Detección de Fraude en Transacciones Comerciales de Clientes Aplicando Métodos de ML

Laura Andrea Florez Bedoya

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor
Javier Fernando Botia Valderrama, Doctor (PhD) en Ingeniería Electrónica

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos

Medellín, Antioquia, Colombia

2021

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Después de realizar una revisión de los métodos de aprendizaje automático que han sido reportados en diferentes literaturas para la predicción de fraude se ha encontrado que con ayuda de la aplicación de técnicas de procesamiento es posible encontrar una ruta adecuada para dar solución a una problemática tan actual como lo es el fraude en comercios electrónicos. Entre los métodos de ML evaluados es posible mencionar especialmente, las técnicas de aprendizaje supervisado, cuyos modelos de clasificación pueden etiquetar eventos como buenos (donde las transacciones son genuinas) o malos (cuando el comportamiento asociado no es correcto o fraudulento).

Además de la aplicación de los modelos supervisados se hacen uso también, de técnicas de procesamiento que por medio de los mismos datos realizan transformaciones y nuevas inferencias que puedan describir con mayor detalle el comportamiento de un cliente a lo largo de una transacción o en el establecimiento de señales de fraude. Estas implementaciones incluyen la generación de características o el agrupamiento de variables sobre información relevante en una transacción electrónica como, las formas de pago o la identidad de un usuario. Durante la investigación se examinó con detalle el uso de diferentes métodos y subprocesos de ML con el fin de determinar cuál es la mejor opción para lograr una adecuada predicción de fraude sobre una fuente abundante de información recopilada (con gran dimensionalidad). La modelización incluyó un análisis de los modelos y del efecto de algunos métodos de procesamiento (como ingeniería de características, escalamiento y ajuste de desbalanceo) sobre la predicción de casos fraudulentos, evaluando su rendimiento por medio de distintas métricas de clasificación.

Los resultados mostraron que al hacer uso del ajuste de hiperparámetros para métodos de aprendizaje supervisado en conjunto con la aplicación de técnicas de procesamiento es posible obtener modelos optimizados que presenten buen comportamiento de predicción en evento fraudulentos. Un factor importante que se observó fue la influencia del escalamiento sobre la predicción, donde la evaluación de dichos modelos con ayuda de distintas métricas evidenció la influencia directa de estas transformaciones numéricas sobre los verdaderos negativos hallados (que representan los casos de fraude). Los resultados de la modelización y del análisis realizado se presentan de forma que puedan ser usados a futuro en estudios más profundos sobre la detección de fraude.

ABSTRACT

After reviewing the machine learning methods that have been reported in different literatures for fraud prediction, it has been found that with the help of the application of processing techniques it is possible to find an adequate route to provide a solution to such a current problem as fraud in electronic commerce. Among the ML methods evaluated, it is possible to mention especially supervised learning techniques, whose classification models can label events as good (where the transactions are genuine) or bad (when the associated behavior is not correct or fraudulent).

In addition to the application of supervised models, processing techniques are also used, where using the same data, it performs transformations and new inferences that can describe in greater detail the behavior of a customer throughout a transaction or in the establishment of fraud signals. These implementations include the generation of features or the grouping of variables about relevant information in an electronic transaction such as, payment methods or the identity of a user. During the research, the use of different ML methods and sub-processes were examined in detail in order to determine which is the best option to achieve adequate fraud prediction on an abundant source of collected information (with high dimensionality). The modeling included an analysis of the models and the effect of some processing methods (such as feature engineering, scaling and unbalance adjustment) on the prediction of fraudulent cases, evaluating their performance by means of different classification metrics.

The results showed that by making use of hyperparameter tuning for supervised learning methods in combination with the application of processing techniques, it is possible to obtain optimized models that exhibit good prediction behavior in fraudulent events. An important factor observed was the influence of scaling on prediction, where the evaluation of these models with the help of different metrics evidenced the direct influence of these numerical transformations on the true negatives found (representing fraud cases). The results of the modeling and analysis performed are presented so that they can be used in the future for more in-depth studies on fraud detection.

I. INTRODUCCIÓN

Con la creciente digitalización de los servicios y las nuevas tecnologías que se van desarrollando, la cantidad de transacciones electrónicas ha ido en aumento de forma bastante considerable, generando movimientos por miles de millones en los diferentes sectores productivos globales. Entre los diferentes métodos de pago que al día de hoy son usados, las tarjetas débito y crédito son uno de los productos financieros más usados para el pago de operaciones en línea (independiente de si es un país desarrollado o no). Con la proliferación de estos medios de pago, el sector financiero también se ha debido enfrentar con la evolución del fraude financiero, que actualmente causa millonarias pérdidas a los sistemas bancarios. Tal cual, como lo reporta en su artículo Awoyemi et al. [1], entre las técnicas que han tomado un papel imperativo en el sector productivo han sido tanto la minería de datos y el aprendizaje automático, los cuales permiten generar aplicaciones enfocadas en los conjuntos de datos, la selección de variables y los métodos de detección usadas.

Con base en lo anterior, el presente estudio pretende avanzar en la exploración de métodos de aprendizaje automático (supervisados o no) que puedan colaborar con la construcción de sistemas de identificación de fraude electrónico. El enfoque principal de la investigación será recopilar información de rendimiento sobre varios modelos de clasificación y su evaluación bajo diferentes métricas que sean correspondientes al problema que se abarca. El trabajo ha consistido en, una revisión genérica de las distintas soluciones para detectar fraude reportadas en diferentes literaturas, lo cual asienta un precedente inicial sobre los modelos de entrenamiento que se pudieran considerar, por otro lado, los demás capítulos de la investigación se centran en el desarrollo de la metodología de entrenamiento y predicción de modelos.

Uno de los retos característicos de estas situaciones de fraude se concentra en la identificación de perfiles de comportamiento y la tendencia de que los datos financieros siempre estén sesgados (es decir, con un desbalanceo de etiquetas). Por lo tanto, es necesario examinar con detalle algunos métodos ya existentes para controlar el desbalanceo y trabajar sobre cómo identificar patrones de fraude (que está completamente ligado a la aplicación de ingeniería de características).

II. MARCO TEÓRICO

Con la llegada de los ordenadores, el internet y los avances que cada día se generan en términos de tecnología, se ha podido evidenciar el crecimiento exponencial de la información, así como la demanda para generar entornos capaces de almacenar dichos datos continuamente, originando nuevos retos tanto de recursos como de conocimiento. Dentro de esta sección se describirá de forma concisa los conceptos básicos necesarios para el desarrollo de este proyecto, desde un compendio que contenga revisiones bibliográficas hasta llegar a un planteamiento de soluciones para la problemática objetivo.

A. Preliminares

El término Big Data es tan genérico y amplio que el origen de su etimología se remonta a muchas décadas atrás (incluso siglos), los cimientos de su historia datan de las primeras recolecciones de información en forma de cálculos y almacenamiento masivo de pergaminos y libros de las primeras bibliotecas de las antiguas civilizaciones. Con el auge de los ordenadores como un cuerpo informático en constante expansión, durante las últimas décadas (a partir de 1960), se empezaron a mostrar los primeros esfuerzos para describir la explosión de información que se preveía para los años siguientes. Y es que no fue hasta 1977, que M. Cox y D. Ellsworth en su publicación [2], usaran el término de Big Data bajo un contexto moderno haciendo referencia al imponente reto que implicaría el crecimiento tecnológico de los sistemas informáticos y de información hacia los recursos actuales de almacenamiento bastantes limitados para esa época.

Actualmente, la transformación digital ha llevado a producir quintillones de bytes de información, los cuales son cada vez más difíciles de organizar, almacenar y analizar. En esencia, para las compañías y diferentes sectores productivos con la potencialidad del Big Data se ha podido usufructuar la agregación masiva de datos ayudando a reducir costes, tomar decisiones y predecir tendencias. En este ámbito, se puede relacionar la definición del Big Data como al estallido tanto en cantidad como calidad de datos disponibles provenientes de diferentes fuentes (desde redes sociales, compras en línea, transacciones financieras y los dispositivos de red); al mismo tiempo, las tecnologías han ido evolucionando hasta comprender herramientas e infraestructuras capaces de abordar el procesamiento de datos

hasta ser aplicado en múltiples sectores productivos [3]. Es por esta razón que el Big Data está en todas partes. Dado su amplia aplicación, los grandes actores del mercado y la industria pueden manejar un sinfín de datos con propósitos de negocio o para procesos meramente analíticos. La gestión de información, de contenidos, el almacenamiento, visualización e integración de datos hacen del Big Data un sistema integral compuesto de herramientas y tecnologías que ayudan a la toma de decisiones estratégicas [4]. En particular, es tanta la importancia del Big Data en la actualidad que, varios campos y sectores de la industria son cada vez más conscientes de que el análisis de datos masivos se está convirtiendo en un aspecto indispensable para ser competitivos, para revelar nuevos conocimientos y para individualizar los servicios; lo que eventualmente demanda la adopción de dicha tecnología para entender mejor su entorno productivo a partir de sus propios datos.

Debido al interesante valor que puede extraerse del Big Data, esta tecnología se ha convertido en una tendencia creciente, no sólo por su utilidad como datos, sino también para la resolución de problemas relacionados tanto al sector público como al privado. Dentro de su naturaleza distintiva, De Mauro et al. [5], describe el Big Data bajo tres características principales: velocidad, volumen y variedad. La velocidad se refiere a la rapidez de generación y transferencia de información, lo que genera retos para las herramientas de procesamiento de datos que se ajusten a dicha demanda; el rasgo de volumen hace relación a la cantidad de datos que se producen en un margen de tiempo (ya sea información pasada o en tiempo real), lo que desafía constantemente las tecnologías de almacenamiento de datos y la variedad que se refiere a la heterogeneidad y diversidad de los datos (en termino de fuentes, temas o contextos). Más adelante, debido al auge del Big Data en el mundo, se caracterizarían más tipologías de la información masiva como: variabilidad, veracidad, valor y visibilidad [6]. Estos últimos, considerados como las características adicionales y modernas del Big Data, entran en detalle sobre el valor intrínseco de la información, la cual constantemente va cambiando, no sólo en términos de valor sino también de acuerdo a su condición de verdad [6].

En el marco del Big Data, uno de los principales desafíos a tratar, es el procesamiento, transformación y análisis de la información masiva; que con la ayuda del aprendizaje de máquina (ML) es posible implementar sistemas que aprendan de esos mismos datos, identifiquen patrones que a simple vista no se ven y permitan tomar decisiones de cualquier

índole sin intervención humana. Dicha integración (entre el Machine Learning y el Big Data) resulta altamente importante, puesto que, con el aumento del tamaño de los conjuntos de datos que alimentan los algoritmos predictivos se pueden filtrar y generar resultados más precisos que afectan la calidad de un servicio, la operación comercial de una empresa o incluso las relaciones con los clientes, entre otras cosas [7].

B. Incursiones del Machine Learning en la detección de fraude

Al día de hoy, la inteligencia artificial abarca una amplia rama de los desarrollos tecnológicos actuales, los cuales involucran la construcción de máquinas inteligentes capaces de realizar tareas que normalmente requieren la comprensión humana o que simulan su comportamiento, dentro de dichas capacidades, existe un amplio conjunto de técnicas o subcampos que describen a grosso modo gran parte de las aplicaciones de la industria y la ciencia, como la robótica, el procesamiento de lenguaje natural hasta la toma de decisiones. El aprendizaje automático (conocido también como Machine Learning), en particular, permite construir soluciones a cualquier tipo de problemáticas con base en algoritmos soportados en datos; de esta forma, se les puede dar la capacidad a las máquinas de descifrar, ejecutar y analizar información de un conjunto determinado de datos [8]. De esta forma, las máquinas pueden ser entrenadas para interpretar grandes volúmenes de datos y predecir nueva información por medio de modelos de aprendizaje estructurados. El efecto del aprendizaje automático se ha sentido ampliamente en una serie de industrias relacionadas con cuestiones de uso intensivo de datos, transformando muchos segmentos, incluidos los servicios financieros, la educación, el transporte, entre otros. El aprendizaje automático ha evolucionado a partir del reconocimiento de patrones y algoritmos de análisis que aprenden a partir de los datos, haciendo posible la fabricación de soluciones, predicciones o decisiones basadas en ellos; su implementación va a tal velocidad que cada vez más se utiliza al ML como un medio para el mejoramiento de objetos, gadgets y servicios que convergen a sistemas inteligentes que se personalizan y adaptan a partir de la experiencia [9].

Dentro de los enfoques del Machine Learning, ya sea con métodos de aprendizaje supervisado o no supervisado, una de las tareas más usadas dentro de este campo tiene que ver con problemas de regresión o clasificación. Especialmente esta última, va usando patrones en la información para asignar estos mismos datos a ciertas clases. La clasificación

es un proceso de categorización de un conjunto dado de datos de acuerdo a etiquetas preestablecidas, el algoritmo planteado aprende de los datos entregados y realiza inferencias sobre las posibles clasificaciones donde puede ser categorizada cada dato. Para la tarea de clasificación, existen dos tipos de categorización que se puedan aplicar a las soluciones analíticas: i) clasificación binaria (que etiqueta en base a sólo dos clases) y ii) clasificación multiclase (que presenta más de dos clases) [9].

Una de las muchas aplicaciones de la clasificación con aprendizaje automático recae en la detección de fraude, cuyo algoritmo normalmente funciona analizando los patrones de usuarios/consumidores, métodos de transacción y otras características para determinar la existencia de una acción fraudulenta. Debido a la rápida reacción del algoritmo programado, es posible identificar de forma oportuna cualquier desviación del comportamiento normal de una transacción en tiempo real, siendo un aporte potencial y rentable para empresas públicas y privadas como las que hacen parte del sector financiero. Por ejemplo, en el reporte anual 2020 entregado por PwC "PricewaterhouseCoopers" [10], el sector de los servicios financieros y las industrias que implican el uso de transacciones financieras hacen parte de las empresas con mayores daños relacionados con fraude, obteniendo más de US$42 B (42.000 millones de dólares) en pérdidas únicamente por fraude en los últimos 24 meses. Dentro de los tipos de fraude en este escenario, el 39% de las compañías financieras encuestadas afirmó que el fraude de clientes (a partir de autores externos) y la ciberdelincuencia eran los delitos más perjudiciales. La situación anterior, realza la necesidad de las empresas de utilizar tecnología destinada a la identificación y lucha contra el fraude, implementando no sólo programas antifraude sino soluciones y herramientas informáticas capaces de procesar cantidades masivas de datos y de predecir con precisión cualquier eventualidad anormal dentro de su sistema financiero [11].

Puesto que la intención inicial de este trabajo es la detección de fraude en transacciones comerciales, durante el desarrollo de la investigación se utilizarán algoritmos de aprendizaje (supervisado y no supervisado) de clasificación binaria, ya que el objetivo del caso de estudio es predecir una etiqueta de fraude o no fraude para transacciones comerciales de una base de datos otorgada por la plataforma Kaggle. En las secciones siguientes se

describirá con más detalle, los métodos de clasificación, las herramientas computacionales a usar y la metodología de la investigación.

C. Métodos y algoritmos enfocados al fraude en escenarios financieros

Con el desarrollo y la expansión del comercio electrónico y la transformación digital de muchas compañías, el fraude por su parte ha adoptado también nuevas formas de hallar los puntos débiles de los sistemas de seguridad para cometer fechorías. El fraude en transacciones electrónicas, las cuales involucran el uso de tarjetas debido/crédito, pueden ser categorizadas de acuerdo con la naturaleza de la actividad fraudulenta, donde es posible encontrar desde robo simple (fraude fuera de línea), solicitudes fraudulentas, fraude bancario hasta fraude de falsificación por comportamiento [12]. De todos, el último es uno de los más perjudiciales debido a que la detección de este tipo de fraude requiere procesos de identificación sofisticados para encontrar comportamientos anormales en transacciones realizadas a distancia (ventas por móvil, en línea, etc.) donde el titular de la tarjeta no necesariamente este presente. Entre los diferentes enfoques para combatir las transacciones ilícitas es posible observar el uso de sistemas expertos basados en reglas o determinísticos (una aproximación de seguridad más tradicional), el cual está configurado para automatizar los procedimientos y comprobaciones sobre las transacciones que se realizan, y por otro lado, los algoritmos de aprendizaje automático están siendo implementados actualmente para abordar con mayor eficiencia la automatización en procedimientos de seguridad además de ser capaces de detectar irregularidades o cambios tenues que puedan indicar un acto fraudulento [13].

El principio que explica la detección del fraude con ayuda del aprendizaje automático consiste en que las transacciones fraudulentas tienen características específicas que no tienen las transacciones legítimas. Basándose en este postulado, los algoritmos de aprendizaje automático pueden detectar patrones en las operaciones financieras y decidir si una determinada transacción es fidedigna. La clasificación de las transacciones (de comercio electrónico) con tarjetas ya sean de débito o crédito es, especialmente, un problema de clasificación binaria. Donde cada transacción electrónica es evaluada con base a dos clases: si una transacción puede ser lícita (clase negativa o cero) o fraudulenta (clase positiva o uno). Diferentes literaturas reportadas han incursionado en la discusión sobre la detección de

fraude haciendo uso de algoritmos de aprendizaje automático supervisados (con entradas/salidas del algoritmo definidas) y algoritmos de aprendizaje no supervisados (que hacen uso de datos no etiquetados) [14].

En el informe técnico descrito por Tiwari et al. [15], se presenta una revisión detallada de varios métodos utilizados para detectar fraudes con tarjetas de crédito. Como parte de las metodologías planteadas para implementar soluciones contra el fraude, se mencionan técnicas como árboles de decisión, regresión logística, máquinas de vectores de soporte (SVM), redes neuronales, Random Forests y Naive Bayes. A partir del análisis exhaustivo, los autores resumen los pros y los contras de las técnicas anteriormente mencionadas con respecto a su aplicación en un escenario financiero. Por otro lado, se investiga también la aplicación de diferentes métodos de clasificación supervisado en la detección de fraude [1], la comparación entre los diferentes planteamientos se hace a través de su rendimiento, usando algoritmos tales como: Naive bayes, K-vecinos más cercanos (KNN) y regresión logística en datos de fraude de tarjetas de crédito muy sesgados (datos condicionados e incompletos). El rendimiento de las técnicas se evalúa en función a varias métricas desde exactitud, sensibilidad, precisión, entre otros, los resultados generales muestran una precisión de clasificación con Naive bayes, KNN y regresión del 97,92%, 97,69% y 54,86% respectivamente, mostrando con la efectividad de algunos clasificadores sobre otros. Así mismo, otro estudio realizó comparaciones en rendimiento de varios modelos de aprendizaje automático [16]. Los resultados mostraron, que la regresión logística y el método SVM tienen buen rendimiento en relación con una red neuronal artificial, en particular en casos donde las observaciones y características son pocas. Los anteriores reportes amplían la investigación sobre el fraude y las técnicas que pueden ser implementadas en la detección e identificación sobre transacciones electrónicas. A nivel general, los diferentes algoritmos encontrados en la literatura para la aplicación en escenarios financieros se listan a continuación [17]:

- **Random Forest:** Algoritmo que usa como base el método de árboles de decisión. En términos generales, ordena las clases de acuerdo a las categorías que considera similares. En ese sentido, Random Forest construye una multitud de árboles de decisión, promedia la información y hace la clasificación al árbol más cercano.

- **Regresión logística:** Descrito como un cálculo en base a probabilidades o connotaciones (de variables dependientes e independientes) que se utiliza para predecir un resultado binario: o sucede algo, o no sucede.

- **K-nearest neighbors:** Es un algoritmo de reconocimiento de patrones, donde a partir de los datos de entrenamiento encuentra K parientes más cercanos para clasificar luego datos futuros.

- **Naive Bayes:** Clasificador probabilístico que usa el teorema de Bayes para calcular la posibilidad/probabilidad de que un dato determinado pertenezca o no a una dada categoría.

- **Otros algoritmos como GradientBoosting y XGBoost:** Gradient Boosting por ejemplo, es un algoritmo que minimiza una función de pérdida eligiendo varias veces sobre funciones con gradiente negativo, que representen una hipótesis débil. Por otro lado, XGBoost, es una forma más regularizada de Gradient Boosting, el cual utiliza parámetros de regularización como L1 y L2.

D. Técnicas de ingeniería de características para datasets de gran volumen

Como prerrequisito antes de la construcción de un modelo analítico, se requiere por lo general, de la generación de conocimiento sobre el problema a resolver, ya sea, recurriendo al contexto particular, la inferencia de nuevas características que describen y dependen del tipo de tarea o con la transformación de observaciones. Estos pasos de preprocesamiento, que convierten datos en bruto en información/características más significativas para su uso en algoritmos de aprendizaje automático, es lo que se conoce como ingeniería de características [18]. Como tarea principal en la preparación de datos, esta práctica permite hacer transformaciones sin ningún tipo de límite contextual sobre cada característica de un conjunto de datos. Por ejemplo, en aplicaciones de ingeniería es común el tener que usar funciones matemáticas, varios tipos de operadores aritméticos e incluso de agregación con el fin de generar nuevas variables que generen una mejor relación entre la característica y la clase objetivo a predecir (lo que eventualmente facilita el aprendizaje de los algoritmos sobre nueva información). Ya que precisamente, la adaptación de la ingeniería de características

implica variado tipo de conversiones de información, es posible resumir los tipos de aproximación de ingeniería para abordar las diferentes necesidades de transformación que requiera una solución en machine Learning. Por esta razón, existe una colección de técnicas que se utilizan durante estas etapas de preprocesamiento, las cuales incluyen, la creación, transformación, extracción y selección de características (o variables), normalmente utilizadas crear algoritmos de ML en cualquier campo [19]. Conforme a lo anterior, es posible describir las posibles aplicaciones de la ingeniería de características de la siguiente manera [20]:

- **Creación:** A partir de la exploración y el análisis de los rasgos de los datos es posible identificar la utilidad de dichas variables para el modelo predictivo. Este proceso implica la examinación de las características y la ideación de hipótesis que puedan mejorar la predicción e interpretación del modelo.

- **Transformación:** Durante este proceso y dependiendo del tipo de dataset es posible decidir sobre los tipos de técnicas que se usan en esta etapa y los cuales están destinados a construir de forma manual o automática nuevas características, el garantizar la flexibilidad del modelo (asegurándose de presentar los datos al modelo de entrenamiento en el formato adecuado, ya sea escala o tipo de variable).

- **Extracción:** El propósito de este paso es reducir eventualmente el volumen de datos en un conjunto más amigable para el proceso de predicción del modelado. La información entonces es extraída de los datos en bruto y manipulados para crear las características que se hayan planeado para el problema; algunas incorporaciones de la extracción se pueden encontrar métodos como el análisis de texto o el análisis de componentes principales.

- **Selección:** Los algoritmos usados en esta etapa permiten evaluar y categorizar todas las características de la base de datos con el fin de determinar cuáles variables son más relevantes y deben ser conservadas por el modelo (dándoles prioridad a la hora de entrenamiento).

III. DESCRIPCIÓN DEL EXPERIMENTO/PROBLEMA

Gran parte de los trabajos anteriores sobre el fraude se centran no sólo en clasificadores sino también sobre técnicas de desbalanceo, manejo de dimensionalidad e ingeniería de características [15]-[17]. En el desarrollo de este proyecto, se pretende experimentar de forma detallada sobre diferentes algoritmos de clasificación haciendo comparaciones a partir de sus rendimientos individuales como modelo y con otras métricas usadas en este tipo de aplicaciones. Además de trabajar directamente sobre modelos de clasificación (supervisados o no supervisados), se analizará el uso de técnicas de preprocesamiento, escalamiento, desbalanceo y reducción de dimensionalidad en la predicción de etiquetas para el caso de fraude en transacciones electrónicas (comerciales). El modelo estará basado en datos, entrenado mediante el uso de información de transacciones electrónicas altamente codificadas cuya base de datos de entrenamiento posee originalmente una etiqueta de fraude, que será usada más adelante para hacer predicción. La clasificación se realizará de forma binaria, pero al mismo tiempo se entregará también una columna de probabilidades que le dan puntuación a una transacción sobre qué tan fraudulenta puede llegar a ser.

Este trabajo tiene por objetivo hacer uso de algoritmos de aprendizaje automático en la interpretación y análisis de datos de transacciones entregado por la plataforma Kaggle. El conjunto de datos original está divulgado de forma pública por lo que está a disposición de cualquier persona para su uso. El dataset proporcionado para este proyecto académico tiene como fuente una base de datos proveniente de la corporación Vesta, la cual es una compañía especializada en la evaluación de riesgos y prevención de fraude a través de modelos de aprendizaje. Dada la naturaleza de la información proveniente de las transacciones electrónicas, las cuales contienen información privada y delicada, desde Vesta se ha realizado un proceso de codificación para ocultar el símbolo y los valores de cada columna. De acuerdo con la exploración de la información se denotará que los datos fueron extraídos a lo largo de un año (realizando particiones para el entrenamiento y la prueba). Los datos constan de algunos valores de identificación interna previamente anonimizada (como dominios de correo electrónico, tipos de producto financiero, entre otros), así como, valores de transacciones, variables de tiempo y geográficas (codificadas).

De forma general, la siguiente lista describe algunas de las experimentaciones que se realizarán para este proyecto:

- Comparación de varios métodos de aprendizaje supervisado y no supervisado conforme se considere durante la experimentación.
- Selección de los mejores modelos en base a métricas de clasificación (y que sean acordes también con los requerimientos de la competencia de Kaggle). Algunas métricas que se pueden mencionar son: F1, curva ROC/AUC, entre otras.
- Uso de técnicas de preprocesamiento, desbalanceo y reducción de dimensionalidad con el fin de determinar cuál método mejora la predicción/ clasificación.

IV. METODOLOGÍA

A. Información inicial del dataset

La corporación Vesta proporciona cuatro datasets que representan una partición para entrenamiento y una para prueba, estos datasets a su vez han sido documentados con las transacciones electrónicas recopiladas por ellos y con la información de identificación del usuario por transacción. Con la concatenación de las tablas de transacción e identificación se obtienen dos datasets para entrenamiento y prueba. Estos dos datasets constan de 434 características incluyendo la columna de predicción de fraude. En total se obtiene 590540 registros únicos guardados en formato .csv y con un peso de los diferentes dataset de 1.6 Gb aproximadamente.

Debido a la dimensionalidad del dataset, se describe de forma general (y de acuerdo con lo detallado durante la competencia) las columnas agrupadas de acuerdo con su connotación:

- **Tabla de transacciones**

La TABLA *I.* una recopilación de información centrada en las transacciones registradas dentro de plataformas electrónicas, gran parte de las características que se encuentran en esta tabla están relacionadas al tipo de transacción que se realizó (como tipo de método de pago o la locación de donde provino la transacción). De esta tabla, existen 393 columnas generadas

por parte de la corporación Vesta como parte de su proceso de ingeniería de características; debido al tipo de información que se maneja, toda la tabla esta codificada.

- **Tabla de identidades**

Las variables de esta tabla (TABLA *II*.TABLA *II*) son información de identidad - información de conexión de red (IP, ISP, Proxy, etc.) y firma digital del usuario (tipo de navegador/versión del SO, etc.) asociada a cada transacción. Como se mencionó anteriormente, los nombres de las características y los valores están enmascarados y su significado no será proporcionado para la protección de la privacidad y de acuerdo con la confidencialidad que tiene la compañía Vesta.

TABLA I.
DESCRIPCIÓN PARA TABLA DE TRANSACCIONES

Columna	Tipo	Descripción
TransactionID	`int64`	Clave correspondiente a cada transacción, es la llave que se utilizará para unir las tablas de entrenamiento y prueba.
TransactionDT	`int64`	time delta a partir de una fecha-hora de referencia dada (no una marca de tiempo real)
TransactionAMT	`float64`	importe del pago de la transacción en USD
ProductCD	`Object`	código de producto, el producto de cada transacción
card1 - card6	`float64, int64, Object`	información de la tarjeta de pago, como tipo de tarjeta, categoría de tarjeta, banco emisor, país, etc.
Addr1 -addr2	`float64`	dirección
Dist1 – dist2	`float64`	distancia
P_ y (R__) emaildomain	`Object`	dominio de correo electrónico del comprador y del destinatario
C1-C14	`float64`	recuento, como cuántas direcciones se encuentran asociadas a la tarjeta de pago, etc. El significado real está enmascarado.
D1-D15	`float64`	time delta, como los días entre la transacción anterior, etc.
M1-M9	`Object`	coincidencia, como los nombres de la tarjeta y la dirección, etc.
Vxxx [V01 – V339]	`float64`	Características ricas en ingeniería de Vesta, incluyendo clasificación, recuento y otras relaciones de entidad.

TABLA II.
DESCRIPCIÓN TABLA DE IDENTIDADES

Columna	Tipo	Descripción
id_12 - id_38	`float64, Object`	Características de identidad, que son recogidas por Vesta como la calificación del dispositivo, ip_domain, Así como la huella digital del usuario.
DeviceType	`Object`	Huella digital del dispositivo de la transacción, en este caso tipo de dispositivo.
DeviceInfo	`Object`	Información extra del dispositivo

Durante el desarrollo de la investigación se explorarán diferentes métodos con los datos para evaluar la mejor ruta que nos permita obtener una predicción acertada sobre las etiquetas de fraude. Para acceder al dataset, se puede utilizar el API de Kaggle (la cual es una clave personal de cada usuario de Kaggle) para descargar cada dataset de la competencia. El procedimiento para obtener cada dataset se describe dentro de los notebooks entregados para la monografía y cuyo procedimiento se puede describir de la siguiente forma, como lo muestra la Figura 1.

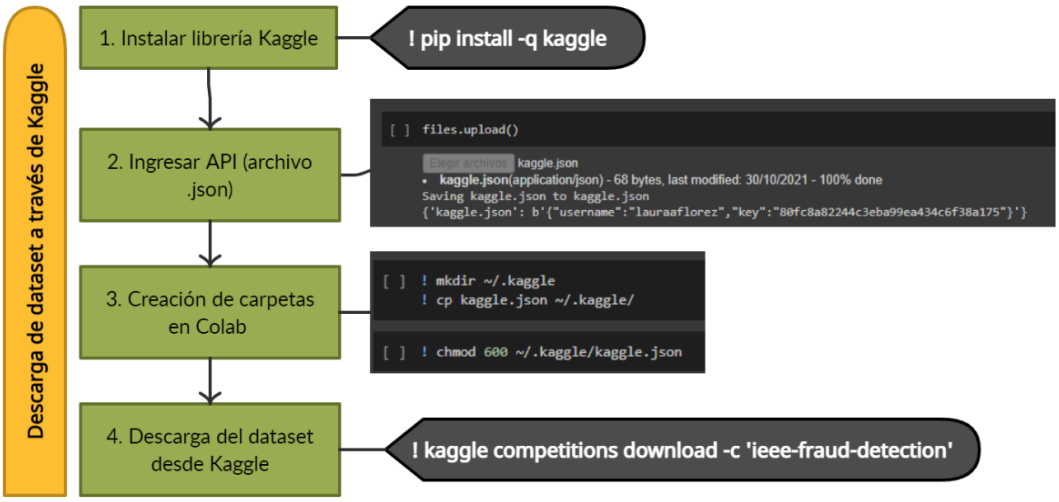


Figura 1. *Procedimiento para obtención del dataset a través de Kaggle*

Los demás datasets que van surgiendo del preprocesamiento, escalamiento y conversión serán descargados a partir de enlaces de Google Drive, asociados a una cuenta institucional (dichos enlaces están configurados para que sean públicos). En caso de no funcionar será necesario solicitar permiso al correo del autor.

B. Descripción general de la metodología

La Figura 2 ofrece una descripción general que muestra las fases principales dentro del enfoque predictivo que se llevará a cabo.

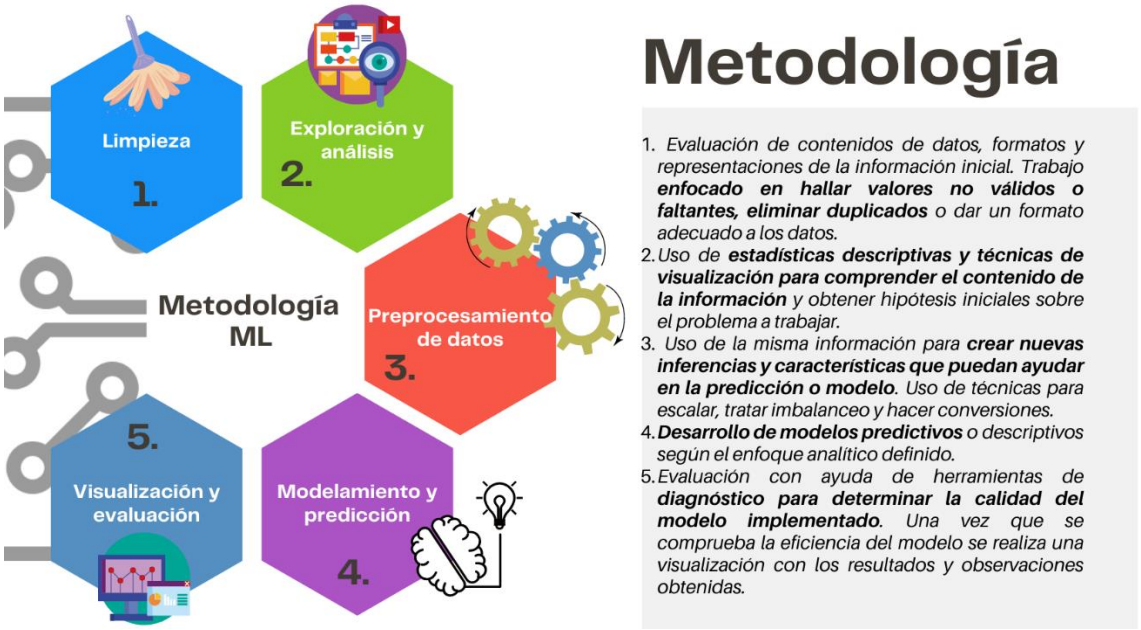


Figura 2. *Descripción general de la metodología del proyecto*

El objetivo de investigación, justo como se ha mencionado en la descripción del experimento es conocer es poder clasificar adecuadamente transacciones electrónicas bajo una categoría de fraude o no fraude. Para efectos de la investigación, se emplean diferentes métricas, técnicas de procesamiento y modelados con el fin de descubrir el mejor método para este caso de clasificación binaria. Como aditamento de la predicción, se evalúa la capacidad de aprendizaje de los modelos con la idea de detectar si los modelos están sobreentrenados o no (la cual es una ocurrencia muy común en datasets de tan grande dimensionalidad). Con el fin de cumplir lo pactado durante la descripción del experimento se plantearon diferentes etapas que se detallan en las secciones a continuación.

C. Limpieza, exploración y análisis preliminar

Antes de iniciar con el planteamiento de un modelo es necesario evaluar la calidad de la información que se va a usar para un entrenamiento o para una predicción. Por calidad, nos referimos a información sin valores nulos o con las condiciones necesarias para que sus atributos puedan ser usados por un modelo de aprendizaje automático. Dentro del desarrollo

de esta etapa, se toma como tarea principal la preparación, corrección, rellenado y suavizado de los datos. Por otro lado, se lleva a cabo una exploración de la información con la idea de comprender las fuentes de datos que se poseen, así como de generar primeras inherencias o hipótesis que puedan describir superficialmente relaciones entre características del dataset. Con base en lo anterior, el siguiente diagrama (ver Figura 3) muestra la ruta de limpieza y exploración que se efectuará sobre el dataset de entrenamiento, además de una descripción de cada aspecto.

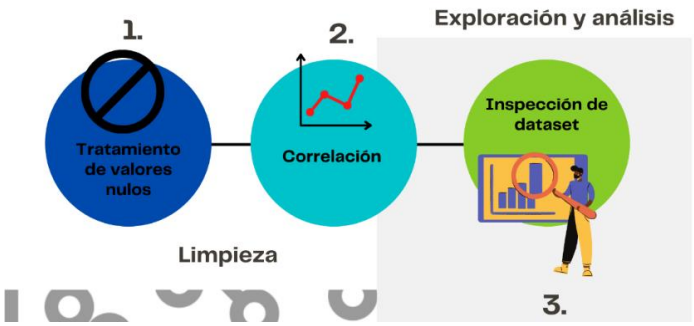


Figura 3. *Paso a paso en el proceso de limpieza y exploración de datos*

- **Tratamiento de valores nulos**

Debido a que no se conoce con exactitud el origen de los valores faltantes, es decir, si existe alguna relación sobre la carencia de su definición como valor o si estos faltantes fueron generados de forma aleatoria, se escoge como estrategia identificar las columnas con mayor cantidad de valores nulos (para su eliminación) con la intención de reducir las dimensiones del dataset. Sin embargo, también se debe mencionar que eliminar características en primera instancia pueden afectar el comportamiento de los modelos, por esta razón, después de seleccionar las columnas con mayores valores nulos estas serán evaluadas en una matriz de correlación con respecto a la etiqueta de fraude para observar si eliminarlas puede afectar verdaderamente la predicción con los modelos de clasificación.

- **Correlación**

El objetivo de calcular la correlación entre las características del dataset recae especialmente en encontrar las relaciones más significativas estadísticamente hablando, obteniendo dos tipos de relación a denotar, por un lado las variables que se asemejan mucho

entre ellas (con una alta correlación) que implica que poseen un significado similar, y las variables altamente correlacionadas con la etiqueta de fraude. En este último caso, dichas características podrían perfectamente ayudar en la predicción de una transacción fraudulenta, por lo que no sería recomendable eliminarlas. Como recurso para estimar la relación entre todas las variables del dataset, se trabajará sobre mapas de color que nos indiquen con una escala su correlación positiva o negativa.

- **Inspección de dataset**

Con la observación de las características del dataset es posible obtener un conjunto de suposiciones que nos permitan describir el comportamiento del contenido, ya sea a través de evento atípicos que puedan ser relevantes o patrones que se deseen mantener a la hora de predecir una clase. Una vez que la exploración haya sido suficiente y se hayan obtenido hipótesis, sus características pueden ser luego traducidas en nuevos atributos o agrupaciones que al utilizarse sean factor para mejorar el análisis o modelado de datos. Para el desarrollo neto de esta etapa se usarán las características de las librerías Pandas, Numpy, Matplotlib y Seaborn.

Como resultado de esta etapa se obtendrá una considerable comprensión de los datos y de la calidad general de la misma, donde el dataset habrá sido convertido conforme a las necesidades del entorno de programación y cuyos valores serán representados mediante gráficas y análisis que describan su comportamiento inicial, lo que da inicio a una etapa posterior de preprocesamiento.

D. Procesamiento y transformación de los datos

Una vez se ha procedido con la debida limpieza y exploración del dataset, al punto de obtener ideas iniciales sobre cómo funcionan o cómo se relacionan las características, es posible, empezar a trabajar dentro del procesamiento de la información (que incurre también en la preparación y transformación de los datos) para que puedan ser usados fácilmente por la máquina de aprendizaje. Esta etapa es indiscutiblemente necesaria antes de proceder con un modelo, ya que con este proceso se aumenta la precisión y eficacia del aprendizaje (y su predicción) al usar formatos adecuados y adaptados a su aplicación. Más allá de la limpieza

de datos, el preprocesamiento hace uso de técnicas que se encargan de transformar y/o codificar la información de manera que los algoritmos sean capaces de interpretarlos; dichos métodos están destinados ya sea a la imputación de valores, a la creación de variables como a conversiones numéricas. El flujo de trabajo en esta sección se refleja en la Figura 4, donde más adelante cada método será detallado para profundizar su uso en esta investigación.

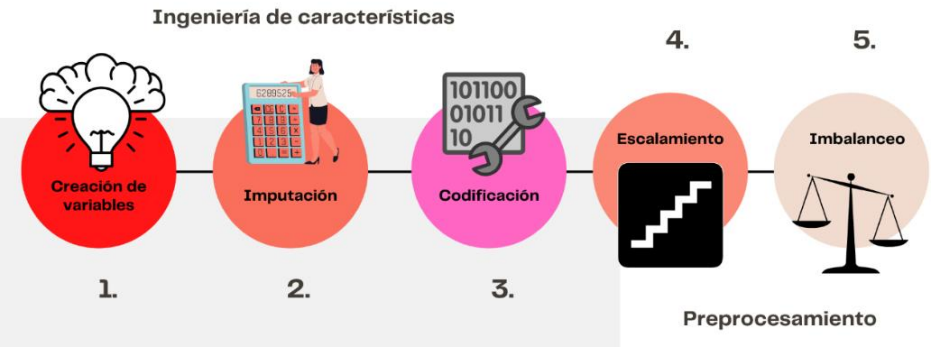


Figura 4. *Paso a paso en proceso de preprocesamiento*

- **Imputación:**

Para evitar que los modelos que puedan ser usados en este proyecto descarten las filas con valores ausentes, se utilizan métodos de imputación para introducir valores que sean coherentes a los de las columnas que presentan vacíos. Antes de aplicar estas técnicas, se hará una reducción de memoria del dataset para no consumir por completo la memoria RAM del ambiente de Google Colab. Esta reducción consiste en convertir columnas numéricas a una versión con menos bits, por ejemplo, pasando un float64 (tipo original numérico del dataset) a float32 o float16, lo que eventualmente reduce el consumo de memoria al trabajar sobre los datos.

Como estrategia para hacer la imputación de datos se utiliza el siguiente formato (ver TABLA *III*):

TABLA III
ESTRATEGIA DE IMPUTACIÓN

Clasificación numérica	Tipos de columnas en dataset	Método de imputación
Reales	`float64` →`float32, float16`	Mediana
Enteros	`int64, int8`	Media
Categóricas	`Object`	Moda

- **Ingeniería de características**

En esta fase y conociendo de antemano cuales variables se puedan trabajar, se irán creando nuevas propiedades de los datos a partir de los atributos ya existentes. El establecimiento de nuevas características puede surgir, por ejemplo, de una conversión de tiempo, de la interacción o validación entre varias columnas para crear un flag o indicador de algún evento. Todas las características que serán creadas funcionaran alrededor de la etiqueta de fraude y su origen proviene de la exploración realizada con anterioridad de las características del dataset.

- **Codificación**

Una vez se hallan creado las nuevas características es necesario transformar las columnas categóricas a valores numéricos para que puedan ser aceptadas por los algoritmos de aprendizaje. Debido a la connotación matemática con la que trabajan algunos modelos no es posible trabajar directamente con datos categóricos, por esta razón, es obligatorio convertir este tipo de variables a forma numérica para poder trabajar con ellos. Existen diferentes métodos para convertir valores categóricos que dependen de su representación, ya sea en forma ordinal o cardinal, entre ellas es posible mencionar: One-hot encoding, Dummy encoding, Target encoding, Label encoding, entre otros.

Para trabajar sobre estos valores categóricos se analiza el tipo de representación que tienen, es decir si tienen demasiada cardinalidad (muchos valores únicos) o si son ordinales (bajos valores únicos). Sobre las columnas con muchos valores únicos se usará Frequency encoding, que utiliza la frecuencia dentro de la misma característica para codificar las categorías, mientras que, para aquellas columnas con bajos valores únicos se usará Label

encoding, que va asignando valores desde 0 y en forma ordenada hasta el número de valores únicos que la columna presente.

- **Escalamiento**

Como medio para evitar un dominio de unas variables numéricas sobre otras, se utilizan técnicas de transformación que permitan escalar todas las características en un mismo rango numérico. Dichos métodos se usan conforme al tipo de aplicaciones o valores numéricos que estén presentes dentro del dataset, por ejemplo, ajustar los valores en un rango desde 0 hasta 1. Para evaluar la influencia del escalamiento en la predicción, se hará uso de las técnicas StandardScaler, Normalizer y MinMaxScaler de la librería de preprocesamiento de Scikit-Learn.

- **Desbalanceo**

Por último se trabajará sobre el desequilibrio de clases presente en el dataset, donde la clase no fraudulenta resulta ser la más predominante (esta situación también se observa dentro de la exploración del dataset). Una de las formas para lidiar con esta situación implica usar técnicas de desbalanceo que se encargan de generar datos sintéticos basados en la clase minoritaria. En particular se utilizará el método SMOTE (*Synthetic Minority Over-sampling Technique*) que toma un conjunto de datos de la clase minoritaria y luego se crean nuevas características sintéticas iguales, que luego son añadidas al dataset original. Durante la experimentación se escogerán dos tasas de generación de datos sintéticos: 0.05 y 0.1, los cuales serán evaluados en términos de métricas y predicción para escoger el de mejor rendimiento.

E. Modelamiento y evaluación

Como paso final del proyecto, el dataset preprocesado será sometido a diferentes modelos de aprendizaje para predecir la probabilidad y la clase de transacciones fraudulentas o no fraudulentas. De forma intencional, se explorará modelos de aprendizaje supervisados, semi-supervisados y no supervisados para evaluar cuál de todos puede dar una mejor predicción. Para encontrar inicialmente el rumbo de modelos de entrenamiento, se utiliza la herramienta AutoML de mano de la librería H2O, la cual entrega una lista de clasificación

de modelos conforme ciertas métricas indicadas y permite elegir los posibles modelos para nuestro problema de clasificación. Como métricas de evaluación se escogen distintas métricas provenientes de la librería de Scikit-Learn como Accuracy, F1 Score, Matriz de confusión y ROC -AUC (la cual es la métrica indicada para entregar dentro de la competencia de Kaggle). El procedimiento que se llevará a cabo durante esta etapa se ve reflejado en la Figura 5.

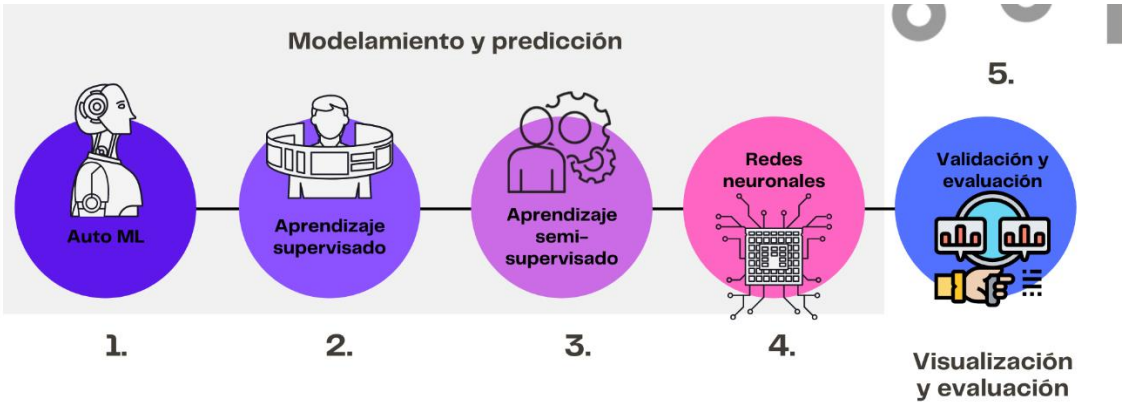


Figura 5. *Paso a paso en proceso de modelamiento y validación*

Las etapas de trabajo en el proceso de modelamiento se describen a continuación:

- **AutoML**

El uso de AutoML consiste en una aplicación de modelos genéricos de diferente índole: supervisados o no sobre un dataset debidamente estructurado y preprocesado. Este tipo de algoritmos entrega una tabla de clasificación sobre el conjunto de parámetros que haya sido definido, por ejemplo, la duración de la aplicación automática, la naturaleza del problema (regresión o clasificación), las métricas y el número de modelos a entregar. En este caso, los anteriores parámetros serán usados para el aprendizaje automático, tomando como tiempo de entrenamiento alrededor de 5 horas y 15 como el número de modelos a evaluar. Cabe mencionar que sólo se tomará en cuenta para el proceso de modelamiento las opciones de aprendizaje que sean diferentes a los modelos ensamblados, esto con el fin de poder explorar sobre las demás opciones de aprendizaje.

- **Aprendizaje supervisado**

Debido a que se posee de antemano una etiqueta de fraude conocida, está será usada para emplear modelos de aprendizaje supervisado, los cuales serán elegidos también dependiendo del resultado de exploración con AutoML. Debido a la dimensionalidad y el volumen del dataset se espera que la ruta de entrenamiento y predicción se decante por modelos basados en árboles ya que son los más usados en este tipo de aplicaciones predictivas. Algunas técnicas que se podrían usar son: GradientBoosting, XGBoost o LightGBM. Los diferentes modelos resultantes de esta etapa serán ajustados (conforme a sus parámetros) y evaluados con las métricas de clasificación definidas anteriormente para observar si no ocurre overfitting y su rendimiento es acorde a lo solicitado por la competencia de Kaggle.

- **Aprendizaje semi-supervisado**

Ya que el dataset a pesar de su preprocesamiento pueda seguir teniendo un volumen considerable, una forma de trabajar dichos problemas de clasificación es haciendo una reducción de dimensionalidad a través de una red neuronal que transforme los datos conforme a un número de capas/características definidas. Al hacer esta conversión es posible utilizar métodos supervisados para predecir de nuevo las diferentes clases de fraude del proyecto. Para complementar este enfoque semi-supervisado se usarán los modelos empleados en la sección anterior y como fase para reducir las dimensiones del dataset, se usarán autoencoders planteados desde la librería Keras.

- **Redes neuronales**

Debido a la naturaleza de las redes neuronales, se espera que con su planteamiento sean capaces de detectar los patrones dentro del dataset para que luego sean capaces de identificar la condición fraudulenta de una transacción. Para evaluar el comportamiento de la red se usarán las mismas métricas empleadas para los métodos supervisados.

- **Validación y evaluación**

Como se estuvo mencionando en las secciones anteriores cada modelo será evaluado por las mismas métricas, obteniendo no solo valores numéricos sino también gráficas que puedan mostrar una mejor visualización sobre el rendimiento de todos los modelos. Aparte

de las métricas, se hará uso de validaciones sobre el aprendizaje de los algoritmos, haciendo curvas de aprendizaje y validaciones cruzadas para observar la tendencia de sus predicciones.

V. RESULTADOS

A. Descarga, limpieza y exploración del dataset

Como se mencionó en el planteamiento de la metodología, el dataset tiene origen en una competencia de Kaggle, desarrollada en 2018 y la cual todavía está disponible para su experimentación. Ya que el entorno de trabajo principal será sobre Google Colab, se utiliza una llave API individual para descargar directamente desde la competencia (observe la Figura 1 para más detalle del proceso). Las primeras observaciones que se realizan sobre el dataset están concentradas en evaluar aspectos de calidad de la información, como su tamaño, los valores nulos, la duplicidad en información, entre otros. Para esta sección se utilizaron especialmente las librerías de Numpy y Pandas, pues el dataset ha sido leído y manipulado en este formato.

Después de una concatenación de la tabla de transacciones e identidad para el entrenamiento, se observó la distribución de las columnas, justo como se ve en la Figura 6.

```
[ ]  test = pd.merge(test_trans, test_ident, on=['TransactionID'],how='left')
     train = pd.merge(train_trans, train_ident, on='TransactionID', how='left')
```

Como se observa, el formato de los datos en las columnas son números de 64 bits, los cuales consumen demasiada memoria, y a su vez, generan datasets con gran tamaño (ambos por encima de 1.5 Gb).

```
[ ]  train.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 590540 entries, 0 to 590539
     Columns: 434 entries, TransactionID to DeviceInfo
     dtypes: float64(399), int64(4), object(31)
     memory usage: 1.9+ GB
```

```
[ ]  test.info()

     <class 'pandas.core.frame.DataFrame'>
     Int64Index: 506691 entries, 0 to 506690
     Columns: 433 entries, TransactionID to DeviceInfo
     dtypes: float64(399), int64(3), object(31)
     memory usage: 1.6+ GB
```

Figura 6. *Información sobre el dataset de entrenamiento y prueba*

Con base en el tamaño de los datasets, después de hacer iteraciones iniciales sobre la información, se comprobó que el entorno en la nube de Google Colab (que normalmente trabaja con 12 Gb de memoria RAM) no es suficiente ni siquiera para realizar un proceso de exploración, razón por la cual, se hace una conversión de formatos numéricos, buscando disminuir el consumo de memoria. Parte del código usado para dicha conversión y cuyo procedimiento se relata también en la metodología se refleja en la Figura 7.

```
if str(col_type)[:3] == 'int':
    if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
        df[col] = df[col].astype(np.int8)
    elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
        df[col] = df[col].astype(np.int16)
    elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
        df[col] = df[col].astype(np.int32)
    elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
        df[col] = df[col].astype(np.int64)
```

```
[ ]  train=reduce_mem_usage(train)

     Uso de memoria disminuyo a 650.48 Mb (66.8% de reducción)

[ ]  test=reduce_mem_usage(test)

     Uso de memoria disminuyo a 565.37 Mb (66.3% de reducción)
```

Figura 7. *Reducción de memoria de datasets*

Una vez realizada esta conversión es posible trabajar de manera más flexible con el dataset entero. Por otro lado, ya que al tener un volumen tan grande de información, es

posible que el tiempo de ejecución de varias líneas de código pueda llegar a ser muy alto, así que se decidió trabajar sobre 1/8 del dataset (es decir, 73817 filas) con el fin de realizar un proceso de entrenamiento, de validación e incluso de ajuste de hiperparámetros de forma más fácil y rápida. Lo descrito anteriormente implica dos cosas:

i. el entrenamiento de los modelos y evaluación de la ruta a trabajar se hizo con el uso de pocos datos, con la idea de tener más disposición de recursos y tiempo para explorar
ii. una vez se determinaron las mejores condiciones de trabajo, se regresó a hacer predicción sobre el conjunto total de datos (y hacer una evaluación sobre su comportamiento en la competencia de Kaggle).

Para poder observar la distribución y la cantidad total de la información, se utilizó el dataset original (aquel con más de 590 mil muestras). Durante el proceso de entrenamiento dicha dimensión se disminuyó, como se mencionó anteriormente; todo el procesamiento y limpieza se aplica de igual forma para la subsección del dataset con la que se hizo entrenamiento.

- **Limpieza**

Haciendo una mirada superficial del dataset de entrenamiento, se denota que existen 434 características destinadas a predecir fraude y que al parecer varias de ellas presentan valores nulos, los cuales más tarde pueden afectar el entrenamiento de los modelos (en caso de que dichos modelos no estén constituidos para lidiar con dicha información carente). Otro paso que se realizó fue la comparación entre el dataset de entrenamiento y prueba con el fin de verificar que tuvieran las mismas columnas. Las Figura 8 (a y b), muestran una vista inicial del dataset con la visualización de los valores que serán manejados más adelante y la comparación realizada para los datasets.

a)

card2	card3	card4	card5	card6	addr1	addr2	dist1	dist2	P_emaildomain	R_emaildomain
NaN	150.0	discover	142.0	credit	315.0	87.0	19.0	NaN	NaN	NaN
404.0	150.0	mastercard	102.0	credit	325.0	87.0	NaN	NaN	gmail.com	NaN
490.0	150.0	visa	166.0	debit	330.0	87.0	287.0	NaN	outlook.com	NaN

b)

```
print(f'Tamaño de dataset de entrenamiento: {train.shape}')
print(f'Tamaño de dataset de prueba: {test.shape}')
common_cols=set(train.columns).intersection(set(test.columns))
print(f'Número de variables en común, entre ambos datasets: {len(common_cols)}')
print(f'Caracteristicas faltantes: {test.shape[1]-len(common_cols)}')
```

```
Tamaño de dataset de entrenamiento: (590540, 434)
Tamaño de dataset de prueba: (506691, 433)
Número de variables en común, entre ambos datasets: 395
Caracteristicas faltantes: 38
```

Figura 8. *a) Visión de valores nulos y b) comparación entre datasets de entrenamiento y prueba*

De acuerdo con lo anterior, los datasets de entrenamiento y prueba presentaron una diferencia en columnas, la cual fue dado por un error de tipología: las columnas de prueba de la tabla de identidad poseían distinto carácter en su nombre, generando tal discrepancia. Con la línea de código: `test.columns = test.columns.str.replace("-", "_")`, fue posible corregir dicha situación.

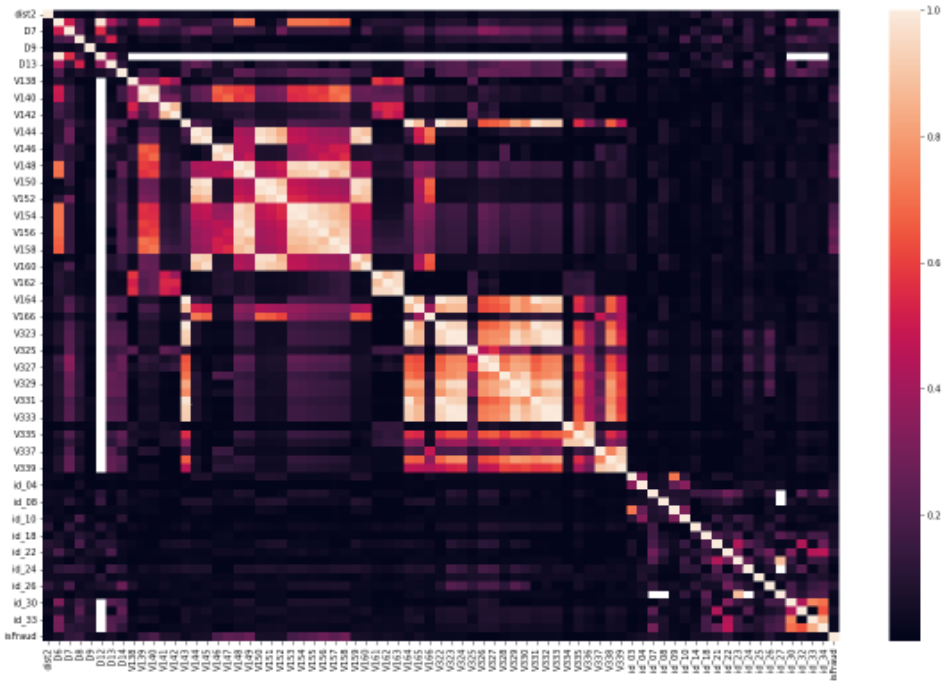
Una vez se hizo la corrección del dataset de prueba, se identificó los valores nulos del dataset. Dentro de dicha información obtenida se destacó la gran cantidad de columnas con nulos, donde incluso varias características presentaban valores nulos en más del 80% de las filas registradas en el dataset. Como primer paso de preprocesamiento se hizo un conteo de las columnas que presentaban más del 85% de valores nulos y se analizaron en una matriz de correlación con respecto a la etiqueta de fraude para determinar la influencia sobre la predicción de fraude. En este caso, si existía alguna columna con muchos valores nulos y que a su vez presentara una alta correlación con la etiqueta de fraude podría ser un indicador de relación entre dicha característica y el fraude, por lo que no debería ser eliminada, mientras que sí, las columnas con mayor número de nulos no tenían alta correlación serían eliminados con la idea de reducir la dimensión del dataset y poder trabajar de forma más sencilla. Como resultado de este paso, se obtuvo una lista de 51 características con baja relación respecto a la etiqueta objetivo, llevando a su posterior eliminación del dataset, además de determinar que no existen duplicados dentro del dataset. Parte del código usado y la matriz de correlación se observa a continuación en las Figura 9 (a y b).

a)

```
[ ]  columns_to_drop.columns

Index(['dist2', 'D6', 'D7', 'D8', 'D9', 'D12', 'D13', 'D14', 'V138', 'V139',
       'V140', 'V141', 'V142', 'V143', 'V144', 'V145', 'V146', 'V147', 'V148',
       'V149', 'V150', 'V151', 'V152', 'V153', 'V154', 'V155', 'V156', 'V157',
       'V158', 'V159', 'V160', 'V161', 'V162', 'V163', 'V164', 'V165', 'V166',
       'V322', 'V323', 'V324', 'V325', 'V326', 'V327', 'V328', 'V329', 'V330',
       'V331', 'V332', 'V333', 'V334', 'V335', 'V336', 'V337', 'V338', 'V339',
       'id_03', 'id_04', 'id_07', 'id_08', 'id_09', 'id_10', 'id_14', 'id_18',
       'id_21', 'id_22', 'id_23', 'id_24', 'id_25', 'id_26', 'id_27', 'id_30',
       'id_32', 'id_33', 'id_34'],
      dtype='object')
```

b)



c)

```
[36]  train.duplicated().sum()

0
```

Figura 9. *a) Selección de columnas con muchos valores nulos, b) matriz de correlación y c) valores duplicados*

- **Exploración e ingeniería de características**

Después del proceso de limpieza se empezó a trabajar sobre un dataset con 353 características tanto numéricas como categóricas. En una primera evaluación de la información, se deseó conocer la ordenación de todas las columnas obteniendo una gráfica por cada variable (como se ve en la Figura 10), denotándose que las distribuciones de algunas columnas numéricas no poseían relación entre sí mismas, además de presentar distribuciones no normales. En cuanto a las variables categóricas (que se plasman también en la Figura 10) se observa que el conteo de casos se concentra más sobre tarjetas MasterCard además de Visa

y un producto determinado como W (cuyo significado real está codificado) y hace relación al tipo de transacción electrónica realizada.

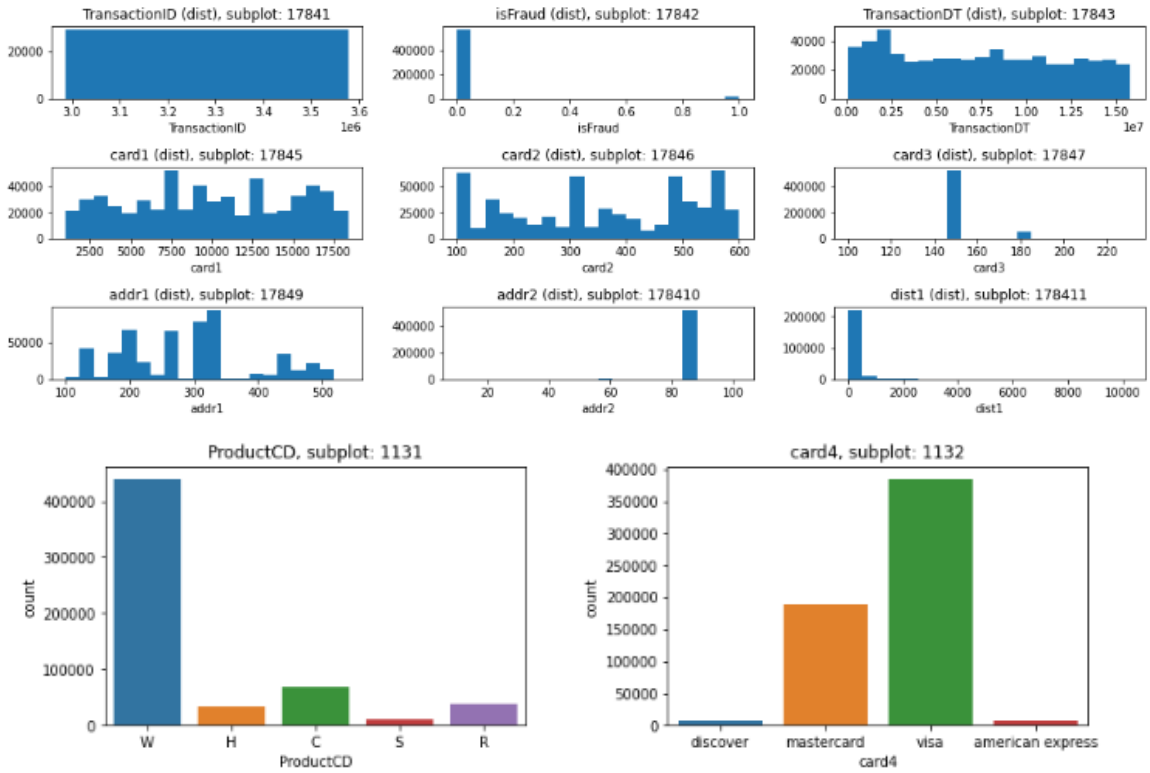
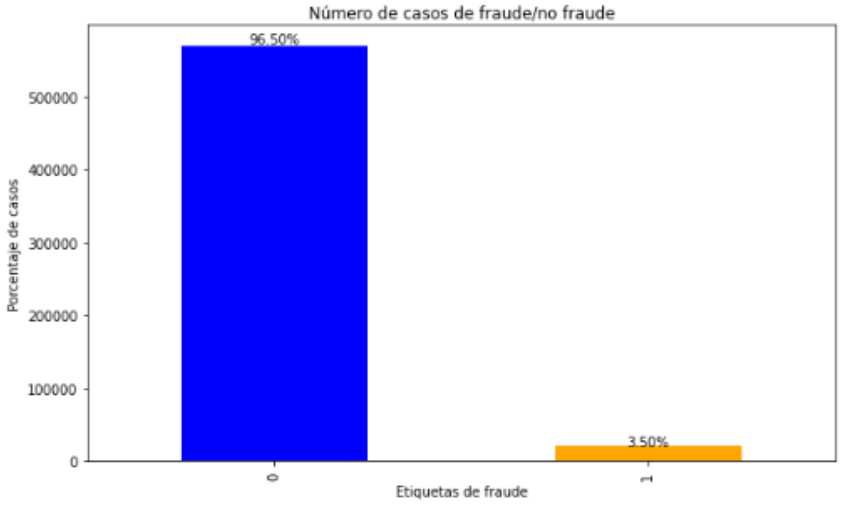


Figura 10. *Distribución de algunas variables numéricas y categóricas*

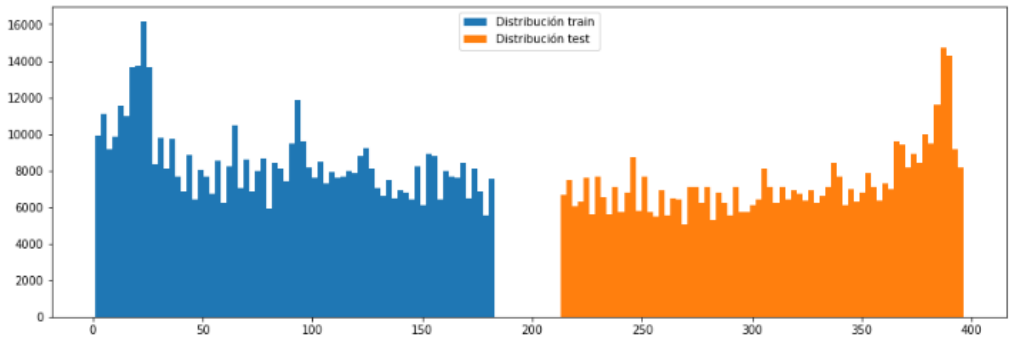
También se hizo una relación del número de casos presentes en el dataset con respecto a las etiquetas objetivo por predecir, es decir, si son de fraude o no. La Figura 11 nos muestra una situación común en eventos financieros, donde por lo general, se presenta cierto desbalanceo de datos. Lo anterior nos indica que la clase de no fraude presenta un conteo mayor de casos (96.5%) con respecto a su etiqueta contraparte (fraude con 3.5%); con el fin de solucionar dicho desequilibrio se implementaron algunas técnicas para observar si se influye mejor sobre la predicción en este tipo de eventos.

Así mismo se hizo un análisis sobre la columna `TransactionDT` (ver Figura 11 (a y b)), que de acuerdo con la definición dada durante la competencia hace referencia a un delta de tiempo, la cual se identificó se encuentra en segundos. Con ayuda de la línea `trainset['TransactionDT']/86400` (que convierte el valor de la columna en días) se comprueba que la distribución de los datasets ha sido a lo largo de un año.

a)



b)




```
count    590540.000000
mean         84.729199
std          53.437277
min           1.000000
25%          35.000000
50%          84.000000
75%         130.000000
max         182.000000
Name: TransactionDT, dtype: float64
```

Entrenamiento (en días)

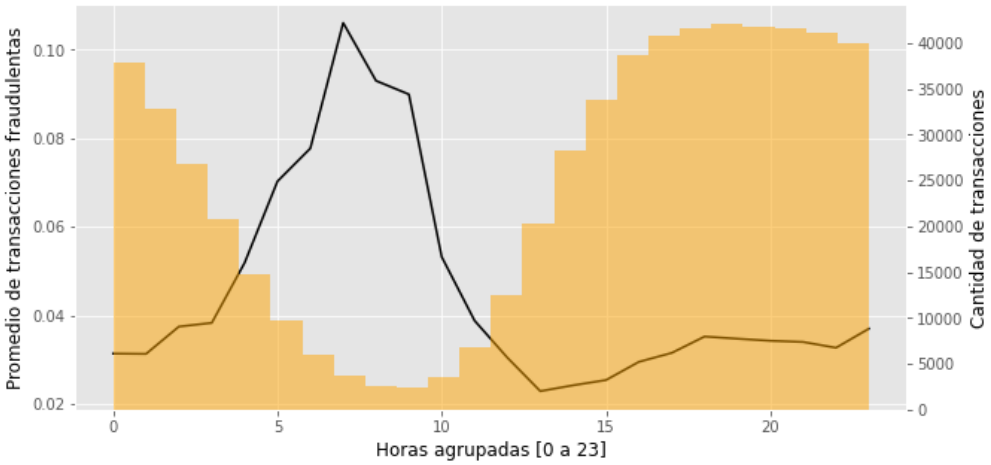
```
count    506691.000000
mean        311.086427
std          55.058802
min         213.000000
25%         263.000000
50%         314.000000
75%         362.000000
max         395.000000
Name: TransactionDT, dtype: float64
```

Prueba (en días)

Figura 11. *a) Conteo de casos de fraude y no fraude y b) Distribución en el tiempo del dataset*

Dada esta estructura de tiempo se fueron creando algunas variables relacionadas a la columna `TransactionDT`. Por un lado, se toma como fecha de partida el año 2017 (ya que la competencia fue llevada a cabo en principios de 2018), y partir de este punto se crearon columnas con respecto a la distribución en horas y meses de las transacciones. Con las Figura 12 (a y b), se observa la distribución en el tiempo (tanto en horas como en meses) del promedio y la cantidad total de transacciones fraudulentas.

a)



b)

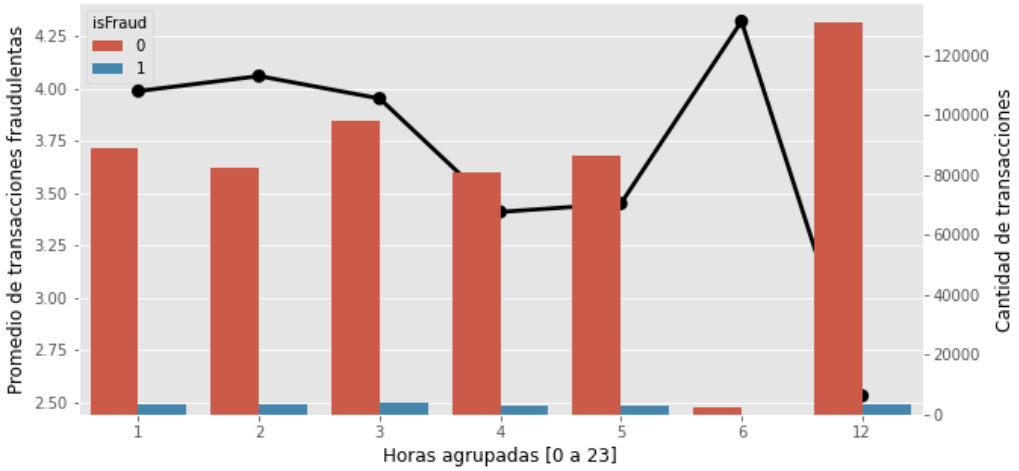
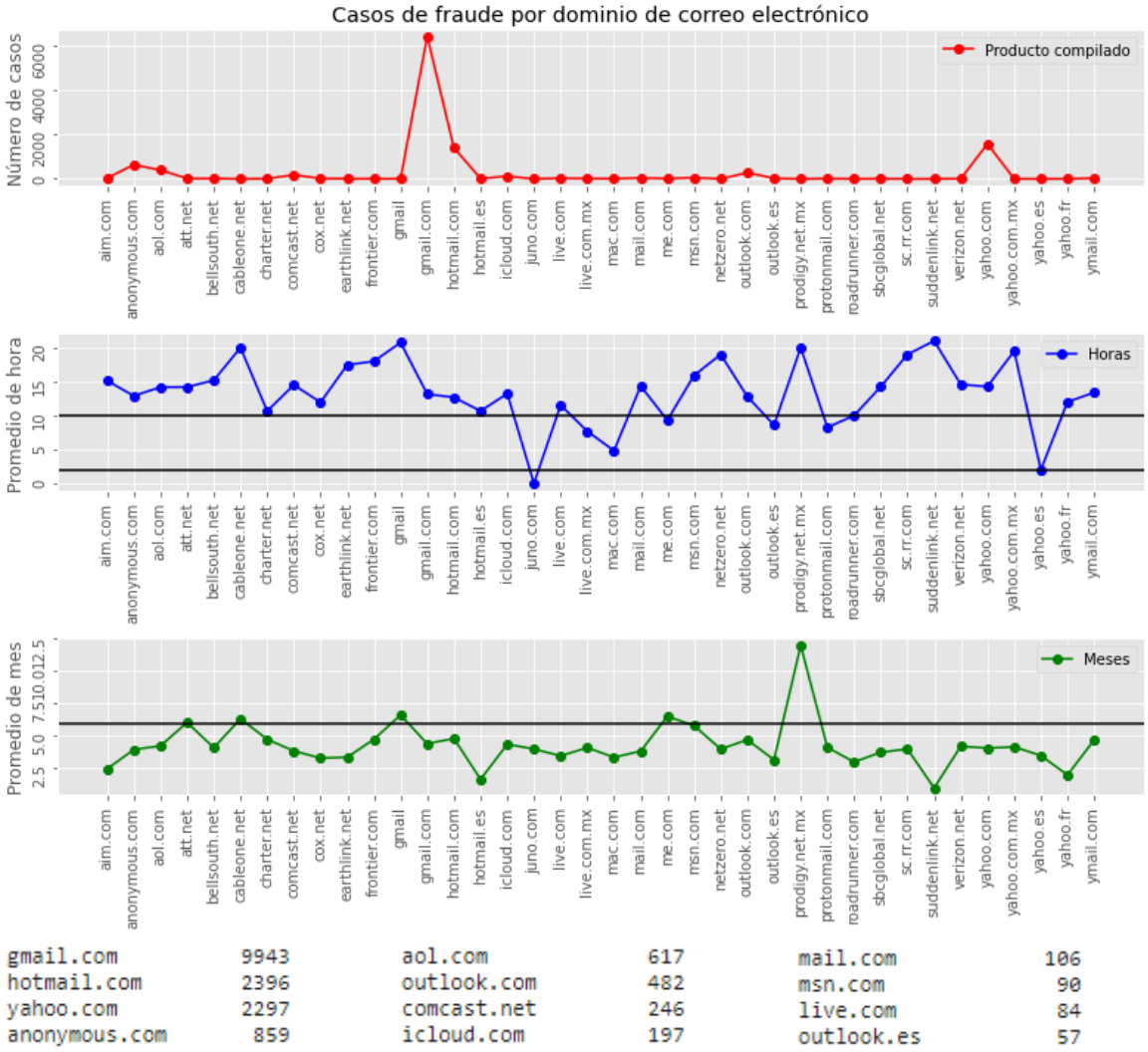


Figura 12. *a) Relación de fraude con respecto a las horas de un día y b) Relación de fraude con respecto a la distribución en meses del dataset de entrenamiento*

De este análisis se identificó que las horas entre las que más se presenta fraude son entre las cinco y las diez de la mañana además del sexto mes (con mayor promedio de fraude) sobre el dataset de entrenamiento. Ya que las características de tiempo realizadas han proporcionado tal información, se dejaron en el dataset para que fueran usadas en el proceso de implementación de modelos.

Otra columna categórica que se evaluó con respecto a las etiquetas de fraude fue el dominio de correo electrónico, los cuales se evaluaron con respecto al número de casos además del promedio de fraude por horas y por meses. En este caso, se destacó que el dominio de **gmail.com**, es el que presenta mayores casos de fraude, lo que incluso puede llegar a ser lógico, pues este dominio de Google, junto con Yahoo y Microsoft son unos de los menos seguros a nivel global [21]. La distribución en el tiempo (en horas y meses) del fraude con

respecto a los dominios de correo electrónico destaca también que este mismo dominio, **gmail.com,** presenta relación tanto con las horas y el mes en el que se detectó se presentaba mayor fraude. Dichas relaciones se pueden observar en la Figura 13.



gmail.com	9943	aol.com	617	mail.com	106
hotmail.com	2396	outlook.com	482	msn.com	90
yahoo.com	2297	comcast.net	246	live.com	84
anonymous.com	859	icloud.com	197	outlook.es	57

Figura 13. *Relación de los dominios de correo electrónico con respecto a las horas y los meses*

Otra de las hipótesis que se manejó en el proceso de ingeniería de características se centró en el tipo de divisa que se manejó dentro de las transacciones, donde se esperaba que parte de las transacciones fraudulentas del dataset estuvieran relacionadas con cantidades de dinero con más de una cifra significativa que podría implicar la conversión por moneda extranjera. Con base en la idea antes mencionada, se creó una variable que describe el número de cifras significativas por cada transacción, la cual fue posteriormente evaluada en una gráfica con respecto al fraude en el dataset (ver Figura 14).

```
[ ]  #extraer los centavos
     trainset['CurrencyCents']=trainset['TransactionAmt']%1
     testset['CurrencyCents']=testset['TransactionAmt']%1
```

```
[ ]  #contar para cada transacción cuantas cifras significativas posee
     trainset['CentsDigits']= trainset['CurrencyCents'].map(lambda a: len(str(round(a,3)).split(".")[1]))
     testset['CentsDigits']= testset['CurrencyCents'].map(lambda a: len(str(round(a,3)).split(".")[1]))
```

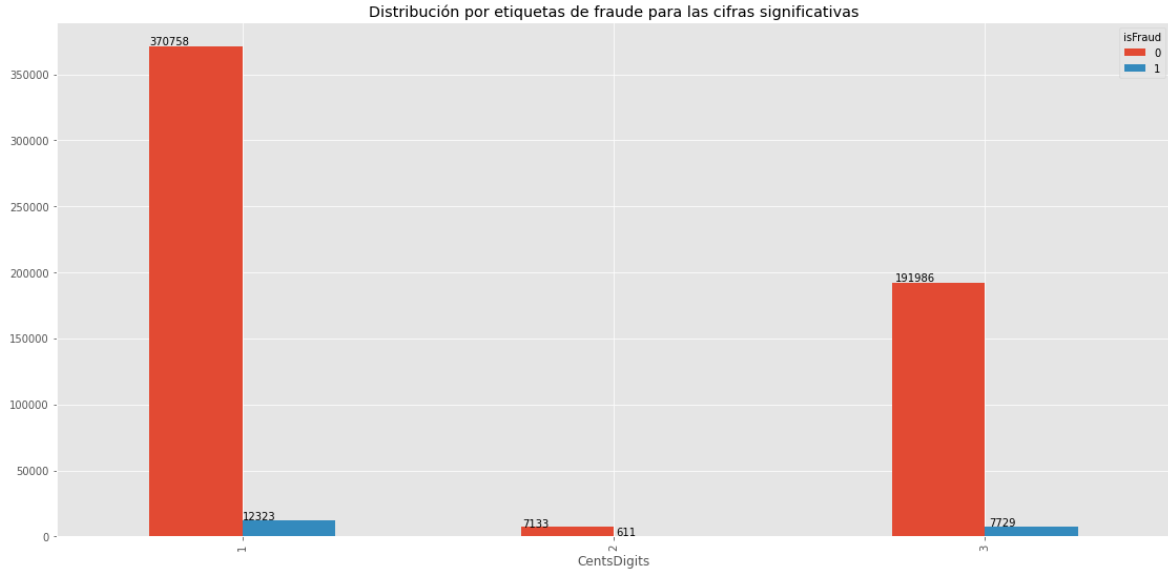


Figura 14. *Distribución de los casos de fraude con respecto al número de cifras significativas*

De la anterior gráfica se puede observar como la distribución de la etiqueta de fraude se concentra en transacciones con sólo una cifra significativa (especialmente porque son transacciones dentro de los Estados Unidos) y en cantidades de tres cifras significativas, lo que apoyaría la idea de la relación entre estas cifras después de la unidad decimal (como si fuera causado por la conversión a otro tipo de moneda) y el fraude. Por otro lado, se trabajó también sobre el tipo de dispositivo desde el que se realizó la transacción, basándonos en la premisa, de que un dispositivo no autentificado o un dispositivo cuya información de agente de usuario (*User Agent*) no sea coherente, podría indicar una relación con la presencia de fraude durante una transacción.

La nueva característica creada, consistió en la creación de un indicador numérico (en un rango de -1 a 1) que revelara la coincidencia entre la información serial, dispositivo y marca que el dataset proveía sobre el tipo de dispositivo que se usó para la transacción con respecto a la información extraída de internet sobre agentes de usuario. La escala calificó el dispositivo de uso de la siguiente manera: se asigna -1 cuando la información del dataset no es suficiente para generar una comparación, 0 cuando la información concuerda (es decir, el

dispositivo concuerda en marca y modelo) y 1 cuando no hay coincidencia entre la información del dispositivo. Con el fin de observar el comportamiento de esta característica se crea un gráfico de barras para observar el comportamiento de la asignación. En este caso, se toma la sección del dataset que corresponde únicamente a los casos de fraude, logrando observar dicha característica. La Figura 15. muestra el resultado obtenido, junto con parte del código usado para dicha fabricación de características.

```
[108] def compare_device(dataset,columna1,columna2):
        comparacion=[]
        for ind in dataset.index:
          if columna1[ind]!= 0 and columna2[ind]!=0:
            if columna1[ind]==columna2[ind]:
              comparacion.append(1)
            else:
              comparacion.append(-1)
          else:
            comparacion.append(0)
        return comparacion
```

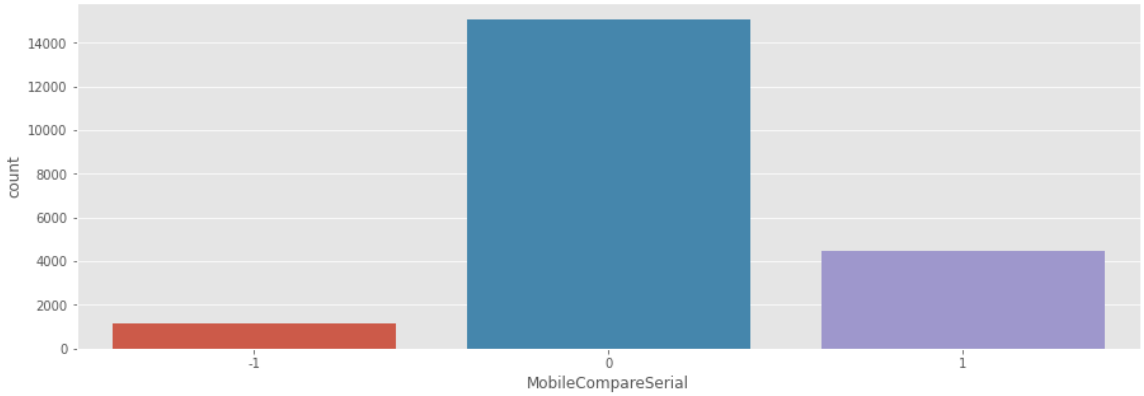


Figura 15. *Agrupación de casos de fraude de acuerdo al indicador de dispositivo*

De la anterior gráfica es posible observar varias cosas, por un lado, la asignación que presento mayor conteo de casos fue 0, el cual representaba una correcta identificación del dispositivo de uso. Mientras que para 1 y -1 se muestra un menor conteo de casos (cercano a los 5000), esto puede indicar que en algunas transacciones se presentó fraude haciendo uso incluso de dispositivos que no correspondía de forma adecuada. Esta característica podría ser altamente útil para ser usada a futuro con el fin de legalizar las transacciones realizadas únicamente sobre dispositivos permitidos.

B. Procesamiento del dataset: codificación y desbalanceo de clases

Una vez se crearon las respectivas características, se procedió a trabajar sobre el resto de la preparación del dataset para poder trabajar más adelante con el entrenamiento de los modelos. Debido a la existencia de valores nulos, al desbalanceo y a la existencia de variables categóricas dentro del dataset, fue necesario hacer uso de distintas técnicas para dar solución a cada necesidad. Precisamente ya que en esta sección se trabaja con respecto a un dataset con filas reducidas para aprovechar los recursos de Google Colab y el tiempo de ejecución, se experimenta distintas formas de hacer procesamiento. La Figura 16 muestra el diagrama de flujo que se llevó a cabo para el procesamiento del dataset.

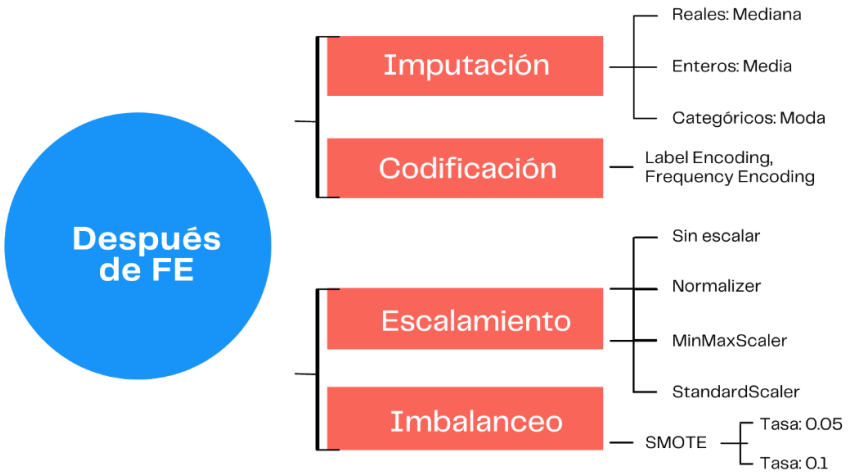


Figura 16. *Descripción del procesamiento llevado a cabo*

En resumen el dataset será sometido a un proceso de imputación, con la idea de eliminar los valores nulos de las columnas; luego será codificado para traducir los valores categóricos a numéricos y que estos puedan ser usados por los modelos a entrenar. Luego del anterior paso, se escaló la información con diferentes técnicas para evaluar como predecía de acuerdo con eso y opcionalmente se evaluó si valía la pena el uso de una técnica de desbalanceo para mejorar la detección de los casos de fraude por los modelos.

Para el proceso de imputación y codificación se realizó una clasificación de las columnas con respecto al tipo de dato que representan, donde por ejemplo, para las variables tipo flotante se reemplazaron los valores nulos por la mediana de esa columna y para las columnas tipo entero se le atribuyeron valores mediante la media. Para las variables categóricas, por otro lado, no sólo se imputaron valores con la moda, sino que también se realizó la conversión de estas columnas a valores numéricos con ayuda de Label Encoding y

Frequency Encoding. Label Encoding fue aplicado sobre columnas con pocos valores únicos y Frequency Encoding sobre columnas con mucha cardinalidad, es decir, columnas con muchos valores únicos. La Figura 17 muestra las líneas de código usadas para el proceso mencionado.

Imputación

```
[ ]  dtipos=['float16', 'float32']
     floats=test_df.select_dtypes(include=dtipos)
     floats=floats.fillna(floats.median())
     dtipos=['int16', 'int32', 'int64']
     enteros=test_df.select_dtypes(include=dtipos)
     enteros=enteros.fillna(enteros.mean())
     categoric=test_df.select_dtypes(include='object')
     categoric=categoric.fillna(categoric.mode().iloc[0])
```

Label Encoding

```
[ ]  cat_columns = train.select_dtypes(include=['object']).copy()

     le = LabelEncoder()

     for feat in cat_columns:
         train[feat] = le.fit_transform(train[feat].astype(str))

     print(train.info())
```

Frequency Encoding

```
[ ]
     # agrupando por frecuencia
     fq = train.groupby('P_emaildomain').size()/len(train)
     # asignando valores al dataframe
     train.loc[:, "{}_freq_encode".format('P_emaildomain')] = train['P_emaildomain'].map(fq)
     # eliminando columna original
     trains = train.drop(['P_emaildomain'], axis = 1)
     fq.plot.bar(stacked = True)
     trains.head(10)
```

Figura 17. *Código de imputación y codificación*

Con respecto al escalamiento, es un paso general, buscar que la información de los datasets no se vea muy afectado por los rangos numéricos de unas columnas con respecto a otras. Por esta razón, se busca generalizar la información y adaptarlo a un rango numérico determinado: con StandardScaler se aplica una transformación relacionada a una media 0 y una desviación estándar 1, con MinMaxScaler la conversión numérica se hace sobre un rango entre [0,1], mientras que para el uso con Normalizer se distribuyen las muestras individualmente a la norma de cada unidad. Se decidió también hacer una evaluación sobre la predicción de los casos en el dataset sin escalar, debido a que la transformación numérica

con el escalamiento pueda no ser adecuado en un dataset con tantas dimensiones. La aplicación de las técnicas de escalamiento se ven reflejadas en la Figura 18.

StandardScaler

```
[ ]  X_scaler = StandardScaler().fit_transform(X)
```

Normalizer

```
[ ]  X_norm = Normalizer().fit_transform(X)
```

MinMaxScaler

```
[ ]  X_minmax = MinMaxScaler().fit_transform(X)
```

Figura 18. *Códigos usados para aplicación de escalamiento*

Por último, se aplica una técnica de desbalanceo con el método SMOTE (*Synthetic Minority Over-sampling Technique*) que genera instancias sintéticas sobre la clase minoritaria. Se usaron dos tipos de tasas de generación para evaluar su comportamiento: 0.05 y 0.1. La Figura 19 muestra la distribución original de las etiquetas 0 y 1 (no fraude y fraude) y el resultado que queda después de aplicar la técnica de desbalanceo. Cabe mencionar que este método se aplicó sobre cada dataset con escalamiento diferente.

Conteo de casos originales en la etiqueta objetivo

```
[ ]  counter = Counter(y)
     print(counter)

     Counter({0: 71040, 1: 1960})
```

SMOTE de 0.05: La etiqueta de fraude pasa de 1960 a 3552 casos.

```
[ ]  #empezar con un valor más pequeño 0.01
     #aplicar selección de modelos para mirar la influencia de los dat
     over = SMOTE(sampling_strategy=0.05)
     #X, y = over.fit_resample(X, y)
     X_train_over, y_train_over = over.fit_resample(X, y)
     counter = Counter(y_train_over)
     print(counter)
     X_trains_005 = pd.DataFrame(X_train_over, columns=X.columns)
     y_trains_005 = pd.DataFrame(y_train_over, columns=['isFraud'])

     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.
       warnings.warn(msg, category=FutureWarning)
     Counter({0: 71040, 1: 3552})
```

SMOTE de 0.1: La etiqueta de fraude pasa de 1960 a 7104 casos

```
[ ]  #empezar con un valor más pequeño 0.01
     #aplicar selección de modelos para mirar la influencia de los dat
     over = SMOTE(sampling_strategy=0.1)
     #X, y = over.fit_resample(X, y)
     X_train_over, y_train_over = over.fit_resample(X, y)
     counter = Counter(y_train_over)
     print(counter)
     X_trains_01 = pd.DataFrame(X_train_over, columns=X.columns)
     y_trains_01 = pd.DataFrame(y_train_over, columns=['isFraud'])

     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.
       warnings.warn(msg, category=FutureWarning)
     Counter({0: 71040, 1: 7104})
```

Figura 19. *Aplicación de técnica SMOTE, con tasa de muestreo de 0.05 y 0.1*

Como resultado final del preprocesamiento nos encontramos varios datasets transformados de acuerdo a varias técnicas y métodos. En un intento inicial por detectar la ruta de entrenamiento que sea más adecuada para el caso de detección de fraude, se aplicará un método de entrenamiento automático que clasifica varios modelos de clasificación al mismo tiempo, permitiendo comparar los modelos por métricas. Esto último permitió reducir los dataset transformados y los modelos a entrenar.

C. Modelos de aprendizaje preliminares

Ya que se decidió explorar varias opciones de dataset, causado por los métodos de escalamiento escogidos, es necesario reducir la exploración de modelos haciendo uso de métodos que permitan abarcar dicho análisis de forma más rápida. Para este caso, se aplicó AutoML, bajo la librería H2O, el cual es un proceso de entrenamiento automático el cual entrega una lista de clasificación de modelos evaluados bajo métricas de clasificación como ROC – AUC. Como se mencionó en la metodología se hicieron procesos automáticos con 15 modelos y con una duración entre 3 y 5 horas. Con base en los resultados se observaron los tres primeros modelos, que se usaron luego para hacer ajuste de hiperparámetros y predicción. El entrenamiento con AutoMl se llevó a cabo únicamente sobre los siguientes datasets, ya que esta comparación se dio para identificar la tasa de muestreo del método SMOTE:

i. StandardScaler con SMOTE 0.05

ii. StandardScaler con SMOTE 0.1

iii. Normalizer con SMOTE 0.05

iv. Normalizer con SMOTE 0.1.

Los resultados de este entrenamiento previo se muestran en la Figura 20.

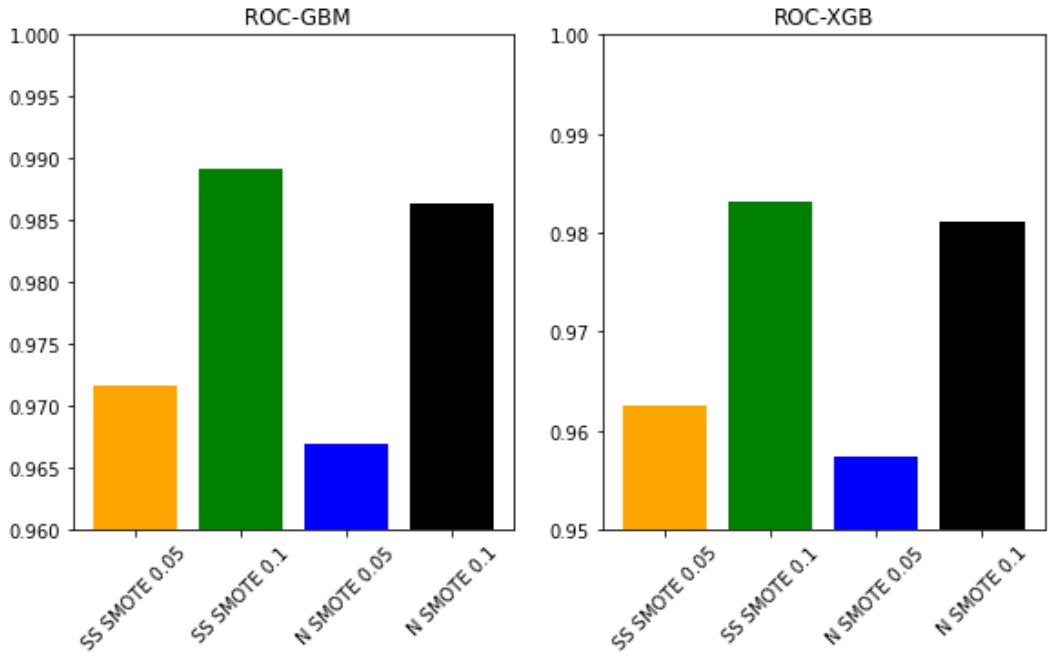
```
aml = H2OAutoML(max_runtime_secs = 3600*5, max_models=15, seed=1, exclude_algos = ["DeepLearning"])
# ir mirando si se necesita aumentar más modelos
aml.train(x=xss_01, y=yss_01, training_frame=trainss_01)

# View the AutoML Leaderboard
lb = aml.leaderboard
lb.head(rows=lb.nrows)
```

```
AutoML progress: |████████████████████████████████████████████████████████| 100%
```

model_id	auc	logloss	aucpr	mean_per_class_error
StackedEnsemble_BestOfFamily_AutoML_20210906_155245	0.990918	0.0382479	0.971106	0.0449066
GBM_grid__1_AutoML_20210906_155245_model_1	0.990896	0.0443207	0.97115	0.0473624
StackedEnsemble_AllModels_AutoML_20210906_155245	0.99074	0.0380389	0.971145	0.0476306
XGBoost_grid__1_AutoML_20210906_155245_model_1	0.983975	0.0543171	0.951491	0.0710028

a)



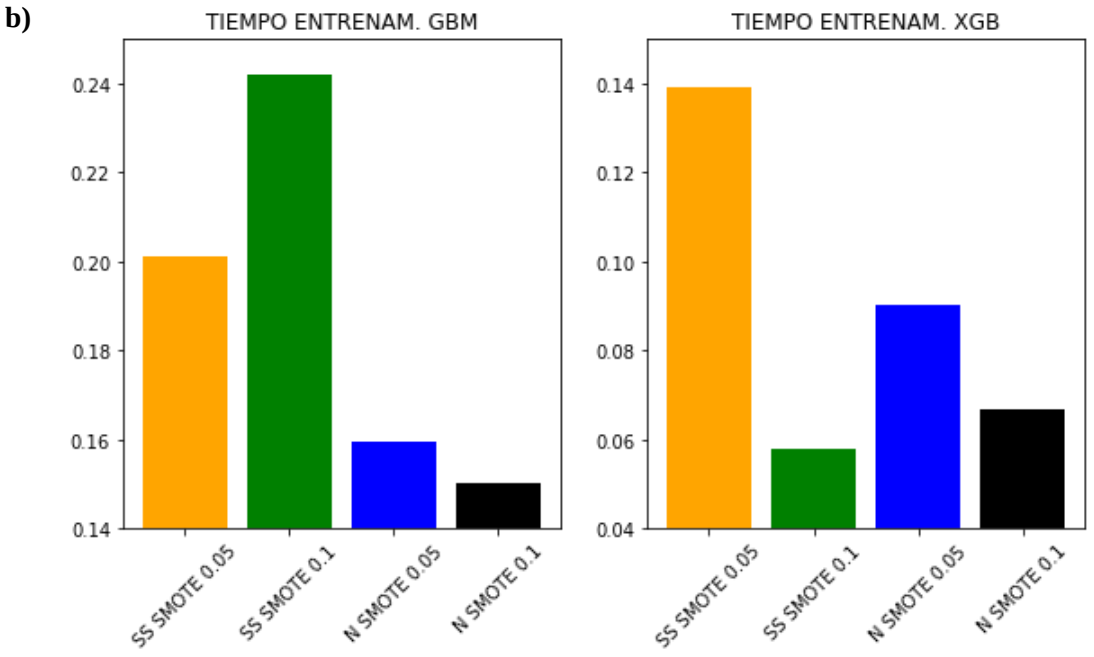


Figura 20. *AutoML: a) Clasificación con métrica ROC para modelos GB y XGB, y b) Tiempo de entrenamiento para GB y XGB*

La aplicación de AutoML permitió definir que los mejores modelos (además de los modelos de ensamble) son GradientBoosting y XGBoost aplicados sobre una dataset estandarizado o normalizado con una estrategia de muestreo de 0.1 (con el método SMOTE), lo que descarta el uso de muestreo con 0.05. Por otro lado, se destaca que GradientBoosting presenta un mayor tiempo de entrenamiento con respecto al otro modelo definido (XGBoost), así que esto puede ser un indicador leve de que el tiempo de ejecución de los códigos pueda llevar un largo tiempo. Con este proceso, se delimitó parte de los tipos de datasets que se usarán para entrenamiento, en este caso se usará solamente aquellos con estrategia de desbalanceo con 0.1 como tasa de muestreo. En caso de que no ocurra buena predicción más adelante se procederá a eliminar el uso de la técnica SMOTE.

D. Ajuste de hiperparámetros, modelos y evaluación

- **Métodos supervisados**

Con el fin de ajustar los valores óptimos de los modelos GradientBoosting y XGBoost se utilizó el proceso GridSearch, el cual de forma exhaustiva hace una búsqueda de cada

parámetro/argumento del estimador con el fin de obtener aquel modelo que mejore verdaderamente la predicción y el entrenamiento. Ya que se observó conforme a los resultados de AutoML que el uso de un escalamiento estandarizado y una estrategia de muestreo de 0.1 presenta un valor ROC cercano a 0.99, se hará ajuste de hiperparámetros sobre este único dataset (pues ya se conoce de antemano que el uso de GridSearch en datasets grandes puede tomar mucho tiempo en entrenar y ajustar los modelos). Los parámetros elegidos con respecto a GradientBoosting y XGBoost haciendo uso de GridSearch se muestran en la tabla a continuación (ver TABLA IV.).

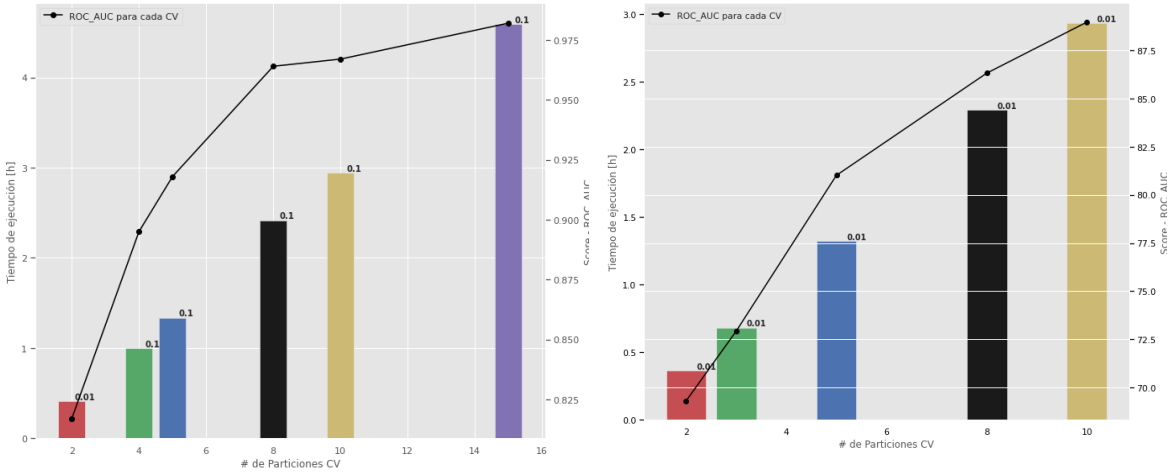
TABLA IV.
RESULTADOS DEL AJUSTE DE HIPERPARÁMETROS CON GRIDSEARCH PARA MODELOS GB Y XGB

Modelo GradientBoosting	
Tiempo de ejecución total: 12 h 3 min 25 s	
Parámetros	**Valor ajustado**
Learning rate	0.1
n_estimators	700
min_samples_split	3
min_samples_leaf	2
max_depth	15
max_features	15
subsample	0.7
cv (splits)*	2,4,5,8,10,15

Modelo XGBoost	
Tiempo de ejecución total: 17 h 18 min 12 s	
Parámetros	**Valor ajustado**
nfolds y seed	3, -1
max_depth	15
gamma	0.0
colsample_bytree	0.6
subsample	0.6
reg_alpha	0.05
reg_lambda	0.005
cv (splits)*	2,3,5,8,10

*Se hace una validación cruzada de la búsqueda con GridSearch sobre la tasa de aprendizaje del modelo, con el fin de conocer su comportamiento y capacidad de aprendizaje. .

Con el ajuste de la mayoría de parámetros de los modelos supervisados elegidos, se hizo un análisis por particiones del entrenamiento de GridSearch y la métrica ROC, realizando una validación cruzada sobre el modelo que fue acabado de ajustar. La gráfica tanto de GradientBoosting y XGBoost se muestra en la Figura 21



GB **XGB**

Figura 21. *Validación cruzada con métrica ROC para los modelos GB y XGB*

Conforme la gráfica que se muestra es posible observar cómo va aumentado la curva ROC con respecto al número de particiones, además del aumento del tiempo de ejecución, siendo XGBoost el modelo que se demora más tiempo (lo que concuerda también con el tiempo medido durante el proceso de GridSearch). Con la idea de observar mejor la predicción de los modelos con respecto a los parámetros ajustado se hace una comparación con ayuda de otra métrica de clasificación, en este caso, con matrices de confusión. Esta métrica permite obtener más información sobre el valor de los verdaderos negativos y verdaderos positivos, los cuales indican el valor real de casos detectados (fraude o no). Con base en lo anterior, se comparan los resultados para los modelos de GB y XGB sobre el dataset con escalamiento estandarizado y normalizado (ver TABLA V.).

TABLA V.
RESULTADOS OBTENIDOS DE MATRIZ DE CONFUSIÓN: VERDADEROS POSITIVOS Y NEGATIVOS PARA DATASETS CON ESCALAMIENTO ESTANDARIZADO Y NORMALIZADO

	StandardScaler SMOTE 0.1	
Modelos	**Verdaderos positivos (TP)**	**Verdaderos negativos (TN)**
GradientBoosting	0.99	0.83
XGBoost	0.99	0.84
	Normalizer SMOTE 0.1	
Modelos	**Verdaderos positivos (TP)**	**Verdaderos negativos (TN)**
GradientBoosting	0.99	0.76
XGBoost	0.99	0.84

De este entrenamiento en particular se observa que el modelo de XGBoost resulta ser el más estable ya que su matriz de confusión resulta muy similar independientemente del

escalamiento (estandarizado o normalizado). Por otro lado, el modelo GradientBoosting falla un poco más en predecir correctamente la etiqueta de fraude sobre el escalamiento normalizado, obteniendo 0.76 en el valor de verdaderos negativos con respecto a cuando se realizó un escalamiento normalizado. En este ensayo de predicción se destaca más el modelo XGBoost de acuerdo a los resultados en la matriz de confusión. Por último, para la evaluación de estos modelos, no sólo se gráfica la curva ROC para determinar su comportamiento con respecto a otras métricas, sino que también se hacen curvas de aprendizaje para determinar si los modelos no se están sobreentrenando. Las curvas ROC se muestran en la Figura 22

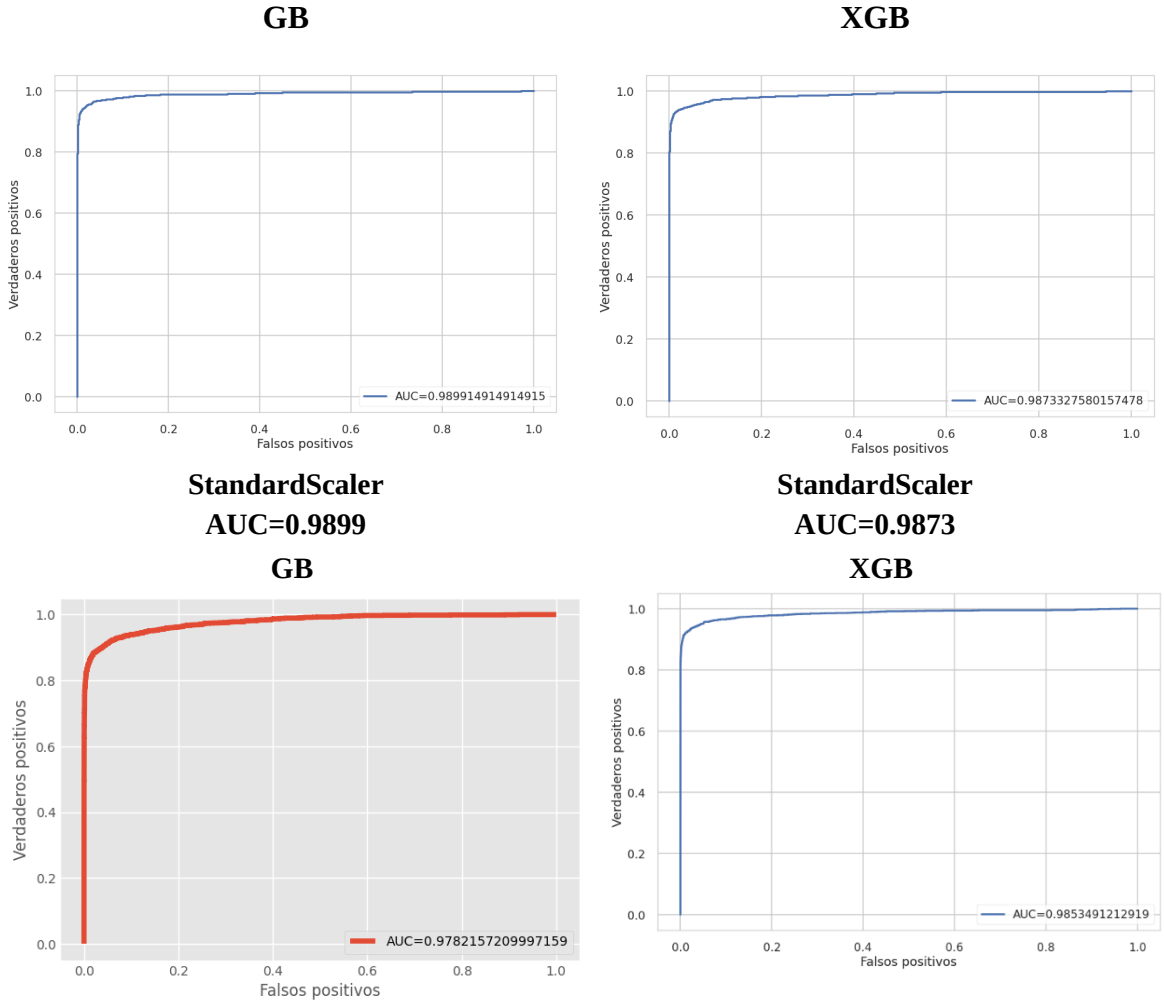


Figura 22. *Resultados obtenidos para modelo GB y XGB de la métrica ROC, se diferencian en términos de escalamiento*

La métrica evaluada en el caso del escalamiento estandarizado y normalizado no presenta una diferencia muy marcada en los cuatros casos en los que se evalúo. Sin embargo,

el uso de StandardScaler como escalamiento es el que presenta mejor rendimiento en la curva ROC, ya sea con el modelo GradientBoosting o XGBoost. En otro orden de ideas, se tomó el modelo realizado con GradientBoosting (tomando en cuenta los parámetros ajustados) y se evalúo el aprendizaje de dicho modelo con base en la métrica ROC -AUC, como se refleja en la gráfica (ver Figura 23).

La curva de aprendizaje nos entrega información sobre como la implementación de GradientBoosting tanto en un proceso de entrenamiento como de validación se mantienen estables a lo largo del cambio del tamaño del dataset. En particular, la curva de validación empieza con una métrica ROC más baja, sin embargo, conforme aumentan las muestras esta va tomando valores cercanos a 1 y asemejándose a la curva graficada para el proceso de entrenamiento (cuya métrica ROC se encuentra relativamente sobre 1).

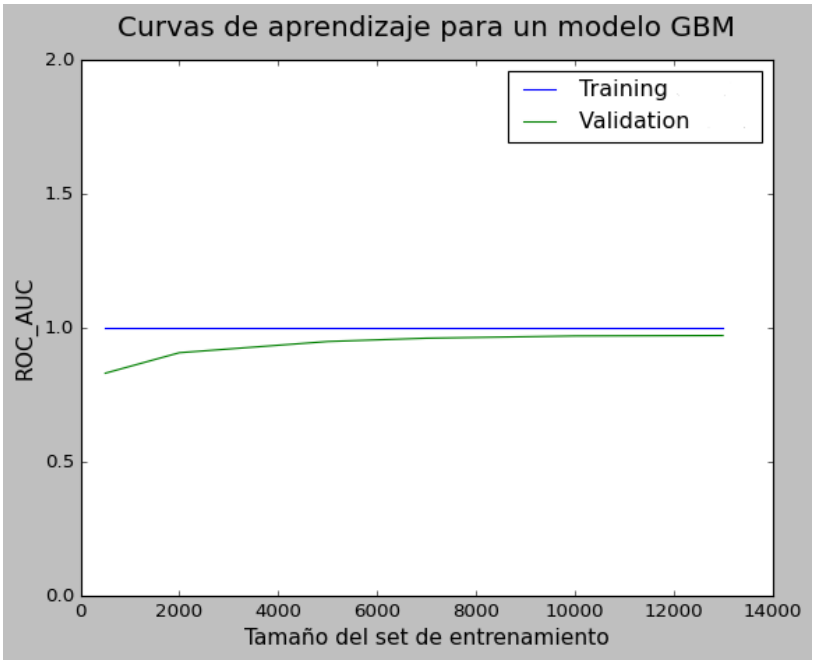


Figura 23. *Curva de aprendizaje para modelo GB*

- **Métodos no supervisados**

Otro tipo de modelo que se usó para predecir se basó en modelos no supervisados, en particular, redes neuronales, cuya función recayó en detectar patrones dentro del mismo dataset que permitieran predecir el fraude. Para la red se define desde el inicio una reducción de dimensiones (como conjunto de entrada), bajo una función de activación (sigmoide) y unas condiciones de entrenamiento definidas (como el número de épocas) se llega hasta una

única capa de salida que se utiliza posteriormente para la predicción. El código de implementación se muestra en la Figura 24, en conjunto con la matriz de confusión que evalúa la predicción.

```
model = Sequential()
model.add(Dense(215, input_dim=430, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
nb_epoch = 200
batch_size = 128
input_dim = X.shape[1] #num of predictor variables,
encoding_dim = 32
hidden_dim = int(encoding_dim / 2)
learning_rate = 1e-3
```

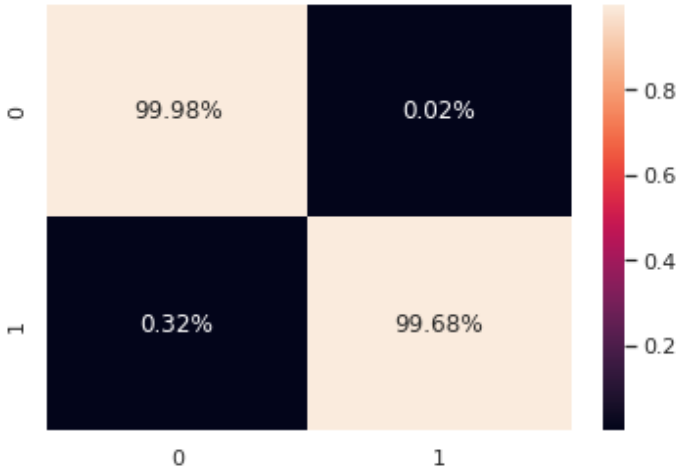
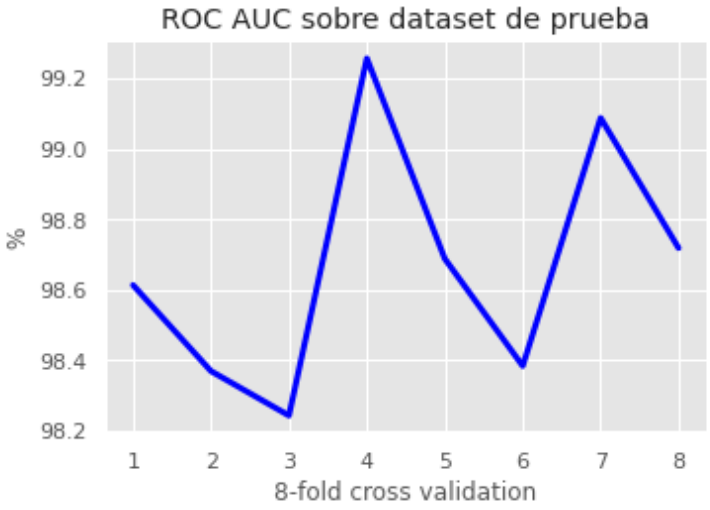
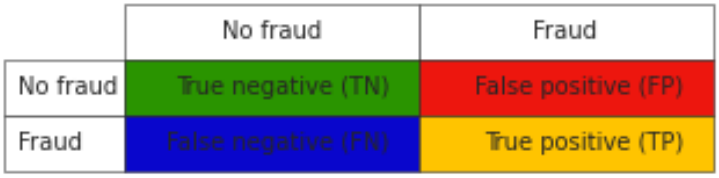


Figura 24. *Rendimiento de red neuronal sobre la predicción de casos de fraude*

Como se muestra, en definitiva la red neuronal parece haber detectado con muy alta eficacia tanto las etiquetas de fraude como las de no fraude. En primera instancia, este modelo sería ideal para usar como futura predicción final, sin embargo, es necesario evaluar su comportamiento con una validación cruzada que determine que tan bien predice con respecto al número de particiones del dataset. En esta situación, la red neuronal predice en particiones desde 2 hasta 15 splits, obteniendo como factor para comparar un porcentaje de verdaderos positivos y verdaderos negativos, justo como se observa en las Figura 25 (a y b).

a)





Composición de matriz de confusión

b)

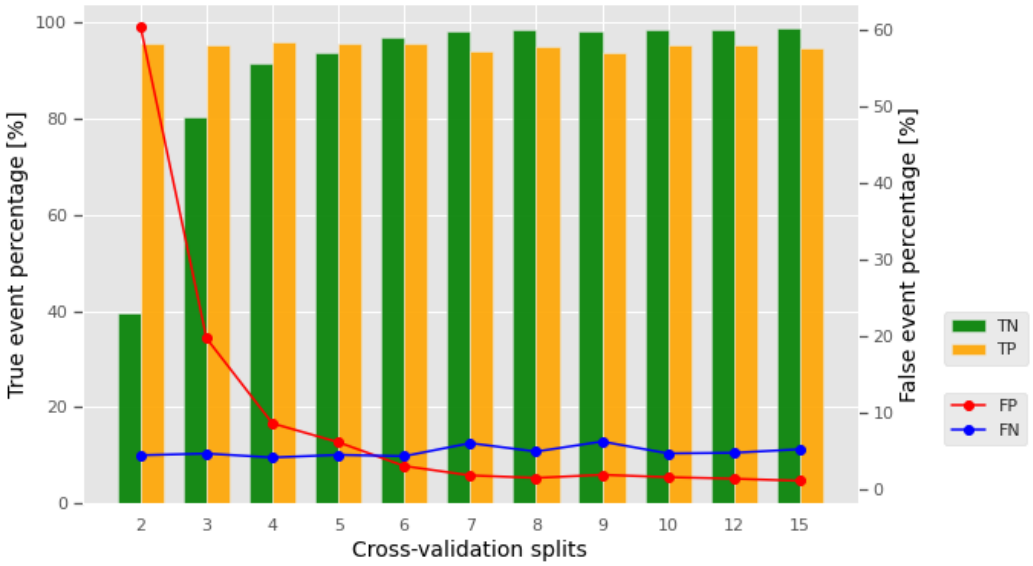


Figura 25. *a) Métrica ROC para red neuronal evaluada con validación cruzada, y b) Validación cruzada con métrica de matriz de confusión*

De la validación cruzada realizada es posible observar la estabilidad de la red neuronal para detectar los casos de fraude (teniendo una tasa de verdaderos positivos cercano a 100/1),

mientras que los verdaderos negativos presentan una tendencia creciente conforme aumenta el número de particiones durante la validación cruzada, la cual también termina siendo una tasa alta de predicción. Este modelo en particular, puede ser una buena opción para predecir fraude sobre el dataset de prueba y que este sea sometido a la competencia de Kaggle. Al mismo tiempo, se puede recalcar como el valor ROC se mantiene entre el rango de 98.2 y 99.2, lo cual es un excelente valor para esta métrica.

- **Competencia Kaggle**

Conforme a la experimentación realizada con los diferentes modelos de aprendizaje: supervisados y no supervisados se pudo detectar que en general presentan una buena predicción, a pesar de no llegar al 100% de casos de fraudes predichos. Como se mencionó anteriormente en esta sección se pasó de predecir y trabajar con respecto a 1/8 del dataset a predecir con el dataset en su tamaño original. A partir de la predicción de probabilidades (que se usa para la determinación de la métrica ROC) se salva la información en un archivo .csv se subió a la competencia de Kaggle para determinar cuan eficiente fue el proceso. El resultado de la calificación por la competencia se observa en la Figura 26.

Figura 26. *Resultado preliminar en Kaggle*

Sorprendentemente, la predicción realizada con el modelo GradientBoosting (sobre el que se ha trabajado) no presentó tan buen comportamiento pues la puntuación pública llegó hasta 69% en una competencia donde la puntuación máxima fue de 96%. Con base en el análisis realizado sobre las probabilidades predichas por el modelo, se detectó que cuando se hizo uso del dataset de prueba se presentaron más casos de fraude (con etiqueta 1) que casos de no fraude.

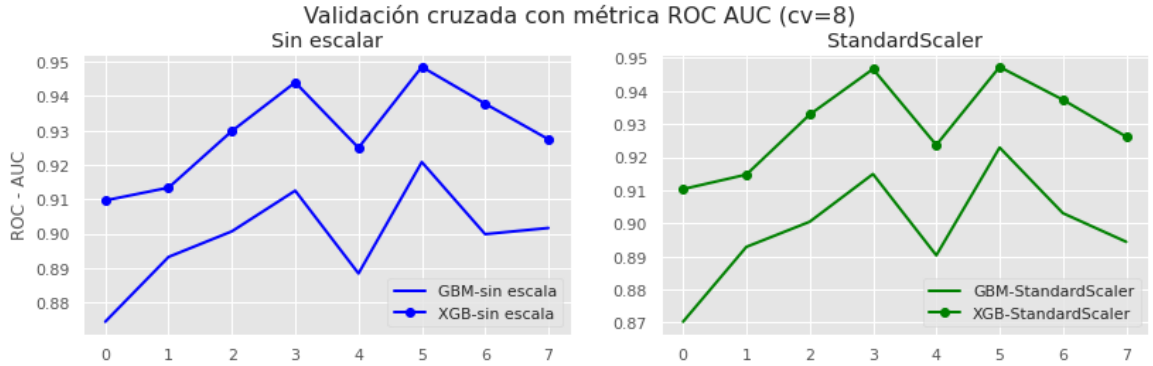
```
array([[     0,   4147],
       [     1, 502544]])
```

El siguiente arreglo refleja la detección del modelo GradientBoosting sobre un dataset de prueba con StandardScaler y SMOTE 0.1. Como se mencionó antes, la clase de no fraude presenta sólo 4147 casos (cuando era la etiqueta más abundante), mientras que la clase de fraude presenta 502544, lo que implica una alta presencia de casos fraudulentos.

El anterior error en la predicción se ligó al uso de la técnica de desbalanceo, pues las etiquetas sintéticas no pudieron ser atribuidas al dataset de prueba debido a la ausencia de la etiqueta objetivo para este dataset. Por esta razón, se elimina el paso de añadir una técnica de desbalanceo al dataset con el que se entrenaron los modelos. En la siguiente sección, se entra en detalle sobre la corrección del entrenamiento y la predicción final de este trabajo.

E. Predicción y resultados finales

En base a lo señalado en la sección previa, se realizó todo el procesamiento sobre los datasets de entrenamiento y prueba eliminando de este proceso el uso de la técnica de desbalanceo. Para evaluar finalmente el comportamiento de las elecciones de procesamiento que se usaron, se aplicaron los modelos ya ajustados (GradientBoosting y XGBoost) y se realizó validación cruzada en base a la métrica ROC, cuya gráfica se puede observar en la Figura 27.



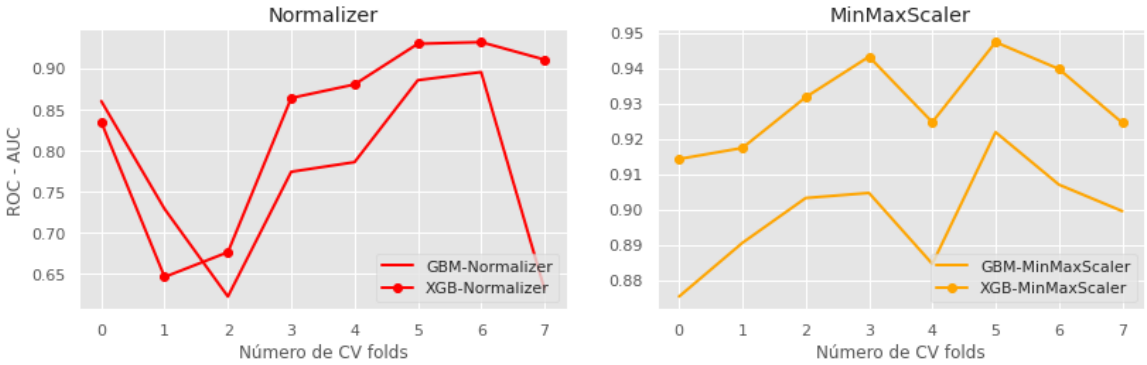


Figura 27. *Validación cruzada de modelos GB y XGB con respecto a diferentes tipos de escalamiento*

La evaluación del comportamiento de los modelos nos da como conclusión:

i. el modelo XGBoost en todos los casos e independiente del escalamiento que se utilice presenta mejor rendimiento sobre la métrica ROC-AUC

ii. se denota como el comportamiento de los escalamientos StandardScaler, MinMaxScaler además del no escalado presentan una tendencia similar sobre cómo se van evaluando las particiones durante la validación cruzada

iii. el escalamiento con normalización presenta mayor variabilidad pues presenta una evaluación ROC-AUC en un rango desde 0.65 hasta 0.90, mientras que para los demás escalamientos presentan un rango desde 0.87 hasta 0.95 (lo que indica un mejor comportamiento).

La predicción de los diferentes modelos entrenados se eleva hasta la competencia de Kaggle para observar si finalmente se logró mejorar la primera subida que se realizó anteriormente (con el uso del método SMOTE). La Figura 28 muestra los mejores resultados obtenidos de mayor a menor dentro de la competencia Kaggle.

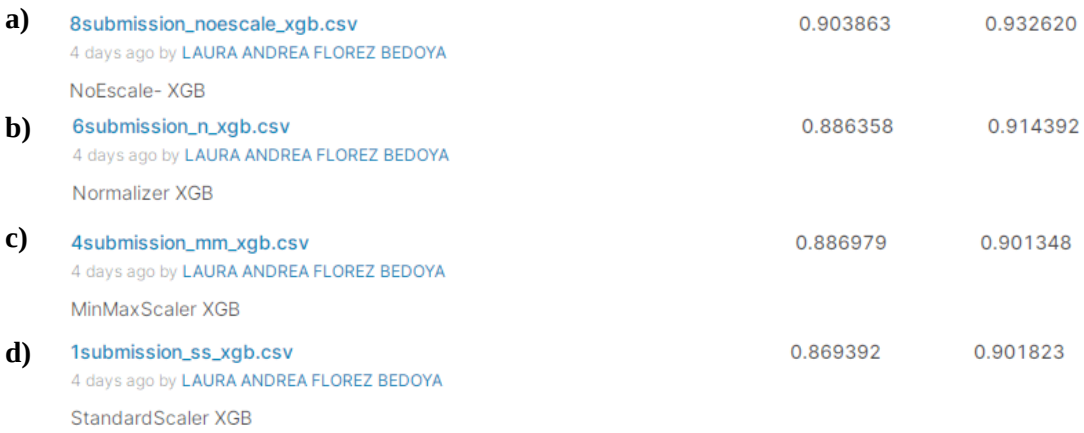


Figura 28. *Puntuación obtenida en la competencia de Kaggle de acuerdo a diferentes predicciones subidas: a) Sin escala, b) Normalizer, c) MinMaxScaler y d) StandardScaler*

Los resultados obtenidos demuestran finalmente, que el uso de la técnica SMOTE si afectó en gran medida el desempeño de predicción de los modelos, llevándolos a un punto de bajo rendimiento. Con respecto a la participación en la convocatoria se destaca que el mejor resultado se da sobre el modelo XGBoost sobre un dataset que no sufrió alguna conversión numérica drástica, es decir, que no fue escalado; obteniendo un puntaje en la clasificación pública de la competencia por encima del 93%, donde la predicción ganadora de dicho concurso llegó únicamente hasta 96%. Una curiosidad resultante de esta última parte recae en cómo el modelo GradientBoosting no cumplió adecuadamente con la predicción, siendo completamente superado en puntuación por XGBoost.

VI. DISCUSIÓN

Este estudio se realiza con el objetivo de conocer la influencia de diferentes aplicaciones de Machine Learning sobre la detección de fraude de transacciones electrónicas. El objetivo principal de este estudio no sólo era explorar el uso del aprendizaje automático sobre algún tipo de tema específico, sino también conocer la relación entre las características entregadas por la competencia de Kaggle y una correcta identificación de fraude. A lo largo de la exploración de la información proporcionada por la organización Vesta (patrocinador de la competencia) se plantearon algunas hipótesis que ayudaron en la creación de otro tipo de variables destinadas a apoyar la detección del fraude.

La primera hipótesis sugería que existía una dependencia en el tiempo con respecto al número de casos de fraude presentados, lo cual fue comprobado al observar una tendencia

en términos de horas y meses con respecto al promedio de los casos de fraude. Se destaca así mismo, como el delta de tiempo de tanto el dataset de entrenamiento y prueba indicaban hacer parte del registro de transacciones electrónicas de casi un año completo, donde la característica del delta del tiempo de la transacción se encontraba en términos de los segundos de un día (siendo 86400 el menor valor de dicha columna). Tras la aceptación de esta conjetura, se planteó otra hipótesis relacionada a la dependencia de los dominios de correos electrónicos registrados por transacción con alguna acción fraudulenta. De acuerdo con el análisis realizado de dichos dominios, los correos electrónicos provenientes de compañías conocidas como Google, Yahoo y Microsoft son los que más presentan casos de fraude, de hecho, es posible encontrar en cualquier portal de internet como los dominios de estas compañías mencionadas son precisamente unos de los más peligrosos pues no proporcionan buena seguridad [21]. Así mismo, fue posible observar que el uso de estos dominios estaba relacionado a los picos en horas y meses los casos de fraude presentados.

Una tercera suposición se construyó como una nueva característica, con la idea de que pudiera describir la relación de los casos de fraude con el número de cifras decimales que contenía el monto registrado por cada caso. De dicho análisis, se pudo concluir varios aspectos: por un lado, gran parte de los casos se concentran en una región (que por la competencia se podría deducir que es Estados Unidos) con una sola cifra significativa y en otro tipo de región/regiones que implicaran el uso de conversión de moneda (ya que presentan más de 2 cifra significativas). Después del análisis de todas estas características y suposiciones, se evalúo por último, la influencia del dispositivo en el que se realizó la transacción de acuerdo al fraude transaccional. Se descubrió que cierta cantidad de casos marcados como fraudulentos presentan una discrepancia entre la información del dispositivo registrado en términos de marca, modelo y tipo de dispositivo.

Con este estudio se logró detectar de forma adecuada transacciones electrónicas fraudulentas o no fraudulentas con base en una serie de hipótesis, procesamiento y entrenamiento de modelos de aprendizaje automático. Del análisis de las conjeturas planteadas es posible indicar algunas recomendaciones en aplicaciones de detección con Machine Learning para este tipo de situaciones:

- Debido al gran volumen de información, recopilada no sólo del sistema transaccional usado, sino también de características creadas por el veedor de dichos sistemas como es la organización Vesta, sería adecuado construir indicadores que recopilaran información sobre varias características sobre una sola variable, por ejemplo, características con indicios de que el dispositivo con el que se realizó la transacción ha sido autentificado por el usuario de la cuenta bancaria.

- Por otro lado, los sistemas de detección de fraude deberían aumentar la verificación de solicitudes de transacción especialmente en dominios de correo electrónico que se han demostrado son los menos seguros y fáciles de interceptar, como se reflejó en el análisis del estudio siendo Google, Yahoo y Microsoft los más fraudulentos.

Para mayor discusión, revisión y análisis del trabajo que se desarrolló se puede acceder al repositorio a través del siguiente enlace: github.com/fblaura/FraudD

VII. CONCLUSIONES Y TRABAJOS A FUTURO

Como se pudo comprobar en el desarrollo de los modelos de aprendizaje, la tasa de representación de los casos de fraude con respecto a los de no fraude, indican un desbalanceo considerable de casos. En un inicio, se planteó modificar el número de casos de fraude para el entrenamiento de forma que los modelos pudieran identificar mejor los casos de fraude en el momento de la prueba y validación conforme a nuevos datos. Con la realización de este ensayo se comprobó que la adición de datos sintéticos sobre el dataset no mejoró la predicción, por el contrario, generó que los modelos al intentar predecir sobre información nueva se equivocarán obteniendo un bajo rendimiento sobre la curva ROC. La predicción obtenida con el uso de la técnica SMOTE fue evaluada con la métrica ROC-AUC en la competencia de Kaggle, logrando una puntuación de 69%. Esto llevó a cambiar el enfoque del experimento, eliminando el uso de la técnica SMOTE y trabajando sobre el desbalanceo original de la información.

El tamaño y la complejidad de los datos financieros recopilados de diferentes fuentes y/o dispositivos va en gran crecimiento, generando una necesidad por trabajar con modelos de aprendizaje automático para obtener conocimientos y hacer predicciones sobre cualquier tipo de necesidad específica. En particular, el estudio muestra que el enfoque basado en

aprendizaje automático supervisado logró obtener un buen rendimiento, caracterizado por un 93% de predicción adecuada, a pesar de que en aplicaciones financieras es preferiblemente más deseado una mayor precisión (para componen sistemas de detección más fiables).

En general, los resultados de este estudio indican que vale la pena seguir trabajando con la creación de características que reduzcan la dimensionalidad de la información y su vez recopilen con indicadores información con respecto a alguna validación de la transacción. En conclusión, el desarrollo de este trabajo cumple con el propósito de detectar el fraude sobre transacciones electrónicas recopiladas de información comercial a partir de la exploración de algoritmos de Machine Learning y los subprocesos que se involucran en este tipo de aplicaciones, como: limpieza, exploración y procesamiento de datos.

VIII. REFERENCIAS

[1] John O. Awoyemi, Adebayo O. Adetunmbi, and Samuel A. Oluwadare, "Credit card fraud detection using Machine Learning Techniques:A Comparative Analysis," in *Proceedings of the IEEE International Conference on Computing, Networking and Informatics (ICCNI 2017)*, 2017, pp. 1–3.

[2] M. Cox and D. Ellsworth, "Application-Controlled Demand Paging for Out-of-Core Visualization," *IEEE Xplore*, pp. 1–2, 1997.

[3] E. Raguseo, "Big data technologies: An empirical investigation on their adoption, benefits and risks for companies," *Int. J. Inf. Manage.*, vol. 38, no. 1, pp. 187–188, Feb. 2018, doi: 10.1016/j.ijinfomgt.2017.07.008.

[4] G. Koffi Kalipe and Rajat Kumar Behera, "Big Data Architectures: A Detailed and Application Oriented Analysis," *Int. J. Innov. Technol. andExploring Eng.*, vol. 8, no. 9, pp. 2182–2183, Jul. 2019.

[5] A. De Mauro, M. Greco, and M. Grimaldi, "A formal definition of Big Data based on its essential features," *Libr. Rev.*, vol. 65, no. 3, pp. 1–3, Apr. 2016, doi: 10.1108/LR-06-2015-0061.

[6] M. K. Saggi and S. Jain, "A survey towards an integration of big data analytics to big insights for value-creation," *Inf. Process. Manag.*, vol. 54, no. 5, pp. 758–790, Sep. 2018, doi: 10.1016/j.ipm.2018.01.010.

[7] M. I. Baig, L. Shuib, and E. Yadegaridehkordi, "Big data adoption: State of the art and research challenges," *Inf. Process. Manag.*, vol. 56, no. 6, Nov. 2019, doi: 10.1016/j.ipm.2019.102095.

[8] S. M. Basha and D. S. Rajput, "Survey on Evaluating the Performance of Machine Learning Algorithms: Past Contributions and Future Roadmap," in *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, vol. 1, Elsevier, 2019, pp. 153–155.

[9] O. Obulesu, M. Mahendra, and M. ThrilokReddy, "Machine Learning Techniques and Tools: ASurvey," in *Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018)*, 2018, pp. 605–606.

[10] PwC - PricewaterhouseCoopers, "Economic crime: PwC's Global Economic Crime and Fraud Survey," 2020. [Online]. Available: www.pwc.co.za.

[11] I. Sadgali, N. Sael, and F. Benabbou, "Performance of machine learning techniques in the detection of financial frauds," in *Procedia Computer Science*, 2019, vol. 148, pp. 46–48, doi: 10.1016/j.procs.2019.01.007.

[12] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, 1st ed., vol. 1. Springer, 2019.

[13] J. Alzubi, A. Nayyar, and A. Kumar, "Machine Learning from Theory to Algorithms: An Overview," in *Journal of Physics: Conference Series*, Nov. 2018, vol. 1142, no. 1, pp. 1–2, doi: 10.1088/1742-6596/1142/1/012012.

[14] R. Roy and T. George K, "Detecting Insurance Claims Fraud using Machine Learning Techniques," *2017 International Conference on circuits Power and Computing Technologies [ICCPCT]*, IEEE, pp. 1–3, 2017.

[15] P. Tiwari, S. Mehta, N. Sakhuja, J. Kumar, and A. K. Singh, "Credit Card Fraud Detection using Machine Learning: A Study," *Cornell University ArXiv*, pp. 2–5, Aug. 23, 2021.

[16] S. Rajora *et al.*, "A Comparative Study of Machine Learning Techniques for Credit Card Fraud Detection Based on Time Variance," *IEEE Comput. Intell. Soc.*, vol. 1, pp. 1958–1960, 2018.

[17] A. A. Soofi and A. Awan, "Classification Techniques in Machine Learning: Applications and Issues," *J. Basic Appl. Sci.*, vol. 13, pp. 459–461, 2017.

[18] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, "Learning Feature Engineering for Classification.pdf," *Int. Jt. Conf. Artifical Intell.*, vol. 17, pp. 2529–2535, 2017, doi: 10.24963-ijcai.2017 -352.

[19] G. T. Reddy *et al.*, "Analysis of Dimensionality Reduction Techniques on Big Data," *IEEE Access*, vol. 8, pp. 54776–54788, 2020, doi: 10.1109/ACCESS.2020.2980942.

[20] G. Dong and H. Liu, *Feature Engineering for Machine Learning and Data Analytics*. Taylor & Francis Group, 2018.

[21] R. Mukherjee, "Best secure email providers of 2021," *techradar.pro*, 2021. https://www.techradar.com/best/best-secure-email-providers.