Universidad del CEMA

Maestría en Dirección de Empresas

Prevención de fraude por contracargo con un modelo de *machine learning*

Trabajo final

Autor: José Ignacio López Sáez

Tutor: ALM, CQRM Alfredo Roisenzvit

14 de octubre de 2022

Resumen

Con el crecimiento del mercado online gracias a las nuevas plataformas y tecnologías, se han hecho cada vez más sofisticados los mecanismos por los cuales los delincuentes pueden llevar adelante operaciones fraudulentas. Un caso particular es el del fraude por contracargo. En este estudio se entrenan modelos de aprendizaje automático para detectar transacciones potencialmente fraudulentas y se detalla cómo se podría implementar dentro de una arquitectura de prevención de fraude, en conjunto con otras capas adicionales. Idealmente, uno elegiría un modelo que entregue buena precisión en ambas clases a detectar: transacción genuina o fraudulenta. Sin embargo, los resultados obtenidos dejan en claro que, con los datos disponibles para entrenar los modelos ensayados, existe un compromiso a asumir entre la tasa de falsos positivos y falsos negativos a aceptar en la plataforma. Típicamente, en un comercio online, se preferirá tener una baja tasa de falsos negativos, de manera de disminuir el número de operaciones fraudulentas que se aprueben sin ser detectadas. Esto se debe a que el caso contrario, cuando las genuinas son incorrectamente etiquetadas, se pueden llevar adelante comprobaciones manuales posteriores. Entre los modelos entrenados esto se observa con métodos de ensamble (*RandomForest* y *XGBoost*) una vez que se ha aplicado un método de *sampleo* disminuyendo la cantidad de registros de la clase mayoritaria (transacciones genuinas) en el set de datos de entrenamiento. Estos métodos de sampleo suelen ser útiles para hacer frente al mayor problema al buscar soluciones de *machine learning* para detección de fraude: el desbalance de los datos, dado que se suele contar con cientos de operaciones genuinas por cada caso de fraude comprobado.

Palabras clave: machine learning, fraude, RandomForest, XGBoost, desbalance de datos

Tabla de contenido

Introducción

En el presente trabajo se ensayarán distintos modelos de aprendizaje supervisado (*machine learning*) sobre un set de datos con información de transacciones realizadas en un *Marketplace* entre los meses de febrero y mayo del 2022, con el objetivo de encontrar un algoritmo que permita identificar transacciones potencialmente fraudulentas en el futuro. Con la correcta clasificación de las transacciones entre genuinas y sospechosas, se bloquearían de antemano las segundas, disminuyendo el número de demandas por contracargo eventuales.

Primero se introducen los conceptos principales que luego permitirán entender qué se hace en el desarrollo experimental, tanto desde la parte del negocio como desde la parte tecnológica. Entonces, en primer lugar, se definirá qué es un caso de fraude por contracargo, cómo esto afecta a la industria a nivel local y regional y cuáles son los mecanismos tradicionales de prevención de fraude. Luego, partiendo de las definiciones de inteligencia artificial y aprendizaje automático y, continuando con el desarrollo de los distintos modelos existentes y cómo medir su performance, se decide qué algoritmos y métodos de *sampleo* serán los más apropiados para resolver el problema.

Por último, se presenta el desarrollo práctico, partiendo de cómo está compuesto el set de datos con el que se cuenta, cuáles son las variables de interés y cuáles son los resultados obtenidos luego de entrenar los distintos modelos. En función de esto, se hace evidente la utilización futura de un modelo de este tipo en conjunto con otros sistemas complementarios dentro de una arquitectura de prevención de fraude y cuáles serían los pasos para llevar adelante dicha implementación, así como una estrategia para el monitoreo de su performance y recalibración periódica.

Objetivo general

El objetivo del estudio se centrará en entrenar modelos de aprendizaje automático que permitan identificar transacciones potencialmente fraudulentas dentro de un universo de operaciones de compra de un comercio online. Luego decidir, a la luz de los resultados obtenidos, cuál es el modelo más apropiado para implementar y poder, finalmente, sugerir una arquitectura para llevar adelante dicha implementación futura.

1. Definiciones

La Real Academia Española define fraude como toda acción contraria a la verdad y a la rectitud que perjudica a la persona contra la que se comete (Real Academia Española, s.f.). Por su parte, el diccionario de Oxford agrega el hecho de que quien lo comete, estará generalmente buscando un rédito económico (Oxford University Press, s.f.). En esa línea, en los últimos años, y con el advenimiento de nuevas tecnologías y plataformas transaccionales se han expandido las posibilidades en las que los usuarios pueden cometer fraude. En el caso del comercio electrónico, podremos distinguir distintos tipos de fraude: fraude con tarjeta de crédito y fraude por contracargo.

1.1. Fraude con tarjeta de crédito

Se define como cualquier delito en el cual intervengan personas con el fin de obtener y utilizar el plástico sin la autorización del dueño legítimo y en beneficio propio. Por ejemplo, cargando compras en la cuenta de la víctima o buscando extraer fondos (Interbank Perú, s.f.). La Escuela de Leyes de Cornell expande esta definición aduciendo que se trata de un tipo de robo de identidad y clasifica los sistemas de fraude con tarjeta de crédito en dos categorías. En el primer caso, el victimario tiene suficiente información de la víctima para poder completar una solicitud de tarjeta de crédito a su nombre; el segundo consiste en la duplicación de una determinada tarjeta de crédito. Por ejemplo, alguien tiene acceso a la información de un plástico: nombre, vencimiento, CVV, etc., y lo utiliza como si fuera el dueño legítimo (Cornell Law School, s.f.).

Interbank Perú incorpora otra clasificación para estos delitos:

- Fraudes con tarjeta presente: el delincuente utiliza el plástico para realizar el fraude.

  Bolton & Hand ofrecen una apertura adicional dentro de esta clasificación:

- Tarjeta robada: en estos casos, el perpetrador intentará gastar el máximo posible de dinero utilizando la tarjeta robada en un intervalo corto de tiempo antes de que el plástico sea denunciado y el crédito cortado (Bolton & Hand, 2022).
- Fraude de aplicación: el perpetrador intentará obtener una tarjeta de crédito utilizando información personal falsa (Bolton & Hand, 2022).

- Fraudes con tarjeta no presente: el delincuente utiliza parte de la información contenida en la tarjeta. En este segundo grupo encontraremos la gran mayoría de los casos de fraude en comercio electrónico.

1.2. Fraude por contracargo

Otro tipo de fraude habitual en las transacciones electrónicas es el fraude por contracargo (o *chargebacks*). Este constituirá el problema principal a atacar en este trabajo dado que es el que más suele afectar a las plataformas de *e-commerce*. La empresa Shopify, en su blog online, define a los *chargebacks* como el resultado de que un cliente cuestione o dispute una transacción contra el banco emisor de la tarjeta (Shopify, 2016). En otras palabras, el titular, real o presunto, de la tarjeta solicita la devolución del dinero de una compra online. Entre las posibles motivaciones, distingue los siguientes tipos:

i. Contracargos fraudulentos: el titular de la tarjeta no reconoce una transacción en su resumen y entiende que su cuenta o tarjeta ha sido comprometida.
ii. Contracargos no reconocidos: mismo caso que el anterior, con la salvedad de que el titular no considera que su cuenta ha sido vulnerada.

En ambos casos se pueden dar casos de “contracargos amistosos” en donde el titular no reconoce una transacción, pero simplemente porque no recuerda haber hecho esa operación.

iii. Contracargos por suscripciones canceladas: cualquiera de los casos anteriores, pero afectando particularmente a aquellos comercios o vendedores que facturan por un servicio con un abono recurrente. Por ejemplo, el titular de la tarjeta aduce que ya ha cancelado una suscripción y que, sin embargo, continúa recibiendo el cargo en forma recurrente.

iv. Contracargos por producto no recibido: como su nombre lo indica, el comprador inicia la disputa cuando sostiene que no ha recibido el producto que ha comprado en forma online.

v. Contracargos por producto no aceptado: se da cuando el cliente entiende que el producto recibido dista mucho de las condiciones esperadas: baja calidad, defectuoso, dañado o alejado de la descripción provista por el vendedor, entre otros.

Adicionalmente, la plataforma argentina Tiendanube presenta el proceso típico en que un usuario, malintencionadamente o no, puede intentar realizar fraude por contracargo contra un vendedor (Tiendanube, 2022):

1. El usuario efectúa una compra online con tarjeta de crédito
2. Recibido el producto, se pone en contacto con el banco emisor del plástico desconociendo el cargo.
3. El banco inicia la investigación pertinente. Para ello solicitará a las partes involucradas determinada información: recibos de compra, recibos de la empresa a cargo del envío o logística, las políticas de envío, entre otras. El comercio tiene un plazo específico para poder presentar esto.
4. El banco analiza la información y toma una decisión. Si considera que la tienda online presenta alguna irregularidad, fallará a favor del estafador, quitando el dinero de la transacción de la tienda y sumando una penalidad o cargo extra. Además de ello, se ha perdido el producto, se han pagado costos logísticos, comisiones, etc.

2. Contexto local, regional e internacional

Los contracargos fueron pensados originalmente como una manera de proteger a los titulares de tarjetas de posibles cargos que se pudieran realizar sin su aval. Sin embargo, para poner en números el problema que este tipo de operaciones cuando se hacen con fines maliciosos pueden representar para los comercios, el *Reporte de fraude online para América Latina 2017* publicado por Visa Cybersource, especifica que, para la región, el índice de contracargos es del orden del 1,7% de las ventas, a la vez que un 9,2% de las operaciones se rechazan por sospecha de fraude (Visa Cybersource, 2017). Si bien estos números resultan de una encuesta realizada a 266 empresas, sirven para contextualizar la problemática. A su vez, estas dos variables se pueden analizar por país, de acuerdo con el mismo estudio:

Figura 1 - Tasa de contracargo y % de ventas marcadas como fraudulentas (Visa CyberSource, 2017)

Esto quiere decir que se sacrifican potenciales compras genuinas (9,2% de la facturación en América Latina) buscando evitar la concreción de fraude (1,7% de las ventas en la misma región). Los números hacen evidente la sobrerreacción por parte de la industria a nivel regional. Estos 7,5 puntos porcentuales de diferencia se transforman, por consiguiente, en el incentivo primario en la búsqueda de mejores modelos de prevención de fraude, en donde no sólo se logren bloquear las transacciones potencialmente fraudulentas sino también en donde la cantidad de transacciones genuinas bloqueadas sea la más baja posible.

Evaluando el caso particular de México, contamos con estadísticas oficiales provistas por el Banco de México y recolectadas por la Comisión Nacional para la Protección y la Defensa de los Usuarios de Servicios Financieros (CONDUSEF): en el año 2021, el 20% del total de los pagos realizados fueron hechos en comercios online. De ellos, el 32% son enviados para obtener autorización (se le pide al comprador que se ponga en contacto con el banco para validar la operación) y el 63% de los mismos son aprobados. Así y todo, el 0,47% de estos últimos terminaron como contracargos (Gobierno de México, 2022). Con estas mismas estadísticas se puede reconstruir la evolución del 2015 al 2021:

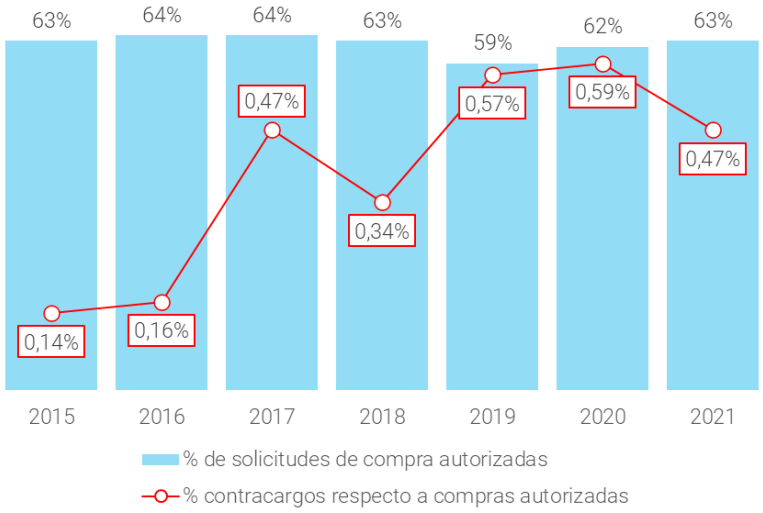


Figura 2 - % de operaciones aprobadas e índice de contracargos para México (Elaboración propia a partir de datos del Gobierno de México)

En el mismo período, el monto de compras enviado para su autorización se multiplicó por diez, lo que da cuenta del crecimiento del *e-commerce* en dicho país (situación que podemos asumir extrapolable al resto de la región). Por su parte, la empresa de servicios logísticos mexicana Cubbo agrega, sobre la base de la misma fuente, que en el año 2021 fueron reportados 2,7 millones de reclamos por fraude online y solo el 16% se resolvió a favor del comercio (Cubbo, s.f.). Esto último parecería indicar que, aunque los vendedores online o las plataformas puedan presentar la documentación pertinente que acredite la transacción, es más probable que la demanda se incline a favor del comprador, sea el reclamo por contracargo legítimo o no.

Datos a nivel global publicados por Daniela Coppola, Senior Researcher en Statista, muestran que en 2021 las pérdidas a nivel global debido a fraude en pagos electrónicos en *e-commerce* sumaron 20Bn USD, cifra 14% superior a la del año 2020 (Statista, 2022). En el mismo reporte, se especifica al impacto global de la pandemia de Covid-19 como un *driver* importante para dicho valor y un aumento sobre el año previo.

Ayat Shukairy, Co-Founder en Invesp, compañía especializada en la optimización de la conversión de campañas de marketing online, compila en su blog las siguientes estadísticas que, aunque corresponden al año 2016, permiten echar luz sobre el tema y detallar más el problema de los contracargos en la industria (Shukairy, 2022):

- Pérdidas por contracargos en comercio electrónico 2016 = 6,7Bn USD
- Por cada dólar de fraude, las empresas pierden 2,4usd.
- 58% de los tarjetahabientes no contacta al comercio, sino que sigue la disputa exclusivamente con el banco.
- 40% de los usuarios que llevan adelante una demanda por contracargo reincide antes de los 60 días y el 50%, antes de los 90.

A nivel global, el número de usuarios de tarjetas VISA y MasterCard superaba los 2 mil millones en el año 2020. A su vez, el creciente número de operaciones con tarjeta resulta en una tendencia al alza de la tasa de fraude para aquellos del tipo tarjeta no presente y la coyuntura particular de la pandemia por COVID entre 2020 y 2021 ha volcado a los usuarios y comerciantes a la utilización masiva de medios electrónicos para realizar los pagos (Faraji, 2022). PriceWaterhouseCoopers refuerza esto exhibiendo los resultados de una encuesta: la industria tecnológica, en sentido contrario al resto, ha experimentado un incremento en actividad fraudulenta desde el 2020 al presente. En cuanto al tipo de fraude, aquel relacionado al incurrido por los clientes (*customer fraud*) explica en torno al 30% del total sin importar el tamaño de las compañías, siendo el segundo factor de riesgo por debajo de crímenes

cibernéticos (PwC, 2022). Desglosado por tipo de empresa, sin embargo, tanto en *retail* como en servicios financieros, *customer fraud* es la causa primera de fraude.

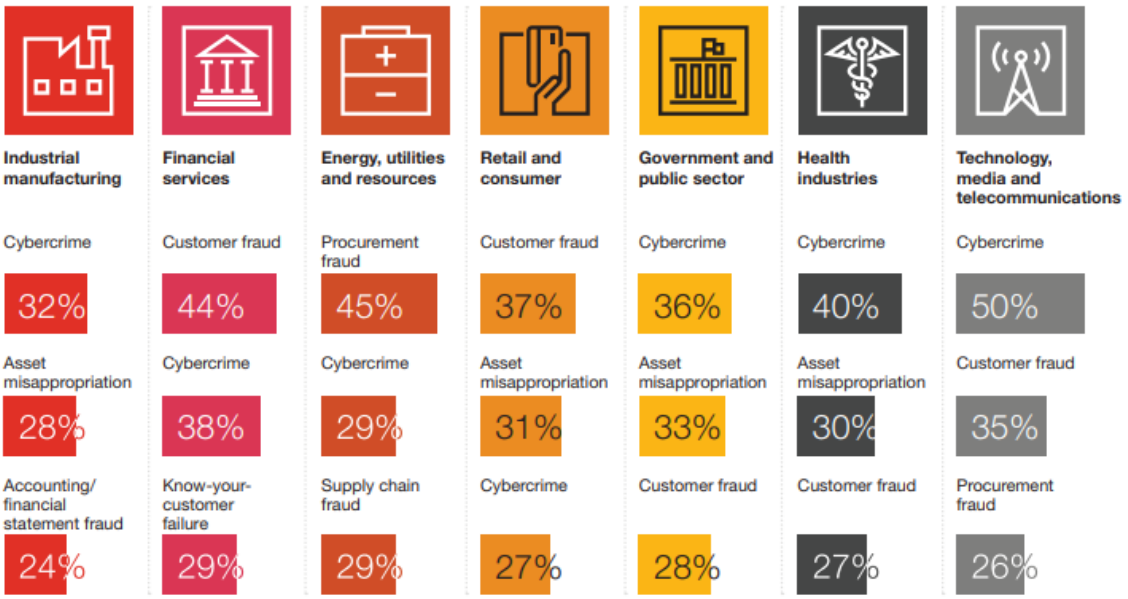


Figura 3 - Tipos de fraude por industria (PwC, 2022)

Es importante aclarar que hay costos que no son considerados ni mensurados, ni en este caso ni en los estudios anteriores, que corresponden a las horas hombre puestas a analizar los distintos casos, así como a recolectar toda la evidencia para responder a los reclamos.

3. Predicción y prevención de fraude

Un primer acercamiento a la prevención de fraude consistiría en ser capaces de revisar cada una de las transacciones realizadas en la plataforma y, mediante la evaluación de ciertas características (como ser el historial del usuario, el historial del vendedor, el método de pago seleccionado, el monto de la transacción, etc.), determinar si la misma es probable que se trate de un intento de fraude y bloquearla o, caso contrario, aprobarla. Esto requeriría de un nivel de recursos muy alto, incluso si se decidiera evaluar únicamente una pequeña fracción del volumen de transacciones a modo de muestra estadística elegida aleatoriamente. Por ejemplo, solo MercadoLibre, con sus operaciones en Latinoamérica, vendió más de 1000 millones de productos en el año 2021 (Compte, 2022). Si sólo se decidiera analizar el 1% de

dichas operaciones, estaríamos hablando de más de 1.100 operaciones por hora para ser revisadas en forma manual por un grupo de agentes.

3.1. Inteligencia artificial

La inteligencia artificial (IA) es la ciencia e ingeniería de hacer máquinas inteligentes y, en particular, programas y piezas de software inteligentes (McCarthy, 2004). Su desarrollo permite disponer de computadoras realizando tareas que previamente requerían total o parcialmente de operación manual, posibilitando así que los tomadores de decisión dentro de las compañías se puedan dedicar en mayor medida a la estrategia de negocios (Li, 2022). Por su parte, Russel y Norvig dividen a las definiciones, y sus autores, de la IA en dos clases y cuatro categorías de acuerdo con si se apunta a la forma de pensar o actuar, sea como los seres humanos o en forma racional, independientemente de lo anterior (Russel & Norvig, 2009):

I. Enfoque humano
    a. Pensando humanamente
        i. "El nuevo y emocionante esfuerzo para hacer que las computadoras piensen... máquinas con mente, en el sentido completo y literal" (Haugeland, 1985)
        ii. "[La automatización de] actividades que asociamos con el pensamiento humano, actividades como la toma de decisiones, la resolución de problemas, el aprendizaje..." (Bellman, 1978)
    b. Actuando humanamente
        i. "El arte de crear máquinas que realizan funciones que requieren inteligencia cuando son realizadas por personas" (Kurzweil, 1990)

ii. "El estudio de cómo hacer que las computadoras hagan cosas en las que, en este momento, las personas son mejores" (Rich y Knight, 1991)

II. Enfoque ideal

a. Pensando racionalmente

i. "El estudio de las facultades mentales mediante el uso de modelos computacionales" (Charniak y McDermott, 1985)

ii. "El estudio de los cálculos que hacen posible percibir, razonar y actuar" (Winston, 1992)

b. Actuando racionalmente

i. "La Inteligencia Computacional es el estudio del diseño de agentes inteligentes" (Poole et al., 1998)

ii. "IA... se preocupa por el comportamiento inteligente en los artefactos" (Nilsson, 1998)

De una u otra manera, todas las definiciones presentadas apuntan en el mismo sentido: dedicar el tiempo que otrora se dedicara a tareas manuales y disponerlo para actividades en donde las personas puedan aportar mayor valor dentro de la compañía, al contar con sistemas que automaticen lo anterior y nos entreguen resultados consistentes. La inteligencia artificial, entonces, combina la informática y conjuntos de datos sólidos y robustos para permitir la resolución de problemas, tal como lo define IBM en su centro de aprendizaje online (IBM, 2020). Por último, y no menos importante, la inteligencia artificial hace posible, además, llevar adelante tareas que, por volumen de información y velocidad de procesamiento, serían imposibles de ser desarrolladas por personas.

3.2. Machine learning

La inteligencia artificial incluye dos subcampos principales:

- Aprendizaje automático, o *machine learning*
- Aprendizaje profundo, o *deep-learning*

A su vez, el aprendizaje profundo es un subcampo del aprendizaje automático:

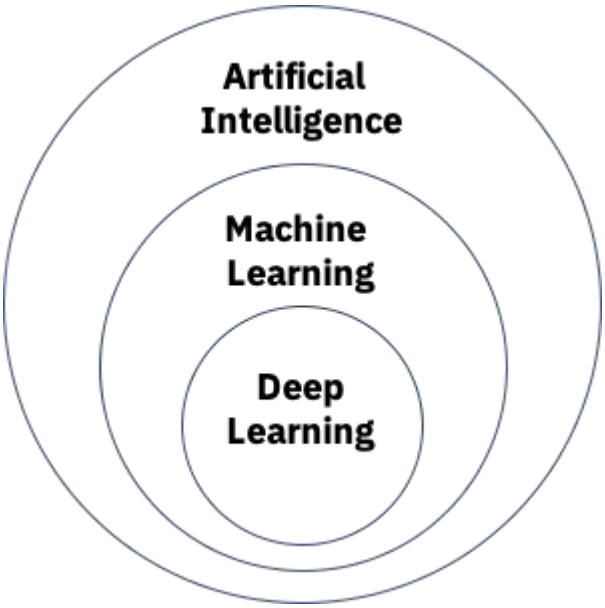


Figura 4 - IA, Machine Learning y Deep Learning (IBM, 2020)

En ambos casos se trata de algoritmos que, a partir de datos de entrada (*input*), generan un valor de salida (*output*), que se puede tratar de una predicción (por ejemplo, estimar el precio de un activo a partir de su cotización histórica) o de una clasificación (por ejemplo, determinar si un paciente posee una enfermedad determinada a partir de la evaluación de sus datos biométricos y otras variables pertinentes). Eda Kavlakoglu, Program Manager en IBM, define que la diferencia entre los dos radica en la forma en que el algoritmo aprende: en el primer caso (aprendizaje automático) es parcialmente dependiente de la intervención humana, requiriendo generalmente datos más estructurados y la determinación del conjunto de características que permiten la diferenciación entre los datos de entrada. En el segundo, se pueden incorporar al modelo datos crudos y no estructurados y dejar que el

algoritmo aprenda y determine el set de características necesarias para lograr el output deseado (Kavlakoglu, 2020).

Una definición adicional es qué implica o qué significa que un agente o algoritmo "aprenda". En ese sentido, un agente estará aprendiendo siempre que logre mejorar su desempeño y performance en tareas futuras luego de hacer observaciones acerca del mundo (Russel & Norvig, 2009). La forma en que los algoritmos aprenden se clasifica en dos grandes grupos, de acuerdo con el artículo del blog de IBM redactado por Julianna DeLua, Data Science, Data Management and AI (Delua, 2021):

- Aprendizaje supervisado: se define por el uso de conjuntos de datos etiquetados. Es decir, una de las variables que se introducen en el modelo es la que se intentará predecir en el futuro. Por ello recibe dicho nombre: al algoritmo ya se le está diciendo cómo ha sido el output esperado en el pasado en conjunto con las demás variables del modelo. El proceso consistirá en generar predicciones en forma iterativa, comparándola con la etiqueta original y ajustando así las respuestas. Estos algoritmos se pueden subdividir en dos grupos de acuerdo con el tipo de problema a resolver:
    - Clasificación: la salida esperada corresponde con una categoría específica, que puede ser binaria o de múltiples niveles. Ejemplos de este tipo de algoritmos: clasificadores lineales, árboles de decisión, máquinas de soporte vectorial (SVM), bosques aleatorios, k-vecinos cercanos (KNN), entre otros.
    - Regresión: la salida es el resultado de la relación entre las variables dependientes e independientes del modelo. Ejemplos de este tipo de algoritmos: regresión lineal, polinómica y logística.

A su vez, algunos de estos algoritmos, tales como los árboles de decisión o los algoritmos de regresión logística pueden ser utilizados para resolver ambos tipos de problemas.

- Aprendizaje no supervisado: en este caso, se utilizan algoritmos para analizar los datos y agruparlos en conjuntos sin que estos hubieran sido previamente definidos o etiquetados. Ahí radica la diferencia con los anteriores: en el set de datos no existe la variable que se espera predecir en el futuro, sino que lo que se busca es que logren descubrir ciertos patrones subyacentes en los registros sin que intervenga una persona (al menos no a priori, aunque sí generalmente a posteriori para validad los resultados). Estos algoritmos se utilizan para resolver los siguientes tipos de problemas:
    - *Clustering*: se agrupan datos en función de sus similitudes. Una aplicación típica es dividir o clusterizar una base de datos de clientes de forma de encontrar subgrupos con características similares y segmentar mejor una estrategia de marketing, en contraposición con agrupaciones típicas: rango etario, género, localidad, etc.
    - Asociación: se utilizan distintas reglas para encontrar relaciones entre variables en un conjunto de datos. Por ejemplo, estamos en presencia de un algoritmo de esta naturaleza cada vez que, en un comercio online, se nos sugiere otro producto en función del que acabamos de incorporar a nuestro canasto.
    - Reducción de dimensionalidad: el objetivo es reducir la cantidad de variables de entrada, pero asegurando la integridad de los datos. Muchas veces se utilizan estos algoritmos para el tratamiento de los datos previo a correr un modelo de clasificación. Por ejemplo, buscan disminuir la complejidad de la información ingresada y así mejorar su rendimiento. Adicional a estas definiciones de IBM, Tony Yiu, Director Quant R&D y Data Science en Nasdaq, define algo usual a lo que se suele hacer mención, que es la “maldición de la dimensionalidad”: si tenemos más variables que registros correremos el peligro de caer en un problema de *overfitting*, a la vez que las observaciones

resultan más difíciles de agrupar o *clusterizar* (Yiu, 2019). Un algoritmo típico para resolver este tipo de problemas es el análisis de componentes principales (PCA).

La elección de un tipo de aprendizaje u otro radicará principalmente en el objetivo que se tenga: utilizaremos los de tipo supervisado siempre que queramos predecir el *output* para nuevos datos, mientras que los segundos se utilizarán cuando busquemos obtener *insights* a partir de los datos.

Shweta Bhatt, *Google Developer Expert for Machine Learning*, define un tercer tipo, denominado Aprendizaje por Refuerzo (*Reinforced Learning*), siendo la diferencia principal con el aprendizaje supervisado en que el algoritmo no corrige iteración tras iteración sus resultados en función de la variable provista, sino que se disponen una serie de premios y castigos a partir de los resultados obtenidos (Bhatt, 2018). Es decir, al agente no se le dice de antemano el resultado a obtener, sino que el objetivo es dejar que aprenda en forma autónoma buscando maximizar una señal numérica. En otras palabras, el aprendizaje se basa en la interacción entre el agente y el entorno (Sutton & Barto, 2018). Un algoritmo de este tipo que cobró gran dimensión mediática fue AlphaGo, desarrollado por Google Deepmind, que fue el primer software que logró derrotar a un jugador profesional en una partida de Go. En primer lugar, se le introdujeron al modelo varias partidas amateurs para que lograra aprender y entender la forma humana en que se desenvuelve una partida. Luego se lo hizo jugar miles de veces contra sí mismo, aprendiendo de sus errores (DeepMind, s.f.). Una versión posterior (AlphaGo Zero) aprendió directamente jugando contra sí misma, comenzando con partidas completamente aleatorias, y buscando la forma de maximizar esa señal de premio o recompensa obtenida por ganar una partida.

3.2.1. Usos en detección de fraude

En el paper *An Analysis of the Most Used Machine Learning Algorithms for online fraud detection*, los autores recopilan los tipos de fraude analizados, así como el modelo utilizado

para su análisis a partir de numerosos artículos publicados entre el 2010 y el 2019 (las fuentes consultadas fueron obtenidas de bancos reconocidos, tales como ACM Digital Library, IEEE Xplorer Digital Library, Science Direct, entre otros) (Minastireanu & Mesnita, 2019). Los resultados principales se aprecian en la Figura 5:

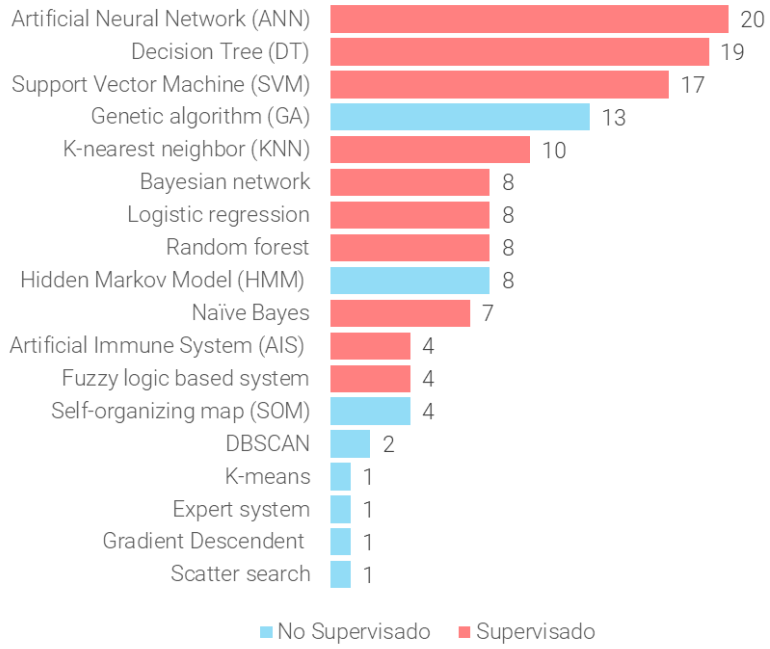


Figura 5 - Algoritmos más utilizados en problemas de detección de fraude de acuerdo con la literatura (elaboración propia a partir de Minastireanu & Mesnita, 2019)

Una vista rápida a estos datos, divididos en los dos tipos de aprendizaje principales, permite una primera conclusión: los algoritmos más usuales para resolver problemas de detección de fraude son aquellos de tipo supervisado. Y esto es consecuencia del objetivo que suele diferenciar a un tipo de aprendizaje de otro: buscamos predecir una variable de salida (por ejemplo: transacción fraudulenta / transacción genuina) en función de variables de entrada (por ejemplo: dimensiones que describan al comprador, al vendedor, al tipo de producto, su precio, tipo y método de pago, etc.); mientras que, en los segundos, buscamos encontrar patrones subyacentes en los datos.

Sin embargo, también se pueden aprovechar las características de algunos algoritmos de aprendizaje no supervisado en la detección de fraude, sean usados como algoritmo principal o como una instancia previa a la implementación de uno supervisado. Por ejemplo,

Lieke Hamelers, de la Universidad de Twente presenta, una aplicación del algoritmo *Isolation Forest* (desarrollado por Fei Tony Liu como parte de su trabajo de PhD en el año 2009), para la detección de anomalías (*outliers*) en datos de recibos financieros (Hamelers, 2021). Se entiende por anomalía a todo aquel patrón dentro de los datos que presenta diferentes características en contraposición con las instancias normales y que, además, suelen ser pocas y diferentes. En el caso particular de transacciones con tarjeta de crédito, las mismas pueden ser síntoma de fraude. El uso de otro algoritmo de aprendizaje no supervisado, tal como métodos de *clustering* suelen presentar dos problemas principales para lograr este objetivo: están optimizados para perfilar instancias normales, pero no para detectar anomalías (generando muchos falsos positivos) y están restringidos a sets de datos de baja dimensionalidad (Liu, Ting, & Zhou, 2009). En el caso de aplicación de la tesis de Hamelers, se contaba con información de recibos financieros que debían ser auditados y que, por supuesto, no estaban previamente etiquetados de forma de poder diferenciar aquellos genuinos de los fraudulentos, lo que hubiera hecho posible entrenar un algoritmo del primer tipo. En consecuencia, el uso de modelos como *Isolation Forest* se presenta como ideal, dado que encontrarán anomalías o *oultiers* dentro de todos los datos disponibles, separando aquellos potencialmente fraudulentos y disminuyendo drásticamente el universo de recibos para controlar por parte de los auditores (Hamelers, 2021). El concepto subyacente en el algoritmo de *Isolation Forest* radica en que, si pudiéramos generar un árbol de decisión tal que en cada hoja terminal del mismo contuviéramos un único registro, entonces las anomalías se deberían encontrar más cercanas a la raíz del árbol en comparación a las instancias normales (Mazzanti, 2021).

Por otro lado, de los resultados de la Figura 5, el modelo más utilizado en la literatura para aquellos del tipo no supervisado es el de GA (*Genetic Algorithm*), que fue desarrollado inspirado a partir de la teoría de la evolución de Darwin y consiste, principalmente, en un algoritmo de búsqueda basado en la mecánica de la selección y genética natural. En otras palabras, es un modelo que, utilizando operadores de selección y recombinación, genera

nuevos puntos de muestra en un espacio de búsqueda (que es el set de todas las posibles soluciones a un problema dado) (Benchaji, Douzi, & Ouahidi, 2019). En el caso particular de la detección de fraude, el algoritmo comienza con una población de transacciones fraudulentas que son codificadas a una representación determinada (binaria, por ejemplo) y se van generando nuevos registros combinando dos operaciones posibles: *crossover* y *mutation*. En la nueva generación, se descartan las muestras inválidas (evaluadas a partir de una función, denominada *fitness function*) y se itera continuamente hasta que se alcanza la condición de terminación prestablecida. Benchaji, Douzi, & Ouahidi utilizan este algoritmo para generar nuevos casos sintéticos de la población minoritaria (transacciones fraudulentas) dentro de un universo de casos a evaluar, como una instancia previa a utilizar un algoritmo de aprendizaje supervisado para poder clasificar nuevas operaciones en el futuro, buscando tener un dataset balanceado.

Estos dos ejemplos anteriores permiten dar cuenta de que, si bien en el caso de detección de fraude, típicamente un problema de clasificación, los algoritmos más ampliamente utilizados son aquellos del tipo aprendizaje supervisado, otros del tipo no supervisado también pueden ser usados. En algunos casos, como una solución integral para detectar transacciones anómalas dentro del universo de las operaciones y en otros, como en una instancia previa para tratar los datos, buscando solucionar el problema del desbalance (muy pocos casos de fraude dentro un gran número de operaciones genuinas) previo a implementar un modelo de *machine learning*.

3.2.2. Machine Learning vs. Sistemas basados en reglas

El enfoque tradicional llevado adelante por las compañías para detectar y prevenir fraude se basa en la aplicación de un conjunto de reglas del tipo *if…else* en donde, en función de casos pasados, se busca bloquear transacciones u operaciones potencialmente fraudulentas en el futuro o separarlas para revisión manual. Consiste, por lo tanto, en el uso de correlaciones estadísticas y comparaciones lógicas para intentar la detección de actos de

fraude, utilizando métodos tradicionales de análisis de datos y requiriendo investigaciones complejas y costosas en tiempo (Kount, s.f.). En cuanto a necesidad de información, es igual que si quisiéramos implementar un algoritmo de aprendizaje automático supervisado (dado que necesitamos contar con casos etiquetados: fraude / no-fraude. Nethone, empresa de origen polaco dedicada a la prevención de fraude online, define en un artículo en su web que este set de reglas se puede establecer siguiendo dos tipos de lineamientos (Karczewski, s.f.):

- Siguiendo las mejores prácticas en la industria. Ejemplos de esto sería bloquear múltiples transacciones de una misma cuenta que se efectuaron con una alta velocidad, aquellas operaciones que se hacen a través de una red virtual privada (VPN) o desde ubicaciones o localidades consideradas riesgosas.
- Analizando casos de fraude positivo y desarrollando nuevas reglas en el intento de cubrir nuevas características que harían que una transacción fuera sospechosa.

Siguiendo estas definiciones, el conjunto de reglas que se pongan a funcionar dentro del túnel de aprobación de un comercio suele ser sencillo de interpretar por cualquier persona. Es decir, dado que se codifican siguiendo esa estructura sencilla ("si se cumple {condición}, entonces {decisión}") su seguimiento, interpretación y análisis es lineal. Por otro lado, intentan replicar la forma en que un ser humano procesaría la información. Estas características hacen que aún hoy en día sean muy utilizadas en la industria. La compañía Decerto, también de origen polaco y desarrolladora de Hyperon, un BRMS (*Business Rules Management System*), define los siguientes beneficios de utilizar un sistema basado en reglas duras (Grbovic, 2022):

- Eficiencia: implementación sencilla y configuración rápida. No es necesario contar con un equipo de técnicos (analistas / ingenieros / científicos de datos) sino que los mismos expertos en el negocio las puedan diseñar e implementar.
- Simplicidad: son reglas fácilmente interpretables
- Respuesta inmediata frente a amenazas: se pueden desarrollar e insertar nuevas reglas una vez que una nueva tendencia en fraude aparece en el mercado.

Por su parte, Ravelin, empresa inglesa que proporciona servicios de prevención de fraude y pagos online, señala los siguientes problemas a la hora de implementar únicamente un sistema basado en reglas (Ravelin, s.f.):

- Alto número de falsos positivos: el utilizar un gran número de reglas conlleva una alta tasa de falsos positivos, que se traduce en una baja conversión: se bloquean muchas operaciones genuinas con tal de bloquear unas pocas fraudulentas. Adicionalmente, muchas reglas se suelen diseñar en función del monto de la transacción, haciendo que las pérdidas por baja conversión sean significativas.
- Resultados fijos: hay que estar permanentemente contrastando los límites (*thresholds*) establecidos dado que el accionar de los ataques fraudulentos varía con el tiempo. En un contexto de alta inflación, si el umbral está establecido en moneda local, pierde validez rápidamente.
- Difícil de escalar: cuando el volumen de transacciones aumenta, también lo hará la cantidad de operaciones a revisar en forma manual, a la vez que será difícil de mantener el sistema de reglas actualizado continuamente: nuevas reglas deben ser desarrolladas en la medida en que nuevos patrones de fraude aparecen.

Los modelos de *machine learning* parecen sortear los problemas que presentan los sistemas basados en reglas, pero el enfoque es diferente: mientras en el segundo caso se requiere mucha *expertise* del negocio para poder entender los patrones y diseñar las reglas en consecuencia; con los modelos de aprendizaje automático, lo más importante es contar con un set de datos de calidad, en donde cada transacción pasada esté correctamente etiquetada y de la cual dispongamos toda la información posible para describirla. Cuanto mayor sea el volumen de información, mejor, tanto en cantidad de registros como en término de dimensiones. En contrapartida, en el caso del sistema de reglas, el aumento del volumen lo hace cada vez más complejo. Al aplicar un modelo de *machine learning* no es necesario (sí deseable) entender exactamente el fenómeno de negocio que subyace. La característica primordial y por la que las

compañías los quieren implementar es por la posibilidad de reducir en gran medida tareas manuales y repetitivas, dedicando los recursos a la investigación y desarrollo de nuevas características y algoritmos.

Con lo expuesto, se puede concluir que el acercamiento óptimo a la problemática implica un uso complementario de las distintas técnicas: un set de reglas que pueda ir etiquetando las transacciones en forma dinámica, de rápido y fácil mantenimiento e interpretabilidad; y un algoritmo de *machine learning* que logre captar comportamientos fraudulentos subyacentes que escapan al análisis tradicional y logre mejorar la precisión y resultados, disminuyendo tanto los falsos positivos (buscando mayor conversión) como los falsos negativos (buscando menor costo por fraude y contracargos). No son técnicas mutuamente excluyentes y, en todo caso, el modelo de aprendizaje automático estaría funcionando como una "regla" adicional dentro de nuestro sistema.

3.3. Selección de los algoritmos a implementar

Minastireanu & Mesnita, ofrecen un ordenamiento de los resultados obtenidos en los distintos artículos estudiados en término de tres dimensiones: cantidad de aciertos (*accuracy*), cobertura ante el fraude y eficiencia de costos; asignando valor 1 en el caso de resultados bajos y 3 en el caso contrario:

Algoritmo de ML	**Cantidad de artículos**	**Accuracy**	**Coverage**	**Costs**
Artificial Neural Network (ANN)	20	2	2	3
Decision Tree (DT)	19	2	2	3
Support Vector Machine (SVM)	17	3	3	3
K-nearest neighbor (KNN)	10	2	2	3
Bayesian network	8	3	2	3
Logistic regression	8	3	2	2
Random forest	8	3	2	2
Naïve Bayes	7	2	2	3
Artificial Immune System (AIS)	4	2	3	1
Fuzzy logic based system	4	3	2	3

Tabla 1 - Resultados obtenidos para modelos de ML de aprendizaje supervisado en aplicación de detección de fraude (Minastireanu & Mesnita, 2019)

Concentrándonos únicamente en los algoritmos de aprendizaje supervisado, los resultados son consistentes entre los distintos modelos utilizados, a excepción del *Artificial Immune System* (AIS) que parece incurrir en un uso más eficiente de los recursos.

Por otra parte, en el ensayo *Machine Learning Model for Credit Card Fraud Detection - A Comparative Analysis*, los autores también recopilan información de distintos artículos y modelos puestos a prueba para evaluar transacciones y detectar fraude, a la vez que ensayan distintos algoritmos sobre un dataset propio y comparan resultados. Para esto último, ensayan los siguientes modelos: Regresión Logística, Máquina de soporte vectorial (SVM), *Random Forest* y *Artificial Neural Network* (ANN). Los resultados que obtienen muestran que, en términos de aciertos, todos consiguen buenos resultados (>94% de aciertos en positivos y negativos) pero la regresión logística lo hace significativamente peor comparando métricas adicionales (*Precision*, *Recall* y *F1 Score*). Finalmente, y acompañado por otros resultados de la literatura, los autores recomiendan *Random Forest* y ANN por sobre los demás modelos (Sharma, Banerjee, Tiwari, & Patni, 2021).

Con esto en consideración, los algoritmos elegidos para el desarrollo de este trabajo serán los siguientes:

- Árbol de decisión (DT): por su fácil interpretabilidad, suelen sentar una buena base de comparación.
- Random Forest (RF)
- Artificial Neural Network (ANN)

En función de los resultados obtenidos con ellos, y que serán discutidos oportunamente, se evaluará ensayar algoritmos adicionales. En dicho caso, la regresión logística (RL) y las máquinas de soporte vectorial (SVM) son los candidatos para considerar.

Nota: Para la definición de cada uno de los algoritmos a utilizar, se utilizará como fuente principal la guía de *Analytics* de IBM (IBM, s.f.). Fuentes adicionales para complementar la información se irán detallando según corresponda.

3.3.1. Árbol de decisión (DT)

Un árbol de decisión es un algoritmo de aprendizaje supervisado no paramétrico. Es decir que, a diferencia de un algoritmo paramétrico, no hace ninguna presunción acerca de la distribución de la población de la cual fue tomada la muestra. Consiste en una estructura jerárquica que, a partir de un nodo raíz, se va particionando en nodos internos a través de múltiples capas utilizando una condición en donde se evalúa un determinado parámetro y se van separando los registros en sub-sets. Finalmente, en el último nivel encontraremos los nodos finales (hojas), que denotan todos los posibles resultados dentro de nuestro *dataset*:

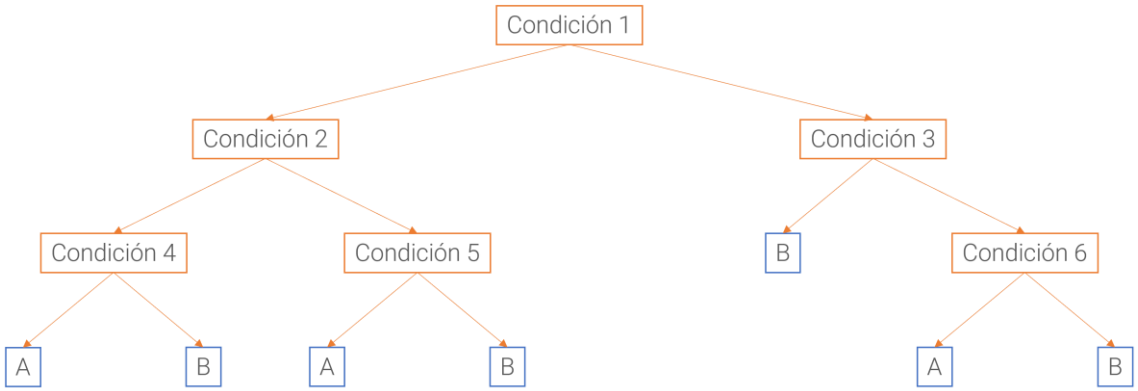


Ilustración 1 - Esquema ilustrativo de un árbol de decisión (elaboración propia)

El aprendizaje de un árbol de decisión busca identificar puntos óptimos dentro de un árbol para efectuar las divisiones del *dataset*, iterando hasta que la mayoría (idealmente, el total) de los registros hayan sido clasificados o se alcance un criterio de terminación. En la ilustración 1, se muestra un caso en el cual en cada hoja del árbol existe una única categoría (A | B); es decir, se trata de nodos puros. Sin embargo, esto no será siempre el caso: que todos los registros se clasifiquen como conjuntos homogéneos depende de la complejidad del árbol. Si el árbol tiene mucha profundidad, es probable que nos encontremos con hojas en donde existan muy pocos registros, conduciendo a lo que se conoce como “fragmentación de datos” y es causa de sobreajuste (*overfitting*). Esto puede suceder, por ejemplo, si muchos de los atributos (columnas o dimensiones del set de datos) son de tipo numérico con muchos niveles (cantidad de valores posibles que puede tomar la variable).

Brett Lantz, autor del libro *Machine Learning with R*, agrega que en su sencilla interpretación, tanto visual (se ve gráficamente el camino que recorre cada sub-set de datos y a partir de qué condiciones se sigue particionando, asimilable a un diagrama de flujo) como lógica (es un conjunto de condiciones si/no que separa registros que la satisfacen o no), los árboles de decisión tienen un gran beneficio frente a otros modelos de *machine learning* cuya lógica para obtener los resultados no es interpretable en forma sencilla (Lantz, s.f.).

Para construir el árbol, el algoritmo evalúa todas las posibles evaluaciones / condiciones con las que puede particionar el *dataset* original y elige en primera instancia aquella más informativa con respecto a la variable objetivo. Luego, con aquellas condiciones (ahora serán 2) que serán capaces de volver a dividir los dos *subsets* generados en la primera instancia en nuevos conjuntos (4) lo más homogéneos posible. Y el proceso continúa hasta la separación total o alcanzar un criterio de terminación. Generalmente, esto último es lo deseado, intentando mantener árboles sencillos de poca profundidad para evitar el sobreajuste de los datos. El árbol finalizado es nuestro modelo y la predicción sobre un nuevo registro se hará chequeando en cada paso qué región de la participación se corresponde en cada nodo de decisión (Müller & Guido, 2016).

Para determinar cuál es el mejor atributo para evaluar en cada nodo de decisión existen dos métodos principales:

I. *Information Gain*: se mide para cada atributo la ganancia de información al realizar la división. Esta ganancia de información corresponde a la diferencia de entropía antes y después de realizada la separación (si todos los registros en un *subset* se correspondieran con una única categoría, la entropía tiene su valor mínimo -0-; mientras que si la partición fuera exactamente 50/50, su valor será máximo -1-). El atributo que presenta mayor ganancia de información será, entonces, el elegido.

II. *Impureza de Gini*: es la probabilidad de clasificar en forma incorrecta un registro aleatorio del set de datos si este fuera etiquetado basándonos en la distribución de las clases del set completo.

Podemos resumir las ventajas y desventajas de los árboles de decisión de acuerdo con lo siguiente:

- Ventajas:
    - Fácilmente interpretables
    - No se requiere (mucho) preprocesamiento de los datos: pueden trabajar indistintamente con diferentes tipos de variables (numéricas, categóricas, etc.) y no son sensibles a valores faltantes. Nota aparte: en la práctica, la implementación a realizar en Python con la librería *sklearn*, sólo permite variables numéricas como input, haciendo necesario aplicar técnicas como *OneHotEncoding*, creando una columna binaria (0/1) para cada valor posible de la columna categórica original.

Registro	Columna 1
1	A
2	B
3	C

→

Registro	Columna 1. A.	Columna 1. B.	Columna 1. C.
1	1	0	0
2	0	1	0
3	0	0	1

Ilustración 2 - Ejemplo de aplicación de OneHotEncoding (elaboración propia)

    - Flexibles: pueden ser utilizados en problemas de clasificación o regresión.
    - Insensibles a relaciones subyacentes entre atributos: se pueden modelar más allá de que dos o más dimensiones estén fuertemente correlacionadas.
- Desventajas:
    - Tendientes a sobre ajustar los datos si se permite que el árbol crezca en profundidad y complejidad.
    - Alta varianza: pequeñas variaciones en los datos de origen pueden generar árboles completamente diferentes.

3.3.2. Ensambles de árboles de decisión: Random Forest (RF)

Se llama “métodos de ensamble” a todos aquellos algoritmos que crean numerosos clasificadores que son entrenados en forma independiente y con diferente *subsets* y el resultado final se obtiene a partir de una votación entre todos los clasificadores. Si el problema es de clasificación, se elige el resultado más frecuente; si es de regresión, se evalúa el promedio de todos los resultados individuales. Lo que se consigue con los métodos de ensamble es disminuir problemas de sesgo que puede haber en algoritmos que entrenan un único clasificador con el total de la información (Karczewski, s.f.).

En algoritmo *Random Forest* (“Bosques Aleatorios”) fue desarrollado por Leo Breiman y Adele Cutler y combina el resultado de múltiples árboles de decisión para alcanzar un único resultado final. Para generar sets de entrenamiento individuales para cada árbol, se utiliza *bagging* (o *bootstrap aggregation*), que consiste en seleccionar cada vez un conjunto aleatorio de registros, incluso más de una vez cada uno de ellos, dentro de mi set de entrenamiento del modelo. Importante destacar es que, a menos que se especifique lo contrario, los sets aleatorios generados serán de igual tamaño que el conjunto original, sólo que podrán tener registros repetidos (Breiman, Bagging Predictors, 1996). Otra característica que se adiciona en el caso de los *Random Forest* es lo que se denomina *feature randomness*: en un árbol común, en cada nodo se evalúan todas las dimensiones; en el caso de los bosques aleatorios, cada árbol sólo dispondrá de un subconjunto de las dimensiones elegidas al azar para hacer estas mismas evaluaciones (Breiman, Random Forests, 2001). En consecuencia, no sólo cada árbol se entrenará con datos diferentes, sino considerando dimensiones distintas. Una derivación de esto último, y que hará que los árboles dentro de un bosque aleatorio sean diferentes a un modelo de árbol individual es que, en general, serán más profundos en este modelo de ensamble. Esto se debe a que, justamente, no se elige la mejor dimensión en cada nodo para ir dividiendo los registros, sino que se elige la mejor posible dentro de un subconjunto de variables y, por lo tanto, el árbol requerirá más profundidad para conseguir subconjuntos lo más homogéneos posible (Müller & Guido, 2016).

Existe una alternativa a los clasificadores *Bagging*: los métodos de *Boosting*. A diferencia del primero, en donde todos los algoritmos individuales se entrenan en forma paralela y luego se combinan los resultados, en los clasificadores *boosting* el trabajo se realiza en serie: el set de entrenamiento utilizado en cada instancia es seleccionado en función de la performance de las instancias anteriores. De esta manera, lo que se consigue es que los ejemplos que fueron predichos incorrectamente en los clasificadores previos sean elegidos con mayor frecuencia que aquellos que fueron correctamente etiquetados (Opitz & Maclin, 1999).

Set Original	1	2	3	4	5	6	7	8

Bagging								
Set de entrenamiento 1	8	7	4	6	6	1	4	8
Set de entrenamiento 2	3	5	5	3	7	6	1	8
Set de entrenamiento 3	5	8	7	5	2	1	5	4
Set de entrenamiento 4	5	3	7	2	8	5	3	7

Boosting								
Set de entrenamiento 1	2	8	8	4	2	8	3	4
Set de entrenamiento 2	1	7	3	4	1	6	4	1
Set de entrenamiento 3	1	2	4	1	1	4	2	1
Set de entrenamiento 4	1	7	7	1	1	1	5	1

Ilustración 3 - Ejemplo de formación de los sets de entrenamiento con Bagging y Boosting, considerando que el valor 1 fuera una anomalía. En el caso de Boosting, irá apareciendo con mayor frecuencia (Opitz & Maclin, 1999)

Todas las consideraciones y descripciones evaluadas para los árboles de decisión individuales también aplican en el caso de este método de ensamble. Sin embargo, además de atacar el problema del sesgo, también se puede resolver, aunque sea parcialmente, el inconveniente muy frecuente en los árboles que es el de sobreajuste de los datos. Por otro lado, un gran número de modelos no correlacionados (logrado con *Bagging* y *Feature Randomness*) operando como un comité (en un problema de clasificación, se elige el output mayoritario entre todos los árboles) performará por encima de cualquiera de los modelos individuales que lo constituyen: cada árbol compensará parcialmente los errores de otro.

Podemos resumir las ventajas y desventajas del modelo *Random Forest* de acuerdo con lo siguiente:

- Ventajas:
    - Reduce el riesgo de sobreajuste de los datos (*overfitting*)
    - Flexibles: pueden ser utilizados en problemas de clasificación o regresión con un elevado nivel de *accuracy*.
- Desventajas:
    - Costo: con sets de datos voluminosos, llevará significativamente más tiempo de entrenar que con un árbol individual.
    - Interpretabilidad: no son sencillos de interpretar como en el caso de los árboles individuales y, por lo tanto, actuará más como una "caja negra" el modelo.
    - No suelen lograr buena performance en los casos en que el *dataset* presentan datos dispersos y una alta dimensionalidad (como datos de texto) (Müller & Guido, 2016)

3.3.3. Artificial Neural Network (ANN)

Reciben su nombre a partir del hecho de que intentan recrear la forma en la que las neuronas (biológicas) interaccionan entre sí. Pueden ser entendidas como una generalización de modelos lineales, pero con múltiples instancias de procesamiento para arribar a una decisión final (Müller & Guido, 2016). Las redes neuronales artificiales constan de capas de nodos en el siguiente orden:

- Capa de entrada (*input layer*)
- Capa(s) intermedia(s) oculta(s)
- Capa de salida (*output layer*)

Las redes de aprendizaje profundo (*Deep Learning*) son aquellas redes neuronales con más de una capa intermedia. Para desarrollar el funcionamiento de las redes neuronales, primero se establece el concepto de perceptrón simple, creado por Frank Rosenblatt en 1958 y esquematizado en la Ilustración 4:

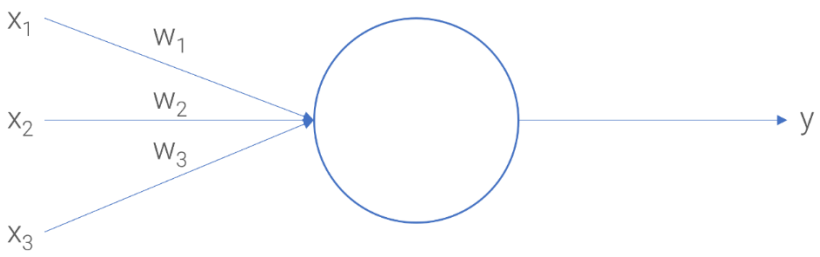


Ilustración 4 - Perceptrón simple (elaboración propia)

El perceptrón simple es una única neurona o nodo. Las redes neuronales tendrán varios en cada una de las capas intermedias (ocultas) y su número puede ser variable (configurable). En este caso, tenemos 3 variables de entrada (*input*) X y una sola de salida Y (*output*). El valor de la variable de salida será una combinación lineal de aquellas de entrada, ponderada cada una de ellas por un peso W, cuyo valor es aleatorio al comenzar el entrenamiento de una red. Estos valores W ayudan, eventualmente, a determinar la importancia de cada una de las variables en términos de predicción de la variable objetivo. A dicha combinación se le suma un valor *bias*, constante, que ayuda al modelo a ajustar de la menor manera posible (Dhingra, 2021):

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + bias$$

El hecho de que, en una primera instancia, los pesos se establezcan en forma aleatoria antes de que el aprendizaje comience, hace que cada red neuronal entrenada diferirá de otra por más que los parámetros establecidos sean los mismos. De todas maneras, esto solo podría afectar en forma significativa la precisión del modelo en el caso de las redes más pequeñas (Müller & Guido, 2016). En un segundo paso, a este valor obtenido (z) se le aplica una función de activación, que consiste en una transformación no lineal aplicada previamente al envío de la señal (el resultado) a la siguiente capa de neuronas en una red. Allí yace la principal diferencia con los modelos de combinación lineal: la función de activación es lo que le inocula la no-linealidad al modelo (Dhingra, 2021). Un ejemplo de función de activación sencilla sería un escalón: se establece un umbral que, en caso de ser superado por el valor de la

combinación lineal (z), toma valor 1, hace que la célula quede activa y el resultado siga su camino hacia adelante en la red. Caso contrario, toma valor 0:

$$y = f(x) = \begin{cases} 0, & x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + bias < umbral \\ 1, & x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + bias \geq umbral \end{cases}$$

Las funciones de activación que se suelen utilizar para activar/inhibir a los nodos son, por ejemplo (Panneerselvam, 2021):

- Función sigmoidea
- Tangente hiperbólica
- *Rectified Linear Unit Function* (ReLU): toma valor 0 cuando la combinación lineal (z) es negativa. Sobre esta misma existen algunas variantes como *Leaky ReLU* y *Parametric ReLU*.
- *Softmax*: es una generalización de la función logística, que convierte un vector de n números reales a una distribución de probabilidad con n posibles resultados.

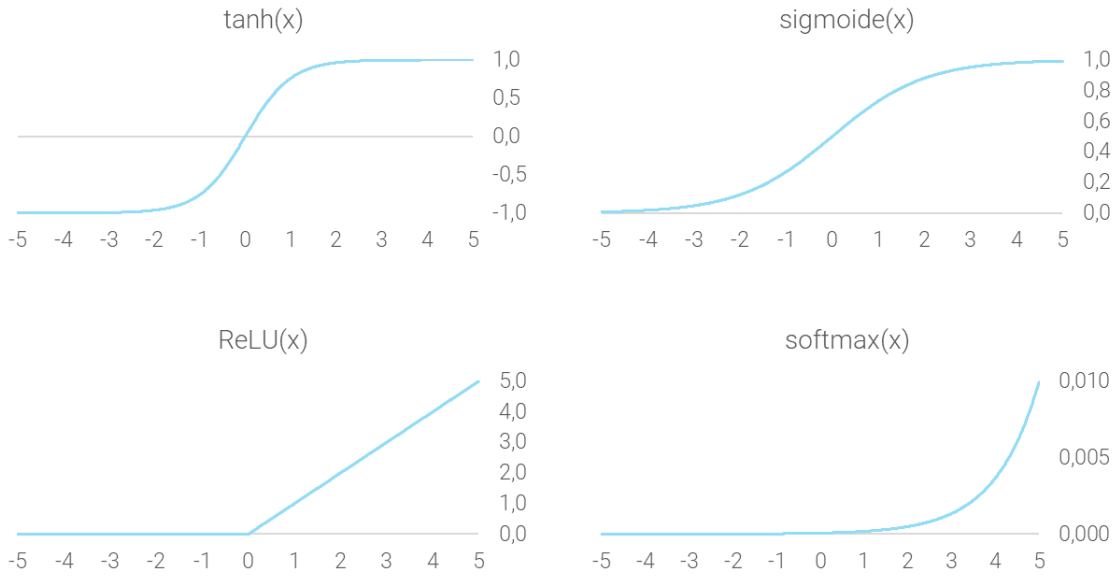


Ilustración 5 - Ejemplos de funciones de activación usuales (elaboración propia)

Una característica por cumplir es que deben ser rápidamente convergentes con respecto a los pesos, con el fin de optimizar el algoritmo. Cuanto más compleja es la información con la que trabajemos, mayor suele ser la no-linealidad del mapeo de las características. Si no hubiera una función de activación presente en cada una de las neuronas

de la red neuronal, el modelo no podría resolver problemas tal como lo hacen (Panneerselvam, 2021).

Conceptualmente, lo visto para un único nodo, no cambia cuando tenemos varias neuronas conectadas entre sí y en distintas capas sucesivas. Cuantas más instancias intermedias tengamos, más información podremos incorporar al modelo. Pasando del perceptrón simple a una red con una sola capa intermedia, podemos modelar el siguiente caso, tomando la función tangente hiperbólica como método de activación:

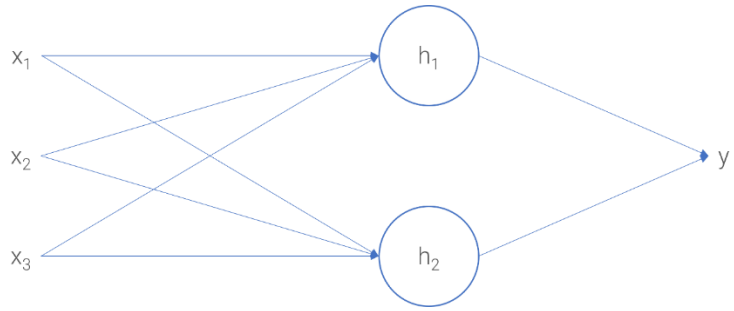


Ilustración 6 - Red Neuronal con 3 input, una capa intermedia de 2 neuronas y un output (elaboración propia)

$$h_1 = tanh\left(x_1 \cdot w_{1,1} + x_2 \cdot w_{2,1} + x_3 \cdot w_{3,1} + b_1\right)$$
$$h_2 = tanh\left(x_1 \cdot w_{1,2} + x_2 \cdot w_{2,2} + x_3 \cdot w_{3,2} + b_2\right)$$

Para llegar al *output Y*, se realiza una combinación lineal entre h_1 y h_2, ponderándolos también con un vector de pesos (tal como el que pondera cada uno de los inputs) (Müller & Guido, 2016). El valor de salida obtenido a partir de esta primera propagación hacia adelante (en el sentido input → output) se comparará con el valor esperado y, a partir de la diferencia entre ambos, se recalibran los pesos que unen todos los nodos (propagación hacia atrás de los errores, también conocido como *Backpropagation*). Luego, se itera el proceso de tal forma de ir actualizando los ponderadores buscando minimizar el error obtenido en la variable objetivo.

Podemos resumir las ventajas y desventajas de las redes neuronales artificiales (ANN) de acuerdo con lo siguiente (Müller & Guido, 2016):

- Ventajas:
    - Son capaces de capturar información contenida dentro de una gran cantidad de datos y construir modelos complejos.

- Suelen tener resultados por encima de otros algoritmos usuales de *machine learning*, tanto en problemas de clasificación como de regresión.

- Desventajas:
  - Costo: con sets de datos voluminosos, llevará significativamente más tiempo de entrenar que otros modelos.
  - Baja o nula interpretabilidad de los resultados en función de las variables de entrada.
  - Requieren un elevado preprocesamiento de los datos: es muy recomendable escalar o normalizar los datos. En este caso, sólo es posible incorporar información numérica al modelo.
  - Trabajan mejor con datos homogéneos, donde todas las características tienen un significado similar, en contraposición con los árboles de decisión, por ejemplo, que podemos trabajar con dimensiones muy distintas entre sí.

3.4. Desbalance de datos

El análisis de los números de fraude a nivel local, regional y global mostraron que, en general, el nivel de transacciones ilegítimas no suele superar el 2% del total. Más allá del problema económico en sí, esto indica que típicamente nos encontraremos con bases de datos en las que, por cada registro fraudulento, tendremos cientos de casos legítimos. Esto es lo que caracteriza a un *dataset* desbalanceado. Una primera clasificación inicial indica que aquellos en los que la relación entre la clase mayoritaria (transacción genuina, en nuestro caso) y la minoritaria está en torno a 2:1, se define como “marginalmente desbalanceado”; cuando la ratio es del orden de 10:1, “modestamente desbalanceado” y 1000:1 o superior, “extremadamente desbalanceados” (He & Ma, 2013).

Una de las dificultades en problemas de detección y prevención de fraude radica en que, típicamente, existen muchas transacciones legítimas por cada una de las fraudulentas.

Entonces cualquier método que identifique correctamente el 99% de los registros puede parecer, en una primera instancia, como un modelo con muy buena performance pero, en realidad, sólo estará siendo efectivo para clasificar la clase mayoritaria (registros legítimos) (Bolton & Hand, 2022). Esto implica una primera necesidad: será indispensable contar con un gran volumen de datos para poder entrenar nuestro modelo, dado que requeriremos una cantidad de registros fraudulentos suficiente para que el algoritmo aprenda a clasificarlos. Esto trae aparejada otra definición, que es la de rareza absoluta y relativa: no es equivalente una ratio de 100:1 cuando tenemos un *dataset* con 10.000 registros positivos y 100 negativos (rareza absoluta), que cuando tenemos 10.000.000 positivos y 100.000 negativos (rareza relativa) (He & Ma, 2013).

En caso de entrenar un modelo sobre un set de datos desbalanceados, datos empíricos muestran que el impacto, medido en términos de ratio de error al predecir la clase minoritaria en contraposición con la mayoritaria, es inclusive significativa en *datasets* con desbalances ligeros de hasta una proporción 3:1. En el caso de una ratio 10:1, por ejemplo, el efecto es severo: la tasa de error al clasificar la clase minoritaria supera al de la mayoritaria más de 20 veces (He & Ma, 2013).

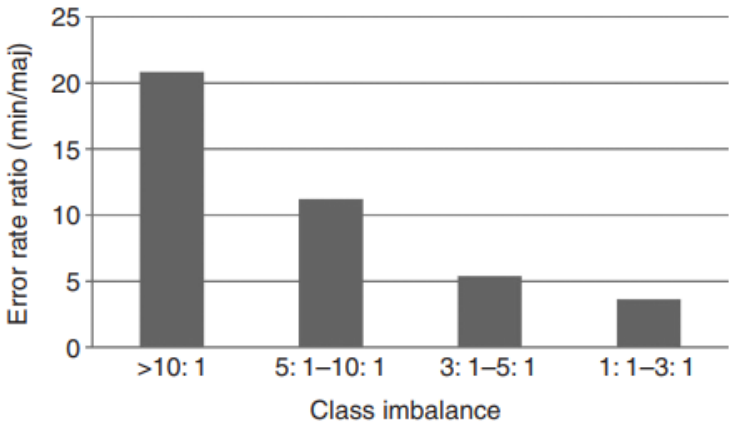


Figura 6 - Impacto en la performance consecuencia del desbalance en la clase minoritaria (Fuente: He & Ma, 2013)

Por otro lado, el costo de clasificar una transacción legítima como fraudulenta (falsos positivos) no es el mismo que en el caso inverso (falsos negativos). En el primer caso, la consecuencia estará por el lado del negocio, dado que se impedirá una operación y eso irá en

detrimento de la experiencia del usuario, la conversión (cuántos intentos de compra terminan siendo exitosos) y los ingresos de la empresa. En el segundo, la consecuencia será directamente económica: si el no haber detectado dicha transacción fraudulenta termina en una demanda por contracargo, no solo habrá que hacerle frente a dicha denuncia, sino que, lo más probable, es que se falle a favor del usuario y haya que abonar el monto (perdiendo el producto y el dinero). En consecuencia, tendremos que buscar un modelo que no solo busque maximizar la cantidad de aciertos globales, sino que lo debe garantizar para ambas clases en particular. Esto también implica no sólo observar la métrica *accuracy* para evaluar el rendimiento de un modelo, sino otras que expliquen qué tan bien se acierta sobre cada clase en forma individual, así como métricas que permiten inferir qué tan bien se separan ambos tipos de categorías (Benchaji, Douzi, & Ouahidi, 2019).

Es importante remarcar que el desbalance entre clases solamente es un problema debido a que los algoritmos usuales no pueden manejar adecuadamente este tipo de datos al momento de aprender (He & Ma, 2013). En consecuencia, si no se dispone de una estrategia para atacar el problema del desbalance de los datos, nuestro modelo servirá de poco. Existen numerosas formas de encarar dicha situación, que pueden ser divididas en tres grandes categorías (Benchaji, Douzi, & Ouahidi, 2019):

- Solución a nivel del problema: consiste básicamente en redefinir el problema a encarar. Es decir, intentar acotar / filtrar los datos (definir un subdominio) de tal manera que el desbalance se reduzca. Un ejemplo de esto en el caso de un algoritmo que busque diagnosticar determinada enfermedad (en un set de datos típicos, la cantidad de registros de personas sanas sobrepasará por mucho a aquellos de las enfermas), se puede acotar únicamente la población dentro de una determinada franja etaria (por ejemplo, mayores de 80 años, en donde es más probable que la frecuencia de aparición de registros positivos para la enfermedad sea superior a la media de la población en su conjunto) (He & Ma, 2013).

- Solución a nivel de los datos: entre estas técnicas, se encontrarán aquellas que buscarán balancear el *dataset*, ya sea creando réplicas de la clase minoritaria u *oversampling* (lo que lleva a prácticamente duplicar el tamaño del set de datos) o seleccionando únicamente una pequeña fracción de los registros de la clase mayoritaria (*undersampling*). Esto no viene sin un costo asociado: las técnicas de *oversampling* pueden conducir a un sobreajuste del modelo, mientras que aquellas de *undersampling* pueden implicar pérdida de información importante (dado que se deja de lado la mayor parte de los registros y esto, además, se efectúa de una manera no supervisada), así como un incremento de la varianza. Por otro lado, el balancear artificialmente el *dataset* puede ayudar con los efectos que esto conlleva, pero no elimina el problema de la distribución de los datos subyacente desbalanceada (He & Ma, 2013).
- Algoritmo elegido: se pueden utilizar métodos de ensamble, dado que al combinar múltiples clasificadores reducirán la varianza de los datos en función del algoritmo:
    - *Bagging*: se puede forzar una selección balanceada de la clase minoritaria (dado que los registros se pueden repetir) y luego completar con igual cantidad de registros de la clase mayoritaria.
    - *Boosting*: esto ya por sí sólo tendrá un mejor resultado (que un método que no sea de ensamble) sin parámetros adicionales dado que en cada instancia sucesiva se van trasladando todos aquellos registros incorrectamente clasificados, progresivamente favoreciéndose así hacia la clase minoritaria.

Adicionalmente, se pueden realizar algunas modificaciones en los parámetros de los modelos para lograr buena respuesta ante datos desbalanceados. Por ejemplo, la técnica de *Cost Sensitive Training*, con la cual se penalizan más los errores de clasificación para la clase minoritaria (Chen & Breiman, 2004).

En nuestro caso en particular, no existiría a priori un subdominio a partir del cual se puedan particionar los datos y de esta manera conseguir datos más balanceados. En cuanto a la elección de los algoritmos, sí evaluaremos el método de ensamble *RandomForest* en el cual se aplica la técnica de *bagging*. Pero la estrategia general será la siguiente: los algoritmos elegidos los probaremos, tanto utilizando el *dataset* original desbalanceado, como luego de haber aplicado las siguientes técnicas de *resampling*: *undersampling* y *oversampling*.

3.4.1. Undersampling

- *Random Under Sampling*: consiste en seleccionar un *subset* de datos elegidos en forma aleatoria (pudiendo haber registros repetidos), de forma tal que se elijan tantos de la clase mayoritaria como existentes de la minoritaria.
- T*omek Links*: consiste en remover instancias de la clase mayoritaria siempre que exista un par de registros, uno de cada clase, en los que ambos sean mutuamente su vecino más próximo (lo que conforma, justamente, un *Tomek-link*). La consecuencia de esto es lograr una mayor separación entre las distintas categorías. Dado que únicamente se eliminarán algunos registros de la clase mayoritaria, el *subset* resultante no estará, necesariamente balanceado, aunque sí existirá una mayor separación entre ambos tipos de datos.

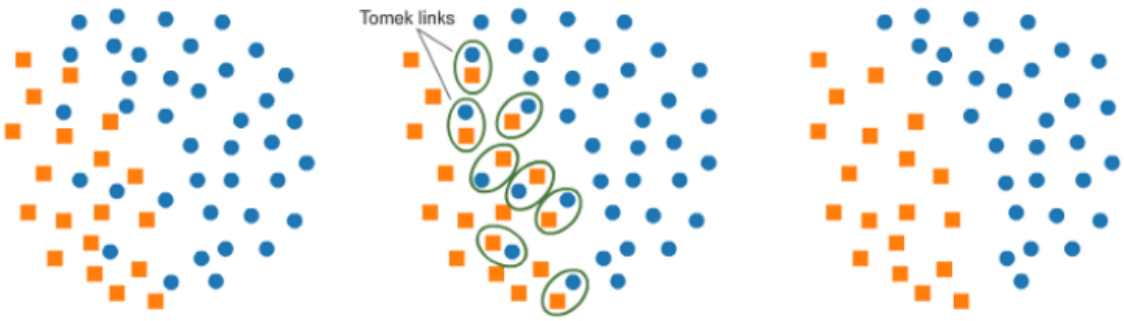


Figura 7 - Undersampling con Tomek Links (Fuente: Kumar, 2020)

3.4.2. Oversampling

- *Random Over Sampling*: consiste en realizar copias de registros de la clase minoritaria elegidos en forma aleatoria del set de datos.

- *Syntethic Minority Oversampling Techinique* (SMOTE): esta técnica genera datos sintéticos de la clase minoritaria. Es decir, no genera copias, sino registros similares interpolándolos entre puntos existentes cercanos. Para lograr esto, lo que se hace es seleccionar aleatoriamente un registro de la clase minoritaria y computando los k-vecinos más cercanos para el mismo. Luego, los registros sintéticos generados serán agregados entre los puntos elegidos y sus vecinos. Esto se repite hasta alcanzar un set de datos balanceado (Fernández, García, Galar, Prati, & Krawczyk, 2018).

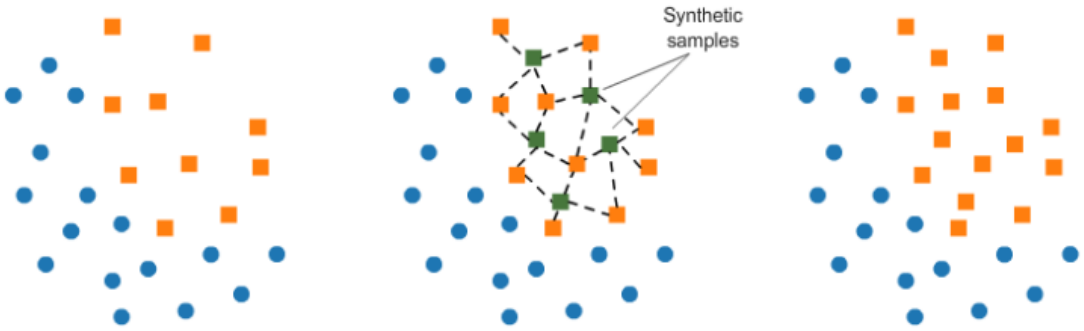


Figura 8 - Oversampling utilizando SMOTE (Fuente: Kumar, 2020)

Tahir, Kittler & Yan adicionan una técnica particular: realizar *undersampling* sobre la clase mayoritaria pero no hasta el punto de igualar la cardinalidad de ambas clases, sino hasta el extremo de invertir la cardinalidad original. Es decir, si nuestro *dataset* tenía una ratio 1000:1, en el *sampleo* posterior, será 1:1000. Para ello, el algoritmo consiste en armar diferentes *subsets* de datos en donde cada uno contiene la totalidad de la clase minoritaria y sólo unos pocos registros de la mayoritaria, entrenar todos los clasificadores por separado y luego hacer el ensamble final de los modelos (Tahir, Kittler, & Yan, 2012). Los resultados que presentan, si bien prometedores, no dejan de mencionar que sólo es posible llevar adelante un ejercicio de esta manera cuando tenemos muchos registros para la clase minoritaria que resulta de interés (rareza relativa).

En la práctica, si bien los métodos de *sampleo* como los anteriores son utilizados con frecuencia para balancear *datasets* con distribución de clases sesgadas, no es claro que esos mismos algoritmos no hubieran podido ser correctamente entrenados con los datos originales.

Por otro lado, tampoco existe un marco teórico desarrollado que explique cómo técnicas de *undersampling*, por ejemplo, pueden afectar la precisión del proceso de aprendizaje (Dal Pozzolo, Bontempi, & Caelen, When is undersampling effective in unbalance classification tasks?, 2015).

Por otro lado, no sólo es importante entender de nuestros datos cuál es la frecuencia de aparición relativa entre las clases, sino también qué tan separadas están una de otra. A modo de ejemplo, en la figura 9 se presenta, en líneas continuas, un ejemplo en que dos clases (C_0 y C_1), tienen una relación 9:1 (C_0 conforma el 90% del *dataset*).

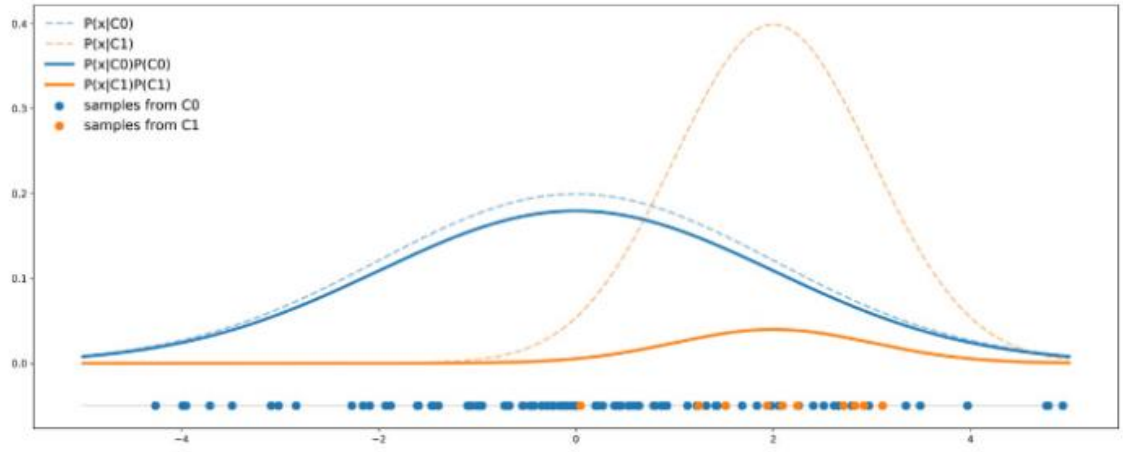


Figura 9 - Ejemplo de 2 distribuciones de clases C0 y C1, con una relación 9:1 y medias 0 y 2, respectivamente (Fuente: Rocca, 2019)

La primera, tiene una media en 0 y varianza 4; mientras que la segunda, media en 2 y varianza 1 (las líneas punteadas corresponden a las distribuciones sin que pese el factor 9:1 entre las clases). Es un buen ejemplo para mostrar que, en este caso, siempre será mayor la probabilidad de que cualquier punto elegido al azar se corresponda con la clase C_0 (la curva azul siempre por encima de la naranja). Es decir, si un algoritmo buscara maximizar la métrica *accuracy*, lo conseguirá prediciendo siempre C_0. Si la media de la curva naranja fuese 10, en lugar de 2, la situación sería la que se observa en la Figura 10, en donde sigue existiendo el mismo desbalance en la cantidad de casos de una clase y la otra, ahora ambas categorías son perfectamente separables y para, aproximadamente, valores entre 7,5 y 10,5, siempre será mayor la probabilidad de que corresponda a la clase C_1.

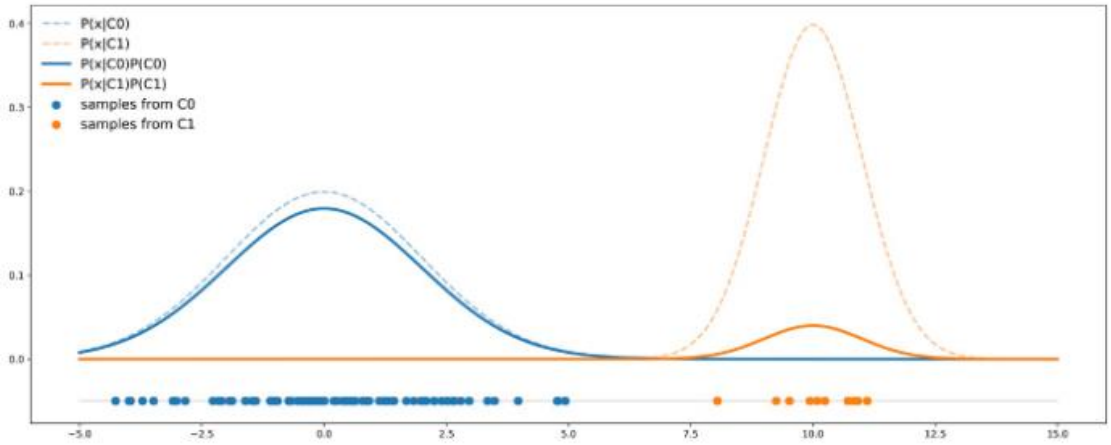


Figura 10 - Ejemplo de 2 distribuciones de clases C0 y C1, con una relación 9:1 y medias 0 y 10, respectivamente (Fuente: Rocca, 2019)

En este caso, sin importar el desbalance, un algoritmo entrenado podrá predecir adecuadamente si se trata de una u otra. Esto pone en evidencia que, en muchos casos, aun aplicando técnicas de *resampleo* de los datos, no siempre puede ser suficiente para lograr buenos resultados y que, incluso, en varios casos, ni siquiera son necesarios (Rocca, 2019).

Por lo tanto, el interés estará centrado en comparar resultados entre los algoritmos a evaluar utilizados sobre los datos crudos, así como sobre los sets balanceados con las técnicas presentadas y ver si la performance se podría haber mejorado aún sin aplicar ninguna de estas técnicas.

3.5. Métricas de evaluación

En un problema de clasificación binaria, los resultados se pueden exponer en una matriz de 2x2 en donde se comparan las dos clases con su valor original y el valor predicho por el modelo mientras fue entrenado:

	Predicha - Txn Genuina	Predicha - Txn Fraudulenta
Real - Txn Genuina	TN	FP
Real - Txn Fraudulenta	FN	TP

Figura 11 - Matriz de confusión para un ejercicio de clasificación binaria (elaboración propia)

En la matriz presentada en la Figura 11, denominada matriz de confusión, se vuelcan la cantidad de casos de acuerdo con cómo haya clasificado el algoritmo entrenado. Podremos tener:

- TN: verdaderos negativos → una transacción genuina es correctamente clasificada
- TP: verdaderos positivos → una transacción fraudulenta es correctamente clasificada
- FP: falsos positivos → una transacción genuina es clasificada como fraudulenta (conlleva una peor experiencia de usuario y una menor conversión, en nuestro caso)
- FN: falsos negativos → una transacción fraudulenta es clasificada como genuina (en nuestro caso, lo que acarraría un mayor costo, dado que es un potencial caso de contracargo que no estaremos detectando)

En función de estas cuatro posibilidades, se definen las siguientes métricas:

I. *Accuracy*: se define como la cantidad de aciertos del modelo sobre el total de clasificaciones realizadas. Será siempre la suma de los valores sobre la diagonal de los verdaderos con respecto al total de la matriz de confusión:

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

Si el set con el que estamos trabajando fuera balanceado, esta métrica por sí sola ya ofrece un buen panorama del rendimiento de nuestro modelo entrenado. Sin embargo, en categorías que presentan un gran desbalance, podemos obtener un valor de *accuracy* cercano a la unidad aun habiendo clasificado erróneamente a la categoría minoritaria en su totalidad (Google, s.f.)

II. *Precision*: busca informar cuál fue la proporción de identificaciones positivas correctas. Es decir, un modelo con precisión muy baja será evidencia de un gran número de falsos positivos:

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

III. *Recall*: busca informar qué proporción de los en realidad positivos fueron clasificados correctamente. En nuestro caso: cuántas transacciones fraudulentas fueron bien clasificadas. Un modelo con una *recall* baja será, entonces, evidencia de la presencia de muchos falsos negativos:

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

Para una determinada clase, podremos tener entonces las siguientes combinaciones de *precision* y *recall*:

- Alto *recall* + alta *precision*: esa clase está correctamente manejada por el modelo.
- Bajo *recall* + alta *precision*: el modelo no puede detectar la clase adecuadamente, pero, cuando lo hace, comete pocos errores.
- Alto *recall* + baja *precision*: la clase es correctamente detectada por el modelo, pero en esa detección bien podrá incluir casos de la otra clase.
- Bajo *recall* + baja *precision*: el modelo no es capaz de manejar adecuadamente esa clase.

Un modelo que no entregue ninguna clasificación errónea (FN = FP = 0), tendrá valores de ambas en 1. En la práctica, sin embargo, suele suceder que estas dos métricas (precisión y recall) presenten comportamientos contrarios: buscando aumentar la precisión de nuestro modelo, podrá tener como consecuencia que se disminuya el recall y viceversa (Google, s.f.). Ambos deben ser analizados en su conjunto y, para ello, se han creado métricas adicionales que contemplan ambos efectos simultáneamente, como el *F1 Score*.

IV. *F1 Score*: corresponde a la media armónica entre la *precision* y el *recall* de nuestro modelo entrenado:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2} \cdot (FP + FN)}$$

Esta métrica resulta muy útil al trabajar con sets de datos desbalanceados, en contraposición con el *accuracy,* que brinda poca información en estos casos. Sin embargo,

como se observa en la fórmula, no contempla el efecto de los verdaderos negativos en su cálculo.

V. *Área bajo la curva ROC*: la curva ROC (*receiver operating characteristic curve*) es un gráfico que muestra la performance de un modelo de clasificación a lo largo de todos los umbrales. Los ejes que conforman esta gráfica son la tasa de verdaderos (equivalente al *recall*) y la de falsos positivos:

$$True\ Positive\ Rate = TPR = \frac{TP}{TP + FN}$$
$$False\ Positive\ Rate = FPR = \frac{FP}{FP + TN}$$

Si nuestro modelo entrega un valor que se corresponde con la probabilidad de que un registro pertenezca a una determinada clase, luego se puede establecer un umbral (*threshold*) que definirá que efectivamente será etiquetado como dicha categoría si esa probabilidad lo supera. Es decir, si nuestro umbral fuera 0,50; si el modelo entrega para un registro que su probabilidad de ser una transacción fraudulenta es, por ejemplo, 0,75, entonces así será etiquetada (dado que 0,75 > 0,50). La curva ROC muestra el resultado de ambas tasas (TPR y FPR) para todos los umbrales posibles:

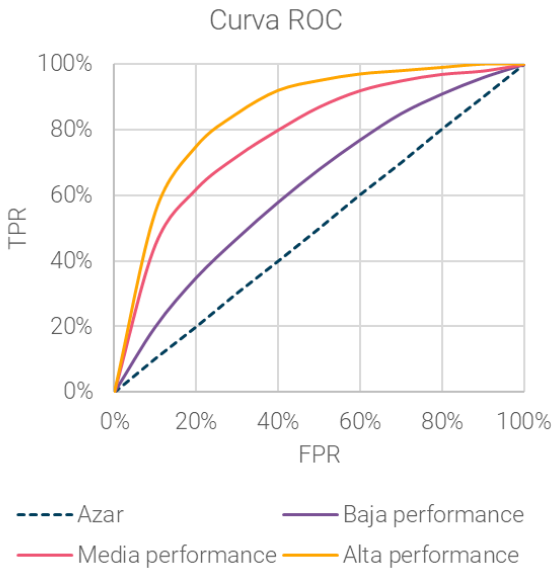


Figura 12 - Ejemplo de curvas ROC y performance del modelo (elaboración propia)

La distinción entre la performance que se observa en la figura 12 corresponde a cuánta precisión debe sacrificar un modelo con tal de conseguir un alto *recall*. A partir de la curva ROC se elabora una métrica adicional, que es el valor del área bajo la misma. Cuanto más se acerque este valor a la unidad, mayor *precision* y *recall* tendremos, dando lugar a un modelo que es más capaz de distinguir entre las clases (Bhandari, 2020). Si su valor es exactamente 1, entonces el modelo es capaz de distinguir perfectamente, en nuestro caso, todas las transacciones seguras de las fraudulentas. Si fuera 0, entonces todas las fraudulentas serían etiquetadas como genuinas y viceversa. Caso particular, si el valor del área bajo la curva (AUC, por su definición en inglés *Area Under Curve*) es 0,5, entonces el clasificador no es capaz de distinguir entre las clases; implicando o bien que está clasificando al azar o que predice una sola clase para todos los puntos. En definitiva, entonces, este valor para el AUC será el menor valor posible esperable (correspondiente a una clasificación totalmente azarosa, como se muestra en la Figura 12).

Sin embargo, el AUC mide la calidad de las predicciones del modelo independientemente del umbral de clasificación elegido, cosa que no es siempre deseable. En casos en los que existan grandes disparidades en el costo asociado a las clasificaciones erróneas (es decir, que no es indistinto cometer error tipo I que error tipo II), el AUC entonces no es una métrica del todo útil (Google, s.f.). Por ejemplo, en nuestro caso particular, la prioridad principal es disminuir los falsos negativos (transacciones fraudulentas que pasan como genuinas). En definitiva, es una métrica adicional para considerar, pero siempre en conjunto con las restantes para poder arribar a una conclusión sobre la calidad y rendimiento de un modelo.

4. Desarrollo experimental

Para el desarrollo de código y entrenamiento de los modelos, se ha utilizado un equipo con el siguiente hardware:

- Procesador: Intel(R) Core$^{(TM)}$ i7-10750H CPU @ 2.60GHz

- RAM instalada: 32,0 GB
- Tipo de sistema: Sistema operativo de 64 bits, procesador x64
- Sistema operativo: Windows 11 Home – Versión 22H2. Compilación: 22622.575

El lenguaje utilizado fue Python, versión 3.9.12 y la IDE, Jupyter Notebook con las siguientes versiones para los paquetes principales:

- IPython: 8.2.0
- ipykernel: 6.9.1
- ipywidgets: 7.6.5
- jupyter_client: 6.1.12
- jupyter_core: 4.9.2
- jupyter_server: 1.13.5
- jupyterlab: 3.3.2
- nbclient: 0.5.13
- nbconvert: 6.4.4
- nbformat: 5.3.0
- notebook: 6.4.8
- qtconsole: 5.3.0
- traitlets: 5.1.1

4.1. Dataset

El set de datos utilizado contiene las órdenes registradas en la plataforma entre el 1/2/2022 y el 31/5/2022, totalizando 498.002 registros, de los cuales solo 1.488 (0,299%) son contracargos comprobados. Esto se corresponde con una ratio entre las clases a predecir de aproximadamente 333:1. De acuerdo con la clasificación presentada, se trata entonces de un desbalance severo (aunque no extremo) y con una rareza absoluta en la clase minoritaria.

Las dimensiones del modelo son las siguientes:

DIMENSIÓN	TIPO	TIPO DE VARIABLE	% VALORES NULOS
buyer_id	Comprador	Alfanumérica	0%
buyer_shipping_address_city	Comprador	Categórica	0%
buyer_shipping_address_district	Comprador	Categórica	100%
buyer_age	Comprador	Numérica	63%
buyer_gender	Comprador	Numérica	66%
fulfilment_channel_id	Envío	Alfanumérica	0%
shipping_channel_id	Envío	Alfanumérica	0%
shipping_method_id	Envío	Alfanumérica	0%
category_id	Objeto	Alfanumérica	0%
global_be_category_id	Objeto	Alfanumérica	0%
group_id	Objeto	Alfanumérica	0%
item_id	Objeto	Alfanumérica	0%
level1_category_id	Objeto	Alfanumérica	100%
level2_category_id	Objeto	Alfanumérica	100%
level3_category_id	Objeto	Alfanumérica	100%
level4_category_id	Objeto	Alfanumérica	100%
level5_category_id	Objeto	Alfanumérica	100%
model_id	Objeto	Alfanumérica	0%
global_be_category	Objeto	Categórica	0%
item_name	Objeto	Categórica	0%
gmv_usd	Objeto	Numérica	0%
payment_be_channel_id	Pago	Alfanumérica	0%
payment_method_id	Pago	Alfanumérica	0%
payment_be_channel	Pago	Categórica	0%
payment_channel	Pago	Categórica	0%
payment_method	Pago	Categórica	0%
buyer_txn_fee_usd	Pago	Numérica	0%
coin_used	Pago	Numérica	0%
checkout_channel_id	Transacción	Alfanumérica	0%
checkout_id	Transacción	Alfanumérica	0%
fsv_promotion_id	Transacción	Alfanumérica	22%
order_id	Transacción	Alfanumérica	0%
order_sn	Transacción	Alfanumérica	0%
pv_promotion_id	Transacción	Alfanumérica	78%
pv_voucher_code	Transacción	Alfanumérica	78%
sv_promotion_id	Transacción	Alfanumérica	90%
is_flash_sale	Transacción	Binaria	0%
is_web_checkout	Transacción	Binaria	0%
checkout_channel	Transacción	Categórica	0%
create_datetime	Transacción	FechaHora	100%
create_timestamp	Transacción	Timestamp	100%
merchant_id	Vendedor	Alfanumérica	100%
seller_id	Vendedor	Alfanumérica	0%
shop_id	Vendedor	Alfanumérica	0%
is_cb_shop	Vendedor	Binaria	0%
is_managed_shop	Vendedor	Binaria	0%
is_official_shop	Vendedor	Binaria	0%
is_preferred_plus_shop	Vendedor	Binaria	0%
is_preferred_shop	Vendedor	Binaria	0%
is_supermarket_shop	Vendedor	Binaria	0%
seller_shipping_address_city	Vendedor	Categórica	0%
seller_shipping_address_district	Vendedor	Categórica	100%

4.2. Preprocesamiento de los datos

La secuencia de tratamiento de los datos desarrollada fue la siguiente:

I. Eliminación de todas aquellas dimensiones que no contienen información (100% de valores nulos).

II. Obtención de variables día de semana y hora del día a partir de create_datetime, dado que no interesa la fecha en sí (a diferencia de si fuera un problema de predicción de una serie temporal).

III. Eliminación de las dimensiones temporales create_datetime y create_timestamp

IV. De la variable payment_channel, solo se extrae el nombre del banco, descartando los 4 números finales de la tarjeta de crédito empleada y el número de cuotas de la transacción (de todas maneras, la variable buyer_txn_fee_usd corresponde al interés del pago en cuotas).

V. Se eliminan dos columnas con muy alta cardinalidad: global_be_category y order_sn.

VI. Conversión de las variables categóricas a numéricas utilizando etiquetamiento (se le asigna un número distinto a cada una de las posibles categorías).

VII. Se completan las columnas con valores nulos:

   a. buyer_gender: se completa hacia abajo, de manera de mantener dentro de lo posible la distribución original.

   b. buyer_age: se imputan los valores nulos con el promedio de los valores originales.

   c. Resto: se completan con valor 0

VIII. Escalamiento de todas las variables con el método MinMaxScaler: todas las variables toman valor entre 0 y 1 en forma proporcional al máximo y mínimo valor posible:

$$valor_{escalado} = \frac{valor_{original} - mín}{máx - mín}$$

IX. División del dataset en set de entrenamiento y testeo, considerando un factor de división de 30% (70% de los registros para entrenar el modelo, 30% para testearlo) y utilizando el parámetro *stratify*, de manera de asegurar que la proporción entre las clases mayoritaria y minoritaria de la variable *target* se mantenga en ambos subsets.

4.3. Resultados

La tabla 2 resume los resultados hallados luego de correr los tres algoritmos seleccionados, tanto para el *dataset* original como evaluando los métodos de *sampleo* descriptos utilizando, en todos los casos, los parámetros por defecto para cada modelo. En cuanto a los métodos de ensamble, se adicionaron corridas con el algoritmo *XGBoost* que, a diferencia de *RandomForest* que utiliza el método de *Bagging*, este utiliza *Boosting* lo que, a priori, debería funcionar mejor con *datasets* desbalanceados dado que en cada *subset* generado se debería ir favoreciendo la clase minoritaria.

	MÉTODO	PRECISION	RECALL	ROCAUC	F1-SCORE	FPR	FNR
1	XGBoost - OS SMOTE	0,227	0,235	0,617	0,231	0,2%	76,5%
2	Random Forest - OS SMOTE	0,393	0,157	0,578	0,224	0,1%	84,3%
3	Random Forest - OS Random Over Sampling	0,776	0,117	0,558	0,203	0,0%	88,3%
4	XGBoost - OS Random Over Sampling	0,096	0,567	0,776	0,164	1,6%	43,3%
5	XGBoost - Raw Data	0,732	0,092	0,546	0,163	0,0%	90,8%
6	XGBoost - US Tomek Links	0,631	0,092	0,546	0,160	0,0%	90,8%
7	Random Forest - Raw Data	0,867	0,087	0,544	0,159	0,0%	91,3%
8	Random Forest - US Tomek Links	0,709	0,087	0,544	0,156	0,0%	91,3%
9	ANN - OS Random Over Sampling	0,040	0,379	0,676	0,072	2,7%	62,1%
10	ANN - OS SMOTE	0,029	0,430	0,694	0,055	4,3%	57,0%
11	XGBoost - US Random Under Sampling	0,018	0,883	0,870	0,036	14,3%	11,7%
12	Random Forest - US Random Under Sampling	0,017	0,890	0,868	0,034	15,3%	11,0%
13	Árbol de decisión - OS Random Over Sampling	0,016	0,818	0,832	0,031	15,4%	18,2%
14	Árbol de decisión - OS SMOTE	0,016	0,818	0,832	0,031	15,4%	18,2%
15	Árbol de decisión - US Random Under Sampling	0,016	0,816	0,830	0,030	15,5%	18,4%
16	ANN - US Random Under Sampling	0,013	0,841	0,823	0,025	19,4%	15,9%
17	Árbol de decisión - Raw Data	1,000	0,000	0,500	0,000	0,0%	100,0%
18	ANN - US Tomek Links	1,000	0,000	0,500	0,000	0,0%	100,0%
19	Árbol de decisión - US Tomek Links	1,000	0,000	0,500	0,000	0,0%	100,0%
20	ANN - Raw Data	1,000	0,000	0,500	0,000	0,0%	100,0%

Tabla 2 - Resultados experimentales iniciales

Si bien se ha calculado la métrica *accuracy* en todos los casos, resulta de poco interés dado que, de acuerdo con lo esperado, supera el 99% en la mayoría de los casos. Los resultados se muestran ordenados de acuerdo con la métrica F1-Score, elegida como la principal a la hora de evaluarlos dado que buscamos un buen balance entre *recall* y *precision*, para atacar simultáneamente los falsos positivos como los falsos negativos.

Aunque los métodos de ensamble *RandomForest* y *XGBoost*, ensayados sobre el *dataset* de entramiento con *oversampling*, muestran los mejores resultados de acuerdo con F1-Score, en ninguno de estos casos podemos afirmar que son números prometedores, siendo la métrica inferior a 0,25 puntos. Por otro lado, aunque con una tasa de falsos positivos muy baja -pocas transacciones genuinas son clasificadas como fraude-, la tasa de falsos negativos sí es elevada, lo que conllevaría un mayor costo en el negocio, dado que se estarán escapando muchos casos de fraude que el modelo catalogará como transacciones genuinas. Son estos mismos algoritmos, pero ensayados sobre el *dataset* de entramiento con *undersampling*, los que consiguen mejores resultados en términos de una baja tasa de falsos negativos:

- XGBOOST - US RANDOM UNDER SAMPLING
- RANDOM FOREST - US RANDOM UNDER SAMPLING

En ambos casos, obtenemos una *recall* aceptable, pero con muy baja precisión: muchas transacciones genuinas serán catalogadas como fraudulentas haciendo necesaria una mayor revisión manual posterior o bien, aceptar una pérdida considerable en la conversión de las transacciones. Por su parte, se obtiene un buen reconocimiento de las transacciones fraudulentas cuando estas ocurren.

4.3.1. Pruebas adicionales

Con esto en consideración, se llevan adelante pruebas para buscar una mejora en la performance del método de *RandomForest,* siguiendo los lineamientos propuestos en la guía *Reproducible Machine Learning for Credit Card Fraud detection - Practical Handbook* (Borgne, Siblini, Lebichot, & Bontempi, s.f.):

I. Ensayo con los parámetros de balance del *dataset* incorporados en el mismo modelo: en clasificador utilizado, disponible en la librería *sklearn*, pone a disposición dos métodos para tratar *datasets* desbalanceados, modificando el parámetro *class_weight*:
    a. *Balanced*: se agrega un ponderador para cada una de las dos clases disponibles que es inversamente proporcional a la frecuencia de aparición para el total del dataset.
    b. *Balanced subsample*: de idéntica aplicación, pero para cada uno de los *subsets* generados para el entrenamiento de cada árbol individual.

II. Prueba con el clasificador *BalancedRandomForestClassifier*, de la librería *imblearn*.

III. Búsqueda exhaustiva de parámetros: aplicando el método *grid search*, se puede ensayar el modelo para múltiples configuraciones de sus parámetros y obtener aquellos que mejor resultan en función de las métricas deseadas.

IV. Combinación de métodos de *sampleo*: en muchas aplicaciones se pueden aprovechar las bondades de los métodos de *oversampling* y *undersampling* utilizados en conjunto para mejorar los resultados. En el primer caso, se utiliza un método para aumentar la cantidad de registros de la clase minoritaria (*oversampling*) y, seguido a ello, uno de *undersampling* para disminuir los de la clase mayoritaria. La librería utilizada dispone de métodos combinados, tales como SMOTE-TomekLinks y SMOTE-ENN (*Edited Nearest Neighbor*). Haremos una prueba con el segundo método. En este caso, el funcionamiento de ENN es una variación del *K-Nearest Neighbors*, en donde se elimina el registro cuya clase no se corresponda con aquella de sus registros vecinos más cercanos, así como estos mismos vecinos (logrando una mayor separación entre las clases). Esto lo diferencia de *TomekLinks*, en donde solo se eliminan los registros de a pares (Nogueira, Lemaitre, Victor, & Aridas, 2022).

Adicionalmente, se corren las siguientes pruebas:

I. *Cost Sensitive Learning*: de igual manera en que se utilizó el parámetro *class_weight* para *RandomForest*, lo empleamos para buscar mejores resultados, tanto con un algoritmo *XGBoost* como con un árbol de decisión individual.

II. Modelo adicional: se hace un ensayo utilizando un modelo de Regresión Logística, que es un modelo ampliamente utilizado en clasificación binaria y que consiste, básicamente, en una regresión lineal incluida en una función logística (y, en consecuencia, los valores de salida quedan acotados al rango [0, 1] y pueden ser interpretados como una probabilidad que, en caso de superar cierto umbral, se predice la categoría "1") (Albon, 2018).

III. Variables categóricas: por otra parte, se identifica la presencia de variables categóricas con una gran cantidad de niveles (valores que pueden adoptar) como el principal problema encontrado durante el preprocesamiento de los datos. Si bien el enfoque ideal hubiera sido utilizar *OneHotEncoding* para tratar todas las categóricas, de haberlo llevado adelante sin miramientos, hubiera generado un set de datos con más de 3000 columnas, haciendo que la memoria requerida para correr los distintos algoritmos fuera muy elevada. En consecuencia, una prueba adicional es tomar, para cada una de las variables categóricas, únicamente los 20 primeros valores en término de frecuencia absoluta y realizar el *encoding* únicamente para esos casos. En otras palabras, sustituir cada una de las columnas categóricas por una cantidad (20) de columnas a ensayar. En el caso de que un elemento no pertenezca a estas primeras clases, entonces tendrá valor 0 en todas las columnas.

Esto último salva un problema muy frecuente que ocurre con *OneHotEncoding* y que es qué sucede cuando aparece una categoría en el set de *testing* o bien, en datos nuevos futuros, que no ha sido previsto en el set de entrenamiento y por lo tanto no pertenece a ninguna de las categorías previstas.

Los resultados de estas pruebas adicionales son los siguientes:

	MÉTODO	PRECISION	RECALL	ROCAUC	F1-SCORE	FPR	FNR
1	Balanced Random Forest	0,013	0,883	0,841	0,026	20,1%	11,7%
2	Weighted DT - Raw Data	0,008	0,803	0,758	0,016	28,7%	19,7%
3	Random Forest - Best Params Raw Data	0,010	0,800	0,787	0,021	22,7%	20,0%
4	Weighted Logistic Regression - Raw Data	0,008	0,791	0,746	0,016	29,9%	20,9%
5	Balanced Logistic Regression - Raw Data	0,008	0,789	0,745	0,016	29,9%	21,1%
6	RandomForest - Combined SMOTE & ENN	0,009	0,769	0,764	0,019	24,1%	23,1%
7	Weighted XGBoost - OS SMOTE	0,026	0,587	0,760	0,049	6,7%	41,3%
8	Weighted XGBoost - Raw Data	0,081	0,565	0,773	0,142	1,9%	43,5%
9	Random Forest - Balanced Subsample	0,806	0,065	0,532	0,120	0,0%	93,5%

Tabla 3 - Resultados experimentales adicionales

Nuevamente, ninguno de estas pruebas proporciona resultados que hagan una diferencia significativa sobre los anteriores: no se consigue tener un buen balance entre *recall* y *precision*. Por ejemplo, tomando los resultados del primer caso *Balanced Random Forest*, con una tasa de falsos positivos de 20,1%, implica que, por cada 10 mil transacciones, tendremos aproximadamente 2 mil que serán etiquetadas como fraudulentas y deberán ser sometidas a revisión manual (si se rechazaran de plano, estaríamos perdiendo 20% de facturación). Por el lado de las transacciones fraudulentas, de acuerdo con la tasa original (0,299%), es esperable tener 30 casos de fraude. Con una tasa de falsos negativos de 11,7%, es esperable entonces que 3 sean consideradas como fraudulentas cuando son en realidad transacciones seguras, mientras que los 27 restantes serán bien etiquetados como casos de fraude (es decir, no se escaparían tantos casos de contracargos). Con esto, resta entonces definir la capacidad de horas hombre disponibles para llevar adelante el análisis posterior de los falsos positivos en el intento de ganar puntos de conversión.

4.3.2. Importancia de las variables

Utilizando los resultados del modelo *RandomForest* con los parámetros resultantes del método *grid search*, se muestra en la Figura 13 la importancia de cada una de las dimensiones ensayadas:

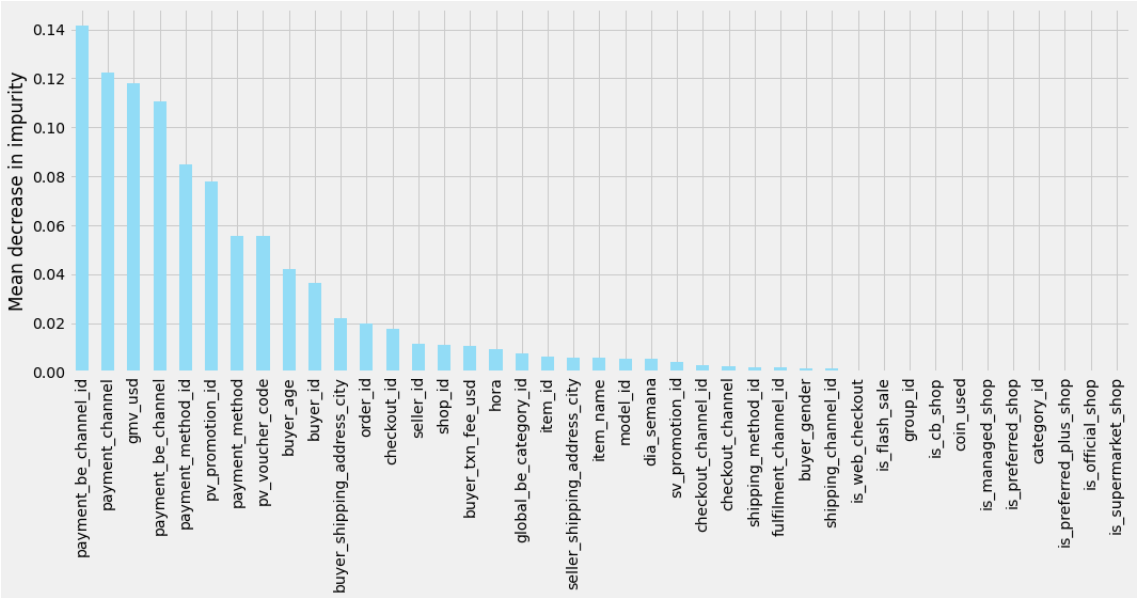


Figura 13 - MDI por dimensión ensayada

El valor obtenido para cada uno se corresponde con la *Mean Decrease in Impurity* (MDI), que calcula acumulando el decrecimiento en impureza de Gini a través de todas las particiones en los nodos de los árboles de todo el bosque aleatorio que incluyen a dicha columna. Lo que se observa es que todas las variables referidas al método de pago, así como el monto mismo de la transacción (*gmv_usd*) son aquellas que resultan de mayor importancia para el modelo. Asimismo, muchas de las variables incorporadas al modelo, tienen importancia muy baja o nula y podrían ser removidas del mismo sin afectar en forma significativa los resultados obtenidos.

4.3.3. Balance recall vs precision

Por otro lado, también con los resultados del mismo modelo ensayado, podemos ver en forma gráfica la relación entre la ganancia en *recall* vs *precision* y cómo, y en línea con lo observado para todos los modelos, no podemos maximizar uno sin disminuir el otro (Sigrist, 2022):

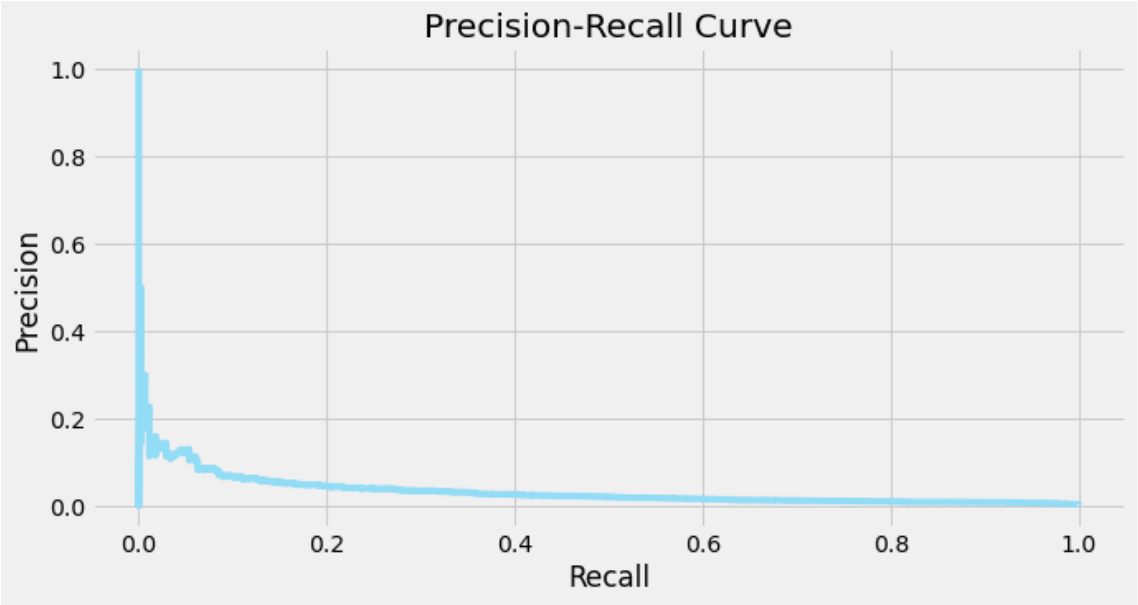


Figura 14 - Curva Recall / Precision

El modelo por elegir consistirá, entonces, en el balance entre falsos positivos y negativos deseados. A priori, entendiendo que con falsos positivos se pueden correr pruebas adicionales posteriores y permitir la transacción si no se detecta comportamiento fraudulento en forma manual, se priorizarían modelos que entreguen la menor tasa posible de falsos negativos. En nuestro caso: *RandomForest* o *XGBoost* aprovechando métodos de *undersampling*.

4.3.4. Incorporación de nuevas variables

Con todo lo expuesto, es evidente que el problema de desbalance de los datos es significativo y la mejor forma de encararlo no será por el lado de los modelos en sí (o el *tuning* de sus parámetros) sino por contar con un buen set de datos de entrada. En este caso, sólo contamos con cuatro meses de información y sólo cerca de 1500 casos de fraude para entrenarlos. Es probable que, con más información acumulada, y enriqueciendo el set de datos con variables complementarias que aporten más información sobre los compradores y vendedores, los resultados a obtener serán mejores y el compromiso a realizar entre falsos positivos y negativos será menor.

Entre los datos que uno podría pensar que correlacionarían con la posibilidad de una transacción fraudulenta, tenemos los siguientes:

- Compradores: mayor nivel de detalle de cada usuario que hace cada compra:
    - Cantidad de transacciones pasadas genuinas
    - Cantidad de transacciones pasadas comprobadas fraudulentas
    - Cantidad de dispositivos / direcciones IP ensayados históricamente: puede ser señal de comportamiento fraudulento que el usuario intente realizar una transacción desde diferentes dispositivos y/o locaciones (IP).
    - Cantidad de métodos de pago registrados: puede ser señal de comportamiento fraudulento que el usuario haya intentado pagar en el pasado con diferentes tarjetas de crédito y/o débito y en gran cantidad.
    - Velocidad de transacciones: múltiples reintentos en un período de tiempo acotado puede ser señal de comportamiento fraudulento.
    - Actividad luego de un largo período de inactividad: múltiples reintentos de compra luego de un gran período de inactividad en la plataforma puede ser señal de comportamiento fraudulento.
- Vendedores: mayor nivel de detalle de cada vendedor al momento de cada transacción realizada:
    - Cantidad de operaciones exitosas en el pasado
    - Tasa de cancelación a causa del vendedor en el pasado.
    - Tasa de cancelación a causa del comprador en el pasado.
- Categorías / Monto de operación:
    - Electrónicos / objetos de elevado valor: es probable que los intentos de compra para llevar adelante una demanda por contracargo en el futuro se concentren en rubros e ítems de elevado valor.

De esta lista, con los datos disponibles se construyen las siguientes dimensiones:

- n_devices_acum: cantidad de dispositivos que ha registrado cada comprador a la fecha de la nueva transacción.
- tuvo_cbks_nro: cantidad de contracargos que ha denunciado el comprador previo a la fecha de la nueva transacción.
- hizo_compras_nro: cantidad de compras que ha efectuado el comprador previo a la fecha de la nueva transacción.

Por otro lado, al mismo tiempo que se evaluará el modelo con estas incorporaciones, se dejarán de lado variables que correlacionan fuertemente con otras dimensiones del modelo y que, por lo tanto, no agregan al modelo información relevante. Con esto, los resultados se aprecian en la tabla 4, y se construyen los gráficos 15 y 16 con la relevancia de las variables y la curva de compromiso entre *recall* y *precision.*

	MÉTODO	PRECISION	RECALL	ROCAUC	F1-SCORE	FPR	FNR
1	XGBoost - US Random Under Sampling	0,017	0,894	0,866	0,034	16,2%	10,6%
2	Random Forest - US Random Under Sampling	0,015	0,890	0,855	0,030	18,0%	11,0%
3	ANN - US Random Under Sampling	0,011	0,822	0,799	0,023	22,4%	17,8%
4	Árbol de decisión - OS Random Over Sampling	0,009	0,822	0,761	0,017	29,9%	17,8%
5	Árbol de decisión - OS SMOTE	0,009	0,822	0,761	0,017	29,9%	17,8%
6	Árbol de decisión - US Random Under Sampling	0,012	0,718	0,769	0,024	18,0%	28,2%
7	XGBoost - OS Random Over Sampling	0,103	0,654	0,818	0,177	1,8%	34,6%
8	ANN - OS SMOTE	0,036	0,499	0,729	0,068	4,2%	50,1%
9	ANN - OS Random Over Sampling	0,039	0,480	0,721	0,072	3,8%	52,0%
10	XGBoost - OS SMOTE	0,291	0,268	0,633	0,279	0,2%	73,3%
11	Random Forest - OS SMOTE	0,745	0,223	0,611	0,343	0,0%	77,7%
12	Random Forest - OS Random Over Sampling	0,851	0,170	0,585	0,283	0,0%	83,0%
13	XGBoost - US Tomek Links	0,774	0,153	0,576	0,255	0,0%	84,7%
14	XGBoost - Raw Data	0,792	0,130	0,565	0,223	0,0%	87,1%
15	Random Forest - Raw Data	0,892	0,123	0,562	0,216	0,0%	87,7%
16	Random Forest - US Tomek Links	0,855	0,113	0,556	0,199	0,0%	88,8%
17	ANN - US Tomek Links	0,700	0,015	0,507	0,029	0,0%	98,5%
18	Árbol de decisión - Raw Data	0,500	0,008	0,504	0,017	0,0%	99,2%
19	ANN - Raw Data	0,400	0,004	0,502	0,008	0,0%	99,6%
20	Árbol de decisión - US Tomek Links	0,200	0,002	0,501	0,004	0,0%	99,8%

Tabla 4 - Resultados experimentales con variables adicionales

En este caso, ordenamos crecientemente por tasa de falsos negativos, obteniendo los métodos de *XGBoost* y *RandomForest* como aquellos con mejor performance evaluando este

aspecto. De todas maneras, al área bajo la curva ROC que devuelven es considerable, superando 0,85 en ambos casos.

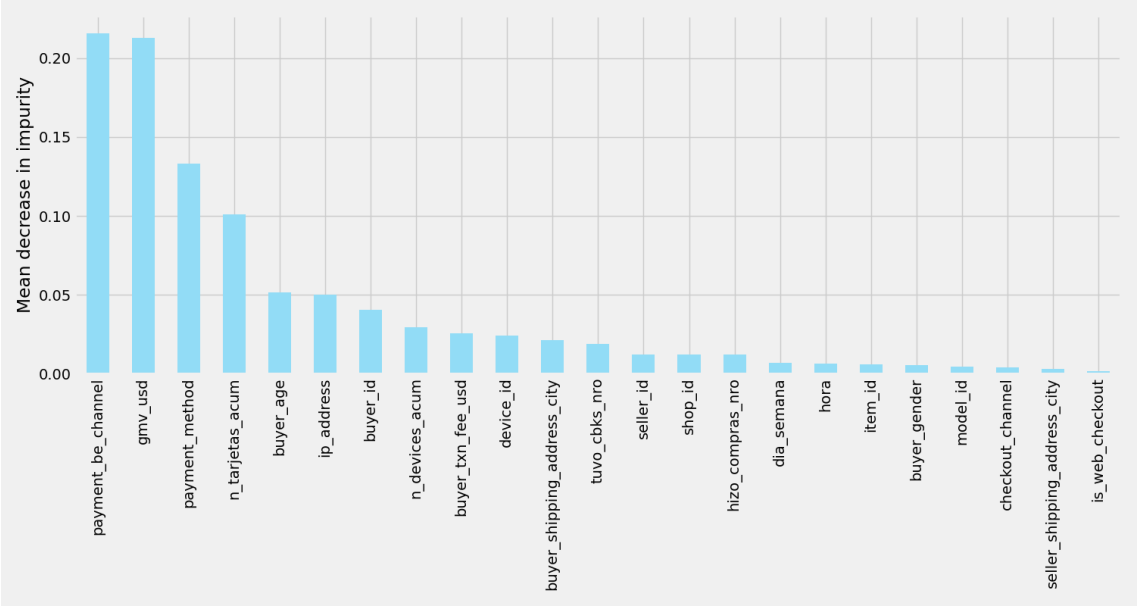


Figura 15 - MDI por dimensión ensayada

Las variables de mayor relevancia siguen siendo las vinculadas al pago en concreto: el canal, el monto y el método de pago. De las incorporadas, la primera en aparecer es el número de tarjetas vinculadas por el usuario, en cuarto lugar. La variable que vincula al usuario con sus contracargos previos no aparece en los primeros lugares, lo cual uno hubiera imaginado que sucedería. Pero esta situación se puede deber a que estamos evaluando pocos meses de historia y, además, son los primeros en la operación de este *Marketplace*, con lo cual, en muchos casos estamos frente a primeras compras y no con fraudes reincidentes.

En la Figura 16, la curva de compromiso entre *precision* y *recall* no muestra una forma muy distinta a la vista con las variables originales.

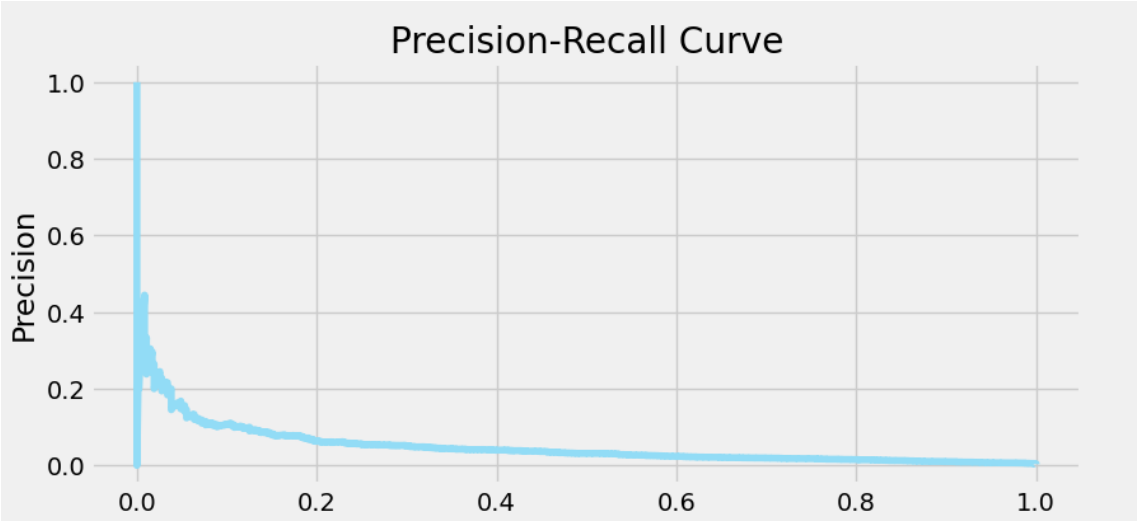
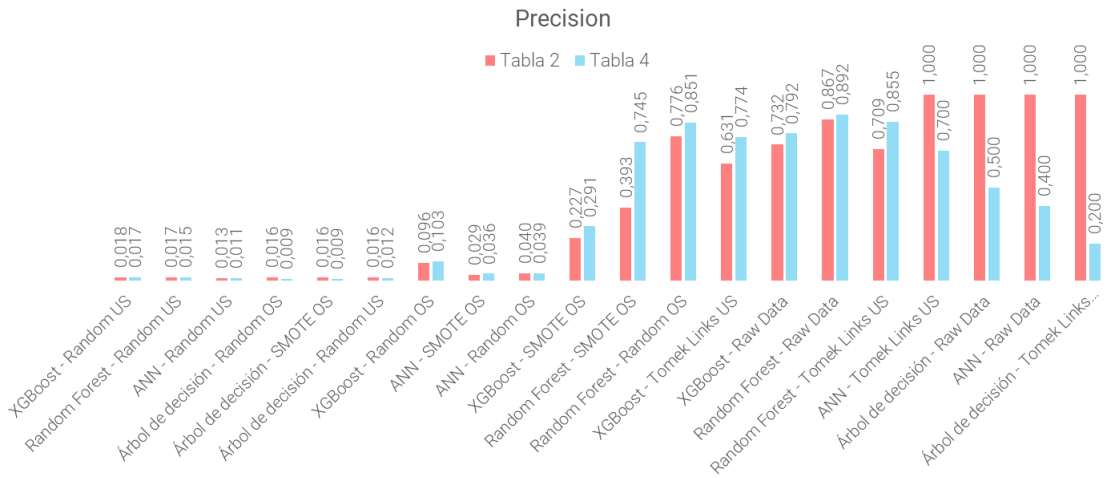


Figura 16 - Curva Recall / Precision

Comparando estos resultados contra los originales, vemos que el máximo valor para el *F1-Score* obtenido por algún modelo (*Random Forest* con técnica *SMOTE*) es de 0.343, 10 puntos porcentuales por encima. Es decir, hay una mejora significativa en la performance de los modelos. Poniendo ambos resultados (tabla 2 y 4) en contraste, vemos la siguiente evolución para las variables *precision*, *recall* y *FNR*:



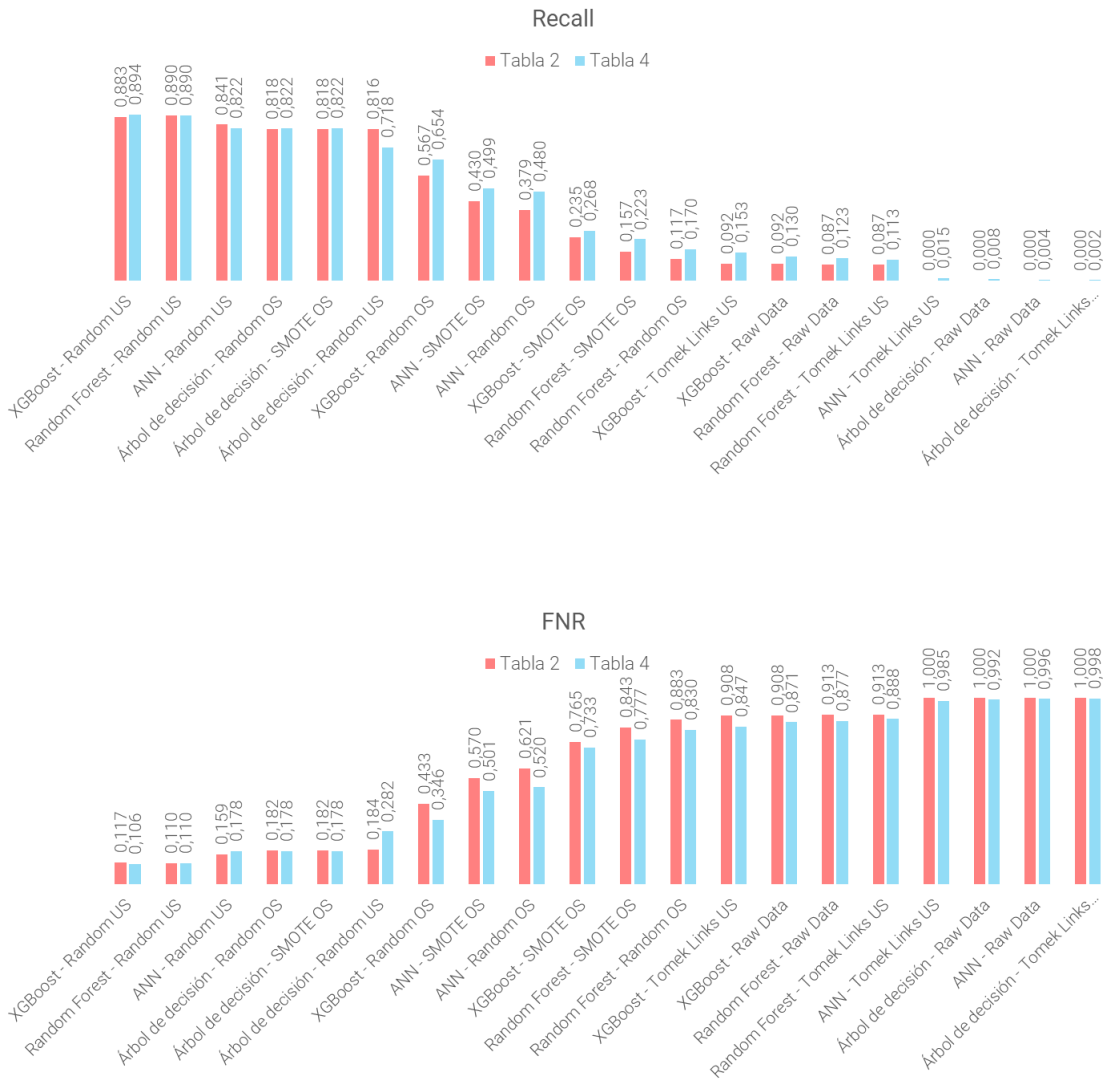


Figura 17 - Comparación resultados originales vs nuevas variables - elaboración propia

Lo que se observa con los dos primeros elementos de la Figura 17 es una pérdida generalizada en la métrica *precision* compensada con una ganancia en *recall*, lo que está alineado al objetivo perseguido en primer lugar: poder bajar el error en la identificación de los falsos negativos, aunque sea a costa de perder precisión en la identificación de los casos fraudulentos. Es decir, más transacciones genuinas serán incorrectamente clasificadas y deberán pasar por una prueba manual posterior, pero pasarán menos casos fraudulentos mal clasificados. Esto se termina de apreciar correctamente con la tercera componente, en donde se observa la diferencia para la tasa de falsos negativos, que es menor en todos los casos, a excepción de la red neuronal con *undersampling* (ANN - US Random Under Sampling) y lo

mismo en el árbol de decisión, en donde se duplica dicha tasa (Árbol de decisión - US Random Under Sampling).

En definitiva, los resultados son positivos en comparación con los originales, pero distan aún de acercarse a un modelo que logre, aunque sea, igualar o mejorar las tasas promedio del mercado informadas oportunamente en el estudio de Visa CyberSource en la Figura 1 (9,2% de pérdida de conversión por 1,7% de contracargos promedio para la región, que podrían emparejarse a las tasas FNR y el complementario de FPR, respectivamente).

4.4. Implementación

El despliegue de un modelo es el proceso por el cual conseguimos que el mismo pase a estar disponible en un entorno productivo, generando predicciones a partir de nuevos registros que se generarán con las transacciones en la plataforma (González, s.f.).

El modelo de *machine learning* se implementará como una capa adicional dentro de la arquitectura de prevención de fraude desarrollada en la empresa. A la luz de los resultados, se sugiere que sea posterior a un sistema basado en reglas (que puede ser desarrollado en forma interna) y anterior a una capa de evaluación manual de los resultados obtenidos en las dos capas (reglas y *machine learning*), buscando aumentar la conversión resultante de los anteriores.

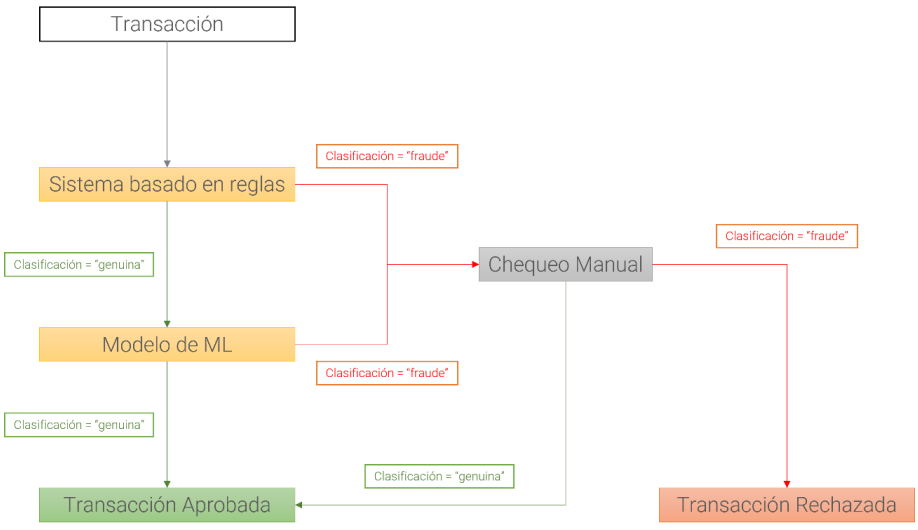


Figura 18 - Arquitectura de prevención de fraude propuesta

En esta línea, la recomendación es que tanto el sistema de reglas como el modelo desarrollado busquen que la cantidad de falsos negativos sea el mínimo posible (es decir, que no se etiquete a una transacción fraudulenta como genuina) para evitar el costo asociado a los contracargos futuros que se hayan podido filtrar por las distintas capas de prevención. La consecuencia de esto, que es una mayor tasa de falsos positivos, es lo que luego se tratará de mejorar con los chequeos manuales.

El proceso para la implementación del modelo en la plataforma se puede resumir en los siguientes pasos:

I. Desarrollar el modelo en un ambiente de entrenamiento: este paso consiste en todo lo expuesto hasta este momento, incluyendo la selección del algoritmo que, potencialmente, entregará los mejores resultados en función del objetivo perseguido.
II. Optimización y limpieza del código: así como en una primera instancia se crea código extenso para llevar adelante todas las pruebas, elección del modelo, incorporación de nuevas variables, etc. (y la mayoría de las veces se hace en forma colaborativa y transversal entre varias áreas de la compañía), en esta instancia se extraerán sólo las porciones de código que sean de estricta relevancia:
    a. Obtención de los datos
    b. Limpieza y pre-procesamiento
        i. Remoción de variables no deseadas
        ii. Remoción de *outliers* si fuera necesario
        iii. Trabajo sobre variables categóricas (con *OneHotEncoding*, por ejemplo)
        iv. Normalización / Escalamiento de las variables numéricas
    c. Entrenamiento del modelo
III. Preparar el código para la implementación en contenedores: el uso de un ambiente en *containers* (ampliamente utilizados desde la aparición y masificación de Docker),

permiten una mayor predictibilidad, repetitividad, inmutabilidad, así como una más fácil coordinación. Los *containers* simplifican el desarrollo y escalamiento de los modelos de *machine learning*, a la vez que permiten una modificación y actualización sencilla, que favorece el mantenimiento.

Lo que permiten plataformas como Docker es que no surjan inconvenientes a la hora de ejecutar el código por cualquier miembro del equipo, en distintas computadoras, en el servidor, etc. Es decir, independientemente del tipo de hardware o software que se esté utilizando en los equipos. En el caso de *machine learning*, al ser un proceso iterativo en el que los datos o mismo el modelo pueden cambiar, esto resulta de vital importancia. Por lo tanto, lo que se hace es empaquetar nuestra aplicación en un contenedor, simplificando el proceso de distribución (Sotaquirá, 2021). De no hacer esto, estaríamos introduciendo múltiples puntos de falla al proyecto, dado que lo más probable es que existan inconsistencias entre las librerías o dependencias del equipo en donde se ha desarrollado el modelo y los restantes. Adicionalmente, utilizando una plataforma de este tipo, todos los contenedores desarrollados se montan sobre un solo sistema operativo, haciendo que sean necesarios menos recursos informáticos y facilitando la portabilidad de los modelos.

IV. Monitoreo continuo y mantenimiento: será necesario pensar un esquema de mantenimiento continuo del modelo, más allá del *go live* inicial, garantizando su performance en el largo plazo. Es imprescindible revisar en forma periódica las métricas asociadas al modelo: tasa de aprobación, tasa de contracargos recibidos, tiempo de respuesta, etc. Por otro lado, dado que es esperable que el comportamiento fraudulento vaya mutando con el tiempo, será imprescindible garantizar que el modelo se vuelva a entrenar en forma periódica contando con nueva información (Klushin, 2022).

Un problema severo que existe en el caso de los fraudes por contracargo es la demora que existe entre que se realiza la transacción hasta que se efectúa el débito (debido a los períodos de espera en cualquier *marketplace* para que el comprador pueda hacer el reclamo, éste sea tomado por el comprador, se eleve la denuncia, el banco emisor inicie la investigación y resuelva). Esto hace que no sea posible un seguimiento muy reactivo en tiempo real de la tasa de contracargo, dado que el valor que veamos en el presente se asocia a un comportamiento fraudulento algunas semanas en el pasado. En consecuencia, es imprescindible el seguimiento en tiempo real de otras variables emparentadas que, si bien no están totalmente correlacionadas con fraude, sí pueden encender alguna alarma que permita una pronta detección y reacción o ser indicio de un cambio en el comportamiento de los usuarios de la plataforma. Por ejemplo, todas aquellas vinculadas al comportamiento de los compradores: cantidad de dispositivos, montos de compra promedio, cantidad de tarjetas registradas, vínculos con vendedores, velocidad de transacciones, etc. Incluso se puede complementar con servicios de terceros. Por ejemplo, un procesador de pagos que puede cruzar a los usuarios que operan en este sitio con sus compras en otros portales e incorporar así variables de entorno para poder prevenir fraudes en el *Marketplace* propio (variables que, de otra manera, sería imposible de obtener).

Conclusión

La investigación presenta el concepto de fraude, siendo aquel por contracargo el de especial interés, así como su impacto económico a nivel local, regional y global en la industria y cómo la tendencia es creciente y la preocupación de las empresas es cada vez mayor. En función de ello, se presenta el concepto de inteligencia artificial y cómo se pueden aprovechar desarrollos en el campo del aprendizaje automático para la implementación de arquitecturas de detección y prevención de fraude.

En los últimos años, con el acceso a nuevas tecnologías, formas de comercio y coyunturas particulares (pandemia global entre 2020 y 2021, por ejemplo), resulta imperativo que las empresas y organizaciones concentren esfuerzos en la prevención de fraude que, lejos de ser algo estático, las técnicas para llevarlo adelante se complejizan cada vez más y se adaptan reactivamente antes las nuevas defensas. Modelos como los ensayados, en conjunto con sistemas de reglas tradicionales y la adopción de nuevos métodos de pagos pueden conformar una buena barrera inicial, que habrá que realimentar con el paso del tiempo, incorporando nuevos patrones que vayan apareciendo. En este estudio se presentan ejemplos concretos de los dos primeros. En cuanto a métodos de pago, por ejemplo, se podría fomentar la adopción de pagos con criptomonedas que, por definición, no están sujetos a denuncias por contracargos: cuando una transacción es realizada, los fondos permanecen en el procesador hasta que la misma sea confirmada por ambas partes. Luego, no es posible revertirla (Chargeback Gurus, 2022).

Aunque los resultados obtenidos en el desarrollo experimental no permiten evitar el compromiso entre perder conversión o dejar pasar casos potencialmente fraudulentos, es importante resaltar que los mismos fueron ensayados con datos correspondientes a pocos meses de operación y que, además, pertenecen al lanzamiento de la plataforma en cuestión. En la medida en que ésta madure y se cuente con mayor historia y volumen de operaciones, ciertas variables como el histórico de transacciones, cantidad de dispositivos y tarjetas

registradas o contracargos previos por usuario, por citar algunas, irán sumando relevancia, realimentando el modelo y obteniendo progresivamente una defensa más robusta contra los fraudes futuros. Por otro lado, un equipo de analistas dispuesto para evaluar los falsos positivos (y ganar puntos adicionales de conversión) también irá ganando en experiencia, desarrollando reglas adicionales y analizando patrones recurrentes que permitirán automatizar, en parte, dicha tarea y mantener la carga de trabajo relativamente estable con el escalamiento de las órdenes.

El comercio electrónico continuará creciendo en los años por venir y el foco en la prevención y detección temprana de los usuarios fraudulentos estará presente en la agenda de la mayoría de las plataformas de comercio *online*. Por otro lado, en la medida en que se vayan desarrollando sistemas más sofisticados y algoritmos de detección más precisos, también los delincuentes aprenderán y se adaptarán, haciendo que la implementación de capas sucesivas y el reaprendizaje de los modelos sean imperativos.

Referencias

Albon, C. (2018). Python Machine Learning Cookbook Practical Solutions from Preprocessing to Deep Learning. Sebastopol, California: O'Reilly.

Benchaji, I., Douzi, S., & Ouahidi, B. E. (2019). Using Genetic Algorithm to Improve Classification of Imbalanced Datasets for Credit Card Fraud Detection: Methos and Protocols. *Springer Nature Switzerland*, 220-229.

Bhandari, A. (2020, Junio 16). *Analytics Vidhya*. Retrieved from AUC-ROC Curve in Machine Learning Clearly Explained: https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

Bhatt, S. (2018, Marzo 19). *Towards Data Science*. Retrieved from Reinforcement Learning 101: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

Bolton, R. J., & Hand, D. J. (2022). Statistical Fraud Detection: A Review. *Statistical Science*, 17 (3) 235 - 255.

Borgne, Y.-A. L., Siblini, W., Lebichot, B., & Bontempi, G. (n.d.). *Reproducible Machine Learning for Credit Card Fraud detection - Practical Handbook*. Retrieved from https://fraud-detection-handbook.github.io/fraud-detection-handbook/Foreword.html#

Breiman, L. (1996). *Bagging Predictors*.

Breiman, L. (2001). *Random Forests*.

Chargeback Gurus. (2022, Agosto 12). *Chargeback Gurus*. Retrieved from Cryptocurrency Chargebacks: What Merchants Should Know: https://www.chargebackgurus.com/blog/chargebacks-more-volatile-complex-than-cryptocurrency

Chen, C., & Breiman, L. (2004). *Using Random Forest to Learn Imabalanced Data*. Berkeley: University of California.

Compte, J. M. (2022, Febrero 22). Mercado Libre imparable: cuanto ganó en su año record de usuarios y transacciones. *El Cronista*, p. https://www.cronista.com/negocios/647041/.

Cornell Law School. (n.d.). *Legal Information Institute*. Retrieved from https://www.law.cornell.edu/wex/es/fraude_con_tarjeta_de_cr%C3%A9dito

Cubbo. (n.d.). *Cubbo Insights*. Retrieved from Fraude en ecommerce, lo que debes saber sobre el contracargo: https://www.cubbo.com/posts/fraude-en-ecommerce-contracargo

Dal Pozzolo, A., Bontempi, G., & Caelen, O. (2015, 8 29). When is undersampling effective in unbalance classification tasks? *ECML PKDD 2015* (pp. 200-215). Porto: Springer International Publishing Switzerland.

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2017). Credit Card Fraud Detection: a Realistic Modeling. *IEEE Transactions on Neural Networks and Learning Systems*, PP. 1-14 10.1109/TNNLS.2017.2736643.

DeepMind. (n.d.). *AlphaGo*. Retrieved from https://www.deepmind.com/research/highlighted-research/alphago

Delua, J. (2021, Marzo 12). *IBM Blog*. Retrieved from Supervised vs. Unsupervised Learning: What's the Difference?: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

Deshpande, P., Hasan Siddiqi, A., Khursheed, A., & Parmar, K. (2016). Applications of Data Mining Techniques for Fraud Detection in Credit-Debit Card Transactions. *National Conference on Technological Advancement and Automatization in Engineering* (pp. 339-345). IJSRD.

developers, x. (n.d.). *XGBoost Documentation*. Retrieved from https://xgboost.readthedocs.io/en/stable/index.html

Dhingra, D. (2021, Mayo 21). *Anlytics Vidhya*. Retrieved from Beginners Guide to Artificial Neural Network: https://www.analyticsvidhya.com/blog/2021/05/beginners-guide-to-artificial-neural-network/

Faraji, Z. (2022). A Review of Machine Learning Applications for Credit Card Fraud Detection with A Case study. *SEISENSE Journal of Management, 5*(1), 49-59.

Fernández, A., García, S., Galar, M., Prati, R. C., & Krawczyk, B. (2018). *Learning from Imbalanced Datasets.* Granada: Springer.

Gobierno de México. (2022). *Conducef Estadísticas*. Retrieved from https://www.condusef.gob.mx/?p=estadisticas

González, L. (n.d.). *AprendeIA*. Retrieved from Guía para implementar los modelos de machine learning: https://aprendeia.com/guia-para-implementar-los-modelos-de-machine-learning/#:~:text=El%20despliegue%20o%20implementaci%C3%B3n%20de,a%20otros%20sistemas%20de%20software.

Google. (n.d.). *Machine Learning Foundational Courses*. Retrieved from Classification: Accuracy : https://developers.google.com/machine-learning/crash-course/classification/accuracy?hl=es_419

Grbovic, N. (2022, Junio 3). *Hyperon*. Retrieved from Why Choose a Rules-Based Fraud Prevention Solution: https://www.hyperon.io/blog/why-choose-a-rules-based-fraud-prevention-solution

Hamelers, L. (2021, Enero 16). Detecting and Explaining Potential Financial Fraud Cases in Invoice Data with Machine Learning. Enschede, Países Bajos: University of Twente.

He, H., & Ma, Y. (2013). Imbalance Learning. Foundations, Algorithms and Applications. IEEE Press.

IBM. (2020, Junio 3). *IBM Cloud Learn Hub*. Retrieved from Artificial Intelligence (AI): https://www.ibm.com/cloud/learn/what-is-artificial-intelligence

IBM. (n.d.). *Data and AI-driven analytics*. Retrieved from https://www.ibm.com/analytics

Imbalance Learn. (n.d.). Retrieved from Feature importances with a forest of trees: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

Interbank Perú. (n.d.). *Fraude de tarjetas*. Retrieved from https://interbank.pe/que-es-el-fraude-de-tarjetas-y-como-se-realiza

Karczewski, J. (n.d.). *Nethone*. Retrieved from Machine Learning Models vs. Rule Based Systems in fraud prevention: https://nethone.com/post/machine-learning-models-vs-rule-based-systems-in-fraud-prevention

Kavlakoglu, E. (2020, Mayo 27). *IBM Blog*. Retrieved from AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Klushin, P. (2022, Abril 20). *Qwak*. Retrieved from What does it take to deploy ML models in production?: https://www.qwak.com/post/what-does-it-take-to-deploy-ml-models-in-production

Kount. (n.d.). *Rule based fraud detection*. Retrieved from https://kount.com/glossary/rule-based-fraud-detection/

Kumar, B. (2020, Julio 23). *Analytics Vidhya*. Retrieved from 10 Techniques to deal with Imbalanced Classes in Machine Learning: https://www.analyticsvidhya.com/blog/2020/07/10-techniques-to-deal-with-class-imbalance-in-machine-learning/

Lantz, B. (n.d.). *packt*. Retrieved from Divide and Conquer – Classification Using Decision Trees and Rules: https://www.packt.com/divide-and-conquer-classification-using-decision-trees-and-rules/

Li, J. (2022, Mayo 9). E-Commerce Fraud Detection Model by Computer Artificial Intelligence Data Mining. (A. Bhardwaj, Ed.) *Hindawi. Computational Intelligence and Neuroscience, 2022*, 1-9.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2009). Isolation Forest.

Mazzanti, S. (2021, Julio 4). *Towards Data Science*. Retrieved from “Isolation Forest”: The Anomaly Detection Algorithm Any Data Scientist Should Know: https://towardsdatascience.com/isolation-forest-the-anomaly-detection-algorithm-any-data-scientist-should-know-1a99622eec2d

McCarthy, J. (2004, Noviembre 24). What is Artificial Intelligence? Stanford, California, Estados Unidos: Stanford University.

Minastireanu, E.-A., & Mesnita, G. (2019). An Analysis of the Most Used Machine Learning Algorithms for Online Fraud Detection. *Informatica Economica, 23*(1/2019), 5-16.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python.* (D. Schanafelt, Ed.) Sebastopol: O'Reilly.

Nogueira, F., Lemaitre, G., Victor, D., & Aridas, C. (2022, Mayo 17). *Imbalanced Learn*. Retrieved from SMOTEENN: https://imbalanced-learn.org/stable/references/generated/imblearn.combine.SMOTEENN.html

Opitz, D., & Maclin, R. (1999). Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research, 11*, 169-198.

Oxford University Press. (n.d.). *Oxford Reference*. Retrieved from https://www.oxfordreference.com/view/10.1093/oi/authority.20110803095833457#:~:text=A%20type%20of%20dishonesty%20calculated,(a%20tort%20or%20crime).

Panneerselvam, L. (2021, Abril 14). *Analytics Vidhya*. Retrieved from Activation Functions and their Derivatives – A Quick & Complete Guide: Activation Functions and their Derivatives – A Quick & Complete Guide

PwC. (2022). *PwC's Global Economic Crime and Fraud Survey 2022.* Retrieved from Protecting the perimeter: The rise of external fraud: https://www.pwc.com/fraudsurvey

Ravelin. (n.d.). *Ravelin Insights*. Retrieved from Machine learning: https://www.ravelin.com/insights/machine-learning-for-fraud-detection

Real Academia Española. (n.d.). *Diccionario de la lengua española*. Retrieved from https://dle.rae.es/fraude

Rocca, B. (2019, Enero 27). *Towards Data Science*. Retrieved from Handling imbalanced datasets in machine learning: https://towardsdatascience.com/handling-imbalanced-datasets-in-machine-learning-7a0e84220f28

Russel, S., & Norvig, P. (2009). *Artificial Intelligence. A modern approach.* New Jersey: Pearson.

Sharma, P., Banerjee, S., Tiwari, D., & Patni, J. C. (2021, Noviembre). Machine Learning Model for Credit Card Fraud Detection - A Comparative Analysis. *The International Arab Journal of Intormation Technology, 18*(6), 789-796.

Shopify. (2016, Octubre 21). *Shopify Blog*. Retrieved from What To Do When Customers Force Refunds and Chargebacks: https://www.shopify.com/blog/credit-card-chargebacks

Shukairy, A. (2022, Mayo 10). *invesp*. Retrieved from E-commerce Fraud And Chargebacks – Statistics And Trends: https://www.invespcro.com/blog/e-commerce-fraud-and-chargebacks-infographic/

Sigrist, F. (2022, Enero 25). *Towards Data Science*. Retrieved from Demystifying ROC and precision-recall curves: https://towardsdatascience.com/demystifying-roc-and-precision-recall-curves-d30f3fad2cbf

Sotaquirá, M. (2021, Abril 24). *codificandobits*. Retrieved from ¿Qué es y porqué usar Docker en el Machine Learning?: https://www.codificandobits.com/blog/docker-machine-learning/

Statista. (2022, Marzo 25). *Value of e-commerce losses to online payment fraud worldwide in 2020 and 2021*. Retrieved from https://www.statista.com/statistics/1273177/ecommerce-payment-fraud-losses-globally/

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning.* Cambridge, Massachussetts: The MIT Press.

Tahir, M. A., Kittler, J., & Yan, F. (2012). Inverse random under sampling for class imbalance problem and its application to multilabel classification. *Pattern Recognition, 45*, 3738–3750.

Tiendanube. (2022, Julio 15). *tiendanube blog*. Retrieved from Gestión de los contracargos: qué son y cómo evitarlos: https://www.tiendanube.com/blog/que-son-los-contracargos-y-como-evitarlos/

Visa Cybersource. (2017). *Reporte de fraude online América Latina 2017.* Retrieved from Visa Merchant Sales & Solutions: https://www.visa.com.sv/content/dam/VCOM/regional/lac/SPA/Default/Documents/PDFs/online-fraud-report-01.pdf

Yiu, T. (2019, Julio 20). *Towards Data Science*. Retrieved from The Curse of Dimensionality: https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e

Autorizaciones

- Repositorio Institucional UCEMA: autorizo a la Universidad del CEMA a publicar y difundir el trabajo final de mi autoría en el Repositorio Institucional UCEMA de la Biblioteca con fines exclusivamente académicos y didácticos.

- Catálogo en línea: autorizo a la Universidad del CEMA a publicar y difundir el trabajo final de mi autoría en el catálogo en línea de la Biblioteca (acceso con usuario y contraseña) con fines exclusivamente académicos y didácticos.

- Página web: autorizo a la Universidad del CEMA a publicar y difundir el trabajo final de mi autoría en la página web de la Universidad como trabajo destacado (si obtuviese la distinción correspondiente) con fines exclusivamente académicos y didácticos.

Nombre y apellido: José Ignacio López Sáez
DNI: 32737997
Carrera: MADE
Firma:

Autorizaciones

- Repositorio Institucional UCEMA: autorizo a la Universidad del CEMA a publicar y difundir el trabajo final de mi autoría en el Repositorio Institucional UCEMA de la Biblioteca con fines exclusivamente académicos y didácticos.

- Catálogo en línea: autorizo a la Universidad del CEMA a publicar y difundir el trabajo final de mi autoría en el catálogo en línea de la Biblioteca (acceso con usuario y contraseña) con fines exclusivamente académicos y didácticos.

- Página web: autorizo a la Universidad del CEMA a publicar y difundir el trabajo final de mi autoría en la página web de la Universidad como trabajo destacado (si obtuviese la distinción correspondiente) con fines exclusivamente académicos y didácticos.

Nombre y apellido: José Ignacio López Sáez

DNI: 32737997

Carrera: MADE

Firma: