



Trabajo Práctico Inicial

Laboratorio de Construcción de Software

Proyecto Profesional I

Alumnos:

Sanchez Rodriguez, Camila (41024628/2018)

Sanchez Rodriguez, Joaquín (42948062/2019)

Docentes:

Ing. Juan Carlos Monteros

Ing. Francisco Orozco de la Hoz

Lic. Leandro Dikenstein

Aula:

7116

Periodo:

Primer semestre 2023

Índice

Índice	2
Introducción	3
Historial de cambios	3
Investigación	4
Tema a elección y justificación	5
Plataformas y herramientas	6
Modelo y método	8
Estructura del trabajo a realizar	10
Diagramas	14
WBS	16
Diccionario WBS	17
Avances en el proyecto	21
Primera parte	21
Recolección, preparación y división de datos	23
Preparación de datos	26
Extracción de features	27
Eliminación de features o columnas redundantes	29
Segundo análisis	29
Interpretación del modelo	49
Despliegue en la nube	50
Introducción	50
Actualización de herramientas utilizadas en el proyecto	51
Estructura del desarrollo	52
Incorporación del modelo de machine learning en la aplicación web	52
Planteamiento de la aplicación	54
Cargar archivos	55
Procesar archivos	55
Consultar manualmente	56
Obtener clasificación	56
Predecir tipos de datos	57
Dificultades y consideraciones	57
Imágenes	59
Despliegue y conclusiones	64
Lecciones aprendidas	64
Bibliografía	65

Introducción

El siguiente trabajo práctico tiene la finalidad de servir como “warming up” o entrada en calor. Se espera la implementación en uso de herramientas útiles para la realización del posterior trabajo práctico central de la materia.

Se plantea la evaluación en la introducción de tomas de decisiones, planificación y ejecución del proyecto donde se tendrán etapas de investigación, diseño de la solución con WBS, desarrollo y posterior implementación en la Nube.

Historial de cambios

Nombre	Fecha	Propietario de la modificación	Descripción
Nuevo documento	09/03/2023	Sanchez, Camila	Creación de documento para el informe
Actualización	09/03/2023	Sanchez, Joaquin	Configuración de entornos de organización
Actualización	10/03/2023	Sanchez, Camila Sanchez, Joaquin	Recopilación de información
Actualización	11/03/2023	Sanchez, Camila Sanchez, Joaquin	Análisis de información recopilada Actualización de documento de informe
Nuevo documento	16/03/2023	Sanchez, Camila	Creación diagrama de flujo
Actualización	17/03/2023	Sanchez, Camila	Incorporación de información sobre investigación al informe
Nuevo documento	17/03/2023	Sanchez, Joaquin	Creación de diagrama WBS y diccionario
Actualización	17/03/2023	Sanchez, Camila	Revisión de diagramas y diccionario Incorporación en informe
Actualización	18/03/2023	Sanchez, Joaquin	Actualización diagrama WBS y diccionario
Actualización	19/03/2023	Sanchez, Camila	Actualización informe avances del proyecto
Actualización	26/03/2023	Sanchez, Camila	Actualización informe avances del proyecto Actualización diagramas
Actualización	01/04/2023	Sanchez, Joaquin	Implementación en la nube
Actualización	02/04/2023	Sanchez, Joaquin Sanchez, Camila	Actualización implementación en la nube

Investigación

Machine Learning

El **Machine Learning** es una rama de la **inteligencia artificial** encargada de crear programas capaces de generalizar comportamientos a partir de los datos recibidos. Hoy en día esta tecnología cada vez tiene un mayor valor puesto que da un nuevo uso y sentido a los datos con que cuentan las organizaciones. Se puede encontrar Machine Learning en:

- Los buscadores web (*clasificación de contenido*)
- Las redes sociales (*reconocimiento de rostros, perfilamiento, recomendaciones de amigos*)
- Medicina (*Modelos predictivos de enfermedades, secuencia genética, etc.*)
- Predicciones de juegos

Tipos de aprendizajes automáticos

Aprendizaje supervisado

El **aprendizaje supervisado** es aquel que para un conjunto de datos de entrada conocemos de antemano los datos correctos de salida. Se puede decir que consta de dos fases:

- En la **fase de entrenamiento** se cuenta con un conjunto de datos (por lo general, entre el 70% u 80% del total de la data disponible) que son con los que se enseña o entrena al algoritmo para encontrar los patrones y relaciones en la data.
- Posteriormente, en la **fase de validación**, se cuenta con una data de pruebas (entre el 30% o 20% del total de data disponible), la cual sirve para validar el rendimiento del algoritmo.

Aprendizaje no supervisado

En el **aprendizaje no supervisado** para el conjunto de datos de entrada, **no se conoce** de antemano los datos de salida. Se utiliza para obtener una agrupación coherente de los datos en función de las relaciones entre las variables definidas en la data.

Problemas de regresión y clasificación

Regresión

Se podría decir que se utiliza regresión cuando el resultado es un número. Es decir, el resultado de la técnica de Machine Learning que estemos usando será un valor numérico dentro de un conjunto infinito de posibles resultados.

Algunos ejemplos posibles para la aplicación de un algoritmo de regresión podría ser:

- Predecir por cuánto se va a vender una propiedad inmobiliaria
- Predecir cuánto tiempo va a permanecer un empleado en una empresa
- Estimar cuánto tiempo va a tardar un vehículo en llegar a su destino
- Estimar cuántos productos se van a vender

Clasificación

Se utiliza clasificación cuando **el resultado es una clase**, entre un número limitado de clases. Con clases nos referimos a categorías arbitrarias según el tipo de problema. Por ejemplo, si se quiere detectar si un correo es spam o no, sólo hay dos clases, por lo que el algoritmo de Machine Learning de clasificación, tras darle un correo electrónico, tendría que determinar a qué clase pertenece: spam o no spam.

Tema a elección y justificación

El tema a elección para la investigación y aplicación de conceptos de Machine Learning fue **aplicación de algoritmos de ML en el procesamiento de datos respecto a detección de fraude**. Se seleccionó dicho tema debido a la relación que puede tener respecto al trabajo principal. Por otro lado, nos decidimos concentrar en **la detección de fraude respecto a las tarjetas de crédito**, esto se debe a que tuvimos cierta experiencia con un trabajo práctico de la materia de bases de datos donde se tenían que detectar posibles transacciones fraudulentas revisando determinadas variables en las tablas.

Machine Learning y la detección de fraude

Hoy en día, la tecnología Machine Learning es aprovechada por bancos y otras instituciones financieras para detectar actividad sospechosa de forma rápida y a gran escala. Esto es debido a que los modelos de ML son útiles para identificar fraude potencial y detectar comportamientos sospechosos antes de que el fraude ocurra.

Beneficios del Machine Learning para el manejo de fraudes

Con la implementación de algoritmos de ML se permite que las máquinas puedan realizar un mejor trabajo procesando grandes conjuntos de datos por lo que se obtiene la habilidad de parcializar y apuntar grandes cantidades de información. Esto conlleva los siguientes beneficios:

- **Detección más rápida y más eficiente:** se detectan rápidamente patrones sospechosos que al personal humano le costaría meses.
- **Reducción en el tiempo de revisión manual:** el tiempo invertido en la revisión manual se reduce cuando se deja a la máquina analizar la información.
- **Mejores predicciones en grandes conjuntos de información:** Mientras más información se le provea al motor de ML, más entrenado llega a ser.
- **Solución económica y efectiva:** Evitar la contratación excesiva de agentes de operaciones de riesgos.

Desventajas del Machine Learning para el manejo de fraudes

A pesar de tener sus ventajas, siempre habrá casos donde las revisiones manuales serán necesarias y preferibles. Algunas desventajas que se pueden presentar son las siguientes:

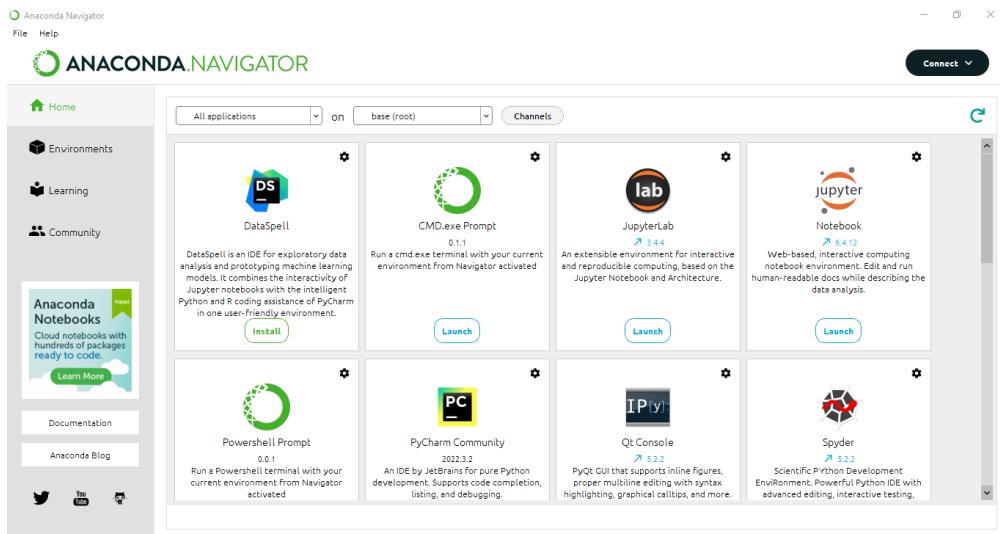
- **Menor control:** el motor de ML puede cometer errores.
- **Positivos falsos:** como el motor puede cometer errores, puede ocurrir que haya positivos falsos, es decir, puede ocurrir que haya acciones legítimas que sean marcadas como fraude, por lo que afectará el sistema negativamente.

Plataformas y herramientas

Para la ejecución del trabajo hemos decidido utilizar y descargar el sistema de **Anaconda**¹, una distribución de los lenguajes de programación **Python** para computación científica. Para esto hemos descargado Anaconda Navigator pues es una interfaz que permite inicializar aplicaciones y administrar fácilmente paquetes y entornos.

¹ <https://www.anaconda.com/products/distribution>

PP1/LS - TP Inicial Machine Learning

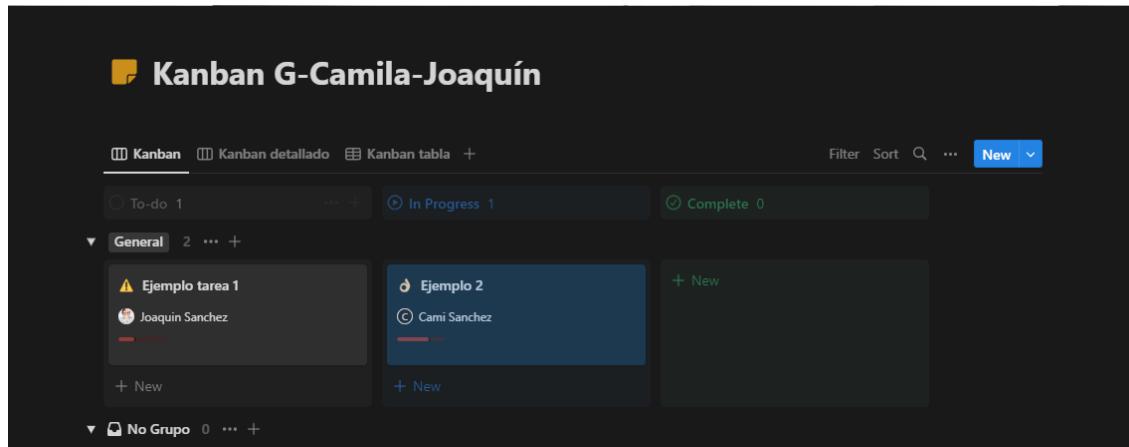


Dentro de Navigator, hemos decidido descargar y utilizar **Jupyter Notebooks**. Para investigar más respecto a la configuración de entorno hemos recurrido al siguiente enlace: <https://www.juanbarrios.com/como-instalar-anaconda-3-en-windows-11/>, en el cual esclarece dudas sobre la instalación de librerías.

A screenshot of the Jupyter Notebook interface. At the top, there are 'Quit' and 'Logout' buttons. Below that is a navigation bar with tabs for 'Files', 'Running', and 'Clusters'. A message says 'Select items to perform actions on them.' On the left is a file tree showing a directory structure with folders like anaconda, Contacts, Desktop, eclipse, Favorites, git, Links, OneDrive, Python - ML, Saved Games, and Searches. On the right is a table listing files with columns for Name, Last Modified, and File size. The table includes rows for 'anaconda', 'Contacts', 'Desktop', 'eclipse', 'Favorites', 'git', 'Links', 'OneDrive', 'Python - ML', 'Saved Games', and 'Searches'.

Para la organización del trabajo, hemos recurrido a la utilización de la plataforma **Notion²**, el cual provee la opción de poder incorporar planillas de trabajo tipo Kanban y la elección de poder compartir el acceso con otros usuarios. Por otro lado, también se decidió la utilización de **Google Drive** para el almacenaje de archivos e información que pueda llegar a utilizar.

² <https://www.notion.so/product>



En cuanto a la herramienta de versionado de código, se acordó trabajar con la plataforma **Github**.

El enlace al repositorio es el siguiente: <https://github.com/lc-sanchez/TP1-MachineLearning>

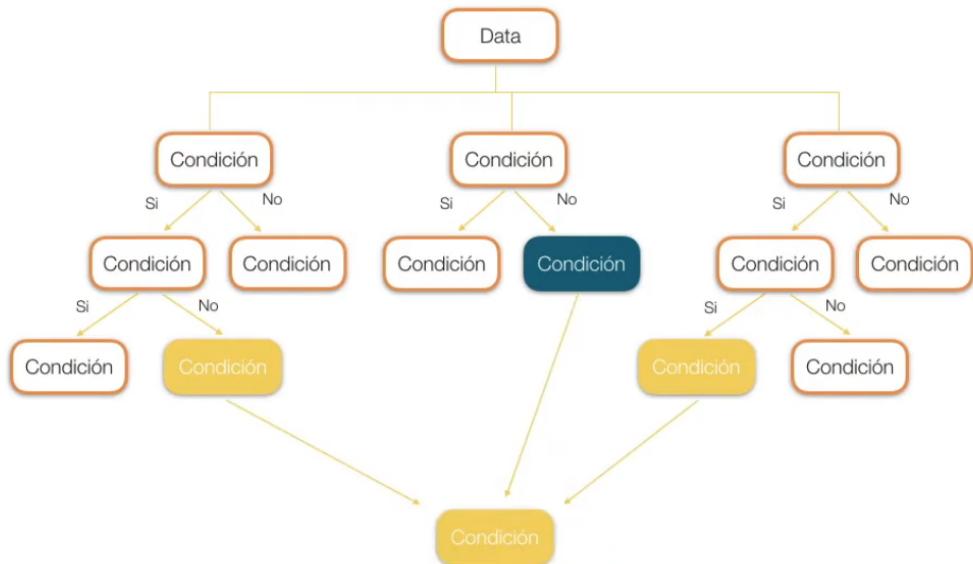
Modelo y método

Gracias al material aportado por los docentes y sumado a la investigación realizada mediante diferentes páginas web y videos, se decidió, por ahora, utilizar el método **Random Forest** en conjunto al tipo de Machine Learning de **Supervised Learning o Aprendizaje Supervisado**. En dicho aprendizaje, como se explicó anteriormente, el Data Scientist establece qué tipos de datos deben relacionarse con ciertos elementos concretos para que la máquina pueda hacer el resto del trabajo. Por lo tanto, se deben introducir los inputs y outputs para que la tecnología pueda hallar patrones en la información.

Como se ha de introducir un conjunto de datos, hemos recurrido a la plataforma de **Kaggle**³ donde hemos podido indagar algunos *datasets* interesantes. Sin embargo, todavía no hemos seleccionado alguno en particular para el trabajo.

Retomando el algoritmo de *Random Forest*⁴, lo orientaremos a una **implementación de clasificación** puesto que el problema que nos percatamos es sobre clasificar si una operación con tarjeta es posible fraude o no. En dicho algoritmo, tenemos una data y se la divide en n árboles de decisión, posteriormente se creará cada árbol de decisión y se extenderá, dando al final una determinada condición y la respuesta que haya tenido más resultados será la predicción del análisis.

Aprendizaje Supervisado: Random Forest Classification



³ <https://www.kaggle.com/>

⁴

<https://www.youtube.com/watch?v=VH7eLWsLCks&list=PLJjOveEiVE4Cbbx1dVjydfmPPpj0pg86&index=14>

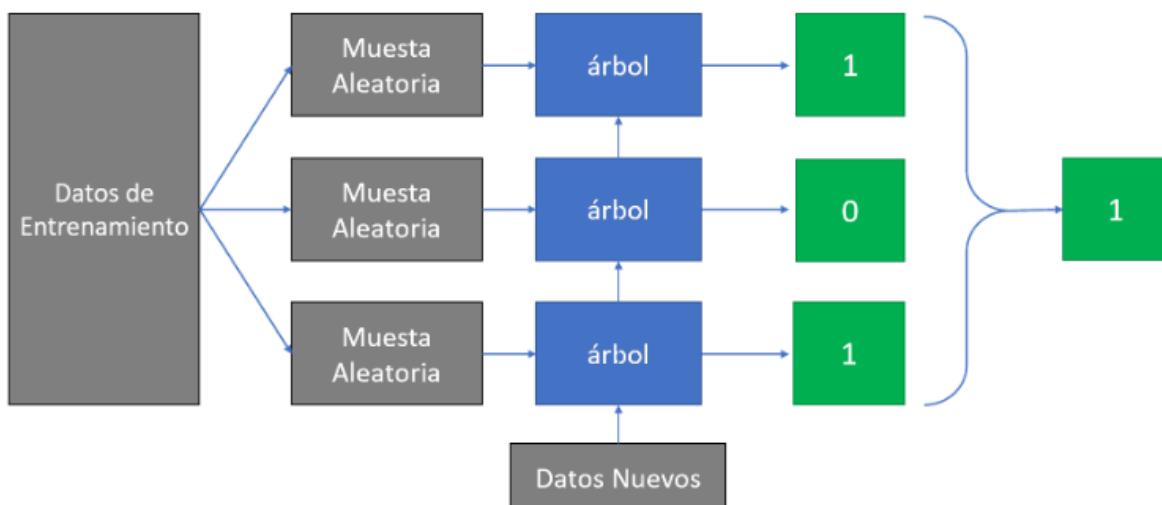
Ventajas de Random Forest

Algunas de las ventajas que presenta este algoritmo son las siguientes:

- Es uno de los algoritmos de aprendizaje más certeros que hay disponible. Si se aporta un dataset lo suficientemente grande produce un clasificador muy certero.
- Puede manejar cientos de variables de entrada sin excluir ninguna.
- Es capaz de dar estimaciones de qué variables son importantes en la clasificación.
- Posee un método eficaz para estimar datos perdidos y mantener la exactitud cuando una gran proporción de datos está perdida.

Random Forest y Bagging

Como se dijo anteriormente, un Random Forest es un conjunto (*ensemble*) de árboles de decisión combinados con bagging. Pero, ¿Qué es bagging? Bagging es un método de aprendizaje conjunto que es comúnmente utilizado para reducir la varianza dentro de un dataset ruidoso. En el caso de Random Forest, distintos árboles ven distintas porciones de datos por lo que ningún árbol ve todos los datos de entrenamiento. Esto hace que cada árbol se entrene con distintas muestras de datos para un mismo problema. Por lo tanto, al combinar sus resultados, unos errores se compensan con otros y tenemos una predicción que generaliza mejor, es decir, una mejor capacidad de obtener buenos resultados con datos nuevos.



Estructura del trabajo a realizar

En una primera instancia la aplicación del algoritmo consta de los siguientes pasos:

1. Entendimiento del problema
2. Criterio de evaluación
3. Recolectar los datos
4. Preparar los datos
5. Elegir el modelo
6. Entrenar nuestra máquina
7. Evaluación
8. Configuración de parámetros
9. Interpretación del Modelo
10. Implementar el Modelo como Servicio

Entendimiento del problema

Uno de los primeros pasos, y más importante, es entender el problema que se desea resolver. Entender dicho problema conlleva tiempo, y aún más si proviene de un sector en el que se tiene poco conocimiento. No entender el problema nos puede llevar a construir un modelo incorrecto y que falle.

Criterio de evaluación

Es necesario establecer un criterio de evaluación para determinar la calidad de nuestro modelo. El criterio de evaluación generalmente se trata de una medida de error.

Recolectar los datos

Dada la problemática, se deberá investigar y obtener los datos que se utilizarán para alimentar el modelo. Importa la **cantidad** y la **calidad** de información que se consiga puesto que impactará en el rendimiento de nuestro modelo.

Las fuentes de extracción pueden ser bases de datos ya existentes o creadas desde cero.

Preparar los datos

- **Mezclar los datos:** Es importante que el orden en que se procesen los datos dentro del modelo no sea determinante.
- **Visualizar y buscar correlaciones:** Este paso permitirá seleccionar características que están más correlacionadas entre sí y con la variable objetivo.

- **Balancear los datos:** Se debe tener balanceada la cantidad de datos que tenemos para cada resultado, para que sea representativo, ya que si no, el aprendizaje podrá ser tendencioso hacia un tipo de respuesta y cuando nuestro modelo intente generalizar el conocimiento, fallará.
- **Separar los datos (Entrenamiento y validación):** Se debe separar los datos en dos grupos: uno para **entrenamiento** y otro para **evaluación** del modelo. Se puede fraccionar aproximadamente en una proporción 70/30 u 80/20, pero puede variar según el caso y el volumen de los datos que tengamos.
- **Normalización, eliminar duplicados y hacer correlación de errores.**

Elegir el modelo

Se debe elegir un modelo de acuerdo al objetivo que tengamos. En resumen tenemos modelos de clasificación, predicción, regresión, clustering, reducción de dimensión, deep learning y redes neuronales.

Entrenar nuestra máquina

En este paso se utilizará el set de datos de entrenamiento para ejecutar el modelo y se deberá ver una mejora incremental. Se deberá inicializar los ‘pesos’ del modelo aleatoriamente, es decir, los valores que afectan las relaciones entre las entradas y las salidas, las cuales se irán ajustando automáticamente por el algoritmo en la medida que se vaya entrenando.

Se deberá revisar los resultados obtenidos y corregir y volver a iterar.

Evaluación

Se deberá comprobar que el funcionamiento del modelo creado con el ser de **datos de validación**, el cual contendrá entradas que el modelo desconoce y con el cual se podrá verificar la precisión del modelo ya entrenado.

Si la exactitud es menor o igual a 50%, ese modelo no será útil ya que sería como lanzar una moneda al aire para tomar decisiones. Si se alcanza un 90% o más se podrá tener una buena confianza en los resultados que otorga el modelo.

Configuración de parámetros

Si durante la evaluación no se obtienen buenas predicciones es posible que se tengan problemas de **overfitting** (o **underfitting**) y se deberá retornar al paso de

entrenamiento realizando antes una nueva configuración de parámetros del modelo. Algunas opciones serían:

- Usar un modelo más complejo
- Usar un modelo más simple
- Darse cuenta de la necesidad de más datos y/o características
- Desarrollar una mejor comprensión del problema

Interpretación del modelo

En esta etapa se intenta explicar el funcionamiento del modelo o presentarlo de una manera comprensible para el ser humano. Algunos beneficios de interpretar los modelos son:

- Dar confiabilidad en los resultados
- Ayudar en el debugging
- Informar características más relevantes en el modelo
- Detectar la necesidad de recolectar nuevas muestras
- Mayor seguridad/robustez en el modelo

Implementar el modelo como servicio

La integración de un sistema de machine learning permite acelerar y administrar el ciclo de vida de los proyectos de aprendizaje automático. Para implementar un modelo de Machine Learning se pueden seguir, en general, los siguientes pasos:

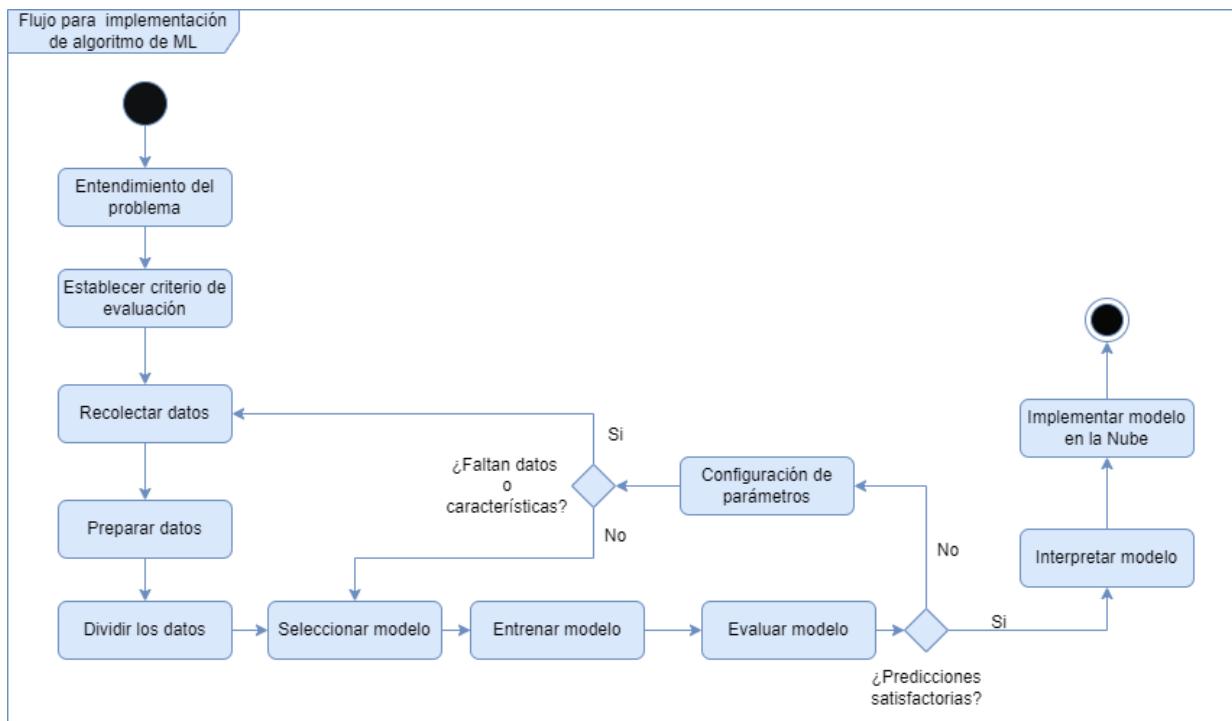
1. Registrar el modelo
2. Preparar un script
3. Preparar la configuración
4. Implementar el modelo localmente
5. Elegir un destino de proceso remoto
6. Implementar el modelo en la nube
7. Probar el servicio web resultante

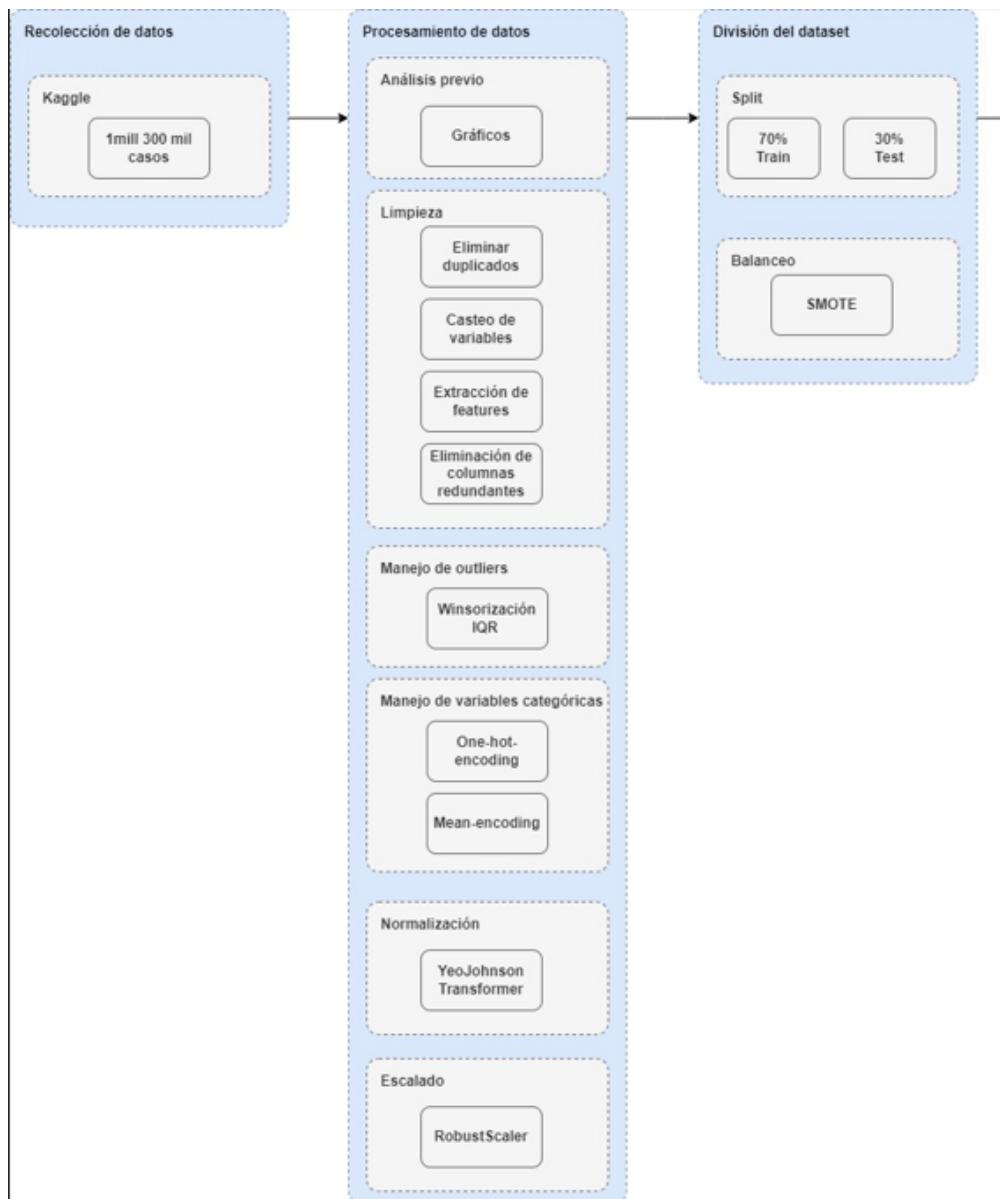
Diagramas

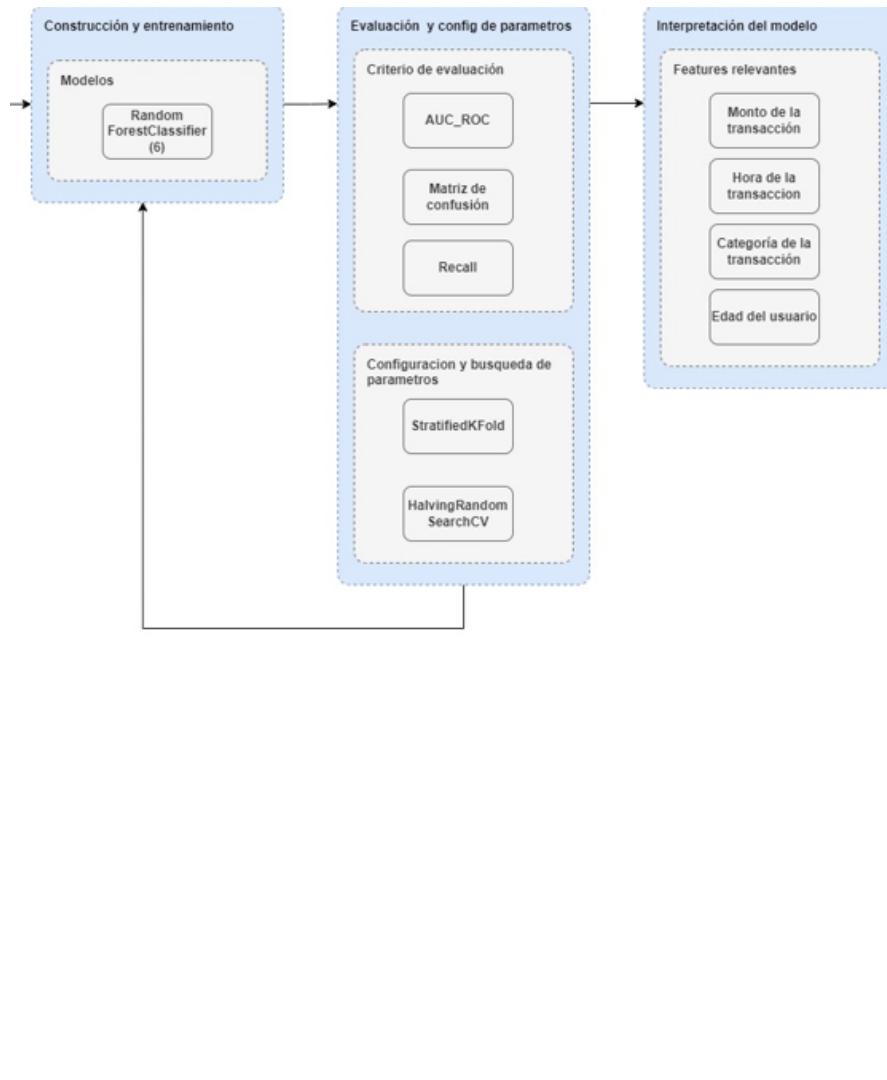
Los diagramas implican una herramienta esencial en el desarrollo de software puesto que pueden brindar ayuda a los desarrolladores a planificar, diseñar y documentar el proyecto, además de brindar facilidades de comunicación con el cliente y otros miembros del equipo. En el caso de los problemas de Machine Learning, algunos de las utilidades que puede tener son las siguientes:

- Visualización de datos:** Los diagramas permiten visualizar los datos y entender mejor la distribución de los mismos cuando se realiza la exploración del dataset. Esto es especialmente importante en los casos de clasificación donde se puede observar cómo se agrupan las distintas clases en el espacio de características.
- Explicación del modelo:** Pueden ayudar a entender cómo funciona el modelo. Por ejemplo, incorporando un diagrama de flujo o actividad o un diagrama de árbol de decisión para ilustrar cómo el modelo toma decisiones basadas en las características de los datos.
- Facilitación y claridad en la comunicación:** Pueden hacer que la presentación de la información sea más clara y fácil de entender facilitando la comunicación de las etapas y resultados a los demás.

En primer lugar, se incluirá un diagrama de actividades en el desarrollo y aplicación del algoritmo de Machine Learning. En segunda instancia, se irán incluyendo diagramas en la sección de **avances del proyecto** para una mejor explicación de las etapas y los resultados obtenidos.



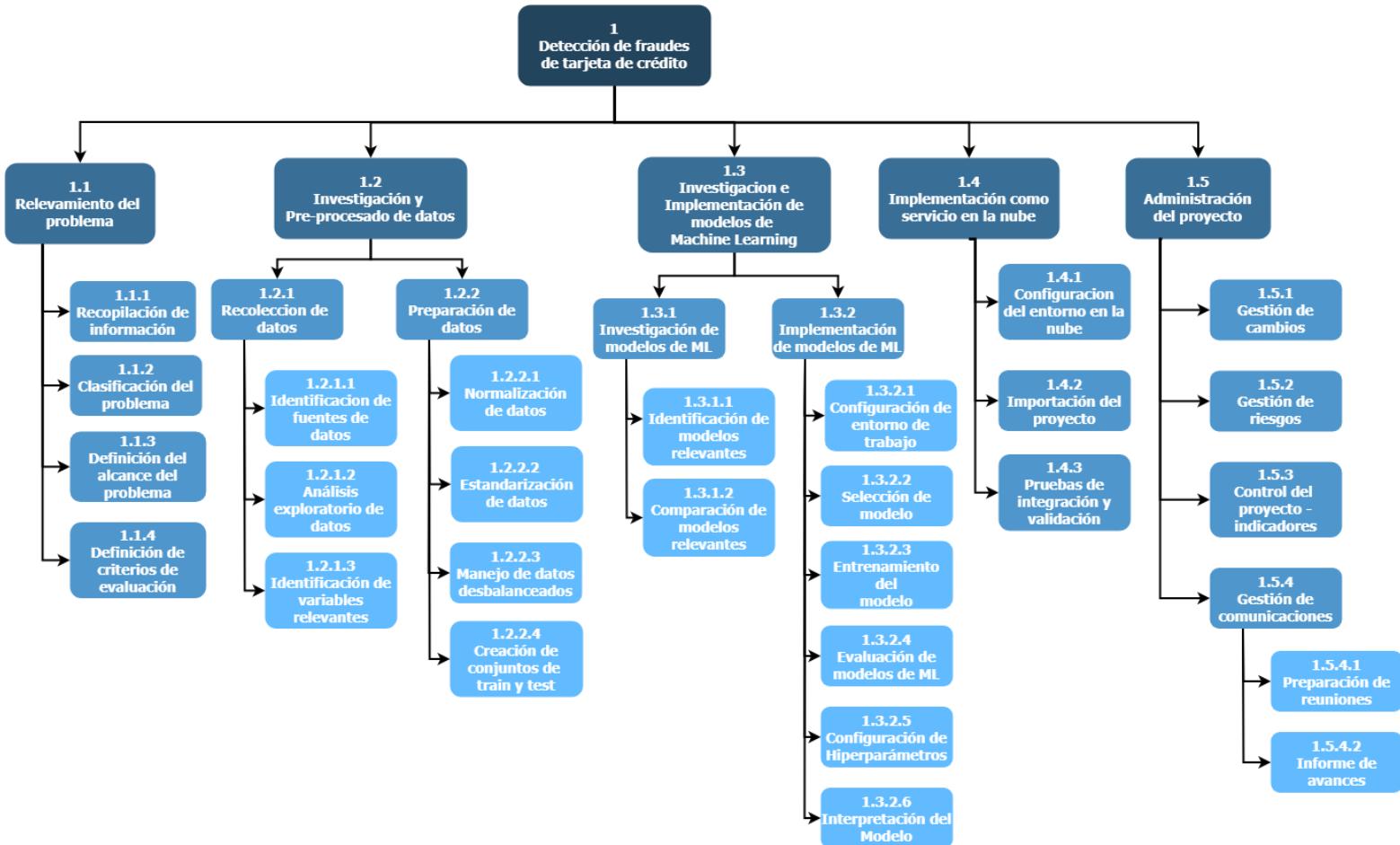




WBS

WBS es una técnica por la cuál el trabajo total a desarrollar en un proyecto se divide y subdivide con propósitos de administración y control. A continuación, se decidió implementar dicha técnica en busca de mitigar la complejidad del proyecto al permitir abstraer de manera superior las tareas que tengamos que realizar.

En este caso, se ha decidido tener un diseño mixto donde se oriente al producto, permitiendo la visibilidad sobre lo que se avanza en cada una de estas partes, y una orientación hacia el proyecto, al incluir partes como la administración, puesto que hay tareas que se harán a lo largo de todo el desarrollo general del proyecto.



Diccionario WBS

El diccionario de la WBS conlleva un documento adjunto en el cual se describen los elementos contenidos en el diseño de la WBS.

id	Nombre	Descripción	Tareas
1	Sistema de detección de fraudes de tarjetas de crédito	Proyecto de sistema de detección de fraudes con Machine Learning	Procesar conjuntos de datos, predecir fraudes
1.1	Relevamiento del problema	Producto encargado de la investigación y definición del problema del proyecto	Combinación de las sub-categorías que lo componen
1.1.1	Recopilación de información	Producto encargado de la selección de información de información	Investigación, análisis y validación

1.1.2	Clasificación del problema	Producto encargado de la clasificación del problema del sistema de detección de fraudes de tarjetas de crédito con machine learning	Análisis, clasificación y validación
1.1.3	Definición del alcance del problema	Producto encargado de la determinación del límite del problema dentro del proyecto	Análisis y definición
1.1.4	Definición de criterios de evaluación	Producto encargado de la determinación de criterios para evaluar la solución al problema de detección de fraudes en tarjetas de crédito	Investigación, análisis y definición
1.2	Investigación y pre-procesado de datos	Producto encargado de la recolección y preparación de datos	Integración de las sub-categorías que lo componen y validación
1.2.1	Recolección de datos	Producto encargado del recopilado de conjuntos de datos para el modelado de la solución	Integración de las sub-categorías que lo componen
1.2.1.1	Identificación de fuentes de datos	Producto encargado de la recopilación de fuentes de datos relevantes para el problema del proyecto	Investigación, análisis, definición y validación
1.2.1.2	Ánálisis exploratorio de datos	Producto encargado de la exploración y análisis de los datos recopilados	Análisis y validación
1.2.1.3	Identificación de variables relevantes	Producto encargado de la definición de variables relevantes en los conjuntos de datos recopilados	Análisis, definición y validación
1.2.2	Preparación de datos	Producto encargado del preparado de datos recopilados para implementación de modelo de machine learning	Integración de las sub-categorías que lo componen y pruebas integrales
1.2.2.1	Normalización de datos	Producto encargado de la normalización de datos	Implementación e integración. Pruebas unitarias y validación
1.2.2.2	Estandarización de datos	Producto encargado de la estandarización de datos	Implementación e integración. Pruebas unitarias y validación
1.2.2.3	Manejo de datos desbalanceados	Producto encargado del balanceamiento del conjunto	Implementación e integración. Pruebas unitarias

		de datos recopilado (casos “fraude” y “no fraude”)	y validación
1.2.2.4	Creación de conjuntos de train y test	Producto encargado de la división del conjunto de datos en conjunto de datos para entrenamiento y conjunto de datos para validación	Implementación e integración. Pruebas unitarias y validación
1.3	Investigación e implementación de modelos de Machine Learning	Producto encargado de la implementación del modelo de machine learning para la detección de fraudes de tarjetas de crédito	Integración de las sub-categorías que lo componen y pruebas integrales
1.3.1	Investigación de modelos de ML	Producto encargado de la recopilación de información sobre modelos de machine learning relevantes para el problema definido	Integración de las sub-categorías que lo componen y validación
1.3.1.1	Identificación de modelos relevantes	Producto encargado de la definición de modelos relevantes y útiles para la solución del problema definido	Análisis y definición
1.3.1.2	Comparación de modelos relevantes	Producto encargado de la comparación de modelos similares para la solución del problema definido	Análisis y comparación de modelos
1.3.2	Implementación de modelo ML	Producto encargado de la implementación del modelo de machine learning	Integración de las sub-categorías que lo componen y pruebas integrales
1.3.2.1	Configuración de entorno de trabajo	Producto encargado de la configuración de entornos, plataformas y herramientas útiles y relevantes para el desarrollo del proyecto	Análisis, diseño, implementación y documentación.
1.3.2.1	Selección de modelo	Producto encargado de la elección del modelo a utilizar para desarrollar la solución al problema definido	Análisis y definición
1.3.2.2	Entrenamiento del modelo	Producto encargado del entrenamiento del conjunto de datos de entrenamiento con un modelo definido de machine learning	Implementación e integración. Pruebas unitarias y validación
1.3.2.3	Evaluación de modelos de ML	Producto encargado de la evaluación del modelo con el	Implementación, pruebas unitarias y validación

		conjunto de datos de test	
1.3.2.4	Configuración de Hiper Parámetros	Producto encargado de la configuración de parámetros del modelo seleccionado	Implementación e integración. Pruebas unitarias y validación
1.3.2.5	Interpretación del modelo	Producto encargado del análisis y validación del modelo seleccionado e implementado	Análisis y validación del modelo definido
1.4	Implementación como servicio en la nube	Producto encargado de la implementación del modelo de machine learning como servicio en la nube	Integración de las sub-categorías que lo componen y pruebas integrales
1.4.1	Configuración del entorno en la nube	Producto encargado de la configuración del entorno en la nube donde se implementará el modelo de machine learning	Implementación e integración. Pruebas unitarias y validación
1.4.2	Implementación del proyecto en la nube	Producto encargado de la implementación del modelo de machine learning en la nube configurada	Implementación e integración. Pruebas unitarias y validación
1.4.3	Pruebas de integración y validación	Producto encargado de la exploración y testing del modelo de machine learning como servicio	Implementación, pruebas unitarias y validación
1.5	Administración del proyecto	Producto encargado del modelado del proyecto.	Integración de las subcategorías que lo componen y pruebas integrales.
1.5.1	Gestión de cambios	Proyecto encargado de gestionar los cambios, identificándolos, documentándolos y aprobándolos o rechazándolos	Análisis, investigación, diseño y documentación.
1.5.2	Gestión de riesgos	Proyecto encargado de gestionar los riesgos, identificándolos, documentándolos y moderándolos	Análisis, investigación, diseño y documentación.
1.5.3	Control del proyecto - indicadores	Proyecto encargado de controlar la correcta gestión del proyecto en base a los distintos indicadores.	Análisis, investigación, diseño y documentación.

1.5.4	Gestión de comunicaciones	Proyecto que incluye los procesos requeridos para garantizar que la generación, la recopilación, la distribución, el almacenamiento, la recuperación y la disposición final de la información del proyecto sean adecuados y oportunos.	Análisis, diseño, implementación y documentación.
1.5.4.1	Preparación de reuniones	Proyecto encargado de la administración de reuniones.	Análisis, diseño, implementación y documentación.
1.5.4.2	Informe de avances	Proyecto encargado de la creación de informes cuyo objetivo es informar sobre los avances realizados en el proyecto.	Análisis, diseño, implementación y documentación.

Avances en el proyecto

Primera parte

Luego de haber configurado el entorno de anaconda mediante la instalación de las librerías que utilizaremos en el entorno, las cuales se podrán agregar más a futuro si es necesario, procedimos a ejecutar **Jupyter**. En dicho entorno, se creó una carpeta donde se creará y trabajará el archivo con el modelo.

Una de las primeras tareas fue la elección de un dataset que contuviera los nombres de las variables en la plataforma de Kaggle. En nuestro caso, se ha elegido un dataset llamado *Credit Card Transactions Fraud Detection Dataset*⁵ la cual es un dataset creado mediante un programa llamado Sparkov. Este conjunto contiene datos de transacciones de tarjetas de crédito simuladas que poseen transacciones legítimas y fraudulentas desde una fecha del 1 de enero de 2019 hasta el 31 de diciembre de 2020. A continuación se realizará una breve descripción con las columnas:

⁵ <https://www.kaggle.com/datasets/kartik2112/fraud-detection>

Nombre	Descripción
index	Identificador para cada fila
trans_date_trans_time	DateTime de transacción
cc_num	Número de tarjeta de crédito del cliente
merchant	Nombre del comerciante
category	Categoría del comerciante
amt	Cantidad de la transacción
first	Primer nombre del titular de la tarjeta de crédito
last	Apellido del titular de la tarjeta de crédito
gender	Género del titular de la tarjeta de crédito
street	Dirección del titular de la tarjeta de crédito
city	Ciudad del titular de la tarjeta de crédito
state	Estado del titular de la tarjeta de crédito
zip	Zip del titular de la tarjeta de crédito
lat	Ubicación de latitud del titular de la tarjeta de crédito
long	Ubicación de la longitud del titular de la tarjeta de crédito
city_pop	Población de la ciudad del titular de la tarjeta de crédito
job	Trabajo del titular de la tarjeta de crédito
dob	Fecha de nacimiento del titular de la tarjeta de crédito
trans_num	Número de transacción
unix_time	UNIX time de la transacción
merch_lat	Ubicación de latitud del comerciante
merch_long	Ubicación de la longitud del comerciante
is_fraud	Flag de fraude

En el caso del dataset elegido, ya se presentaba dividido para test y train pero sin haber sido procesados. Se decidió solo elegir el csv de entrenamiento para procesar debido a que los dos archivos, en conjunto, eran casi medio gigabyte. A continuación, se describe el proceso realizado hasta el momento para el análisis del dataset y la preparación lograda hasta el momento.

Recolección, preparación y división de datos

En primer lugar, se importan las librerías que se utilizarán hasta el momento.

```
#En primer Lugar, importamos Las Librerías
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
plt.style.use('ggplot')
import warnings
warnings.filterwarnings('ignore')
```

Una breve descripción de las mismas sería la siguiente:

- *%matplotlib inline* es una ‘magic function’ que permite mostrar los gráficos en la celda siguiente facilitando la visualización.
- *pandas*, permite la manipulación y análisis de datos.
- *numpy*, permite cálculos numéricos con arrays y matrices.
- *matplotlib.pyplot*, para la visualización y gráficos.
- *seaborn*, para visualización de datos estadísticos.
- *time*, para la manipulación de métodos que requieran el manejo del tiempo.

Posteriormente, se accedió al dataset y se comenzó el análisis de su constitución, donde se encontró que hay 1.296.675 filas y 23 columnas. Luego, se verificó el tipo de datos que contiene cada columna.

```
data.columns (total 29 columns)
#   Column           Non-Null Count   Dtype  
--- 
0   Unnamed: 0        1296675 non-null  int64  
1   trans_date_trans_time 1296675 non-null  object  
2   cc_num            1296675 non-null  int64  
3   merchant          1296675 non-null  object  
4   category          1296675 non-null  object  
5   amt               1296675 non-null  float64 
6   first              1296675 non-null  object  
7   last               1296675 non-null  object  
8   gender             1296675 non-null  object  
9   street             1296675 non-null  object  
10  city               1296675 non-null  object  
11  state              1296675 non-null  object  
12  zip                1296675 non-null  int64  
13  lat                1296675 non-null  float64 
14  long               1296675 non-null  float64 
15  city_pop           1296675 non-null  int64  
16  job                1296675 non-null  object  
17  dob                1296675 non-null  object  
18  trans_num          1296675 non-null  object  
19  unix_time          1296675 non-null  int64  
20  merch_lat          1296675 non-null  float64 
21  merch_long         1296675 non-null  float64 
22  is_fraud           1296675 non-null  int64  
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
```

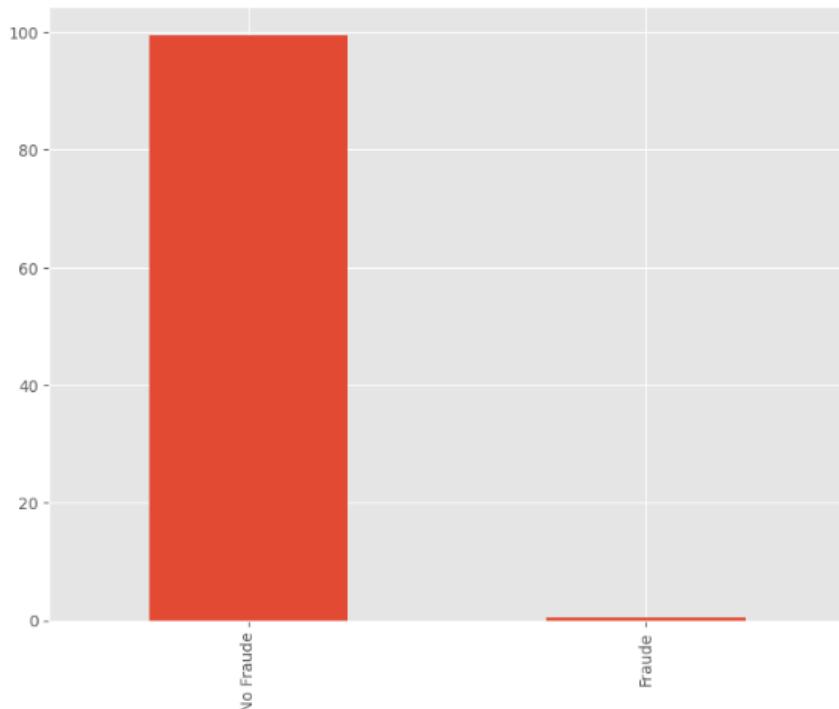
Posteriormente, se procedió a verificar si hay valores nulos, ya que estos hay que eliminarlos. Se observó que no hay datos nulos.

In [6]: `data.isnull().sum()`

```
Out[6]: Unnamed: 0      0
trans_date_trans_time  0
cc_num                 0
merchant               0
category               0
amt                     0
first                   0
last                    0
gender                  0
street                  0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                 0
unix_time                 0
merch_lat                 0
merch_long                 0
is_fraud                  0
dtype: int64
```

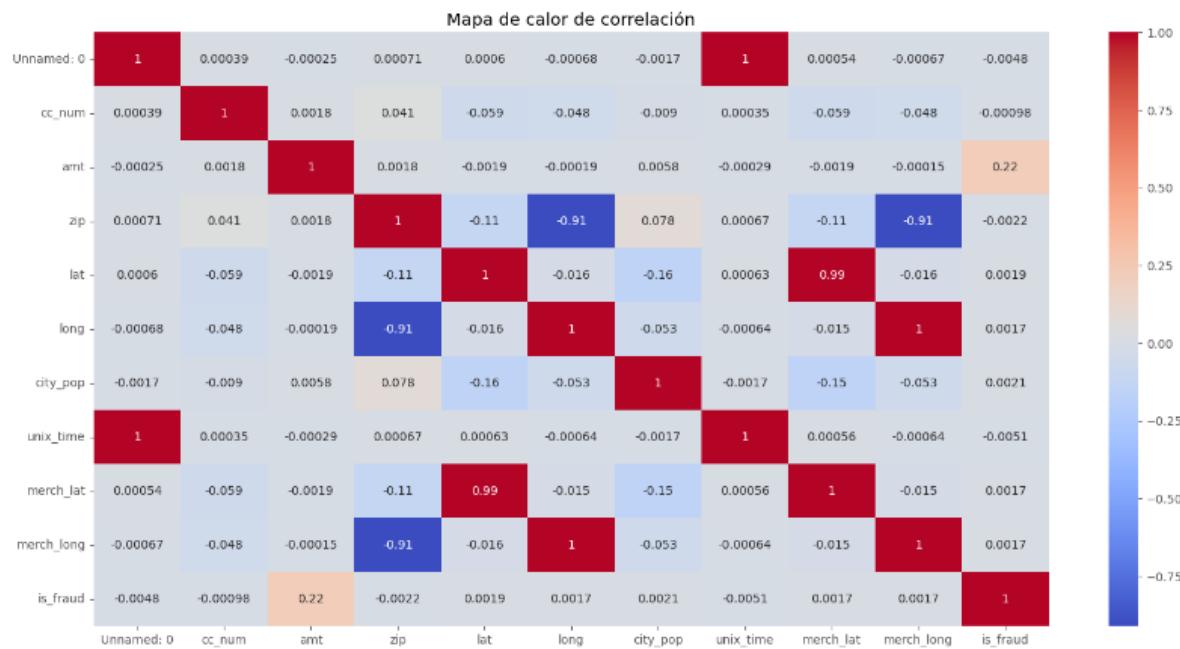
Se decidió realizar una primera aproximación para evaluar la cantidad de valores, y su porcentaje, que son legítimos y fraudulentos.

	Cantidad	Distribucion en porcentaje
No Fraude	1289169	99.421135
Fraude	7508	0.578865



Como se puede estudiar, el dataset se encuentra muy desbalanceado, presentando un 99,42% de transacciones legítimas y sólo aproximadamente un 0.57% de transacciones fraudulentas.

Otro gráfico que utilizamos para visualizar el dataset fue un **mapa de calor de correlación**. Dicho mapa es una herramienta de visualización que se utiliza para analizar las relaciones entre las diferentes variables en un conjunto de datos. Cada celda en el mapa representa el coeficiente de correlación de Pearson entre dos variables. Dicho coeficiente es una medida estadística que evalúa la relación lineal entre dos variables continuas. Esta varía entre -1 y 1, donde -1 indica una correlación negativa perfecta(inversa), 0 indica ausencia de correlación y 1 indica una correlación positiva perfecta. En resumen, una correlación cercana a 1 indica que las variables están altamente correlacionadas lo que significa que a medida que aumenta una variable, la otra también aumenta en proporción.



Preparación de datos

Primeramente, se dispuso a ejecutar una limpieza inicial donde se eliminarían elementos duplicados, se castearían las variables que sean necesarias, se revisará la cardinalidad de las columnas, es decir, la cantidad de valores posibles únicos de cada columna, y eliminar las columnas que sean redundantes luego de haber realizado el análisis anterior.

Durante el proceso se observó que no había filas duplicadas por lo que se procedió a castear las variables de **trans_date_trans_time** y **dob** puesto que pertenecían al tipo *object* y era necesario que sean de tipo *time*, esto es debido a que se utilizarán para extraer información de ellas.

Luego, se propuso chequear la cardinalidad. La **cardinalidad** de las columnas de categorías consiste en verificar la cantidad de valores únicos en una columna. Esto nos puede proporcionar información y de cómo se pueden tratar para el análisis. Cuando una columna tiene alta cardinalidad, es decir, muchos valores únicos, esto puede indicar que la columna no está bien estructurada o que hay demasiados valores posibles para que sean útiles para el análisis. Por otro lado, si la cardinalidad es baja, puede indicar que los datos están muy agrupados y que se pierde información.

```
out[80]: gender      2
category     14
state       51
first      352
last       481
job        494
merchant   693
city       894
street     983
trans_num  1296675
dtype: int64
```

Se procedió a eliminar columnas que sean repetidas, redundantes y consideramos que no nos servirán para el entrenamiento del modelo. Entre ellas están **Unnamed: 0**, **street**, **zip**, **first**, **last**, **trans_num**, **job** y **merchant**. Luego, nuestro dataset conserva 15 columnas.

	trans_date_trans_time	cc_num	category	amt	gender	city	state	lat	long	city_pop	dob	unix_time	merch_la
0	2019-01-01 00:00:18	2703186189652095	misc_net	4.97	f	moravian falls	nc	36.0788	-81.1781	3495	1988-03-09	1325376018	36.011293
1	2019-01-01 00:00:44	630423337322	grocery_pos	107.23	f	orient	wa	48.8878	-118.2105	149	1978-06-21	1325376044	49.159047
2	2019-01-01 00:00:51	38859492057061	entertainment	220.11	m	malad city	id	42.1808	-112.2620	4154	1982-01-19	1325376051	43.150704
3	2019-01-01 00:01:16	3534093764340240	gas_transport	45.00	m	boulder	mt	46.2306	-112.1138	1939	1987-01-12	1325376076	47.034331
4	2019-01-01 00:03:06	375534208663984	misc_pos	41.96	m	doe hill	va	38.4207	-79.4629	99	1986-03-28	1325376188	38.674996
...
1296670	2020-06-21 12:12:08	30263540414123	entertainment	15.56	m	hatch	ut	37.7175	-112.4777	258	1961-11-24	1371816728	36.841266
1296671	2020-06-21 12:12:19	6011149206456097	food_dining	51.70	m	tuscarora	md	39.2687	-77.5101	100	1979-12-11	1371816739	38.906881
1296672	2020-06-21 12:12:32	3514865930894895	food_dining	105.93	m	high rolls mountain park	nm	32.9396	-105.8189	899	1987-08-30	1371816752	33.619513
1296673	2020-06-21 12:13:36	2720012583106919	food_dining	74.90	m	manderson	sd	43.3526	-102.5411	1126	1980-08-18	1371816816	42.788940
1296674	2020-06-21 12:13:37	4292902571056973207	food_dining	4.30	m	sula	mt	45.8433	-113.8748	218	1995-08-16	1371816817	46.565983

296675 rows × 15 columns

Extracción de features

La extracción de *features*, o características, consiste en seleccionar, reducir o transformar variables del dataset. La idea principal es extraer la información más relevante y significativa para utilizarla en análisis o para el aprendizaje del modelo. En nuestro caso, se decidió desglosar las variables **trans_date_trans_time**, **unix_time**, **dob**, **merch_lat** y **merch_long**, y **lat** y **long**.

Variable trans_date_trans_time

La variable **trans_date_trans_time** se separó en **trans_hora**, **trans_mes** y **trans_dia**.

Variable unix_time

Mediante la variable `unix_time`, se calculó el UNIX time de la transacción previa correspondiente a la misma tarjeta de crédito. Estos valores se almacenaron en una columna llamada `unix_time_previo_trans`. Luego, una vez obtenidos dichos datos, se calculó la diferencia de unix time entre la transacción actual y la previa de cada tarjeta, posteriormente, se almacenaron dichos datos en la columna `delay_entre_trans`.

Variable dob

Mediante la variable `dob` se calculó la edad de los usuarios y se almacenó los datos en la columna `edad_usuario` en años.

edad_usuario	
1017	32.894610
2724	32.896889
2726	32.896894
2882	32.897333
2907	32.897395
...	...
1294934	64.492818
1295369	64.493229
1295587	64.493470
1296206	64.494072
1296427	64.494290

1296675 rows × 1 columns

Variable lat, long, merch_lat y merch_long

Con dichas variables, se propuso calcular los siguientes puntos:

1. La distancia de latitud entre la localización del comprador y el comerciante.
2. La distancia de longitud entre la localización del comprador y del comerciante.
3. La distancia de latitud entre la localización del comerciante actual y el anterior.
4. La distancia de longitud entre la localización del comerciante actual y el anterior.

Para realizar lo pedido, se calcularon las diferencias correspondientes.

- Para (1) se calculó la diferencia y se almacenó los datos en la columna con variable `dif_lat_comprador_merch`.
- Para (2) se calculó la diferencia y se almacenó los datos en la columna con la variable `dif_long_comprador_merch`.
- Para (3) y (4) se procedió a crear dos columnas `lat_previo_merch` y `long_previo_merch` para posteriormente calcular las diferencias entre la latitud actual y la previa, y la longitud actual y la previa, almacenando los datos

correspondientes en las columnas con variables **dif_lat_prev_merch** y **dif_long_prev_merch**.

	dif_lat_prev_merch	dif_long_prev_merch
1017	0.000000	0.000000
2724	1.955945	0.697732
2726	0.942569	0.113392

Eliminación de features o columnas redundantes

En consecuencia al proceso anterior, habían quedado columnas que ya no aportaban nueva información, por lo tanto se procedió a su eliminación. Entre ellas se encontraban:

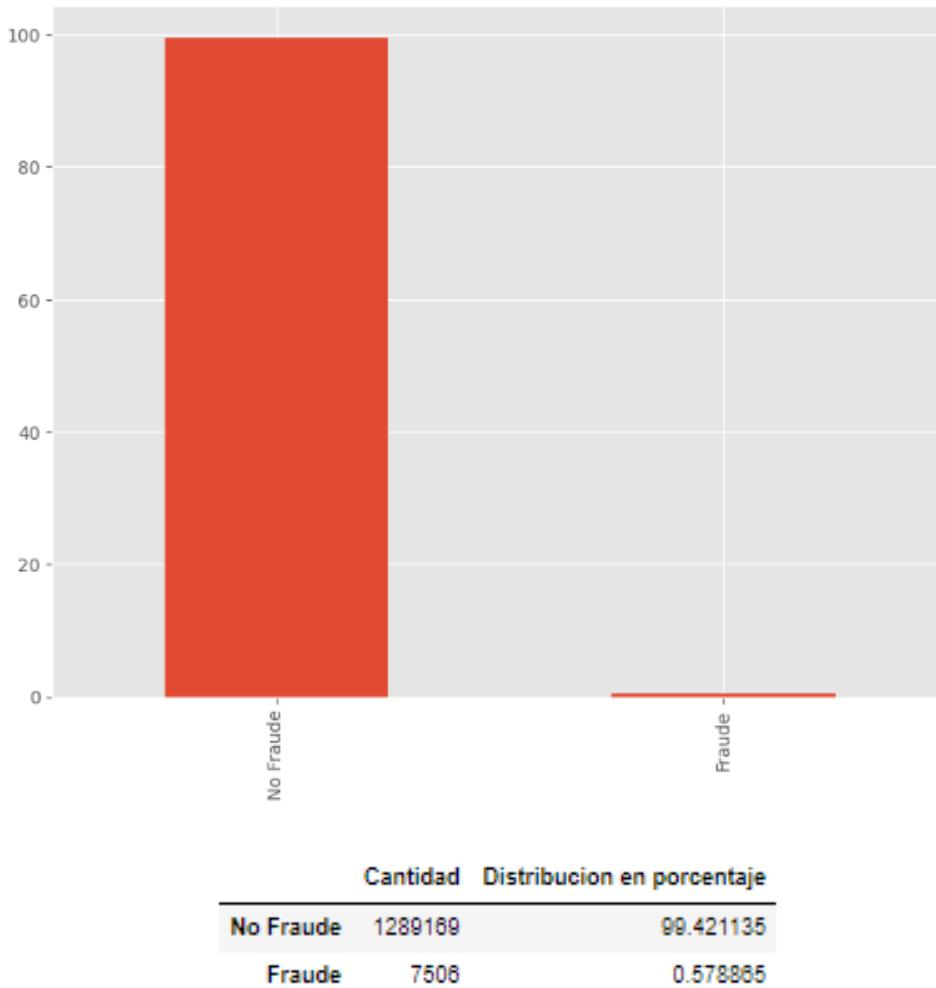
- trans_date_trans_time
- cc_num
- unix_time
- dob
- unix_time_previo_trans
- lat
- long
- merch_lat
- merch_long
- lat_previo_merch
- long_previo_merch

Segundo análisis

Luego del procedimiento anterior, decidimos realizar una segunda exploración de los datos mediante el uso de tablas y gráficos.

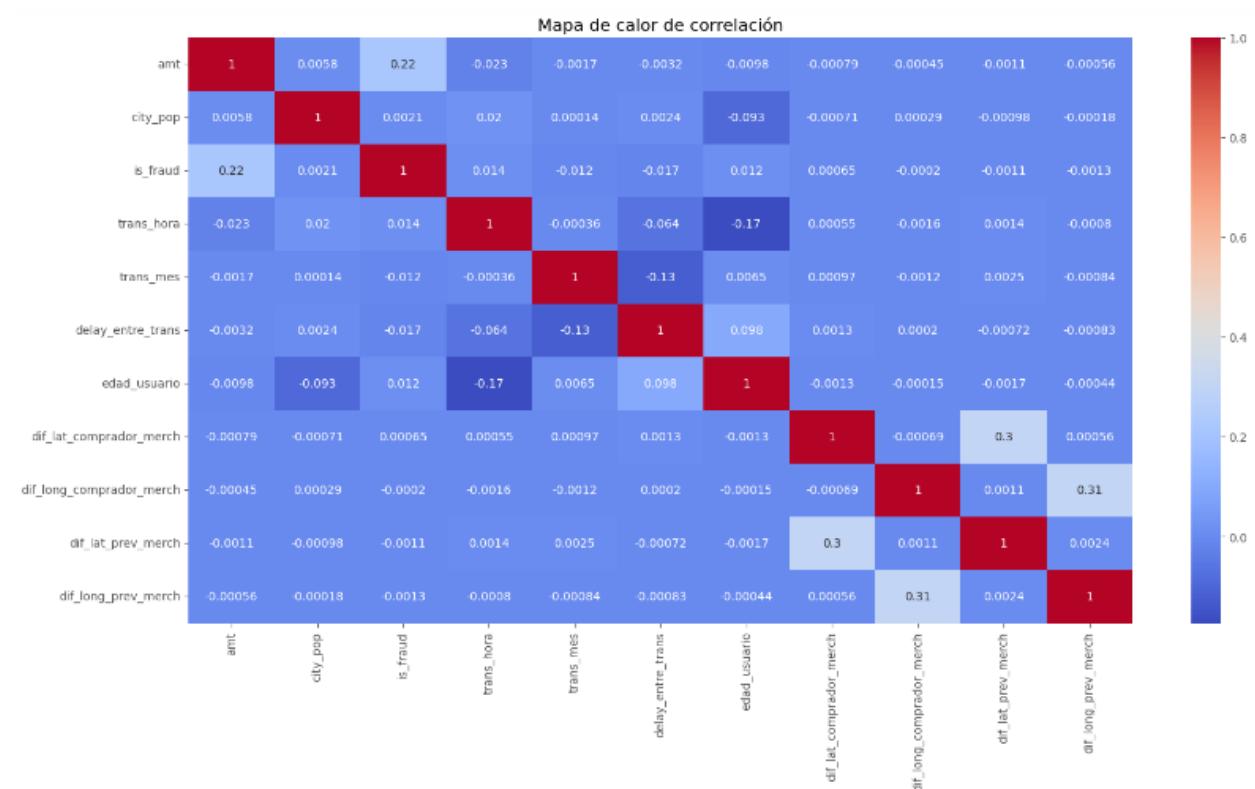
En este caso, se realizó una copia del dataset ya procesado y se crearon dos variables ‘normal’ y ‘fraudes’, donde se almacenaron las partes del dataset donde en una se encontraban los datos de transacciones legítimas y otro con los datos de transacciones fraudulentas.

Distribución de fraudes y no fraudes



Como se puede observar, no se presentan fluctuaciones en cuanto a la cantidad y porcentaje de transacciones legítimas y fraudulentas luego del pequeño procesamiento realizado.

Mapa de calor de correlación



Análisis por distribución respecto a la cantidad y el delay

Se observó que el promedio del monto de dinero de las transacciones falsas es de \$531,32 mientras que en las transacciones legítimas es del \$67.66.

Por otro lado, podemos decir que las transacciones fraudulentas se concentran en los \$390,56.

	count	mean	std	min	25%	50%	75%	max
is_fraud								
0	1289169.0	67.667110	154.007971	1.00	9.6100	47.280	82.540	28948.90
1	7506.0	531.320092	390.560070	1.06	245.6625	396.505	900.875	1376.04

Respecto al delay entre la transacción previa y la actual correspondiente a cada tarjeta, se encontró que las transacciones fraudulentas, más aún las que son sucesivas, son más rápidas que las legítimas.

	count	mean	std	min	25%	50%	75%	max
is_fraud								
0	1289169.0	542.631735	789.894813	0.0	101.0	277.0	673.00	22357.0
1	7506.0	365.552225	612.956650	0.0	25.0	84.0	509.75	14417.0

Análisis por distribución respecto a la distancia de latitud y longitud entre el cliente y el comerciante

En ambos análisis, no se observaron diferencias relevantes.

Respecto a la latitud:

	count	mean	std	min	25%	50%	75%	max
is_fraud								
0	1289169.0	0.500249	0.288584	0.000000	0.250397	0.500516	0.750061	0.999999
1	7506.0	0.502739	0.286166	0.000225	0.253632	0.507158	0.746795	0.999968

Respecto a la longitud:

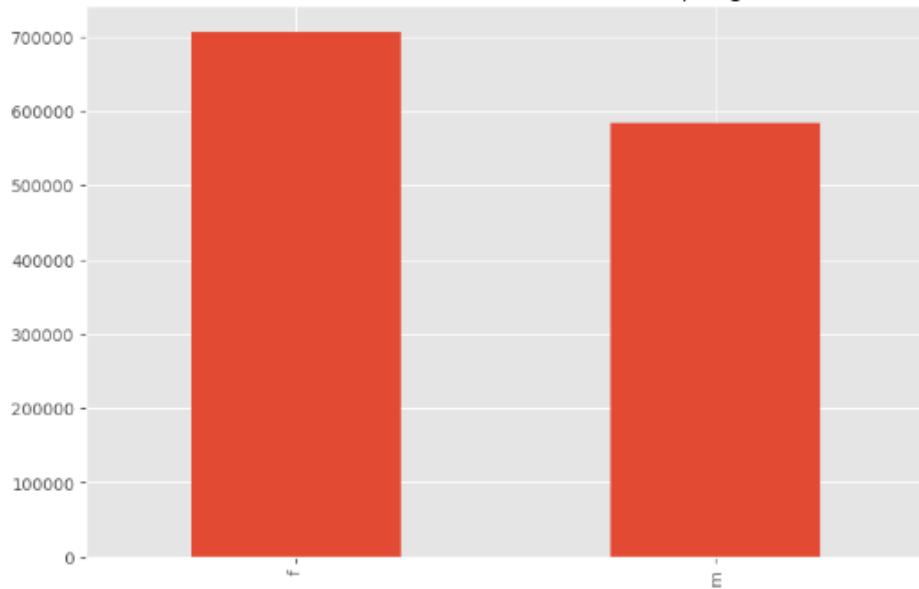
	count	mean	std	min	25%	50%	75%	max
is_fraud								
0	1289169.0	0.500341	0.288866	0.000000	0.249968	0.500321	0.750566	0.999997
1	7506.0	0.499577	0.289625	0.000438	0.248196	0.496840	0.745902	0.999914

Gráfica por género

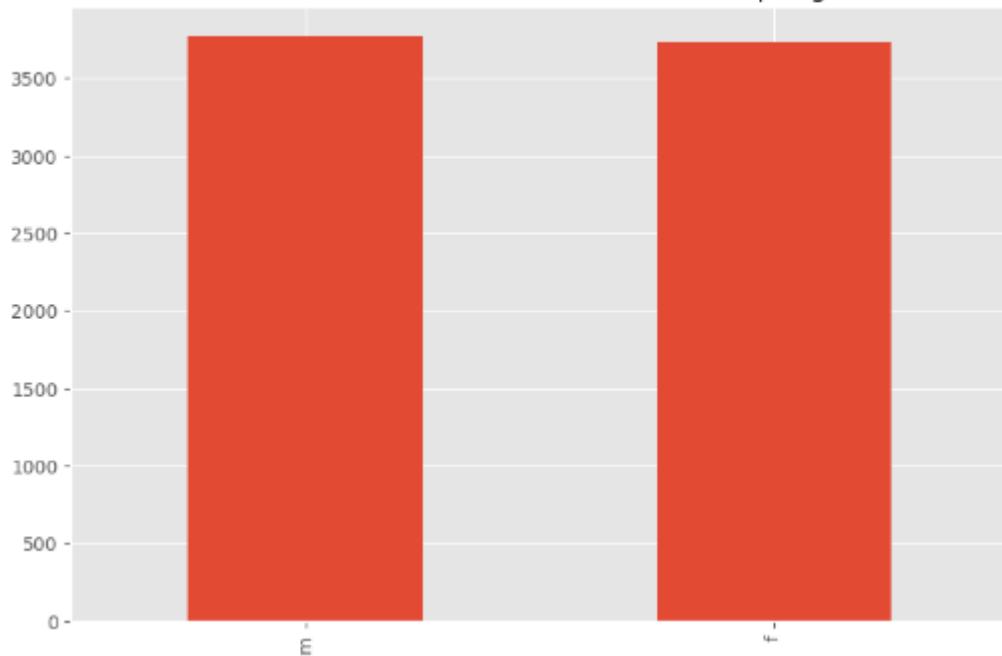
En el caso de la distribución por género, no se encontraron diferencias significativas.

	normal	fraude
f	54.773889	49.760192
m	45.226111	50.239808

Distribucion de transacciones normales por género



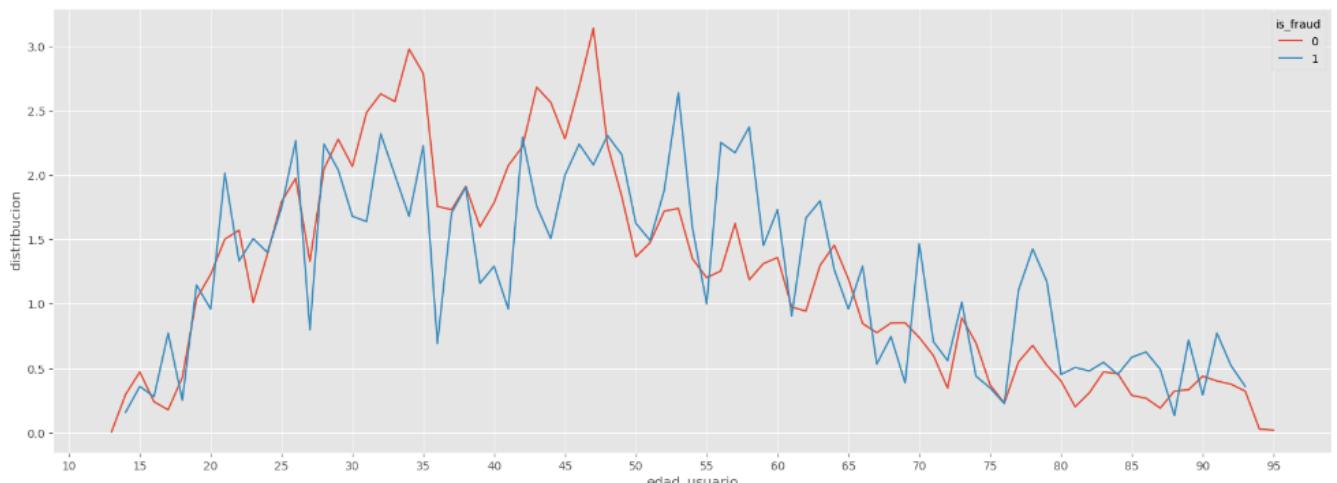
Distribucion de transacciones fraudulentas por género



Gráfica por edad

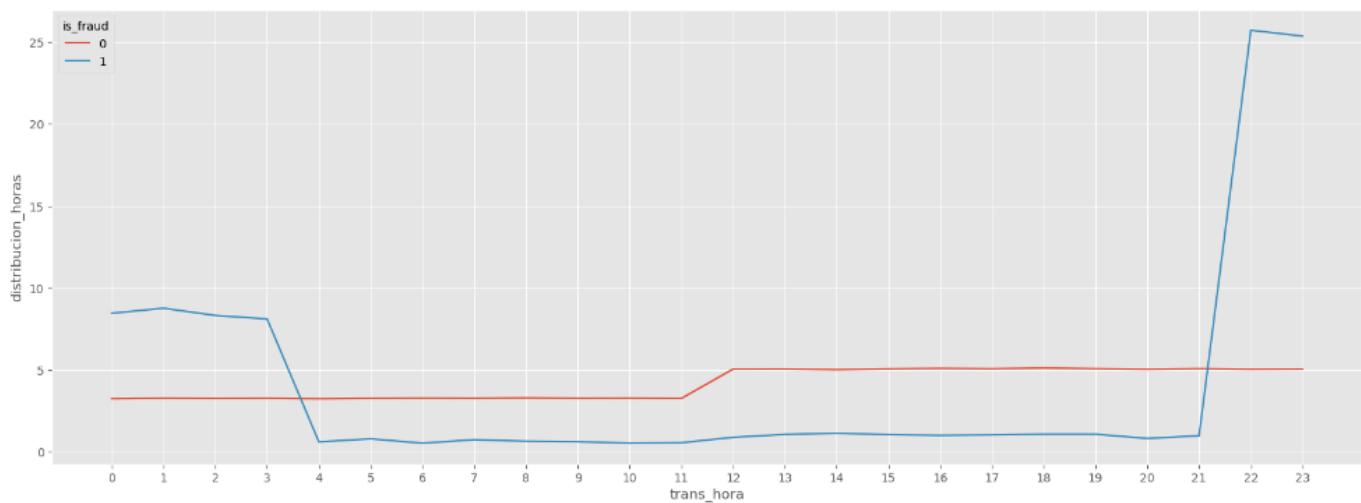
	count	mean	std	min	25%	50%	75%	max
is_fraud								
0	1289169.0	45.511960	17.398818	13.0	32.0	43.0	57.0	95.0
1	7506.0	48.321609	18.864543	14.0	33.0	47.0	60.0	93.0

En el caso de la edad, se observó que las transacciones, en su mayoría, son de personas entre 30 y 50 años. Por otro lado, las transacciones fraudulentas están concentradas entre los 45 y 60 años.



Análisis por distribución respecto a la hora

En este caso, se observó el aumento de las transacciones fraudulentas entre las 21 hs y las 4 hs.



Continuación preparación de datos

En este caso, se realizaron las siguientes actividades:

- **Manejo de outliers:** Manejo de valores atípicos mediante winsorización.
- **Codificación de variables categóricas:** Codificación mediante one-hot y codificación target.
- **Transformación de las variables**
- **Escalado de features:** Escalar valores numéricos para que estén en un rango similar.

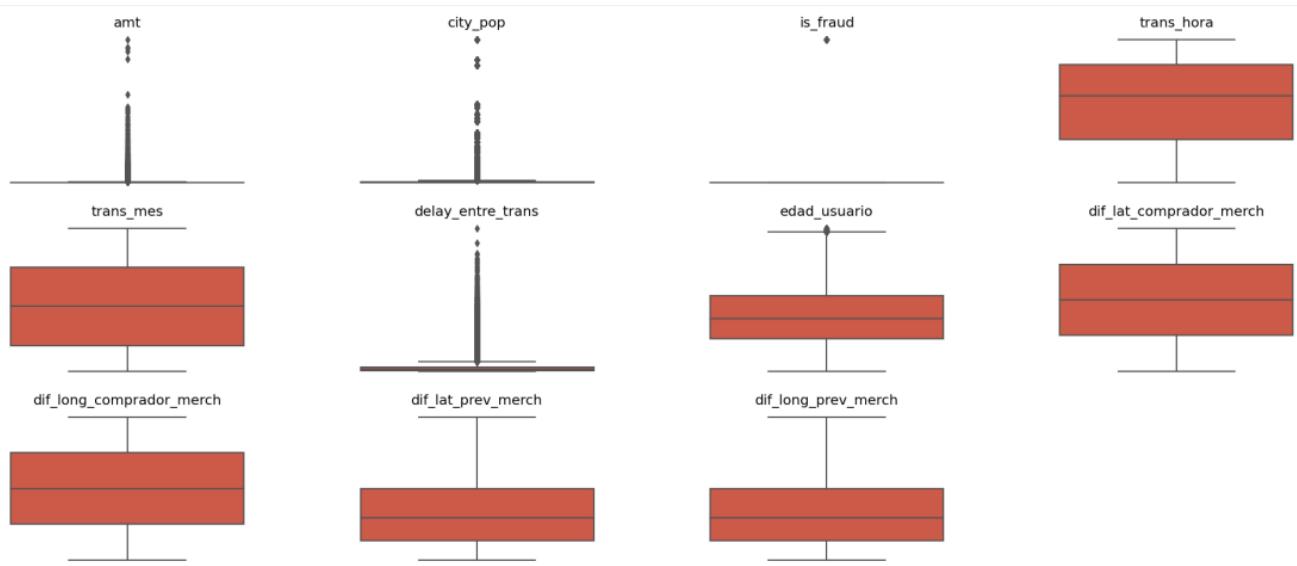
Manejo de valores atípicos (outliers)

Los valores atípicos son valores que se encuentran significativamente fuera del rango esperado de los demás datos en un conjunto de datos. Es necesario tratar dichos valores puesto que puede distorsionar los resultados y afectar la precisión de los modelos o el análisis. Algunas opciones puede ser:

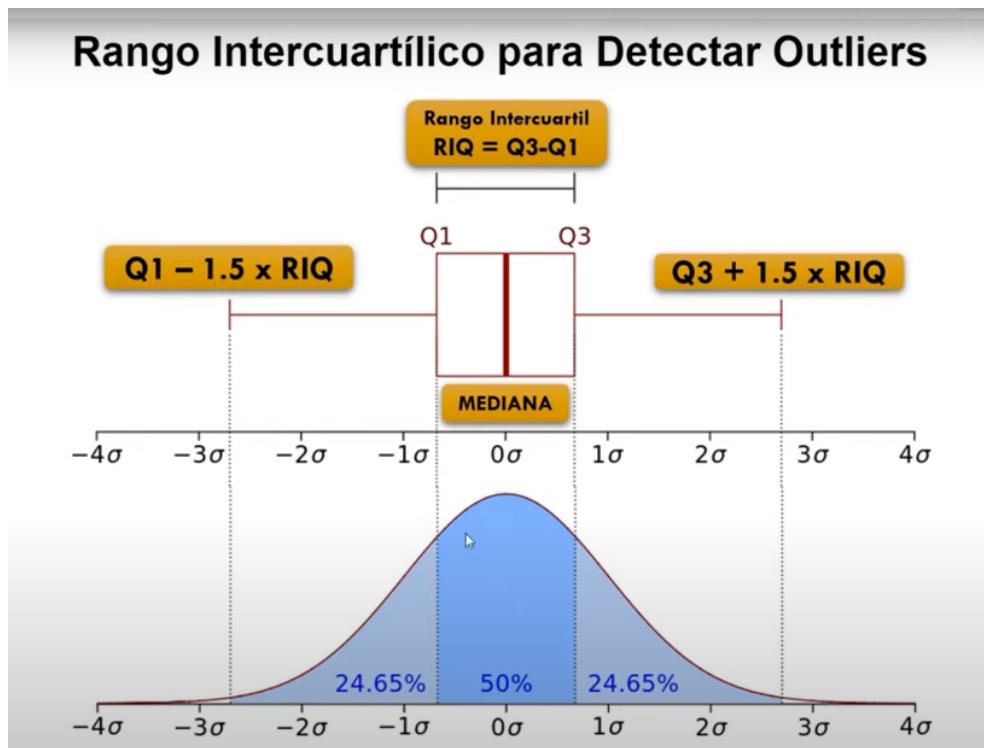
1. Eliminar los valores atípicos: Es la más sencilla y directa pero puede eliminar valores que podrían resultar en pérdida de información valiosa.
2. Transformación de los datos: En estos casos se puede aplicar técnicas para transformar los datos y reducir la cantidad de valores.

En nuestro caso, se buscará la visualización de los outliers mediante diagramas de caja para luego utilizar winsorización. Dicha técnica consiste en reemplazar los valores atípicos por valores más extremos dentro del rango esperado de valores. Esto es útil cuando queremos conservar la información contenida en ella.

Se utilizó la librería de python **feature_engine** el cual provee herramientas para codificación de variables categóricas y escalado de features. Luego mediante los diagramas de caja se pudo observar que las variables '**amt**', '**city_pop**', '**delay_entre_trans**' y '**edad_usuario**' presentaban outliers que había que manejar.



Para la winsorización se utilizó el método IQR que consiste en establecer los valores dentro del rango intercuartil. Este se define como la diferencia entre el tercer cuartil (q3) y el primer cuartil (q1) de los datos, es decir, $RIQ = q3 - q1$. Al calcularse el rango intercuartil, se establecen los límites superior e inferior del rango de Winsorización.



```

from feature_engine.outliers import Winsorizer

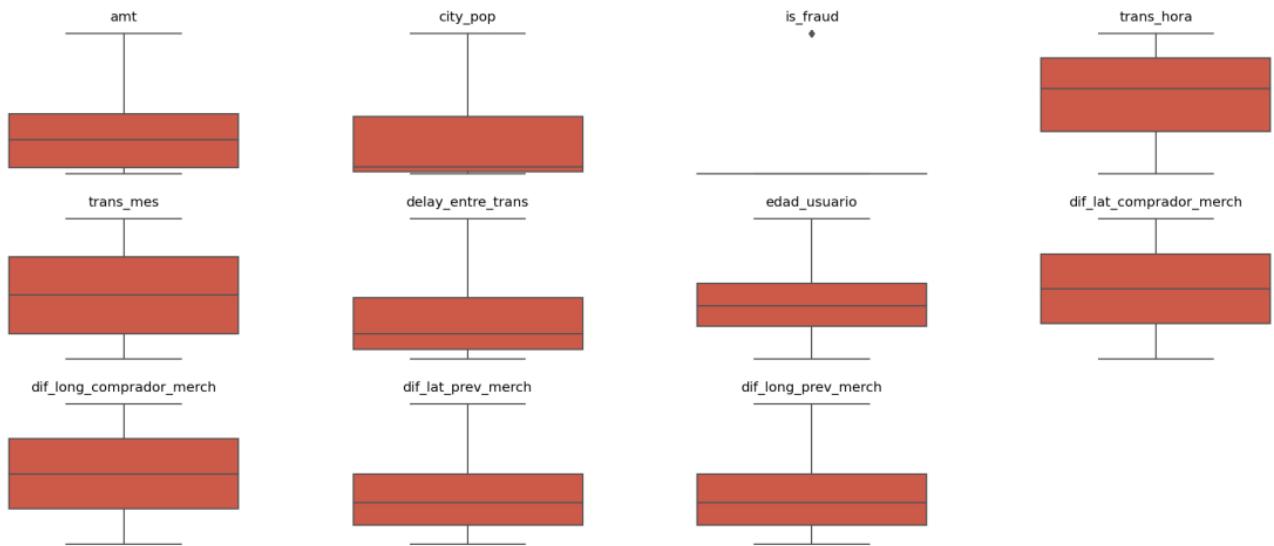
variables_a_manejar=['amt','city_pop','delay_entre_trans','edad_usuario']
capper_iqr=Winsorizer(capping_method='iqr', tail='both', fold=1.5, variables=variables_a_manejar)
#Ahora lo aplicamos a nuestro dataset
capper_iqr.fit(data)

Winsorizer
Winsorizer(capping_method='iqr', fold=1.5, tail='both',
            variables=['amt', 'city_pop', 'delay_entre_trans', 'edad_usuario'])

#Ahora aplicamos a los valores a los datos
data=capper_iqr.transform(data)

```

Luego de la aplicación, se pudo observar, nuevamente mediante diagramas de caja, como las variables presentaban una buena distribución.



Codificación de variables categóricas

El objetivo es convertir las variables no numéricas (categóricas) en variables numéricas para poder usarlas durante la construcción del modelo. Es importante manejarlas puesto que hay muchos modelos que no soportan variables categóricas.

Dentro de los tipos de codificación posibles utilizamos las técnicas de **one-hot-encoding** y **mean-encoding(target-encoding)**. Esto se decidió al evaluar la cardinalidad de las variables categóricas.

- **One-hot-encoding:** Esta técnica, básicamente, convierte cada valor posible de una variable categórica en una nueva columna binaria (0 o 1). Esta técnica funciona bien cuando la variable categórica tiene pocos valores posibles y no hay una relación natural de orden entre ellos.

- **Mean Encoding:** Sustituye cada valor posible de una variable categórica por el promedio (o la media) de la variable objetivo (la variable que queremos predecir) para ese valor. Esta técnica funciona mejor cuando la variable categórica tiene muchos valores posibles.

```
columnas_categoricas=data.select_dtypes(exclude=np.number).columns
data[columnas_categoricas].nunique()

category      14
gender         2
state        51
trans_dia      7
dtype: int64
```

Se decidió que el género se codificara con one-hot-encoding mientras que ‘state’, ‘trans_dia’, ‘category’ y ‘trans_dia’ con mean-encoding por el interés que este podría tener con la variable objetivo.

Posteriormente, se verificó que todas las variables sean numéricas.

```
: data.dtypes

category          float64
amt               float64
state              float64
city_pop          float64
is_fraud          int64
trans_hora         int64
trans_mes          int64
trans_dia          float64
delay_entre_trans float64
edad_usuario       float64
dif_lat_comprador_merch float64
dif_long_comprador_merch float64
dif_lat_prev_merch   float64
dif_long_prev_merch   float64
gender_f           int32
dtype: object
```

Normalización y escalado de variables

En esta etapa, se busca normalizar o estandarizar las variables numéricas a una escala común. La idea principal consiste en que estén en un rango similar puesto que hay modelos que pueden ser sensibles a la escala de variables. El escalado puede prevenir en parte el sobreajuste (overfitting) del modelo. Además, en caso de necesitar modificar el modelo que tenemos en mente, tenemos la posibilidad de ya tener los datos normalizados y escalados.

En primer lugar, se verificó la asimetría de las variables numéricas. Esto debido a que las distribuciones asimétricas pueden tener valores atípicos no manejados y pueden afectar la precisión de la escala. Como se observó cierta asimetría, se decidió aplicar una transformación para que se pueda normalizar la distribución y reducir la asimetría.

category	1.165691
amt	1.061963
state	285.670958
city_pop	1.188109
is_fraud	13.029122
trans_hora	-0.282825
trans_mes	0.298516
trans_dia	-0.012131
delay_entre_trans	1.157775
edad_usuario	0.611148
dif_lat_comprador_merch	-0.001202
dif_long_comprador_merch	-0.001450
dif_lat_prev_merch	0.565637
dif_long_prev_merch	0.565088
gender_f	-0.190655
dtype:	float64

Para la normalización, se utilizó el método **Yeo-Johnson**, la cual es una transformación similar a la **Box-Cox**, pero más flexible puesto que puede ser utilizada cuando hay valores negativos o ceros. Por otro lado, **Box-Cox** es una transformación paramétrica que se utiliza para estabilizar la varianza⁶ y hacer que la distribución de las variables sean más normales. Luego se procedió al escalado, dentro de los diferentes tipos se decidió por **RobustScaler**. Dicha técnica consiste en escalar los datos de manera que se eliminen los efectos de los valores extremos, utilizando medidas de posición y dispersión más robustas que la media y la desviación estándar, que son sensibles a los valores extremos. Una vez aplicadas ambas técnicas se pudo obtener una asimetría más acorde.

data.skew()	
category	1.165691
amt	1.061963
state	-0.315369
city_pop	1.188109
is_fraud	13.029122
trans_hora	-0.282825
trans_mes	0.298516
trans_dia	-0.012131
delay_entre_trans	1.157775
edad_usuario	0.611148
dif_lat_comprador_merch	-0.001202
dif_long_comprador_merch	-0.001450
dif_lat_prev_merch	0.565637
dif_long_prev_merch	0.565088
gender_f	-0.190655
dtype:	float64

División de dataset

Una vez preparados los datos se procedió a la división del dataset en un set de datos de entrenamiento y otro set para testeo. La proporción que se decidió fue 70% de datos para el entrenamiento y 30% para testeo.

⁶ <https://es.wikipedia.org/wiki/Varianza>

```
: from sklearn.model_selection import train_test_split

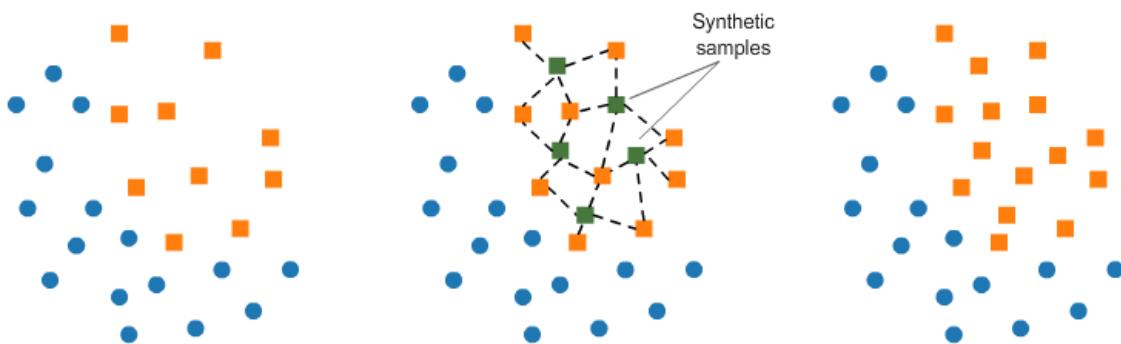
: #Dividimos nuestro dataset
X=data[['category', 'amt', 'state', 'city_pop', 'trans_hora',
       'trans_mes', 'trans_dia', 'delay_entre_trans', 'edad_usuario',
       'dif_lat_comprador_merc', 'dif_long_comprador_merc',
       'dif_lat_prev_merc', 'dif_long_prev_merc', 'gender_f']]
X_train, X_test, y_train, y_test= train_test_split(X, data['is_fraud'], test_size=0.30)
```

Luego de la división se procedió a un pequeño análisis para ver la distribución de set de entrenamiento donde se observó que había un 99,42% de casos legítimos y un 0.57% de casos fraudulentos, concluyendo en que estaba muy desbalanceado.

```
: print('Primero la distribución de nuestro trainset','\n')
print('El X_train tiene {} casos'.format(X_train.shape[0]),'\n')
print('El y_train tiene {} casos'.format(y_train.shape[0]),'\n')
print('Ahora en porcentajes','\n')
print(y_train.value_counts(normalize=True)*100,' \n')
print('hay {} cantidad de casos legítimos.'.format(no_fraudes.shape[0]))
print('Hay {} cantidad de casos fraudes.'.format(fraudes.shape[0]))
```

Primero la distribución de nuestro trainset
 El X_train tiene 907672 casos
 El y_train tiene 907672 casos
 Ahora en porcentajes
 0.0 99.424241
 1.0 0.575759
 Name: is_fraud, dtype: float64
 hay 902446 cantidad de casos legítimos.
 Hay 5226 cantidad de casos fraudes.

Dentro de los diferentes métodos de balanceo, decidimos utilizar **SMOTE** (Synthetic Minority Oversampling Technique) en el cual, básicamente, se realiza oversampling sobre la parte de datos minoritaria generando muestras sintéticas y no copias, como podría ocurrir si utilizáramos oversampling solamente. Esto reduciría las posibilidades de sobreajuste de nuestro modelo.



Luego de la aplicación, se obtuvo un set de entrenamiento balanceado.

```
#Ahora verificamos los trainsets balanceados
print('Primero la distribución de nuestro trainset','\n')
print('El X_train_resampled tiene {} casos'.format(X_train_resampled.shape[0]),'\n')
print('El y_train_resampled tiene {} casos'.format(y_train_resampled.shape[0]),'\n')
print('Ahora en porcentajes','\n')
print(y_train_resampled.value_counts(normalize=True)*100,'\
\n')
print('hay {} cantidad de casos legítimos.'.format(no_fraudes_resampled.shape[0]))
print('Hay {} cantidad de casos fraudes.'.format(fraudes_resampled.shape[0]))
```

Primero la distribución de nuestro trainset

El X_train_resampled tiene 1804892 casos

El y_train_resampled tiene 1804892 casos

Ahora en porcentajes

0.0	50.0
1.0	50.0
Name:	is_fraud, dtype: float64

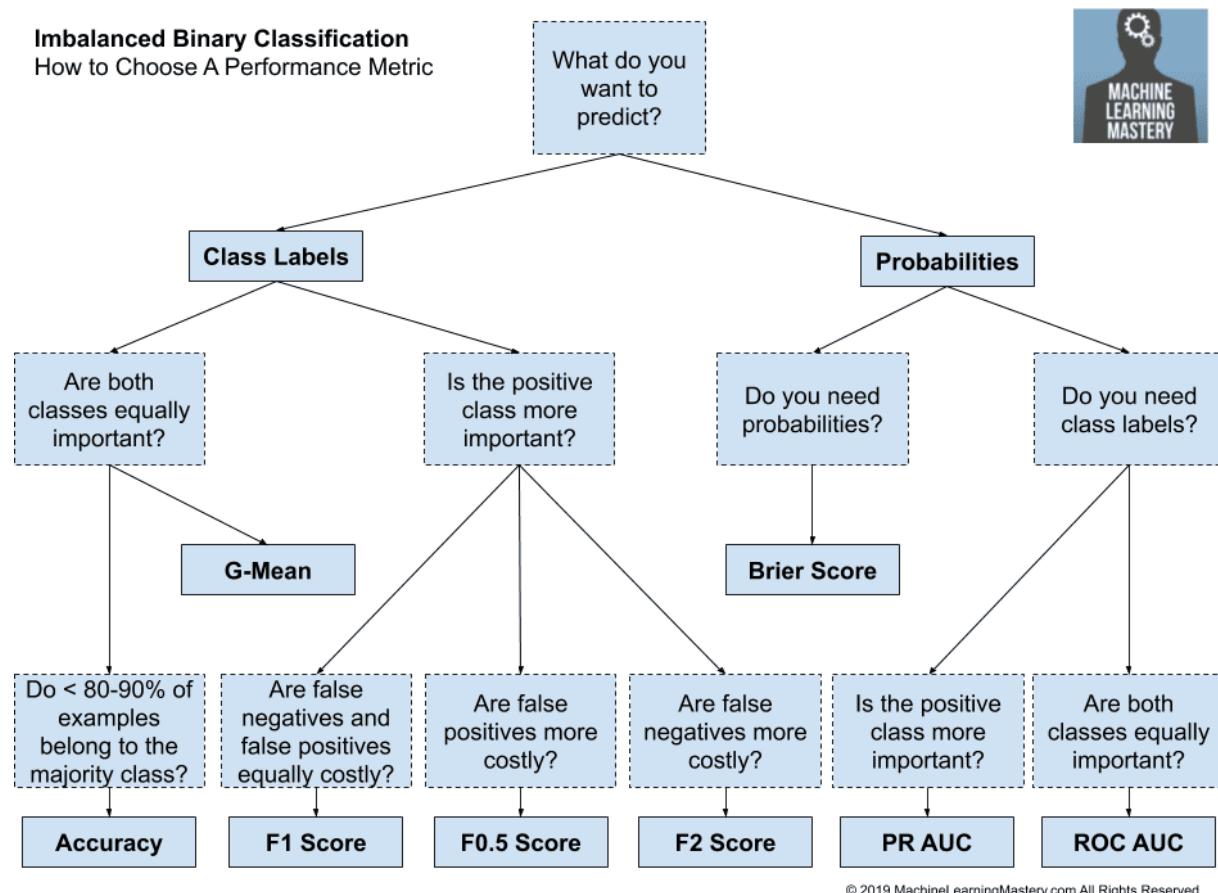
hay 902446 cantidad de casos legítimos.
 Hay 902446 cantidad de casos fraudes.

Construcción y evaluación del modelo

En este punto, se decidió las métricas de medición para poder medir la generalización de nuestros modelos.

- **ROC AUC:** Dicha métrica sirve para evaluar la capacidad de clasificación binaria de un modelo, es decir, mide la capacidad del modelo de distinguir entre la clase positiva y negativa, y nos provee una medida de calidad global del modelo. Esta medida es útil para minimizar tanto los falsos positivos como los falsos negativos. En el caso de la detección de fraudes, es importante minimizar tanto el número de transacciones fraudulentas que pasan desapercibidas (falsos negativos) como el número de transacciones legítimas que se identifican como fraudulentas (falsos positivos).
- **Recall:** Mide la proporción de casos positivos que el modelo ha identificado correctamente al total de casos positivos en los datos de prueba. Es útil en casos donde es importante no perder ningún caso positivo, es decir, es peor identificar un caso como negativo cuando en realidad es positivo.
- **Accuracy:** Es la métrica más básica y representa la proporción de predicciones correctas en relación al total de predicciones realizadas por el modelo. Se calcula como el número de predicciones correctas dividido por el número total de predicciones.
- **F1 Score:** Métrica que consiste en la media armónica de precisión y recall, y proporciona una medida entre ambas métricas.

- **Precision:** Métrica donde mide la proporción de verdaderos positivos respecto a los valores positivos predichos.
- **Matriz de confusión:** La matriz de confusión es una herramienta utilizada para evaluar el rendimiento de un modelo de clasificación. Consiste en una tabla que muestra la frecuencia con la que el modelo clasifica correctamente o incorrectamente las observaciones en cada una de las clases. Se divide en cuatro celdas:
 - **Verdaderos positivos (TP):** la observación es positiva y se clasifica como positiva.
 - **Falsos positivos (FP):** la observación es negativa pero se clasifica como positiva.
 - **Verdaderos negativos (TN):** la observación es negativa y se clasifica como negativa.
 - **Falsos negativos (FN):** la observación es positiva pero se clasifica como negativa.



Dentro de los valores posibles de métricas se eligió como primordial **ROC AUC** debido a que consideramos necesario, debido al contexto del negocio, tener en cuenta tanto los fraudes verdaderos como los legítimos verdaderos, puesto que si bien se quiere funcionar como una barrera para los fraudes, se considera también la experiencia del usuario cuya transacción sea legítima.

Por otro lado, también se tiene en cuenta el recall y la matriz de confusión para una mejor visualización de la capacidad del modelo.

Se evaluaron, primeramente, 3 modelos RandomForest. El primer modelo era un RandomForest default. Para el segundo modelo se identificaron parámetros para su modificación y el tercero, similar al segundo, sin la modificación a los pesos en las clases.

Primer modelo

```
#Modelo 1, parametros simples
rfc_1= RandomForestClassifier(n_estimators=100)

#Ajustamos el modelo a nuestros datos resampled
%time rfc_1.fit(X_train_resampled,y_train_resampled)

CPU times: total: 11min 38s
Wall time: 11min 39s
:
RandomForestClassifier
RandomForestClassifier()
```

```
#Construimos el modelo 2 modificando la distribucion de los pesos y max_depth
rfc_2= RandomForestClassifier(n_estimators=100,max_depth=10,random_state=42,n_jobs=-1, class_weight='balanced')
```

```
#Ajustamos el modelo a nuestros datos resampled
%time rfc_2.fit(X_train_resampled,y_train_resampled)

CPU times: total: 13min 18s
Wall time: 1min 13s
:
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', max_depth=10, n_jobs=-1,
random_state=42)
```

```
#Creamos el tercer modelo
rfc_3=RandomForestClassifier(n_estimators=100, max_depth=10,
min_samples_split=75,min_samples_leaf=10, random_state=42, n_jobs=-1)
```

```
#Entrenamos el modelo
%time rfc_3.fit(X_train_resampled,y_train_resampled)
```

```
CPU times: total: 13min 11s
Wall time: 1min 11s
:
RandomForestClassifier
RandomForestClassifier(max_depth=10, min_samples_leaf=10, min_samples_split=75,
n_jobs=-1, random_state=42)
```

Cada modelo fue entrenado y evaluado. Posteriormente, se procedió a ajustar a los parámetros de cada uno para buscar la mejor configuración posible. Dentro de los métodos posibles para ajuste de parámetros están:

- **Grid Search:** Implica definir una cuadrícula de valores para cada hiperparámetro y probar todas las combinaciones posibles. Lo descartamos debido a que es muy costoso computacionalmente cuando se tienen muchos valores a considerar e hiperparámetros. Por otro lado, durante la ejecución se procesó primero utilizando **Random Search** y nuestros equipos tardaron demasiado e incluso hubo

complicaciones, así que directamente se descartó incluso tratar de ejecutar el Grid Search.

- **RandomSearch:** Este método implica mostrar aleatoriamente valores de hiperparámetros dentro de un rango específico y ejecutar el modelo con esos valores. Este método es menos costoso y a menudo produce resultados comparables o mejores. En nuestro caso, se optó por este método de búsqueda al implementar las técnicas de validación cruzada con **StratifiedKFold** y **HalvingRandomSearchCV**.

Para tener en cuenta, **StratifiedKFold** es una técnica de validación cruzada para dividir un conjunto de datos en k conjuntos de una manera estratificada, entonces cada conjunto mantiene la misma proporción de las clases target que el conjunto de datos original. En cada iteración, se utiliza un subconjunto diferente como conjunto de prueba y el resto de subconjuntos se combinan y se utilizan como conjunto de entrenamiento.

Por otro lado, **Halving Random Search CV**, es una técnica de búsqueda de hiperparámetros que reduce el tiempo de búsqueda en comparación a la búsqueda aleatoria normal. En el proceso divide los conjuntos de entrenamiento y prueba, va repitiendo el proceso donde evalúa y descarta modelos.

En nuestro caso, definimos primero nuestro modo de realizar cross validation para luego instanciar los objetos de búsqueda de los mejores parámetros para cada uno de los 3 modelos entrenados anteriormente. Posteriormente, se evaluaron con los parámetros resultantes en cada uno de los casos mostrando diferencias en el desempeño de cada uno.

Modelo 1

Antes de parametrizar

```
: %time evaluar(rfc_1)

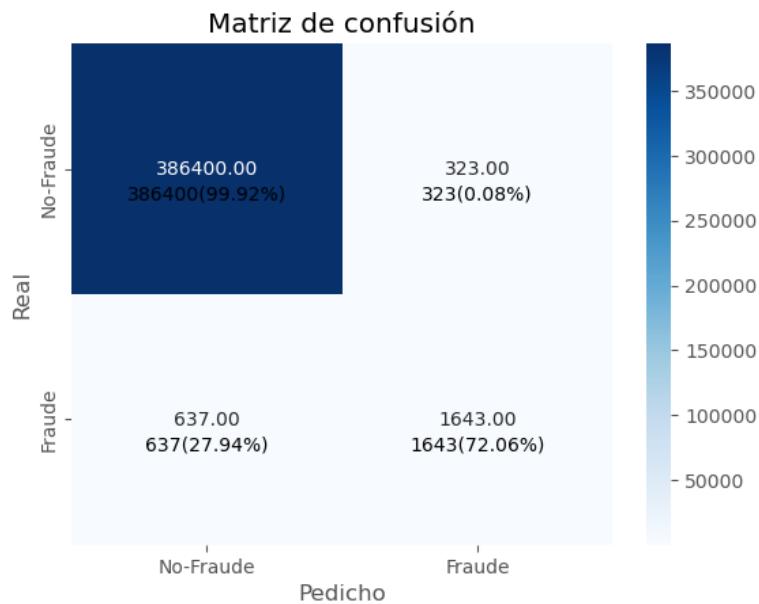
ROC AUC Score:  0.8598894059717525

Precision Score:  0.8357070193285859

Recall Score:  0.7206140350877193

F1 Score:  0.7739048516250588

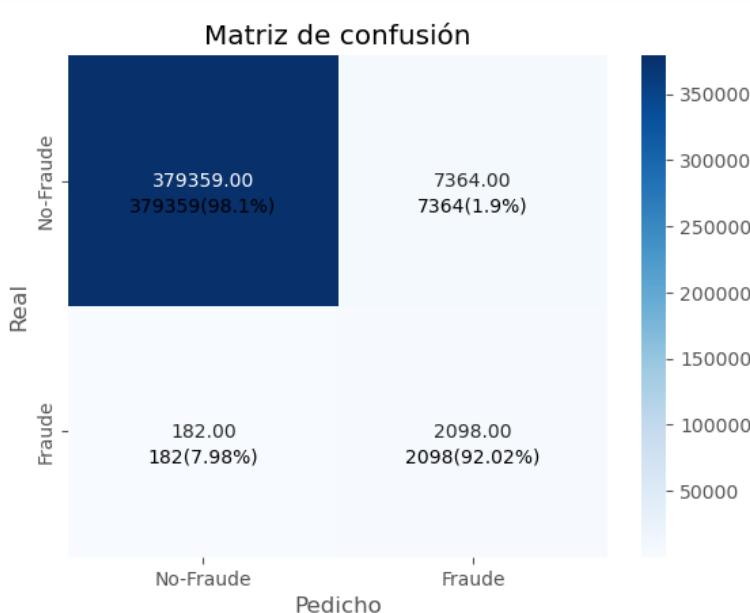
Accuracy Score:  0.9975321527083338
```



Después de parametrizar

```
: #veamos los resultados
%time evaluar(rfc_1)
```

ROC AUC Score: 0.9505666926202356
Precision Score: 0.2217290213485521
Recall Score: 0.9201754385964912
F1 Score: 0.3573496848918413
Accuracy Score: 0.9806016920177993

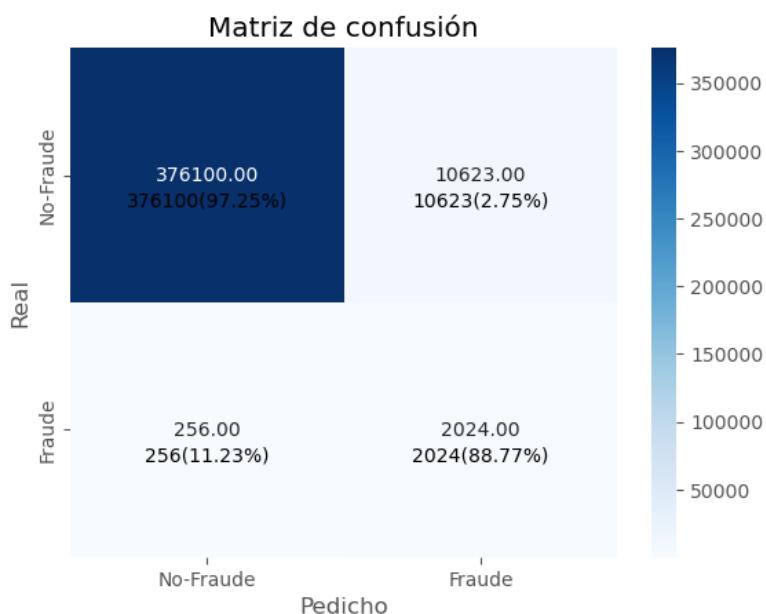


Modelo 2

Antes de parametrizar

```
: #evaluamos el segundo modelo  
%time evaluar(rfc_2)
```

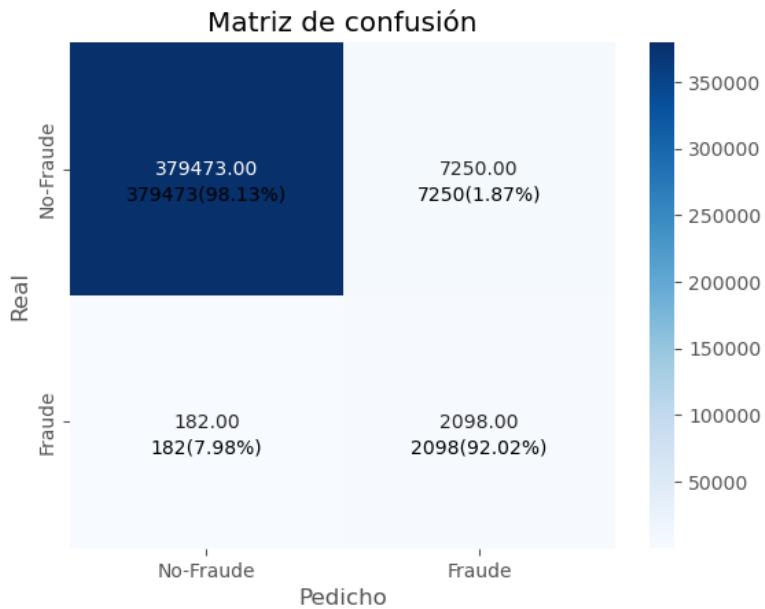
ROC AUC Score: 0.9301250121862918
Precision Score: 0.16003795366490076
Recall Score: 0.887719298245614
F1 Score: 0.2711864406779661
Accuracy Score: 0.972033634702046



Después de parametrizar

```
: #evaluamos el segundo modelo  
%time evaluar(rfc_2)
```

ROC AUC Score: 0.9507140849398029
Precision Score: 0.22443303380402224
Recall Score: 0.9201754385964912
F1 Score: 0.3608531131750946
Accuracy Score: 0.9808947488836847

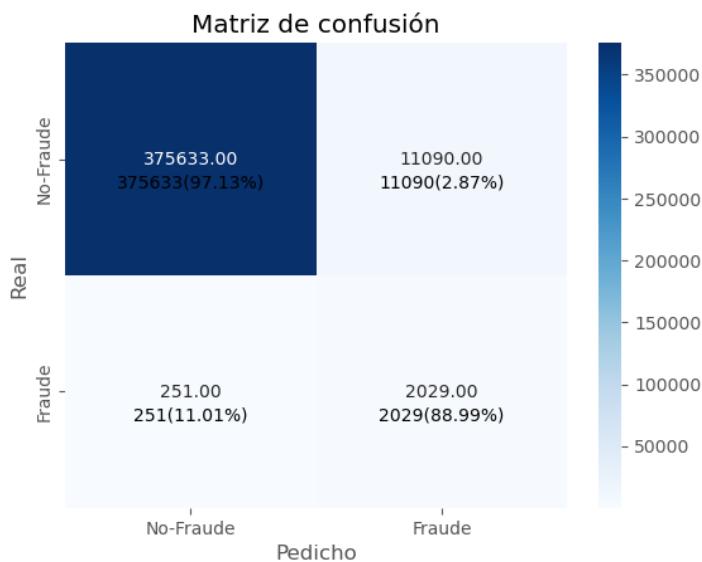


Modelo 3

Antes de parametrizar

```
#Evaluamos el modelo
%time evaluar(rfc_3)

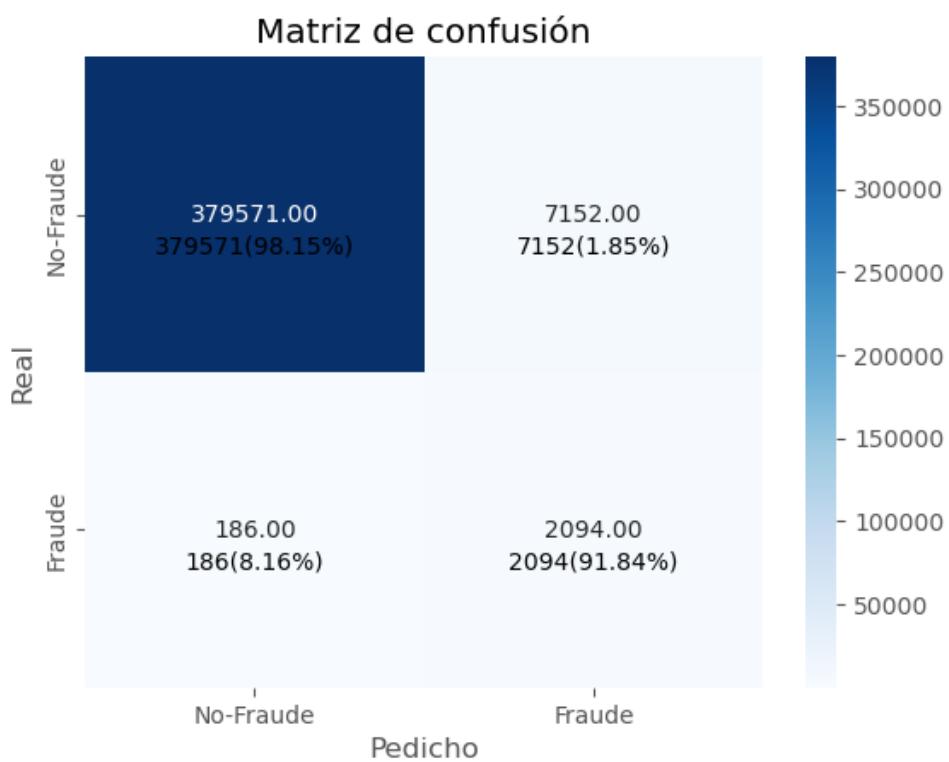
ROC AUC Score:  0.93061771207017
Precision Score:  0.15466117844347893
Recall Score:  0.8899122807017544
F1 Score:  0.26352360542892395
Accuracy Score:  0.9708459831929317
```



Después de parametrizar

```
#evaluamos el 3er modelo
%time evaluar(rfc_3)
```

```
ROC AUC Score: 0.9499635976355714
Precision Score: 0.22647631408176508
Recall Score: 0.9184210526315789
F1 Score: 0.3633524206142634
Accuracy Score: 0.981136392264327
```



Para la decisión del modelo a elegir, se decidió realizar un cuadro comparativo donde se evalúan los desempeños de todos los modelos, teniendo principalmente en cuenta la métrica **ROC AUC** y la matriz de confusión conjunta al recall de cada una.

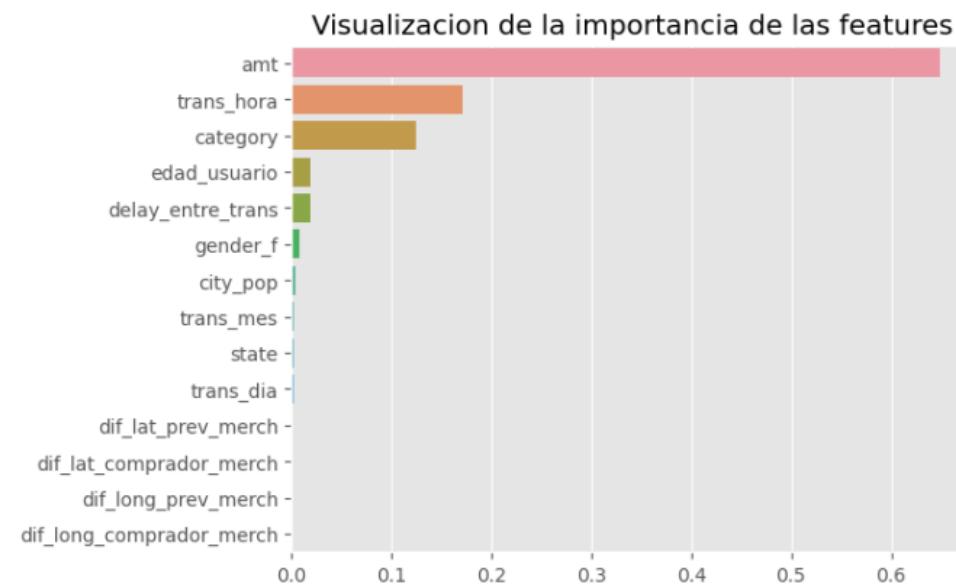
Modelo	Tipo de métrica					Matriz de confusión				
	ROC_AUC	Precision	Recall	F1	Accuracy	Predicho	No Fraude	No Fraude	Fraude	Fraude
rfc_1	0.85988	0.8357	0.72061	0.7739	0.99752	-	386400(99,92%)	323(0,08%)	637(27,94%)	1643(72,06%)
rfc_1 (parametrizado)	0.95056	0.22172	0.92017	0.35734	0.9806	-	379359(98,1%)	7364(1,9%)	182(7,98%)	2098(92,02%)
rfc_2	0.93012	0.16003	0.88771	0.27118	0.97203	-	376100(97,25%)	10623(2,75%)	256(11,23%)	2024(88,77%)
rfc_2 (parametrizado)	0.95071	0.22443	0.92017	0.36085	0.9808	-	379473(98,13%)	7250(1,87%)	182(7,98%)	2098(92,02%)
rfc_3	0.93061	0.15466	0.88991	0.26352	0.97084	-	375633(97,13%)	11090(2,87%)	251(11,01%)	2029(88,99%)
rfc_3 (parametrizado)	0.94996	0.22647	0.91842	0.36335	0.98113	-	379571(98,15%)	7152(1,85%)	186(8,16%)	2094(91,84%)

Se optó por el modelo 2 parametrizado de Random Forest debido a los resultados evaluados.

Interpretación del modelo

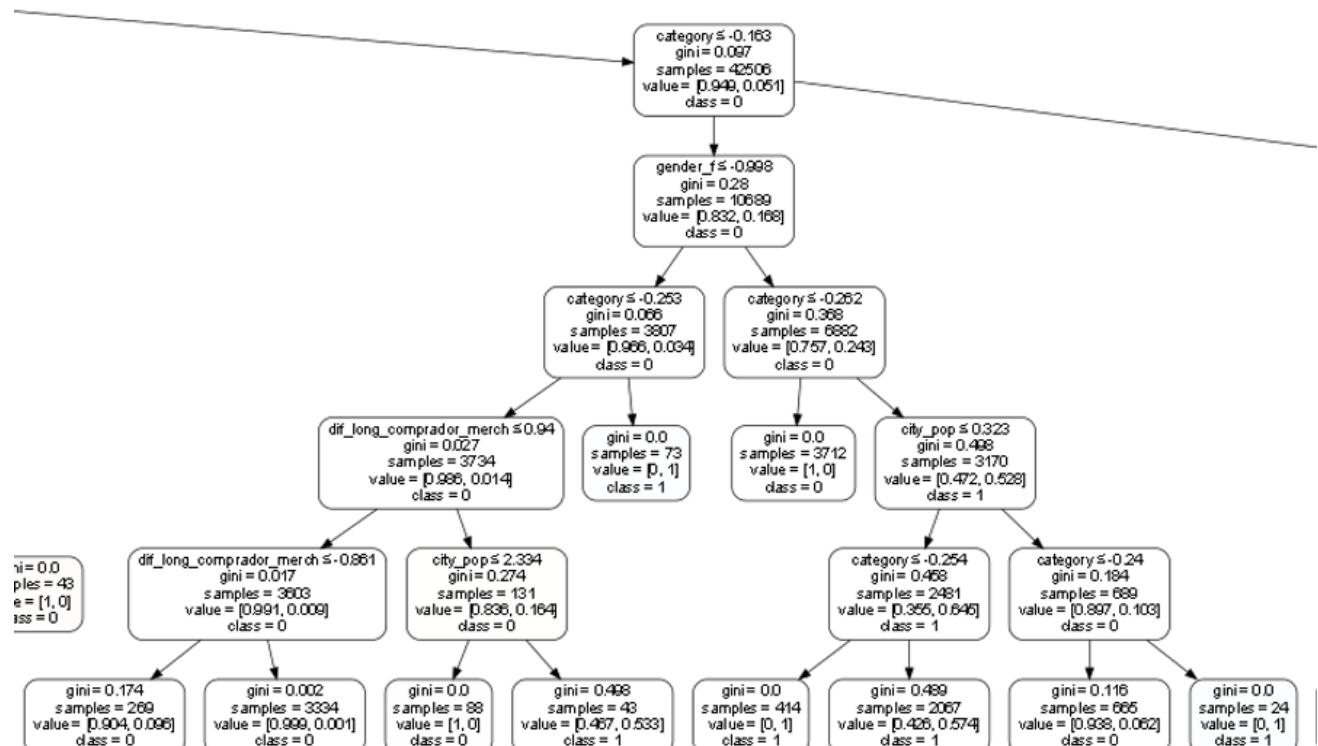
Debido a que modelo seleccionado es un modelo de caja negra, es decir, un modelo donde se conoce la entrada y la salida pero no se conoce como el modelo interpreta las relaciones, en esta etapa nos limitamos a buscar cuáles eran las características más relevantes que tomaba en cuenta el modelo a la hora de tomar sus decisiones.

feature_scores	
:	amt
:	trans_hora
:	category
:	edad_usuario
:	delay_entre_trans
:	gender_f
:	city_pop
:	trans_mes
:	state
:	trans_dia
:	dif_lat_prev_merch
:	dif_lat_comprador_merch
:	dif_long_prev_merch
:	dif_long_comprador_merch
	dtype: float64



Se pudo observar que la cantidad del monto en la transacción y la hora de la transacción tiene mucho impacto en la clasificación de los casos, a la vez que la categoría de las transacciones. Esto, en parte, se pudo observar con los primeros análisis utilizando mapas de calor y gráficas.

Por otro lado, también se decidió extraer una muestra de un árbol del bosque para ver como era visualmente. Debido a la inmensidad del árbol, solo se pudo capturar una pequeña parte de él.



Despliegue en la nube

Introducción

La implementación y el despliegue de un modelo de machine learning son etapas cruciales en cualquier proyecto de data science. Una vez que se ha creado un modelo que cumple con los requisitos de precisión y rendimiento, es necesario ponerlo a disposición de los usuarios de manera eficiente y escalable. En esta sección del informe, se describe el proceso de implementación y despliegue del modelo de detección de fraudes de tarjetas de crédito previamente desarrollado en el proyecto. El objetivo de este proceso es ofrecer el

modelo como servicio a través de una aplicación web interactiva, que se ha construido utilizando **Streamlit**, una biblioteca de Python que permite crear aplicaciones web de manera sencilla.

Además en esta sección, se discutirán los desafíos y limitaciones encontrados durante el proceso de implementación y despliegue. En resumen, esta sección del informe se centrará en cómo hemos implementado y desplegado el modelo de detección de fraudes de tarjetas de crédito como servicio en la nube, a través de una aplicación web interactiva construida con **Streamlit**.

Actualización de herramientas utilizadas en el proyecto

Durante el desarrollo del proyecto, se ha decidido actualizar las herramientas utilizadas en la implementación de la aplicación web. La primera herramienta que se ha incorporado es "**PyCharm**". **PyCharm** es un IDE (entorno de desarrollo integrado) de Python, que proporciona una amplia gama de características y herramientas para el desarrollo de aplicaciones. Algunas de sus características destacadas son el análisis de código, el depurador interactivo, la finalización de código y la integración con herramientas de control de versiones.

Otra herramienta que se ha incorporado en la implementación de la aplicación web es "**Streamlit**". **Streamlit** es un framework de Python para la creación de aplicaciones web de manera rápida y sencilla. Permite crear aplicaciones interactivas con gráficos y visualizaciones, que se pueden actualizar en tiempo real con tan solo modificar el código. Además, es fácil de utilizar y ofrece una amplia gama de widgets y elementos interactivos para la creación de la interfaz de usuario.

Con la incorporación de estas herramientas, se ha logrado mejorar el flujo de trabajo y aumentar la eficiencia en el desarrollo de la aplicación web. **PyCharm** ha facilitado la creación y mantenimiento del código, mientras que **Streamlit** ha permitido desplegar la aplicación en la web como servicio de manera sencilla y rápida.

Estructura del desarrollo

En este proyecto de detección de fraude en tarjetas de crédito utilizando machine learning, se ha utilizado una estructura de código modular, separando diferentes funcionalidades en diferentes archivos.

El archivo [**app.py**](#) es el archivo principal de la aplicación y se encarga de desplegar la interfaz gráfica de usuario utilizando la biblioteca *Streamlit*. Además, este archivo se comunica con los otros dos archivos del proyecto para procesar los datos del usuario y hacer la predicción correspondiente.

El archivo [**utils.py**](#) contiene varias funciones útiles para el manejo del modelo de machine learning, incluyendo la carga del modelo y la codificación de etiquetas de clase. También se incluyen funciones para la lectura y escritura de archivos en formato CSV.

El archivo [**procesador.py**](#) contiene la clase Procesador que se utiliza para procesar los datos de entrada del usuario antes de hacer la predicción con el modelo. Esta clase tiene funciones para escalar, normalizar y winsorizar los datos, además de una función para realizar la predicción del modelo.

La separación en diferentes archivos permite una mejor organización del código y facilita su mantenimiento y actualización. Además, se puede reutilizar el código de manera más efectiva en diferentes partes del proyecto.

Incorporación del modelo de machine learning en la aplicación web

Para incorporar el modelo de machine learning ya entrenado en la aplicación web, se ha utilizado la biblioteca de Python "**pickle**". **Pickle** es una biblioteca de serialización de objetos de Python, que permite guardar y cargar objetos en un formato binario.

Se ha utilizado la función "pickle.dump()" para guardar el objeto del modelo de machine learning en un archivo con formato ".pkl", y la función "pickle.load()" para cargar el objeto en la aplicación web desarrollada con *Streamlit*. De esta manera, se ha logrado incorporar el modelo en la aplicación y ofrecer el servicio de detección de fraudes de tarjetas de crédito en línea.

Pickle es una herramienta muy útil en Python porque permite guardar objetos de Python de una manera muy fácil y rápida, sin tener que preocuparse por la codificación y decodificación de los datos. Además, es muy flexible y se puede utilizar para guardar y cargar casi cualquier tipo de objeto de Python.

Primero realizamos la exportación del modelo 2 ya entrenado en formato “.pkl” desde *Jupyter Notebook*:

Exportacion del modelo 2

```
.. ruta_archivo2='C:/Users/Freyja/Desktop/Modelos/modelo2_entrenado.pkl'  
.. #Guardamos el modelo entrenado en un archivo pkl  
.. with open(ruta_archivo2, 'wb') as f:  
..     pickle.dump(rfc_2, f)
```

Para leer el archivo “.pkl” en el código de la aplicación, definimos una función que nos permite cargar el modelo desde un archivo.pkl pasando la ruta del mismo:

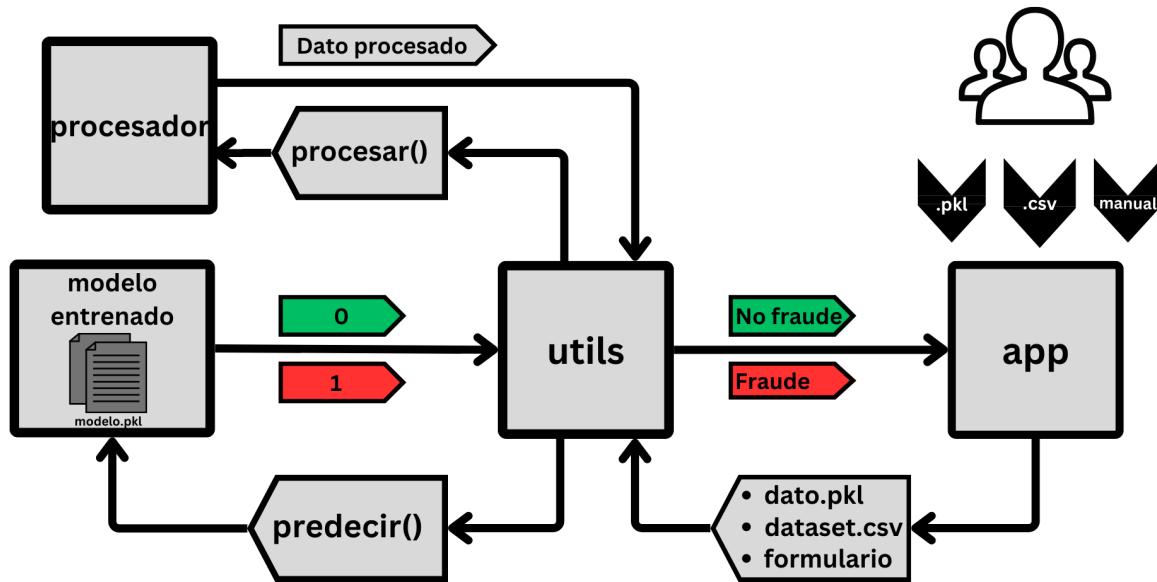
```
def cargar_modelo(ruta):  
    # Cargar el modelo entrenado desde el archivo pkl  
    with open(ruta, 'rb') as pkl:  
        archivo = pickle.load(pkl)  
    return archivo
```

De esta manera podemos utilizar la serie de métodos afines al modelo de machine learning ya entrenado. En particular el método “predict()”, que nos permite realizar la predicción de los datos en fraude y no fraude. Este lo incorporamos en el desarrollo por medio de una función:

```
def predecir(modelo, datos):  
    # Realizar la predicción con el modelo cargado  
    prediccion = modelo.predict(datos)  
    return prediccion
```

Planteamiento de la aplicación

La idea de la aplicación pensada para el procesamiento de datos fue la siguiente:



El usuario podrá procesar los datos de 3 formas diferentes. La primera forma es la de cargar un archivo de tipo **.pkl**, este archivo representaría a una fila de un conjunto de datos. En segundo lugar, se podrá cargar un archivo de tipo **.csv**, este archivo representaría a un conjunto de datos. Y en tercer lugar se podrá **ingresar manualmente los datos** por medio de un formulario.

Estos datos o archivos son procesados por **utils.py**, el cual llama a la función **predecir(modelo, datos)**, que a su vez llama al método `predict(x)` de la biblioteca de `sklearn`, utilizando el modelo entrenado previamente cargado desde el archivo `".pkl"`. Esta función devuelve en definitiva un **1 o 0**, o bien una **lista de 1 o 0**.

Una vez que **utils.py** tiene la predicción del dato o conjunto de datos, clasifica la predicción en **"Fraude"** o **"No fraude"** con la función **obtener_clasificacion(prediccion)**, la cual envía a **app.py** para que se muestre en pantalla.

Alternativamente, si se trata de un dato ingresado manualmente por medio del formulario, el dato es procesado por **procesador.py** antes de realizar la predicción preparando el valor entregado para que este sea entendido por el modelo y realizar la predicción. De igual manera, esta predicción es clasificada y mostrada al usuario por pantalla. En el caso del procesamiento de archivos de tipo **.csv** y **.pkl**, no existe un pre-procesamiento, por lo tanto

los datos cargados deben de estar previamente procesados para el correcto funcionamiento del modelo entrenado.

Finalmente, para el procesamiento del archivo **.csv**, se agregó la opción de descargar el resultado de la predicción, agregando una columna llamada “**clasificación**” que cuenta con la misma clasificación que la de los datos individuales, es decir, “**Fraude**” y “**No fraude**”.

Cargar archivos

```
archivo_subido = st.file_uploader("archivo", label_visibility="hidden", type=['pkl', 'csv'], key="1")
```

Procesar archivos

```
if archivo_subido is not None:
    box_info.empty()
    box_warning.empty()

    nombre, extension = os.path.splitext(archivo_subido.name)

    if extension == '.pkl':
        categoria = predecir_pkl(archivo_subido, modelo)
        info_resultado = st.info(f"La transacción analizada es: {categoria}")
        st.stop()

    elif extension == '.csv':
        df_prediccion = predecir_csv(archivo_subido, modelo)
        info_box_wait = st.info('Cargando...')
        st.dataframe(df_prediccion)
        info_box_wait.empty()
        colA, colB, colC = st.columns([1, 1, 1])
        with colB:
            boton_crear_descargable = st.button('CREAR ARCHIVO', use_container_width=True)
        if boton_crear_descargable:
            crear_descargable(df_prediccion, nombre)
```

Consultar manualmente

```
else:
    procesador = Procesador('data.pkl')
    st.header("CONSULTAR DATOS MANUALMENTE:")
    dato_sin_procesar = agregar_datos_manualmente()
    colA, colB, colC = st.columns([1, 1, 1])
    with colB:
        boton_predecir = st.button("PREDECIR", use_container_width=True)
    if boton_predecir:
        spinner = st.spinner('Procesando dato... esto puede llevar un tiempo.')
        with spinner:
            dato_procesado = procesar_dato(dato_sin_procesar, procesador, procesador.get_df())
            time.sleep(5)
        # st.write(dato_procesado) #ver dataframe creado

        categoria = predecir_dato(dato_procesado, modelo)
        info_resultado = st.info(f"La transacción analizada es:{categoria}")
```

Obtener clasificación

```
3 usages
def obtener_clasificacion(prediccion):
    # Aclaración: la predicción es un valor de 1 o 0
    # No se requiere el uso de "for" pues tendremos sólo 1 dato
    if prediccion == 1:
        clasificacion = 'Fraude'
    else:
        clasificacion = 'No fraude'
    return clasificacion
```

Predecir tipos de datos

```

def predecir_dato(dato_procesado, modelo):
    info_box_wait_clasificacion = st.info('Realizando la clasificación...') # Cargamos el dato en una variable
    prediccion = predecir(modelo, dato_procesado) # Obtenemos la predicción
    categoria = obtener_clasificacion(prediccion) # Obtenemos la clasificación
    info_box_wait_clasificacion.empty()
    return categoria

1 usage
def predecir_pk1(archivo, modelo):
    info_box_wait_clasificacion = st.info('Realizando la clasificación...')
    dato = pickle.loads(archivo.getvalue()) # Obtenemos el valor de la fila subida
    prediccion = predecir(modelo, dato)
    categoria = obtener_clasificacion(prediccion)
    info_box_wait_clasificacion.empty()
    return categoria

1 usage
def predecir_csv(archivo, modelo):
    info_box_wait_clasificacion = st.info('Realizando la clasificación...')
    dataset_subido = pd.read_csv(archivo) # Obtenemos el valor de la dataset subido
    prediccion = predecir(modelo, dataset_subido)
    df_prediccion = agregar_prediccciones(dataset_subido, prediccion) # Se agrega la clasificación al dataframe
    info_box_wait_clasificacion.empty()

    return df_prediccion

```

Dificultades y consideraciones

Durante el desarrollo del entregable hemos tenido diferentes dificultades y contras para poder llegar a la solución de forma amigable.

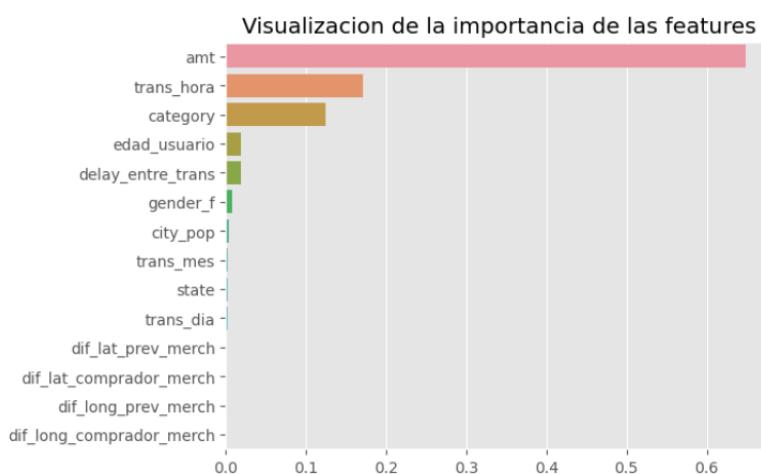
Uno de los principales inconvenientes se dio antes de siquiera empezar el desarrollo, esta fue durante la etapa de decisión sobre en qué plataforma plasmar el código y despliegue en la nube del modelo. Pasamos por diferentes alternativas y realizamos sus debidas investigaciones sobre ellas, y decidimos finalmente optar por **Streamlit** debido a su facilidad de uso y flexibilidad.

En segundo lugar, tuvimos dificultades en cómo integrar el modelo entrenado en la aplicación web. Tras una investigación sobre formas posibles de hacer esto, vimos la posibilidad de guardar el modelo entrenado en formato **pickle**, que permite serializar objetos de Python, lo que lo hace fácilmente transportable al código que lo utilizará. Además, de esta forma se pudo probar diferentes modelos fácilmente, simplemente cambiando el archivo del modelo.

En tercer lugar, la dificultad más grande ya empezada la entrega fue durante el desarrollo del formulario que permite el ingreso manual de datos. Durante este, tuvimos dificultades en la concordancia de datos que recibe nuestro modelo entrenado, en relación con los datos ingresados por el usuario. Esto se debía a que nuestro modelo, esperaba cierta cantidad de

datos, con tipos específicos y con un orden concreto, en la cual todos los datos son números. Por otra parte, el formulario no tenía todos los datos que esperaba el modelo, esto es así, ya que, datos como la **latitud y longitud** de donde se encontraba el comprador cuando realizó la transacción, o bien, el **tiempo de la transacción anterior** a la transacción consultada son datos que usuario normal no tiene o bien no son realistas de ingresar en un formulario de este tipo.

Por esta razón se decidió armar el formulario con solamente 7 datos, para ello tuvimos en cuenta los datos con mayor importancia obtenidos durante la interpretación del modelo. Los datos elegidos a incluir en el formulario son: Monto, Categoría, Edad de usuario, Género, Hora de la transacción, Día de la transacción, Mes de la transacción.



Teniendo en cuenta esto, debíamos procesar los datos previamente a realizar la predicción, ya que, o daba un error al no ser lo que el modelo esperaba, o bien no realizaba una predicción confiable.

Por esta razón se implementó la clase Procesador, que cuenta con la serie de funciones necesarias para procesar los datos. Adicionalmente, para aliviar el procesamiento, se utilizaron también funciones en **utils.py** para procesar por ejemplo, la variable **categoría, género y día**, traduciendo los valores del formulario de lenguaje natural al que entiende el modelo.

Finalmente, cabe decir que se optó por no realizar el preprocesamiento de los archivos .pkl y .csv debido a las dificultades que conllevaba hacerlo debido a que el modelo espera determinados datos procesados y determinadas columnas en orden:

Columnas admitidas: [category], [amt], [state], [city_pop], [trans_hora], [trans_mes], [trans_dia], [delay_entre_trans], [edad_usuario], [dif_lat_comprador_merch], [dif_long_comprador_merch], [dif_lat_prev_merch], [dif_long_prev_merch], [gender_f]

En caso de no cumplirlo, directamente da un error. Esto se decidió puesto que al ingresar archivos, estos pueden no tener las columnas pedidas ni en el orden deseado, también, en caso de querer realizar una verificación, podrían estar en otro idioma o escrito de otra manera, aunque signifique lo mismo que la columna que el modelo espera. Por otro lado, también puede suceder que el tipo de dato guardado no sea el esperado, por lo que tampoco se podría saber bien cuantos son los necesarios para castejarlos y procesarlos. Esos son algunos motivos por los cuales se descartó realizar el preprocesamiento de los archivos.

Imágenes

Visualización de la página:

DETECCIÓN DE FRAUDES CON ML

CONSULTAR CONJUNTO DE DATOS:

Columnas admitidas

Columnas admitidas: [category], [amt], [state], [city_pop], [trans_hora], [trans_mes], [trans_dia], [delay_entre_trans], [edad_usuario], [dif_lat_comprador_merc], [dif_long_comprador_merc], [dif_lat_prev_merc], [dif_long_prev_merc], [gender_f]

Drag and drop file here
Limit 200MB per file • PKL, CSV

Browse files

Puede agregar un dataset completo o una fila

CONSULTAR DATOS MANUALMENTE:

AGREGAR DATOS MANUALMENTE

INGRESA EL MONTO (MONTO)

0,00

INGRESA LA HORA DE LA TRANSACCIÓN (HORA)

0 23

SELECCIONA LA CATEGORÍA (CATEGORÍA)

Entretenimiento

INGRESA EL DÍA DE LA TRANSACCIÓN (DÍA)

Lunes

INGRESA LA EDAD DEL USUARIO (EDAD)

18

SELECCIONA EL MES DE LA TRANSACCIÓN (MES)

1

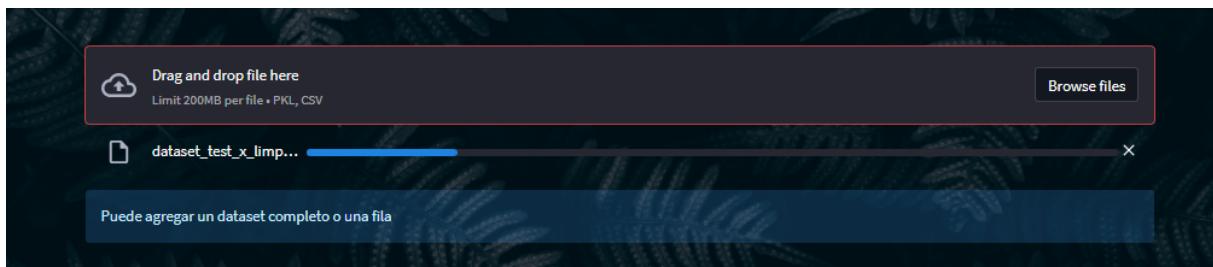
SELECCIONA EL GÉNERO DEL USUARIO (GENERO)

Femenino

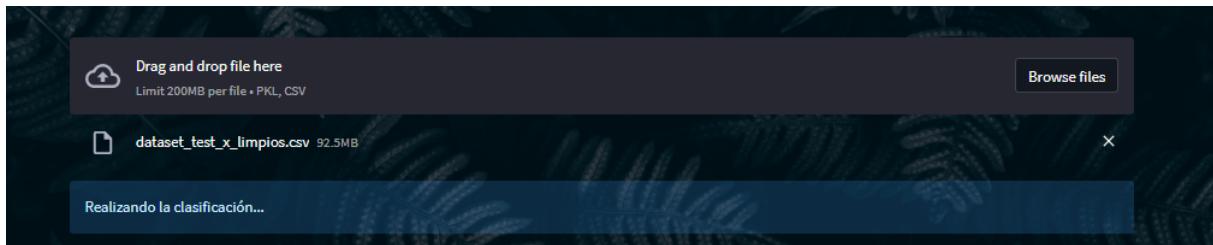
Masculino

PREDECIR

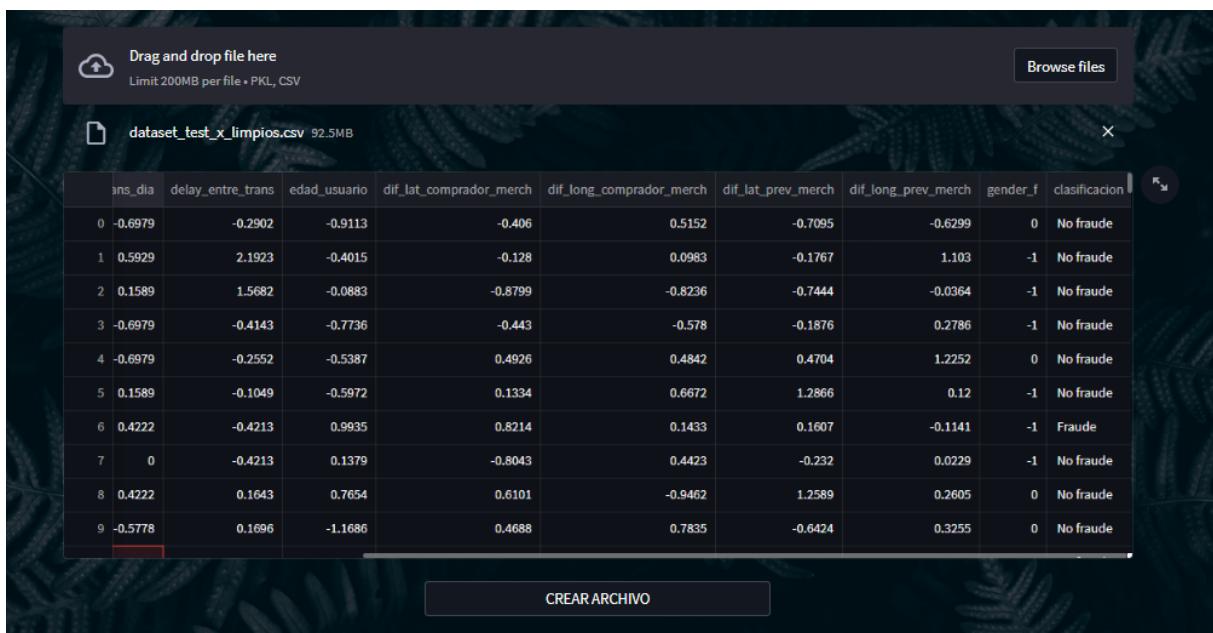
Cargar conjunto de datos



Procesamiento de conjunto de datos



Visualización de clasificación de archivo .csv



A screenshot of a CSV visualization interface. It displays a table of data with 10 rows and 11 columns. The columns are labeled: 'ans_dia', 'delay_entre_trans', 'edad_usuario', 'dif_lat_comprador_merc', 'dif_long_comprador_merc', 'dif_lat_prev_merc', 'dif_long_prev_merc', 'gender_f', and 'clasificacion'. The last column contains values like 'No fraude' and 'Fraude'. At the bottom right of the table area, there is a button labeled "CREAR ARCHIVO".

	ans_dia	delay_entre_trans	edad_usuario	dif_lat_comprador_merc	dif_long_comprador_merc	dif_lat_prev_merc	dif_long_prev_merc	gender_f	clasificacion
0	-0.6979	-0.2902	-0.9113	-0.406	0.5152	-0.7095	-0.6299	0	No fraude
1	0.5929	2.1923	-0.4015	-0.128	0.0983	-0.1767	1.103	-1	No fraude
2	0.1589	1.5682	-0.0883	-0.8799	-0.8236	-0.7444	-0.0364	-1	No fraude
3	-0.6979	-0.4143	-0.7736	-0.443	-0.578	-0.1876	0.2786	-1	No fraude
4	-0.6979	-0.2552	-0.5387	0.4926	0.4842	0.4704	1.2252	0	No fraude
5	0.1589	-0.1049	-0.5972	0.1334	0.6672	1.2866	0.12	-1	No fraude
6	0.4222	-0.4213	0.9935	0.8214	0.1433	0.1607	-0.1141	-1	Fraude
7	0	-0.4213	0.1379	-0.8043	0.4423	-0.232	0.0229	-1	No fraude
8	0.4222	0.1643	0.7654	0.6101	-0.9462	1.2589	0.2605	0	No fraude
9	-0.5778	0.1696	-1.1686	0.4688	0.7835	-0.6424	0.3255	0	No fraude

Crear archivo descargable

dataset_test_x_limpios.csv 92.5MB

ans_dia	delay_entre_trans	edad_usuario	dif_lat_comprador_merc	dif_long_comprador_merc	dif_lat_prev_merc	dif_long_prev_merc	gender_f	clasificacion	
4	-0.6979	-0.2552	-0.5387	0.4926	0.4842	0.4704	1.2252	0	No fraude
5	0.1589	-0.1049	-0.5972	0.1334	0.6672	1.2866	0.12	-1	No fraude
6	0.4222	-0.4213	0.9935	0.8214	0.1433	0.1607	-0.1141	-1	Fraude
7	0	-0.4213	0.1379	-0.8043	0.4423	-0.232	0.0229	-1	No fraude
8	0.4222	0.1643	0.7654	0.6101	-0.9462	1.2589	0.2605	0	No fraude
9	-0.5778	0.1696	-1.1686	0.4688	0.7835	-0.6424	0.3255	0	No fraude
10	0.5929	0.5262	0.2098	-0.1759	-0.0504	-0.6559	-0.089	0	No fraude
11	-0.5778	-0.4528	0.5963	0.7856	0.2168	0.4404	0.2472	0	No fraude
12	0.1589	-0.3252	-0.3628	-0.9834	-0.1615	0.109	-0.329	0	No fraude
13	0.4222	1.993	-0.4839	0.3818	0.6813	-0.3782	0.0768	0	No fraude

CREAR ARCHIVO

Creando descargable... esto puede llevar un tiempo.

dataset_test_x_limpios.csv 49.6MB

ans_dia	delay_entre_trans	edad_usuario	dif_lat_comprador_merc	dif_long_comprador_merc	dif_lat_prev_merc	dif_long_prev_merc	gender_f	clasificacion	
4	-0.6979	-0.2552	-0.5387	0.4926	0.4842	0.4704	1.2252	0	No fraude
5	0.1589	-0.1049	-0.5972	0.1334	0.6672	1.2866	0.12	-1	No fraude
6	0.4222	-0.4213	0.9935	0.8214	0.1433	0.1607	-0.1141	-1	Fraude
7	0	-0.4213	0.1379	-0.8043	0.4423	-0.232	0.0229	-1	No fraude
8	0.4222	0.1643	0.7654	0.6101	-0.9462	1.2589	0.2605	0	No fraude
9	-0.5778	0.1696	-1.1686	0.4688	0.7835	-0.6424	0.3255	0	No fraude
10	0.5929	0.5262	0.2098	-0.1759	-0.0504	-0.6559	-0.089	0	No fraude
11	-0.5778	-0.4528	0.5963	0.7856	0.2168	0.4404	0.2472	0	No fraude
12	0.1589	-0.3252	-0.3628	-0.9834	-0.1615	0.109	-0.329	0	No fraude
13	0.4222	1.993	-0.4839	0.3818	0.6813	-0.3782	0.0768	0	No fraude

CREAR ARCHIVO

DESCARGAR RESULTADOS

dataset_test_x_limpios.xlsx 49.6/49.6 MB. Faltan 0 s.

Mostrar todo X

Ejemplo de consultar datos manualmente

CONSULTAR DATOS MANUALMENTE:

AGREGAR DATOS MANUALMENTE

INGRESA EL MONTO (MONTO)

 INGRESA LA HORA DE LA TRANSACCIÓN (HORA)
 0 23

SELECCIONA LA CATEGORÍA (CATEGORÍA)

Compras en línea

INGRESA EL DÍA DE LA TRANSACCIÓN (DÍA)

Viernes

INGRESA LA EDAD DEL USUARIO (EDAD)

 SELECCIONA EL MES DE LA TRANSACCIÓN (MES)

SELECCIONA EL GÉNERO DEL USUARIO (GENERO)

Femenino Masculino

PREDECIR

Procesamiento de dato ingresado

CONSULTAR DATOS MANUALMENTE:

AGREGAR DATOS MANUALMENTE

INGRESA EL MONTO (MONTO)

 INGRESA LA HORA DE LA TRANSACCIÓN (HORA)
 0 23

SELECCIONA LA CATEGORÍA (CATEGORÍA)

Compras en línea

INGRESA EL DÍA DE LA TRANSACCIÓN (DÍA)

Viernes

INGRESA LA EDAD DEL USUARIO (EDAD)

 SELECCIONA EL MES DE LA TRANSACCIÓN (MES)

SELECCIONA EL GÉNERO DEL USUARIO (GENERO)

Femenino Masculino

PREDECIR

Procesando dato... esto puede llevar un tiempo.

Visualización de clasificación de consulta manual

CONSULTAR DATOS MANUALMENTE:

AGREGAR DATOS MANUALMENTE

INGRESA EL MONTO (MONTO)

SELECCIONA LA CATEGORÍA (CATEGORÍA)

Compras en línea

INGRESA LA EDAD DEL USUARIO (EDAD)

SELECCIONA EL GÉNERO DEL USUARIO (GENERO)

Femenino
 Masculino

INGRESA LA HORA DE LA TRANSACCIÓN (HORA)

0 9 23

INGRESA EL DÍA DE LA TRANSACCIÓN (DÍA)

Viernes

SELECCIONA EL MES DE LA TRANSACCIÓN (MES)

5

PREDECIR

La transacción analizada es: No fraude

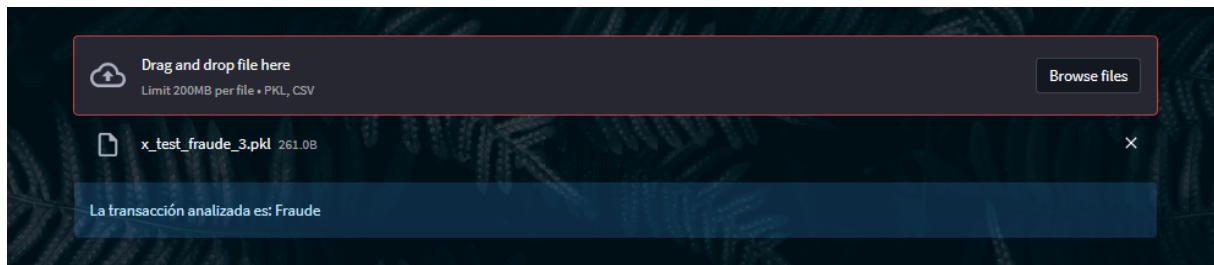


Cargar dato(fila) .pkl

Drag and drop file here
Limit 200MB per file • PKL, CSV

x_test_fraude_3.pkl 261.0B

La transacción analizada es: Fraude

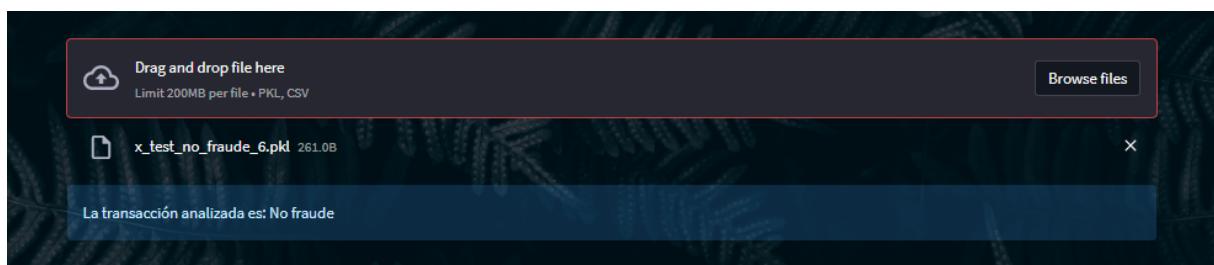


Visualización de clasificación archivo .pkl

Drag and drop file here
Limit 200MB per file • PKL, CSV

x_test_no_fraude_6.pkl 261.0B

La transacción analizada es: No fraude



Despliegue y conclusiones

Una vez logrado la implementación en el procesado de datos con sus debidas predicciones se realizaron diferentes pruebas para las diferentes funcionalidades de la aplicación. Posterior a esto se realizó el despliegue en sí por medio de la plataforma de **Streamlit**, proporcionándole el [git del proyecto](#).

En esta entrega se describe el proceso de implementación y despliegue del modelo de machine learning para la detección de fraude en tarjetas de crédito como servicio en la nube a través de una aplicación web interactiva construida con *Streamlit*. Además, se discutieron los desafíos y limitaciones encontrados durante el proceso de implementación y se explicó la estructura modular del desarrollo utilizado.

Por último, se ha incorporado el modelo de machine learning ya entrenado en la aplicación web utilizando la biblioteca de Python "pickle". En general, se ha logrado mejorar el flujo de trabajo y aumentar la eficiencia en el desarrollo de la aplicación web.

Link de la aplicación desplegada: <https://ml-ungs-deteccion-fraude.streamlit.app/>

Link de descarga de datos de prueba: [Descargar](#)

Lecciones aprendidas

Algunos aspectos que se quieren resaltar respecto al desarrollo de este pequeño proyecto es que si bien se pudo llevar a cabo lo deseado, hubo muchos aspectos que se hubiesen querido mejorar y no se pudieron por cuestiones de tiempo. Entre ellos está el tiempo de capacitación en nuevas tecnologías y temas, como lo es machine learning, que se desconocían. Por otro lado, también es que si bien se tiene una estructura general de lo planteado, había aspectos, como la plataforma sobre la cual se iba a implementar el modelo, que se dejaron hasta el final para su elección. Como último, se hubiese gustado poder realizar un correcto procesamiento de los datos para los archivos que se podían subir dando lugar a una mejor experiencia y uso para el usuario. Estas cuestiones son puntos que nos gustaría tener en cuenta para proyectos futuros.

Finalmente, queremos agradecer por la incorporación de nuevos temas para la ampliación de conocimientos y tecnologías dando la posibilidad de visualizar y entender, aunque sea un poco, herramientas que se utilizan cotidianamente y, muchas veces, pasan desapercibidas.

Bibliografía

- Martinez, Jose. "Guía rápida de IArtificial.net." IArtificial.net, 26 January 2021, https://www.iartificial.net/guia-rapida-iartificial-net/#Modelos_de_Machine_Learning. Accessed 12 March 2023.
- *AprendeIA con Ligdi Gonzalez*. (n.d.). YouTube. Retrieved March 12, 2023, from <https://www.youtube.com/@aprendeIA>
- *Python: Machine Learning*. (n.d.). Udemy. Retrieved March 12, 2023, from <https://www.udemy.com/course/python-machine-learning-desde-cero>
- Martinez, J. (2020, September 18). *Random Forest (Bosque Aleatorio): combinando árboles*. IArtificial.net. Retrieved March 12, 2023, from <https://www.iartificial.net/random-forest-bosque-aleatorio/>
- (n.d.). Inicio - Aprende IA. Retrieved March 12, 2023, from <https://aprendeia.com/>
- Martinez, J. (2020, September 19). *machine learning - Las 7 Fases del Proceso de Machine Learning*. IArtificial.net. Retrieved March 12, 2023, from <https://www.iartificial.net/fases-del-proceso-de-machine-learning/>
- Tanant, F. (n.d.). *Machine learning para detectar el fraude: Cómo usarlo*. SEON. Retrieved March 12, 2023, from <https://seon.io/es/recursos/machine-learning-para-detectar-fraude/>
- *What is Bagging?* (n.d.). IBM. Retrieved March 18, 2023, from <https://www.ibm.com/topics/bagging>
- Degracia, O. (2022, September 6). *El papel del Machine Learning en la detección de fraude*. Featurespace. Retrieved March 18, 2023, from <https://www.featurespace.com/es/newsroom/el-papel-del-machine-learning-en-la-deteccion-de-fraude/>
- *Interpretación de Modelos de Machine Learning*. (2019, April 30). Aprende Machine Learning. Retrieved March 18, 2023, from <https://www.aprendemachinelearning.com/interpretacion-de-modelos-de-machine-learning/>
- *Best techniques and metrics for Imbalanced Dataset*. (n.d.). Kaggle. Retrieved March 20, 2023, from <https://www.kaggle.com/code/marcinrutecki/best-techniques-and-metrics-for-imbalanced-dataset#2.-Metrics-for-imbalanced-data>
- *Taller técnico | ¿Cómo balancear un dataset de Python?* (2022, September 30). YouTube. Retrieved March 27, 2023, from <https://www.youtube.com/watch?v=WPYR-fLyP1A>

- (*Code*) *Capping outliers using the IQR method* | *Machine Learning*. (2020, September 25). YouTube. Retrieved March 27, 2023, from
<https://www.youtube.com/watch?v=ADY2co093vY>
- *Escalamiento, Normalización y Estandarización de Datos con Python - Ciencia de Datos*. (2021, September 27). YouTube. Retrieved March 27, 2023, from
<https://www.youtube.com/watch?v=-VuR14Qyl7E>
- (n.d.).
<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler>
- *Datos Anómalos (outliers) y Diagramas de Caja (BoxPlots) con la Prueba de Tukey en Python*. (2021, August 15). YouTube. Retrieved March 27, 2023, from
<https://www.youtube.com/watch?v=wsfV4AO8UFw>
- *Datos de Entrenamiento, Validación y Prueba: ¿Cómo crearlos y qué objetivos tienen? Machine Learning*. (2021, October 11). YouTube. Retrieved March 27, 2023, from <https://www.youtube.com/watch?v=vdYzm4xC7mc>
- *Feature-engine: A new open source Python package for feature engineering*. (n.d.). Sole from Train in Data. Retrieved March 27, 2023, from
<https://trainindata.medium.com/feature-engine-a-new-open-source-python-package-for-feature-engineering-29a0ab88ea7c>
- *Random Forest (Bosque Aleatorio) para Clasificación con Python*. (2021, November 23). YouTube. Retrieved March 27, 2023, from
https://www.youtube.com/watch?v=yOCJQLf_YFI
- *Random Forest Classifier Tutorial*. (n.d.). Kaggle. Retrieved March 27, 2023, from
<https://www.kaggle.com/code/prashant111/random-forest-classifier-tutorial#Random-Forest-Classifier-Tutorial-with-Python>
- *Random Forest python*. (n.d.). Cienciadedatos.net. Retrieved March 27, 2023, from
https://www.cienciadedatos.net/documentos/py08_random_forest_python.html
- Roepke, B., & guide, s. (2022, March 5). *5-10x Faster Hyperparameter Tuning with HalvingGridSearch*. Data Knows All. Retrieved March 27, 2023, from
<https://www.dataknowsall.com/hyperparameter.html>
- *sklearn.ensemble.RandomForestClassifier — scikit-learn 1.2.2 documentation*. (n.d.). Scikit-learn. Retrieved March 27, 2023, from
<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>
- *sklearn.model_selection.HalvingGridSearchCV — scikit-learn 1.2.2 documentation*. (n.d.). Scikit-learn. Retrieved March 27, 2023, from
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingGridSearchCV.html#
- *sklearn.model_selection.HalvingRandomSearchCV — scikit-learn 1.2.2 documentation*. (n.d.). Scikit-learn. Retrieved March 27, 2023, from
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingRandomSearchCV.html#

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingRandSearchCV.html

- *sklearn.model_selection.StratifiedKFold* — *scikit-learn 1.2.2 documentation*. (n.d.). Scikit-learn. Retrieved March 27, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html#sklearn.model_selection.StratifiedKFold
- *SMOTE (Synthetic Minority Oversampling Technique) for Handling Imbalanced Datasets*. (2019, May 8). YouTube. Retrieved March 27, 2023, from https://www.youtube.com/watch?v=U3X98xZ4_no
- *3.3. Metrics and scoring: quantifying the quality of predictions*. (n.d.). Scikit-learn. Retrieved March 27, 2023, from https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
- *2. Over-sampling — Version 0.10.1*. (n.d.). Imbalanced-Learn. Retrieved March 27, 2023, from https://imbalanced-learn.org/stable/over_sampling.html#smote-variants
- *User Guide — 1.3.0*. (n.d.). Feature-engine. Retrieved March 27, 2023, from https://feature-engine.trainindata.com/en/1.3.x/user_guide/index.html
- *User guide: contents — Version 0.10.1*. (n.d.). Imbalanced-Learn. Retrieved March 27, 2023, from https://imbalanced-learn.org/stable/user_guide.html
- *Using Power Transformers (Box-Cox & Yeo-Johnson) to make features Gaussian-like | Machine Learning*. (2020, July 24). YouTube. Retrieved March 27, 2023, from <https://www.youtube.com/watch?v=ev7wkRL8OUk>
- *YeoJohnsonTransformer — 1.3.0*. (n.d.). Feature-engine. Retrieved March 27, 2023, from https://feature-engine.trainindata.com/en/1.3.x/user_guide/transformation/YeoJohnsonTransformer.html
- (n.d.). Streamlit documentation. Retrieved April 2, 2023, from <https://docs.streamlit.io/>

