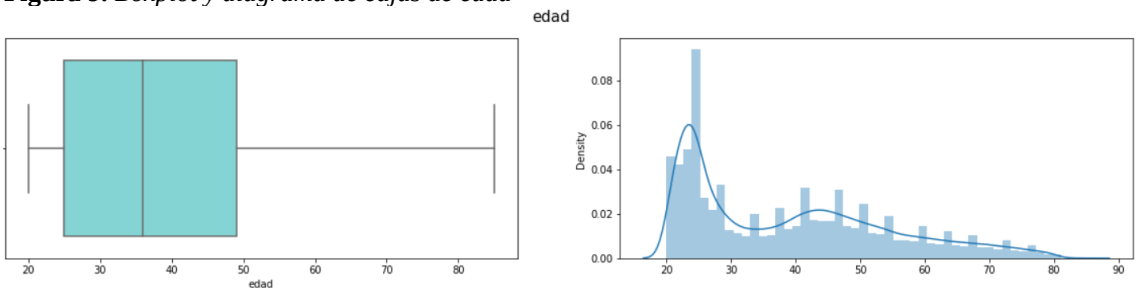
Debido a la alta cardinalidad de esta variable, hay un problema de codificación para las distintas categorías que, al igual que la variable *codigo_postal*, se abordará en el capítulo 6. Al estar anonimizados, en función del número de cada categoría no se puede suponer que números próximos entre sí forman parte de un canal similar.

4.2. Análisis previo de variables numéricas

edad

Los valores modales de la edad de los potenciales clientes están comprendidos en torno a los 20 y 30 años. Sin embargo, la mediana de edad es 36 años y las personas con edad más elevada poseen 85 años. Su distribución es ligeramente asimétrica positiva.

Figura 9. *Boxplot y diagrama de cajas de edad*



Fuente: Elaboración propia

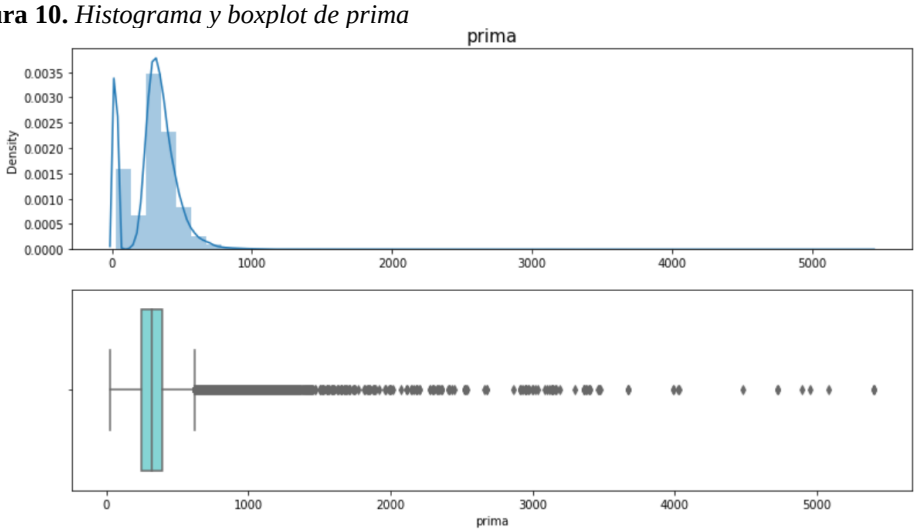
prima

La prima anual que el cliente pagaría en caso de que contratase el seguro de automóvil tiene una distribución asimétrica positiva, dado que tiene una gran cola ligeramente pesada hacia la derecha.

Como se puede observar en el *boxplot* inferior, la cola de la distribución es larga. A partir del valor 1.000$ hay muchas primas atípicas con valores muy elevados. Estas primas es probable que se correspondan con seguros de automóviles todo riesgo sin franquicia.

Como se explicará en el capítulo 6 en el que se preprocesa la base de datos, los valores atípicos que se aprecian en el *boxplot* podrían dificultar la predicción de la conversión de los modelos. Por ello se decide aplicar el método de Tukey para detectar los que se consideran más anómalos y eliminarlos. Además, debido a la grandísima cola en esta variable va a ser necesario escalarla. Así, los algoritmos de Machine Learning que se ejecutarán funcionan mejor si las variables están en una escala relativamente similar.

Figura 10. *Histograma y boxplot de prima*

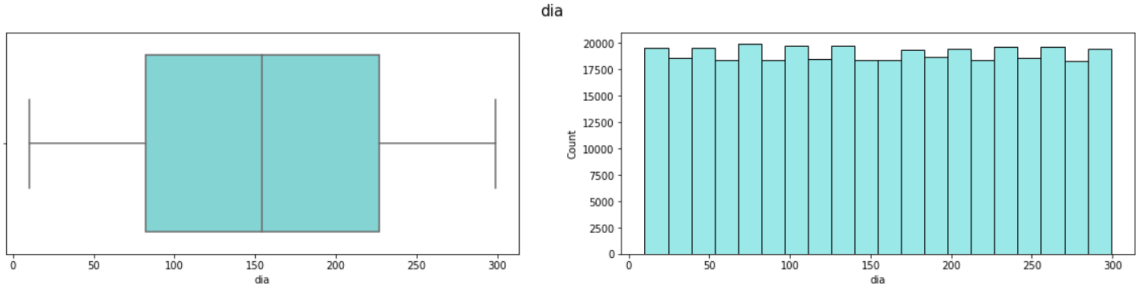


Fuente: Elaboración propia

4.3. Relaciones relevantes entre variables iniciales

dia & conversion

Figura 11. *Boxplot e histograma de dia*



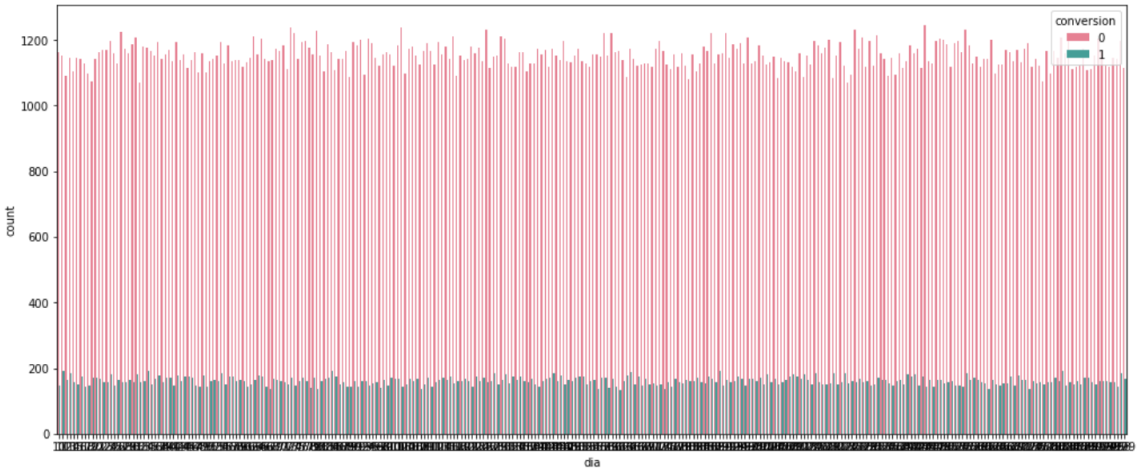
Fuente: Elaboración propia

La distribución de la variable *dia* se muestra en la figura superior. En ella, se puede apreciar que los datos se han recogido de manera uniforme y con un comportamiento ligeramente periódico durante todo el periodo considerado, que ha sido de 300 días.

En la figura inferior, las barras coloreadas en rojo representan a los clientes que no han convertido en ese día concreto, y las barras coloreadas en verde los que sí han decidido contratar el seguro de automóviles en ese día. Tal y como se muestra en el histograma, la tasa de conversión durante todo el periodo considerado también se ha mantenido uniforme.

Por tanto, se podrá prescindir de esta variable en la modelización posterior, ya que no aporta ningún carácter predictivo.

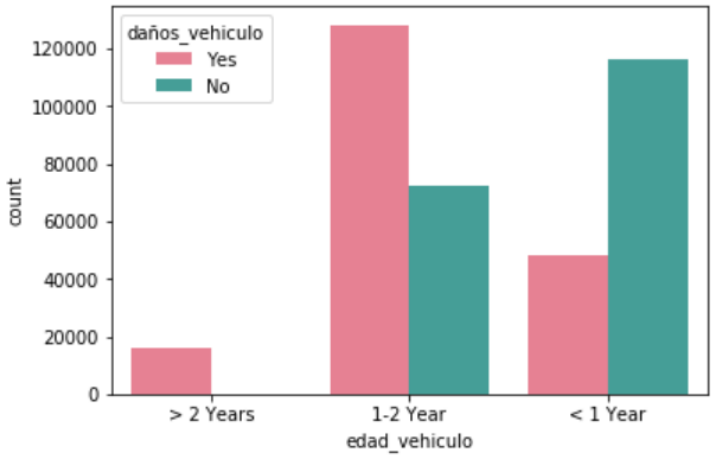
Figura 12. *Histograma de dia en función de la conversión*



Fuente: Elaboración propia

edad_vehiculo & daños_vehiculo

Figura 13. *Diagrama de barras de edad_vehiculo en función de daños_vehiculo*



Fuente: Elaboración propia

Tabla I. *Proporción de edad_vehiculo en función de daños_vehiculo*

daños_vehiculo	No	Yes
edad_vehiculo		
> 2 Years	0.000937	0.999063
1-2 Year	0.359886	0.640114
< 1 Year	0.707524	0.292476

Fuente: Elaboración propia

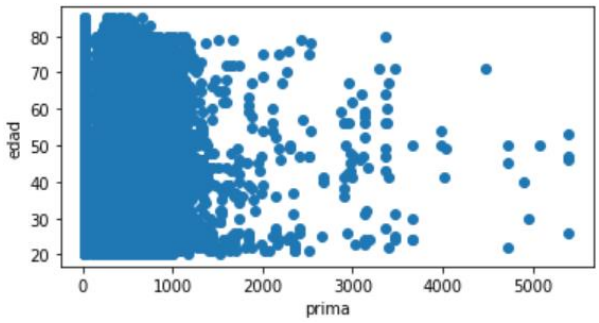
Como se analizó anteriormente, hay poca proporción de vehículos con más de 2 años de antigüedad en la base de datos. Acorde al diagrama de barras y a la tabla, es muy probable que los vehículos de esta categoría hayan presentado daños.

Por el otro lado, es lógico que la categoría formada por vehículos más nuevos es la que menos ha experimentado daños en el pasado.

prima & edad

Intentando identificar las características de los *outliers* que previamente se hallaron en la variable *prima*, cabe señalar que a priori no existe evidencia para justificar que la edad influya en el valor de la prima. A partir de primas superiores a 4000$, parece que es más probable que se correspondan con edades inferiores a 60 años. Sin embargo, hay que tener en cuenta la proporción más baja de las personas mayores en la base de datos.

Figura 14. *Diagrama de dispersión entre las variables edad y prima*



Fuente: Elaboración propia

5. MODELOS DE INTELIGENCIA ARTIFICIAL

Los modelos estadísticos son tan buenos como la base de datos se adecúe a sus requerimientos y la calidad de los datos sea la mejor posible. Por ello, en el capítulo anterior se realizó una evaluación previa de la base de datos para decidir cuáles son los modelos más adecuados. Como la variable objetivo es dicotómica, se usarán modelos que predigan una clasificación binaria.

Actualmente, las personas no poseen la capacidad de analizar e interpretar detenidamente la grandísima cantidad de datos disponible para tomar las decisiones críticas en las compañías de seguros. Con este fin, se planteó la posibilidad de crear máquinas que imiten el razonamiento humano. Y así fue como se originó la **Inteligencia Artificial**, suponiendo un gran desarrollo tecnológico y científico en el siglo XXI que está en plena evolución y ocasionará un gran impacto en múltiples campos.

Machine Learning es una rama de la Inteligencia Artificial que consiste en entrenar un subconjunto de datos con el objetivo de identificar patrones que permitan resolver problemas comerciales o de cualquier otro tipo. A diferencia de Machine Learning, se conoce como **Deep Learning** a los modelos que razonan y estudian los datos proporcionando a los ordenadores la capacidad de aprender sin ser programados explícitamente.

En este capítulo se explicará la teoría detrás de cada modelo, haciendo hincapié en asimilar muy bien su funcionamiento. Los modelos que se van a exponer son los que han mostrado mejores resultados en la predicción del *cross-selling*. Estos modelos han sido Random Forest, XGBoost, CatBoost, una red neuronal artificial y la regresión logística aplicada a Machine Learning.

Los modelos aplicados en este trabajo pertenecen a Machine Learning, en concreto a la categoría de **aprendizaje supervisado**[6], que constituye prácticamente el 99% de Machine Learning. La red neuronal también se puede englobar dentro del denominado Deep Learning.

A varios de estos modelos se les suele conocer como "**cajas negras**" ya que generan predicciones muy buenas, pero no resulta tan sencillo entender la automatización que ha llevado a tales resultados.

[6] Los modelos de aprendizaje supervisado se caracterizan por entrenarse a partir de una muestra de variables explicativas y la variable objetivo que es la que se predecirá.

5.1. GLM: Regresión logística

La regresión logística es otro modelo de Machine Learning utilizado para clasificación. La regresión logística pertenece al conjunto de los modelos lineales generalizados (GLM, del inglés *Generalized Linear Models*) que generaliza la regresión lineal como se explicará a continuación.

Aunque a priori se posee la creencia de que la regresión logística difiere completamente de otros modelos como las redes neuronales por su falta de complejidad y sus buenos fundamentos matemáticos que facilitan su interpretación, la regresión logística se podría considerar una red neuronal con una única neurona.

Los modelos lineales generalizados nacen a partir del modelo lineal general. Dado un par de variables aleatorias (X, Y) en el que X es una variable aleatoria con m dimensiones e Y una variable aleatoria unidimensional, entonces el **modelo lineal general** se define a partir de la siguiente ecuación:

$$\boldsymbol{Y} = \beta_0 + \sum_{j=1}^{m} \boldsymbol{X_j}\beta_j + \boldsymbol{\varepsilon}$$

donde:

- β_0 es el intercepto,
- β_j son los coeficientes de regresión,
- $\boldsymbol{X_j}$ representa las observaciones de las variables independientes o regresoras,
- $\boldsymbol{Y}$ representa las observaciones de la variable respuesta o target, y
- $\boldsymbol{\varepsilon}$ representa a variables aleatorias normales incorreladas de media 0 y varianza constante σ^2.

A partir de las hipótesis de normalidad e incorrelación de $\boldsymbol{\varepsilon}$, se deduce que la variable respuesta $\boldsymbol{Y}$ también se distribuye como una variable aleatoria normal y no presenta correlación.

Para explicar la limitación que conlleva la suposición de las hipótesis anteriores, se introduce el siguiente ejemplo en el que $\boldsymbol{Y}$ es una variable aleatoria que indica si el cliente ha contratado otro seguro para i = 1, ..., n:

$$y_i = x'_i\beta + \varepsilon_i$$

Es decir, y_i tomaría el valor 1 si el cliente ha contratado el seguro, y 0 en caso contrario. Al ser la variable respuesta dicotómica, los errores ε_i sólo pueden tomar 2 valores: $\varepsilon_i = \begin{cases} 1 - x'_i\beta, & si\ y_i = 1 \\ -x'_i\beta, & si\ y_i = 0 \end{cases}$. Por tanto, la hipótesis de normalidad de los errores del modelo no se cumpliría.

Los **GLM** surgen para permitir variables dependientes cuya distribución sea distinta de la normal. En los modelos lineales generalizados la distribución de la variable dependiente debe pertenecer a la **familia exponencial**, que incluye las distribuciones normal, binomial, binomial negativa, exponencial, gamma, Poisson y normal inversa. Todas estas distribuciones se pueden expresar a partir de la siguiente ecuación en la que φ es un parámetro de dispersión y θ se conoce como parámetro de ubicación natural:

$$f(y,\theta,\varphi) = e^{\frac{y\theta - b(\theta)}{a(\varphi)} + h(y,\varphi)}$$

Los GLM unifican los modelos de regresión lineales y no lineales mediante el desarrollo de un modelo lineal para una función adecuada g de los valores esperados de la variable respuesta ($g[E(y)]$). A esta función adecuada g también se la conoce como **función de enlace**. El resultado de aplicar la función de enlace a E(y) se denomina **predictor lineal**.

En la regresión logística, el predictor lineal que se suele utilizar es $ln(\frac{p}{1-p})$, siendo p la media de la variable respuesta, o también, la probabilidad de que la variable respuesta tome el valor 1. A este predictor lineal en la regresión logística se le conoce como **transformación logit**, y garantiza que la probabilidad p pertenezca al intervalo (0,1).

En resumen, mediante la metodología de los modelos lineales generalizados, para cada distribución de la familia exponencial se consideran distintas transformaciones de la variable respuesta y en el caso de la regresión logística, dicha transformación suele ser la transformación logit.

A veces, la transformación logit también se denomina **transformación log(odds)**, siendo odds la proporción $\frac{p}{1-p}$, es decir, el cociente entre la probabilidad de que el cliente contrate el nuevo seguro y de que no lo contrate. Por lo que indica cuánto más probable es que se produzca el *cross-selling* respecto a que no se produzca.

Por último, además de la transformación logit o transformación log(odds), en la regresión logística también se aplica la **transformación probit**. Esta transformación se define a partir de la distribución normal acumulativa: $\Phi^{-1}(p)$.

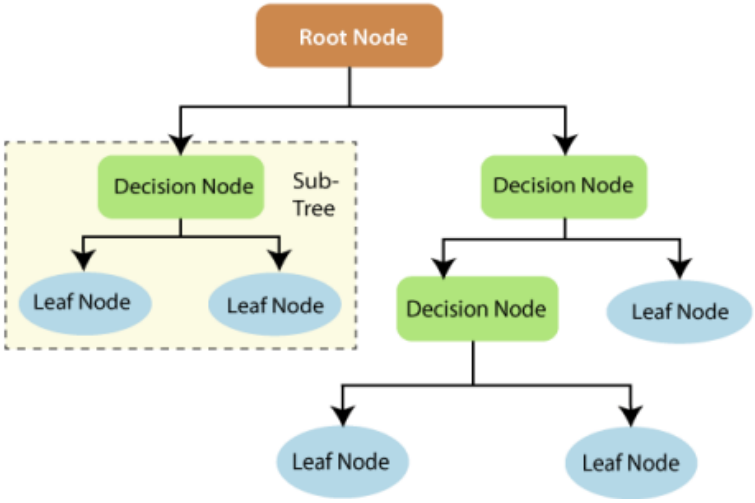
5.2. Random Forest para clasificación

Random Forest es un algoritmo de aprendizaje supervisado desarrollado por Leo Breiman en el año 2001, en el que se combinan diferentes árboles de decisión que han sido generados independientemente a partir de la base de datos original muestreada y con la misma distribución para cada uno de ellos (Breiman, 2001). También incorpora las ideas de Tin Kam Ho de 1995 de entrenar el modelo solamente empleando una selección aleatoria de variables para cada árbol de decisión (Ho, 1995).

En primer lugar, se explicará brevemente la construcción de un árbol de decisión para posteriormente proceder con la exposición del modelo Random Forest.

Los árboles de decisión consisten en predecir la variable respuesta o target a partir de los valores de las variables explicativas. Para ello, en el primer **nodo raíz** y en cada una de las siguientes ramas (o nodos de decisión como se muestra en la figura inferior) se expresa una condición en función de los valores de una variable explicativa, y en función de si se cumple o no esa condición se realiza la división en dos nuevos nodos. Generalmente, si la condición es verdadera se sigue con el siguiente nodo izquierdo y si la condición es falsa con el derecho. Los nodos a partir de los cuales no hay más ramificaciones se denominan **nodos hoja**.

Figura 17. *Árbol de decisión*



Fuente: (Charbuty & Abdulazeez, 2021)

Los árboles de decisión para clasificación binaria se entrenan de la siguiente manera: en cada uno de los nodos resultantes después de una ramificación se cuenta el número de valores de la variable respuesta en cada categoría. Lo ideal es que después de realizar una ramificación, uno de los 2 nodos resultantes obtenga todo ceros en la variable respuesta y el otro todo unos. Si un nodo resultante de una ramificación obtiene una mezcla de ceros y unos se denomina **nodo impuro**.

El criterio para especificar la condición de cada nodo rama se basa en cuantificar la impureza de los nodos resultantes y calcular una media ponderada de los 2 nodos. La variable con la que se obtengan valores más bajos en la cuantificación de la impureza de los nodos resultantes será la mejor. A continuación, se muestran las fórmulas de los 2 métodos más populares para elegir la variable utilizada en la condición con la que se obtiene la mejor ramificación.

Se denota por p_0 a la probabilidad de que el cliente no contrate el seguro adicional y, por tanto, no se produzca el *cross-selling*. La probabilidad de que sí se produzca es p_1.

- **Impureza de Gini**: $1 - p_0^2 - p_1^2$
- **Ganancia de información**: $-p_0 log_2(p_0) - p_1 log_2(p_1)$

Se podrían aplicar estos métodos hasta que ningún nodo hoja sea impuro. No obstante, en ese caso es muy probable que se sobreajuste a los datos (*overfitting*) al captar detalles o particularidades en los datos de entrenamiento que puede que no ocurran en los datos de prueba. Por tanto, es necesario parar antes especificando límites que funcionen bien testeados previamente con validación cruzada, o **podar el árbol** para que a partir de los datos de entrenamiento el árbol de decisión generalice bien. Para conocer distintas técnicas de podado se recomienda la lectura de (Mingers, 1989).

El modelo **Random Forest** se construye a partir de múltiples árboles de decisión entrenados sobre diferentes muestras bootstrap de las observaciones de la base de datos original. Además, para cada ramificación de cada árbol de decisión sólo se tiene en cuenta un subconjunto aleatorio de variables. A esta técnica se la conoce como **bootstrap aggregating** (abreviado popularmente como **bagging**). La cardinalidad del subconjunto de variables que se tiene en cuenta en cada ramificación en el primer Random Forest es la raíz cuadrada del número de variables explicativas totales.

Una vez que se han entrenado todos los árboles de decisión del primer modelo Random Forest, se cuenta el número de veces que se ha predicho que el *cross-selling* tendría éxito para cada árbol de decisión y el número de árboles que han predicho que no tendría éxito. La categoría con mayor frecuencia constituiría la predicción final del Random Forest para esa observación.

De esta manera, se obtiene una amplia variedad de árboles de decisión que logra aumentar notablemente el rendimiento del modelo Random Forest en comparación al árbol de decisión.

Finalmente, cabe destacar que aproximadamente un tercio de la base de datos original no se incluye en cada muestra bootstrap. A este tercio de la base de datos se le conoce por el adjetivo de "**out-of-bag**". Por tanto, dado que la base de datos out-of-bag de cada árbol de decisión no ha sido utilizada para entrenar el resto de árboles de decisión, se puede

utilizar para comprobar la cantidad de aciertos del Random Forest. Repitiendo el procedimiento con el resto de bases de datos out-of-bag del resto de los árboles se puede cuantificar la capacidad predictiva del Random Forest.

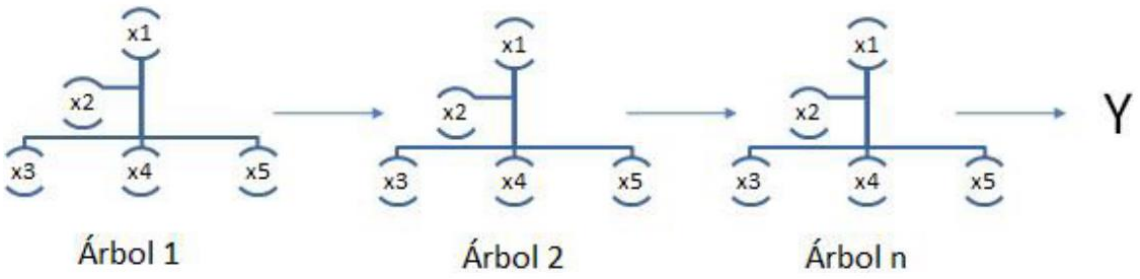
La proporción de veces que se clasifica mal esta muestra se denomina **error out-of-bag**. En base a esta métrica, se mejoran los árboles de decisión modificando el número óptimo de variables mediante distintas pruebas en las que se aumenta y disminuye un poco el número resultante de efectuar la raíz cuadrada del número total de variables. Así, se optimizará la cardinalidad del subconjunto de variables escogido para que se obtengan los mejores resultados.

5.3. XGBoost para clasificación

XGBoost (del inglés, *Extreme Gradient Boosting*) es un algoritmo de aprendizaje supervisado ampliamente utilizado actualmente por su gran eficacia. Suele ser considerado el algoritmo de Machine Learning más potente por sus buenos resultados en competiciones de ciencia de datos, como las de la plataforma *Kaggle*. XGBoost fue desarrollado por Chen y Guestrin en 2016.

De manera resumida, este algoritmo consiste en un **sistema de árboles de decisión** (conocido como CART, del inglés *Classification and Regression Trees*) concatenados secuencialmente para aprender de los resultados de los árboles previos y mejorar los errores producidos por ellos (Espinosa-Zúñiga, 2020).

Figura 18. *Ejemplo de ensamblado secuencial de árboles de decisión de XGBoost*



Fuente: (Espinosa-Zúñiga, 2020)

A cada árbol de decisión que forma parte del algoritmo XGBoost se le autodenomina ***learner* débil**, ya que proporciona una predicción ligeramente superior a una predicción aleatoria. La combinación de todos estos árboles resulta realmente eficaz al minimizar una función de pérdida en la que se ha tenido en cuenta la regularización.

Los resultados tan buenos que se obtienen con XGBoost se deben principalmente al ensamblado de varios *learners* débiles en los que se utilizan métodos estadísticos

modernos, junto con un uso muy eficiente de la memoria y de la computación paralela y distribuida. Así, con gran rapidez se ejecutan procesos muy optimizados.

5.3.1. Algoritmo

Aunque XGBoost a su vez, permite distintos procedimientos en su propio algoritmo, a continuación, se explicará de manera sencilla pero rigurosa un ejemplo de los pasos que se ejecutan cuando se utiliza el algoritmo XGBoost para clasificación. Es decir, XGBoost para clasificación predice una variable dicotómica como se hará en este trabajo, que toma el valor 0 si no se ha producido el *cross-selling* y toma el valor 1 si el cliente ha decidido contratar el otro seguro propuesto y por tanto el *cross-selling* ha tenido éxito.

Para cuantificar lo bien que se ha predicho se utiliza la **función de pérdida**. La principal diferencia entre el uso de XGBoost para regresión o para clasificación es la función de pérdida. Sea $\{(x_i, y_i)\}_{i=1}^{n}$ donde x_i representan las filas de observaciones que se utilizan para predecir y_i, entonces el logaritmo de la verosimilitud de los datos observados dada la probabilidad predicha es:

$$\sum_{i=1}^{n} y_i \ * \ log(p_i) \ + \ (1 - y_i) \ * \ log(1 - p_i)$$

siendo p_i las probabilidades predichas y y_i los valores de la variable *target* que pueden ser 0 o 1.

Así, cuanto mejor sea la predicción, mejor será el logaritmo de la verosimilitud, por lo que el objetivo será maximizar la función anterior. Sin embargo, como se quiere usar el logaritmo de la verosimilitud como una función de pérdida en la que valores pequeños representan mejores modelos predictivos, se multiplicará el logaritmo de la verosimilitud por -1:

$$L(\boldsymbol{y}, \boldsymbol{p}) = \ -\sum_{i=1}^{n} y_i \ * \ log(p_i) \ + \ (1 - y_i) \ * \ log(1 - p_i)$$

Por tanto, la ecuación que se buscará minimizar al realizar cada árbol es:

$$L(\boldsymbol{y}, \boldsymbol{p} + Output) + \ \gamma T + \frac{1}{2} \lambda \ Output^2$$

siendo T el número de nodos terminales u hojas en un árbol, γ una penalización definida por el usuario destinada a fomentar la poda del árbol, λ es un término de regularización y Output representa el valor que minimiza la ecuación anterior.

Como los árboles de decisión de XGBoost se construyen en base a los residuos obtenidos en los anteriores árboles de decisión, se supone una predicción inicial de 0.5 para todas las observaciones. Así, se obtienen los primeros residuos en base a los cuales construir el primer nodo raíz aplicando la ecuación anterior.

Minimizando la ecuación anterior en función de Output se obtiene la fórmula específica para calcular el valor Output que tendrá cada hoja resultante del nodo:

$$Output = \frac{-(g_1 + g_2 + \ldots + g_n)}{h_1 + h_2 + \ldots + h_n + \lambda}$$

siendo g_i la primera derivada de la función de pérdida y h_i la segunda derivada de la función de pérdida.

Como los valores de Output se expresan en función de log(odds), la ecuación anterior que se minimiza (*función de pérdida + término para podar + término para regularización*) se ha transformado teniendo en cuenta que odds = p/(1-p), de manera que se exprese en función de log(odds) en vez de en función de p.

También se han simplificado los cálculos utilizando la aproximación del polinomio de Taylor de segundo orden:

$$L(y, p_i + Output) \approx L(y, p_i) + \left[\frac{\boldsymbol{d}}{\boldsymbol{dp_i}} L(y, p_i)\right] \boldsymbol{Output} + \frac{\mathbf{1}}{\mathbf{2}}\left[\frac{\boldsymbol{d^2}}{\boldsymbol{dp_i^2}} L(y, p_i)\right] \boldsymbol{Output^2}$$

$$= L(y, p_i) + g_i\, Output + \frac{1}{2}\, h_i\, \boldsymbol{Output^2}$$

Así se obtienen los valores de Output que se corresponden con los valores de las hojas resultantes del primer nodo raíz.

Usando este valor de Output se obtiene la **función de puntuación del árbol**:

$$Puntuación\ del\ árbol = \frac{-(g_1 + g_2 + \ldots + g_n)^2}{2(h_1 + h_2 + \ldots + h_n + \lambda)} + \gamma T$$

Con la función de puntuación anterior se mide la calidad de la estructura de cada árbol (Chen & Guestrin, 2016). Esta es la ecuación que viene en el manuscrito original de XGBoost. Sin embargo, la ecuación de puntuación que aparece en la documentación de la librería de XGBoost[7] con la que se implementa el modelo, es dos veces la ecuación anterior.

Ambos resultados finales son iguales ya que la función de puntuación se utiliza para evaluar los posibles árboles y elegir el mejor árbol. Al escalar todas las funciones de puntuación de los distintos árboles, el resultado de la comparación no cambia. La razón por la que se decide no incluir la multiplicación por ½ se debe a reducir la cantidad de cálculos de computación, lo que prueba el ahínco de XGBoost por mejorar la rapidez y optimizar los cálculos.

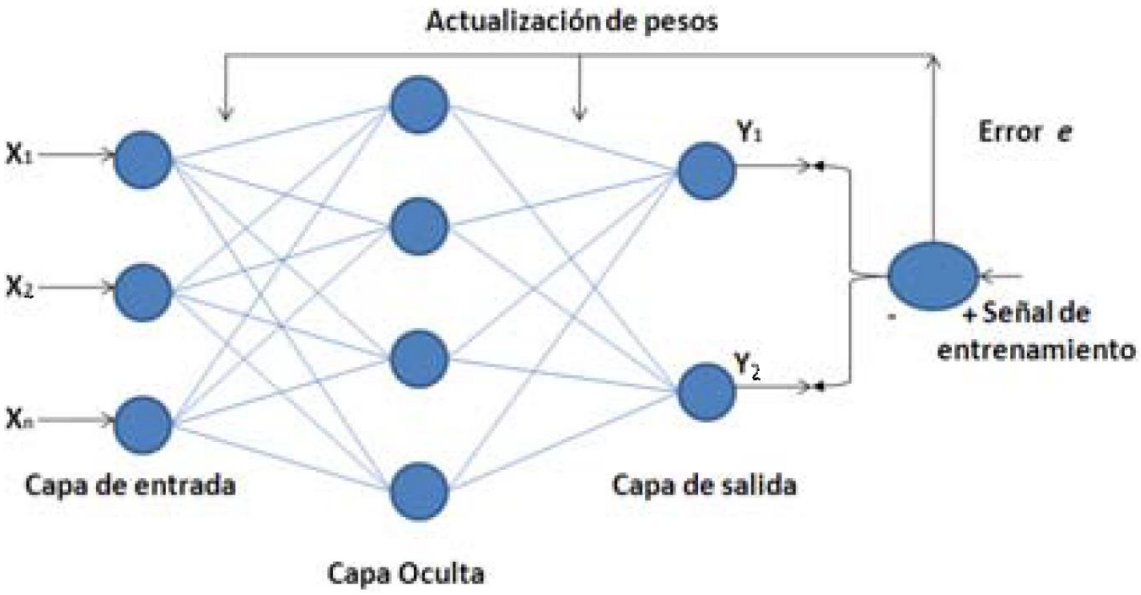
[7] https://xgboost.readthedocs.io/en/latest/index.html

5.4. Redes neuronales multicapa para clasificación binaria

Las redes neuronales constituyen una rama de la Inteligencia Artificial en la que modelos computacionales pretenden simular el funcionamiento de las neuronas en el cerebro humano. Así, la red de neuronas humanas se puede representar como un grupo de nodos de la red neuronal conectados entre sí que tratan de aprender nuevas cosas y adaptarse a distintos entornos.

Una red neuronal multicapa con el objetivo de clasificar en función de si se produce el *cross-selling* está compuesta por distintos elementos: la capa input formada por los nodos que recibirán la información de las variables predictoras, la capa output constituida por dos nodos para realizar la clasificación binaria, las capas de nodos intermedios, y una red de conexiones entre cada capa de nodos. Estas capas de nodos intermedios se conocen como **capas ocultas** (del inglés, *hidden layers*).

Figura 19. *Red neuronal con una sola capa oculta*



Fuente: (Vicens-Salort et al., 200916-04)

A continuación se explican brevemente procurando no entrar en detalle excesivo, algunos conceptos que es necesario conocer para entender el funcionamiento de estas redes neuronales.

5.4.1. Backpropagation

Gracias a la técnica de *backpropagation* se obtienen los **pesos** y el **sesgo** asociado a cada conexión de neuronas entre distintas capas próximas. *Backpropagation* constituye el paso inicial para el entrenamiento de la red neuronal.

Backpropagation es el proceso inverso de la red neuronal que consiste en recorrer la red de derecha a izquierda (desde la capa output hasta la capa input). Se basa en propagar el error desde las últimas capas hacia las primeras mediante el error imputado de una capa. De esta manera, se actualizan los pesos de forma iterativa. Es en este proceso cuando los pesos de las neuronas aprenden realmente.

Esta técnica para reducir los errores consiste en calcular las derivadas parciales de la suma del cuadrado de los residuos en función de los parámetros desconocidos. Posteriormente se aplica el método del descenso del gradiente para optimizar el parámetro desconocido. Estos parámetros desconocidos representan los pesos asignados a la información recibida de cada neurona de la capa anterior y el sesgo.

5.4.2. Funciones de activación

Cada neurona que compone la red neuronal y que trata de simular el comportamiento de una neurona del cerebro humano, posee una función que determina la activación de esa neurona.

Por eso estas funciones se conocen como "**funciones de activación**" y se aplican a la suma ponderada de los valores de entrada de cada neurona por sus pesos sumado al sesgo correspondiente. Se recuerda que estos parámetros, los pesos y el sesgo, se obtuvieron en *backpropagation*.

Las principales funciones de activación que se pueden especificar en las redes neuronales son:

- **Función sigmoide**:

$$P(t) = \frac{1}{1 + e^{-t}}$$

  Debido al coste computacional tan elevado de esta función, no se suele especificar en las neuronas de las capas intermedias. La imagen de la función comprende todos los valores entre 0 y 1.

- **Función relu**:

$$f(x) = max(0, x)$$

  Es la que más se suele utilizar en la práctica debido a su bajo coste computacional. La imagen de la función comprende valores positivos incluyendo al cero.

- **Función tangente hiperbólica**:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

  El coste computacional de esta función es un poco más elevado, y su imagen comprende todos los valores en el intervalo (-1,1).

- **Función de salto binaria**:

$$f(x) = \begin{cases} 0 \ \ si\ x \leq U \\ 1 \ \ si\ x > U \end{cases} \quad siendo\ U\ un\ cierto\ umbral$$

  En las primeras redes neuronales se utilizaba esta función también conocida como escalonada debido a su discontinuidad inevitable. Sin embargo, en algunos casos presenta desventajas al no ser derivable cuando x=U.

- **Función soft max**:

$$\alpha(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\ , \quad j\ \epsilon\ \{1, \ldots, K\}$$

  Esta función supone un gran coste computacional en comparación con la función relu, que es la más eficiente. La imagen de esta función está comprendida en el intervalo [0,1].

En resumen, la información que reciben los nodos de la capa input pasa hacia los nodos de las capas ocultas multiplicándose por los pesos y sumando el sesgo, habiéndose obtenido estos parámetros con el método de ***backpropagation***. En cada uno de los nodos por los que pasa esta información se aplica la **función de activación** correspondiente. El objetivo de los nodos de las capas ocultas consiste en transformar la información de entrada de manera que se obtenga un resultado que se ajuste muy bien a las predicciones conocidas. Al final, cada resultado que se obtiene de los nodos de la última capa oculta se escala por un cierto número (un peso). A la suma de todos estos resultados se le resta un valor (un sesgo) y finalmente se obtiene un ajuste de las variables explicativas a las predicciones que se repetirá numerosas veces.

De esta manera es como se entrena la red neuronal. El error obtenido con la diferencia entre las predicciones finales y las predicciones reales, se vuelve a propagar hacia atrás, desde la capa output hasta las capas ocultas y desde estas hasta la capa input. Este flujo vuelve a actualizar los pesos y el sesgo y así es como la red neuronal va aprendiendo de los datos cada vez más, tratando de simular las conexiones de las células del sistema nervioso humano con otras células.

Una vez entrenada la red neuronal y habiendo logrado un gran ajuste, se seguiría el mismo procedimiento que se realiza en los modelos de Machine Learning en los que se prueba la capacidad predictiva de la red neuronal con los datos de prueba.

A continuación, se explicarán distintas medidas de validación para evaluar el rendimiento de cada uno de los modelos.

5.5. Medidas de evaluación de los modelos

Una vez que se han preparado los datos y se han entrenado los modelos, el siguiente paso consiste en medir la capacidad predictiva de cada uno de ellos. Esta evaluación ha de realizarse en datos que el modelo no ha conocido anteriormente. Se comparará la predicción estimada por los modelos con los valores reales.

Como se ha observado en el anterior capítulo, la variable conversión indica que los clientes que deciden contratar el nuevo seguro representan un porcentaje muy pequeño en comparación a los que rechazan convertir al nuevo seguro. En concreto, el porcentaje de conversión es de 12.25%.

Por tanto, modelos que siempre predigan que el *cross-selling* nunca tiene éxito obtendrían un 87.75% de aciertos. Sin embargo, no cumplirían el objetivo de identificar las características de los clientes que estarían interesados.

Por ello, es crucial comprender las principales medidas de evaluación que se suelen utilizar en los modelos de Machine Learning para poder elegir los criterios más adecuados que cuantifiquen lo bien que se han identificado a los clientes interesados en el *cross-selling*.

A partir del conocimiento de la matriz de confusión se expondrán las principales medidas para evaluar los modelos.

5.5.1. Matriz de confusión

El caso de estudio consiste en predecir los valores de la variable conversión, que serán 0, si el cliente no ha contratado el nuevo seguro y por tanto el *cross-selling* no ha tenido éxito, y 1 en caso contrario.

La matriz de confusión, como su propio nombre indica, es una matriz en la que se representan los valores reales frente a las predicciones de los modelos de la siguiente manera:

Tabla III. *Matriz de confusión*

		Valores predichos	
		1	0
Valores reales	**1**	TP	FN
	0	FP	TN

Fuente: Elaboración propia

donde:

- TP (del inglés, *True Positive*): representa a las personas interesadas en el *cross-selling* (1) y que el modelo también ha predicho que están interesadas (1).

- FN (del inglés, *False Negative*): representa a las personas interesadas en el *cross-selling* (1) aunque el modelo ha estimado que no estarían interesados (0).

- TN (del inglés, *True Negative*): representa a las personas no dispuestas a realizar el *cross-selling* (0) y también lo ha predicho así el modelo (0).

- FP (del inglés, *False Positive*): representa a las personas no dispuestas a realizar el *cross-selling* (0) aunque el modelo haya predicho que estarían interesadas (1).

5.5.2. Precisión

Se define la precisión como:

$$\frac{TP}{TP + FP}$$

Es decir, es el porcentaje de todos los 1 (o positivos) que se han predicho que se han acertado. También se conoce como la proporción de verdaderos positivos.

5.5.3. Recall

Se define recall como:

$$\frac{TP}{TP + FN}$$

Es decir, es el porcentaje de todos los 1 (o positivos) que hay en los datos reales que se han acertado.

5.5.4. Exactitud

Se define la exactitud como:

$$\frac{TP + TN}{TP + FN + TN + FP}$$

Es decir, es el porcentaje de valores que se han acertado en los datos de prueba. El mejor valor de la exactitud sería el 1 y el peor el 0.

Como se ha comentado anteriormente, esta medida no es la más adecuada para elegir el mejor modelo de conversión cuando la mayoría de los valores son 0, es decir, rechazo a la oferta. Un modelo que siempre prediga 0 tendría un alto valor de exactitud dado que todos los 0 los predeciría bien.

En vez de la definición de exactitud usual, es más apropiado para el estudio de la conversión el uso de la **exactitud balanceada**, que consiste en realizar la media de recall para cada clase. Así se evitaría el problema de los datos de prueba que no están balanceados.

5.5.5. F1-Score

Se define F1-Score como:

$$2 * \frac{precisión * recall}{precisión + recall}$$

Por tanto, es la media armónica entre la precisión y recall, combina ambas medidas en un único valor. En el caso de que ambas medidas se consideren igual de importantes, F1-Score constituye un buen criterio para seleccionar el mejor modelo.

Sin embargo, al predecir la conversión se dará mayor importancia al valor obtenido con recall ya que es importante evitar predecir erróneamente que personas interesadas en el nuevo contrato no estén interesadas. Lo anterior también es conocido como minimizar el error tipo II.

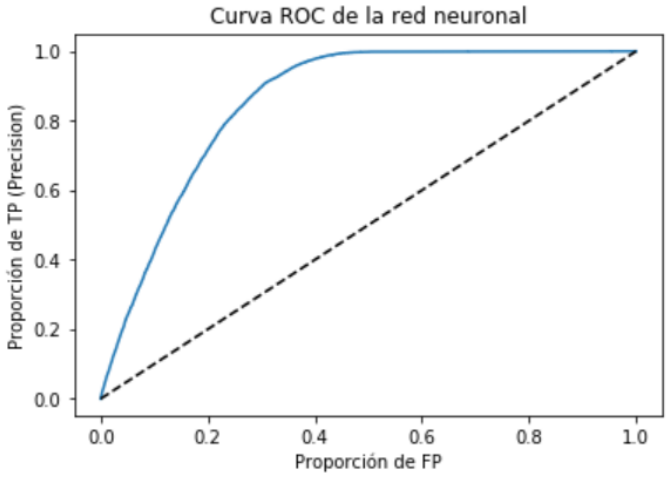
5.5.6. Área bajo la curva ROC

La curva ROC (del inglés, *Receiver Operating Characteristic*) representa la proporción de TP en el eje Y frente a la proporción de FP en el eje X en función de distintos valores del umbral de discriminación.

Así, la curva ROC tiene en cuenta a todas las personas que el modelo ha predicho que estarían interesadas en el *cross-selling* en función de distintos valores del umbral comprendidos entre 0 y 1.

Teniendo en cuenta todos estos umbrales de discriminación, para determinar qué modelo predice mejor en base a este criterio se calcula el área bajo la curva ROC (AUC, del inglés *Area Under the Curve*).

Figura 20. *Explicación de la curva ROC y AUC*



Fuente: Elaboración propia

Cuanto más próxima a una L invertida esté la curva ROC mejor será el modelo. En ese caso ideal, al calcular el área bajo la curva ROC y teniendo en cuenta que tanto la base como la altura son 1, el valor máximo de AUC sería 1.

Si la curva ROC viniese representada por la curva discontinua de la imagen superior, el valor de AUC sería de 0.5. Como los valores que predice el modelo son ceros y unos, en este caso el modelo tendría la misma capacidad predictiva que un modelo aleatorio que lanzase monedas al aire.

6. ESTUDIO Y PREPARACIÓN DE LOS DATOS

Además de preparar los datos de las variables que se van a utilizar en los modelos, en este capítulo se presentará una propuesta metodológica para predecir la tasa de *cross-selling* aplicable a las compañías de seguros.

Atendiendo a ese propósito será necesario antes que nada tener una noción inicial de las posibles relaciones de cada una de las variables con la variable objetivo que es la conversión. Esto se realizará gráficamente y posteriormente se cuantificará la relación de esas fuerzas con un análisis de las correlaciones.

Se hará especial hincapié en la exposición de la metodología que se llevará a cabo en esta etapa tan importante, pero a la que no se le suele prestar la atención que merece. Para asegurar que los datos que se van a utilizar en los modelos posean la máxima calidad, se discutirá cuál es la mejor técnica a emplear desde el análisis de correlaciones hasta todas las etapas del preprocesamiento incluyendo la selección de las variables predictoras.

Recalcando el párrafo anterior se discutirá por ejemplo cuál es la mejor técnica para realizar la codificación a cada variable, en qué casos realizarla y cuando no en función del modelo que se vaya a utilizar; por qué se utiliza un coeficiente de correlación para cuantificar las relaciones entre las variables y no otro; en qué consiste el balanceado de los datos y cómo aplicarlo de manera correcta, etc.

Teniendo en cuenta todo ello, el orden que se seguirá en esta propuesta metodológica para garantizar buena calidad de los datos en los modelos de conversión es:

1. Relaciones de las variables con la variable objetivo
2. Análisis de correlación
3. Preprocesamiento de los datos
    3.1. Codificación de las variables
    3.2. División de la base de datos en training set y test set
    3.3. Estandarización
    3.4. Aplicación de técnica de balanceado de datos
4. Selección de variables predictoras

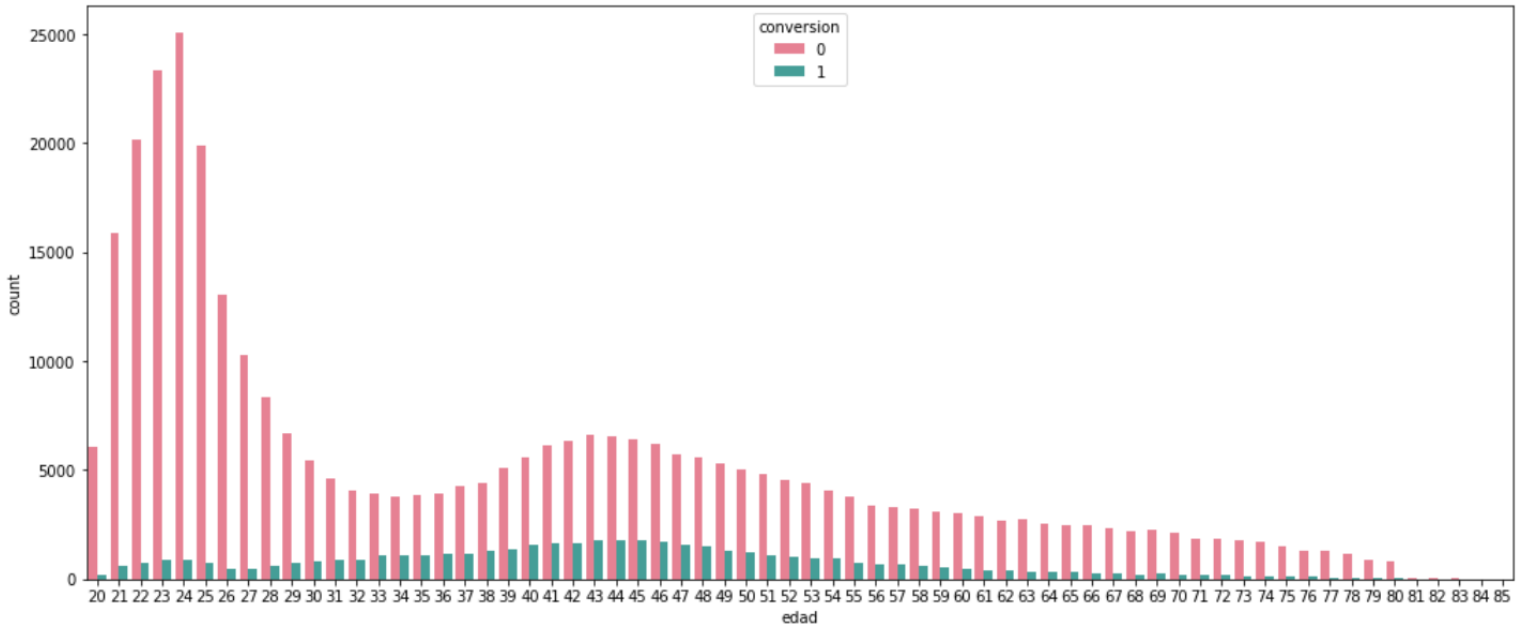
6.1. Relaciones de las variables con la variable objetivo

edad

En el diagrama de cajas inferior se aprecia que, aunque haya una gran proporción de personas jóvenes, este grupo de edad es el que presenta la menor tasa de conversión. Probablemente se deba a que los jóvenes pueden ser los que más tiempo inviertan buscando distintos precios en distintas compañías de seguros (como ocurre en este caso dado que en la base de datos de las personas que han obtenido el presupuesto hay una gran proporción de jóvenes) y, por tanto, decidan no contratar el seguro de la mayoría de compañías de seguros de las que han obtenido el presupuesto.

Por el contrario, la mayor tasa de conversión la presenta el grupo de edad comprendido entre los 40 y 49 años.

Figura 21. *Diagrama de cajas de edad en función de conversión*



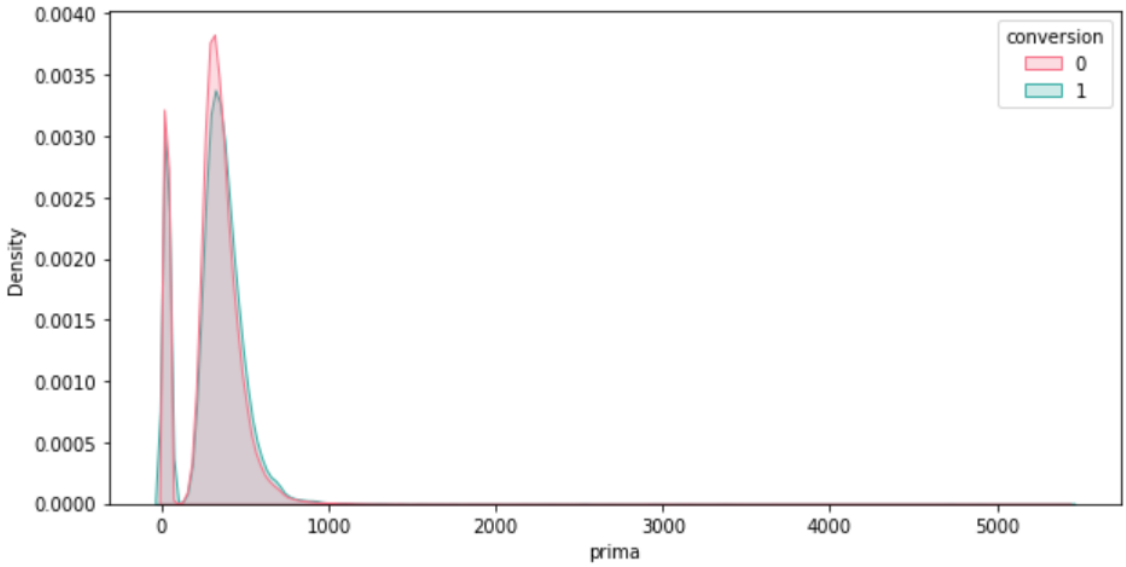
Fuente: Elaboración propia

Se ha obtenido que la edad media de las personas que deciden contratar el seguro es de 43 años, y de las que deciden no contratarlo es de 38 años. El cálculo de este momento centrado de orden 1 también es un indicador de que las personas más jóvenes de la base de datos son más propensas a convertir que las personas correspondientes a grupos de edad más mayores.

prima

La lógica inicial parece indicar que primas más bajas están más relacionadas con la contratación del seguro que primas más elevadas. En el histograma inferior se consigue apreciar levemente esta relación, porque en cuantías de primas bajas, la tasa de **no** conversión (representada en color rojo) alcanza mayor densidad de probabilidad que las tasas de conversión representadas en color verde.

Figura 22. *Histograma de prima en función de conversion*

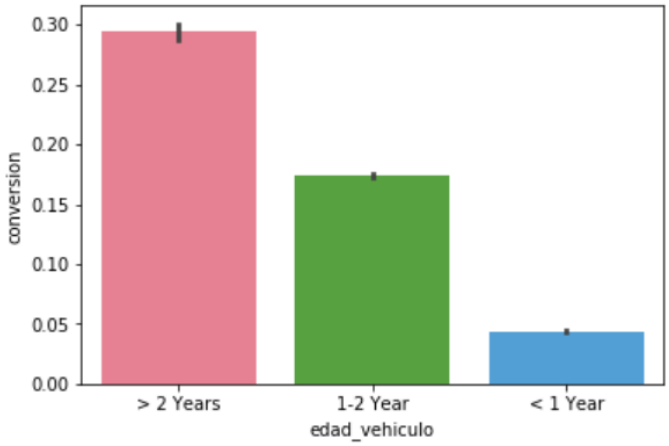


Fuente: Elaboración propia

edad_vehiculo

En el estudio inicial de las variables del capítulo 3 se observó que los vehículos cuya antigüedad es mayor que 2 años representan poca frecuencia absoluta en la base de datos y, además, es muy probable que estos vehículos hayan sufrido daños en el pasado. Como se comprobará a continuación, los vehículos con daños en el pasado están muy correlacionados con la conversión, lo que explica muy bien el diagrama inferior: a mayor antigüedad del vehículo, mayores daños en el pasado y mayor conversión.

Figura 23. *Diagrama de cajas de edad_vehiculo en función de conversion*



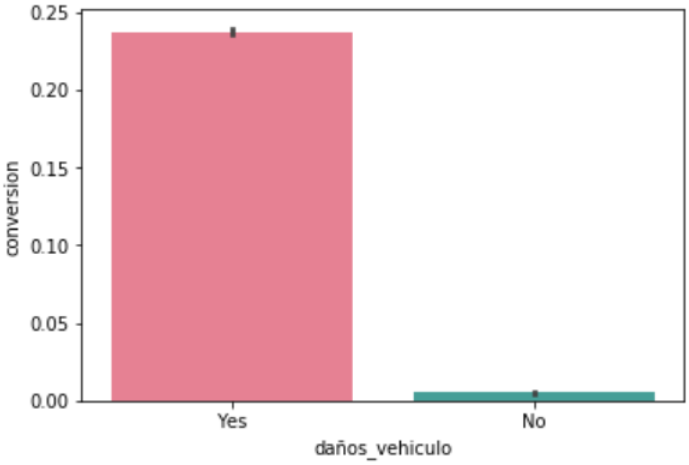
Fuente: Elaboración propia

daños_vehiculo

En el diagrama de cajas inferior se puede apreciar, como se acaba de adelantar, que las observaciones de la base de datos en las que los automóviles han sufrido daños en el pasado, están muy ligadas a la contratación del seguro.

Una posible causa reside en el hecho de que si un cliente presenta varios siniestros entonces por el sistema Bonus-Malus de las compañías de seguros, se les penalizará subiéndoles la prima o incluso impidiendo la renovación de la póliza si está dentro de los límites del contrato. Por tanto, es más probable que el cliente quiera abandonar esa compañía de seguros en búsqueda de una prima más atractiva a la que probablemente convierta.

Figura 24. *Diagrama de cajas de daños_vehiculo en función de conversion*

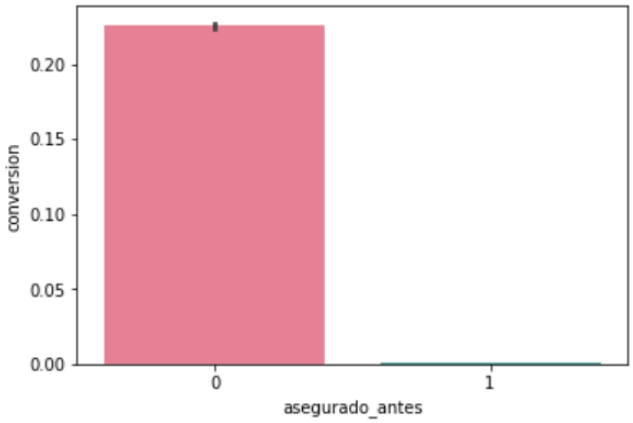


Fuente: Elaboración propia

asegurado_antes

Obviamente, para las personas que no poseían un seguro de automóvil anteriormente resulta más lógico que decidan contratar el seguro en comparación con las personas que ya están aseguradas. Esta relación se manifiesta en el diagrama inferior:

Figura 25. *Diagrama de cajas de asegurado_antes en función de conversion*



Fuente: Elaboración propia

carnet_conducir

Es importante tener en cuenta antes de observar el diagrama de cajas que el 99,79% de las personas de la base de datos poseen licencia de conducir.

Entre el porcentaje restante de los que no han obtenido la licencia, el 94,94% no ha contratado el seguro frente al 87,72% de los que sí han obtenido la licencia necesaria para conducir.

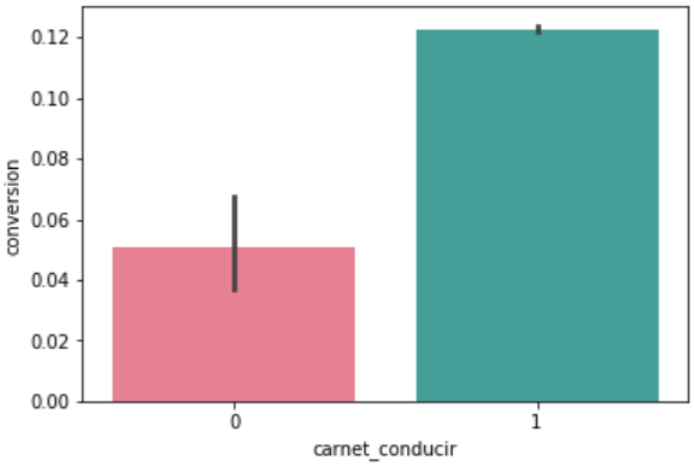
Tabla IV. *Proporción de carnet_conducir en función de conversion*

carnet_conducir	0	1
conversion		
0	0.949507	0.877283
1	0.050493	0.122717

Fuente: Elaboración propia

Por tanto, es razonable que las personas sin posesión del carnet de conducir no sean propensas a convertir.

Figura 26. *Diagrama de barras de carnet_conducir en función de conversion*

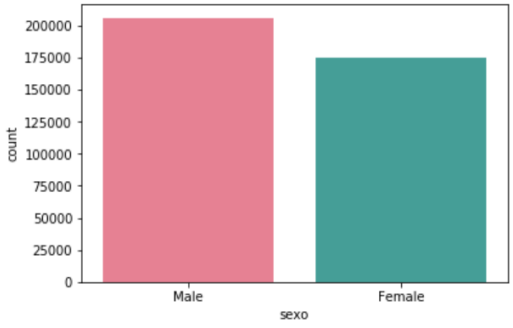


Fuente: Elaboración propia

sexo

En base a la figura inferior se puede concluir que la tasa de conversión es ligeramente más alta en el género masculino en comparación con el género femenino: un 60% frente a un 40% aproximadamente.

Figura 27. *Diagrama de barras de sexo en función de conversion*



Fuente: Elaboración propia

6.2. Análisis de correlación

Una vez que se ha estudiado las características de cada variable y se ha observado de manera gráfica las posibles relaciones entre ellas, para inferir la fuerza de esa relación es muy útil realizar un análisis de correlación.

El resultado de este análisis se tendrá especialmente en cuenta en la selección de las variables predictoras para los modelos. Generalmente, un modelo con pocas variables y gran variabilidad de los datos presentará tanta información como un modelo con muchas más variables y niveles similares de variabilidad.

Para realizar el análisis de correlación, hay que tener en cuenta la naturaleza de cada variable para aplicar de forma adecuada el coeficiente de correlación que cumpla las hipótesis de las variables de estudio. A continuación, se señalan las principales hipótesis de cada uno de los coeficientes de correlación para justificar la elección del coeficiente de correlación de **Cramer:**

1. Coeficiente de correlación de **Pearson** ($\rho_{X,Y}$)

   Dadas 2 variables aleatorias unidimensionales X e Y, el coeficiente de correlación de Pearson está definido por:

   $$\rho_{X,Y} = \frac{Cov\ (X,\ Y)}{\sqrt{Var(X)\ Var(Y)}}$$

   Hipótesis:

   - Variables continuas
   - Relación lineal entre las variables
   - Ausencia de valores atípicos
   - Variables distribuidas normalmente

   En base al análisis inicial de las variables realizado en el capítulo tercero, este coeficiente no resulta adecuado al caso de estudio ya que ninguna de las variables cumple las hipótesis.

2. Coeficiente de correlación de **Spearman** (ρ)

   Este coeficiente es un estadístico no paramétrico ya que no exige que se cumpla la hipótesis de normalidad y permite aplicarlo a variables ordinales. Se define como:

   $$\rho = 1 - \frac{6\ \sum D^2}{N\ (N^2 - 1)}$$

siendo D la diferencia entre los estadísticos de orden de X – Y (donde X e Y son las variables aleatorias unidimensionales entre las que se realiza el cálculo de la correlación) y N es el número de parejas de datos.

Hipótesis:

- Variables ordinales
- Relación monótona entre las variables

El coeficiente de Spearman es una medida de relaciones monótonas. Por tanto, un valor en el coeficiente de correlación de Spearman de 0 no implica que no haya una relación entre las variables, sino que no hay una relación monótona.

Como en la base de datos no se puede garantizar una relación monótona entre las variables, el coeficiente de correlación de Cramer que a continuación se introduce resulta una medida más explicativa.

3. Coeficiente de correlación de **Cramer** (ν)

Este coeficiente permite medir la posible relación entre dos variables categóricas si estas variables categóricas están codificadas adecuadamente. Por tanto, es un estadístico no paramétrico que también se puede aplicar a variables ordinales siempre que presenten al menos 2 valores independientes (como la variable *edad* de la base de datos, por ejemplo).

Se define como:

$$\nu = +\sqrt{\frac{\chi^2 / n}{min(r-1,\ c-1)}}$$

siendo χ^2 el estadístico chi-cuadrado, n el tamaño muestral, r el número de filas de la tabla de contingencia y c el número de columnas.

Por definición, ν toma valores reales comprendidos entre 0 y 1. Una correlación de 0 indica que no existe correlación, y una correlación de 1 representa una correlación perfecta. Destacar que al coger siempre el valor positivo del resultado de la raíz cuadrada, no se puede distinguir si la correlación es inversa o directa.

Hipótesis:

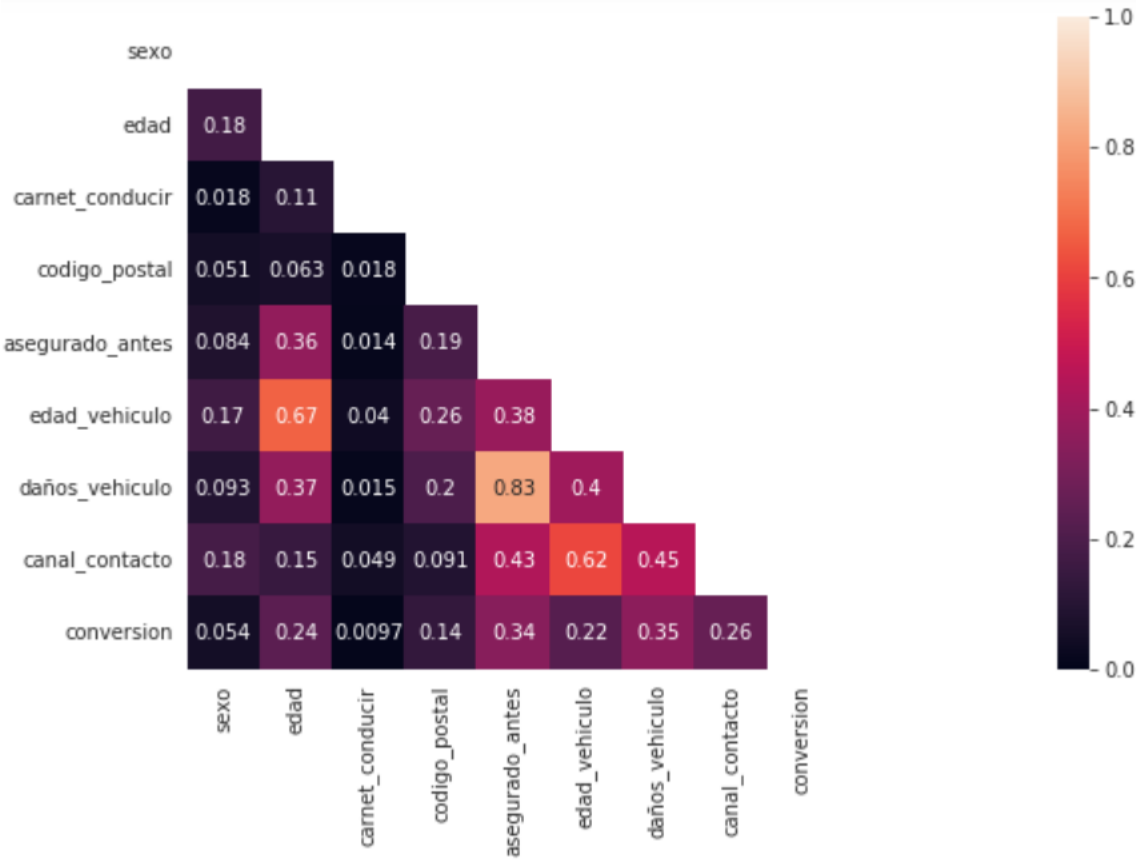
- Variables ordinales, o variables categóricas que presenten un orden en su codificación
- Variables que posean como mínimo 2 valores independientes

Teniendo en cuenta las hipótesis del coeficiente de correlación de Cramer, se podrá aplicar a todas las variables menos a la prima, ya que es una variable

continua. El resto de variables son variables categóricas excepto la edad que es una variable ordinal ya que toma valores enteros.

Previamente a calcular la matriz de correlaciones de Cramer para cuantificar las fuerzas de las relaciones que se han encontrado en los gráficos de los apartados anteriores, ha sido necesario codificar y estandarizar las variables correspondientes que han cumplido las hipótesis. Este preprocesamiento para que cada variable presente un orden lógico y esté en una escala adecuada se explicará en los siguientes apartados.

Figura 28. *Matriz de correlaciones de Cramer junto con mapa de calor*



Fuente: Elaboración propia

En el capítulo 3 se comprobó que la distribución de las variables *daños_vehiculo* y *asegurado_antes* es una Bernoulli con medias 0.458 y 0.504, respectivamente. A pesar de que estas 2 variables poseen frecuencias similares en sus 2 categorías, en el apartado anterior se observó una clara relación entre los vehículos que habían presentado daños en el pasado con la conversión, y entre los que no estaban asegurados previamente con la conversión. La matriz de correlaciones corrobora que existe una correlación moderada entre estas variables con la conversión. Además, existe una correlación muy fuerte (0.83) entre las 2 variables.

En la matriz de correlaciones también se aprecian correlaciones fuertes entre *edad* y *edad_vehiculo* (0.67) y entre *canal_contacto* y *edad_vehiculo* (0.62). La primera correlación probablemente indica que las personas más mayores tiendan a tener vehículos más antigüos. Sin embargo, es más difícil establecer a priori una relación entre vehículos más nuevos o antigüos, y entre ponerse en contacto con la aseguradora mediante email, teléfono, página web, agentes, corredores, etc.

En la matriz de correlaciones, también caben destacar correlaciones moderadas entre la variable *canal_contacto* con *daños_vehiculo* (0.45) y con *asegurado_antes* (0.43). El resto de variables presentan correlaciones más débiles entre ellas y con la variable objetivo (*conversion*), excepto las **variables *sexo* y *carnet_conducir*** que aportan muy poca información ya que tienen una **correlación inexistente** con *conversion*, al igual que con el resto de variables.

Por tanto, en el algoritmo de selección de las variables predictoras probablemente se obtenga como resultado que las variables *sexo* y *carnet_conducir* no son significativas a la hora de predecir la conversión y, por ende, los resultados de los modelos no varíen notoriamente al excluir estas variables.

6.3. Preprocesamiento de los datos

Una vez se ha realizado el análisis exploratorio de los datos, el siguiente paso es preparar las variables que se utilizarán en los modelos para predecir la tasa de conversión.

En primer lugar, se han eliminado las variables *id* y *dia* ya que no aportan ningún carácter predictivo:

- La **variable *id*** se comporta como un índice del conjunto de datos: indica el número de identificador del cliente con números enteros positivos, enumerando cada registro de las 381109 observaciones que componen la base de datos. Por tanto, no es significante para predecir la conversión.

- La **variable *dia*** como se estudió en el apartado 3.2, indica la cantidad de registros de cada día obtenidos entre los 300 días de recogida de datos. Se comprobó que sigue una distribución uniforme, es decir, en cada día se obtuvo una cantidad de datos similar. Además, la tasa de conversión también presenta un comportamiento uniforme en cada uno de los 300 días, por lo que no proporciona ninguna información que ayude a predecir la conversión.

La base de datos no contiene ningún dato faltante en las variables, por lo que no ha sido necesario realizar la limpieza de datos y evaluar qué procedimiento seguir con esos datos faltantes.

En la variable *prima* se encuentran numerosos datos anómalos que sobresalen de los demás. Estos datos atípicos u *outliers* podrían distorsionar el resultado de los modelos predictivos, por lo que ha sido necesario tratarlos adecuadamente. La base de datos inicial recoge 381109 observaciones y se han identificado 98 observaciones atípicas con el método de Tukey, lo que supone el 0.02% del total. Por tanto, se ha decidido eliminar esos registros ya que no supondría mucha pérdida de información.

Como se ha indicado en el apartado anterior, antes de llevar a cabo el análisis de correlaciones ha sido necesario codificar y estandarizar las variables de distintas maneras de acuerdo a la naturaleza de cada variable y a sus características. En los siguientes subapartados se explicará en mayor profundidad el procedimiento llevado a cabo.

6.3.1. Codificación de las variables

Para empezar con la codificación, hay que distinguir las variables en función de su tipo:

- Las variables ***sexo***, ***daños_vehiculo*** y ***edad_vehiculo*** son variables puramente categóricas ya que sus valores son cadenas de caracteres. Estas variables es necesario transformarlas ya que en la mayoría de modelos predictivos no se permiten variables explicativas cuyos valores sean palabras.

  El procedimiento que se llevará a cabo se conoce como ***Label Encoder***: simplemente consiste en reemplazar cada una de las cadenas de caracteres por números enteros. Así, los valores de estas variables categóricas se convertirán en números que seguirán un orden lógico. En vez de usar funciones ya creadas en librerías, se ha programado de manera sencilla.

  A continuación, se muestran las transformaciones realizadas para estas 3 variables:

  - La variable ***sexo*** toma los valores *Female* y *Male*, que se convertirán en 0 y 1 respectivamente.
  - La variable ***daños_vehiculo*** toma los valores *No* y *Yes*, que se convertirán a 0 y 1 respectivamente.
  - La variable ***edad_vehiculo*** presenta una clasificación de tres categorías cuyos valores son cadenas de caracteres. Estas cadenas *"< 1 Year*", "*1-2 Year*" y *"> 2 Years*", se transformarán en 0, 1 y 2 respectivamente.

En vez de realizar este procedimiento, también se probó a aplicar otro tratamiento conocido como *One-Hot Encoding* que consiste en crear variables indicadoras de cada categoría (también conocidas como variables *dummy*).

Sin embargo, la aplicación de ***One-Hot Encoding*** **perjudicaría** a modelos como los árboles de decisión. Su ejecución no sería tan exitosa ya que las variables *dummy* tendrán un alto porcentaje de ceros, por lo que estarán altamente desequilibradas.

Además, **no es buena opción** aplicar ***One-Hot Encoding*** a las variables *sexo* y *daños_vehiculo* porque ya están codificadas como 0 y 1, siendo indicadores respectivamente de si el género es masculino y de si se han producido daños en el vehículo. En el caso de que no se tome el valor 1, se estaría indicando el género femenino y la ausencia de daños, por lo que no resulta necesario crear variables indicadoras adicionales para estas categorías.

Respecto a la variable *edad_vehiculo*, la **desventaja** de aplicarle ***One-Hot Encoding*** es que se perdería el orden lógico que presentan sus categorías, ya que indican la antigüedad del vehículo de manera organizada. Al haber aplicado *One-Hot Encoding* se hubiera perdido esta información.

- Las variables ***codigo_postal*** y ***canal_contacto*** son variables numéricas, pero de naturaleza categórica porque están constituidas por una cantidad muy elevada de valores enteros únicos de los que se desconoce su verdadera información al estar anonimizados.

  El procedimiento ideal hubiese sido agrupar las categorías de estas variables de manera que se formaran grupos que compartiesen características similares. Sin embargo, al estar las categorías anonimizadas no se pueden establecer relaciones entre ellas.

  En todos los modelos que se emplearán posteriormente para predecir la conversión excepto para el modelo *CatBoost*, es necesario agrupar las categorías de las variables con alta cardinalidad. En cambio, modelos como ***LightGBM*** y ***CatBoost*** ya tienen implementado algoritmos para tratar adecuadamente las variables que presentan muchas categorías. El tratamiento que se llevará a cabo de hecho proviene de la librería CatBoost, y se conoce como *CatBoost Encoder* o también como *Ordered Target Statistics*.

  La idea detrás de ***CatBoost Encoder*** es reflejar en las categorías de las variables con alta cardinalidad un orden relacionado con la variable objetivo. En una primera intuición, se reemplazaría cada categoría con la media de los valores de la variable objetivo para esa categoría.

Esto podría llevar a un **sobreajuste** de los datos porque al codificar categorías con muy poca frecuencia en los datos de entrenamiento, es más improbable que ese comportamiento se reproduzca en los datos de prueba y, por tanto, la codificación no sería la adecuada.

Ese procedimiento también se podría considerar inadecuado porque al ser la variable objetivo binaria, el valor codificado revelaría directamente el valor esperado de la variable que se va a predecir para esa categoría. La mejora consistiría en codificar un conjunto de datos de entrenamiento más pequeño, para que así sea más complicado deducir el valor de la variable objetivo en la base de datos (Biarnes, 2021; Hemavati, 2021).

La **solución** para minimizar este problema es en vez de aplicarlo a todos los datos de entrenamiento a la vez, ir codificando dato a dato de cada variable que se quiere transformar. Esta codificación se realiza aplicando el estadístico que a continuación se explicará, hasta codificar todos los valores de la variable de la base de datos que es de interés.

La fórmula específica del **estadístico** que se usa en *Ordered Target Statistics* es la siguiente:

$$\frac{SumaTarget + constante}{ContadorCategoria + 1}$$

siendo:

- SumaTarget: La suma de los valores que toma la variable *conversion* en las observaciones ya codificadas. El valor para el primer dato que se codifica es 0.

- constante: Media de la variable *conversion* en toda la base de datos. Es decir, suma de todos los valores de *conversion* dividido por el número de datos.

- ContadorCategoria: Número de veces en las observaciones ya codificadas y en la que se está codificando, que la variable que se quiere transformar (en este caso, *codigo_postal* o *canal_contacto*) indica esa categoría.

Posteriormente, se repetirá numerosas veces esta metodología mediante **permutaciones aleatorias** de los datos de la variable que se está transformando. Finalmente, se aplicará la media de dichas permutaciones para codificar las distintas categorías.

Hay que tener en cuenta que el primer subconjunto de datos aleatorios que se codificaría siguiendo esta metodología tendría por tanto gran varianza en comparación a los siguientes subconjuntos de datos aleatorios y, por tanto, no se estaría codificando adecuadamente.

Para solucionar esto, se repite numerosas veces esta metodología mediante permutaciones aleatorias de los datos de la variable que se está transformando y el resultado de la codificación para cada categoría se estima con el **promedio de las permutaciones**. De esta manera también se evita el problema indicado que podría producirse al codificar cada una de las categorías de todos los datos de entrenamiento a la vez (Biarnes, 2021; Hemavati, 2021).

Por tanto, se espera que el resultado para cada categoría converja con la fórmula superior en el caso de que el número de permutaciones realizadas haya sido suficientemente grande. Y en ese caso, el valor de esta transformación para cada categoría es el que se aplicará a los datos de prueba de los que en principio obviamente se desconoce los valores de la variable que se predice.

6.3.2. División de la base de datos en *training set* y *test set*

Para evaluar correctamente qué modelo es el que predice mejor, se dividirá la base de datos en ***training set*** y ***test set***. El ***training set*** se usará para entrenar los modelos y el ***test set*** será útil para evaluar la capacidad predictora de cada uno de los modelos. El ***training set*** comprenderá el 80% de la base de datos y el ***test set*** el 20% restante. La división se ha realizado de manera aleatoria.

Teniendo en cuenta el apartado anterior, la codificación de las variables *codigo_postal* y *canal_contacto* es necesario realizarla después de dividir la base de datos entre el conjunto de datos de entrenamiento y el conjunto de datos que se usará para evaluar la capacidad predictora de los modelos.

Es muy importante no aplicar directamente *CatBoost Encoder* a los datos que luego vamos a utilizar como test, ya que estaríamos teniendo en cuenta los valores de la variable que se va a predecir en la modificación de la muestra que se usará para evaluar el mejor modelo de predicción. Por tanto, se hará la codificación a los datos de entrenamiento y en función de ello, se aplicará la transformación resultante a los datos de prueba.

Por todo ello, el orden de los pasos realizados en el preprocesamiento ha sido:

1. Codificación de *sexo*, *daños_vehiculo* y *edad_vehiculo*

2. División de la base de datos
3. Codificación de *codigo_postal* y *canal_contacto*
4. Estandarización de *edad* y *prima*

Se conoce como ***data leakage*** al uso de información que no provenga del conjunto de datos de entrenamiento para entrenar los modelos de predicción. De esta manera al igual que ocurre con la codificación de *codigo_postal* y *canal_contacto*, no se pueden estandarizar las variables teniendo en cuenta la escala de las variables de los datos de prueba. Esta es la razón por la que la estandarización también se debe realizar después de dividir la base de datos.

6.3.3. Estandarización

Las variables ***edad*** y ***prima*** son variables que toman valores numéricos y poseen una escala muy diferente al resto de variables presentes en la base de datos. Además, los algoritmos de los modelos predictivos que se ejecutarán funcionan mejor si las variables están en una escala relativamente similar.

Por esta razón se procede a estandarizar estas 2 variables de la manera más apropiada, para que posean una medida parecida al resto de variables. Los 2 procedimientos que se tendrán en cuenta para escalar son:

- ***MinMaxScaler***: Es el procedimiento más usado, ya que consiste simplemente en restar el valor mínimo de la variable a estandarizar y luego dividir por la diferencia entre el máximo original y el mínimo original.

- ***RobustScaler***: Consiste en eliminar la mediana y dividir por el rango intercuartílico. De esta manera los valores más anómalos tienen menos influencia que con *MinMaxScaler* y el rango también es mayor que con *MinMaxScaler*.

En base al *boxplot* de *prima* del capítulo 3 se recuerda que esta variable tiene una larga cola derecha que hay que tener en cuenta en la estandarización. Sin embargo, previamente se han eliminado los valores atípicos de la variable *prima*.

Por esta razón, el procedimiento que se ha considerado más adecuado para estandarizar esta variable ha sido la aplicación de *MinMaxScaler* en lugar de *RobustScaler* como se hubiera realizado en caso de que no se hubiese decidido eliminar las observaciones relacionadas con los *outliers*.

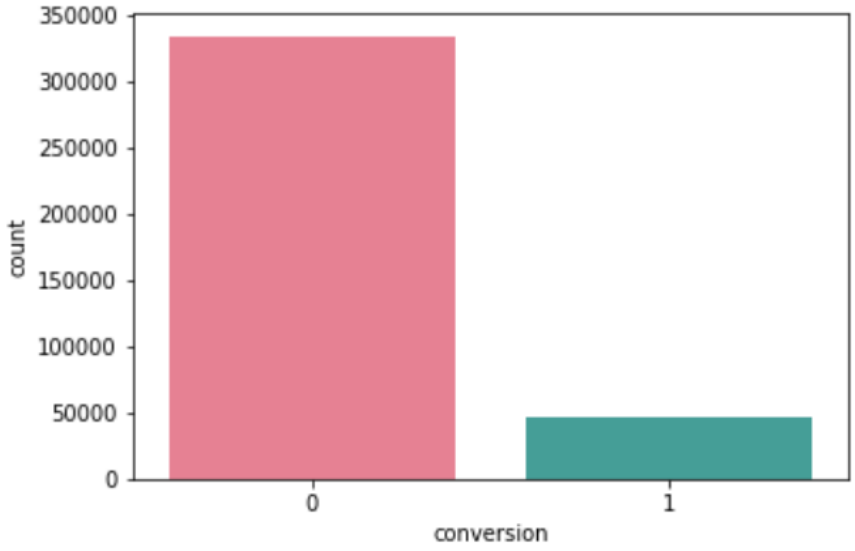
En cuanto a la variable *edad*, también es necesario escalarla para que posea una medida similar al resto de variables. Por ello se aplicará *MinMaxScaler*, de manera que los valores de esta variable quedarán comprendidos en el intervalo [0,1].

Por último, señalar que si se hubiera decidido aplicar *RobustScaler* a *prima* los valores resultantes quedarían centrados en el valor 0. Como la variable *prima* no toma valores negativos, a todos los valores obtenidos en el escalamiento se les habría que sumar el valor más negativo obtenido (que es el valor mínimo obtenido con la estandarización) para que todos resultasen positivos.

6.3.4. Aplicación de técnica de balanceado de datos

A continuación, se puede observar como la variable que vamos a predecir presenta una gran cantidad de 0 en comparación a los 1. Esto es lógico ya que en las situaciones reales lo normal es que se decida no contratar el nuevo seguro.

Figura 29. *Diagrama de barras de conversion*



Fuente: Elaboración propia

La submuestra de observaciones en las que la variable *conversion* toma el valor 1 es muy pequeña en comparación a la submuestra con *conversion* igual a 0. Por tanto, se ha realizado el balanceado de los datos de entrenamiento aplicando el **algoritmo *smote***, para que ambas submuestras tengan la misma cantidad de observaciones.

Al balancear los datos hay que destacar que no se deben balancear los datos de prueba. Suena tentador realizarlo ya que evaluar la capacidad predictiva de los modelos en datos de prueba no balanceados no es tarea sencilla. Por ejemplo, modelos que predigan todo ceros tendrán una tasa alta de aciertos debido a que normalmente sucede que el cliente no

convierte. Sin embargo, las medidas que se emplearán también tendrán muy en cuenta lo bien que se han predicho los casos en los que el cliente convierte.

6.4. Selección de variables predictoras

En el análisis de correlaciones ya se reveló la importancia de una buena elección de las variables que se van a utilizar para predecir la conversión en los modelos. En concreto, se destacó que un modelo con pocas variables y gran variabilidad de los datos presentará tanta información como un modelo con muchas más variables y niveles similares de variabilidad. Por eso es muy útil descartar aquellas variables que no aportan información adicional.

Con ese fin, se ejecutará el posible mejor algoritmo para seleccionar las variables predictoras: ***BorutaShap***.

Lo que diferencia a *BorutaShap* de otros algoritmos es cómo se evalúa la importancia de cada una de las variables. Habitualmente, esto se realiza con algoritmos que suelen usar árboles donde las variables compiten entre ellas para destacar cuáles son las que mejor capacidad predictiva poseen (Trevisan, 2022).

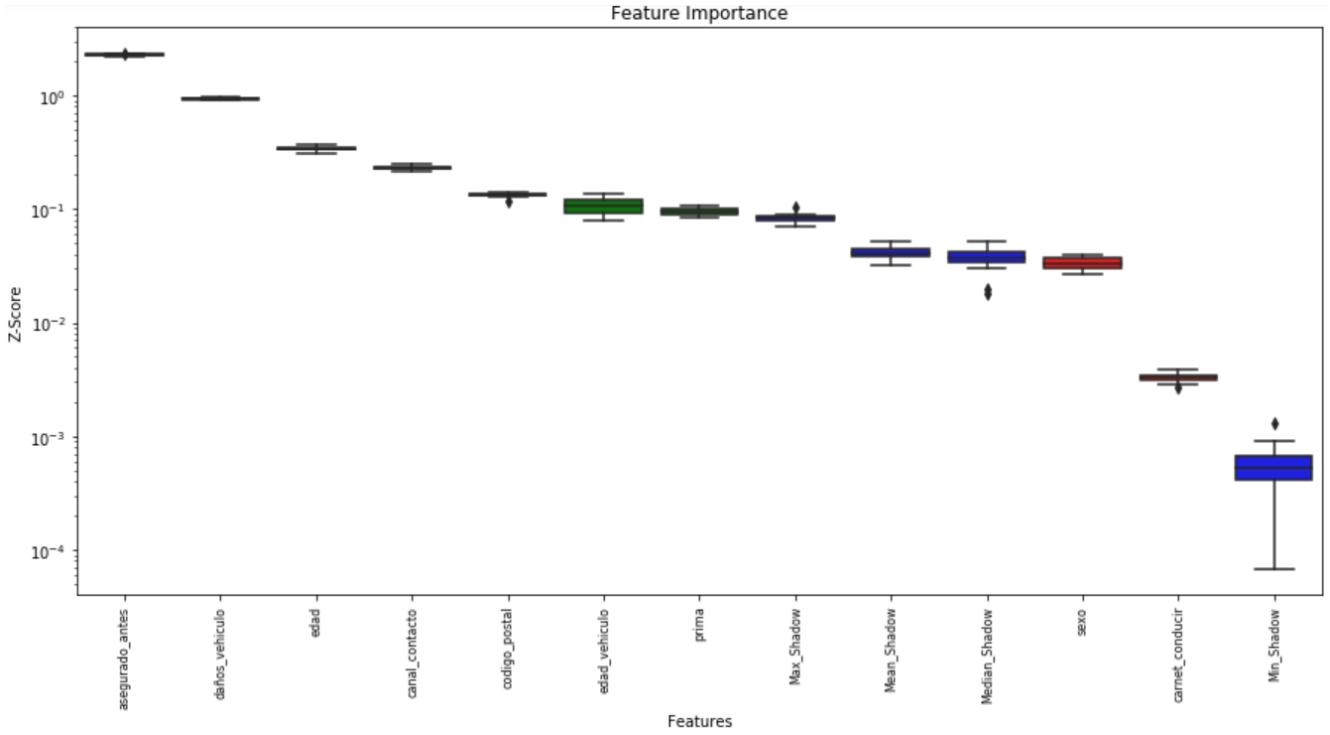
Sin embargo, al emplear *BorutaShap* las variables compiten contra una **versión aleatorizada** de ellas mismas en la que no se tiene en cuenta la correlación de esa variable con la conversión. De esta manera, el primer paso es crear otra base de datos que incluye también a la versión aleatorizada de cada una de las variables, duplicando así el número de variables en la base de datos. A las versiones aleatorizadas de las variables se las conoce con el nombre de ***shadow***.

El siguiente paso es evaluar con el modelo predictivo que se prefiera, si cada variable ha predicho mejor que su versión aleatorizada considerando un cierto umbral. En este caso se usará el modelo *XGBoost* debido a sus buenos resultados y a su fácil implementación.

Realizando 20 iteraciones, se considera una distribución binominal cuyos parámetros son: el número de veces que se repite el experimento (es decir, 20) y la probabilidad de éxito 0.5. Así, se obtienen los Z-Scores para cada variable indicando su importancia.

El resultado final del algoritmo BorutaShap viene dado por este gráfico:

Figura 30. *Boxplot de BorutaShap para la selección de las variables predictoras*



Fuente: Elaboración propia

Los *boxplots* de los Z-Scores de las medidas principales de las **versiones aleatorizadas** (conocidas como *Shadow*) vienen representados en **color azul**. Estas medidas son el máximo, mínimo, la media y la mediana de los Z-Scores de las versiones aleatorizadas.

El umbral a partir del cual se considera que una variable ha predicho mejor que su versión aleatorizada es el valor del Z-Score máximo de las versiones aleatorizadas. Por lo que se considerarán **buenos predictores** a las variables que excedan este umbral. El *boxplot* de estas variables viene representado en **color verde**.

Teniendo lo anterior en cuenta, las mejores variables predictoras son *asegurado_antes*, *daños_vehiculo*, *edad*, *canal_contacto*, *codigo_postal*, *edad_vehiculo* y *prima*.

En contraste, las variables cuyo Z-Score se aleja notablemente del valor del Z-Score máximo de las versiones aleatorizadas vendrán caracterizadas por *boxplots* de **color rojo**. Estas variables son *sexo* y *carnet_conducir* que, como ya se sospechaba en el análisis de correlaciones por su inexistente correlación con la variable *conversion*, son **malos predictores**.

Por todo ello, en el siguiente capítulo en el que se mostrarán los resultados de los modelos de conversión se excluirán las variables *sexo* y *carnet_conducir* por su poco

carácter predictor, tal y como ha deducido uno de los mejores algoritmos (*BorutaShap*) corroborando los hallazgos del estudio de correlación.

7. RESULTADOS DE LOS MODELOS

Una vez que se ha realizado una exposición teórica de los modelos de Inteligencia Artificial y de las medidas de evaluación para elegir el mejor modelo, en este capítulo se pondrán en práctica los conceptos anteriores aplicados a la base de datos preprocesada en el capítulo anterior.

El objetivo principal de los modelos es el de predecir la tasa de que se produzca el *cross-selling* e identificar las características de las personas y de sus vehículos que contratarían el nuevo seguro de automóviles. Se recuerda que si el cliente ha convertido la variable conversión toma el valor 1, y si el cliente no ha convertido, la variable conversión es 0.

Por tanto, además de identificar a las personas que es más probable que contraten el nuevo seguro, se debe **minimizar el error tipo II** que se produce cuando se predice que no van a contratar el nuevo seguro personas que están interesadas en él.

Así, las medidas de evaluación que se tendrán más en cuenta son:

- **Recall**, dado que es el porcentaje de aciertos de conversión de todas las personas que han decidido realizar el *cross-selling*.

- **Exactitud balanceada**, dado que corrige el problema de aplicar la definición de exactitud usual, en la que se obtendrían muy buenos resultados si el modelo siempre prediciera 0.

- El valor del **Área Bajo la Curva** (AUC en inglés), ya que se obtiene al hallar el área debajo de la curva ROC calculada al representar la precisión frente a la proporción de falsos positivos.

Estas medidas son las más relevantes puesto que la clase de ceros en la variable conversión representa una alta proporción en comparación con la clase de unos. Una vez que se haya terminado este trabajo y el mejor modelo haya sido elegido, en la vida real el porcentaje de conversión es muy bajo por lo que es necesario evaluar los modelos en este escenario real. Por ello, las medidas de evaluación anteriores son las más adecuadas.

Cabe destacar que el éxito en obtener buenos modelos con una alta capacidad predictiva en los que ni se sobreajuste a los datos ni se subajuste, reside en que éstos posean la **máxima calidad** posible. Deben estar codificados, estandarizados y con el correspondiente tratamiento de los datos faltantes en base a las necesidades de cada modelo. Esto se ha realizado cuidadosamente en el capítulo anterior.

Se han entrenado numerosos modelos de Inteligencia Artificial como se puede comprobar en el Anexo al final del trabajo. Sin embargo, en este capítulo se expondrán con más detalle los resultados de los mejores modelos.

7.1. Regresión logística

Las variables explicativas del modelo final de la regresión logística son las mismas que las obtenidas con el algoritmo *BorutaShap* en el capítulo anterior: *edad*, *codigo_postal*, *asegurado_antes*, *edad_vehiculo*, *daños_vehiculo*, *prima* y *canal_contacto*. En la siguiente tabla se muestran sus coeficientes correspondientes. El valor del intercepto es de -2.2918891.

Tabla V. *Coeficientes de la regresión logística*

edad	-1.45158436
codigo_postal	2.87787373
asegurado_antes	-3.99481241
edad_vehiculo	0.3674897
daños_vehiculo	1.91105928
prima	-0.92844106
canal_contacto	6.00447871

Fuente: Elaboración propia

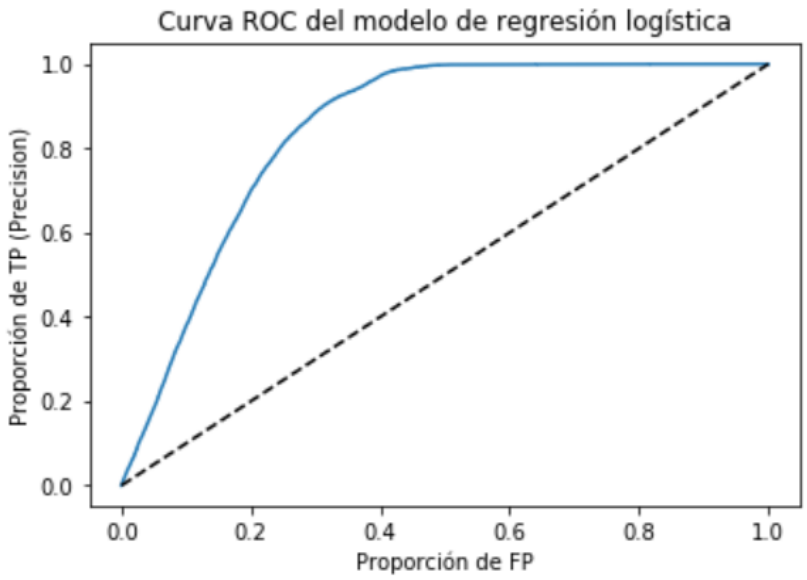
Las variables cuyos coeficientes de regresión asociados son positivos, indicando que aumentan la probabilidad de que ocurra el *cross-selling* son *edad_vehiculo*, *daños_vehiculo*, *canal_contacto* y *codigo_postal*. Es decir, la probabilidad de contratar el nuevo seguro aumenta notablemente si el vehículo ha sufrido daños y si la antigüedad del vehículo es mayor, como ya se adelantó en el análisis de correlación y en el análisis de las relaciones de las variables con la conversión.

El algoritmo que se ha utilizado para estimar los coeficientes no ha sido el habitual de máxima verosimilitud, sino que se ha empleado un algoritmo de descenso de coordenadas (CD) basado en la excelente biblioteca C++ liblinear (R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, 2008). Además, también se ha aplicado regularización para evitar que el modelo se sobreajuste a los datos y capte mejor las relaciones más relevantes[8].

En la figura inferior se muestra la **curva ROC** obtenida. El área bajo esta curva ha sido de 0.845484, un resultado bastante bueno.

[8] Para conocer la función que se minimiza visitar: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

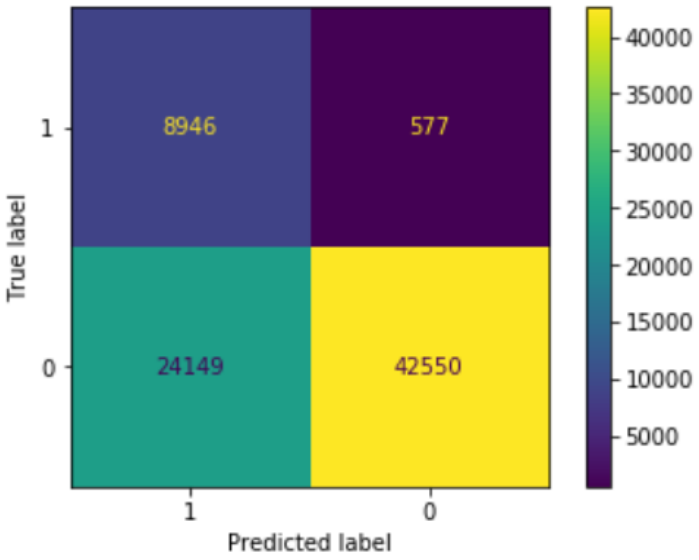
Figura 31. *Curva ROC del modelo de regresión logística*



Fuente: Elaboración propia

A continuación, se muestra la **matriz de confusión** en base a la cual se obtienen el resto de medidas de evaluación:

Tabla VI. *Matriz de confusión de la regresión logística*

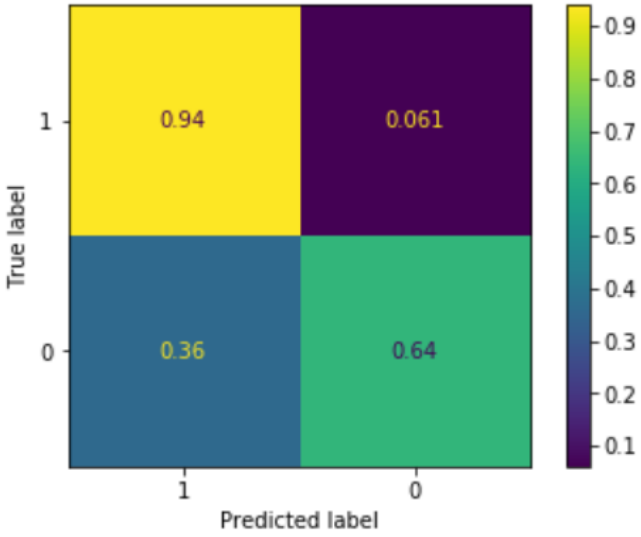


Fuente: Elaboración propia

Idealmente todos los datos deberían estar clasificados en la diagonal que va desde la esquina superior izquierda hasta la esquina inferior derecha, es decir, desde los verdaderos positivos hasta los verdaderos negativos. Y la diagonal contraria idealmente estaría compuesta por ceros.

Para interpretar mejor los resultados de la matriz de confusión, en vez de mostrar la frecuencia absoluta de cada categoría, se obtendrá la **matriz de confusión normalizada**. Así, es más sencillo evaluar el porcentaje de verdaderos positivos que se ha conseguido predecir y cuantificar mejor los errores de la diagonal contraria.

Tabla VII. *Matriz de confusión normalizada de la regresión logística*



Fuente: Elaboración propia

Se observan muy buenos resultados en la matriz superior:

- El valor de **recall** es de 0.94, obtenido a partir de la primera matriz de confusión haciendo $\frac{8946}{8946+577}$.

- El error más importante en el que se predice que no van a convertir personas que están interesadas en el nuevo seguro, ha ocurrido muy pocas veces.

- La **exactitud balanceada** es 0.78867 = $\frac{0.94+0.64}{2}$, es decir, la media del recall en cada clase. Aunque en el caso de estudio es más importante obtener un valor mucho más alto de recall asociado a las personas intersadas en el nuevo seguro en comparación con las que no.

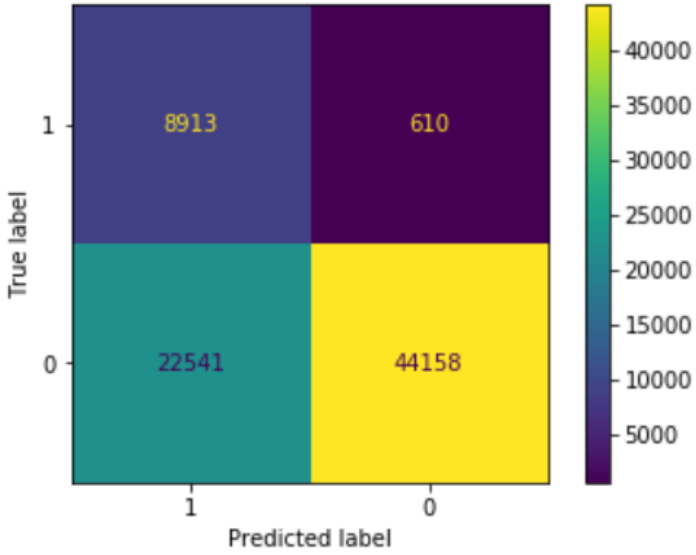
7.2. Random Forest

El entrenamiento del modelo Random Forest se ha realizado junto con una optimización de los hiper parámetros mediante el método "*Randomized Parameter Optimization*" también perteneciente a la librería *scikit-learn* (Bergstra & Bengio, 2012).

En vez de que el programador fije de antemano los posibles mejores parámetros y entrene varias veces el modelo para buscar la mejor combinación paramétrica, lo que realiza *Randomized Parameter Optimization* es una búsqueda aleatoria de los hiper parámetros. Cada combinación paramétrica es el resultado de un muestreo a partir de una distribución sobre posibles valores de los parámetros, que posteriormente se evalúa con validación cruzada.

Los resultados de las predicciones del modelo Random Forest se muestran en la siguiente matriz de confusión:

Tabla VIII. *Matriz de confusión del modelo Random Forest*



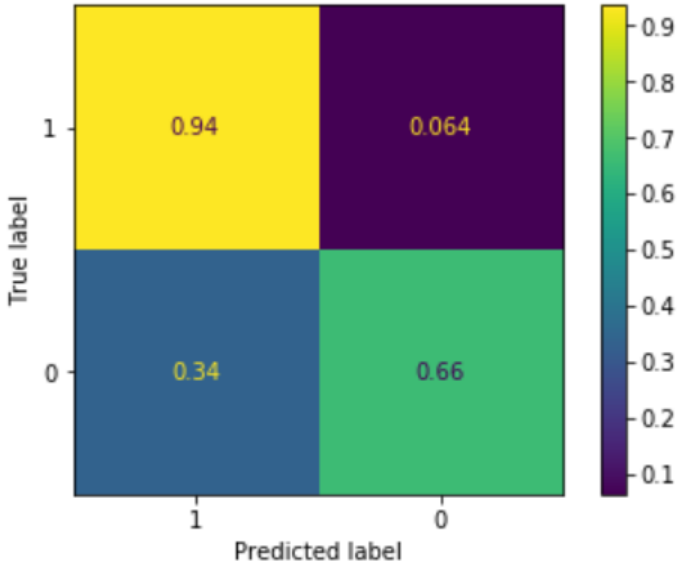
Fuente: Elaboración propia

Se observa que la mayor frecuencia absoluta corresponde cuando el verdadero valor es 0, y la predicción también ha sido 0, siendo esta predicción la no contratación del nuevo seguro. Sin embargo, 22541 veces que el cliente no estaba interesado en realizar el nuevo contrato el modelo ha predicho que sí.

No obstante, lo más importante consiste en predecir bien los casos en los que el cliente está interesado en el *cross-selling* y minimizar el error que se produce cuando el cliente está interesado pero el modelo predice que no. Como se puede observar en la siguiente matriz de confusión normalizada se ha predicho que el cliente va a contratar el nuevo

seguro en el 94% de los casos en los que ocurre, y el **error tipo II** sólo ha ocurrido un 6.4% de las veces.

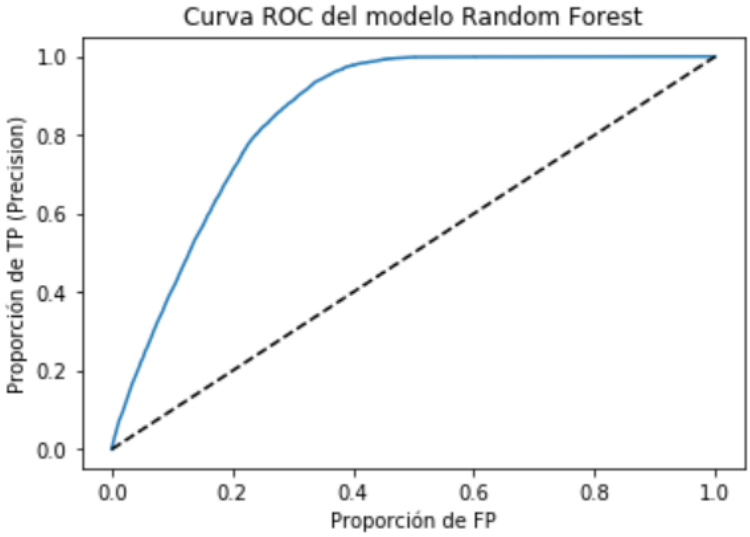
Tabla IX. *Matriz de confusión normalizada del modelo Random Forest*



Fuente: Elaboración propia

La representación de la **curva ROC** también es buena, obteniendo un valor del área bajo la curva de 0.852695.

Figura 32. *Curva ROC del modelo Random Forest*



Fuente: Elaboración propia

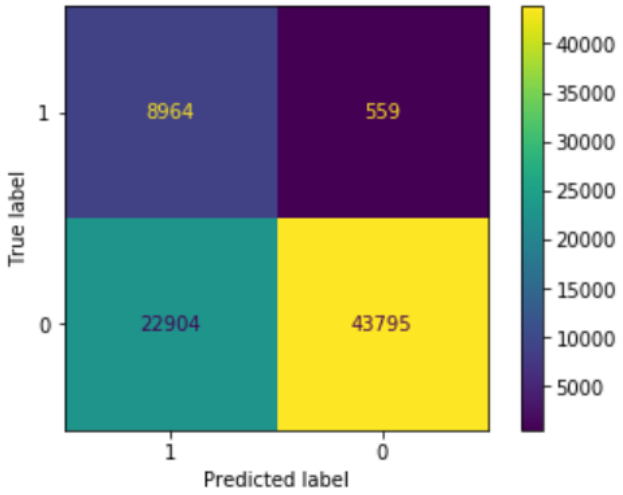
Finalmente, a partir de la matriz de confusión se pueden obtener los valores de **recall**, 0.9359, y de la **exactitud balanceada**, que es igual a 0.798996, por lo que Random Forest proporciona resultados ligeramente superiores al modelo anterior.

7.3. XGBoost

XGBoost consiste en un sistema de árboles de decisión concatenados secuencialmente para mejorar los errores de los árboles previos. Se ha utilizado el modelo implementado en la librería *XGBoost*[9] especificando el parámetro *booster* como "*gbtree*", para crear el modelo de árboles dividido a su vez en submodelos. En función de este parámetro principal se ha determinado el resto de parámetros tomando las ideas de un kernel de la plataforma Kaggle[10].

Los resultados de las predicciones del modelo XGBoost se muestran en la siguiente **matriz de confusión**:

Tabla X. *Matriz de confusión del modelo XGBoost*



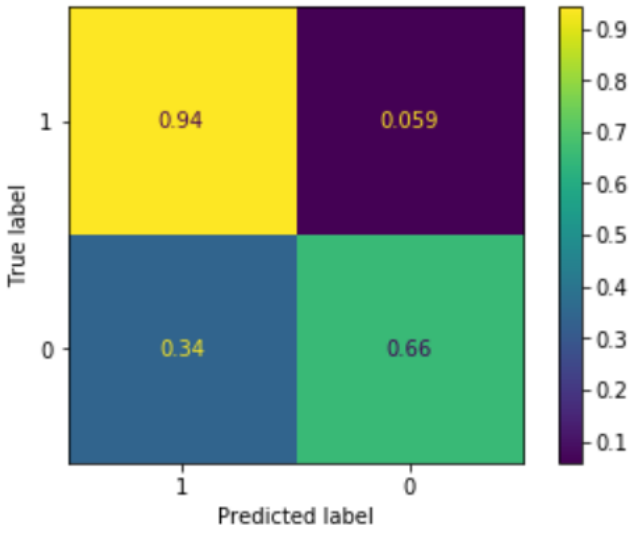
Fuente: Elaboración propia

En comparación con los anteriores modelos, XGBoost obtiene un porcentaje un poco mejor de todos los 1 que hay en los datos reales que se han acertado. Esto se debe a que se ha minimizado el error de no predecir el *cross-selling* a las personas interesadas en el nuevo contrato.

[9] https://xgboost.readthedocs.io/en/stable/

[10] https://www.kaggle.com/code/prashant111/a-guide-on-xgboost-hyperparameters-tuning/notebook#6.-References-

Tabla XI. *Matriz de confusión normalizada del modelo XGBoost*



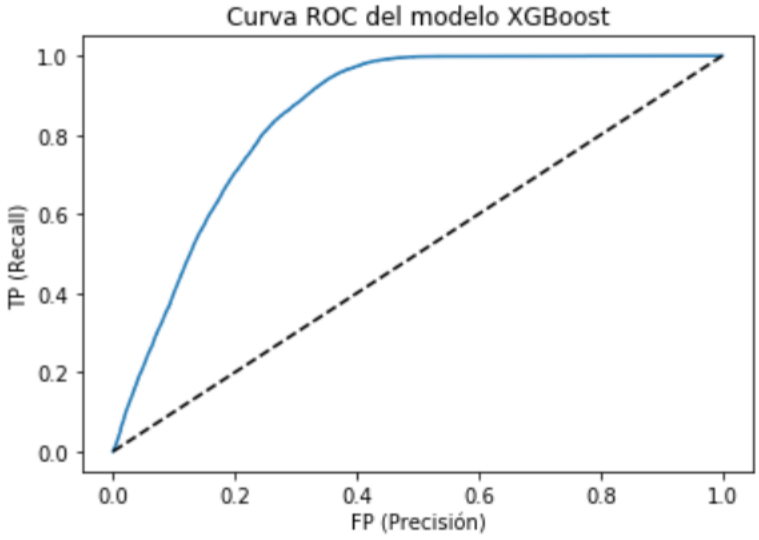
Fuente: Elaboración propia

En la tabla superior se muestra la **matriz de confusión normalizada**, en la que también se observa un porcentaje un poco mayor que el de la regresión logística de todos los 0 que hay en los datos reales que se han acertado, siendo de 0.66. A partir de estos resultados, el valor de la **exactitud balanceada** obtenido es 0.798953.

Como los modelos que se muestran son los que mejores resultados han presentado, las diferencias de este modelo en comparación a los demás se aprecia mejor en los terceros decimales como se mostrará en la tabla comparativa final.

También se obtienen resultados ligeramente mejores en comparación con el resto de modelos al calcular el **área debajo de la curva ROC**, que es de 0.853917.

Figura 33. *Curva ROC del modelo XGBoost*



Fuente: Elaboración propia

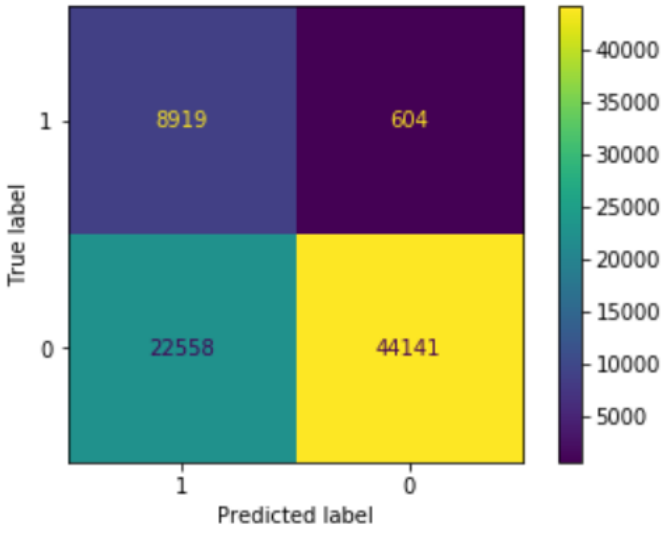
7.4. CatBoost

Al igual que XGBoost, CatBoost también consiste en un modelo formado por árboles de decisión que mejoran las predicciones de los anteriores árboles. La principal ventaja de CatBoost frente a XGBoost es que CatBoost preprocesa las variables categóricas de un modo razonable y eficiente, por tanto, no es necesario realizar esta codificación previamente. El método que se utiliza es *CatBoostEncoder* (se explicó en el capítulo anterior en el apartado de codificación de las variables categóricas).

Otras diferencias respecto a XGBoost son que los árboles de decisión que CatBoost emplea son simétricos y que, aunque CatBoost admita menos parámetros, los parámetros por defecto generalmente proporcionan muy buenos resultados. Por ello, este ha sido el único modelo de los que se muestran que, aunque se haya probado con otras combinaciones paramétricas, los parámetros por defecto son los que mejor han funcionado.

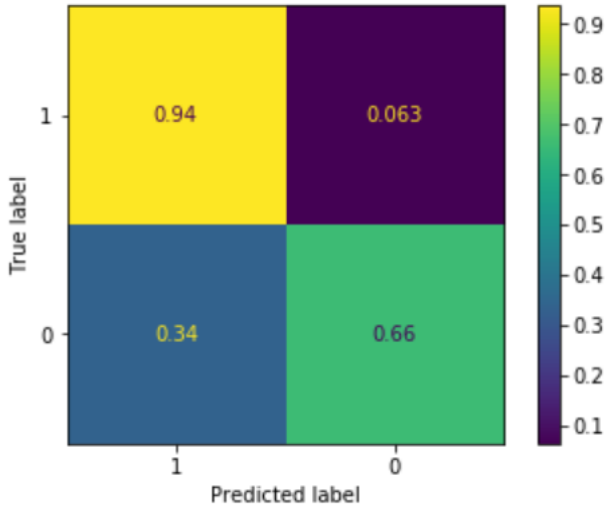
A continuación, se muestran los resultados de la **matriz de confusión** con la frecuencia absoluta en cada categoría y la **matriz de confusión normalizada**:

Tabla XII. *Matriz de confusión del modelo CatBoost*



Fuente: Elaboración propia

Tabla XIII. *Matriz de confusión normalizada del modelo CatBoost*

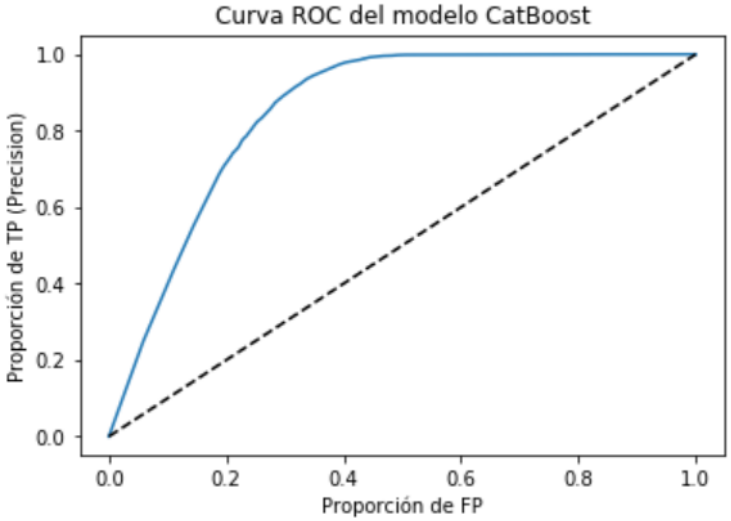


Fuente: Elaboración propia

Los resultados de este modelo también son muy buenos, a partir de la matriz de confusión se obtiene que **recall** es 0.936574 y la **exactitud balanceada** es 0.799184 (media de recall de la clase del 0 y la clase del 1).

El valor de **área bajo la curva ROC** es 0.851546, muy similar al resto de modelos. En la figura inferior se muestra el gráfico de la curva ROC.

Figura 34. *Curva ROC del modelo CatBoost*



Fuente: Elaboración propia

7.5. Red neuronal

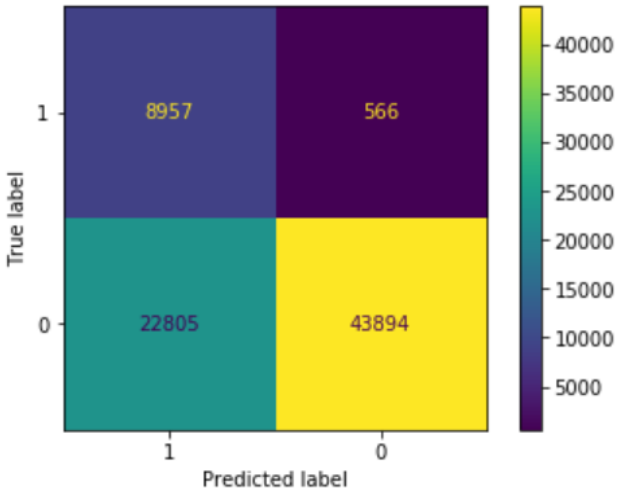
El entrenamiento de la red neuronal multicapa para clasificación binaria se ha realizado con el modelo MLPClassifier (Hinton et al., 2022) junto con una optimización de los hiper parámetros mediante el método *GridSearchCV*[11], ambos pertenecientes a la librería scikit-learn.

La combinación parámetrica con la que mejores resultados se han obtenido es la que se explica a continuación:

- 5 **capas ocultas** de 11, 40, 11, 40 y 11 **nodos** respectivamente en cada capa oculta.
- La **función de activación** empleada en cada nodo es **relu**, que se recuerda que es $f(x) = máx.(0, x)$.
- El algoritmo para la optimización estocástica de los pesos basado en gradientes es ***adam***. Este algoritmo funciona muy bien en bases de datos grandes por los resultados que ofrece con un tiempo de entrenamiento óptimo.
- La fuerza del término de parametrización se ha especificado con un parámetro llamado ***alpha***, que toma el valor 10^{-7}.
- La **tasa de aprendizaje** para las actualizaciones de los pesos se ha especificado como *constante*.
- El número de ***epochs*** indicado es 200, por lo que se han ejecutado 200 veces el proceso de *backpropagation* en todos los datos de entrenamiento.

Los resultados de las predicciones de la red neuronal se muestran en la siguiente **matriz de confusión:**

Figura 35. *Matriz de confusión de la red neuronal*

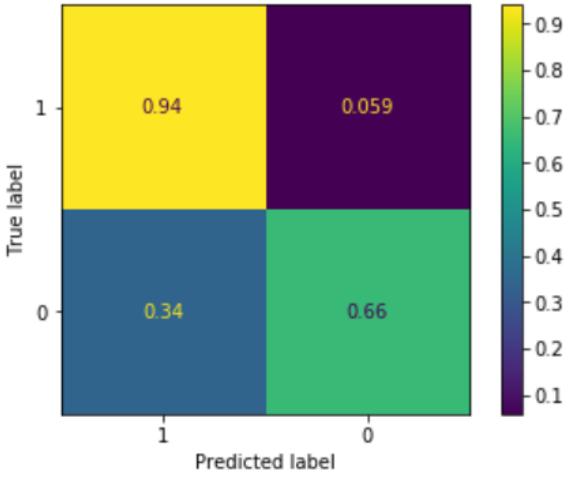


Fuente: Elaboración propia

[11] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Para evaluar mejor la matriz de confusión anterior se muestran los resultados normalizados:

Figura 36. *Matriz de confusión normalizada de la red neuronal*



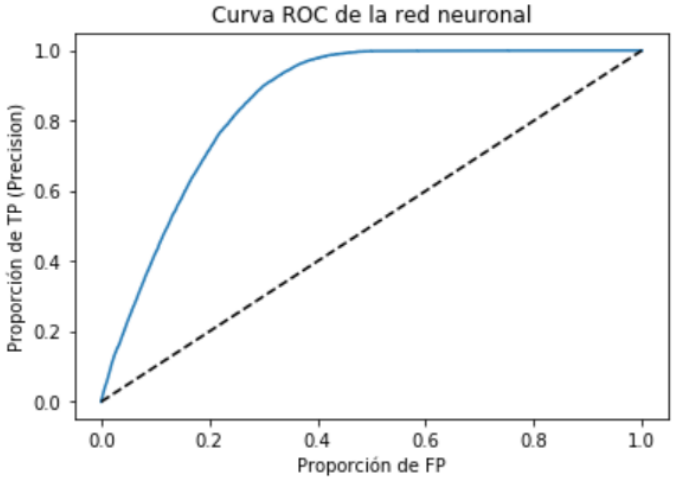
Fuente: Elaboración propia

Los resultados obtenidos son, junto al modelo XGBoost, los mejores. No obstante, debido a que en este capítulo se están mostrando los modelos que mejor carácter predictivo han presentado y, además, cada uno de estos modelos se ha optimizado con la mejor combinación paramétrica posible no se aprecia gran diferencia entre estos modelos.

A partir de los resultados de la matriz de confusión, el valor de la **exactitud balanceada** obtenido es de 0.799327, y el de **recall** es 0.940565. Estos resultados son ligeramente superiores a partir de los terceros decimales en comparación a los demás modelos.

El valor del **área bajo la curva** es el máximo que se ha obtenido, siendo de 0.855619. En la figura inferior se muestra la curva ROC, cuyo comportamiento es similar al de los modelos anteriores.

Figura 37. *Curva ROC de la red neuronal*



Fuente: Elaboración propia

7.6. Comparación final entre los modelos

Al entrenar numerosos modelos de Inteligencia Artificial los que mejores resultados han presentado son los que se muestran en la tabla inferior. Debido a la evaluación en una simulación de **escenario real** en el que la clase formada por las personas que no han convertido es desproporcionada en comparación a la clase minoritaria formada por las personas que sí contratarían el seguro adicional, las medidas de evaluación escogidas son recall, el valor del área bajo la curva ROC y la exactitud balanceada. La explicación de por qué estas medidas son las más adecuadas se puede hallar en el apartado 5.8. en el que se razona cada una de ellas, y al comienzo de este capítulo.

Tabla XIV. *Tabla comparativa final entre los mejores modelos*

	Regresión logística	Random Forest	XGBoost	CatBoost	Red neuronal
Recall	0.939410	0.935945	0.941300	0.936575	0.940565
AUC	0.845484	0.852695	0.853917	0.851546	0.855616
Exactitud balanceada	0.788675	0.798997	0.798953	0.799184	0.799328

Fuente: Elaboración propia

A partir de la tabla anterior se observa que XGBoost y la red neuronal son los modelos que se elegirían finalmente por sus mejores resultados, aunque no destaquen notablemente frente a los demás ya que las diferencias se encuentran sobre todo a partir de los segundos decimales. Aunque parezca sorprendente a primera vista que la regresión logística obtenga resultados similares a la red neuronal y al resto de modelos de Machine Learning, se recuerda que no ha sido la regresión logística clásica en el que los coeficientes se estiman por máxima verosimilitud.

Como conclusión, los modelos que se muestran son los que han destacado por sus buenos resultados frente a otros realizados como árboles de decisión, KNN, etc. Sacándole el máximo provecho a estos 5 modelos mediante la combinación paramétrica y los métodos que potencian su rendimiento los resultados obtenidos no difieren tanto entre sí. Por tanto, la elección del mejor modelo (siempre que sea uno de estos) no resulta tan sustancial, pero sí que es necesario que los datos posean la máxima calidad, habiéndose tratado previamente de la mejor manera para que cumplan los requerimientos del modelo que se aplica y aumenten su rendimiento, así como la especificación de los mejores parámetros y métodos para el modelo seleccionado.

8. CONCLUSIONES

Debido a la enorme cantidad de datos disponible hoy en día, la **Inteligencia Artificial** constituye un gran desarrollo tecnológico y científico con el que automatizar múltiples tareas y extraer información muy valiosa que se puede aplicar a una amplia variedad de campos, no sólo a las compañías de seguros. Este trabajo constituye un ejemplo de su aplicación a una de las estrategias más significativas actualmente como es el *cross-selling*.

En el **capítulo 3** se ha introducido el ***cross-selling***, que constituye una de las estrategias de aumento de la rentabilidad más relevantes hoy en día. Para alcanzar el éxito con esta táctica es indispensable realizar un estudio previo para identificar las características de los clientes más propensos a la contratación de productos adicionales adecuados a sus necesidades. Se debe llevar a cabo una oferta adecuada de *cross-selling* para los clientes en el momento idóneo. En el sector asegurador, si esta oferta consiste en seguros adicionales, se deberá prestar especial atención a las coberturas incorporadas para que el producto resulte más atractivo de cara al cliente.

En este trabajo, la tasa de conversión que se produce en el *cross-selling* al nuevo contrato de seguros se ha predicho mediante un análisis de nueva cartera. Se ha llevado a cabo un estudio de nuevo negocio sobre clientes que ya poseen un seguro de salud con una compañía de seguros, y esa misma compañía de seguros les ofrece seguros de automóvil. Este estudio es de nuevo negocio porque no se dispone de información de una cartera de vehículos existente en esa compañía. Por ello, se deberían de tener en cuenta las primas del seguro de automóviles ofrecidas por otras compañías aseguradoras que forman parte de la competencia para ofrecer un precio más atractivo.

Para aumentar notablemente las probabilidades de éxito del *cross-selling*, se ha propuesto una metodología para predecir la tasa de conversión en función de las características de los clientes y de sus vehículos que ya tienen el seguro de salud, en la que se han empleado los modelos predictivos de Inteligencia Artificial con gran eficacia. Así, no sólo se pueden aumentar los ingresos de la compañía, sino que también aumentará la relación con el cliente y sus probabilidades de fidelización.

El uso de la Inteligencia Artificial constituye uno de los grandes avances de este siglo, en el que gracias a la gran cantidad de datos disponible se puede extraer información muy valiosa con la que se generará gran riqueza y constituirá una revolución en varios ámbitos. Con el avance de la Inteligencia Artificial, sigue aumentando su impacto en una sociedad cada vez más digital. Su aplicación en el *cross-selling* tampoco pasa desapercibida suponiendo un gran progreso para las compañías de seguros.

Por tanto, para determinar los factores de tarificación más significativos en la conversión al nuevo seguro de automóviles, varios modelos de Inteligencia Artificial han demostrado ser técnicas muy potentes como las **redes neuronales**, **XGBoost**, **Random**

Forest, **CatBoost**, o incluso la **regresión logística** con mejoras en su implementación como incorporación de términos de regularización o el empleo de un algoritmo de descenso de coordenadas.

En el **capítulo 5** se ha explicado de manera sencilla pero rigurosa los modelos de Inteligencia Artificial más relevantes en la industria: los modelos **XGBoost** y **CatBoost** utilizados consisten en un sistema de árboles de decisión concatenados secuencialmente para mejorar los resultados de los árboles previos; el modelo **Random Forest** representa una combinación de varios árboles de decisión independientes; y respecto a la **red neuronal**, constituye un modelo novedoso en el que se intenta simular el funcionamiento de las neuronas en el cerebro humano para aprender de los datos y realizar predicciones eficientes y razonables.

Antes incluso de pensar en los posibles modelos que se han aplicado ha sido necesario realizar una evaluación previa de los datos para decidir cuáles son los modelos más adecuados. No es lo mismo utilizar un modelo para clasificación que para regresión. En este trabajo como la variable que se ha predicho es dicotómica, se han utilizado modelos para clasificación. Además, dependiendo de la naturaleza categórica o numérica de los datos el modelo que se aplica puede ser diferente. Por ejemplo, el modelo CatBoost ya tiene implementado un método de codificación eficiente para las variables categóricas. Por todo ello, en el **capítulo 4**, el anterior al de la exposición de los modelos de Inteligencia Artificial, se ha realizado una evaluación previa de la calidad de los datos.

Para obtener resultados consistentes en los modelos, en el **capítulo 6** se ha presentado una propuesta metodológica que se suele seguir en Machine Learning para que los datos tengan alta calidad y cumplan con los requerimientos de la mayoría de modelos. Se ha razonado adecuadamente el uso de cada método y elegido para su aplicación los que verifican las hipótesis necesarias. Además del preprocesamiento, se debe prestar especial atención al análisis de correlaciones, en concreto a la correlación de las variables con la variable objetivo, ya que resulta realmente útil para identificar las variables del cliente, de su vehículo o de la propia póliza que más influyen en la contratación final del producto. Para ello se han visualizado gráficamente cada una de las relaciones entre las distintas variables y la conversión y mediante el coeficiente de correlación de Cramer se han cuantificado dichas relaciones entre las variables que cumplen las hipótesis.

Como las variables deben presentar un formato adecuado para poder aplicar los modelos de Inteligencia Artificial y también aumentar su eficiencia, es bastante recomendable tratar adecuadamente los datos faltantes, los valores atípicos, eliminar las variables redundantes, etc. Asimismo, muchos modelos no suelen admitir variables cuyos valores sean caracteres en lugar de números, por lo que para las diferentes variables se ha buscado el método de codificación más apropiado y se ha aplicado. También hay que tener en cuenta que la variable objetivo presenta una gran cantidad de valores de la clase 0 en comparación a la clase 1. Esto es lógico, ya que en el mundo real la mayoría de

personas potenciales no suelen aceptar la oferta que se les ofrece. Sin embargo, es necesario entrenar los modelos en ambas clases con una muestra suficiente para que posteriormente sean capaces de identificar a las personas que contratarían el nuevo seguro. Por ello, a partir de la división de la base de datos en datos de entrenamiento y datos de prueba, se han balanceado los datos de entrenamiento para posteriormente evaluar mediante los datos de prueba el funcionamiento de los modelos en casos reales.

Para elegir las variables explicativas de los modelos se ha usado uno de los mejores algoritmos de selección de variables predictoras, conocido como *BorutaShap*. Este método se diferencia de los demás en que, en lugar de comparar a las variables entre sí para evaluar su capacidad predictiva, las compara con versiones aleatorizadas de ellas mismas en las que no se tiene en cuenta la relación con la conversión. Los resultados de este algoritmo han corroborado el análisis de correlaciones descartando las variables indicadoras del género y de la posesión del carnet de conducir por su bajo carácter predictivo de la conversión.

En el **capítulo 7** en el que se exponen los resultados predictivos de los mejores modelos, se aprecia como exprimiendo al máximo cada uno de ellos escogiendo los métodos y las combinaciones paramétricas que potencian su rendimiento se obtienen puntuaciones similares en las principales métricas de evaluación. Por tanto, al descartar previamente los modelos que peores resultados habían presentado, entre los modelos XGBoost, CatBoost, Random Forest, la red neuronal y la regresión logística, no hay ningún modelo que destaque notablemente frente a los demás. Es necesario recalcar que la regresión logística aplicada no ha sido el modelo clásico en el que los coeficientes se estiman por máxima verosimilitud, sino una versión de Machine Learning que aplica principalmente un algoritmo de descenso de coordenadas, términos de regularización y parámetros adicionales para mejorar más aún la capacidad predictiva y la eficacia del modelo.

Finalmente, **futuras líneas de investigación** en este campo podrían incluir una evolución hacia los modelos preconcedidos. Por ejemplo, con los datos del seguro de salud, sin conocer los datos del seguro de automóvil, se podría proponer al cliente una prima atractiva buscando asegurar su vehículo para intentar conseguir esa venta adicional con menos esfuerzo. Esta oferta sería más provechosa que se realizara a clientes que tienen un buen historial de salud con baja siniestralidad porque es probable que ese mismo comportamiento se traslade al seguro de automóviles.

Otro posible estudio sería en vez de aplicar una clasificación binaria en la que se divide a los clientes en función de si contratan el seguro adicional o no, se podrían aplicar los modelos de Machine Learning y la red neuronal con una clasificación multiclase, en la que se podría incluir una tercera clase de clientes dudosos, a los que se les puede mejorar la oferta comercial mediante modificaciones en la prima, coberturas, etc. A estos clientes

que previamente se había predicho que no contratarían el seguro, sería a los que resultaría muy oportuno que se les realizase la oferta comercial.

La generación de la matriz de valor con el eje vertical seguro de origen (salud) y con el eje horizontal seguro de destino (automóviles) podría constituir otra futura línea de investigación. Con ella, se podría determinar el valor que aportan los clientes a la compañía en función del seguro, o los seguros, que tengan contratado y aprovechar esta información. Por ejemplo, los que generen más valor financiero-actuarial para los 2 seguros sería muy óptimo aplicarles campañas comerciales con el objetivo de que tengan contratados más seguros con la compañía ya que ello conllevaría numerosos beneficios para esta.

Para concluir, también se podría optimizar la cartera de manera que se identifique el *cross-selling* más exitoso que minimizaría el SCR. Esta decisión formaría parte de la gerencia de la compañía ya que se podrían diversificar los riesgos y con ello, obtener un SCR más óptimo que daría lugar a mayores beneficios para la empresa.

9. BIBLIOGRAFÍA

Aishwarya Singh. (2020). *4 Boosting Algorithms You Should Know – GBM, XGBoost, LightGBM & CatBoost.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/02/4-boosting-algorithms-machine-learning/

Ali, J., Khan, R., Ahmad, N., & Maqsood, I. (2012). Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI), 9*(5), 272.

amananandrai. (2021). *Performance measures for Imbalanced Classes.* DEV. https://dev.to/amananandrai/performance-measures-for-imbalanced-classes-2ojj

Amat Rodrigo, J. (2020). *Árboles de decisión, random forest, gradient boosting y C5.0.* https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50#Boosting

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research,* https://scikit-learn.org/stable/modules/grid_search.html#randomized-parameter-search

Biarnes, A. (2021). *How CatBoost encodes categorical variables?* Towards Data Science. https://towardsdatascience.com/how-catboost-encodes-categorical-variables-3866fb2ae640

Biau, G. (2012). Analysis of a random forests model. *The Journal of Machine Learning Research, 13*(1), 1063-1095.

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

Brownlee, J. (2018). *How to Configure the Number of Layers and Nodes in a Neural Network.* Machine Learning Mastery. https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/

Brownlee, J. (2020a). *Extreme Gradient Boosting (XGBoost) Ensemble in Python.* Machine Learning Mastery. https://machinelearningmastery.com/extreme-gradient-boosting-ensemble-in-python/

Brownlee, J. (2020b). *Tour of Evaluation Metrics for Imbalanced Classification.* Machine Learning Mastery. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

Charbuty, B., & Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends, 2*(01), 20-28.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Paper presented at the *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785-794.

Chen, T., He, T., Benesty, M., & Khotilovich, V. (2019). Package 'xgboost'. *R Version, 90*

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., & Chen, K. (2015). Xgboost: extreme gradient boosting. *R Package Version 0.4-2, 1*(4), 1-4.

Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. *Ensemble machine learning* (pp. 157-175). Springer.

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *arXiv Preprint arXiv:1810.11363,*

Espinosa-Zúñiga, J. J. (2020). Aplicación de algoritmos Random Forest y XGBoost en una base de solicitudes de tarjetas de crédito. *Ingeniería, Investigación Y Tecnología, 21*(3)

Fauzan, M. A., & Murfi, H. (2018). The accuracy of XGBoost for insurance claim prediction. *Int.J.Adv.Soft Comput.Appl, 10*(2)

Ghoshal, A., Mookerjee, V. S., & Sarkar, S. (2021). Recommendations and Cross-selling: Pricing Strategies when Personalizing Firms Cross-sell. *Journal of Management Information Systems, 38*(2), 430-456. https://doi.org/10.1080/07421222.2021.1912930

Gopagoni, D. R., Lakshmi, P. V., & Siripurapu, P. (2020). Predicting the Sales Conversion Rate of Car Insurance Promotional Calls. (pp. 321-329). Singapore: Springer Singapore. https://doi.org/10.1007/978-981-15-6014-9_37

Hancock, J. T., & Khoshgoftaar, T. M. (2020). CatBoost for big data: an interdisciplinary review. *Journal of Big Data, 7*(1), 1-45.

Hashmi, F.*How to use Artificial Neural Networks for classification in python?* Thinking Neuron. https://thinkingneuron.com/how-to-use-artificial-neural-networks-for-classification-in-python/

Hastie, T., Tibshirani, R., & Friedman, J. (2009). Random forests. *The elements of statistical learning* (pp. 587-604). Springer.

Hemavati. (2021). *Categorical Encoding with CatBoost Encoder.* GeeksforGeeks. https://www.geeksforgeeks.org/categorical-encoding-with-catboost-encoder/

Hinton, Geoffrey, E., Glorot, Xavier, Yoshua Bengio, He, Kaiming, Kingma, Diederik & Jimmy Ba. (2022). *Multi-layer Perceptron classifier.* https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Ho, T. K. (1995). Random decision forests. Paper presented at the *Proceedings of 3rd International Conference on Document Analysis and Recognition, , 1* 278-282.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(8), 832-844.

Kaishev, V. K., Nielsen, J. P., & Thuring, F. (2013). Optimal customer selection for cross-selling of financial services products. *Expert Systems with Applications, 40*(5), 1748-1757.

Kukreja, H., Bharath, N., Siddesh, C. S., & Kuldeep, S. (2016). An introduction to artificial neural network. *Int J Adv Res Innov Ideas Educ, 1*, 27-30.

Mazzanti, S. (2020). *Boruta Explained Exactly How You Wished Someone Explained to You.* Towards Data Science. https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a

Mingers, J. (1989). An empirical comparison of pruning methods for decision tree induction. *Machine Learning, 4*(2), 227-243.

Nielsen, D. (2016). Why does xgboost win" every" machine learning competition? *Tree Boosting with Xgboost,*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., & Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research, 12*, 2825-2830.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems, 31*

Quinlan, J. R. (1996). Learning decision tree classifiers. *ACM Computing Surveys (CSUR), 28*(1), 71-72.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin. (2008). LIBLINEAR: A library for large linear classification Journal of Machine Learning Research [computer software]

Ramya Bhaskar Sundaram. (2018). *An End-to-End Guide to Understand the Math behind XGBoost.* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/?utm_source=blog&utm_medium=4-boosting-algorithms-machine-learning#h2_9

Rohith Gandhi. (2018). *Gradient Boosting and XGBoost.* HackerNoon. https://medium.com/hackernoon/gradient-boosting-and-xgboost-90862daa6c77

Sahil Kamboj. (2020). *Performance metrics for evaluating a model on an imbalanced data set?* Data Science Story. https://medium.com/datasciencestory/performance-metrics-for-evaluating-a-model-on-an-imbalanced-data-set-1feeab6c36fe

Salazar, M. T., Harrison, T., & Ansell, J. (2004). CRM in the Insurance Industry: An Attempt to Use Survival Analysis in Retention and Cross Selling.

Song, Y., & Ying, L. U. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*(2), 130.

Trevisan, V. (2022). *Boruta SHAP: A Tool for Feature Selection Every Data Scientist Should Know.* Towards Data Science. https://towardsdatascience.com/boruta-shap-an-amazing-tool-for-feature-selection-every-data-scientist-should-know-33a5f01285c0

Vicens-Salort, E., Cruz, F., Esteban, L., Cruz, F., & Esteban, L. (200916-04). *Colaboration with Brazil View project Planificación de Operaciones para Cadenas de Suministro de productos innovadores View project*

Wang, C., Deng, C., & Wang, S. (2020). Imbalance-XGBoost: Leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. *Pattern Recognition Letters*, *136*, 190-197.

10. ANEXO

10. ANEXO

INTELIGENCIA ARTIFICIAL APLICADA AL CROSS-SELLING

1. Observación inicial de los datos

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [ ]:

```
tfm = pd.read_csv('train.csv')
tfm.shape
```

In [ ]:

```
tfm.head()
```

In [ ]:

```
tfm = tfm.rename(columns={"Gender": "sexo", "Age": "edad", "Driving_License": "carnet_conducir",
                          "Region_Code": "codigo_postal", "Previously_Insured": "asegurado_antes",
                          "Vehicle_Age": "edad_vehiculo", "Vehicle_Damage": "daños_vehiculo",
                          "Annual_Premium": "prima", "Policy_Sales_Channel":"canal_contacto",
                          "Vintage": "dia", "Response": "conversion"})
tfm.head()
```

In [ ]:

```
tfm['prima'] = tfm['prima']/100
tfm.describe()
```

In [ ]:

```
tfm.describe(include = 'O')
```

In [ ]:

```
tfm.isnull().sum() #No hay datos missings
```

2. EDA: Boxplots, histogramas

In [ ]:

```
continuas = tfm.describe().columns
print(continuas)
discretas = tfm.describe(include = 'O').columns
discretas
```

In [ ]:

```
tfm.info()
```

- Distribución de variables

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['edad'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.distplot(x=tfm['edad'])

fig.suptitle('edad', fontsize=15)

plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['carnet_conducir'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.histplot(x=tfm['carnet_conducir'], data=tfm, bins= 20, color= '#78E1E1', ax=ax2)

fig.suptitle('carnet_conducir', fontsize=15)

plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['codigo_postal'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.histplot(x=tfm['codigo_postal'], data=tfm, bins= 20, color= '#78E1E1', ax=ax2)

fig.suptitle('codigo_postal', fontsize=15)

plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['asegurado_antes'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.histplot(x=tfm['asegurado_antes'], data=tfm, bins= 20, color= '#78E1E1', ax=ax2)

fig.suptitle('asegurado_antes', fontsize=15)

plt.show()
```

In [ ]:

```
plt.figure(figsize = (13,7))
plt.subplot(2,1,2)
sns.boxplot(x=tfm['prima'], data=tfm, orient='h', color= '#78E1E1')

plt.subplot(2,1,1)
sns.distplot(x=tfm['prima']) #, bins= 20, color= '#06748C')
plt.title('prima', fontsize=15)

plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['canal_contacto'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.histplot(x=tfm['canal_contacto'], data=tfm, bins= 20, color= '#78E1E1', ax=ax2)

fig.suptitle('canal_contacto', fontsize=15)

plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['dia'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.histplot(x=tfm['dia'], data=tfm, bins= 20, color= '#78E1E1', ax=ax2)

fig.suptitle('dia', fontsize=15)

plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (20,4))
ax1=fig.add_subplot(1,2,1)
sns.boxplot(x=tfm['conversion'], data=tfm, orient='h', color= '#78E1E1', ax=ax1)

ax2=fig.add_subplot(1,2,2)
sns.histplot(x=tfm['conversion'], data=tfm, bins= 20, color= '#78E1E1', ax=ax2)

fig.suptitle('conversion', fontsize=15)

plt.show()
```

In [ ]:

```
#plt.figure(figsize = (11, 3))
sns.countplot(tfm['asegurado_antes'], palette = 'husl')
plt.show()
```

In [ ]:

```
#plt.figure(figsize = (11, 3))
sns.countplot(tfm['carnet_conducir'], palette = 'husl')
plt.show()
```

In [ ]:

```
(tfm[tfm['carnet_conducir']==1].shape[0])/(tfm.shape[0])
```

In [ ]:

```
plt.figure(figsize = (21, 5))
sns.countplot(tfm['codigo_postal'], palette = 'husl')
plt.show()
```

In [ ]:

```
plt.figure(figsize = (11, 3))
sns.countplot(tfm['canal_contacto'], palette = 'husl')
plt.show()
```

In [ ]:

```
plt.figure(figsize = (70, 15))
sns.countplot(tfm['canal_contacto'], palette = 'husl')
plt.show()
```

In [ ]:

```
#plt.figure(figsize = (11, 3))
sns.countplot(tfm['sexo'], palette = 'husl')
plt.show()
```

In [ ]:

```
# Datos a graficar
etiquetas = ['Femenino','Masculino']
valores = [175020,206089]
colores = ['green','blue']

plt.pie(x=valores, labels=etiquetas, colors = colores)
plt.title('sexo')
plt.show()
```

In [ ]:

```
#plt.figure(figsize = (11, 3))
sns.countplot(tfm['edad_vehiculo'], palette = 'husl')
plt.show()
```

In [ ]:

```
#plt.figure(figsize = (11, 3))
sns.countplot(tfm['daños_vehiculo'], palette = 'husl')
plt.show()
```

- Relación de variables categóricas con la conversión

In [ ]:

```
sns.barplot(x = tfm['sexo'], y = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['conversion'], columns = tfm['sexo'], normalize = 'index')
```

In [ ]:

```
sns.barplot(x = tfm['edad_vehiculo'], y = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['conversion'], columns = tfm['edad_vehiculo'], normalize = 'ind
ex')
```

In [ ]:

```
sns.barplot(x = tfm['edad_vehiculo'], y = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['conversion'], columns = tfm['edad_vehiculo'], normalize = 'ind
ex')
```

- Relación entre variables

In [ ]:

```
plt.figure(figsize = (6, 3))
plt.scatter(x = tfm['prima'], y = tfm['edad'])
plt.xlabel("prima")
plt.ylabel("edad")
plt.show()
```

In [ ]:

```
sns.countplot(tfm['edad_vehiculo'], hue=tfm['daños_vehiculo'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index= tfm['edad_vehiculo'], columns = tfm['daños_vehiculo'], normalize =
'index')
```

In [ ]:

```
sns.barplot(x = tfm['edad_vehiculo'], y= tfm['conversion'], hue = tfm['edad_vehiculo'],
palette = 'husl')
plt.show()
```

In [ ]:

```
plt.figure(figsize = (18, 5))
sns.countplot(tfm['edad_vehiculo'], hue = tfm['asegurado_antes'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['asegurado_antes'], columns = tfm['edad_vehiculo'], normalize =
'columns')
```

- Relación de variables numéricas con la conversión

In [ ]:

```
plt.figure(figsize = (17, 7))
sns.countplot(tfm['edad'], hue = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['conversion'], columns = 'edad', values = tfm['edad'], aggfunc=
'mean')
```

In [ ]:

```
plt.figure(figsize = (17, 7))
sns.countplot(tfm['dia'], hue = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
fig = plt.figure(figsize = (10,5))
sns.kdeplot(data=tfm, x= 'prima', hue = 'conversion', fill=True, common_norm=False, pal
ette = 'husl')

plt.xlabel('prima')
fig
plt.show()
```

In [ ]:

```
tfm.describe()
```

In [ ]:

```
sns.barplot(x = tfm['asegurado_antes'], y = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['asegurado_antes'], columns = 'Average edad', #values = tfm['ed
ad'], aggfunc='mean'
            normalize = 'columns')
```

In [ ]:

```
sns.barplot(x = tfm['carnet_conducir'], y = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['conversion'], columns = tfm['carnet_conducir'], normalize = 'columns')
```

In [ ]:

```
sns.barplot(x = tfm['daños_vehiculo'], y = tfm['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
pd.crosstab(index = tfm['conversion'], columns = tfm['daños_vehiculo'], normalize = 'columns')
```

3. Feature engineering

In [ ]:

```
tfm = tfm.drop(['id', 'dia'], axis=1)
```

Encoding sexo y daños_vehiculo

In [ ]:

```
tfm['sexo'] = tfm['sexo'].map( {'Female': 0, 'Male': 1} ).astype(int)
tfm['daños_vehiculo'] = tfm['daños_vehiculo'].map( {'No': 0, 'Yes': 1} ).astype(int)
tfm['edad_vehiculo'] = tfm['edad_vehiculo'].map( {'< 1 Year': 0, '1-2 Year':1, '> 2 Years':2} ).astype(int)
```

In [ ]:

```
tfm2 = tfm.copy()
tfm2.columns
```

4. Train, validation, test split

In [ ]:

```
tfm2.columns
```

In [ ]:

```
X = tfm2.drop(columns = 'conversion')
```

In [ ]:

```
y = tfm2['conversion']
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 42, test_size = 0.2)
```

In [ ]:

```
X.head()
```

5. CatBoost encoding

In [ ]:

```
import category_encoders as ce
```

In [ ]:

```
# Para el CatBoost
X_train_cb = X_train.copy()
X_test_cb = X_test.copy()
y_train_cb = y_train.copy()
y_test_cb = y_test.copy()
```

In [ ]:

```
columns_to = ['codigo_postal', 'canal_contacto']
count = ce.CatBoostEncoder(cols=columns_to, return_df=False)

columns_to_te = list(map(lambda i: i #+ '_codificada'
                         , columns_to))

X_train[columns_to_te] = count.fit_transform(X_train[columns_to], y_train)
X_test[columns_to_te] = count.transform(X_test[columns_to])
```

In [ ]:

```
X_train[columns_to_te]
```

6. Estandarización

In [ ]:

```
tfm2.prima.describe()
```

In [ ]:

```
tfm2.edad.describe()
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, RobustSca
ler

mms = MinMaxScaler()
X_train[['edad']] = mms.fit_transform(X_train[['edad']])
X_test[['edad']] = mms.fit_transform(X_test[['edad']])

X_train_cb[['edad']] = mms.fit_transform(X_train_cb[['edad']])
X_test_cb[['edad']] = mms.fit_transform(X_test_cb[['edad']])

rs = RobustScaler()
X_train[['prima']] = mms.fit_transform(X_train[['prima']])
X_test[['prima']] = mms.fit_transform(X_test[['prima']])

X_train_cb[['prima']] = mms.fit_transform(X_train_cb[['prima']])
X_test_cb[['prima']] = mms.fit_transform(X_test_cb[['prima']])
```

In [ ]:

```
X_train.prima.describe()
```

In [ ]:

```
X_train.edad.describe()
```

Como la variable prima no toma valores negativos, a todos los valores obtenidos en el escalamiento se le suma el valor más negativo obtenido para que todos resulten positivos:

In [ ]:

```
X_train[['prima']] = X_train[['prima']] + X_train[['prima']].min().abs()
X_test[['prima']] = X_test[['prima']] + X_test[['prima']].min().abs()

X_train_cb[['prima']] = X_train_cb[['prima']] + X_train_cb[['prima']].min().abs()
X_test_cb[['prima']] = X_test_cb[['prima']] + X_test_cb[['prima']].min().abs()
```

In [ ]:

```
X_train.prima.describe()
```

7. Análisis de correlación

In [ ]:

```
def calcCramerV(x, y):
    cm = pd.crosstab(x, y).values
    n = cm.sum()
    r, k = cm.shape

    chi2 = stats.chi2_contingency(cm)[0]
    chi2corr = max(0, chi2 - (k-1)*(r-1)/(n-1))

    kcorr = k - (k-1)**2/(n-1)
    rcorr = r - (r-1)**2/(n-1)

    return np.sqrt((chi2corr/n) / (min(kcorr-1, rcorr-1)))
```

In [ ]:

```
df_train = X_train.copy()
df_train['prima'] = round(df_train['prima'], 2)
df_train['edad'] = round(df_train['edad'], 2)
df_train['codigo_postal'] = round(df_train['codigo_postal'], 2)
df_train['canal_contacto'] = round(df_train['canal_contacto'], 2)
df_train['sexo'] = df_train['sexo'].astype(str)
df_train['edad'] = df_train['edad'].astype(str)
df_train['carnet_conducir'] = df_train['carnet_conducir'].astype(str)
df_train['codigo_postal'] = df_train['codigo_postal'].astype(str)
df_train['asegurado_antes'] = df_train['asegurado_antes'].astype(str)
df_train['edad_vehiculo'] = df_train['edad_vehiculo'].astype(str)
df_train['daños_vehiculo'] = df_train['daños_vehiculo'].astype(str)
#df_train['prima'] =df_train['prima'].astype(str)
df_train['canal_contacto']= df_train['canal_contacto'].astype(str)
y_train2 = y_train.astype(str)
```

In [ ]:

```
df_train.info()
```

In [ ]:

```
df_train = pd.concat([df_train, y_train2], axis = 1)
```

In [ ]:

```
from scipy import stats

cat_attributes = df_train.select_dtypes(include='object')
dict_corr = {}
columns = cat_attributes.columns.tolist()

for column in columns:
    dict_corr[column] = {}

    for column2 in columns:
        dict_corr[column][column2] = calcCramerV(cat_attributes[column], cat_attributes
[column2])

corr = pd.DataFrame(dict_corr)
```

In [ ]:

```
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True

fig = plt.figure(figsize = (39,6))
with sns.axes_style("white"):
    ax = sns.heatmap(corr, annot=True, mask=mask, vmin=0, vmax=1, square=True)
```

7. Balanceamiento de los datos

In [ ]:

```
sns.countplot(tfm2['conversion'], palette = 'husl')
plt.show()
```

In [ ]:

```
tfm2['conversion'].value_counts()
```

In [ ]:

```
46710 / (334399+46710)
```

In [ ]:

```
1- (46710 / (334399+46710))
```

In [ ]:

```
import imblearn
from imblearn.over_sampling import SMOTE
```

In [ ]:

```
smote = SMOTE(random_state = 30)
```

In [ ]:

```
X_train_smt, y_train_smt = smote.fit_resample(X_train, y_train)

X_train_smt_cb, y_train_smt_cb = smote.fit_resample(X_train_cb, y_train_cb)
```

In [ ]:

```
train_smt = pd.concat([X_train_smt, y_train_smt], axis = 1)
```

In [ ]:

```
train_smt['conversion'].value_counts()
```

In [ ]:

```
X_train.head()
```

8. Selección de variables predictoras

In [ ]:

```
from BorutaShap import BorutaShap
from xgboost import XGBClassifier

model = XGBClassifier(n_jobs=-1)

Feature_Selector = BorutaShap(model = model, importance_measure='shap', classification=
True)
Feature_Selector.fit(X=X_train_smt_cb,y=y_train_smt_cb,n_trials=15,sample=True,train_or
_test='test',normalize=False, verbose=True)
```

In [ ]:

```
Feature_Selector.plot(which_features='all', figsize=(12,9))
```

In [ ]:

```
featuressubset2 = Feature_Selector.Subset()
featuressubset2
```

9. Modelos

In [ ]:

```
from sklearn.metrics import recall_score,precision_score,f1_score, balanced_accuracy_sc
ore
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,roc
_auc_score,ConfusionMatrixDisplay
```

In [ ]:

```
X_train0 = X_train_smt.copy()
X_test0 = X_test.copy()
```

In [ ]:

```
X_train_smt = X_train_smt[featuressubset2.columns]
X_test = X_test[featuressubset2.columns]

X_train_smt_cb = X_train_smt_cb[featuressubset2.columns]
X_test_cb = X_test_cb[featuressubset2.columns]
```

XGBClassifier

In [ ]:

```
xgb=XGBClassifier()
XGB_fit=xgb.fit(X_train_smt, y_train_smt)
y_predict = XGB_fit.predict(X_test)
XGB_probability = XGB_fit.predict_proba(X_test)[:,1]
```

In [ ]:

```
XGB_fit.feature_importances_
```

In [ ]:

```
from xgboost import plot_importance
plot_importance(XGB_fit)
plt.show()
```

In [ ]:

```
acc_xgb=accuracy_score(y_test,y_predict)
recall_xgb=recall_score(y_test,y_predict)
precision_xgb=precision_score(y_test,y_predict)
f1score_xgb=f1_score(y_test,y_predict)

AUC_xgb=roc_auc_score(y_test,XGB_probability)

print("Accuracy : ", acc_xgb)
print("Precision:",precision_score(y_test,y_predict))
print("Recall:",recall_score(y_test,y_predict))
print("F1-Score:",f1_score(y_test,y_predict))
print("ROC_AUC Score:",AUC_xgb)
```

In [ ]:

```
acc_balanced_xgb=balanced_accuracy_score(y_test,y_predict)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,y_predict))
```

In [ ]:

```
print(classification_report(y_predict,y_test))
```

In [ ]:

```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(y_test, XGB_probability, pos_label=1)

plt.title('Curva ROC del modelo XGBoost')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
cm=confusion_matrix(y_test,y_predict, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='GnBu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels = [1,0])
cm_.plot()
plt.show()
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_predict,normalize='true', labels =
[1,0])
```

XGBoost - Hyperparameter tuning

In [ ]:

```
from xgboost import XGBClassifier
model_xgb = XGBClassifier()
model_xgb.fit(X_train_smt, y_train_smt,eval_metric='mlogloss')

pred_xgb = model_xgb.predict(X_test)
predictions_xgb = [round(value) for value in pred_xgb]
accuracy_xgb = accuracy_score(y_test, predictions_xgb)
print("Accuracy: %.2f%%" % (accuracy_xgb * 100.0))
```

In [ ]:

```
model_xgb_tuned = XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                                colsample_bynode=1, colsample_bytree=1, enable_categori
cal=False,
                                gamma=10, gpu_id=-1, importance_type=None, #gamma 0
                                interaction_constraints='', learning_rate=0.02, max_del
ta_step=10, #learning_rate=0.1 #max_delta_step=0
                                max_depth=3, min_child_weight=1,eval_metric='mlogloss',
#max_depth=3, min_child_weight=1
                                monotone_constraints='()', n_estimators=180, n_jobs=12,
#nº estimator 180
                                num_parallel_tree=1, predictor='auto', random_state=1,
#random state 1
                                reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsampl
e=1,
                                tree_method='exact', validate_parameters=1, verbosity=N
one)
model_xgb_tuned.fit(X_train_smt, y_train_smt)
pred_xgb_tuned = model_xgb_tuned.predict(X_test)
predictions_xgb_tuned = [round(value) for value in pred_xgb_tuned]
```

In [ ]:

```
from xgboost import plot_importance
plot_importance(model_xgb_tuned)
plt.show()
```

In [ ]:

```
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
space={'max_depth': hp.quniform("max_depth", 3, 18, 1),
        'gamma': hp.uniform ('gamma', 1,9),
        'reg_alpha' : hp.quniform('reg_alpha', 40,180,1),
        'reg_lambda' : hp.uniform('reg_lambda', 0,1),
        'colsample_bytree' : hp.uniform('colsample_bytree', 0.5,1),
        'min_child_weight' : hp.quniform('min_child_weight', 0, 10, 1),
        'n_estimators': 180,
        'seed': 0
    }
```

In [ ]:

```
def objective(space):
    clf=XGBClassifier(
                    n_estimators =space['n_estimators'], max_depth = int(space['max_dep
th']), gamma = space['gamma'],
                    reg_alpha = int(space['reg_alpha']),min_child_weight=int(space['min
_child_weight']),
                    colsample_bytree=int(space['colsample_bytree']))

    evaluation = [(X_train_smt, y_train_smt), (X_test, y_test)]

    clf.fit(X_train_smt, y_train_smt,
            eval_set=evaluation, eval_metric="auc",
            early_stopping_rounds=10,verbose=False)

    pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, pred>0.5)
    print ("SCORE:", accuracy)
    return {'loss': -accuracy, 'status': STATUS_OK }
```

In [ ]:

```
trials = Trials()

best_hyperparams = fmin(fn = objective, space = space, algo = tpe.suggest, max_evals =
100, trials = trials)
```

In [ ]:

```
best_hyperparams
```

In [ ]:

```
xgb_tuned_probability = model_xgb_tuned.predict_proba(X_test)[:,1]


acc_xgb_tuned=accuracy_score(y_test,predictions_xgb_tuned)
recall_xgb_tuned=recall_score(y_test,predictions_xgb_tuned)
precision_xgb_tuned=precision_score(y_test,predictions_xgb_tuned)
f1score_xgb_tuned=f1_score(y_test,predictions_xgb_tuned)
AUC_xgb_tuned=roc_auc_score(y_test,xgb_tuned_probability)

print("Accuracy : ", accuracy_score(y_test,predictions_xgb_tuned))
print("Precision:",precision_score(y_test,predictions_xgb_tuned))
print("Recall:",recall_score(y_test,predictions_xgb_tuned))
print("F1-Score:",f1_score(y_test,predictions_xgb_tuned))
print("ROC_AUC Score:",AUC_xgb_tuned)
```

In [ ]:

```
fpr, tpr, _ = roc_curve(y_test, predictions_xgb_tuned, pos_label=1)

plt.title('Curva ROC del modelo XGBoost')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
acc_balanced_xgb_tuned=balanced_accuracy_score(y_test,predictions_xgb_tuned)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,predictions_xgb_tuned))
```

In [ ]:

```
cm=confusion_matrix(y_test,predictions_xgb_tuned, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='GnBu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,predictions_xgb_tuned, labels = [1,0
])
#cm_.plot()
#plt.show()
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,predictions_xgb_tuned,normalize='tru
e', labels = [1,0])
#cm_.plot()
#plt.show()
```

Regresión logística

In [ ]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LogisticRegressionCV
modelSMOTE = LogisticRegressionCV(solver='liblinear', class_weight='balanced', intercep
t_scaling=0.9, Cs=15)

modelSMOTE.fit(X_train_smt, y_train_smt)
y_pred_SMOTE_logreg = modelSMOTE.predict(X_test)
lr_probability =modelSMOTE.predict_proba(X_test)[:,1]

acc_logreg = accuracy_score(y_test, y_pred_SMOTE_logreg)
recall_logreg = recall_score(y_test, y_pred_SMOTE_logreg)
prec_logreg = precision_score(y_test, y_pred_SMOTE_logreg)
f1_logreg = f1_score(y_test, y_pred_SMOTE_logreg)

AUC_LR=roc_auc_score(y_test,lr_probability)

print("Accuracy : ", acc_logreg)
```

In [ ]:

```
X_test.describe()
```

In [ ]:

```
modelSMOTE.feature_names_in_
```

In [ ]:

```
modelSMOTE.coef_
```

In [ ]:

```
modelSMOTE.intercept_
```

In [ ]:

```
AUC_LR=roc_auc_score(y_test,lr_probability)

print("ROC_AUC Score:",AUC_LR)
```

In [ ]:

```
acc_balanced_logreg=balanced_accuracy_score(y_test,y_pred_SMOTE_logreg)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,y_pred_SMOTE_logreg))
```

In [ ]:

```
print(classification_report(y_test, y_pred_SMOTE_logreg))
```

In [ ]:

```
fpr, tpr, _ = roc_curve(y_test, lr_probability)

plt.title('Curva ROC del modelo de regresión logística')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
#Matriz de confusión
cm=confusion_matrix(y_test,y_pred_SMOTE_logreg, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='BuPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay(confusion_matrix=cm, display_labels = [1,0])
cm_.plot()
plt.show()
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_SMOTE_logreg,normalize='true'
, labels = [1,0])
```

In [ ]:

```
#exactitud balanceada
(0.9394098498372362+0.6379405988095773)/2
```

In [ ]:

```
8946/(8946+577)
```

In [ ]:

```
577/(8946+577)
```

In [ ]:

```
42550/(42550+24149)
```

In [ ]:

```
24149/(42550+24149)
```

In [ ]:

```
acc_logreg,recall_logreg,prec_logreg,f1_logreg
```

Árbol de decisión (Decision tree)

In [ ]:

```
modelSMOTEDT = DecisionTreeClassifier()
modelSMOTEDT.fit(X_train_smt, y_train_smt)
y_pred_SMOTE_DT = modelSMOTEDT.predict(X_test)
DT_probability =modelSMOTEDT.predict_proba(X_test)[:,1]

acc_DT = accuracy_score(y_test,y_pred_SMOTE_DT)
prec_DT = precision_score(y_test, y_pred_SMOTE_DT)
rec_DT = recall_score(y_test, y_pred_SMOTE_DT)
f1_DT = f1_score(y_test, y_pred_SMOTE_DT)

AUC_DT=roc_auc_score(y_test,DT_probability)

print("Accuracy : ", acc_DT)
```

In [ ]:

```
AUC_DT=roc_auc_score(y_test,DT_probability)

print("ROC_AUC Score:",AUC_DT)
```

In [ ]:

```
print(classification_report(y_test, y_pred_SMOTE_DT))
```

In [ ]:

```
acc_balanced_DT=balanced_accuracy_score(y_test,y_pred_SMOTE_DT)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,y_pred_SMOTE_DT))
```

In [ ]:

```
fpr, tpr, _ = roc_curve(y_test, DT_probability)

plt.title('Curva ROC del árbol de decisión')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
#Matriz de confusión
cm=confusion_matrix(y_test,y_pred_SMOTE_DT, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='BuPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_SMOTE_DT,normalize='true', la
bels = [1,0])
```

Random Forest

In [ ]:

```
modelSMOTERF = RandomForestClassifier()
modelSMOTERF.fit(X_train_smt, y_train_smt)
y_pred_SMOTE_RF = modelSMOTERF.predict(X_test)
RF_probability = modelSMOTERF.predict_proba(X_test)[:,1]

acc_RF = accuracy_score(y_test,y_pred_SMOTE_RF)
prec_RF = precision_score(y_test, y_pred_SMOTE_RF)
rec_RF = recall_score(y_test, y_pred_SMOTE_RF)
f1_RF = f1_score(y_test, y_pred_SMOTE_RF)

AUC_RF=roc_auc_score(y_pred_SMOTE_RF,y_test)

print("Accuracy : ", acc_RF)
print("ROC_AUC Score:",AUC_RF)
```

In [ ]:

```
print(classification_report(y_test, y_pred_SMOTE_RF))
```

In [ ]:

```
acc_balanced_RF=balanced_accuracy_score(y_test,y_pred_SMOTE_RF)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,y_pred_SMOTE_RF))
```

In [ ]:

```
fpr, tpr, _ = roc_curve(y_test, RF_probability)

plt.title('Curva ROC del modelo Random Forest')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
cm=confusion_matrix(y_pred_SMOTE_RF,y_test, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='RdPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_SMOTE_RF,normalize='true', la
bels = [1,0])
```

Random Forest - hyperparameter tuning

In [ ]:

```
X_train_smt.dtypes
```

In [ ]:

```
from sklearn.model_selection import RandomizedSearchCV
random_search = {'criterion': ['entropy', 'gini'],
                 'max_depth': [2,3,4,5,6,7,10],
                 'min_samples_leaf': [4, 6, 8],
                 'min_samples_split': [5, 7,10],
                 'n_estimators': [300]}

clf = RandomForestClassifier()
model = RandomizedSearchCV(estimator = clf, param_distributions = random_search, n_iter
= 10,
                                cv = 4, verbose= 1, random_state= 2, n_jobs = -1)

model.fit(X_train_smt,y_train_smt)
```

Guardamos el modelo

In [ ]:

```
import pickle
filename = 'random_forest.sav'
pickle.dump(model, open(filename, 'wb'))
```

In [ ]:

```
import pickle
filename = 'random_forest.sav'
rf_load = pickle.load(open(filename, 'rb'))
```

Evaluación del modelo

In [ ]:

```
y_pred_SMOTE_RF2=rf_load.predict(X_test)
```

In [ ]:

```
import time

start_time = time.time()
importances = rf_load.feature_importances_
std = np.std([tree.feature_importances_ for tree in rf_load.estimators_], axis=0)
elapsed_time = time.time() - start_time

print(f"Elapsed time to compute the importances: {elapsed_time:.3f} seconds")
```

In [ ]:

```
print (classification_report(y_test, y_pred_SMOTE_RF2))
```

In [ ]:

```
from sklearn.metrics import auc
y_score = rf_load.predict_proba(X_test)[:,1]
fpr, tpr, _ = roc_curve(y_test, y_score)

plt.title('Curva ROC del modelo Random Forest')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plot(fpr,tpr)
plot((0,1), ls='dashed',color='black')
plt.show()
print ('Area under curve (AUC): ', auc(fpr,tpr))
```

In [ ]:

```
roc_auc_score(y_test, y_score)
```

In [ ]:

```
RFK_probability =rf_load.predict_proba(X_test)[:,1]

acc_RFK=accuracy_score(y_test,y_pred_SMOTE_RF2)
recall_RFK=recall_score(y_test,y_pred_SMOTE_RF2)
precision_RFK=precision_score(y_test,y_pred_SMOTE_RF2)
f1score_RFK=f1_score(y_test,y_pred_SMOTE_RF2)
AUC_RFK=roc_auc_score(y_test,y_score)

print("Accuracy : ", accuracy_score(y_test,y_pred_SMOTE_RF2))
print("Precision:",precision_score(y_test,y_pred_SMOTE_RF2))
print("Recall:",recall_score(y_test,y_pred_SMOTE_RF2))
print("F1-Score:",f1_score(y_test,y_pred_SMOTE_RF2))
print("ROC_AUC Score:",AUC_RFK)
```

In [ ]:

```
acc_balanced_RFK=balanced_accuracy_score(y_test,y_pred_SMOTE_RF2)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,y_pred_SMOTE_RF2))
```

In [ ]:

```
cm=confusion_matrix(y_pred_SMOTE_RF2,y_test, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='RdPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_SMOTE_RF2, labels = [1,0])
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_SMOTE_RF2,normalize='true', l
abels = [1,0])
```

KNN

In [ ]:

```
from sklearn.neighbors import KNeighborsClassifier
modelSMOTE_KNN = KNeighborsClassifier()
modelSMOTE_KNN.fit(X_train_smt, y_train_smt)
y_pred_SMOTE_KNN = modelSMOTE_KNN.predict(X_test)
KNN_probability = modelSMOTE_KNN.predict_proba(X_test)[:,1]

acc_KNN = accuracy_score(y_test, y_pred_SMOTE_KNN)
recall_KNN = recall_score(y_test, y_pred_SMOTE_KNN)
prec_KNN = precision_score(y_test, y_pred_SMOTE_KNN)
f1_KNN = f1_score(y_test, y_pred_SMOTE_KNN)

AUC_KNN =roc_auc_score(y_test,KNN_probability)

print("Accuracy : ", acc_KNN)
print("ROC_AUC Score:",AUC_KNN)
```

In [ ]:

```
AUC_KNN =roc_auc_score(y_test,KNN_probability)

print("ROC_AUC Score:",AUC_KNN)
```

In [ ]:

```
acc_balanced_KNN=balanced_accuracy_score(y_test,y_pred_SMOTE_KNN)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,y_pred_SMOTE_KNN))
```

In [ ]:

```
print(classification_report(y_test, y_pred_SMOTE_KNN))
```

In [ ]:

```
fpr, tpr, _ = roc_curve(y_test, KNN_probability)

plt.title('Curva ROC del modelo KNN')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
cm=confusion_matrix(y_pred_SMOTE_KNN,y_test, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='RdPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_SMOTE_KNN,normalize='true', l
abels = [1,0])
```

CatBoost

In [ ]:

```
X_train_smt_cb['codigo_postal'] = X_train_smt_cb['codigo_postal'].astype(str)
X_train_smt_cb['canal_contacto'] = X_train_smt_cb['canal_contacto'].astype(str)

X_test_cb['codigo_postal'] = X_test_cb['codigo_postal'].astype(str)
X_test_cb['canal_contacto'] = X_test_cb['canal_contacto'].astype(str)
```

In [ ]:

```
from catboost import CatBoostClassifier

cat_features = ['codigo_postal', 'canal_contacto']
cat_model = CatBoostClassifier()
cat_model = cat_model.fit(X_train_smt_cb, y_train_smt_cb, cat_features=[1,6],
                          eval_set = (X_test_cb, y_test_cb), early_stopping_rounds = 10
, verbose = 100)

predictions = [pred[1] for pred in cat_model.predict_proba(X_test_cb)]
print('Validation ROC AUC Score:', roc_auc_score(y_test_cb, predictions, average = 'wei
ghted'))
```

In [ ]:

```
cat_model=model.fit(xtrain,ytrain)
predictions=cat_model.predict(X_test)
cb_probability =cat_model.predict_proba(X_test)[:,1]

acc_cb=accuracy_score(y_test,predictions)
recall_cb=recall_score(y_test,predictions)
precision_cb=precision_score(y_test,predictions)
f1score_cb=f1_score(y_test,predictions)
AUC_cb=roc_auc_score(y_test,cb_probability)

print("Accuracy : ", accuracy_score(y_test,predictions))
print("Precision:",precision_score(y_test,predictions))
print("Recall:",recall_score(y_test,predictions))
print("F1-Score:",f1_score(y_test,predictions))
print("ROC_AUC Score:",AUC_cb)
```

In [ ]:

```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(y_test, cb_probability)

plt.title('Curva ROC del modelo CatBoost')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
from sklearn.metrics import precision_recall_fscore_support
precision_recall_fscore_support(y_test_cb,predictions)
```

In [ ]:

```
acc_balanced_cb=balanced_accuracy_score(y_test,predictions)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,predictions))
```

In [ ]:

```
cm=confusion_matrix(predictions,y_test, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='RdPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,predictions,labels = [1,0])
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,predictions,normalize='true', labels
= [1,0])
```

Red neuronal

In [ ]:

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
import keras
from keras.models import Sequential
from keras.layers import Dense
import warnings
```

Scikit Learn

In [ ]:

```
from sklearn.neural_network import MLPClassifier
clf = MLPClassifier(solver='adam', alpha=1e-7, hidden_layer_sizes=(11, 40,11,40,11,), r
andom_state=19)

clf.fit(X_train_smt, y_train_smt)
```

In [ ]:

```
predictions_rn = neural_network_load.predict(X_test)
```

In [ ]:

```
[coef.shape for coef in neural_network_load.coefs_]
```

In [ ]:

```
rn_probability = neural_network_load.predict_proba(X_test)[:,1]
```

In [ ]:

```
acc_rn=accuracy_score(y_test,predictions_rn.round())
recall_rn=recall_score(y_test,predictions_rn.round())
precision_rn=precision_score(y_test,predictions_rn.round())
f1score_rn=f1_score(y_test,predictions_rn.round())
AUC_rn=roc_auc_score(y_test,rn_probability)
#print accuracy and Auc values of model
print("Accuracy : ", accuracy_score(y_test,predictions_rn.round()))
print("Precision:",precision_score(y_test,predictions_rn.round()))
print("Recall:",recall_score(y_test,predictions_rn.round()))
print("F1-Score:",f1_score(y_test,predictions_rn.round()))
print("ROC_AUC Score:",AUC_rn)
```

In [ ]:

```
acc_balanced_rn=balanced_accuracy_score(y_test,predictions_rn)
print("Accuracy balanced : ", balanced_accuracy_score(y_test,predictions_rn))
```

In [ ]:

```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(y_test, rn_probability)

plt.title('Curva ROC de la red neuronal')
plt.xlabel('Proporción de FP')
plt.ylabel('Proporción de TP (Precision)')

plt.plot(fpr,tpr)
plt.plot((0,1), ls='dashed',color='black')
plt.show()
```

In [ ]:

```
cm=confusion_matrix(predictions_rn,y_test, labels = [1,0])
print(cm)
sns.heatmap(cm,annot=True,cmap='RdPu')
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,predictions_rn, labels = [1,0])
```

In [ ]:

```
cm_=ConfusionMatrixDisplay.from_predictions(y_test,predictions_rn,normalize='true', lab
els = [1,0])
```

Se guarda el modelo

In [ ]:

```
import pickle
filename = 'neural_network.sav'
#pickle.dump(clf, open(filename, 'wb'))
```

In [ ]:

```
X_test = X_test0[['sexo','prima','edad','asegurado_antes','codigo_postal','edad_vehicul
o','daños_vehiculo','canal_contacto']]
```

In [ ]:

```
import pickle
neural_network_load = pickle.load(open('red_neuronal.sav', 'rb'))
```

Hyperparameter tuning

In [ ]:

```
from sklearn.neural_network import MLPClassifier
from sklearn import svm
from sklearn.model_selection import GridSearchCV

#donde el alpha he cambiado 10 por 7 como sugiere el paquete
parameters = {'solver': ['lbfgs', 'adam'], 'max_iter': [5,10], 'alpha': [1e-7, 0.0001,
0.05], 'hidden_layer_sizes':[(11, 40,11,40,11,),(150,100,50), (120,80,40), (100,50,30
)], 'random_state':[10,11,12,13,14,15,16,17,18,19]}
clf = GridSearchCV(MLPClassifier(), parameters, n_jobs=-1, cv=5)

clf.fit(X_train_smt, y_train_smt)
print(clf.score(X_train_smt, y_train_smt))
print(clf.best_params_)
```

In [ ]:

```
predictions = clf.predict(X_test)
```

In [ ]:

```
acc_cb=accuracy_score(y_test,predictions.round())
recall_cb=recall_score(y_test,predictions.round())
precision_cb=precision_score(y_test,predictions.round())
f1score_cb=f1_score(y_test,predictions.round())
AUC_cb=roc_auc_score(predictions.round(),y_test)

print("Accuracy : ", accuracy_score(y_test,predictions.round()))
print("Precision:",precision_score(y_test,predictions.round()))
print("Recall:",recall_score(y_test,predictions.round()))
print("F1-Score:",f1_score(y_test,predictions.round()))
print("ROC_AUC Score:",AUC_cb)
```

10. Resumen de resultados

In [ ]:

```
resultados = {
    "XGB": [acc_xgb,acc_balanced_xgb,precision_xgb,recall_xgb,f1score_xgb,AUC_xgb],
    "XGB tuned": [acc_xgb_tuned,acc_balanced_xgb_tuned,precision_xgb_tuned,recall_xgb_t
uned,f1score_xgb_tuned,AUC_xgb_tuned],
    "Regresión Logística": [acc_logreg,acc_balanced_logreg,prec_logreg,recall_logreg,f1
_logreg,AUC_LR],
    "KNN": [acc_KNN,acc_balanced_KNN, prec_KNN, recall_KNN, f1_KNN, AUC_KNN],
    "Árbol de Decisión": [acc_DT,acc_balanced_DT, prec_DT, rec_DT, f1_DT, AUC_DT],
    "Random Forest" : [acc_RF,acc_balanced_RF, prec_RF, rec_RF, f1_RF, AUC_RF],
    "Random Forest K" : [acc_RFK,acc_balanced_RFK, precision_RFK, recall_RFK, f1score_R
FK, AUC_RFK],
    "CatBoost": [acc_cb,acc_balanced_cb, precision_cb, recall_cb, f1score_cb, AUC_cb],
    "Red neuronal": [acc_rn,acc_balanced_rn, precision_rn, recall_rn, f1score_rn, AUC_r
n]
    }

resumen = pd.DataFrame(data = resultados, index = ['Accuracy','Accuracy balanced','Prec
ision', 'Recall', 'F1-Score', 'AUC'])
resumen
```