Trabajo Fin de Máster

Aplicación de técnicas estadísticas para modelos de clasificación supervisada con muestras de datos desbalanceadas

Alejandro Arias Piñeiro

Máster en Técnicas Estadísticas

Curso 2021-2022

Propuesta de Trabajo Fin de Máster

Título en galego: Aplicación de técnicas estatísticas para modelos de clasificación supervisada con mostras de datos desequilibrada
Título en español: Aplicación de técnicas estadísticas para modelos de clasificación supervisada con muestras de datos desbalanceadas
English title: Application of statistical techniques for models of supervised classification with data samples unbalanced
Modalidad: Modalidad B
Autor/a: Alejandro Arias Piñeiro, Universidade da Coruña
Director/a: Manuel Oviedo de la Fuente, Universidade da Coruña
Tutor/a: Jorge García Romarís, ABANCA
Breve resumen del trabajo: Desarrollo de modelos de clasificación sobre datos con distribuciones muy desbalanceadas. Particularmente, modelos de prevención de ciberfraude.
Recomendaciones: Interés por trabajar en un banco innovador y comprometido con su entorno que trabaja con y para las empresas y las familias. Capacidad de trabajo autónomo y habilidades comunicativas. Interés y disposición para trabajar en un equipo multidisciplinar de alto desempeño.
Otras observaciones: El alumno se incorporará a la Entidad en calidad de Estudiante en Prácticas. Estas prácticas están remuneradas.

Don Manuel Oviedo de la Fuente, profesor da Universidade da Coruña, don Jorge García Romarís, Director de Data, Analytics & R2Tech de ABANCA, informan que el Trabajo Fin de Máster titulado

Aplicación de técnicas estadísticas para modelos de clasificación supervisada con muestras de datos desbalanceadas

fue realizado bajo su dirección por don Alejandro Arias Piñeiro para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En [lugar], a xx de [mes] de 2022.

OVIEDO DE LA FUENTE MANUEL - 46766202A

Firmado digitalmente por OVIEDO DE LA FUENTE MANUEL - 46766202A
Fecha: 2022.06.11 19:59:32 +02'00'

El director:
Don Manuel Oviedo de la Fuente

El tutor:
Don Jorge García Romarís

El autor:
Don Alejandro Arias Piñeiro

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el/la autor/a declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,...sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen

Resumen en español

En las entidades bancarias, es fundamental la detección temprana de fraudes electrónicos con el fin de ahorrar los costes asociados a dichos fraudes. En este trabajo partiremos de un dataset formado por diferentes variables relacionadas con una transacción bancaria, como pueden ser el importe de la transacción o el navegador y predeciremos si dicha transacción es fraudulenta o no.

Para obtener dicha predicción, es necesario primero realizar el preprocesado de los datos, donde nos encargaremos de la limpieza de los mismos y la selección de diferentes variables, para posteriormente, testear diversos modelos y ver cual se adecua más a nuestros objetivos. De esta forma, empezaremos trabajando con modelos clásicos como las regresiones logísticas para, posteriormente, utilizar modelos más actuales como el Random Forest o el Gradient Boosting Machine. Tras ajustar dichos modelos, estos serán comparados en función de tres métricas, las cuales son la *precision*, la *recall* y el F_1 *score*.

Posteriormente, evaluaremos en detalle el modelo que obtenga mejores resultados, realizando gráficos Shap y viendo cuales son aquellas variables que afectan más a las observaciones mal clasificadas. Por último, extraeremos conclusiones del estudio y mencionaremos otras alternativas.

English abstract

In banking institutions, early detection of electronic fraud is essential in order to save the costs associated with such frauds. In this work we will start from a dataset formed by different variables related to a bank transaction, such as the amount of the transaction or the browser, and we will predict whether the transaction is fraudulent or not.

To obtain this prediction, it is first necessary to preprocess the data, where we will take care of the cleaning of the data and the selection of different variables, and then test different models and see which one best suits our objectives. In this way, we will start working with classical models such as logistic regressions to later use more current models such as the Random Forest or the Gradient Boosting Machine. After fitting these models, they will be compared according to three metrics, which are the *precision*, the *recall* and the $F_1 score$.

Subsequently, we will evaluate in detail the model that obtains the best results, making Shap plots and seeing which are those variables that most affect the misclassified observations. Finally, we will draw conclusions from the study and mention other alternatives.

Prefacio

En los últimos años, el uso de negocios y servicios online ha aumentado de manera considerable, por lo que debido a esto, el sistema ha experimentado un aumento de las transacciones electrónicas y el pago de facturas online. Todas estas tecnologías facilitan nuestra vida, sin embargo, debido a ellas, también se han creado nuevos tipos de fraude utilizando estos medios como pueden ser: pago con tarjetas fraudulento, fraudes al seguro, robo online de identidad, transacciones bancarias fraudulentas...

Además, estos tipos de fraude son cambiantes y con el desarrollo de la tecnología van surgiendo nuevos métodos. Debido a ellos, anualmente se pierden varios millones de euros solamente con el dinero estafado, además de afectar negativamente a la confianza del cliente y en el sistema bancario, pudiéndose ver seriamente perjudicada la reputación de un banco.

En concreto nos centraremos específicamente en las transacciones bancarias fraudulentas, siendo fundamental en las entidades bancarias la detección temprana de fraudes electrónicos con el objetivo de ahorrar los costes mencionados anteriormente. Sin embargo, este no es un problema clásico de clasificación binaria (fraude vs no fraude), ya que debido a que los datos correspondientes a la clase no fraudulenta son mucho más numerosos que los de la clase fraudulenta tenemos como consecuencia un conjunto de datos muy desbalanceado. Debido a esto, debemos de tener especial cuidado a la hora de tratar y limpiar los datos, ya que en caso de hacerlo erróneamente podríamos ajustar un modelo que nos lleve a interpretaciones erróneas y sesgos. Además, muchos de los modelos de machine learning no funcionarán correctamente con unos datos tan desbalanceados.

Otro detalle importante es la funcionalidad del modelo, ya que no nos interesa unicamente un modelo el cual sea capaz de predecir perfectamente los casos fraudulentos, si no que también nos interesa que dicho modelo sea capaz de dar una respuesta rápida a la pregunta de si una transacción es o no es fraudulenta. En caso de que esta respuesta no se diese de manera veloz, cierto tipo de operaciones fraudulentas se realizarían igualmente, dando lugar a que el modelo fuese inservible.

En este trabajo, empezaremos realizando el análisis exploratorio de los datos, desde tratar con los valores faltantes y la creación de nuevas variables hasta el tratamiento de los outliers, ya que la extracción de estos puede mejorar considerablemente las métricas usadas para evaluar los modelos. Además, en este tipo de problemas es habitual tener un gran número de variables, por lo que seleccionar adecuadamente las que vamos a utilizar será un paso muy importante. Para ello, realizaremos un estudio de las correlaciones a través de la correlación de Pearson y correlación de la distancia.

Tras esto utilizaremos diferentes pruebas de bondad de ajuste como Kolmogorov-Smirnov para las variables continuas y un test χ^2 para las variables discretas. En ambos casos nos servirá para comparar las distribuciones de las clases fraudulenta y no fraudulenta, pudiendo eliminar las variables en las cuales estas distribuciones sean parecidas.

Después de esto, los datos ya estarán listo para poder aplicarlos en los diferentes modelos. De esta forma, separaremos la muestra en datos de entrenamiento y datos test, aplicando validación cruzada

en el conjunto de datos de entrenamiento con el objetivo de encontrar los hiperparámetros óptimos en los modelos de machine learning.

Una vez aplicada la validación cruzada encararemos directamente uno de los principales problemas de la prevención del fraude, las muestras desbalanceadas. Para ello, consideraremos los siguientes métodos para remuestrear los datos: undersampling, oversampling, SMOTE (Synthetic Minority Oversampling Technique) y CT-GAN (Tabular Generative Adversarial Networks), siendo el objetivo de estos métodos tener un dataset balanceado en el que funcionen de manera más adecuada las técnicas de machine learning que aplicaremos.

De esta forma, entrenaremos cada uno de los modelos utilizando el conjunto de datos desbalanceado y los conjuntos con los datos remuestreados, donde las métricas a evaluar serán la *precision*, la *recall* y el F_1 *score*. Consideraremos estas métricas debido al carácter desbalanceado de los datos, ya que la métrica más común en los problemas de clasificación es el *accuracy*, pero con datos desbalanceados podría llevar a resultados engañosos.

Los modelos a evaluar serán la Regresión Logística, Bosques aleatorios, Gradient Boosting Extremo y Redes Neuronales. Cada uno de ellos será evaluado a través de las métricas mencionadas anteriormente en el conjunto de datos test. Tras esto, nos centraremos en el modelo que devuelva mejores resultados, viendo cuales fueron sus variables más importantes y estudiando las observaciones mal clasificadas mediante gráficos Shap.

Capítulo 1

El Problema del Fraude

En este capítulo contextualizaremos el Problema de Fraude en un marco global, aportando datos sobre los costes asociados con el fraude y sus tendencias. Posteriormente, llevaremos a cabo el planteamiento del problema, el cual nos servirá de guía a través de todo el trabajo.

De esta forma, en la Sección 1.1 destacaremos ciertos puntos del 17º reporte de fraude con tarjetas del Banco Central Europeo (BCE) y hablaremos de los costes asociados con el fraude. Tras esto, en la Sección 1.2 haremos un resumen de los diferentes pasos a realizar en el trabajo.

1.1. Contexto del Fraude con tarjetas

Hemos utilizado los dos siguientes artículos para contextualizar el alcance del fraude y sus repercusiones en el mundo actual.

1.1.1. Reporte de fraude de tarjetas

Para hacernos una idea del alcance de las transacciones fraudulentas y sus costes, hemos resumido los siguientes segmentos de texto del reporte Bank. (2021) del BCE.

Valor total de las transacciones fraudulentas

El valor total de las transacciones fraudulentas utilizando tarjetas emitidas dentro de SEPA (Single Euro Payments Area) y adquirida en todo el mundo ascendió a 1.870 millones de euros en 2019. Solo en el caso de las tarjetas emitidas en la zona del euro, el valor total de las transacciones con tarjetas fraudulentas ascendió a 1.030 millones de euros.

Medios de fraude

La gran mayoría de las transacciones fraudulentas siguen estando relacionadas con el fraude con tarjeta no presente (CNP), es decir, pagos a través de Internet, correo o teléfono. En 2019 el 80 % del valor del fraude con tarjeta fue resultado de transacciones CNP. Por el contrario, las transacciones en terminales de punto de venta físicos (TPV), como los pagos presenciales en comercios o restaurantes y en cajeros automáticos (ATM) sólo representaron el 15 % y el 5 % del valor total del fraude con tarjeta en 2019, respectivamente.

Valor total de las transacciones fraudulentas con tarjeta no presente

El fraude CNP representó 1.500 millones de euros en pérdidas por fraude en 2019, un $4,3\,\%$ más que el año anterior. Los datos parcialmente disponibles sobre el total de transacciones CNP sugieren que el

fraude creció a un ritmo considerablemente más lento que las transacciones CNP totales en 2019. Con respecto a transacciones con tarjeta, el fraude cometido en los terminales de punto de venta aumentó un $2,2\,\%$ en 2019, mientras que el fraude cometido en cajeros automáticos disminuyó un $6,1\,\%$.

Destinos de las transacciones fraudulentas

Por otro lado, más de la mitad del valor total del fraude en 2019 estuvo relacionado con las transacciones transfronterizas dentro de la SEPA. Desde una perspectiva geográfica, las transacciones nacionales representaron el $89\,\%$ del valor de todas las transacciones con tarjeta en 2019, pero solo el $35\,\%$ de las las transacciones fraudulentas. Las transacciones transfronterizas dentro de la SEPA representaron el $9\,\%$ de todas las transacciones con tarjeta en términos de valor, pero el $51\,\%$ del fraude notificado. Aunque solo el $2\,\%$ de todas las transacciones se adquirieron fuera de la SEPA, representaron el $14\,\%$ del fraude.

Valor medio de una transacción fraudulenta

Mientras que el número de transacciones fraudulentas con tarjeta creció a un ritmo más rápido que el valor correspondiente del fraude en 2019, el valor medio de una transacción fraudulenta siguió disminuyendo. En 2019, una transacción fraudulenta ascendió de media a unos 77 euros, lo que supone aproximadamente un $41\,\%$ menos que la cifra registrada en 2015. Este descenso del importe medio del fraude fue sustancialmente más pronunciado que el correspondiente descenso del pago medio con tarjeta en general. Podemos ver esto en la Figura (1.1).

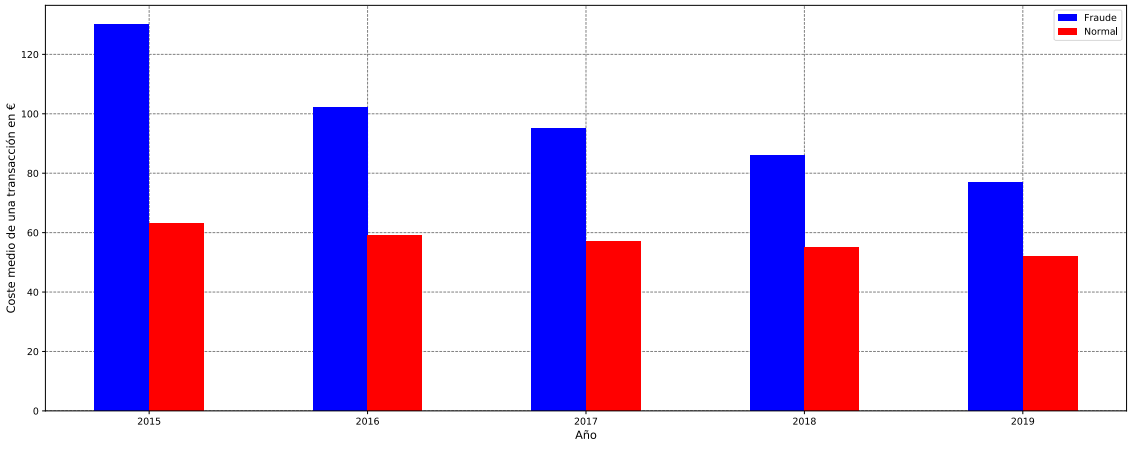

Figura 1.1: Coste medio de las transacciones fraudulentas y no fraudulentas correspondientes a los años 2015-1019. Fuente: Todos los operadores de sistemas de pago con tarjeta que informan.

Al igual que en años anteriores, la proporción de fraude en el valor total de las transacciones con tarjeta varió considerablemente entre los Estados miembros de la UE en 2019. Desde la perspectiva emisora, las tasas de fraude en Francia el Reino Unido y España fueron las más altas en 2019, mientras que las tasas de Rumanía, Hungría y Polonia estuvieron entre las más bajas.

Conclusiones del artículo

En general, el panorama del fraude con tarjetas ha mejorado ligeramente desde la última edición de este informe, pero la industria, los reguladores y los consumidores deben permanecer atentos. Aunque el fraude en términos relativos disminuyó ligeramente en 2019, los niveles generales siguen siendo

persistentes y están aumentando en términos absolutos. Además, el reciente aumento de los pagos con tarjeta para compras en línea durante la pandemia de COVID-19 puede hacer que cada vez más dichos pagos sean el objetivo de actividades delictivas (por ejemplo, mediante phishing). En consecuencia, se anima a los sistemas de pago con tarjeta y a sus PSP (proveedores de servicios de pago) participantes a que sigan intercambiando información relacionada con la prevención del fraude y las mejores prácticas de seguridad de seguridad.

1.1.2. Los verdaderos costes del fraude

Por otro lado, podemos hacernos una idea de los coste que conlleva el fraude en los consumidores a través del artículo LTD (2022) del cual hemos sacado los siguientes fragmentos:

Un informe de 2014 mostró que la empresa típica pierde el 5 % de los ingresos anuales debido al fraude. Cuando esto se aplica al PIB mundial del año anterior, el resultado es un estimado de 3,7 billones de dólares en pérdidas anuales por fraude. Esta cifra es, como mínimo, asombrosa. Lo que es aún más asombroso es que sólo un 14 % de las empresas son capaces de recuperarse completamente de las transacciones no autorizadas y otras actividades fraudulentas.

Las actividades fraudulentas han aumentado rápidamente en los últimos años, impulsadas por la tecnología y por los planes de fraude más complejos de los delincuentes. Un estudio de Lexis Nexis de 2016 sobre el verdadero coste del fraude ponderó los datos de los comerciantes estadounidenses centrándose en las transacciones fraudulentas que se intentaron al mes, en comparación con las que tuvieron éxito. Los resultados mostraron que de 2012 a 2016, los intentos de fraude por mes aumentaron de 170 a 442, un incremento del 160 %. Estos mismos datos mostraron que solo en 2016, los defraudadores tuvieron éxito en el 47 % de las transacciones fraudulentas contra los comerciantes. Estos datos son bastante alarmantes cuando se anualizan los datos solo en 2016, 5.304 intentos de transacciones fraudulentas por año, 2.493 transacciones exitosas, ¡o 7 transacciones fraudulentas exitosas por día! Las cifras son asombrosas en esta encuesta sobre cómo el fraude está afectando negativamente a las empresas y los bancos a nivel mundial.

El aumento de los costes del fraude como porcentaje de los ingresos debería ser motivo de preocupación para la economía mundial. El mismo estudio de Lexis Nexis descubrió que, entre 2013 y 2016, el coste del fraude como porcentaje de los ingresos de los comerciantes creció del $0,51\,\%$ al $1,47\,\%$. Cuando el fraude se lleva de los ingresos de las empresas, los costes adicionales suelen repercutir en el mercado, lo que demuestra que el fraude es realmente un problema global que afecta a los de todo el mundo. El coste por dólar de las pérdidas por fraude también está aumentando rápidamente, lo cual es una tendencia preocupante tanto para las empresas como para los bancos. En 2016, los comerciantes estadounidenses informaron de que el coste total por dólar de las pérdidas por fraude era de 2,40 dólares.

Esto significa que por cada dólar de pérdidas, los comercios deben gastar 2,40 dólares más en devoluciones de cargos, tasas y reposición de mercancía. ¿Se pregunta a dónde van a parar estos costes? Correcto, al consumidor, ya que muchas empresas simplemente no pueden permitirse retener estas pérdidas y mantenerse a flote. Estas estadísticas muestran exactamente cómo el fraude nunca es un problema localizado, sino que afecta a los mercados financieros de todo el mundo.

1.2. Planteamiento del Problema

En este trabajo abordamos el problema de la prevención del fraude desde la perspectiva de la entidad ABANCA, donde dado un conjunto de datos sobre transacciones, el objetivo será crear un

modelo capaz de predecir si una nueva transacción es o no fraudulenta y llevarlo a producción. Esto lo haremos mediante los siguientes pasos:

- **Análisis Exploratorio de los Datos**: Llevaremos a cabo las transformaciones necesarias del conjunto de datos para poder aplicar los modelos de clasificación.
  - Limpieza de los datos: Empezaremos limpiando los datos faltantes. Este procedimiento lo llevaremos a cabo sustituyendo el dato faltante por una predicción.
  - Creación de Variables: A partir de las variables dadas, crearemos nuevas variables que puedan ser de utilidad a la hora de ajustar nuevos modelos. También eliminaremos aquellas que no aporten información o resulten redundantes.
  - Representación Gráfica de los Datos: Representaciones gráficas de las densidades de probabilidad de diferentes variables, diferenciándolas en función de la categoría a la que pertenezcan. Esto nos dará una idea de las variables más importantes y como desarrollar futuros modelos de machine learning.
  - Estudio de correlaciones: Estudio de correlaciones utilizando tanto la correlación lineal de Pearson como la correlación de la distancia. Esta última nos permitirá conocer relaciones no lineales entre las variables, de manera que podamos tomar un subconjunto de estas a la hora de calcular los diferentes modelos.
  - Test de Hipótesis: Realizaremos diferentes pruebas de bondad de ajuste a los datos continuos y categóricos. Llevaremos a cabo la prueba de Kolmogorov-Smirnov para los datos continuos mientras que usaremos un test χ^2 para los datos categóricos. Estas pruebas nos servirán para contrastar las distribuciones de las variables en las clases fraudulenta y no fraudulenta, eliminando aquellas variables que no aporten información en la distinción de estas clases.
  - Tratamiento de Outliers: Consideramos outliers aquellos datos que se desvían del comportamiento normal. Estos datos pueden afectar de manera extrema a las estimaciones de ciertos modelos de machine learning, por lo que su tratamiento es de vital importancia.
- **Remuestreo de los datos de entrenamiento**: Separaremos la muestra en dos. Utilizaremos un 90 % de los datos para ajustar los modelos, mientras que el 10 % restante la usaremos para comprobar el funcionamiento de estos. Llevaremos a cabo un remuestreo de los datos de entrenamiento con la idea de aumentar la proporción de observaciones en la clase fraudulenta. Esto los haremos utilizando undersampling, oversampling, SMOTE y Ct-GAN. Es importante dejar la muestra de datos test al margen sin remuestrear.
- **Comparación de modelos**: Con el objetivo de obtener el mejor modelo de predicción, compararemos los diferentes modelos mediante un criterio, aquel que obtenga un mejor valor será el modelo elegido.
  - Regresión Logística: Ajustaremos un modelo de regresión logística clásico. Esto nos servirá como punto de partida de diferentes modelos de machine learning y empezaremos a obtener los primeros resultados respecto a las diferentes métricas que consideraremos en los problemas.
  - Regresión Logística: Tras esto ajustaremos un modelo de Random Forest. Esta es una de las técnicas más utilizadas en los problemas de clasificación y será una de las candidatas a obtener mejores resultados.
  - XGBoost: Llevaremos a cabo XGBoost, el cual es un modelo ensamblado a partir de árboles de decisión. Nuevamente, es una técnica puntera en lo que se refiere a problemas de clasificación y de la que se espera obtener buenos resultados.
  - Redes Neuronales: También ajustaremos varios modelos de redes neuronales, los cuales están muy en auge hoy en día y representan una opción muy interesante debido a la flexibilidad que aportan dichos modelos.

- Comparación de los modelos: Compararemos los diferentes modelos en función de las métricas consideradas y sacaremos conclusiones.

Capítulo 2

Revisión Teórica

En este capítulo revisaremos diferentes conceptos teóricos que utilizaremos a lo largo del trabajo. Entre estos conceptos veremos:

- Análisis de las correlaciones
- Tratamiento de Outliers
- Contrastes de Bondad de Ajuste
- Remuestreo de los Datos
- Modelos de predicción
- Criterios de selección de modelos

Empezaremos definiendo teóricamente las correlaciones de Pearson y de la distancia en la Sección 2.1. En la Sección 2.2 comentaremos lo que es un outlier y el tratamiento de este tipo de datos. La Sección 2.3 la dedicaremos a revisar diferentes contrastes de bondad de ajuste y su utilidad. Posteriormente, en la Sección 2.4 hablaremos de las técnicas de remuestreo de datos más comunes. Finalmente, en las Secciones 2.6 y 2.7 comentaremos la teoría tras los diferentes modelos de predicción y las métricas que usaremos para cuantificar sus resultados.

2.1. Análisis de las correlaciones

El Análisis de las correlaciones es una técnica habitual en el análisis de datos y se utiliza antes de construir un modelo. Este análisis nos indica las relaciones que hay entre dos variables, es decir, si una variable nos da información acerca de la otra. El estudio de estas correlaciones es importante de cara a obtener un modelo más simple, el cual sea fácilmente interpretable y con un coste computacional menor y, por lo tanto, un menor tiempo de ejecución. Estudiaremos dos tipos de correlación:

- Correlación de Pearson
- Correlación de la Distancia

2.1.1. Correlación de Pearson

El coeficiente de correlación de Pearson fue desarrollado por Karl Pearson a partir de una idea introducida por Francis Galton en la década de 1880. Este coeficiente es una medida de dependencia

lineal entre dos variables aleatorias cuantitativas. Sea $((x_1, y_1), \ldots, (x_n, y_n))$ una muestra de datos, definiremos el coeficiente de correlación de Pearson muestral como:

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{2.1}$$

donde:

- n es el tamaño de la muestra.
- x_i, y_i son puntos muestrales individuales indexados con i.
- $\bar{x}$ denota la media muestral definida por $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ (análogamente para $\bar{y}$).

De esta forma, distinguiremos 3 casos:

- $1 > r_{xy} > 0$ indica una correlación positiva, altos valores de una de las variables suelen ir acompañados por altos valores de la otra variable.
- $0 > r_{xy} > -1$ indica una correlación negativa, bajos valores de una de las variables suelen ir acompañados por altos valores de la otra variable.
- $r_{xy} = 0$ indica que no existe relación lineal entre las variables.

Podemos utilizar el coeficiente de correlación de Pearson para hacer una selección de las variables del problema, ya que aquellas muy relacionadas linealmente entre ellas podrían estar expresando la misma información. Por otro lado, el problema de este coeficiente es que es una medida de la relación lineal entre dos variables, pero en caso de que esta relación no sea lineal el coeficiente podría no mostrarlo. Por ello, consideraremos también la correlación de la distancia.

2.1.2. Correlación de la distancia

La correlación de la distancia fue introducida en 2005 por Gábor J. Székely para solucionar las deficiencias de la correlación de Pearson. El coeficiente de correlación de la distancia poblacional es 0 si y solo si los vectores aleatorios son independientes. La correlación de la distancia mide tanto la asociación lineal como la no lineal entre dos variables aleatorias o vectores aleatorios.

Para definirla, debemos de mencionar primero la covarianza de la distancia de la muestra. Sea (X_k, Y_k) con $k = 1, 2, \ldots, n$ una muestra estadística de un par de valores reales o variables aleatorias con valores vectoriales (X, Y). Primero, calcularemos las matrices de distancias $(a_{j,k})$ y $(b_{j,k})$ que contiene todas las distancias por pares:

$$a_{j,k} = \|X_j - X_k\|, \qquad j, k = 1, 2, \ldots, n,$$
$$b_{j,k} = \|Y_j - Y_k\|, \qquad j, k = 1, 2, \ldots, n,$$

donde $\|.\|$ denota la norma vectorial. Luego, se toman todas las distancias doblemente centradas:

$$A_{j,k} := a_{j,k} - \overline{a}_{j\cdot} - \overline{a}_{\cdot k} + \overline{a}_{\cdot\cdot}, \qquad B_{j,k} := b_{j,k} - \overline{b}_{j\cdot} - \overline{b}_{\cdot k} + \overline{b}_{\cdot\cdot},$$

donde $\overline{a}_{j\cdot}$ es la media de la fila j-ésima, $\overline{a}_{\cdot k}$ es la media de la columna k-ésima, y $\overline{a}_{\cdot\cdot}$ es la gran media de la matriz de la distancia de la muestra X. La notación es similar para los valores de b. En las matrices de la distancias centradas $(A_{j,k})$ y $(B_{j,k})$, todas las filas y todas las columnas suman cero.

La covarianza de la distancia de la muestra al cuadrado es simplemente el promedio aritmético de los productos $A_{j,k}, B_{j,k}$:

$$\mathrm{dCov}_n^2(X,Y) := \frac{1}{n^2}\sum_{j=1}^{n}\sum_{k=1}^{n} A_{j,k}\, B_{j,k}. \tag{2.2}$$

Por otro lado, la varianza de la distancia de una muestra es la raíz cuadrada de:

$$\mathrm{dVar}_n^2(X) := \mathrm{dCov}_n^2(X,X) = \tfrac{1}{n^2}\sum_{k,\ell} A_{k,\ell}^2, \tag{2.3}$$

Utilizando (2.2) y (2.3), llegamos a la definición de correlación de la distancia:

$$\mathrm{dCor}(X,Y) = \frac{\mathrm{dCov}(X,Y)}{\sqrt{\mathrm{dVar}(X)\ \mathrm{dVar}(Y)}} \tag{2.4}$$

2.2. Tratamiento de los Outliers

En estadística, un outlier es una observación que es numéricamente distante del resto de los datos. Estos valores serán especialmente peligrosos cuando además de ser puntos atípicos, sean influyentes, es decir, si al ajustar un modelo sin esa observación, el modelo cambia de manera notable.

Es importante tratar estos puntos con especial cuidado, ya que pueden ser indicativos de datos que pertenecen a una población diferente del resto de las muestras establecidas o datos mal tomados. Además, existen diferentes métodos para clasificarlos, siendo el más utilizado el rango intercuartílico. A pesar de ello, usaremos la Z-Score. La detección de atípicos la llevaremos a cabo unicamente en la clase no fraudulenta con el fin de no eliminar observaciones de la clase fraudulenta.

2.2.1. Z-score

Sea X una muestra de una variable aleatoria con media muestral $\bar{x}$ y desviación típica muestral S definimos la Z-score de una observación x_i como:

$$z_i = \frac{x_i - \bar{x}}{S}. \tag{2.5}$$

Este valor nos dará una idea de como de lejos está una observación de su media, así, una Z-score igual a 1, significará que el valor esta a una desviación estándar de la media. De esta forma, aquellos valores que se encuentren muy lejos de ella podrían ser considerados como outliers y por lo tanto eliminados.

Sea t el valor límite o threshold, tomaremos el valor usual $t = 3$, eliminando aquellas observaciones que estén 3 desviaciones estándar por encima de la media.

$$[\bar{x} - 3S_{n-1}, \bar{x} + 3S_{n-1}] \tag{2.6}$$

donde $\bar{x}$ es la media muestral de la variable X, S_{n-1} es la cuasidesviación típica muestral de la variable X y n es el tamaño muestral.

2.3. Contrastes de Bondad de Ajuste

Sean $(X_1, \ldots, X_n)$ y $(Y_1, \ldots, Y_m)$ dos muestras aleatorias simples con distribuciones F_x y F_y respectivamente, un contraste de bondad de ajuste consiste en decidir si puede admitirse que la distribución de ambas muestras coincide, o, si por el contrario, tienen diferente distribución. De esta forma, podemos plantear un contraste de hipótesis, de la siguiente manera:

$$H_0 : F_x = F_y$$

$$H_1 : F_x \neq F_y$$

Además, debido a que la muestra de datos contiene variables numéricas y categóricas, realizaremos diferentes contrastes de bondad de ajuste. De esta forma, a las variables numéricas les realizaremos un contraste de Kolmogorov-Smirnov, mientras que a las variables categóricas les realizaremos un contraste χ^2.

2.3.1. Contraste de Kolmogorov-Smirnov

Sea $(Z_1, \ldots, Z_p)$ una muestra aleatoria simples y ordenada de tamaño muestral p, definiremos la función de distribución empírica de los datos como:

$$\hat{F}_p(z) = \begin{cases} 0 \text{ si } z < Z_1 \\ \frac{k}{p} \text{ si } Z_k \leq z < Z_{k+1} \text{ para } k = 1, 2, \ldots, p-1 \\ 1 \text{ si } z \geq Z_p \end{cases} \tag{2.7}$$

Esta función de distribución empírica es útil debido a que usualmente, en la práctica, no conocemos las funciones de distribución de los datos, por lo que debemos de estimarlas.

El contraste de Kolmogorov-Smirnov se basa en comparar la distribución empírica $\hat{F}_n$ obtenida de la muestra $(X_1, \ldots, X_n)$ con la distribución empírica $\hat{F}_m$ obtenida de la muestra $(Y_1, \ldots, Y_m)$, obteniendo el siguiente estadístico:

$$D_{m,n} = \max_x |\hat{F}_n(x) - \hat{F}_m(x)|,$$

De esta forma, rechazaremos la hipótesis nula H_0 cuando:

$$D_{m,n} \geq c_\alpha \sqrt{\frac{m+n}{mn}},$$

donde c_α vienen dados por:

$$c_\alpha = \sqrt{-\frac{1}{2}\ln(\frac{\alpha}{2})}$$

2.3.2. Contraste χ^2

Por otro lado, el contraste χ^2 mide la discrepancia entre dos distribuciones observadas, indicando si existen diferencias significativas entre ambas. Además, utilizaremos esta prueba para contrastar las distribuciones de las variables categóricas.

De esta forma, sean A y B dos variables categóricas con las categorías $A_1, \ldots, A_k$ y $B_1, \ldots, B_l$ respectivamente y sea una muestra aleatoria simple $((X_1, Y_1), \ldots, (X_n, Y_n))$, podemos obtener la tabla de contingencia para representar la relación entre ambas como se muestra en el Cuadro (2.1), donde $n_{i,j}$ representa el número de pares (i, j) que existen en los datos, $n_{i\bullet}$ representa el número de observaciones

X Y	B_1	B_2	...	B_l	Total
A_1	n_{11}	n_{12}	...	n_{1l}	$n_{1\bullet}$
A_2	n_{21}	n_{22}	...	n_{2l}	$n_{2\bullet}$
...	...	...	...	...	...
A_k	n_{k1}	n_{k2}	...	n_{kl}	$n_{k\bullet}$
Total	$n_{\bullet 1}$	$n_{\bullet 2}$	...	$n_{\bullet l}$	$n_{\bullet\bullet}$

Cuadro 2.1: Tabla que muestra la malla de posibles valores.

de la clase A_i, $n_{\bullet j}$ representa el número de observaciones de la clase B_j y n es el total de datos de la muestra.

Definiremos entonces el test χ^2 como:

$$\chi^2 = \sum_{i,j} \frac{(n_{i,j} - e_{i,j})^2}{e_{i,j}},$$

donde $e_{i,j}$ es la frecuencia esperada y la calcularemos como:

$$e_{i,j} = n_{i\bullet} \Delta n_{\bullet,j}/n$$

Si las frecuencias $e_{i,j}$ son realmente los valores esperados de las frecuencias $n_{i,j}$, se puede calcular un parámetro que dependa de ambas que tiene distribución χ^2. Por otra parte, si las variables no son independientes, las diferencias entre las series de frecuencias observadas y esperadas serán mayores que las atribuibles al efecto del azar y, al estar elevadas al cuadrado en el numerador de la expresión anterior, ésta tenderá a ser mayor que lo que suele ser el valor de una variable χ^2. Rechazaremos el estadístico de contraste cuando χ^2 sea mayor que un valor previamente prefijado.

2.4. Remuestreo de los Datos

Como hemos mencionado en el capítulo introductorio, uno de los principales problemas a resolver es el carácter desbalanceado de los datos. En el conjunto de datos a considerar tenemos unicamente 644 observaciones de la clase fraudulenta, representando un 0,27 % del total. Con el objetivo de aumentar el número de observaciones en esta clase, testearemos los siguientes métodos de remuestreo:

- Undersampling
- Oversampling
- SMOTE
- Ct-GAN

Todos ellos son técnicas estadísticas diseñadas para equilibrar la distribución de clases para un conjunto de datos de clasificación en el que el número de elementos de una clase es mucho menor que en la otra.

2.4.1. Undersampling y Oversampling

Las técnicas de undersampling eliminan ejemplos del conjunto de datos de la clase mayoritaria para equilibrar las clases y reducir el sesgo. De esta forma, si en nuestro conjunto de datos tenemos 644 datos de la clase fraudulenta, tomaremos una submuestra aleatoria de 644 datos de la clase no fraudulenta, quedándonos con un conjunto de datos de 1288 observaciones en total.

Por otro lado, las técnicas de oversampling aumentan el conjunto de datos con múltiples copias de la clase minoritaria. El sobremuestreo se puede realizar más de una vez y es una de los métodos de remuestreo mas sencillos. Utilizando esto, aumentaríamos el número de datos de la clase fraudulenta hasta 237276, por lo que tendríamos un dataset formado por 474552 observaciones. Sin embargo, los datos de la clase fraudulenta estarían repetidos múltiples veces, por lo que es posible que al aplicar ciertos métodos de machine learning no mejore el resultado o que se produzca overfitting.

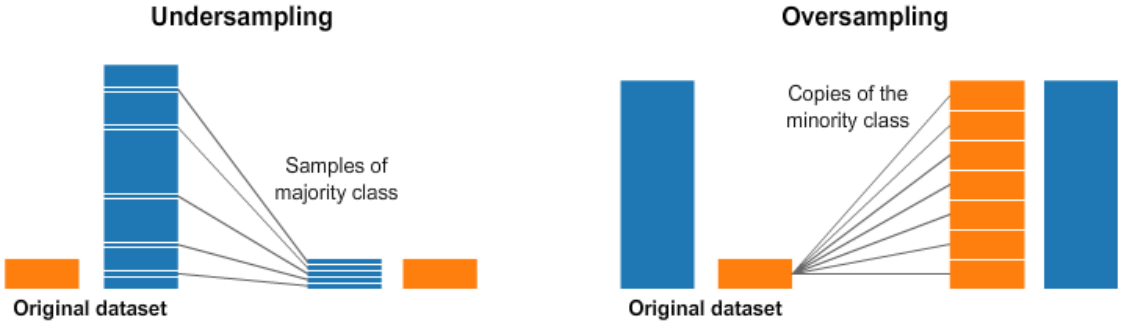

Figura 2.1: Funcionamiento de las técnicas de remuestreo undersampling (izquierda) y oversampling (derecha). Imagen obtenida en el Artículo Al-Serw (2021).

2.4.2. SMOTE (Synthetic Minority Oversampling Technique)

Como hemos mencionado, al utilizar oversampling simplemente estamos duplicando datos ya conocidos, pero no obtenemos nueva información. Una mejora a esto sería sintetizar nuevos datos de la clase minoritaria, lo cual suele resultar más efectivo y es conocido como SMOTE, introducido en Chawla et al. (2002). Esta técnica funciona seleccionando ejemplos que están próximos en el espacio de variables y tomando nuevos puntos en las rectas que unen dichos ejemplos. Podemos observar el funcionamiento de la técnica en la Figura (2.2) creada por Orellana (2020).

Utilizando SMOTE aumentaríamos el número de datos de la clase fraudulenta hasta los 237276, consiguiendo nuevamente un dataset balanceado.

2.4.3. Ct-GAN

Las GAN (Generative Adversarial Networks) es un algoritmo de inteligencia artificial en el que dos redes neuronales compiten mutuamente el cual diseñado principalmente por Ian Goodfellow. Podemos encontrar el desarrollo total del método en Goodfellow et al. (2014). Está técnica es usualmente utilizada para la generación de fotografías que parecen auténticas a observadores humanos, pudiendo apreciar un ejemplo de esto en la Figura (2.3).

En este algoritmo una red genera los datos y la otra los evalúa. Generalmente, la red generativa aprende a formar elementos con una distribución de datos determinada, mientras que la red discriminativa diferencia entre los elementos que queremos replicar y los elementos formados por la red

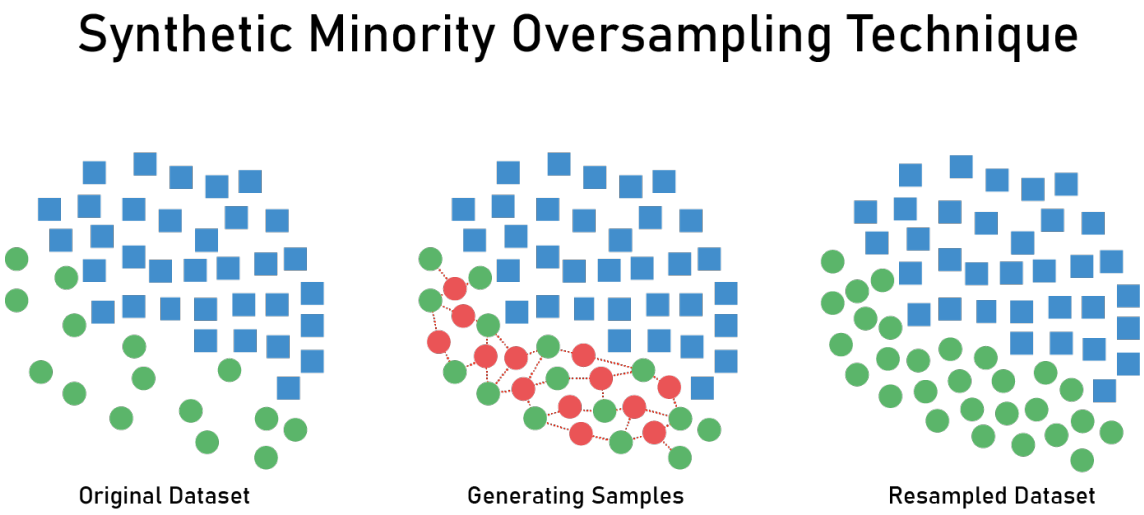

Figura 2.2: Funcionamiento de la técnica de remuestreo SMOTE.

generativa. El objetivo de la red generativa es engañar a la red discriminativa, produciendo elementos sintéticos que se parecen mucho a los auténticos.

En el contexto del problema de prevención de fraude, podemos utilizar dicho algoritmo para generar datos de la clase fraudulenta, en el que la red generativa empezaría formando ruido aleatorio y poco a poco los datos sintéticos irían pareciéndose más a los datos fraudulentos originales. Con ello, conseguiríamos nuevamente que el conjunto de datos estuviese balanceado.

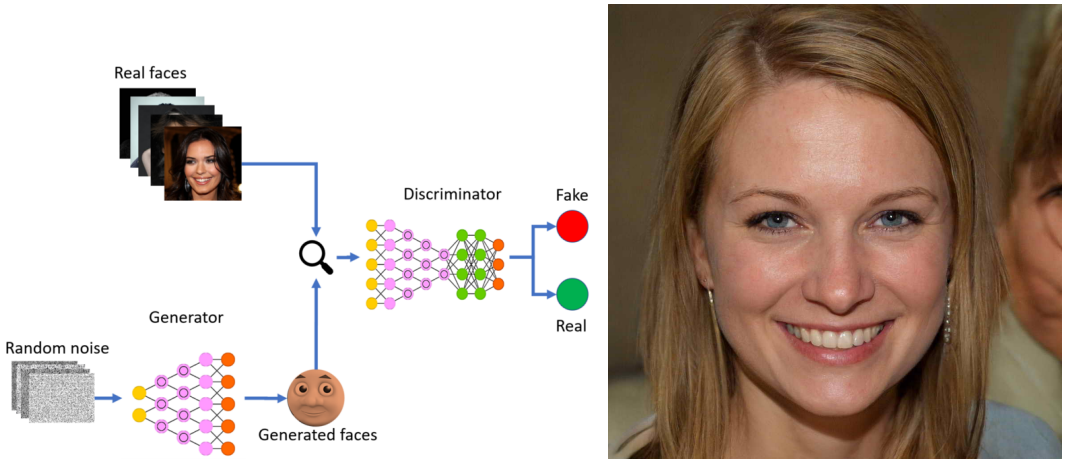

Figura 2.3: Izquierda: Esquema del funcionamiento de una GAN. En ella, a partir de ruido aleatorio es posible generar caras sintéticas. Creado por Kulpati (2019). Derecha: Imagen de una cara generada utilizando una GAN.

2.5. Validación Cruzada

La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y prueba.

En este trabajo utilizaremos la validación cruzada de K iteraciones, en la cual los datos de muestra se dividen en K subconjuntos. De esta forma, utilizaremos como datos de validación uno de los subconjuntos y el resto de los $(K-1)$ como datos de entrenamiento. Este proceso se repite k iteraciones con cada uno de los posibles subconjuntos de prueba, calculando en cada caso el F_1 *score* definido en la Sección 2.7. Tras obtener k valores de el F_1 *score*, promediaremos los resultados, obteniendo de esta forma una métrica global del modelo en el conjunto de datos original. Podemos ver una representación gráfica de este método en la Figura (2.4) creada por Joan.domenech91 (2011).

Este proceso lo realizaremos para cada uno de los posibles valores de los hiperparámetros que queramos ajustar, de manera que se acaban tomando los hiperparámetros que optimicen el F_1 *score*.

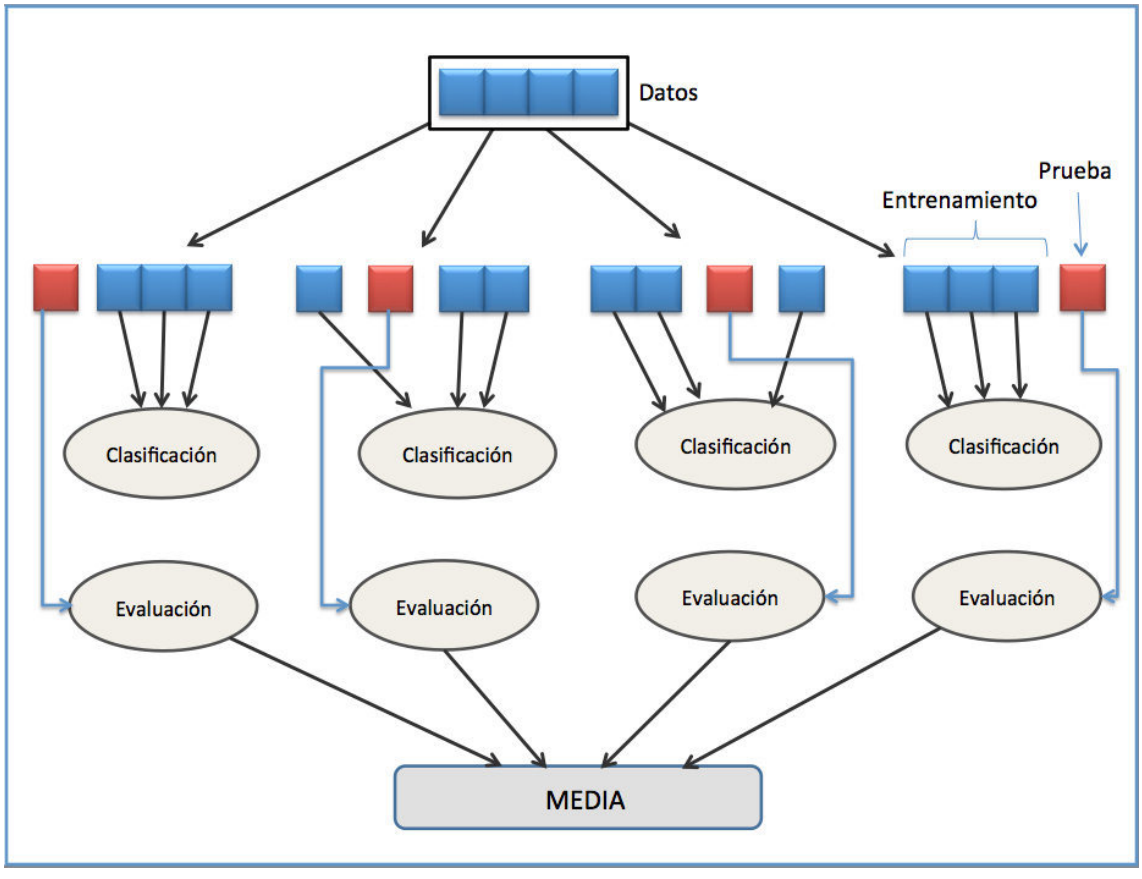

Figura 2.4: Esquema de funcionamiento del método de validación cruzada.

2.6. Problema de clasificación y modelos de predicción

Empezaremos recordando que se entiende por problema de clasificación. En algunos problemas, existe relación entre dos o más variables, una variable de interés, sobre la cual nos interesa predecir y a la que nos referiremos como variable respuesta Y y un conjunto de variables con las que intentaremos explicar dicha variable respuesta, las cuales conocemos como variables explicativas $(X_1, \ldots, X_{p-1})$. Esta relación, la podríamos escribir matemáticamente de la siguiente forma:

$$Y = f(X) + \varepsilon, \tag{2.8}$$

donde en (1.1) f es una función de X desconocida y ε es el término del error, el cual es independiente de X y tiene media cero. En nuestro problema, la variable respuesta Y cae dentro de una de las dos siguientes categorías:

$$Y = \begin{cases} \text{No hay fraude} \\ \text{Si hay fraude} \end{cases} \tag{2.9}$$

esto lo podríamos modelizar de la siguiente forma:

$$Y = \begin{cases} 0, & \text{si la transacción no es fraudulenta} \\ 1, & \text{si la transacción es fraudulenta} \end{cases} \tag{2.10}$$

Por lo que estaríamos frente a un problema de predecir una respuesta cualitativa, lo cual se conoce como un problema de clasificación. Para resolver esto, podíamos utilizar múltiples técnicas de clasificación, como pueden ser la regresión logística, los árboles de decisión, bosques aleatorios...

Sin embargo, debido a la naturaleza del problema, tenemos una peculiaridad en el conjunto de datos, y es que el número de transacciones fraudulentas es mucho más pequeño que el número de transacciones en las que no hay fraude, es decir, el conjunto de datos esta desbalanceado. Por ello, ciertos modelos podrían no funcionar adecuadamente, por lo que es importante tenerlo en cuenta a la hora de realizar un procedimiento.

2.6.1. Regresión logística

El modelo de regresión logística es un modelo estadístico el cual modela la probabilidad de que una observación pertenezca a una clase a través de funciones lineales en las variables explicativas X. De esta forma, en el problema considerado, el objetivo será clasificar a partir de las variables explicativas X si una determinada observación pertenece a la clase de Fraude o no Fraude.

Más formalmente, dado $(X_1, \ldots, X_{p-1})$ variables explicativas, una primera aproximación seria plantearnos expresar la probabilidad de que $Y = 1$ a través de funciones lineales:

$$p(X) = Pr(Y = 1|X_1, \ldots, X_{p-1}) = \beta_0 + \beta_1 X_1 + \cdots + \beta_{p-1} X_{p-1} \tag{2.11}$$

el problema de la ecuación (2.11) es que las probabilidades $P(X)$ podrían ser negativas o mayores que 1. Para solucionar esto utilizaremos la función logística $g(t) = \frac{e^t}{1+e^t}$, la cual toma valores en el intervalo $[0, 1]$. De esta forma, llegaríamos a la expresión:

$$P(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_{p-1} X_{p-1}}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_{p-1} X_{p-1}}} \tag{2.12}$$

donde $\beta_0, \ldots, \beta_{p-1}$ son los parámetros a ajustar del modelo, los cuales serán ajustados utilizando máxima verosimilitud.

Ejemplo 2.1 *El conjunto de datos considerado procede de una submuestra de datos de las operaciones bancarias junto con las variables explicativas IMPORTE, DISTANCIA_NAVEGADOR y NUMERO_LOGINS, además de la variable respuesta ETIQUETA. Tras entrenar un modelo de regresión logística con los datos de entrenamiento, se han obtenido los siguientes parámetros:*

$$\beta_0 = 7{,}4; \quad \beta_1 = -5{,}2 \times 10^{-6}; \quad \beta_2 = -0{,}12; \quad \beta_3 = -0{,}0022.$$

Dada una nueva observación $X = (135{,}95,\ 60,\ 1)$*, podemos calcular la probabilidad de que esta pertenezca a la clase fraudulenta o no fraudulenta a partir de la ecuación (2.12). De esta forma:*

$$P(X) = \frac{e^{7{,}4-5{,}2\times10^{-6}*135{,}95-0{,}12*60-0{,}0022*1}}{1+e^{7{,}4-5{,}2\times10^{-6}*135{,}95-0{,}12*60-0{,}0022*1}} = 0{,}69$$

Como $P(X) > 0{,}5$*, entonces predecimos que* $Y = 1$*, o lo que es lo mismo, la nueva observación pertenece a la clase fraudulenta.*

2.6.2. Árboles de Clasificación

Los árboles de clasificación fueron propuestos por Leo Breiman en el libro Breiman et al. (1984) y son árboles de decisión que tienen como objetivo predecir la variable respuesta Y en función de covariables. En ellos, se establece una serie de reglas de decisión en función de valores límite para una de las variables y se va dividiendo la muestra hasta obtener una probabilidad de pertenencia a un grupo. Dada una observación $(x_1, \ldots, x_{p-1})$, predeciremos que dicha observación pertenece a la clase en la que ha ocurrido más usualmente en el conjunto de entrenamiento.

Ejemplo 2.2 *El conjunto de datos Iris de Fisher contiene* 50 *muestras de cada una de las* 3 *especies de Iris (Iris setosa, Iris virginica e Iris versicolor). Se midió cuatro rasgos de cada muestra: el largo y ancho del sépalo y pétalo, en centímetros. Basado en la combinación de estos cuatro rasgos y como podemos ver en la Figura (2.5), cada una de estas categorías tiene una región en función de las variables. Dada una nueva observación con una longitud de pétalo igual a 4 y una anchura de pétalo igual a 1, la clasificaríamos como versicolor, ya que pertenece a una clase dominada por flores de la especie versicolor.*

Lamentablemente, a pesar de que los árboles de clasificación son fácilmente explicables y tienen buenas propiedades como la robustez a outliers o la no necesidad de transformar los datos, este tipo de modelos no suele dar buenos resultados en términos de accuracy debido a que tienen una alta varianza. Para solucionar esto utilizaremos técnicas de ensamblaje. Estas técnicas están basadas en la utilización de una colección de modelos para predecir en lugar de solamente uno, intentando mantener las buenas propiedades de los modelos basados en árboles pero mejorando sus resultados. Los modelos de ensamblaje usan dos métodos:

- **Bagging**: Crea diferentes subconjuntos de entrenamiento a partir de la muestra de datos con reemplazamiento y el resultado final se consigue a través del voto de la mayoría de los árboles. Un ejemplo de esto sería Random Forest.

- **Boosting**: Combina predictores débiles convirtiéndolos en predictores fuertes, esto lo lograremos creando un modelo secuencial de manera que el modelo final tenga un mejor accuracy. En esta categoría entrarían el ADABoost o el XGBoost.

Podemos ver una representación gráfica de la idea del algoritmo del boosting y bagging en la Figura (2.6).

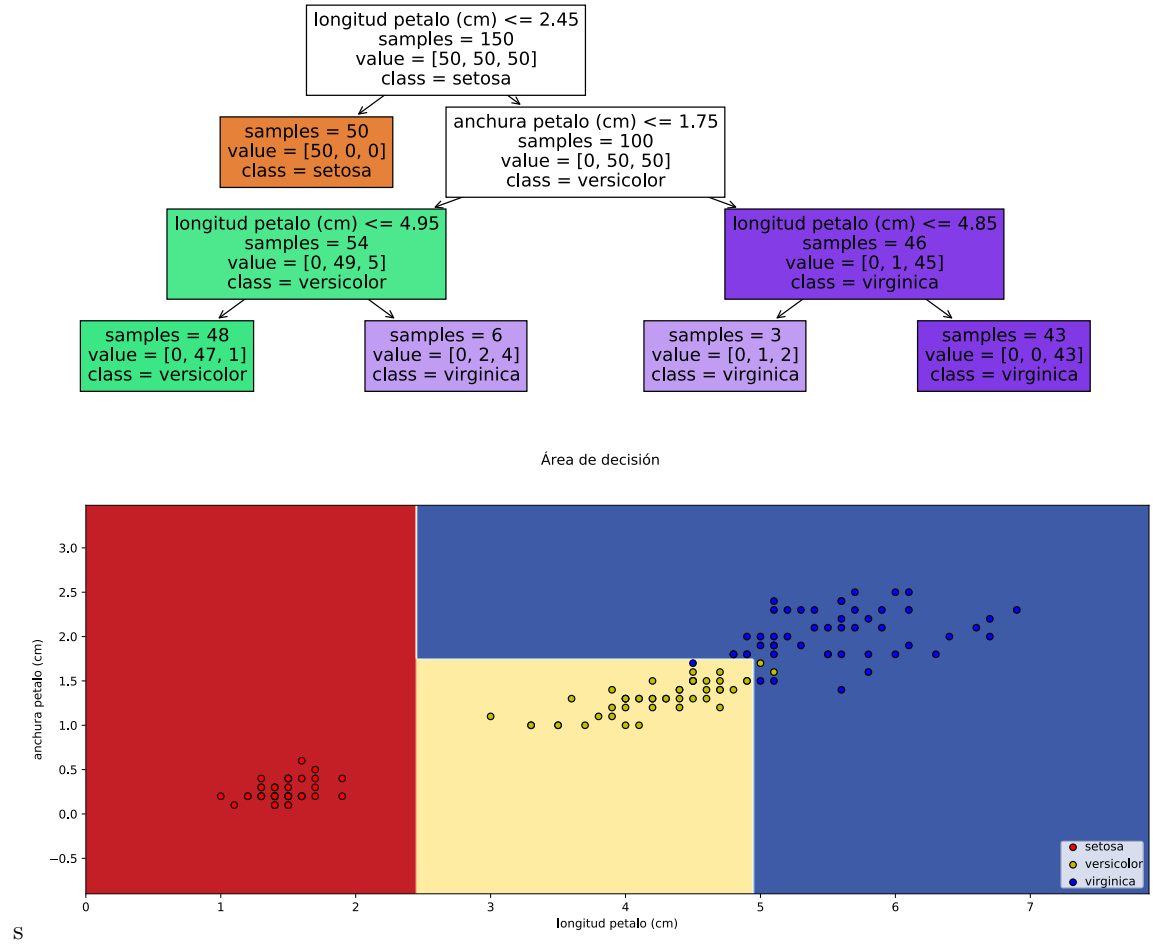

Figura 2.5: Gráfico superior: diagrama un árbol de decisión en función de las diferentes características de una flor. Gráfico inferior: diagrama de dispersión de la longitud del pétalo frente a la anchura del pétalo. Las diferentes especies se representan mediante colores, rojo (setosa), amarillo (versicolor) y azul (virginica familiar).

2.6.3. Random Forest

El algoritmo de Random Forest fue creado en 1995 por Tin Kam Ho en el artículo Ho (1995) y posteriormente desarrollada una extensión del mismo por Leo Breiman y Adele Cutler. El funcionamiento de esta algoritmo esta basado en la combinación de diferentes árboles predictores no correlacionados para luego promediarlos. Random Forest es uno de los algoritmos de clasificación más populares y utilizados debido a que en muchos casos obtiene rendimientos similares a los del Boosting y es más fácil de utilizar y entrenar.

La idea tras este procedimiento es la siguiente: dado un conjunto de observaciones independientes $X_1, \ldots, X_n$ cada una con varianza σ^2, la varianza de la media $\bar{X}$ viene dada por σ^2/n, es decir, promediar las observaciones reduce la varianza.

En el Random Forest, construiremos un conjunto de arboles de decisión sobre el conjunto de entrenamiento como podemos ver en el Algoritmo 1. Además, podemos ver una representación gráfica

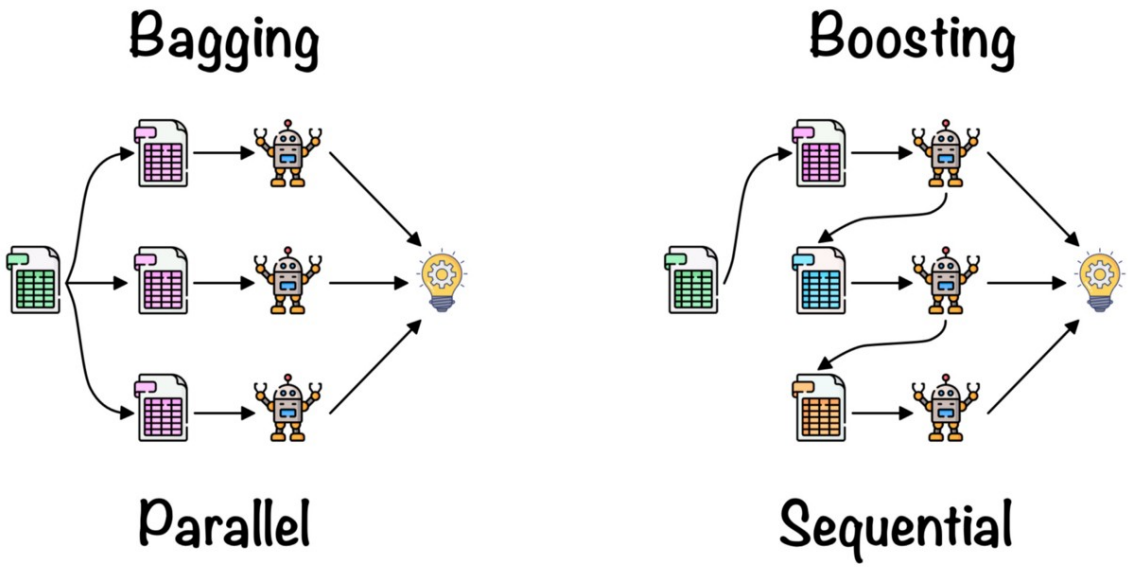

Figura 2.6: Izquierda: Imagen que representa el procedimiento del Random Forest. Partimos de una muestra inicial que será remuestreada con Bootstrap, posteriormente aplicaremos un árbol de decisión en cada una de las muestras y promediaremos los resultados. Derecha: Imagen que representa el procedimiento del Boosting. Partimos de una muestra inicial e iremos aplicándole árboles de decisión, de manera que aquellos datos mal clasificados ganan un mayor peso y son corregidos en iteraciones posteriores. Finalmente los árboles son promediados.

de como funciona Random Forest en la Figura (2.6).

2.6.4. Métodos de descenso por gradiente

Boosting

El Boosting es un algoritmo de aprendizaje automático que reduce el sesgo y varianza en un contexto de aprendizaje supervisado. Este, está basado en el cuestionamiento planteado por Kearns y Valiant en Kearns (1988): ¿Puede un conjunto de clasificadores débiles (aquel que predice ligeramente mejor que una elección aleatoria) crear un clasificador robusto?

Pues justamente de eso trata el Boosting, consiste en combinar los resultados de varios clasificadores débiles para obtener un clasificador robusto. A diferencia del Random Forest que se promediaban los resultados de los diferentes modelos, en este caso, combinaremos de manera secuencial los diferentes clasificadores débiles (en este trabajo consideraremos árboles de decisión).

Gradient Boosting

Gradient Boosting es un algoritmo de boosting para la regresión el cual fue desarrollado por Jerome H. Friedman en Friedman (2000). Dado un conjunto de datos $D = ((x_1, y_1), \ldots, (x_N, y_N))$, el objetivo del gradient boosting es encontrar una aproximación $\hat{F}(x)$ de la función $F^*(x)$, la cual lleva los valores de x a sus valores de salida y, minimizando la función de pérdida dada $L(y, F(x))$. Esta aproximación

Algorithm 1 Random Forest para Clasificación

Input: x
Output: $\hat{y}$

1. Desde $b = 1$ a B:
   a) Tomar una muestra bootstrap Z^* de tamaño N desde los datos de entrenamiento.
   b) Crear un árbol del Random Forest T_b desde los datos remuestreados con bootstrap, repitiendo recursivamente los siguientes pasos para cada nodo terminal del árbol, hasta que el mínimo tamaño de nodo, n_{min}, es alcanzado.
      1) Seleccionar m variables aleatoriamente de las p variables totales.
      2) Tomar la variable que mejor divide los datos entre las m.
      3) Dividir el nodo en dos nodos hijos.
2. Devolver el árbol de decisión ensamblado $\{T_b\}_1^B$

   Para predecir una nueva observación:
   Sea $\hat{C}_b(x)$ la predicción de clase para el b-ésimo árbol del Random Forest. Luego, $\hat{C}_{rf}^B(x) =$ *voto de la mayoria*$\{\hat{C}_b(x)\}_1^B$.

será construida como:

$$F_m(x) = F_{m-1}(x) + \rho_m h_m(x) \tag{2.13}$$

donde $h_m(x)$ son las funciones de ensamblaje y ρ_m es el peso asociado a la m-ésima función $h_m(x)$. Por otro lado, construiremos la función iterativamente, obteniendo una primera aproximación de $F^*(x)$ como:

$$F_0(x) = \arg\min_{\alpha} \sum_{i=1}^{N} L(y_i, \alpha). \tag{2.14}$$

De esta forma, los siguientes modelos minimizarán:

$$(\rho_m, h_m(x)) = \arg\min_{\rho,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i)). \tag{2.15}$$

Podemos ver cada h_m como un paso en la función de descenso por gradiente de F^*, teniendo que entrenar cada modelo h_m en un nuevo conjunto de datos $D = ((x_1, r_m 1), \ldots, (x_N, r_{mN}))$, donde los pseudo-residuos se calcularán como:

$$r_{mi} = \left[\frac{\partial L(y_i, F(x))}{\partial F(x)}\right]_{F(x)=F_{m-1(x)}} \tag{2.16}$$

Posteriormente calcularemos el valor de ρ_m solucionando un problema de optimización. Podemos ver el Algoritmo completo en 2, presente en el libro Breiman et al. (1984).

XGBoost

XGBoost es un modelo ensamblado a partir de árboles de decisión basado en el Gradient Boosting y diseñado para ser altamente escalable. De igual forma que Gradient Boosting, XGBoost construye una expansión aditiva de la función objetivo minimizando la función de pérdida. Como el XGBoost

Algorithm 2 Algoritmo Gradient Boosting para árboles

Input: x, y
Output: $\hat{F}(x)$

1. Iniciamos $F_0(x) = \arg\min_\lambda \sum_{i=1}^{N} L(y_i, \alpha)$.
2. Desde $m = 1$ a M:
   a) Para $i = 1, 2, \ldots, N$ calcular
   $$r_{mi} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F=F_{m-1}}$$
   b) Ajustar un árbol para los objetivos r_{mi} dadas las regiones terminales $R_{mj},\ j = 1, 2, \ldots, J_m$
   c) Para $j = 1, 2, \ldots, J_m$ calcular:
   $$(\rho_m, h_m(x)) = \arg\min_{\rho,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i)).$$
   d) Actualizar
   $$F_m(x) = F_{m-1}(x) + \rho_m h_m(x)$$
3. Devolver $\hat{F}(x) = F_M(x)$

esta unicamente centrado en los árboles como clasificadores base, se usa una variación de la función de pérdida para controlar la complejidad de los mismos:

$$L_{xgb} = \sum_{i=1}^{N} L(y_i, F(x_i)) + \sum_{m=1}^{M} \beta(h_m) \tag{2.17}$$

siendo:

$$\beta(h_m) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{2.18}$$

donde T es el numero de hojas del árbol y w son las puntuaciones de las hojas. De esta forma, el valor γ controla la mínima reducción de la pérdida requerida para dividir un nodo interno, de manera que valores altos de γ resultan en árboles más sencillos. Podemos ver una explicación más detallada de esto en Bentejac et al. (2020).

2.6.5. Redes Neuronales

Una red neuronal es un modelo de regresión o clasificación el cual esta formado por un número determinado de elementos llamados neuronas, las cuales a su vez están agrupadas en unas estructuras denominadas capas, de manera que cada neurona de cada capa se trata de una combinación de la capa anterior. Comenzaremos entonces definiendo lo que es una neurona para, posteriormente, hablar de las capas y explicar en detalle las redes neuronales.

La neurona

En 1943 el neuroanatomista Warren McCulloch y el matemático Walter Pitts propusieron un modelo de neurona artificial creado con el fin de llevar a cabo tareas simples en el artículo Mcculloch and Pitts (1943). Los resultados de este modelo sirvieron como base para las futuras redes neuronales. Este modelo aplicaba la función:

$$y = f\left(\sum_j w_{ij}x_j + b\right)$$

donde:

- x_j son los datos de entrada.
- w_{ij} son los pesos relativos de las entradas.
- b es un término aditivo o sesgo.
- f es una función de activación.

Podemos observar una representación gráfica de esto en la Figura (2.7).

Capas de una red neuronal

Una red neuronal esta formada por agrupaciones de capas, las cuales a su vez están formadas por neuronas como la vista en la Sección 2.6.5. Existen tres tipos de capas según su posición:

- **Capa de entrada o Input Layer**: Capa de entrada de los datos, los cuales serán las observaciones $(x_1, \ldots, x_n)$.
- **Capas ocultas o Hidden Layers**: Capas ocultas que forman la parte central de una red neuronal. Su interpretación es más compleja y a menudo suelen funcionar como una caja negra. Están formadas por las combinaciones lineales de anteriores capas.
- **Capa de salida o Output Layer**: Capa de salida, donde obtendremos las predicciones de que una determinada observación pertenezca a una clase.

Podemos representar una capa en un diagrama como el de la Figura (2.8). De esta forma, sea:

- $X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{pn} \end{pmatrix}$ el conjunto de datos de entrenamiento, donde n es el número de observaciones y p el número de variables.
- $W = \begin{pmatrix} w_{11} & \cdots & w_{1n^{[l-1]}} \\ \vdots & \ddots & \vdots \\ w_{n^{[l]}1} & \cdots & w_{n^{[l]}n^{[l-1]}} \end{pmatrix}$ la matriz de pesos relativos, donde $n^{[l-1]}$ es el número de neuronas en la anterior capa y $n^{[l]}$ es el número de neuronas en la siguiente.
- $b = \begin{pmatrix} b_1 \\ \vdots \\ b_{n_x} \end{pmatrix}$ el sesgo.

Podemos definir las salidas de la siguiente capa:

$$Z = \begin{pmatrix} z^{(1)} & \vdots & z^{(1)} \\ \vdots & \ddots & \vdots \\ z^{(n^{[l]})} & \vdots & z^{(n^{[l]})} \end{pmatrix} = \begin{pmatrix} w_{11} & \cdots & w_{1n^{[l-1]}} \\ \vdots & \ddots & \vdots \\ w_{n^{[l]}1} & \cdots & w_{n^{[l]}n^{[l-1]}} \end{pmatrix} \cdot \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{p1} & \cdots & x_{p,n} \end{pmatrix} + \begin{pmatrix} b_1 \\ \vdots \\ b_{n^{[l]}} \end{pmatrix}$$

y

$$A = \begin{pmatrix} a^{(1)} & \vdots & a^{(1)} \\ \vdots & \ddots & \vdots \\ a^{(n^{[l]})} & \vdots & a^{(n^{[l]})} \end{pmatrix} = \begin{pmatrix} f(z^{(1)}) & \vdots & f(z^{(1)}) \\ \vdots & \ddots & \vdots \\ f(z^{(n^{[l]})}) & \vdots & f(z^{(n^{[l]})}) \end{pmatrix}$$

donde f es una función de activación, siendo las mas utilizadas la función ReLU Brownlee (2019), la tangente hiperbólica y la función logística.

Redes Neuronales

Hasta ahora, hemos visto la estructura de una neurona y de una capa. El poder de este tipo de modelos surge cuando se añaden múltiples capas a la red. De ahora en adelante usaremos el superscript $[l]$ para referirnos a la l-ésima capa de una red neuronal y el subscript i para referirnos a la i-ésima neurona de una capa.

Con esto, podemos calcular todas las activaciones de una capa l usando la matriz de pesos $W[l]$.

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]} \qquad A^{[l]} = g(Z^{[l]}) \tag{2.19}$$

En una red neuronal, tomaremos la salida de la capa anterior para alimentar la entrada de la siguiente capa. De esta forma, podremos apilar tantas capas como queramos, consiguiendo estructuras muy diversas como la que se muestra en la Figura (2.9).

Para entrenar este tipo de modelos, es necesario una función de pérdidas. Dada una predicción $\hat{y}$ y una observación y, definiremos la función de pérdida como:

$$p(\hat{y}, y) = -(y \ log(\hat{y}) + (1 - y) \ log(1 - \hat{y})) \tag{2.20}$$

esta función representa la pérdida respecto a un único ejemplo de entrenamiento. Para ver la pérdida global debemos de definir la función de costes:

$$J(W, b) = \frac{1}{n} \sum_{i=1}^{n} p(\hat{y}^{(i)}, y^{(i)}) \tag{2.21}$$

donde n será el número de observaciones del conjunto de datos y $p(\hat{y}, y)$ ha sido definida en la ecuación (2.20). El objetivo del problema será minimizar esta función, la cual depende de los pesos W y el sesgo b. Para obtener unos buenos valores de estos hiperparámetros, la red llevará a cabo dos pasos:

- **Forward propagation**: Dada una observación x, la propagación hacia delante son los pasos necesarios para obtener una predicción $\hat{y}$. Estos pasos se corresponden con los llevados a cabo en las ecuaciones presentes en (2.19).

- **Backward propagation**: Una vez que hemos propagado la red hacia delante, obtenemos una predicción $\hat{y}$ con la que podemos comparar la observación y. Con la función de costes definida en la ecuación (2.21), podemos saber si el modelo es bueno o no. En caso de que el modelo no sea el adecuado, debemos de recalcular los pesos para obtener unos mejores resultados en términos de la función de costes. El proceso que llevamos a cabo para recalcular los pesos es el que conocemos como backward propagation.

Podemos encontrar una explicación más detallada de las redes neuronales y los pasos de Forward y Backward propagation en el Libro Goodfellow et al. (2016).

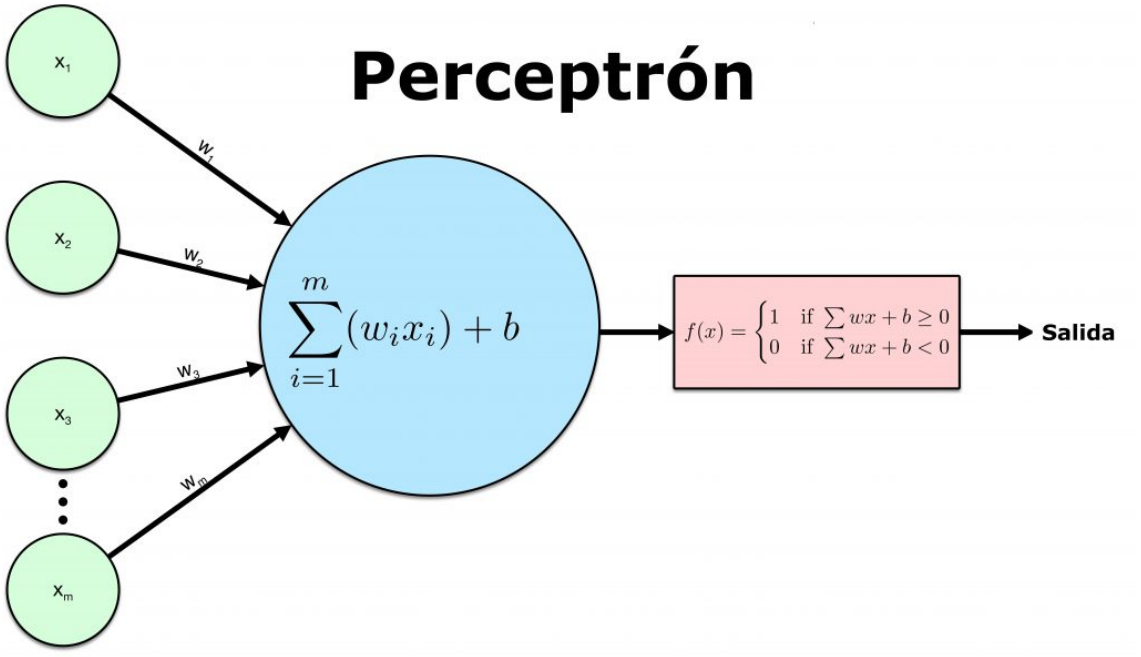

Figura 2.7: Representación gráfica del funcionamiento de una neurona. Imagen sacada del artículo Alvarez (2018).

2.7. Criterios de selección de modelos

Los criterios de selección de modelos son aquellos por los que nos guiaremos para comprobar si un modelo se impone a otro. Estos criterios son la *precision*, la *recall*, el F_1 *score* y el tiempo de una nueva observación.

2.7.1. Precision, Recall y F1-Score

Para definir estas métricas necesitamos definir en un primer momento las matrices de confusión. Una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Considerando el problema de prevención de fraude y dado que trabajamos bajo el marco de un problema de aprendizaje supervisado, conocemos si una operación es fraudulenta o no fraudulenta. Por otro lado, dado que nos interesa conocer el rendimiento de un algoritmo, conoceremos las diferentes

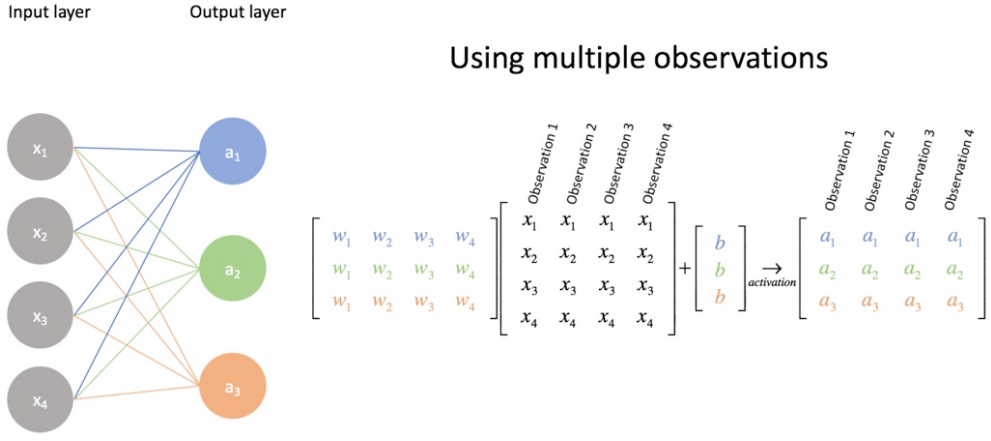

Figura 2.8: Representación gráfica de la interacción de los distintos elementos de una capa en una red neuronal. Imagen sacada del artículo Jordan (2017).

predicciones que haga dicho algoritmo. De esta forma, podemos representar los datos como se ve en el Cuadro (2.2).

Observado / Predicción	No Fraude	Fraude
No Fraude	Verdaderos Negativos (TN)	Falsos Positivos (FP)
Fraude	Falsos Negativos (FN)	Verdaderos Positivos (TP)

Cuadro 2.2: División de los posibles escenarios en un problema de clasificación.

Usualmente, para los problemas de clasificación, la medida de referencia es el accuracy:

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

siendo este una medida de la proporción de valores bien clasificados por nuestro algoritmo. Sin embargo, al llevar esta medida a un problema con datos desbalanceados, podemos llegar a resultados engañosos como se muestran en el Ejemplo (2.3).

Ejemplo 2.3 *En el conjunto de datos del problema de prevención de fraude tenemos un total de* 644 *observaciones fraudulentas y* 237276 *no fraudulentas. Supongamos un modelo el cual clasificase todos los datos como no fraudulentos. Entonces, obtendríamos la siguiente matriz de confusión:*

Observado / Predicción	No Fraude	Fraude
No Fraude	237276	0
Fraude	644	0

Este modelo obtendría un accuracy igual a 0,9973*, pero sin embargo no clasificaría correctamente ningún dato de la clase fraudulenta.*

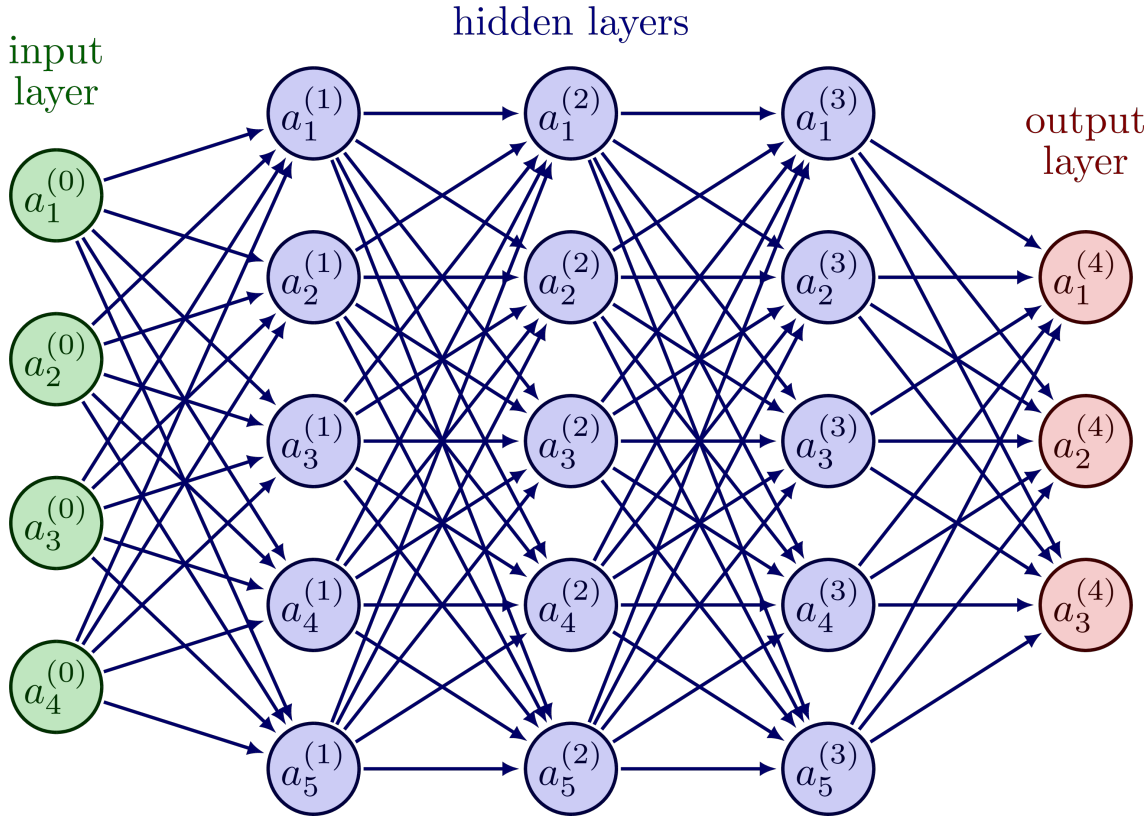

Figura 2.9: Representación gráfica de un modelo de redes neuronales. Imagen obtenida en Neutelings (2021).

Por lo tanto, debido al bajo número de datos que conocemos de la clase fraudulenta, es necesario utilizar métricas que cuantifiquen el rendimiento en esta clase. Así, utilizaremos:

- **Precision**: Cuantifica el número total de predicciones positivas correctas realizada.

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

  donde TP se refiere al número de verdaderos positivos y FP al número de falsos positivos.

- **Recall**: Cuantifica el número total de predicciones positivas correctas realizada sobre el total de predicciones positivas que se podrían haber realizado.

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

  donde TP se refiere al número de verdaderos positivos y FN al número de falsos negativos.

- F_1 **score**: Combina la precision y la recall en una sola medida.

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Por otro lado, también será importante el **tiempo de predicción de una nueva observación**. Esta métrica es importante a la hora de evaluar un modelo pues es necesario un tiempo de respuesta rápido (menor que 1 segundo), por lo que ciertos modelos podrían ser invalidados por tardar mucho en predecir.

Capítulo 3

Análisis exploratorio de los datos

En este capítulo realizaremos el análisis exploratorio de los datos, realizando los pasos necesarios para poder obtener un conjunto de datos con el que poder trabajar.

Para llevar esto a cabo, en la Sección 3.1 realizaremos una descripción de los datos considerados en el problema. En la Sección 3.2 llevaremos a cabo la limpieza de los datos faltantes del problema. La Sección 3.3 la dedicaremos a la creación de nuevas variables a partir de las variables que ya conocemos. En la Sección 3.4 realizaremos representaciones gráficas sobre distintas variables de interés en las clases de datos fraudulentos y no fraudulentos. Más tarde en las Secciones 3.5 y 3.6 realizaremos un estudio de correlaciones y el tratamiento de outliers, ambos importantes para el funcionamiento de los modelos de machine learning. Tras esto, en la Sección 3.7 realizaremos diferentes test de hipótesis sobre las clases del problema, contrastando sus funciones de densidad para comprobar si las variables son significativas. Finalmente, en la Sección 3.8 realizaremos el remuestreo de los datos del problema.

3.1. Conjunto de datos del Problema

El conjunto de datos considerado para el problema de detección de fraude procede de una submuestra realizada sobre el total de las transacciones correspondientes a las fechas comprendidas entre el 01/02/2021 y el 20/05/2022. El conjunto inicial de datos cuenta con un total de $p = 90$ variables y $n = 238020$ observaciones, aunque tras el procesamiento de los datos estos valores serán diferentes.

3.1.1. Descripción de variables

Podemos encontrar una descripción completa de las variables en el Apéndice A.

Por otro lado, distinguiremos entre dos clases de observaciones, aquellas en las que la variable ETIQUETA toma el valor Y, lo cual indicará que la transacción es fraudulenta y aquellas en las que toma el valor N, lo cual indica que no.

Dentro de la clase fraudulenta, contaremos con un total de 644 observaciones (aproximadamente un 0.27 % del total) mientras que las demás 237276 observaciones pertenecerán a la clase no fraudulenta. Esto hará que el conjunto de datos sea extremadamente desbalanceado.

3.2. Limpieza de los datos

La limpieza de los datos es un proceso que se lleva a cabo para asegurar la calidad de los mismos. Este paso es fundamental, pues en caso de no realizarlo tendríamos que utilizar los algoritmos de machine learning con datos erróneos o incompletos, lo que podría llevarnos a predecir erróneamente algunas observaciones.

Conociendo esto, enfocaremos la limpieza de los datos en los siguientes puntos:

- Datos faltantes.
- Datos duplicados.
- Selección de aquellas variables más relevantes.

3.2.1. Datos Faltantes

Realizando una búsqueda de las observaciones con datos faltantes, ya podemos apreciar que es un problema que ocurre en varias variables. Podemos observar esto en la Figura (3.1). Estas variables se corresponden con ID_CTA_DESTINO, IMPORTE, PAIS_ORIGEN, RECOMMENDATION, IN_EALE, IN_BMOVIL, IN_BE, NAVEGADOR_LIST y COUNTRY_LIST.

Para solucionar el problema de los datos faltantes, procederemos de diferente manera en función de si la variable es categórica o continua:

- **Continua**: La única variable continua que sufre de datos faltantes es IMPORTE. Debido a los altos valores de los importes de ciertas transacciones, se ha considerado que el valor conveniente para sustituir los datos faltantes es la **mediana** debido a su robustez.
- **Categórica**: En este tipo de variables añadiremos la categoría **Desconocido**, de esta forma, cuando desconozcamos un valor simplemente lo sustituiremos por esta nueva categoría.

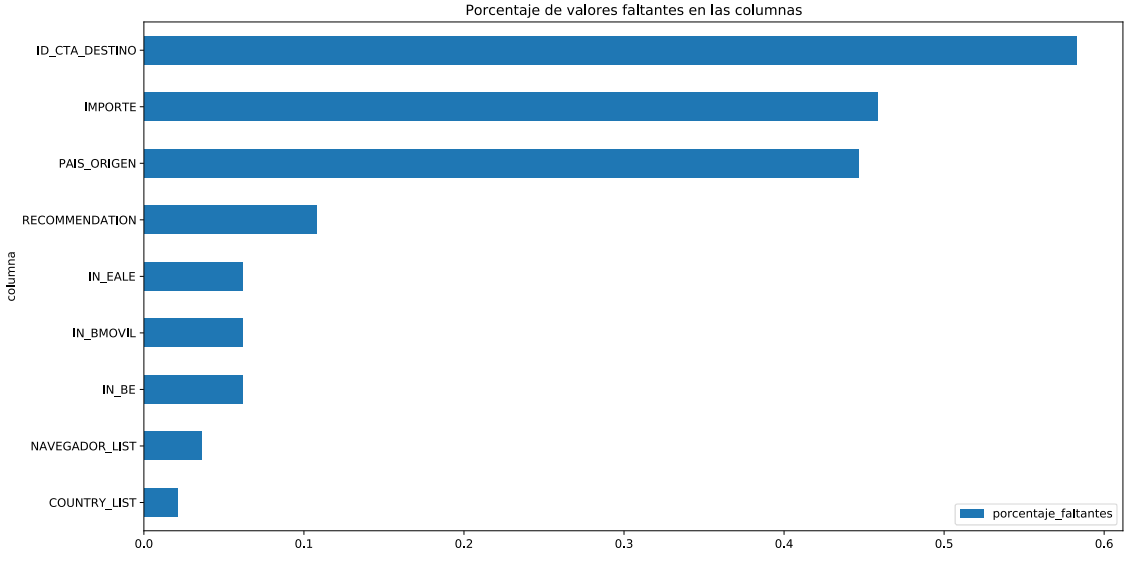

Figura 3.1: Representación gráfica del porcentaje de datos faltantes en las variables del problema.

3.2.2. Datos Duplicados

El tratamiento de este tipo de datos también es necesario a la hora de realizar este proceso. Sin embargo, el conjunto de datos de prevención de fraude no padece de este problema, por lo que no tendremos que eliminar ninguna observación.

3.2.3. Eliminación de variables con información poco relevante

En este apartado nos encargaremos de eliminar aquellas variables del Apéndice A las cuales no aporten ninguna información relevante para la predicción de fraude o aporten información repetida. En esta etapa, eliminaremos las siguientes variables:

- RECOMMENDATION: La variable se obtiene a partir de otro modelo de predicción, no la tenemos en el momento en el que llega información de una operación.
- PAIS_ORIGEN: Reformularemos esta variable junto con COUNTRY_LIST, de manera que sea más informativa y no simplemente las siglas del país de procedencia de la transacción.
- ACCESSES_BMOVIL_W1_ZS: Variable con información repetida, ya conocemos ACCESSES_BMOVIL_W1.
- ACCESSES_BMOVIL_M1_ZS: Variable con información repetida, ya conocemos ACCESSES_BMOVIL_M1.
- ACCESSES_BMOVIL_3M_ZS: Variable con información repetida, ya conocemos ACCESSES_BMOVIL_3M.
- ACCESSES_BMOVIL_ZS: Variable con información repetida, ya conocemos ACCESSES_BMOVIL.
- ACCESSES_BE_W1_ZS: Variable con información repetida, ya conocemos ACCESSES_BE_W1.
- ACCESSES_BE_M1_ZS: Variable con información repetida, ya conocemos ACCESSES_BE_M1.
- ACCESSES_BE_3M_ZS: Variable con información repetida, ya conocemos ACCESSES_BE_3M.
- ACCESSES_BE_ZS: Variable con información repetida, ya conocemos ACCESSES_BE.

3.3. Creación de variables

Este apartado es fundamental a la hora de obtener buenos resultados cuando nos enfrentamos a un problema de clasificación o regresión. Se trata de, a partir de las variables conocidas, obtener nuevas variables las cuales puedan aportar información adicional para futuros modelos de clasificación.

3.3.1. Variable Navegador

Dadas las variables NAVEGADOR y NAVEGADOR_LIST descritas en el Apéndice A, crearemos dos nuevas variables que nos indiquen si el navegador desde el que se realizo la transacción está dentro de la lista de navegadores usuales.

Para ello, dado un navegador y la lista de navegadores usuales, compararemos ambas strings utilizando la función token_set_ratio del paquete fuzzywuzzy Cohen (2020), la cual nos devolverá un valor entre 0 y 100 del parecido de ambos strings. Podemos ver el funcionamiento de esto en el Ejemplo (3.1).

Ejemplo 3.1 *Sea X = NAVEGADOR e Y= NAVEGADOR_LIST y dada una observación (x_1, y_1) de manera que:*

$$x_1 = \text{'Dalvik/2.1.0 (Linux; U; Android 10; Redmi Note 7 MIUI/V12.0.2.0.QFGEUXM'}$$

y_1 = *'Dalvik/2.1.0 (Linux; U; Android 10; Redmi Note 7 M$Mozilla/5.0(Macintosh; Intel MAC...*

la función token_set_ratio nos devolverá un valor de 83*, indicándonos el parecido entre las cadenas de texto.*

Utilizando esta función crearemos dos nuevas variables:

- **DISTANCIA_NAVEGADOR**: Variable la cual nos indica el valor de la función token_set_ratio al aplicarla sobre las variables NAVEGADOR y NAVEGADOR_LIST.
- **CLASE_NAVEGADOR**: Variable categórica la cual tomará diferentes clases en función del valor de DISTANCIA_NAVEGADOR. De esta forma, cuando el valor de DISTANCIA_NAVEGADOR = 100, indicaremos que el navegador coincide con la lista de navegadores, tomando otras categorías (Igual/Muy Parecido/Parecido/Diferente/Desconocido) cuando esta distancia se vaya alejando.

3.3.2. Variable País Origen

En este apartado realizaremos un proceso análogo al visto en la Sección 3.3.1, pero utilizando las variables PAIS_ORIGEN y COUNTRY_LIST. Como en el anterior caso, el objetivo será comparar si el país desde el que se realizo la transacción está dentro de la lista de países usuales del usuario que la llevó a cabo, ya que en caso de no ser así, podría ser un indicador de fraude.

De esta forma, crearemos la variable **CLASE_PAIS**, la cual nos indica si el país pertenece a la lista de países usuales(Igual), si no pertenece (Diferente) o si se desconoce parte de la información (Desconocido).

3.3.3. Variable País Destino

Nuevamente realizaremos un proceso similar al visto en las Secciones 3.3.1 y 3.3.2. En este escenario, debemos de extraer primero el país al que se manda la transacción utilizando la variable ID_CTA_DESTINO, en la cual los dos primeros elementos de un número de cuenta indican el país. De esta forma, crearemos la variable **PAIS_DESTINO**.

Tras esto, utilizando las variables PAIS_ORIGEN y PAIS_DESTINO, crearemos la variable **CLASE_PAIS_DESTINO**, la cual nos indica si el país de origen y de destino es coincide (Igual), no coincide (Diferente) o si se desconoce parte de la información (Desconocido).

3.3.4. Variable Hora del Día

A partir de la variable FECHA_COMPLETA, extraeremos la hora a la que se produjo una transacción, creando así la variable **HORA_DEL_DIA**. El objetivo de esta variable es evaluar si las operaciones fraudulentas se realizan a horas inusuales (por ejemplo, de madrugada).

3.4. Representación gráfica de los datos

La representación gráfica de los datos es una parte fundamental a la hora de trabajar con un conjunto de datos y ajustar modelos de clasificación. Esta nos ayudará a entender las relaciones que existen entre las diferentes variables con las que trabajaremos. Además, nos permite conocer la calidad de los propios datos, pudiéndonos hacer una idea de como de la calidad de una variable a la hora de ajustar un modelo de clasificación.

3.4.1. t-SNE

t-SNE (t-Distributed Stochastic Neigbor Embedding) es un método estadístico para visualizar datos de altas dimensiones desarrollado por Sam Roweis y Geoffrey Hinton. Este método es una técnica de reducción no lineal de la dimensión, concretamente, modela cada objeto de alta dimensión volviéndolo un objeto dos-dimensional o tres-dimensional, de manera que objetos similares están modelados como puntos cercanos.

Podemos ver el resultado de aplicar esta técnica al conjunto de datos de prevención de fraude en la Figura (3.2). En ella, podemos apreciar que la mayoría de las observaciones fraudulentas están agrupadas en la misma zona y que hay un poco de solapamiento con las operaciones no fraudulentas. Sin embargo, esta es una reducción del conjunto de datos en dos dimensiones, por lo que es probable que a pesar del solapamiento entre clases los modelos de machine learning aporten buenos resultados.

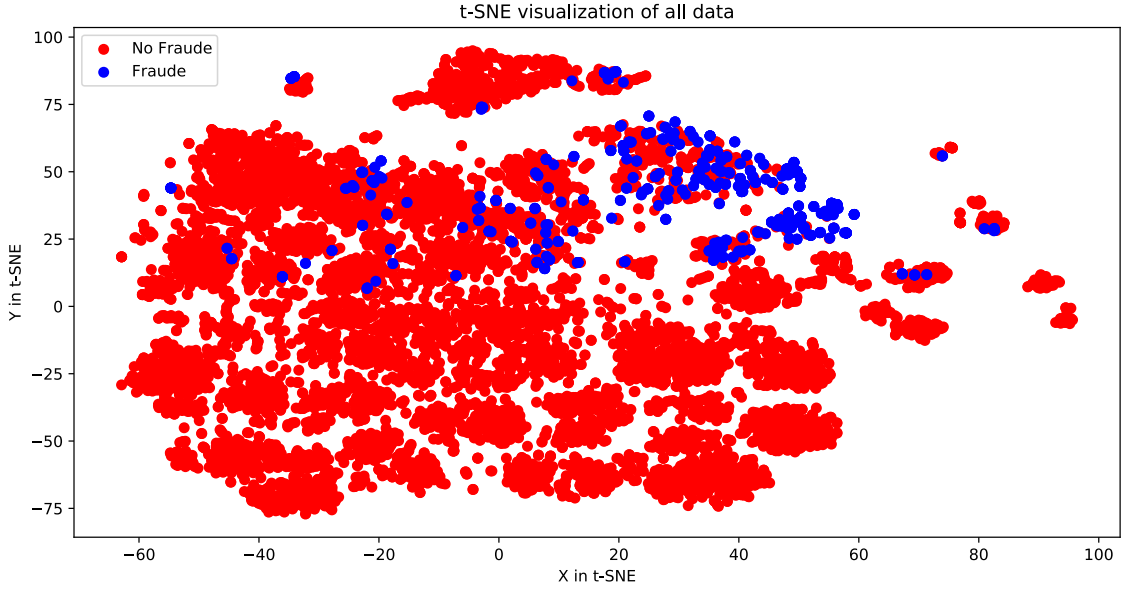

Figura 3.2: Visualización de los datos mediante t-SNE.

3.4.2. Funciones de densidad de probabilidad

Tras ver un resumen de todas las variables del problema con t-SNE, nos centraremos en cada variable en particular. De esta forma, separaremos los conjuntos de datos fraudulento y no fraudulento y representaremos las funciones de densidad de probabilidad de cada variable. Aquellas variables en las que la distribución de los datos de la clase fraudulenta y no fraudulenta sea más diferenciada serán más predictivas a la hora de ajustar un modelo.

Podemos ver la representación de varias de las variables en la Figura (3.3). En ella, podemos ver como parece haber diferencias significativas en las variables LOG_IMPORTE y DISTANCIA_NAVEGADOR entre las clases de Fraude y no Fraude. Estas diferencias serán posteriormente contrastadas utilizando test de bondad de ajustes, pero nos sirven como guía para saber que variables serán las más útiles a la hora de ajustar los modelos. Además de las mencionadas, en un gran número de variables se han encontrado diferencias entre ambas clases, siendo las más destacadas EDAD y DIGITALIDAD_RELATIVA.

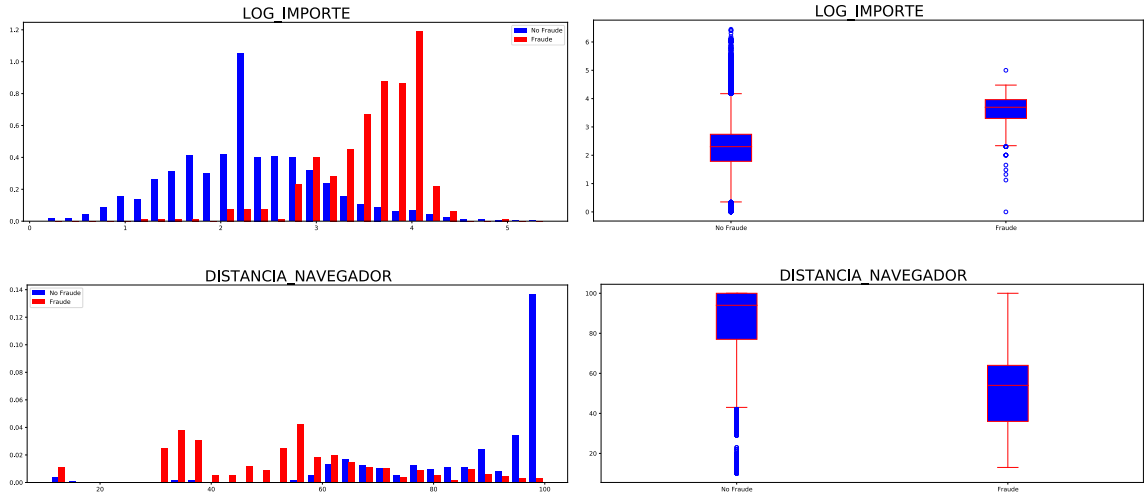

Figura 3.3: Superior Izquierda: Histograma que muestra las funciones de densidad de probabilidad de la variable LOG_IMPORTE de las clases Fraude y no Fraude. Superior Derecha: Boxplot de la variable LOG_IMPORTE de las clases Fraude y no Fraude. Inferior Izquierda: Histograma que muestra las funciones de densidad de probabilidad de la variable DISTANCIA_NAVEGADOR de las clases Fraude y no Fraude. Inferior Derecha: Boxplot de la variable DISTANCIA_NAVEGADOR de las clases Fraude y no Fraude.

3.5. Estudio de correlaciones

Como hemos mencionado en la Sección 2.1, el análisis de las correlaciones nos indica las relaciones que hay entre dos variables y si una nos aporta información sobre la otra. Estudiaremos dos tipos de correlaciones.

3.5.1. Correlación de Pearson

Dadas dos variables aleatorias (X, Y), el coeficiente de correlación de Pearson indica las relaciones lineales que hay entre ambas variables. De esta forma, aplicando el coeficiente de correlación de Pearson al conjunto de variables del problema de prevención de Fraude, podemos estudiar la relación lineal que existe entre las variables. En la Figura (3.4) podemos apreciar una representación gráfica de la matriz de correlaciones del problema, donde en ciertos conjuntos de variables parece haber una alta correlación positiva o negativa.

3.5.2. Correlación de la distancia

La correlación de Pearson es útil como una primera aproximación a conocer la relación que existe entre las diferentes variables del problema. Sin embargo, el coeficiente de correlación de Pearson solo indica las relaciones lineales que existen entre las variables, obviando las relaciones no lineales. Utilizando la correlación de la distancia definida en la Sección 2.1.2 podemos conocer las relaciones lineales y no lineales entre las variables.

Seleccionaremos aquella variables predictoras las cuales superen un valor mínimo $t = 0{,}01$ de correlación con la variable respuesta. Por otro lado, dada la alta dimensionalidad del problema es muy costoso computacionalmente utilizar un método forward para la selección de variables, donde

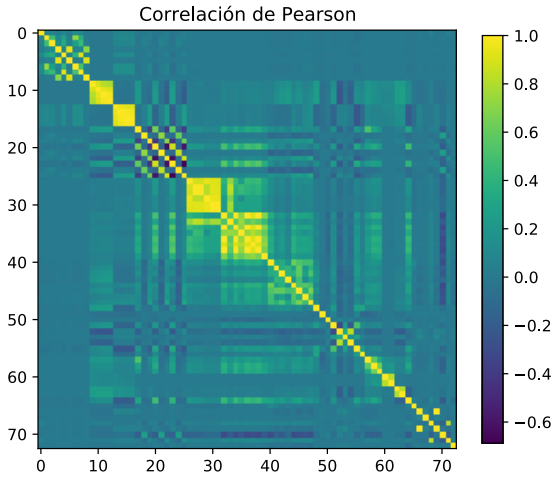

Figura 3.4: Representación gráfica de la matriz de correlaciones de Pearson entre las distintas variables, podemos apreciar en tonos amarillos aquellas variables mas correlacionadas

incluiríamos primero aquellas variables más relacionadas con la variable respuesta y evaluaríamos en función de una métrica.

En lugar de esto, estudiaremos aquellas variables que estén muy correlacionadas entre ellas y tomaremos aquella que tenga un mayor valor de correlación de la distancia con la variable respuesta. Este conjunto de variables será de utilidad a la hora de ajustar los modelos de Random Forest y Redes Neuronales, en los cuales incluir variables poco predictivas empeoraría las métricas de los modelos.

Las variables continuas seleccionadas siguiendo este proceso son las que podemos ver en el Cuadro (3.1).

3.6. Tratamiento de los Outliers

Como hemos mencionado en la Sección 2.2, un outlier es una observación que es numéricamente distante del resto de los datos y la cual puede producir que el ajuste de un modelo cambie de manera drástica.

En un primer momento tratamos de llevar a cabo este proceso mediante el Z-Score. Tras testear la normalidad de las variables continuas y ver que en muchos casos no se cumplían descartamos este método, ya que no se cumplían las hipótesis requeridas.

Tras esto comprobamos el funcionamiento de los rangos intercuartílicos junto con una transformación Box-Cox, pero también los descartamos debido a que el la práctica no se obtenía el funcionamiento requerido. Esto es debido a la propia estructura de los datos, donde el primer y el tercer cuartil están muy próximos en algunas variables como IMPORTE.

Finalmente optamos por no eliminar ningún dato atípico, ya que realizar estos procedimientos sin las hipótesis adecuadas pueden llevar a la pérdida de información importante como puede ser eliminar observaciones que no son atípicas.

3.7. Test de Hipótesis

Utilizaremos ahora los métodos vistos en la Sección 2.3. De esta forma, distinguiremos entre 3 tipos de variables:

- Variables Categóricas.
- Variables Continuas.
- Variables Mixtas.

3.7.1. Variables Categóricas

Sea X una variable discreta y sean X_f aquellas categorías de la variable X las cuales pertenecen a la clase fraudulenta y X_{nf} aquellas que pertenecen a la clase no fraudulenta. Para comparar si la distribución en ambas clases es la misma, utilizaremos el contraste χ^2 visto en la Sección 2.3.2. Tenemos dos opciones:

- **Hay discrepancias entre las clases**: Esto nos indica que en la variable original X hay discrepancia entre las clases fraudulenta y no fraudulenta y, por lo tanto, la variable es de utilidad.
- **No hay discrepancias entre las clases**: Concluimos que no hay indicios que nos hagan pensar que en la variable X haya diferencias entre las clases fraudulenta y no fraudulenta. De esta forma, la variable X no sería predictiva y por lo tanto puede ser descartada.

3.7.2. Variables Continuas

Realizaremos un proceso análogo al visto en la anterior sección. Sea X una variable continua y sean X_f aquellos valores de la variable X los cuales pertenecen a la clase fraudulenta y X_{nf} aquellos que pertenecen a la clase no fraudulenta. Con el objetivo de comparar si la distribución es igual dentro de los datos de la clase fraudulenta y no fraudulenta, utilizaremos el contraste de Kolmogorov-Smirnov. Este contraste comparará las funciones de distribución empíricas de ambas muestras, pudiendo ocurrir dos alternativas:

- **Hay discrepancias entre las clases**: Esto nos indica que en la variable original X hay discrepancia entre las clases fraudulenta y no fraudulenta y, por lo tanto, la variable es de utilidad.
- **No hay discrepancias entre las clases**: Concluimos que no hay indicios que nos hagan pensar que en la variable X haya diferencias entre las clases fraudulenta y no fraudulenta. De esta forma, la variable X no sería predictiva y por lo tanto puede ser descartada.

3.7.3. Variables Mixtas

Definiremos como variables mixtas aquellas variables continuas en las cuales un valor esté repetido en más del 10 % de las observaciones y las trataremos como continuas y categóricas al mismo tiempo. Sea X una variable aleatoria con el valor k repetido más de un 10 % de las veces, crearemos la variable Y de la siguiente forma:

$$y_i = \begin{cases} 0, & \text{si } x_i \text{ no toma el valor } k \\ 1, & \text{si } x_i \text{ toma el valor } k \end{cases} \tag{3.1}$$

Una vez creada la variable Y, separaremos las observaciones en función de la variable respuesta ETIQUETA, obteniendo así el conjunto de datos Y_f en los que todas las transacciones son fraudulentas e Y_{nf} en el que ninguna es fraudulenta. Una vez realizado este paso, tenemos todo lo necesario para aplicar el contraste χ^2 visto en la Sección 2.3.2. De esta forma podremos medir la discrepancia entre

las dos distribuciones observadas y ver si existen diferencias significativas entre ellas. Llegado a este punto, pueden pasar dos cosas:

- **Hay discrepancias entre las clases**: Rechazamos $H_0 : F_f = F_{nf}$ donde F_f es la función de distribución de la muestra de las transacciones fraudulentas y F_{nf} es la función de distribución de las transacciones no fraudulentas. Esto nos indica que en la variable original X hay discrepancia entre las clases fraudulenta y no fraudulenta y, por lo tanto, la variable es de utilidad.
- **No hay discrepancias entre las clases**: Aceptamos $H_0 : F_f = F_{nf}$ donde F_f es la función de distribución de la muestra de las transacciones fraudulentas y F_{nf} es la función de distribución de las transacciones no fraudulentas. No hemos encontrado hasta el momento indicios que nos hagan pensar que en la variable X haya diferencias entre las clases fraudulenta y no fraudulenta.

Una vez realizado el contraste χ^2, seleccionaremos aquellas variables en las cuales no hayamos encontrado discrepancias entre la clase fraudulenta y no fraudulenta. El objetivo ahora será evaluarlas como variables continuas.

Sea X una aleatoria con el valor k repetido más de un 10 % de las veces en la cual no se encontraron discrepancias entre la clase fraudulenta y no fraudulenta, eliminaremos dicho valor k repetido y realizaremos a los valores restantes el contraste de Kolmogorov-Smirnov. Nuevamente el objetivo es evaluar si la distribución entre ambas muestras es coincidente o si tienen diferente distribución. En este escenario, como ocurría con el contraste χ^2, pueden pasar dos cosas:

- **Hay discrepancias entre las clases**: Esto nos indica que en la variable original X hay discrepancia entre las clases fraudulenta y no fraudulenta y, por lo tanto, la variable es de utilidad.
- **No hay discrepancias entre las clases**: Concluimos que no hay indicios que nos hagan pensar que en la variable X haya diferencias entre las clases fraudulenta y no fraudulenta. De esta forma, la variable X no sería predictiva y por lo tanto puede ser descartada.

3.8. Remuestreo de los datos

Realizaremos el remuestreo de los datos utilizando los métodos descritos en la Sección 2.4. El objetivo de esto será la obtención de un conjunto de datos balanceado sobre el cual los modelos de clasificación funcionen mejor. Esto lo haremos siguiendo el siguiente procedimiento:

- **Separación del conjunto de datos en entrenamiento y test**: Comenzaremos este apartado separando ambas muestras. De esta forma, tomaremos el 90 % de los datos como muestra de entrenamiento, mientras que utilizaremos el 10 % restante como muestra test.
- **Remuestreo del conjunto de datos de entrenamiento**: Empezaremos aplicando validación cruzada. De esta forma, dividiremos el conjunto de entrenamiento en K subconjuntos, utilizando como datos de validación uno de los subconjuntos y como datos de entrenamiento los $(K-1)$ restantes. Tras esto, remuestrearemos unicamente el conjunto de datos de entrenamiento. Este procedimiento no se puede llevar a cabo en la muestra test ni en el subconjunto de validación, ya que supondría la contaminación de la propia muestra y nos podría llevar a resultados erróneos. En cada iteración del proceso de validación cruzada remuestrearemos los subconjuntos utilizados para entrenar el modelo.

De esta forma, tras aplicar las técnicas de remuestreo, contaríamos con una única muestra de datos test sobre la que evaluaremos el ajuste de los modelos y con 5 muestras de datos de entrenamiento:

- Muestra de datos sin remuestrear (Sin Remuestreo).

- Muestra de datos remuestreados utilizando undersampling (Undersampling).
- Muestra de datos remuestreados utilizando oversampling (Oversampling).
- Muestra de datos remuestreados utilizando SMOTE (SMOTE).
- Muestra de datos remuestreados utilizando Ct-GAN (Ct-GAN).

Podemos ver una representación de los datos remuestreados en la Figura (3.5). Además, también podemos apreciar diferencias significativas entre las distintas técnicas, desde undersampling, en la cual reducimos el número de observaciones de la clase dominante, hasta Ct-GAN, en la cual los datos de las diferentes clases están más agrupados entre ellos. Tras el remuestreo de los datos, ya tenemos todo listo para entrenar los diferentes modelos de clasificación en las muestras de entrenamiento.

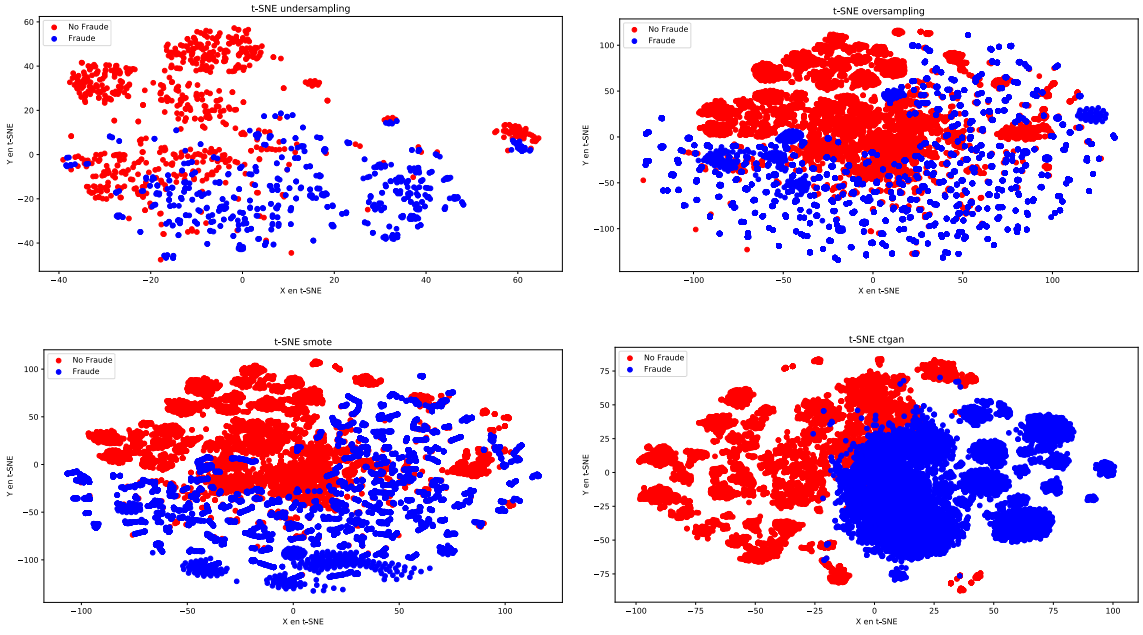

Figura 3.5: Superior Izquierda: Visualización de los datos remuestreados mediante undersampling utilizando t-SNE. Superior Derecha: Visualización de los datos remuestreados mediante oversampling utilizando t-SNE. Inferior Izquierda: Visualización de los datos remuestreados mediante SMOTE utilizando t-SNE. Inferior Derecha: Visualización de los datos remuestreados mediante Ct-GAN utilizando t-SNE.

Variable	Correlación de la distancia
IMPORTE	0.096
TXN_M1_TXN_AMT_MEAN	0.019
ACCESSES_BMOVIL	0.014
ACCESSES_BMOVIL_M1	0.01
ACCESSES_BE_W1	0.016
OP_TRANSACTION_W1_CNT_RATIO	0.027
GLOBAL_BMOVIL_OP_D1_3M_CNT_RATIO	0.011
GLOBAL_TRANSACCION_OP_M1_CNT_RATIO	0.038
GLOBAL_LOGIN_OP_M1_CNT_RATIO	0.028
GLOBAL_LOGIN_OP_3M_CNT_RATIO	0.041
USAGE_HOURS_WK_MEAN	0.063
USAGE_WKDAY_MEAN	0.074
USAGE_MNTHDAY_STD	0.08
USO_TARJETAS_12M	0.01
MENSAJERIA	0.016
DIGITALIDAD_RELATIVA	0.021
EDAD	0.04
DIGITALIDAD_RELATIVA_EDAD	0.01
DISTANCIA_NAVEGADOR	0.11
HORA_DEL_DIA	0.02

Cuadro 3.1: Cuadro en el que podemos visualizar las variable seleccionadas utilizando la correlación de la distancia entre la propia variable y la variable respuesta ETIQUETA.

Capítulo 4

Modelos de Clasificación y resultados

En este Capítulo llevaremos a cabo el entrenamiento de los modelos de machine learning junto con las predicciones en el conjunto de datos test y la comparación entre los modelos. Como hemos mencionado en la Sección 3.8, utilizaremos los 5 conjuntos de datos de entrenamiento para ajustar los modelos, mientras que usaremos los datos test para contrastarlos.

Los modelos utilizados serán los vistos en la Sección 2.6 : Regresión Logística, Random Forest, XGBoost y Redes Neuronales.

La mayoría de los modelos planteados tienen una serie de hiperparámetros óptimos, los cuales en la práctica son difíciles de localizar. Para utilizar unos hiperparámetros que den buenos resultados, tomaremos una malla de valores y contrastaremos el funcionamiento de los modelos en términos de la F_1 *score* y el *recall* vistos en la Sección 2.7, todo ello utilizando validación cruzada.

De esta forma, llevaremos a cabo el estudio de los modelos de la siguiente forma. Comenzaremos definiendo el objetivo del capítulo junto con la metodología utilizada en las Secciones 4.1 y 4.2 respectivamente. En la Sección 4.3 ajustaremos los diferentes modelos de machine learning y obtendremos los resultados. Así, ajustaremos el modelo de Regresión Logística a cada uno de los conjuntos de datos remuestreados, contrastando los ajustes con los datos test y viendo los resultados. Realizaremos un proceso análogo con los modelos de Random Forest, XGBoost y Redes Neuronales, además de centrarnos en la búsqueda de hiperparámetros. Más adelante, en la Sección 4.4, compararemos los modelos ajustados utilizando métricas globales como el F_1 *score* o el *recall*. Por último, interpretaremos el modelo que obtenga mejores resultados en la Sección 4.5 y extraeremos conclusiones en la Sección 4.6.

4.1. Objetivo

Como hemos visto, tenemos varias herramientas (F_1 *score*, *precision* y *recall*) que nos permiten comprobar el funcionamiento de los diferentes modelos de machine learning. El objetivo será evaluar la calidad de los modelos presentados en la Sección 4.2 siguiendo estas métricas. Además, presentaremos el tiempo de predicción en una nueva observación y el el AUC (Area Under the Curve), métricas las cuales son también de interés a la hora de evaluar dichos modelos.

De esta forma seleccionaremos el modelo que mejor resultados obtenga, para posteriormente, centrarnos en dicho modelo y evaluar más de cerca su comportamiento.

4.2. Metodología

Para la realización de este trabajo, se ha empleado el lenguaje de programación Python Core Team (2019). Por otro lado, en muchos de los escenarios los modelos han sido ajustados a partir de una malla de hiperparámetros usando validación cruzada, descrita en la Sección 2.5.

Es importante remarcar que la validación cruzada debe de realizarse antes del remuestreo de los datos. De esta forma, el primer paso será dividir los datos de entrenamientos en K subconjuntos y tras esto, aplicar la técnica de remuestreo correspondiente en los $(K-1)$ subconjuntos que se utilicen para entrenar el modelo. Es primordial llevar a cabo el proceso siguiendo los pasos en este orden, ya que en caso de realizar la validación cruzada con una muestra previamente remuestreada, el conjunto de validación podría no tener la misma distribución que el conjunto de datos test, pudiéndonos llevar esto a resultados erróneos. Podemos ver una representación gráfica de esto en la Figura (4.1).

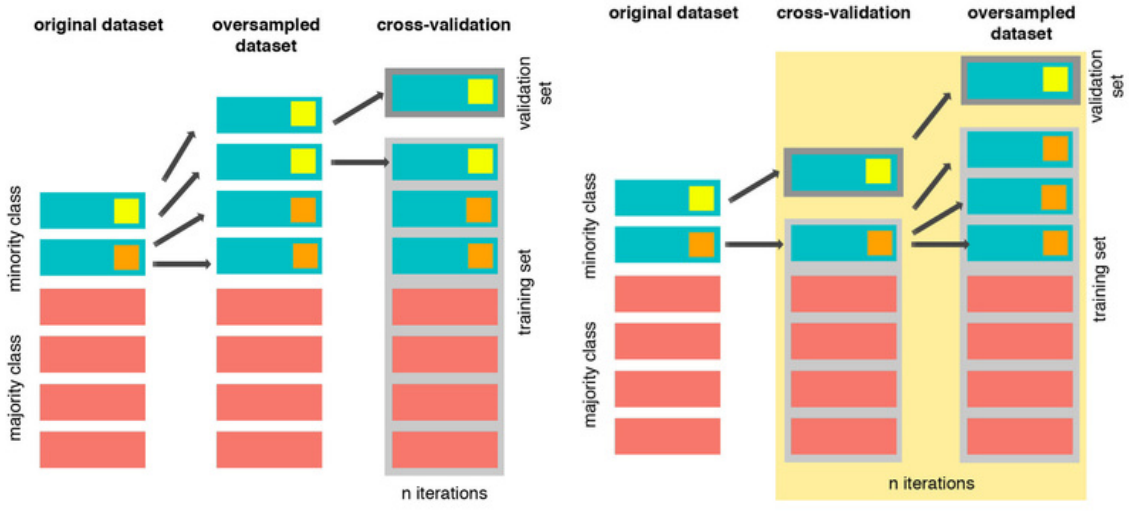

Figura 4.1: Izquierda: Representación gráfica de lo que supone el remuestreo de los datos antes de utilizar validación cruzada. Estaríamos utilizando para entrenar los mismos datos que para validar. Además, en caso de utilizar otras técnicas como SMOTE, testearíamos datos sintéticos, pudiendo tener una distribución diferente que el dataset original. Derecha: Representación gráfica de como realizar el proceso correctamente. Imágenes obtenidas es Altini (2015).

Con esto en mente, se ha procedido de la siguiente forma para cada modelo:

4.2.1. Regresión Logística

La regresión logística ha sido utilizada como modelo base, con el fin de comparar sus resultados con los resultados de los siguientes modelos. En este caso, hemos ajustado el modelo utilizando un término de penalización $\mathcal{L}_2$. Además, debido al carácter desbalanceado de los datos, hemos seleccionado que los pesos se ajusten automáticamente de manera inversamente proporcional a la frecuencia de las clases.

Por otro lado, se ha intentado realizar una selección de variables, pero esta ha sido descartada debido a que se han obtenido peores resultados que utilizando el conjunto total de variables. También se ha estudiado la implementación de un modelo Lasso, nuevamente sin conseguir los resultados deseados.

Los escenarios a evaluar serán los siguientes:

- Sin Remuestrear (LR1)
- Undersampling (LR2)
- Oversampling (LR3)
- SMOTE (LR4)
- Ct-GAN (LR5)

4.2.2. Random Forest

Como hemos mencionado en el Algoritmo 1, para cada árbol creado debemos de seleccionar aleatoriamente m variables de las p totales. Debido a esto, en el Random Forest es primordial realizar una selección de variables adecuada, ya que nos ahorraremos construir aquellos árboles con variables poco predictivas y obtendremos por lo tanto mejores resultados.

En este caso, los escenarios a evaluar serán:

- Sin Remuestrear (RF1)
- Undersampling (RF2)
- Oversampling (RF3)
- SMOTE (RF4)
- Ct-GAN (RF5)

Además, estudiaremos la siguiente malla de hiperparámetros:

- **n_estimators**: Número de árboles en el bosque aleatorio.
- **max_depth**: Máxima profundidad del árbol. Si es None, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos que min_samples_split.
- **class_weight**: Peso asociado con cada una de las clases. Por defecto se supone que todas las clases tienen peso igual a 1. El modo "balanced" ajusta los pesos de manera inversamente proporcional a la frecuencia de las clases.
- **min_samples_split**: Mínimo número de observaciones que se requieren en cada división de un nodo.
- **min_samples_leaf**: Mínimo número de observaciones que se requieren para ser un nodo hoja.

Por otro lado, testearemos cada una de las 72 posibles combinaciones de lo siguientes hiperparámetros:

- **n_estimators**: $[100, 500, 1000]$
- **max_depth**: $[50, 200, None]$
- **class_weight**: $[balanced\_subsample, None]$
- **min_samples_split**: $[2, 5]$
- **min_samples_leaf**: $[1, 2]$

4.2.3. XGBoost

En este caso, a diferencia de lo que ocurre en el algoritmo Random Forest, no será necesario una selección de variables. Esto es debido a la naturaleza del propio algoritmo, el cual utiliza aquellas variables más predictivas para construir el árbol e ignora las que no son útiles.

Estudiaremos los escenarios:

- Sin Remuestrear (XGB1)
- Undersampling (XGB2)
- Oversampling (XGB3)
- SMOTE (XGB4)
- Ct-GAN (XGB5)

Como en el caso del modelo de Random Forest, ajustaremos el mejor modelo a partir de una malla de los siguientes hiperparámetros:

- **max_depth**: Máxima profundidad del árbol.
- **gamma**: Mínima reducción en la loss function requerida para hacer una partición en un nodo hoja del árbol.
- **colsample_bytree**: Ratio de columnas con las que se construirá cada árbol.
- **subsample**: Ratio de observaciones que se tomarán en cada árbol.
- **learning_rate**: Reducción en el tamaño de paso usada en cada actualización para prevenir el overfitting.
- **scale_pos_weight**: Controla el balance de los pesos positivos y negativos, útil para clases desbalanceadas.

Por otro lado, estudiaremos las diferentes combinaciones entre cada uno de los valores de cada categoría, testeando en este caso 144 posibilidades:

- **max_depth**: $[4, 5, 7]$
- **gamma**: $[0, 1]$
- **colsample_bytree**: $[0{,}7, 1]$
- **subsample**: $[0{,}7, 1]$
- **learning_rate**: $[0{,}2, 0{,}3]$
- **scale_pos_weight**: $[0{,}1, 1, 10]$

4.2.4. Redes Neuronales

Las redes neuronales son modelos de machine learning en los cuales es necesario ajustar una gran cantidad de parámetros. Debido a esto, realizar una selección de variables adecuadas ahorrará muchos costes computacionales y agilizará los cálculos de una nueva predicción. Además, también realizaremos una estandarización de las variables por el mismo motivo.

Los últimos escenarios a evaluar serán:

- Sin Remuestrear (NN1)
- Undersampling (NN2)
- Oversampling (NN3)
- SMOTE (NN4)
- Ct-GAN (NN5)

Ahora, los parámetros de interés serán los siguientes:

- **batch_size**: Número de datos que utilizaremos en cada paso de forward y backward propagation para recalcular los pesos W.
- **n_neuronas**: Número de neuronas utilizadas en cada capa.
- **dropout**: El dropout es una técnica utilizada en redes neuronales para evitar el overfitting. Cada neurona, tendrá una probabilidad de ser "desactivada", haciendo que se repartan más los pesos del modelo y que la predicción no dependa unicamente de un número pequeño de neuronas. Este valor nos indicará la probabilidad que tiene cada neurona de ser "desactivada".
- **n_layers**: Número de capas que tiene el modelo.

Por otro lado, ahora evaluaremos un total de 24 modelos:

- **batch_size**: $[100, 300, 500]$
- **n_neuronas**: $[20, 25]$
- **dropout**: $[0{,}2, 0{,}25]$
- **n_layers**: $[3, 4]$

4.3. Resultados

Una vez conocido el ajuste de los modelos, procederemos a su evaluación a partir de las métricas descritas en la Sección 2.7. Los resultados obtenidos son los siguientes:

Regresión Logística sin remuestreo (LR1)

- *precision*: 0,85
- *recall*: 0,233
- F_1 *score*: 0,366
- Tiempo medio de predicción de una nueva observación: 0,004 segundos
- *AUC*: 0,616

Observado / Predicción	No Fraude	Fraude
No Fraude	29944	3
Fraude	56	17

Regresión Logística undersampling (LR2)

- *precision*: 0,032
- *recall*: 0,918
- F_1 *score*: 0,062
- Tiempo medio de predicción de una nueva observación: 0,004 segundos
- *AUC*: 0,925

Observado / Predicción	No Fraude	Fraude
No Fraude	27941	2006
Fraude	6	67

Regresión Logística oversampling (LR3)

- *precision*: 0,035
- *recall*: 0,918
- F_1 *score*: 0,068
- Tiempo medio de predicción de una nueva observación: 0,004 segundos
- *AUC*: 0,928

Observado / Predicción	No Fraude	Fraude
No Fraude	28129	1818
Fraude	6	67

Regresión Logística SMOTE (LR4)

- *precision*: 0,048
- *recall*: 0,904
- F_1 *score*: 0,092

- Tiempo medio de predicción de una nueva observación: 0,004 segundos
- *AUC*: 0,930

Observado / Predicción	No Fraude	Fraude
No Fraude	28651	1296
Fraude	7	66

Regresión Logística Ct-GAN (LR5)

- *precision*: 0,067
- *recall*: 0,301
- F_1 *score*: 0,11
- Tiempo medio de predicción de una nueva observación: 0,004 segundos
- *AUC*: 0,646

Observado / Predicción	No Fraude	Fraude
No Fraude	29643	304
Fraude	51	22

Random Forest sin remuestreo (RF1)

- *precision*: 0,584
- *recall*: 0,712
- F_1 *score*: 0,642
- Tiempo medio de predicción de una nueva observación: 0,3 segundos
- *ROC_AUC*: 0,856

Observado / Predicción	No Fraude	Fraude
No Fraude	29910	37
Fraude	21	52

Random Forest undersampling (RF2)

- *precision*: 0,446
- *recall*: 0,616
- F_1 *score*: 0,517
- Tiempo medio de predicción de una nueva observación: 0,205 segundos
- *AUC*: 0,807

Observado / Predicción	No Fraude	Fraude
No Fraude	29891	56
Fraude	28	45

Random Forest oversampling (RF3)

- *precision*: 0,676
- *recall*: 0,685
- F_1 *score*: 0,680
- Tiempo medio de predicción de una nueva observación: 0,198 segundos
- *AUC*: 0,842

Observado / Predicción	No Fraude	Fraude
No Fraude	29923	24
Fraude	23	50

Random Forest SMOTE (RF4)

- *precision*: 0,505
- *recall*: 0,658
- F_1 *score*: 0,571
- Tiempo medio de predicción de una nueva observación: 0,2 segundos
- *AUC*: 0,828

Observado / Predicción	No Fraude	Fraude
No Fraude	29900	47
Fraude	25	48

Random Forest Ct-GAN (RF5)

- *precision*: 0,56
- *recall*: 0,644
- F_1 *score*: 0,599
- Tiempo medio de predicción de una nueva observación: 0,204 segundos
- *AUC*: 0,821

Observado / Predicción	No Fraude	Fraude
No Fraude	29910	37
Fraude	26	47

XGBoost sin remuestreo (XGB1)

- *precision*: 0,769
- *recall*: 0,685
- F_1 *score*: 0,725
- Tiempo medio de predicción de una nueva observación: 0,009 segundos
- *AUC*: 0,842

Observado / Predicción	No Fraude	Fraude
No Fraude	29932	15
Fraude	23	50

XGBoost undersampling (XGB2)

- *precision*: 0,411
- *recall*: 0,603
- F_1 *score*: 0,489

- Tiempo medio de predicción de una nueva observación: 0,014 segundos
- *AUC*: 0,8

Observado / Predicción	No Fraude	Fraude
No Fraude	29884	63
Fraude	29	44

XGBoost oversampling (XGB3)

- *precision*: 0,75
- *recall*: 0,699
- F_1 *score*: 0,723
- Tiempo medio de predicción de una nueva observación: 0,02 segundos
- *AUC*: 0,849

Observado / Predicción	No Fraude	Fraude
No Fraude	29930	17
Fraude	22	51

XGBoost SMOTE (XGB4)

- *precision*: 0,684
- *recall*: 0,712
- F_1 *score*: 0,698
- Tiempo medio de predicción de una nueva observación: 0,021 segundos
- *AUC*: 0,856

Observado / Predicción	No Fraude	Fraude
No Fraude	29923	24
Fraude	21	52

XGBoost Ct-GAN (XGB5)

- *precision*: 0,787
- *recall*: 0,658
- F_1 *score*: 0,716
- Tiempo medio de predicción de una nueva observación: 0,013 segundos
- *AUC*: 0,829

Observado / Predicción	No Fraude	Fraude
No Fraude	29934	13
Fraude	25	48

Redes Neuronales sin remuestreo (NN1)

- *precision*: 0,44
- *recall*: 0,603
- F_1 *score*: 0,509
- Tiempo medio de predicción de una nueva observación: 0,054 segundos
- *AUC*: 0,8

Observado / Predicción	No Fraude	Fraude
No Fraude	29891	56
Fraude	29	44

Redes Neuronales undersampling (NN2)

- *precision*: 0,188
- *recall*: 0,699
- F_1 *score*: 0,297
- Tiempo medio de predicción de una nueva observación: 0,058 segundos
- *AUC*: 0,846

Observado / Predicción	No Fraude	Fraude
No Fraude	29727	220
Fraude	22	51

Redes Neuronales oversampling (NN3)

- *precision*: 0,396
- *recall*: 0,521
- F_1 *score*: 0,45
- Tiempo medio de predicción de una nueva observación: 0,056 segundos
- *AUC*: 0,759

Observado / Predicción	No Fraude	Fraude
No Fraude	29889	58
Fraude	35	38

Redes Neuronales SMOTE (NN4)

- *precision*: 0,583
- *recall*: 0,479
- F_1 *score*: 0,526
- Tiempo medio de predicción de una nueva observación: 0,056 segundos
- *AUC*: 0,739

Observado / Predicción	No Fraude	Fraude
No Fraude	29922	25
Fraude	38	35

Redes Neuronales Ct-GAN (NN5)

- *precision*: 0,527
- *recall*: 0,397
- F_1 *score*: 0,453

- Tiempo medio de predicción de una nueva observación: 0,06 segundos

- *AUC*: 0,698

Observado / Predicción	No Fraude	Fraude
No Fraude	29921	26
Fraude	44	29

4.4. Comparación de los modelos

Tras el cálculo de las métricas de los modelos vistas en la Sección 4.3, pasaremos a realizar una comparación de estas con el objetivo de encontrar el modelo que consiga los mejores resultados.

En el Cuadro (4.1) podemos encontrar una comparativa de las métricas utilizadas en cada uno de los modelos. En ella, el modelo que obtiene mejores resultados en término de F_1 *score* es el modelo XGB1. Sin embargo, debido a que el modelo XGB3 obtiene resultados muy parecidos en términos de F_1 *score* y mejores resultados en términos de *recall*, tomaremos este como el mejor modelo. Además, el tiempo de predicción de una nueva observación es 0,02 siendo menor de 1 segundo y por lo tanto apto para poder ser puesto en producción.

Para finalizar la comparación de los modelos, mostraremos gráficamente el valor de los F_1 *score* de los diferentes modelos y el tiempo de ejecución. Podemos ver claramente en la Figura (4.2) que los modelos basados en árboles de decisión obtienen mejores resultados que el modelo de regresión logística y las redes neuronales densas. Además, de igual forma que en el Cuadro (4.1), los modelos que mejores resultados obtienen son el XGB1 y el XGB3.

Para comprobar estos resultados, se realizó el ajuste de los modelos de Random Forest y XGBoost tomando 20 muestras diferentes de entrenamiento y test. En ambos casos los hiperparámetros han sido ajustados a partir de los que se obtuvieron utilizando validación cruzada. Podemos ver los resultados en la Figura (4.3), en la cual los modelos XGB1 y XGB3 obtienen los mejores resultados.

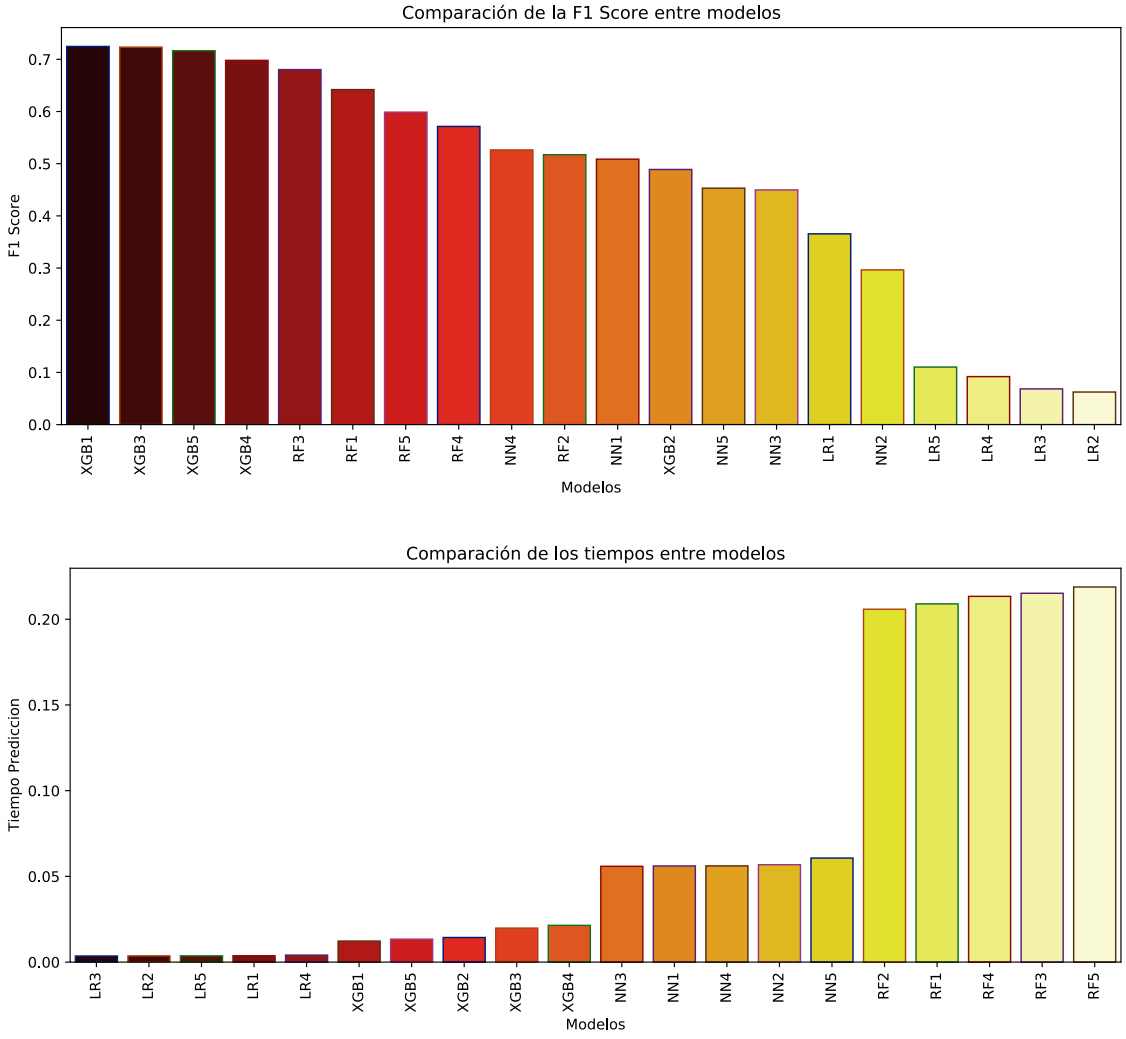

Figura 4.2: Gráfico superior: Comparación del F_1 *score* de los modelos estudiados en el Capítulo 4. Gráfico inferior: Comparación del tiempo de ejecución de los modelos.

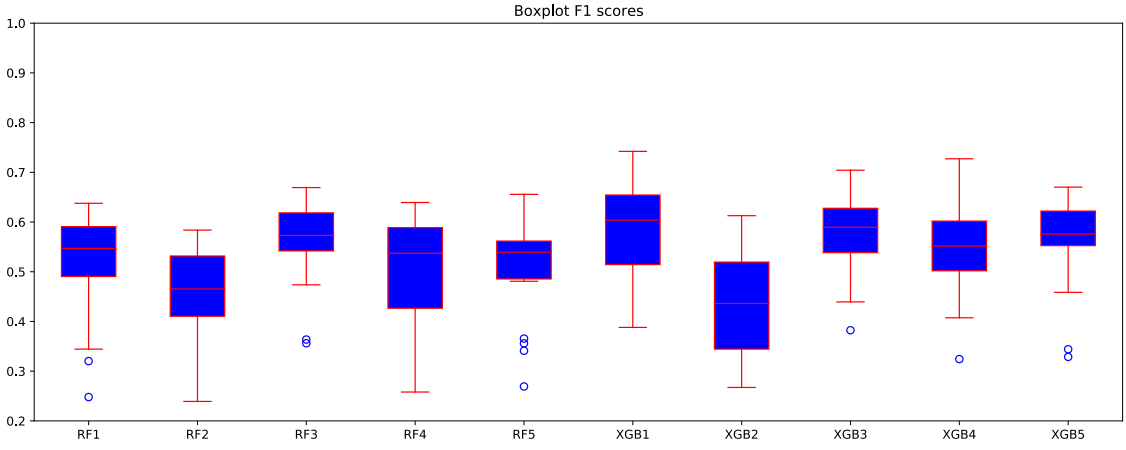

Figura 4.3: Representación gráfica de los F_1 obtenidos al ajustar los modelos de Random Forest y XGBoost. En ellos se realizó la división entre el conjunto de entrenamiento y test aleatoriamente.

4.5. Interpretación del mejor modelo

Una vez seleccionado el mejor modelo en la Sección 4.4, el cual se corresponde con XGB3 (XGBoost con oversampling), pasaremos a su interpretación. Para eso, nos ayudaremos de la librería SHAP de Python, la cual esta basada en el Artículo Lundberg and Lee (2017) y sirve para interpretar modelos. SHAP significa SHapley Additive exPlanations y esta basado en el valor de Shapley, es decir, en la media de las contribuciones marginales de cada variable al modelo.

4.5.1. Efectos Globales

Comenzaremos con el gráfico de importancia de las variables. Este gráfico está basado en los valores SHAP y muestra las variables más significativas en orden descendente, de manera que aquellas que se encuentran más arriba contribuyen más al modelo y tienen un alto nivel predictivo.

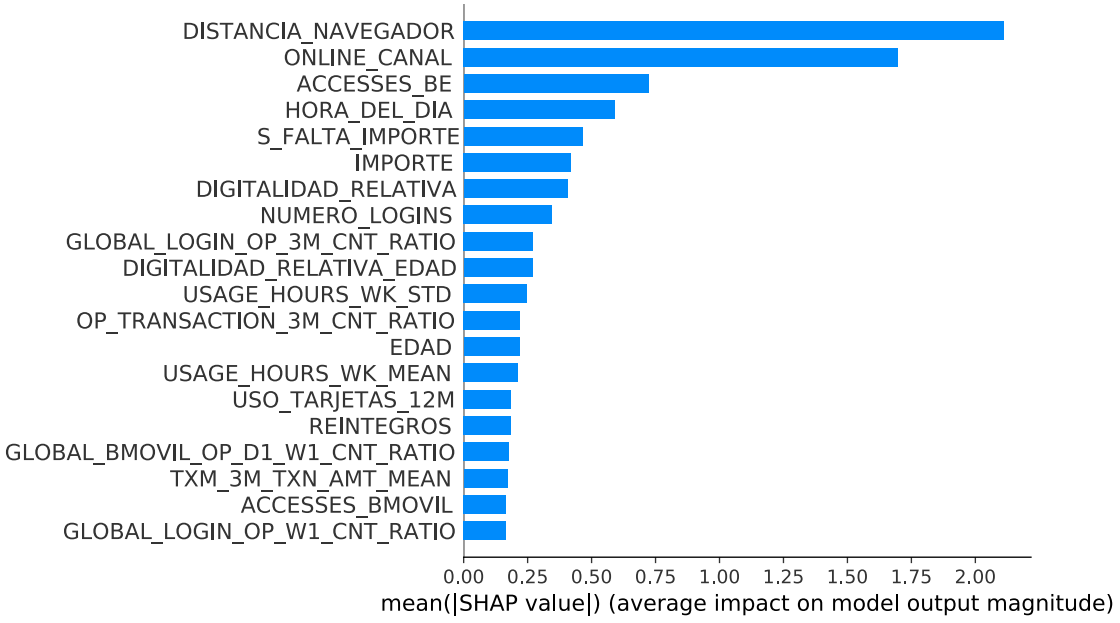

Figura 4.4: Gráfico que muestra la importancia de las variables en función del impacto que tengan en el modelo.

Como podemos ver en la Figura (4.4), la variable más importante sería DISTANCIA_NAVEGADOR, seguida por CANAL y ACCESSES_BE. Es importante mencionar que estamos representando solamente aquellas variables con mayor contribución a pesar de que existen variables con contribución positiva las cuales no aparecen representadas. El problema del gráfico de importancia de las variables es que no muestra la relación de las variables predictoras con la variable respuesta. Esto lo podemos solucionar utilizando la función 'shap.summary_plot()', la cual nos aporta la siguiente información:

- Importancia de las características: Las variables están ordenadas en orden descendente.
- Impacto: La localización horizontal muestra si el efecto del valor esta asociado con una predicción mayor o menor.
- Valor original: El color muestra si la variable toma valores altos o bajos para una observación.

- Correlación: Podemos ver la relación que hay entre los valores altos de una variable (color) y el impacto en el eje de las X.

De esta forma, en la Figura (4.5), interpretaríamos que la variable DISTANCIA_NAVEGADOR es la más importante. Además, podemos apreciar que valores altos de la variable (color rojo) están asociados con un impacto negativo en el eje de las X y viceversa, lo cual nos indica que hay una correlación negativa con la variable objetivo. Esto lo podemos interpretar de la siguiente forma: cuando una operación proviene de un navegador que no es el usual, la variable DISTANCIA_NAVEGADOR tomará valores bajos y por lo tanto la probabilidad de que la operación sea fraudulenta es mayor.

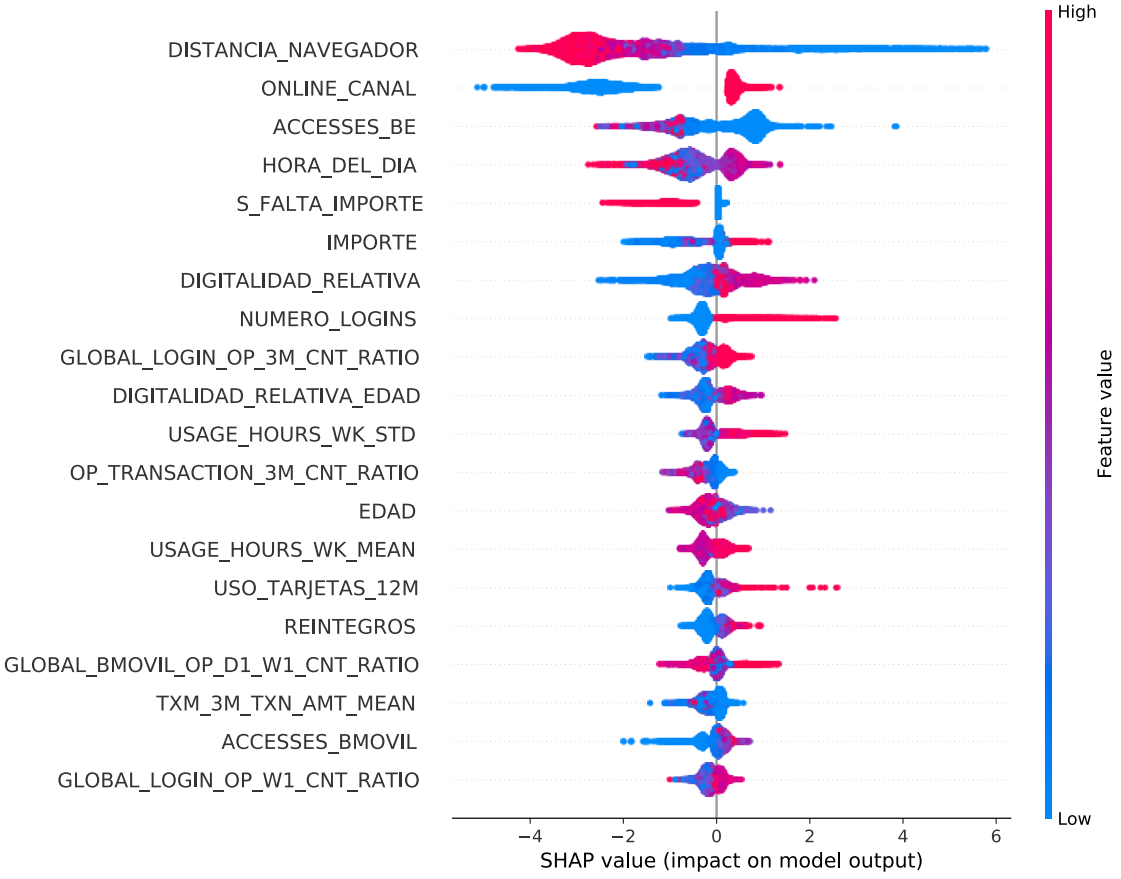

Figura 4.5: Gráfico resumen de la importancia de las variables y sus interacciones con la variable respuesta.

Mientras que los gráficos resumen ofrecen una visión general de cada característica, un gráfico de dependencia SHAP muestra como varía el resultado del modelo según el valor de la característica. La variable utilizada en el color se toma automáticamente para ver que podría impulsar estas interacciones. Podemos ver una representación de esto en la Figura (4.6), donde valores más altos de la variable DISTANCIA_NAVEGADOR están asociados a una menor probabilidad de fraude. Por otro lado, también podemos apreciar la interacción de la variable ACCESSES_BE. Así, cuando la variable DISTANCIA_NAVEGADOR toma valores altos, vemos que hay una mayor probabilidad de fraude cuando la variable ACCESSES_BE también los toma.

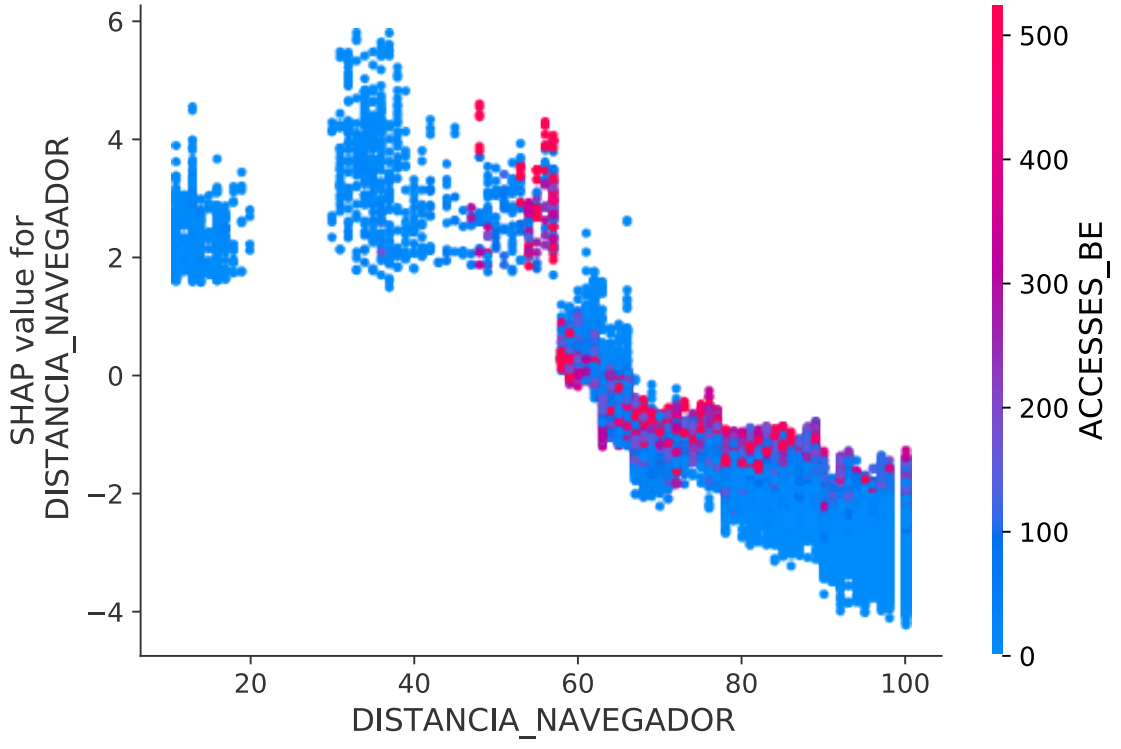

Figura 4.6: Gráfico que muestra la relación entre la variable DISTANCIA_NAVEGADOR y la variable respuesta junto con el efecto que tiene en estas la variable GLOBAL_TRANSACCION_OP_M1_CNT_RATIO.

4.5.2. Efectos Locales

Por otro lado, también nos interesa la interpretación de algunas observaciones particulares y su funcionamiento en el modelo, especialmente aquellas que fueron mal clasificadas. De esta forma, tomaremos dos observaciones, una bien clasificada y otra mal clasificada y estudiaremos como las variables llegan a esa predicción.

Empezaremos así con los gráficos de decisión, los cuales nos muestran como evoluciona la probabilidad de fraude de una observación en función de las variables del problema. Podemos ver una representación de esto en la Figura (4.7). Así, los números en paréntesis son los valores predichos, mientras que la línea gris nos muestra el valor base y la línea que se mueve a izquierda o derecha las fuerzas de las variables.

En el gráfico superior de la Figura (4.7) nos encontraremos con la observación mal clasificada. En ella podemos apreciar que las variables DISTANCIA_NAVEGADOR y DIGITALIDAD_RELATIVA son las dos variables más influyentes, donde DISTANCIA_NAVEGADOR empuja a dicha observación a una mayor probabilidad de fraude. Sin embargo, esto no es suficiente, obteniendo una probabilidad de 0,12 y siendo por lo tanto la predicción No Fraudulenta. Además, parece que hay ciertos datos faltantes, como la EDAD de la persona, la cual se indica que es de 0 años.

Por otro lado, en el gráfico inferior apreciamos como hay un gran número de variables explicativas que llevan al modelo a predecir que la observación es Fraudulenta, siendo las más relevantes

DISTANCIA_NAVEGADOR y NUMERO_LOGINS.

Podemos ver una representación alternativa de esto en la Figura (4.8), donde se representa la fuerza de cada una de las variables y como empujan la probabilidad de fraude hacia niveles mayores o menores.

4.6. Conclusión

Tras la realización del estudio en los distintos escenarios, evaluando los modelos en términos del F_1 *score* y *recall*, podemos observar:

- El modelo que obtiene mejores resultados es el nombrado como XGB3, el cual se basa en el modelo XGBoost combinado con la técnica de oversampling. Además, obtenemos un resultado aceptable en términos del tiempo de predicción de una nueva observación, por lo que sería el modelo candidato a poner en producción.
- Parece que en general el modelo XGBoost consigue los mejores resultados, seguido de cerca por Random Forest. Por contra, el modelo de redes neuronales densas y de regresión logística se alejan del rendimiento de estos. Los resultados coinciden con lo esperado, ya que tanto XGBoost como Random Forest son actualmente dos técnicas punteras a la hora de afrontar problemas de clasificación.
- Por otro lado, parece que las técnicas de remuestreo no han conseguido en muchos casos una mejora en el rendimiento.
- En caso de descartar el modelo XGB3 y necesitar otro modelo, el XGB1 o RF3 también obtienen buenos resultados y son aptos para ser puestos en producción. Además, los modelos de Random Forest suelen ser más fácilmente interpretables y más sencillo el ajuste de sus hiperparámetros que en el caso del XGBoost, por lo que RF3 es una buena alternativa.
- En general, los modelos basados en árboles de decisión suelen obtener buenos resultados en los problemas de clasificación. Con estos modelos podemos utilizar variables discretas y continuas y no requieren de una normalización de los datos. Por otro lado, son más robustos que otros modelos frente a outliers y en el caso del XGBoost no es necesario realizar una selección de variables. Unicamente su interpretabilidad podría ser un problema, pero como hemos visto en la Sección 4.5, disponemos de herramientas las cuales nos hacen más fácil este trabajo.
- Por último, mencionar la gran cantidad de técnicas que se pueden aplicar en este trabajo, siendo prácticamente imposible poner todas en funcionamiento. Así, se podría profundizar en la creación de nuevas variables o en algún criterio de selección de las mismas más optimizado. También podríamos estudiar más profundamente la combinación de la selección de variables con las técnicas de remuestreo, o incluir una malla de hiperparámetros más grande para ver que efecto tienen en los modelos. En general no hay una solución directa para este tipo de problemas, ya que la modificación de una de las técnicas correspondientes con el análisis exploratorio puede llevar a alterar los resultados en las demás técnicas.

Modelos	*Precision*	*Recall*	F_1 *score*	Tiempo predicción
LR1	0,85	0,233	0,366	0,004
LR2	0,032	0,918	0,062	0,004
LR3	0,036	0,918	0,068	0,004
LR4	0,048	0,904	0,092	0,004
LR5	0,067	0,301	0,11	0,004
RF1	0,584	0,712	0,642	0,2
RF2	0,446	0,616	0,517	0,2
RF3	0,676	0,685	0,680	0,21
RF4	0,505	0,658	0,571	0,21
RF5	0,56	0,644	0,599	0,22
XGB1	0,769	0,685	0,725	0,012
XGB2	0,411	0,603	0,489	0,014
XGB3	0,75	0,699	0,723	0,02
XGB4	0,684	0,712	0,698	0,021
XGB5	0,787	0,658	0,716	0,013
NN1	0,44	0,603	0,509	0,056
NN2	0,188	0,698	0,297	0,056
NN3	0,396	0,521	0,45	0,056
NN4	0,583	0,479	0,526	0,056
NN5	0,527	0,397	0,453	0,06

Cuadro 4.1: Comparativa de las métricas obtenidas a partir de los modelos de machine learning.

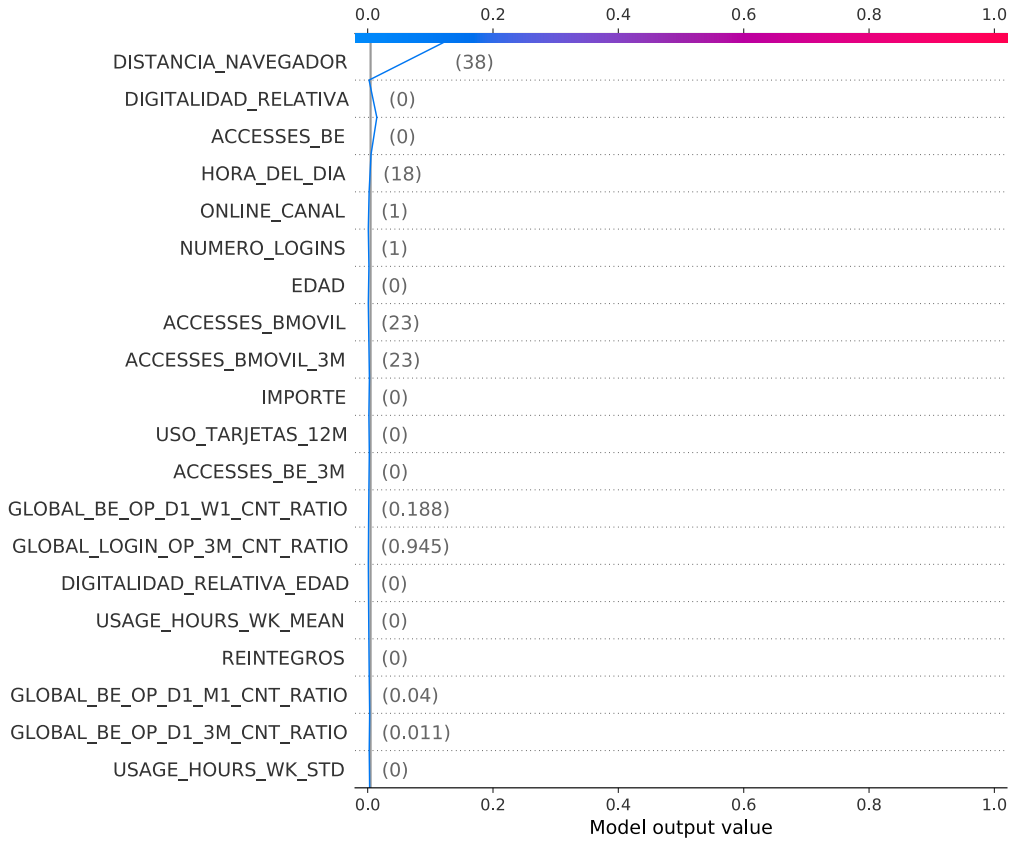

Figura 4.7: Gráfico Superior: Representación del gráfico de decisión de una observación mal clasificada. En el podemos ver el papel de cada una de las variables para llegar a dicha predicción. Gráfico Inferior: Representación del gráfico de decisión de una observación bien clasificada. Nuevamente podemos ver el papel de las diferentes variables para llegar a la predicción.

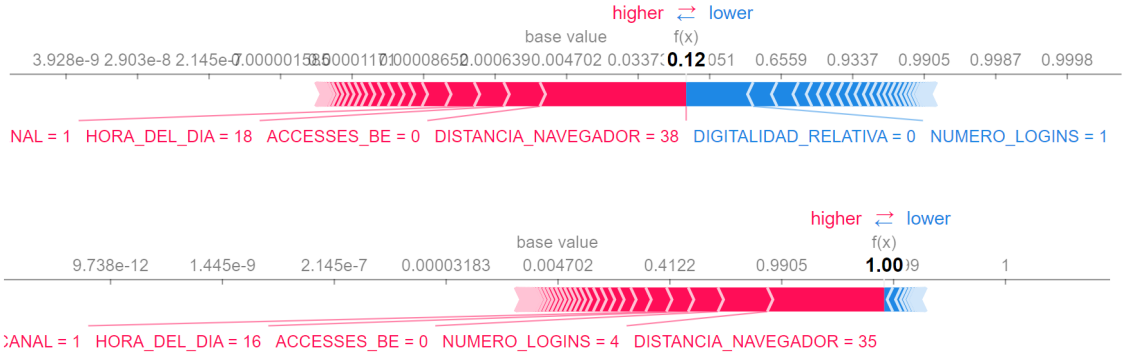

Figura 4.8: Gráfico Superior: Representación del gráfico de fuerza de una observación mal clasificada. Gráfico Inferior: Representación del gráfico de fuerza de una observación bien clasificada.

Apéndice A

Variables del problema

Dentro de las 90 variables, podemos encontrar:

- **SESION**: Código identificador de la sesión actual.
- **FECHA_COMPLETA**: Fecha y hora en la que se produzco una determinada operación.
- **CANAL**: Medio por el cual se produjo la operación (mobile/online).
- **OPERACION**: Tipo de operación (login/transaccion/sesion).
- **ETIQUETA**: Variable indicadora de si una operación es o no fraudulenta (Y/N).
- **IMPORTE**: Importe por el cual se llevo a cabo la transacción.
- **NAVEGADOR**: Navegador desde el que se realiza la transacción.
- **RECOMMENDATION**: Predicción obtenida al enfrentar este conjunto de datos con otros modelos de machine learning.
- **ID_CTA_DESTINO**: Número de cuenta al cual se mando la transacción.
- **PAIS_ORIGEN**: País desde el que se realizo la operación.
- **FECHA**: Unicamente la fecha en el que se produjo la operación.
- **USERPUID**: Usuario el cual llevo a cabo la operación.
- **TXN_W1_TXN_AMT_MEAN**: Importe medio de las transacciones de la última semana.
- **TXN_M1_TXN_AMT_MEAN**: Importe medio de las transacciones del último mes.
- **TXN_3M_TXN_AMT_MEAN**: Importe medio de las transacciones de los últimos 3 meses.
- **TXN_M1_TXN_AMT_STD**: Desviación típica de las transacciones del último mes.
- **TXN_M3_TXN_AMT_STD**: Desviación típica de las transacciones de los últimos 3 meses.
- **TXN_W1_TXN_MAX_AMT**: Máximo importe de las transacciones de la última semana.
- **TXN_M1_TXN_MAX_AMT**: Máximo importe de las transacciones del último mes.
- **TXN_3M_TXN_MAX_AMT**: Máximo importe de las transacciones de los últimos 3 meses.
- **ACCESSES_BMOVIL_W1**: Número de accesos a la banca móvil en la última semana.

- **ACCESSES_BMOVIL_M1**: Número de accesos a la banca móvil en el último mes.
- **ACCESSES_BMOVIL_3M**: Número de accesos a la banca móvil en los últimos 3 meses.
- **ACCESSES_BMOVIL**: Historial de número de accesos a la banca móvil.
- **ACCESSES_BE_W1**: Número de accesos a la banca electrónica en la última semana.
- **ACCESSES_BE_M1**: Número de accesos a la banca electrónica en el último mes.
- **ACCESSES_BE_3M**: Número de accesos a la banca electrónica en los últimos 3 meses.
- **ACCESSES_BE**: Historial de número de accesos a la banca electrónica.
- **OP_LOGIN_W1_CNT_RATIO**: Ratio de operaciones Login en la última semana.
- **OP_SESION_W1_CNT_RATIO**: Ratio de operaciones Sesión en la última semana.
- **OP_TRANSACTION_W1_CNT_RATIO**: Ratio de operaciones Transacción en la última semana.
- **OP_LOGIN_M1_CNT_RATIO**: Ratio de operaciones Login en el último mes.
- **OP_SESION_M1_CNT_RATIO**: Ratio de operaciones Sesión en el último mes.
- **OP_TRANSACTION_M1_CNT_RATIO**: Ratio de operaciones Transacción en el último mes.
- **OP_LOGIN_3M_CNT_RATIO**: Ratio de operaciones Login en los 3 meses.
- **OP_SESION_3M_CNT_RATIO**: Ratio de operaciones Sesión en los últimos 3 meses.
- **OP_TRANSACTION_3M_CNT_RATIO**: Ratio de operaciones Transacción en los últimos 3 meses .
- **ACCESSES_BMOVIL_W1_ZS**: Número de accesos a la banca móvil en la última semana normalizado.
- **ACCESSES_BMOVIL_M1_ZS**: Número de accesos a la banca móvil en el último mes normalizado.
- **ACCESSES_BMOVIL_3M_ZS**: Número de accesos a la banca móvil en los últimos 3 meses normalizado.
- **ACCESSES_BMOVIL_ZS**: Historial de número de accesos a la banca móvil normalizado.
- **ACCESSES_BE_W1_ZS**: Número de accesos a la banca electrónica en la última semana normalizado.
- **ACCESSES_BE_M1_ZS**: Número de accesos a la banca electrónica en el último mes normalizado.
- **ACCESSES_BE_3M_ZS**: Número de accesos a la banca electrónica en los últimos 3 meses normalizado.
- **ACCESSES_BE_ZS**: Historial de número de accesos a la banca electrónica normalizado.
- **GLOBAL_BMOVIL_OP_D1_W1_CNT_RATIO**: Ratio de las transacciones realizadas por el cliente (último día)/ (última semana) a través de la banca móvil.

- **GLOBAL_BMOVIL_OP_D1_M1_CNT_RATIO**: Ratio de las transacciones realizadas por el cliente (último día)/ (último mes) a través de la banca móvil.
- **GLOBAL_BMOVIL_OP_D1_3M_CNT_RATIO**: Ratio de las transacciones realizadas por el cliente (último día)/ (últimos 3 meses) a través de la banca móvil.
- **GLOBAL_BE_OP_D1_W1_CNT_RATIO**: Ratio de las transacciones realizadas por el cliente (último día)/ (última semana) a través de la banca electrónica.
- **GLOBAL_BE_OP_D1_M1_CNT_RATIO**: Ratio de las transacciones realizadas por el cliente (último día)/ (último mes) a través de la banca electrónica.
- **GLOBAL_BE_OP_D1_3M_CNT_RATIO**: Ratio de las transacciones realizadas por el cliente (último día)/ (últimos 3 meses) a través de la banca electrónica.
- **GLOBAL_LOGIN_OP_D1_CNT_RATIO**: Ratio de las operaciones Login realizados por el cliente el último día.
- **GLOBAL_TRANSACCION_OP_D1_CNT_RATIO**: Ratio de las operaciones Transacción realizados por el cliente el último día.
- **GLOBAL_LOGIN_OP_W1_CNT_RATIO**: Ratio de las operaciones Login realizados por el cliente la última semana.
- **GLOBAL_TRANSACCION_OP_W1_CNT_RATIO**: Ratio de las operaciones Transacción realizados por el cliente la última semana.
- **GLOBAL_LOGIN_OP_M1_CNT_RATIO**: Ratio de las operaciones Login realizados por el cliente el último mes.
- **GLOBAL_TRANSACCION_OP_M1_CNT_RATIO**: Ratio de las operaciones Transacción realizados por el cliente el último mes.
- **GLOBAL_LOGIN_OP_3M_CNT_RATIO**: Ratio de las operaciones Login realizados por el cliente los últimos 3 mes.
- **USAGE_HOURS_WK_MEAN**: Número medio de horas de uso a la semana.
- **USAGE_HOURS_WKND_MEAN**: Número medio de horas de uso el fin de semana.
- **USAGE_HOURS_WK_STD**: Desviación típica de las horas de uso a la semana.
- **USAGE_HOURS_WKND_STD**: Desviación típica de las horas de uso el fin de semana.
- **USAGE_WKDAY_MEAN**: Número medio de horas de uso los días laborables.
- **USAGE_WKDAY_STD**: Desviación típica de las horas de uso los días laborables.
- **USAGE_MNTHDAY_MEAN**: Número medio de horas de uso al día el último mes.
- **USAGE_MNTHDAY_STD**: Desviación típica de las horas de uso al día el último mes.
- **IN_BE**: Variable indicadora de si el usuario dispone de banca electrónica (S/N).
- **IN_BMOVIL**: Variable indicadora de si el usuario dispone de banca móvil (S/N).
- **IN_EALE**: Indicador de si el cliente tiene el servicio de alertas a móviles (S/N).
- **USO_TARJETAS_12M**: Coeficiente entre 0 y 1 el cual indica cuanto ha usado la tarjeta el usuario de la transacción.

- **PROPORCION_ELECTRONICA**: Coeficiente entre 0 y 1 el cual indica la proporción electrónica del usuario.
- **PROPORCION_TARJETAS**: Coeficiente entre 0 y 1 el cual indica la proporción de tarjetas del usuario.
- **MENSAJERIA**: Alertas a móviles o/y e-correspondencia.
- **DIGITALIDAD_RELATIVA**: Coeficiente el cual indica el grado de digitalidad del usuario.
- **EDAD**: Edad de la persona que lleva a cabo la transacción.
- **DIGITALIDAD_RELATIVA_EDAD**: Coeficiente el cual indica la digitalidad del usuario en función de su edad.
- **IN_DIGITAL**: Indicador del tipo de digitalidad del cliente.
- **REINTEGROS**: Número de reintegros en cajero y oficina realizados por el cliente en los últimos 3 meses.
- **COUNTRIES_W1_CNT**: Número total de países visitados en la última semana.
- **COUNTRIES_M1_CNT**: Número total de países visitados en el último mes.
- **COUNTRIES_SWITCH_30_MIN_W1_CNT**: Número de países desde donde los usuarios realizaron operaciones de riesgo. La cuenta de los países cambian en la actividad cada 30 minutos de duración a través de la última semana.
- **COUNTRIES_SWITCH_30_MIN_M1_CNT**: : Número de países desde donde los usuarios realizaron operaciones de riesgo. La cuenta de los países cambian en la actividad cada 30 minutos de duración a través del último mes.
- **SAME_IP_ACROSS_CANAL_W1_CNT**: Número de IPs iguales desde la que se conecta un usuario la última semana.
- **SAME_IP_ACROSS_CANAL_M1_CNT**: Número de IPs iguales desde la que se conecta un usuario el último mes.
- **COUNTRY_LIST**: Lista de países habituales de la persona que lleva a cabo la operación.
- **NAVEGADOR_LIST**: Lista de navegadores usuales de la persona que lleva a cabo la operación.

Bibliografía

Al-Serw, N. A.-R. (2021). Undersampling and oversampling: An old and a new approach. *Medium.com*.

Altini, M. (2015). Dealing with imbalanced data: Undersampling, oversampling and proper cross-validation.

Alvarez, J. M. (2018). El perceptron como neurona artificial. *Blog de Jose Mariano Alvarez*.

Bank., E. C. (2021). *Seventh report on card fraud: October 2021.* Publications Office.

Bentejac, C., Csorgő, A., and Martínez-Muñoz, G. (2020). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3):1937–1967.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Brownlee, J. (2019). A gentle introduction to the rectified linear unit (relu).

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Cohen, A. (2020). Fuzzywuzzy. *https://pypi.org/project/fuzzywuzzy/*.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. Book in preparation for MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Ho, T. K. (1995). Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, page 278, USA. IEEE Computer Society.

Joan.domenech91 (2011). Esquema de la validacion cruzada de k iteraciones.

Jordan, J. (2017). Neural networks: representation.

Kearns, M. (1988). Thoughts on hypothesis boosting. Unpublished.

Kulpati, S. (2019). A brief introduction to gans.

LTD, F. I. (2022). The actual cost of fraud.

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874.

Mcculloch, W. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147.

Neutelings, I. (September 2021). Neural networks.

Orellana, E. (2020). Smote.

Python Core Team (2019). *Python: A dynamic, open source programming language*. Python Software Foundation. Python version 3.7.