KEEPCODING
Tech School

PROYECTO FINAL DEL XI BOOTCAMP EN BIG DATA, AI & ML

PREDICCIÓN DEL FRAUDE EN TRANSACCIONES CON TARJETA DE CRÉDITO Y DÉBITO

Grupo: Los Gatos de Madriz

Katelyn Marie Miller
Víctor Rey García
Alfredo Sánchez Sánchez
Jesús García Velázquez
Najli Yaya Estévez

Madrid, 26 de Octubre de 2023

ÍNDICE

Objetivo

Para el Proyecto Final del Bootcamp de KeepCoding en Big Data, AI & ML hemos escogido crear un sistema de predicción de transacciones fraudulentas cuando se paga con una tarjeta de crédito/débito.

Consideramos que se trata de un problema a nivel mundial, que genera grandes pérdidas a las empresas y que también afecta a los consumidores y entidades bancarias.

Hoy en día, casi todo lo pagamos con tarjeta de crédito o débito y resulta muy molesto descubrir que nos han robado la tarjeta o que han intentado hacer compras con nuestros datos.

Además, el fraude inicia un proceso legal que puede alargarse en el tiempo, con denuncias y la necesidad de probar que no hemos sido nosotros los que realizamos esa compra.

El final de dicho proceso es muy variopinto, si bien es cierto que una parte del problema se soluciona por parte de las entidades bancarias con seguros que asumen parte de la pérdida hasta ciertas cantidades.

Bajo nuestro parecer, la mejor solución a esta cuestión es, como casi siempre, la prevención de estos delitos, es decir, **la predicción de transacciones que son fraudulentas** con el objetivo de no autorizarlas en el momento de realizarse, evitando el problema.

Por ello, el objetivo es **desarrollar mecanismos de detección de fraude basados en el Big Data y el Machine Learning en transacciones realizadas con tarjeta de crédito o débito.**

Definición del Dataset

Notebook de trabajo:
Jupyter Notebook: 1. Obtención dataset Final Completo.ipynb

El primer paso para poder resolver el problema es contar con un conjunto de datos amplio y que represente el problema.

Frente a la dificultad de encontrar datos reales, en gran parte por la Ley de Protección de Datos, hemos partido de varios datasets encontrados, decantándonos por un conjunto de datos publicado en Kaggle.

Tras un primer análisis, vimos que los datos contenidos tenían sentido y nos podían servir para nuestro propósito. Se trata de transacciones con tarjetas de crédito y débito en distintos puntos de Estados Unidos, a lo largo de los años 2019 y 2020.

Pero la información recogida en el dataset, pese a tener un conjunto de campos interesante, debe ser completada cruzándola con información complementaria que pueda ayudar a pronosticar el fraude.

Nos acercamos al problema con el ejercicio de pensar qué otros datasets podrían aportar información relevante en este problema. Tras un ejercicio de *brainstorming,* estos son los datos que aportamos al dataset original y las hipótesis que manejamos:

- Información sobre festividades de Estados Unidos. Es más probable que en fechas señaladas, de compras masivas, las transacciones fraudulentas sean más y queden “disimuladas” entre el caos de compras generalizado.
- Índices de criminalidad de cada estado. Queremos analizar la existencia de una relación entre el índice de criminalidad del lugar de la transacción y la probabilidad de que la transacción sea fraude.
- Valores económicos. De todos los valores posibles escogemos los valores de cierre de la Bolsa y el valor del cambio euro-dólar, pues ambos reflejan la fortaleza de la economía estadounidense. Queremos observar si en épocas económicamente convulsas, el fraude se dispara.
- Tasa de desempleo de cada Estado. ¿Son las zonas con alta tasa de desempleo proclives a tener un mayor porcentaje de robo o suplantación de identidades de propietarios de tarjetas?

Arquitectura

Notebook de trabajo:
Google Colab Notebook: 2. Arquitectura.ipynb
Google Colab Notebook: 2.1 Procesamiento Datos Preproducción.ipynb

En la parte de arquitectura para resolver el problema, tendríamos una primera parte donde el dataset sería nutrido constantemente desde la base de datos del banco, obteniendo datos fidedignos en tiempo real.

En el trabajo actual, como ya hemos comentado, por motivos de inviabilidad de obtener datos reales, usamos un conjunto de datos de Kaggle enriquecido con festividades nacionales, tasas de desempleo estatal, tasas de criminalidad por estado y valores de la bolsa.

Almacenamos el dataset resultante en Google Drive para su uso posterior.

La segunda parte del proyecto de arquitectura comienza con la descarga de cuadernos y archivos CSV desde GitHub, que se almacenan en el depósito de Google Cloud Storage.

Estos cuadernos y archivos están disponibles para que cualquier persona interesada los descargue y ejecute el modelo por sí misma. Los cuadernos están divididos entre cuatro notebooks (creación del dataset final, análisis exploratorio, preprocesamiento, y arquitectura en la nube con Google Cloud).

Los aspectos destacados de la última etapa de arquitectura son:

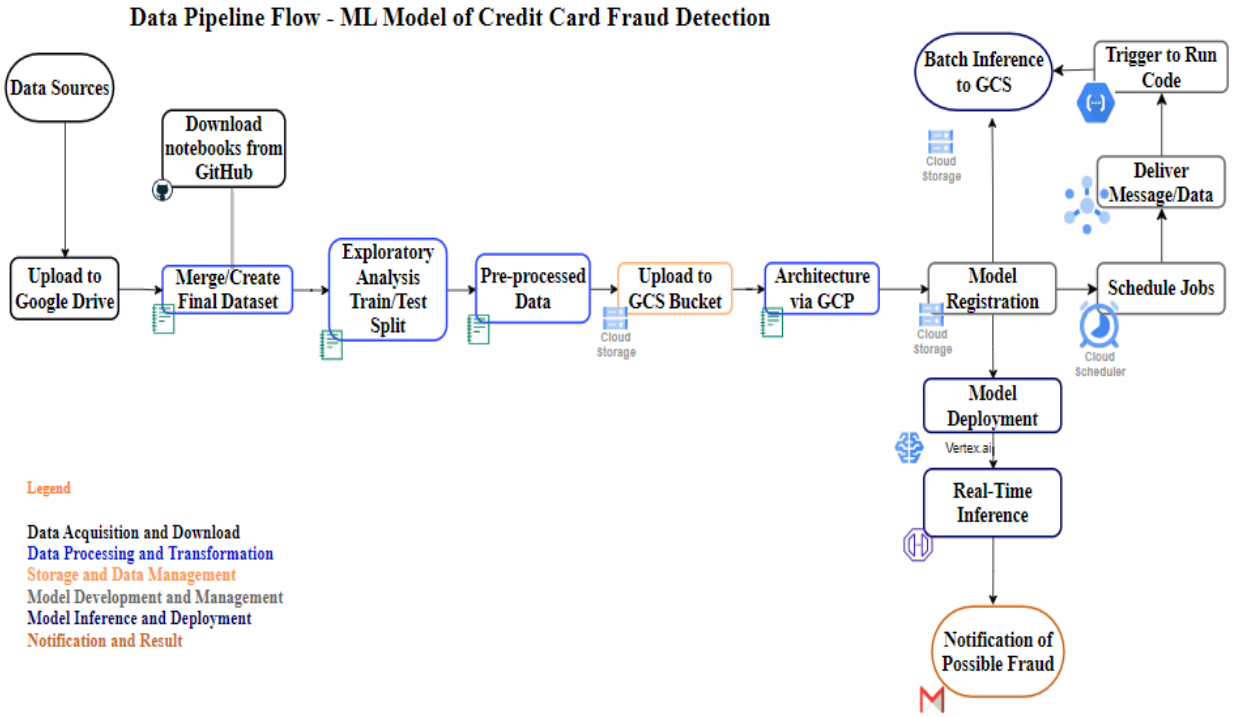
- Preparación para la implementación en Google Cloud.
- Implementación con Vertex AI.
- Creación del endpoint para real-time inference.
- Notificaciones por correo electrónico en caso de transacciones fraudulentas.

Finalmente, se realiza la inferencia por lotes (*batch inference*) con Google AI Platform para procesar conjuntos de datos y almacenar los resultados en el mismo depósito. Conseguimos automatizar este último paso de inferencia por lotes con herramientas de Cloud.

Este proceso global garantiza una detección eficiente y escalable de fraudes en transacciones de tarjetas de crédito, al tiempo que se mantiene la integridad y seguridad de los datos.

Por último, hemos tomado la decisión de separar las últimas 10.000 filas del dataset original (ordenado cronológicamente) para poder simular predicciones evocando la puesta en escena del modelo.

Aquí el esquema con los pasos más detallados.


Data Pipeline Flow - ML Model of Credit Card Fraud Detection
Data Sources
Download notebooks from GitHub
Upload to Google Drive
Merge/Create Final Dataset
Exploratory Analysis Train/Test Split
Pre-processed Data
Upload to GCS Bucket
Cloud Storage
Architecture via GCP
Model Registration
Cloud Storage
Batch Inference to GCS
Cloud Storage
Trigger to Run Code
Deliver Message/Data
Schedule Jobs
Cloud Scheduler
Model Deployment
Vertex.ai
Real-Time Inference
Notification of Possible Fraud
Legend
Data Acquisition and Download
Data Processing and Transformation
Storage and Data Management
Model Development and Management
Model Inference and Deployment
Notification and Result

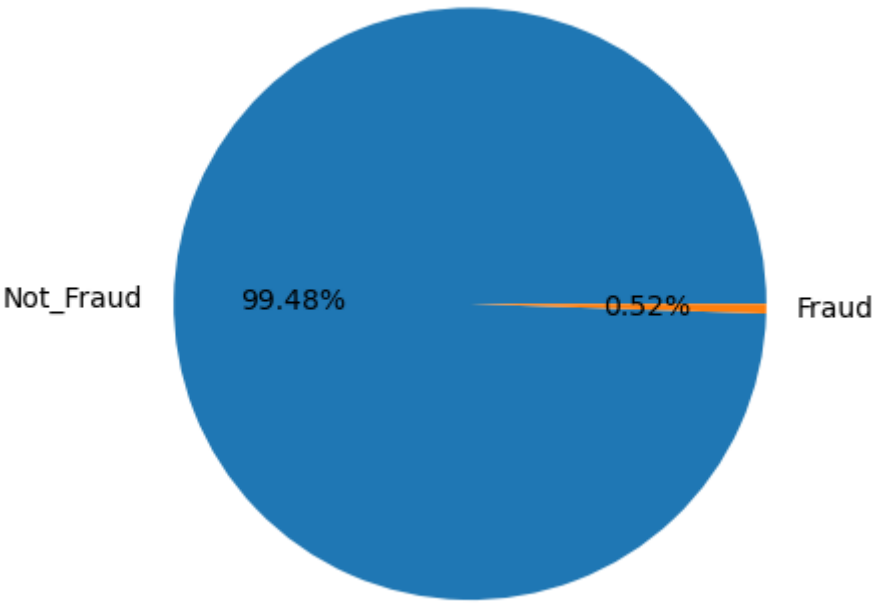
Análisis Exploratorio

Notebook de trabajo:
Jupyter Notebook: 3. Análisis exploratorio.ipynb

El primer paso que hacemos es ver los *data types* de cada variable, sobre todo para identificar variables no numéricas que debamos transformar.

Después, evaluamos si tenemos valores nulos en alguna variable, cosa que afortunadamente no nos sucede.

El problema al que nos enfrentamos es de tipo **clasificación binaria**: fraude / no fraude. Es importante evaluar el balance (proporción) de cada clase en el conjunto de datos que tenemos. Hacemos este cálculo y observamos el gran desbalance que existe hacia la clase "no fraude":



El motivo del paso siguiente en el que separamos las últimas 10.000 filas del dataset ordenado por fecha ascendente es para usarlas como una fase de "validación" del modelo, de forma que le lleguen transacciones que suceden en un futuro temporal con respecto a los tiempos con los que se ha entrenado. Lo hacemos así porque no tenemos manera de conseguir datos reales o similares y queremos probar el modelo de esa forma.

Ese csv se llama *last_10000_rows.csv* y se puede encontrar en la carpeta de *data*, a la que hacemos referencia en el Anexo, con los demás ficheros csv.

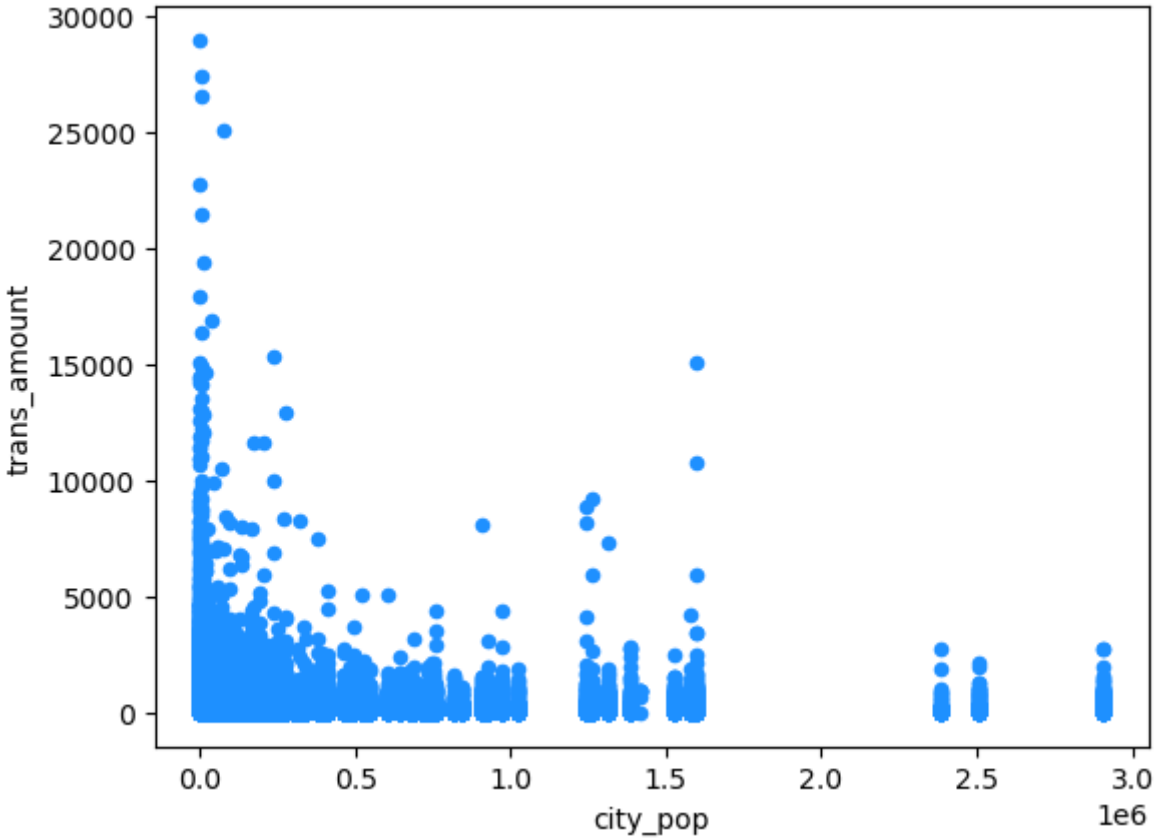
Ahora ya estamos listos para separar nuestro dataset en los conjuntos de *train* y *test*. Hacemos una partición 80-20 (80% de los datos en train y 20% de los datos en test) y estratificamos con respecto a la variable objetivo, tal y como se recomienda en problemas desbalanceados como este.

Estratificando nos aseguramos tener muestras de las dos categorías en ambos conjuntos (aunque de la categoría “fraude”, dada la frecuencia absoluta de la misma, habrá muy poquitas).

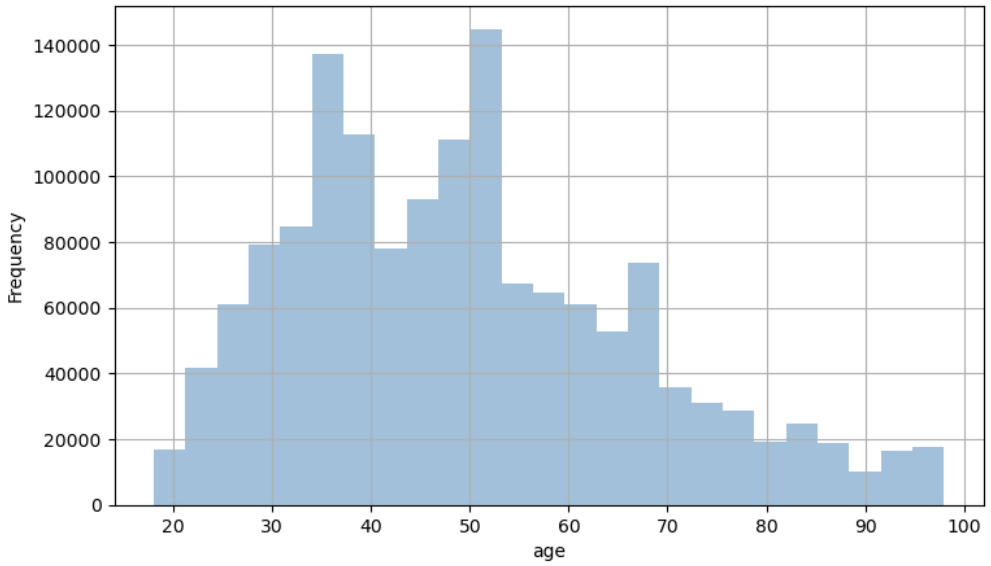
Pasamos a representar con histogramas algunas de las variables numéricas que pensamos más relevantes: la cantidad de la transacción, los meses y las horas. También se observa un aumento de las transacciones en el mes de diciembre, probablemente debido a la cantidad de festivos que hay en ese mes.

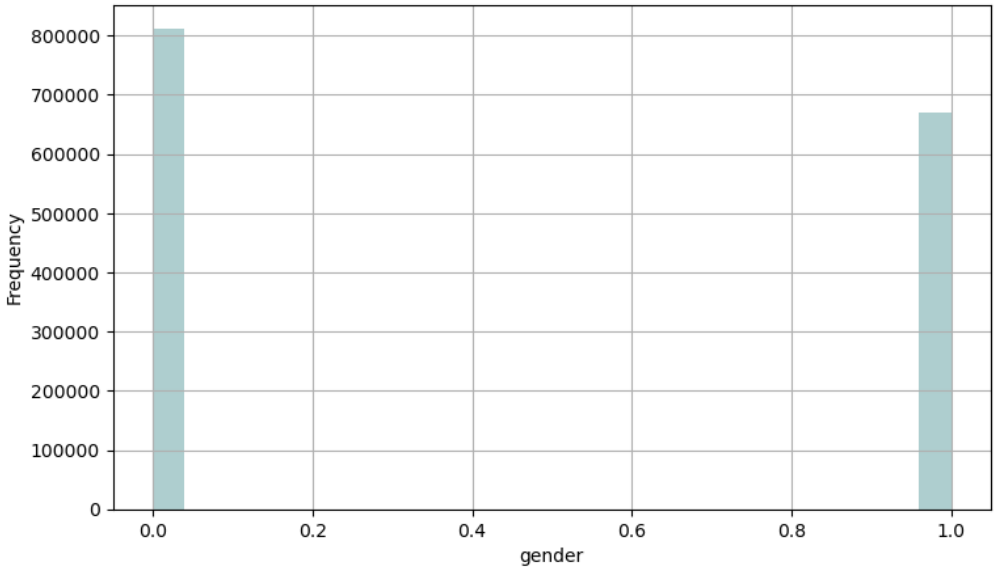
Con respecto a las horas de mayor volumen de fraude, está bastante repartido, aunque el mediodía y la noche son los momentos con mayor frecuencia de fraude.

Pintamos otros gráficos, por ejemplo un *scatter plot* de la población de la ciudad y la cantidad de las transacciones:



Queremos ir más allá en el análisis, así que no nos queda más remedio que empezar a aplicar encoders para transformar las variables categóricas a numéricas, como por ejemplo en el caso del género. Con esto podemos pintar los siguientes histogramas:





Después, vamos a ver cuántas "categorías" diferentes tenemos en cada variable, para hacernos una idea de la dimensionalidad a codificar en valor numérico:

```
In [23]: df.select_dtypes(include=['object']).describe()
```

Out[23]:

	trans_date_trans_time	category	city	state	job	state_abbr	date	day_of_week	time
count	1481915	1481915	1481915	1481915	1481915	1481915	1481915	1481915	1481915
unique	1460810	14	906	51	497	51	730	7	86400
top	2020-12-13 17:53:47	gas_transport	Birmingham	TX	Film/video editor	TX	2020-11-30	Monday	23:33:40
freq	4	150564	6444	108167	11134	108167	5190	295376	47

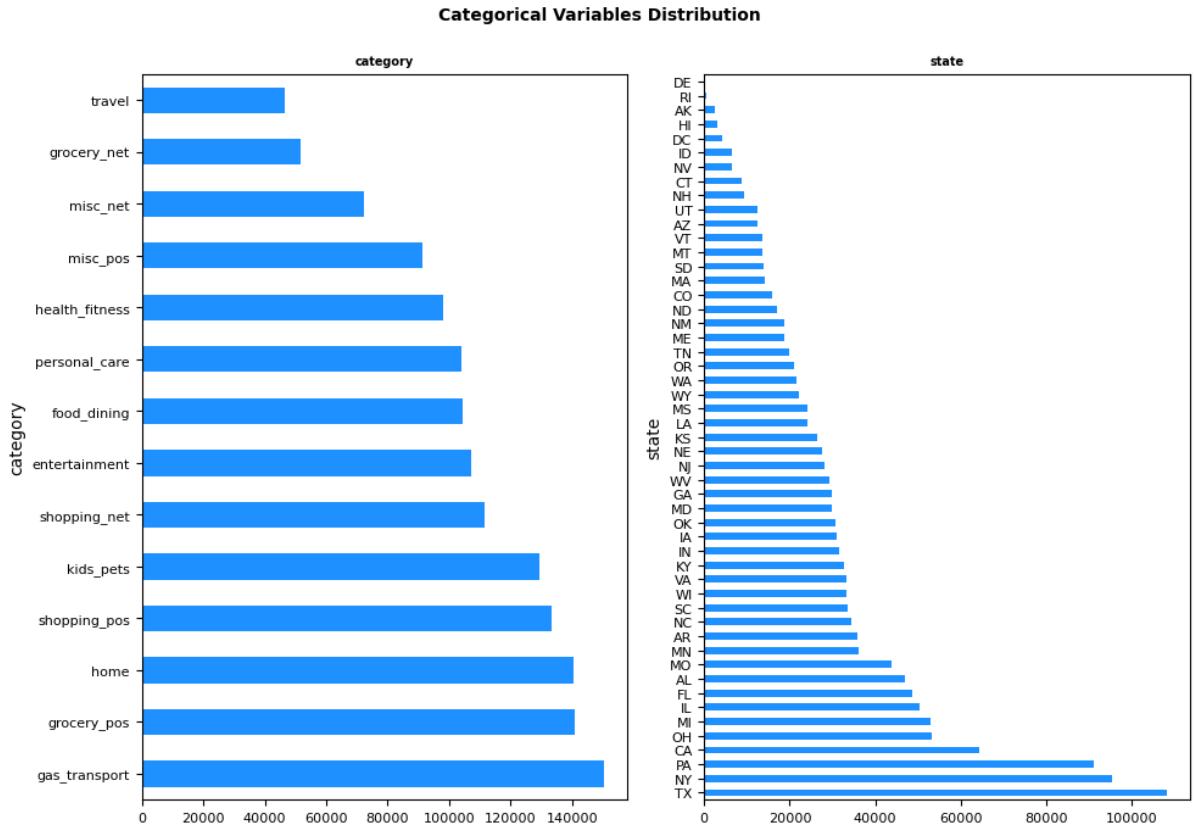
De las demás, estas son que nos interesan especialmente:

```
In [24]: print(f'Merchant Categories: ' , len(df['category'].unique()))
         print(f'City Categories: ' , len(df['city'].unique()))
         print(f'State Categories: ' , len(df['state'].unique()))
         print(f'Job Categories: ' , len(df['job'].unique()))

         Merchant Categories:  14
         City Categories:  906
         State Categories:  51
         Job Categories:  497
```

Hay bastante diferencia entre las categorías de comercios y las de oficios/trabajos. Por supuesto, era de esperar que tuviéramos amplia variedad de ciudades y estados.

Vamos a graficar las variables de *city* y *state*, ya que las otras tienen tanta variabilidad en los valores posibles que el gráfico no aportaría información. Pintamos *city* y *state*:

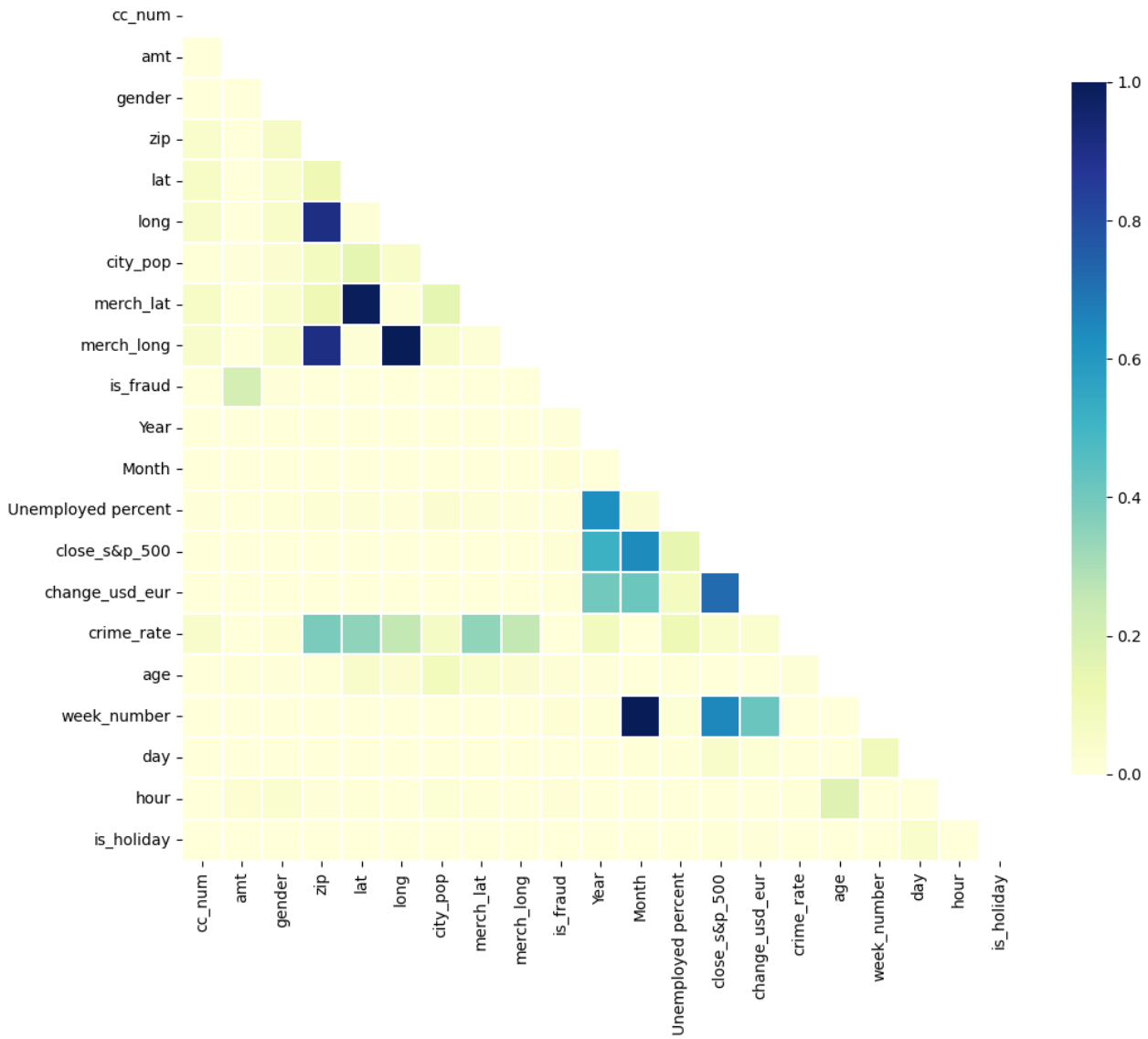


Vemos que la mayor parte de las transacciones se han hecho en gasolineras, tiendas de alimentación y de artículos del hogar, aunque todas las categorías tienen gran volumen transaccional.

Respecto a los estados, Texas tiene muchísima más frecuencia de fraude, seguido por Nueva York y Pennsylvania.

Con esta operación, hemos comprobado que en el conjunto de *train* tenemos los 50 estados miembros de EE.UU. (más el Distrito de Columbia). Según el tipo de variable, vamos a tener que usar *encoders* distintos.

Antes de codificar variables categóricas, vamos a echarle un vistazo a la matriz de correlación entre variables numéricas:

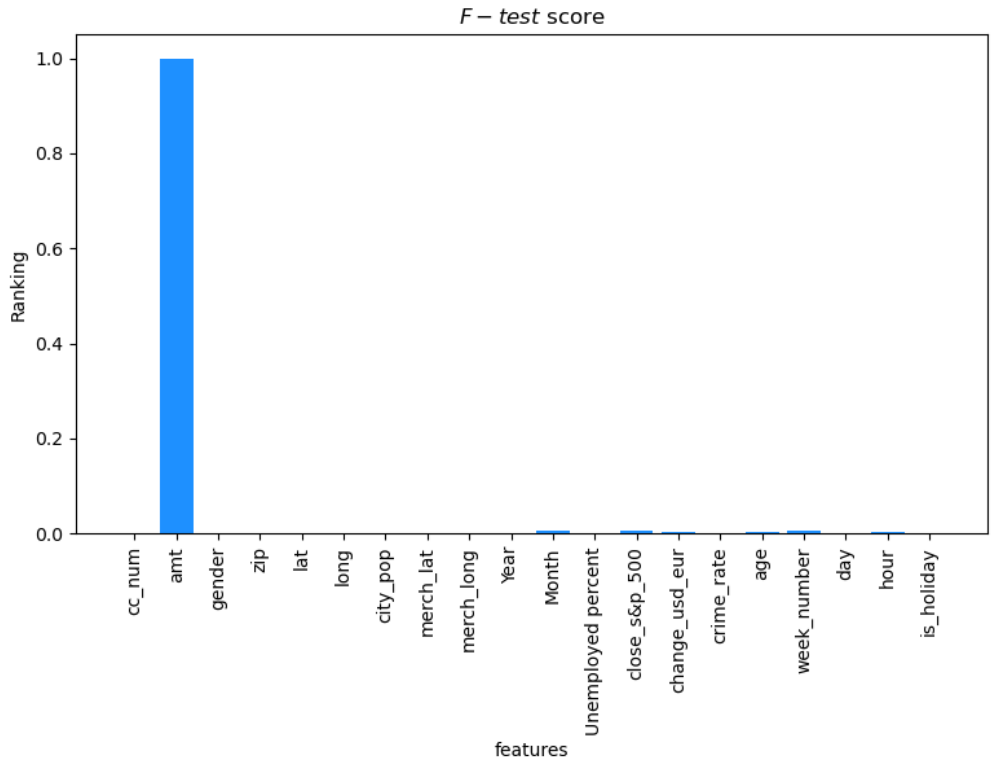


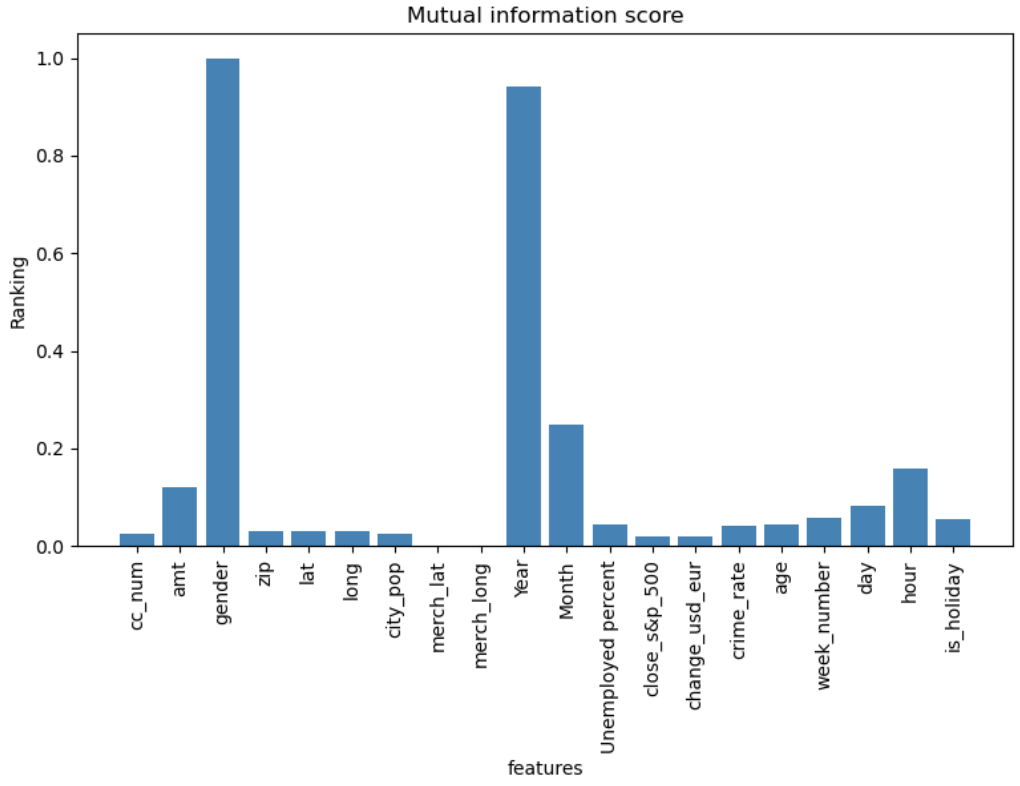
No se ven apenas correlaciones especialmente claras, más allá las que relacionan el valor del dólar-euro según el mes o año, las latitudes y longitudes, la tasa de criminalidad según la ubicación y el año.

Se ve una ligeramente superior correlación entre el fraude/no fraude y la cantidad de la transacción.

Por último, queremos evaluar cuáles son las *features* más relevantes con respecto a la variable objetivo. Para ello usamos las herramientas de la librería *sklearn* ***f_classif*** y ***mutual_info_classif***.

Este es el resultado:





Lo cierto es que nos esperábamos un *f-test score* algo más distribuido, pero este resultado, en el que parece que todo recae sobre *amount*, nos da para pensar que el dataset puede ser “fabricado”/”sintético”. Es una pena, pues desde el principio tratamos de encontrar un

dataset real, pero no hay forma de saberlo hasta hacer un análisis muy concienzudo. Se ha intentado encontrar otro dataset real, pero no ha sido factible.

Aún así, nuestro objetivo no cambia y seguiremos tratando los datos como reales y trabajando para resolver el problema. De nuevo hemos tratado de buscar alternativa, pero estos datos son los mejores que hemos encontrado para resolver este problema.

Una vez que hemos hecho el análisis exploratorio, tenemos bastantes claros los pasos que vamos a ejecutar en la etapa siguiente: el Preprocesamiento.

Preprocesamiento

Notebook de trabajo:
Jupyter Notebook: 4. Preprocesamiento.ipynb

Comenzamos el Notebook cargando el csv correspondiente a los datos de *train* (recordemos que los datos de *test* están guardados en un "cofre" y no podemos abrirlo ni mirarlos para que el proceso sea limpio y no tomemos decisiones condicionadas).

Tras unas comprobaciones iniciales de que la importación ha ido bien, lo primero que observamos es que tenemos una variable repetida: *state* y *state_abbr*. Eliminamos *state_abbr* del dataset.

Ya habíamos comentado que las variables categóricas nos iban a suponer un reto, porque no podíamos aplicar el mismo razonamiento para pasarlas a datos numéricos en todos los casos. Empezamos por los más sencillos:

1. *day_of_week* → mapeamos los días de la semana con un índice en la semana, que va del 0 (lunes) a 6 (domingo).
2. Eliminamos variables que queríamos desagregar y así hicimos (*trans_date_trans_time* y *time*)
3. Codificamos género con un *LabelEncoder* → asumimos que en *train* y en *test* vamos a tener muestras de "hombres" y "mujeres" y que al modelo no entrarían en un futuro nuevos códigos de esta variable.
4. Comercios → para la variable *merchants* usaremos *one-hot-encoding*, ya que tampoco nos va a aumentar mucho la dimensionalidad. Presuponemos que estas categorías son fijas y vienen dadas, por ejemplo, de una BBDD del Banco/Tarjeta con los tipos de comercios estándar. Aunque, para evitar problemas al evaluar el conjunto de *test* o en fase de producción, activamos la opción *handle_unknown* = *ignore* en el *OneHotEncoder*, de forma que **cuando el modelo se encuentre una categoría desconocida durante la transformación, las columnas resultantes codificadas en *one-hot* para esta característica serán todos ceros**. En la transformación inversa, una categoría desconocida se denotará como *None*
5. Para el caso de las profesiones, tenemos muchísimos valores distintos y aquí es muy probable que aparezcan nuevos en el conjunto de *test* en producción. Optamos por elegir un target encoder, que establece una relación matemática entre la variable categórica y la variable objetivo.
6. Ahora nos queda pasar a variable numérica los 51 estados. De igual manera que hemos razonado anteriormente con las categorías de *merchants*, los estados de Estados Unidos se agrupan en una variable con "categorías fijas", en el sentido de que esta organización del país está establecida de esta manera por leyes y un cambio en los estados podría darse, pero después de un largo proceso de formalización. Hemos comprobado que en *train* tenemos 51 valores distintos de estados, así que **podemos asumir que no tendremos estados nuevos en el conjunto de *test* ni cuando el sistema entre en producción.**
7. Probamos a aplicar *dummy encoding* (la única opción que veíamos viable) para la

variable *city*, pero la dimensionalidad aumentaba tanto que el dataset se volvía inmanejable. Hemos decidido eliminar esta variable, mete mucho ruido y tenemos información suficiente respecto a la ubicación.

8. Con la variable del código postal *zip* existe un caso particular: aparentemente se trata de una variable numérica, pero los modelos de ML no van a interpretar correctamente la información y se van a limitar a definir un "gradiente" según el valor del código postal, es decir, lo van a interpretar de forma cuantitativa. Por eso y porque tenemos información numérica real de las localizaciones, así como información útil de las ciudades como el número de habitantes, decidimos eliminar esta variable.
9. Con la variable del número de tarjeta asociado a la transacción *cc_num*, nos pasa lo mismo que con *zip*: pese a ser un número, su magnitud no tiene correspondencia con una realidad cuantitativa real, se trata al fin y al cabo de un dato personal, hay que considerarlo igual que el nombre o el apellido. Por todo esto, la eliminamos.
10. Cambiamos a valor numérico también la variable booleana “is_holiday”, por si acaso los modelos no la permiten.

También queremos crear variables derivadas de las que tenemos, porque creemos que pueden aportar información adicional muy interesante.

Por tanto, tras comentarlo, hemos decidido crear estas dos variables adicionales:

- *merch_home_distance* → Distancia entre el domicilio del propietario (latitud, longitud) y la ubicación (latitud, longitud) del comercio en donde se hizo la compra con tarjeta.
- *is_online_shopping* → Las categorías con la palabra clave *_net* son de compras realizadas por internet (*category_grocery_net*, *category_shopping_net*, *category_misc_net*, etc), en estos casos la distancia relativa entre el domicilio del propietario de la tarjeta y la ubicación del comercio no tiene sentido. Creamos esta variable adicional para aportar más información al modelo ML.

Una vez que tenemos claros los pasos a realizar en la etapa de preprocesamiento, hemos creado una función, *dataset_processing( ),* que contiene todos estos pasos de forma que el preprocesamiento del conjunto de *test* (y de cualquier conjunto de datos a futuro) sea limpio, sencillo y ordenado. Esto nos previene, sobre todo, de tener errores en el código y también da versatilidad en el futuro si se necesita cambiar cosas en el preprocesamiento.

Modelos Machine Learning

Notebook de trabajo:
Jupyter Notebook: 5.1 Búsqueda Modelos Machine Learning.ipynb
Jupyter Notebook: 5.2 Modelo Machine Learning.ipynb

A la hora de trabajar el problema desde el punto de vista del Machine Learning, tras el trabajo anterior partimos de una situación en la cual **todas las variables son numéricas**.

Preparación del conjunto de train

Nos centramos en el dataset de train, dejando el dataset de test aparte.

El primer problema a solucionar es un claro desbalanceo en las muestras para cada categoría en función de la variable objetivo.

Existen varias estrategias para solucionar el desbalanceo (undersampling, oversampling, SMOTE…), pero por el problema que estamos tratando de modelizar nos decantamos por **undersampling**, es decir, por reducir el número de muestras de la categoría dominante manteniendo las muestras de las categorías minoritarias.

A continuación debemos elegir qué variables deberían ser normalizadas. Por el acercamiento al dataset en preprocesamiento sabemos que varias variables ya se encuentran en el rango de 0 a 1 o provienen de un encoder, por lo que no deben ser normalizadas, pero otro grupo debe ser normalizado.

Tras la normalización de las variables que lo requieren podemos empezar el modelado, pero antes aplicamos al conjunto de test todos los cambios realizados al conjunto de train.

Modelado

Nos enfrentamos al siguiente problema: predecir con éxito si una transacción realizada con tarjeta de crédito es fraude o no.

Para encontrar un modelo de Machine Learning que resuelva de forma satisfactoria el problema vamos a probar varias posibilidades.

Regresión Logística

Empezamos por una **regresión logística** sencilla, que vamos aumentando en complejidad para analizar si es el mejor enfoque para este problema. Se prueba con *grid search* y *cross validation* buscando optimizar el parámetro *“c”*.

Es importante destacar que se aplica un balanceo al peso de la clase para penalizar las clases mayoritarias y permitir a clases secundarias mostrar su relevancia en la detección del fraude (*class_weight = ‘balanced’*).

En el caso de *grid search* usamos como *scoring* la curva ROC, recomendable en problemas desbalanceados.

Los resultados no son malos, pero muy mejorables. El modelo trabaja muy bien en cuanto a falsos negativos, con una tasa muy baja de fraudes no detectados. Sin embargo el porcentaje de no fraudes que son marcados como fraude es muy elevado, superando el 3%.

Por ello, debemos trabajar para afinar este modelo. En el proceso hemos graficado las mejoras en métricas de fiabilidad del modelo para cada *"c"*, por lo que probamos con valores de *"c"* diferentes que parecen tener un efecto en métricas similar, pero las altas tasas de falsos positivos nos indican que este modelo no es el idóneo.

Decision Tree Classifier

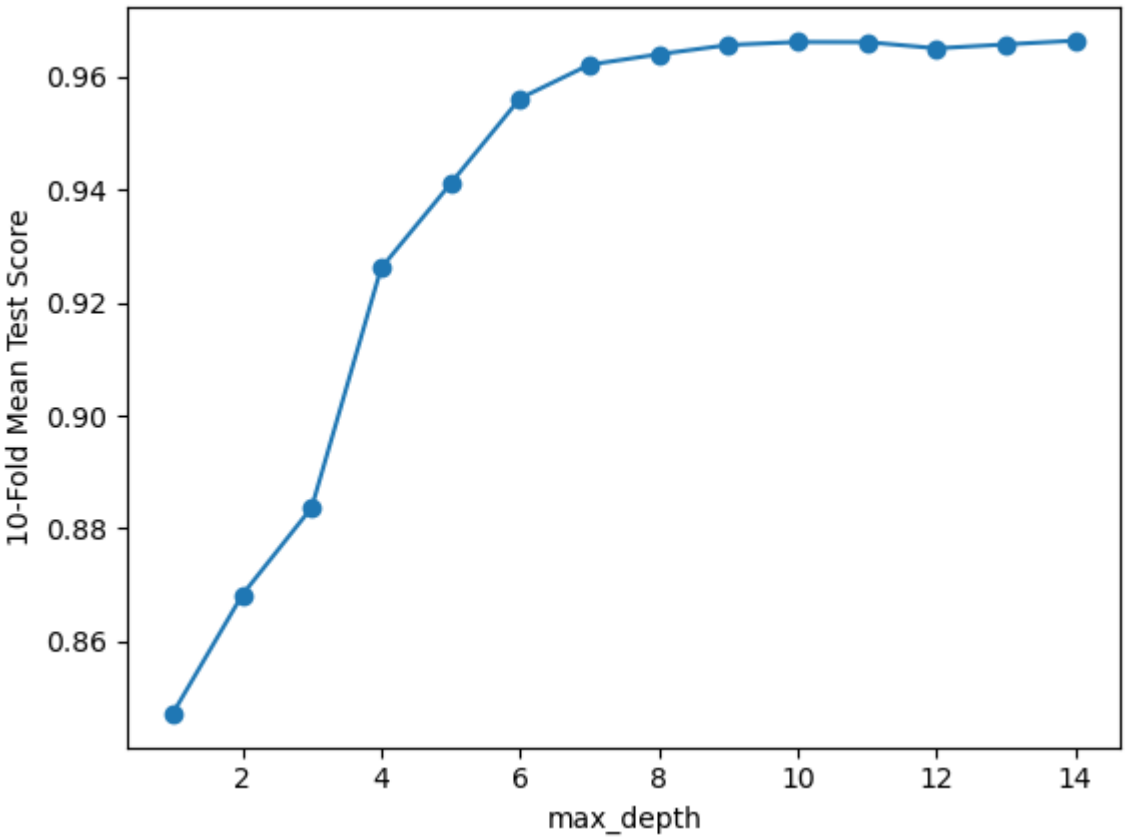
Pasamos a usar un modelo basado en un clasificador por árboles de decisión.

Empezamos configurando el *Decision Tree Classifier* con *Grid Search* y probamos varios puntos de profundidad del árbol, optimizando sobre la métrica *"f1-score"*.

De nuevo, es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado, combinando la precisión y el recall, para obtener un valor mucho más objetivo.

Los resultados son mejores que la regresión logística pero aún no son óptimos, seguimos teniendo un alto número de falsos positivos.

Al graficar la evolución de la curva *Grid Search* se ve que la mejora a partir de una profundidad de 10 ramificaciones apenas varía en pro de mejorar el modelo, aumentando la complejidad sin apenas mejorar el resultado.



Al probar con una profundidad de 10 ramificaciones el resultado no es malo, pero los falsos positivos se disparan, lo cual no es óptimo.

Es cierto que puede parecer que marcar como fraude transacciones que no son fraude es mejor que no detectar auténticos fraudes, pero a nivel de negocio no es interesante solucionar el problema por la vía de un exceso de control sobre las transacciones, pues los usuarios no aceptarían esta medida al ralentizar o incluso cancelar sus transacciones.

Probamos otro enfoque, pasamos a no limitar las ramificaciones del árbol. Al entrenar el modelo expandiendo el árbol al máximo los resultados son buenos, pero no mejoran lo obtenido, por lo que se descarta también este modelo por aumentar mucho la complejidad y, con probabilidad, llevar a un *overfitting* en situaciones de trabajo con datos en tiempo real (nuevas entradas de información).

Por último probamos con *Grid Search Cross - Validation* limitando todos los parámetros de definición del árbol, limitando profundidad, *split* mínimo de muestras y número mínimo de muestras por hoja del árbol.

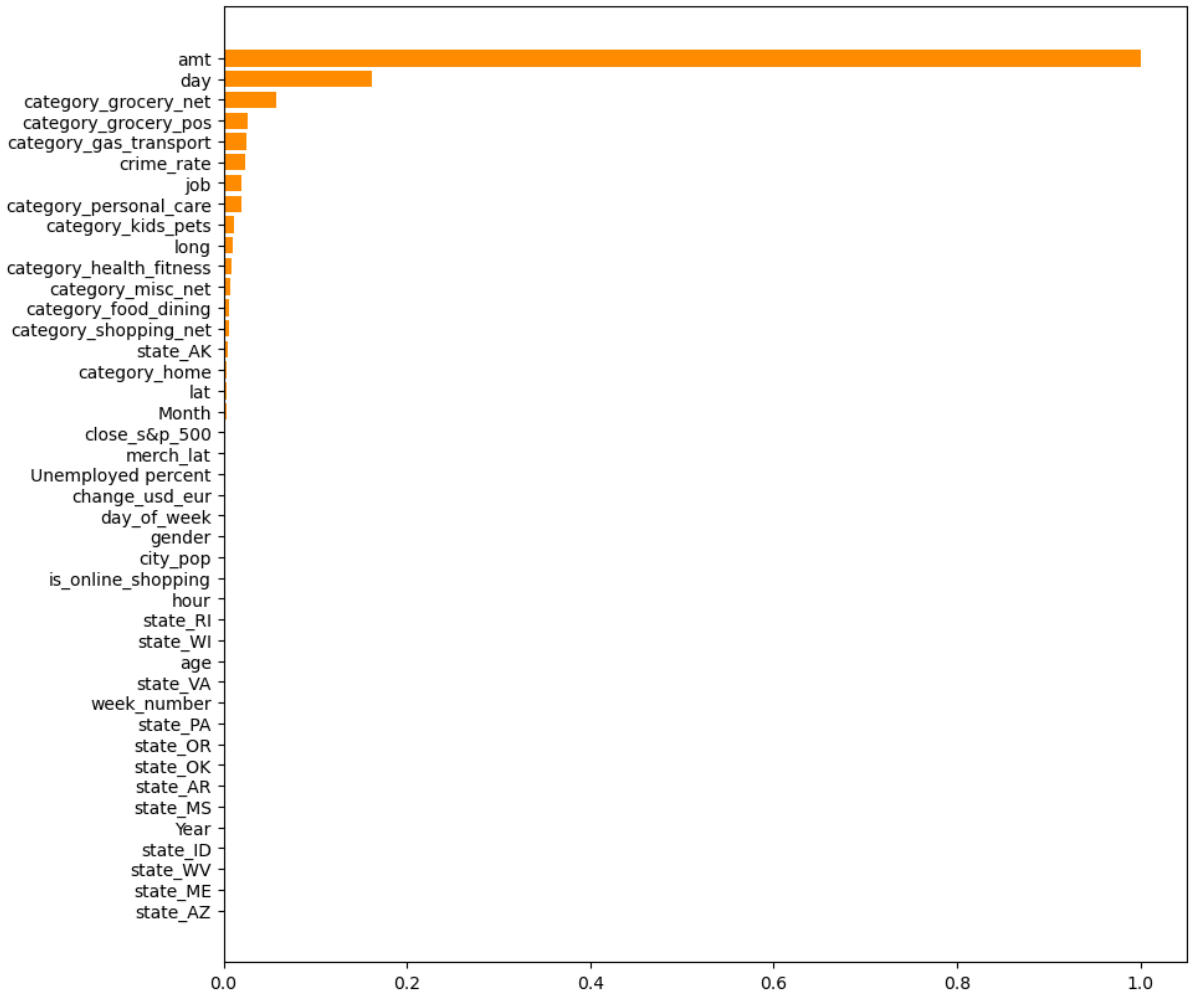
El entrenamiento lleva más tiempo pero los resultados son los mejores obtenidos de todas las aproximaciones al problema, así que nos quedamos con éste modelo.

Aquí están los resultados del modelo:

- Accuracy del modelo DecisionTree escogido: 0.9734856766510382
- Precisión del modelo DecisionTree escogido: 0.15952031748770598

- Recall del modelo DecisionTree escogido: 0.9580310880829016
- F1-Score del modelo DecisionTree escogido: 0.27350048073367356
- Verdaderos Positivos DecisionTree: 1849
- Falsos Negativos DecisionTree: 81
- Falsos Positivos DecisionTree: 9742
- Verdaderos Negativos DecisionTree: 358807

Y la gráfica de relevancia de cada variable en la predicción:



En el notebook 5.2 simplemente hacemos un resumen para entrenar y sacar métricas del modelo elegido en el notebook 5.1.

Visualización con Tableau

Archivo de trabajo:
Dataset: combined_dataset.csv
Archivo de Tableau: proyecto_final_los_gatos_de_madriz.twbx

El acercamiento al problema desde el punto de vista de la visualización sigue un patrón más clásico. Frente a lo anteriormente expuesto, buscando relevancia de ciertas variables pero, principalmente, dejando a los diferentes modelos que interpreten y marquen la relevancia, en un acercamiento mediante visualización somos nosotros los encargados de indagar y sacar conclusiones.

Por ello, en el proceso trabajado la parte de visualización se ha realizado por parte de una persona que no ha trabajado en la parte de modelado, no tanto porque lo segundo afecte a lo primero, sino más bien porque el acercamiento al problema desde la visualización construye una comprensión del problema que puede ser trasladada al modelado, arriesgando la capacidad del Machine Learning y el Deep Learning de encontrar patrones no esperados.

A la hora de visualizar, ya sea con una herramienta como Tableau o con cualquier otra de similares características, el objetivo es ir mezclando la variable objetivo con diferentes variables para intentar dilucidar cuáles de ellas tienen relevancia para poder predecir el resultado.

El dataset utilizado en origen tiene una serie de variables interesantes. Las variables registradas se han enriquecido con otras variables de posible utilidad. Tenemos variables relacionadas con la localización del usuario que realiza la transacción, la localización del comercio que recibe la transacción, fecha, cantidad de la transacción, edad del usuario…

Por otro lado, se ha buscado cruzar los datos con variables económicas como evolución de la bolsa, cambio de la moneda oficial (dólar), índices de paro, índices de criminalidad o referencias del día de la transacción (si es festivo, por ejemplo).

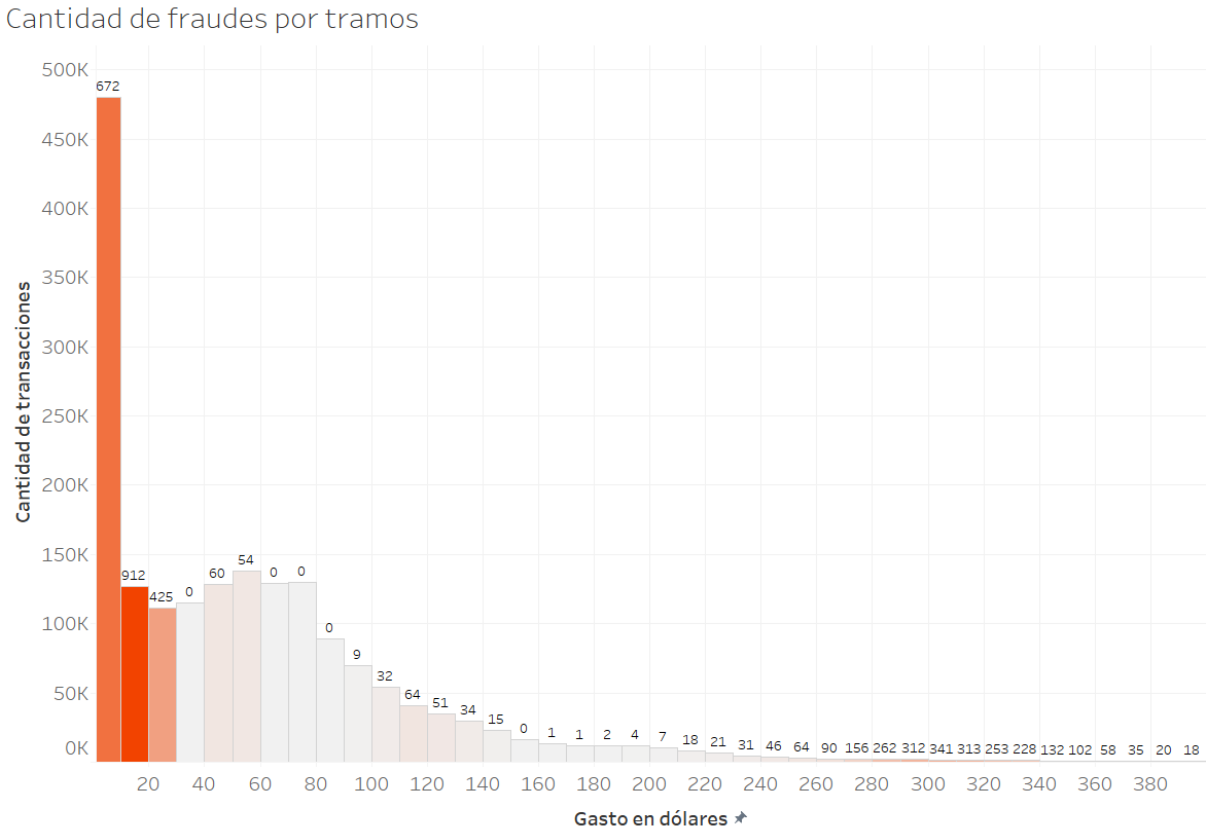
Por último, hay un par de variables que se han añadido al dataset en apartados anteriores del proceso (preprocesamiento) que no aparecen en este punto, pues hemos partido del de un dataset anterior, con la intención de no condicionar los resultados, como bien se ha dicho antes. Dichas variables son *merch_home_distance* y *is_online_shopping*.

Con todo lo anterior comenzamos el acercamiento al problema.

Tratamos de detectar patrones que pronostiquen la probabilidad de que una transacción sea fraudulenta en relación a la distribución de otras variables.

Tras probar con diferentes variables se recogen los datos más relevantes.

Empecemos mostrando la cantidad absoluta de transacciones:

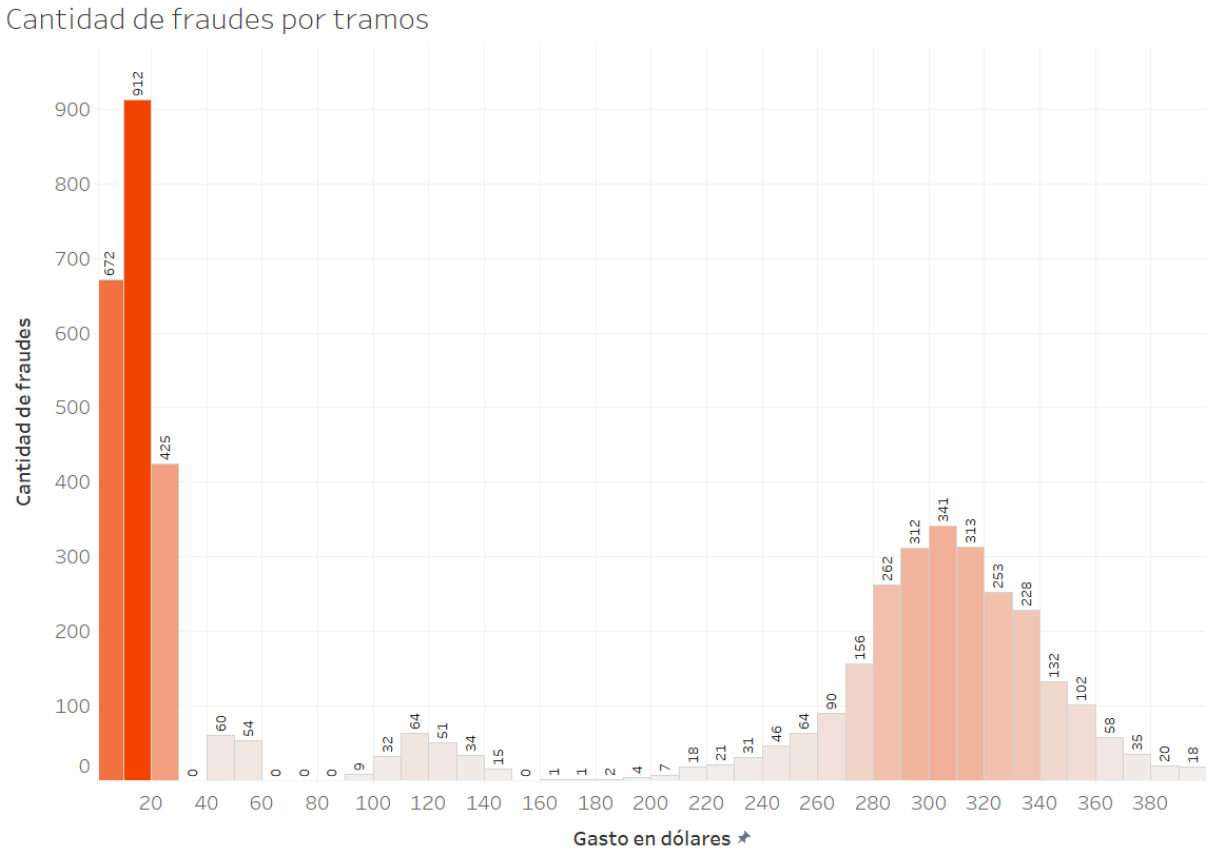


En la gráfica anterior se observa claramente que la cantidad de transacciones de poca entidad es muy superior a las transacciones de varios cientos de dólares.

Probabilidad de fraude en función de la cantidad de la transacción

Pese a ser el valor de la transacción una variable discreta (el crecimiento de la variable se da céntimo a céntimo), se enfoca como continua por la variabilidad de los datos. Al trabajar con una variable interpretada como continua, el diagrama idóneo es el histograma.

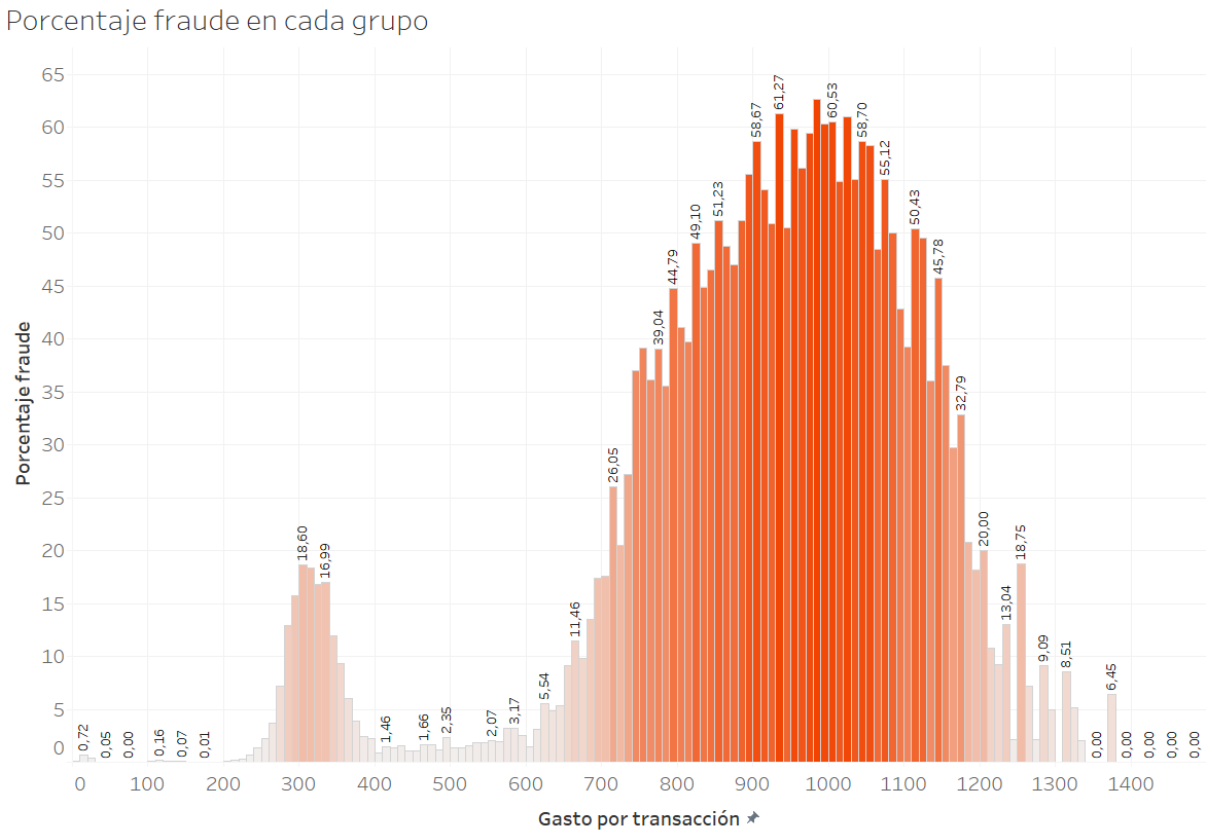
Empecemos con la frecuencia absoluta de fraudes por cada grupo:



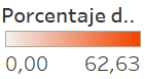
La tendencia de suma de is_fraud para amt (agrupación). El color muestra suma de is_fraud. Las marcas se etiquetan por suma de is_fraud.

La frecuencia absoluta de fraudes es muy superior en los grupos de bajo gasto por transacción. Tras ello hay un tramo de baja frecuencia absoluta hasta los 200 dólares. A partir de ahí parece haber una subida en la cantidad de fraude.

Pero la frecuencia absoluta es un indicador de baja relevancia, pues debe ser contrastada con la cantidad de transacciones por grupo, así que mostramos otra gráfica que indica el porcentaje de fraudes en cada grupo respecto del total de transacciones en dicho grupo:



La tendencia de Porcentaje fraude para amt (agrupación). El color muestra Porcentaje fraude. Las marcas se etiquetan por Porcentaje fraude.



En este caso podemos ver que la gráfica tiene una forma muy diferente, demostrando que la probabilidad de fraude en gastos más elevados es muy superior a la probabilidad de fraude en gastos de poca entidad.

De cara a marcar probabilidad de fraude las transacciones que varían entre los 200 y los 400 dólares, así como las transacciones en el entorno de los 1000 dólares reflejan un riesgo muy superior de ser fraude que las transacciones de baja entidad.

Las conclusiones en este apartado son varias:

- La cantidad gastada tiene una clara relación con la probabilidad de que sea fraude. A efectos de detener una transacción y estudiarla por su posibilidad de ser fraude claramente las transacciones en el entorno de los 1000 dólares son un objetivo claro.
- Por otro lado, el volúmen de transacciones elevadas es muy inferior al volúmen de transacciones de baja cantidad. Por ello, estas últimas deben ser estudiadas también por suponer un alto volúmen de fraude pese a la baja probabilidad de ser fraudulentas.

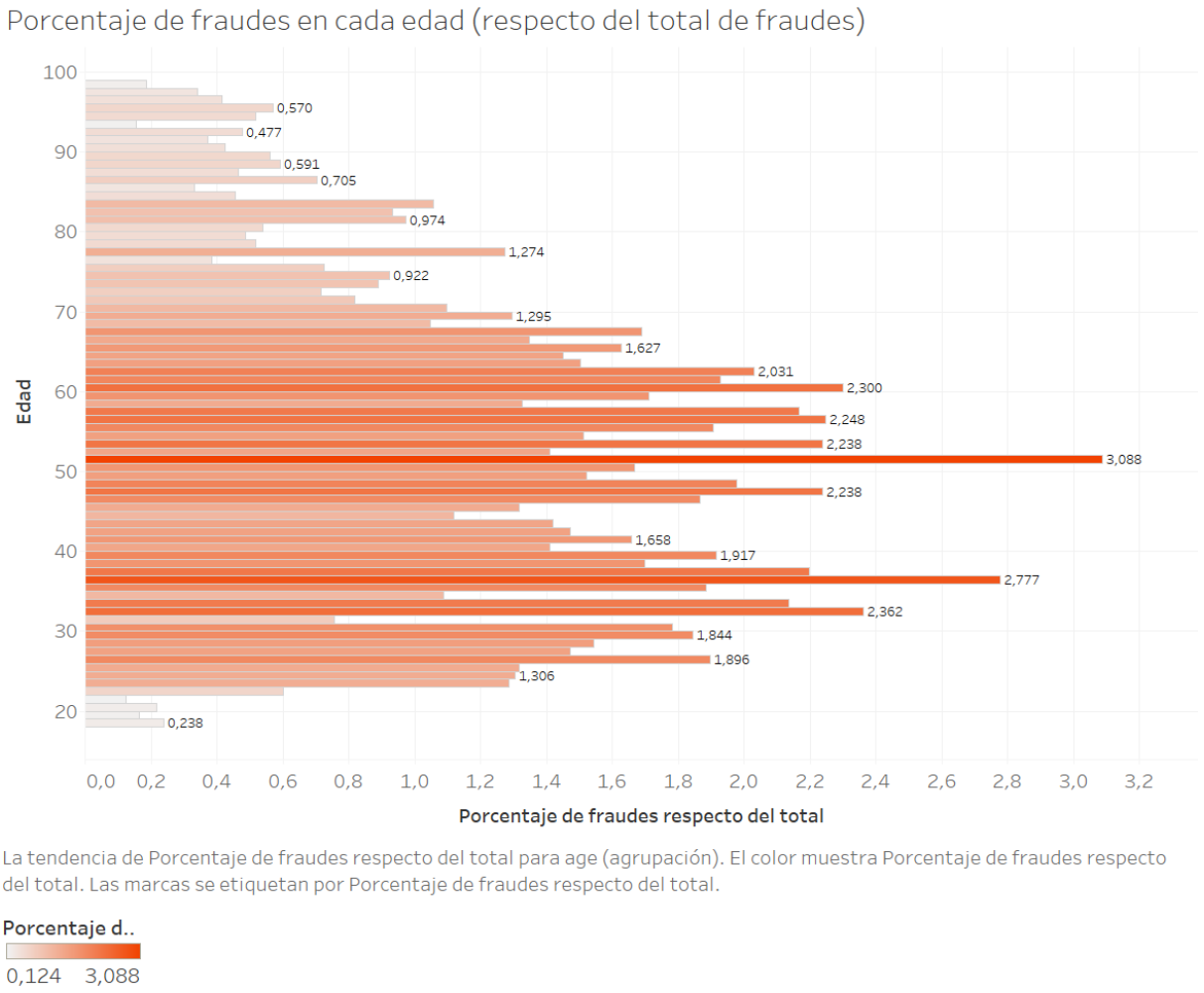
Concluimos que la cantidad de la transacción es una variable muy determinante a la hora de pronosticar un fraude.

Probabilidad de fraude en función de la edad

Analicemos una variable diferente. La edad de la persona que realiza la compra.

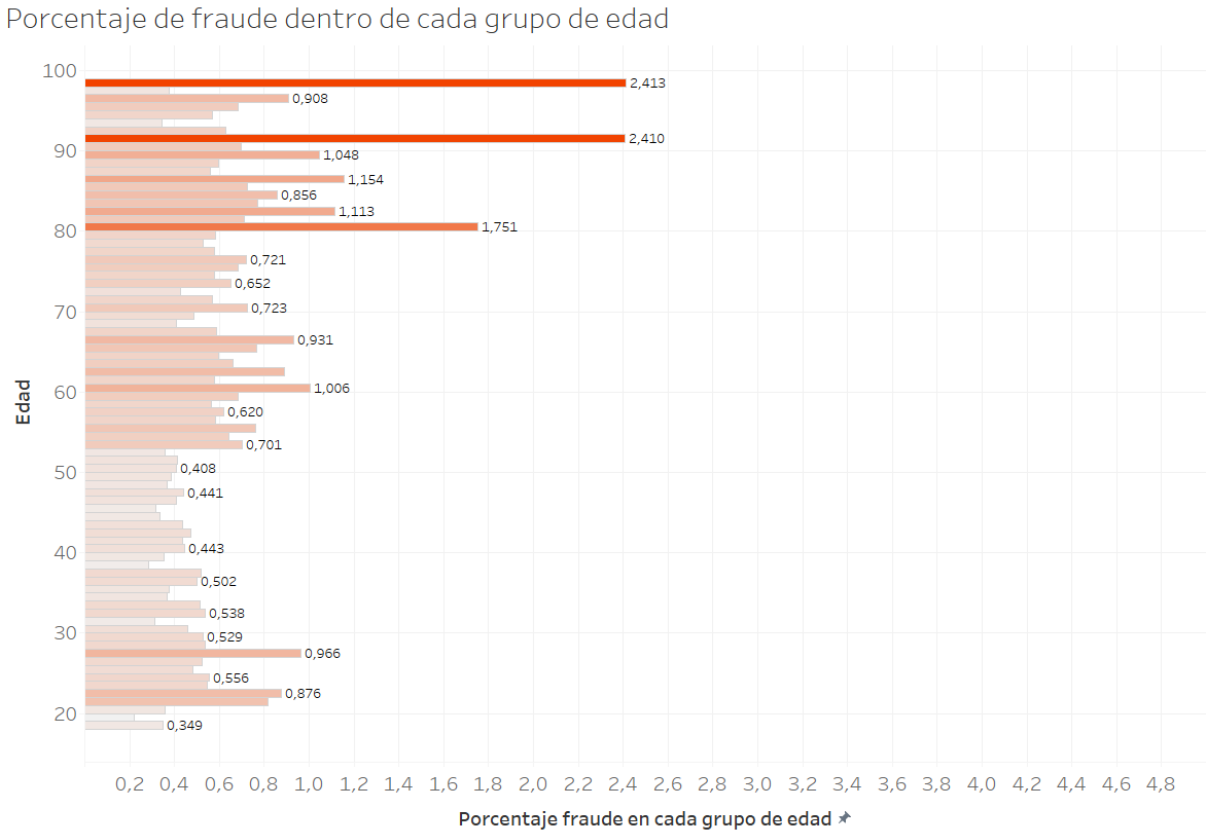
Estamos ante una variable discreta, que en el caso de nuestro dataset se mueve en un rango de los 18 a los 100 años.

Empezamos mostrando una gráfica que indica la frecuencia absoluta, en porcentaje, de fraudes en cada edad:



La gráfica parece indicar que la probabilidad de fraude es superior en las edades comprendidas entre los 18 años y el entorno de los 70 años, encontrando un par de picos cerca de los 40 y al poco de pasar los 50.

Sin embargo, la gráfica anterior no muestra la probabilidad de que una transacción sea fraudulenta en cada grupo de edad, por lo que mostraremos una gráfica que recoja la cantidad de fraude en función del total de transacciones en cada tramo de edad, en porcentaje:



La gráfica anterior muestra que, al obtener la probabilidad dentro de cada tramo de edad, la probabilidad de fraude no sigue un patrón lineal en su variabilidad. Hay determinadas edades concretas que duplican la probabilidad frente al resto, pero al ser casos aislados no parece relevante y más fruto de la casualidad.

El no encontrar un incremento en tramos de edad concretos, sino en edades concretas, y ser la probabilidad de fraude siempre un valor relativamente constante, podemos desechar esta variable como no relevante a la hora de pronosticar el fraude.

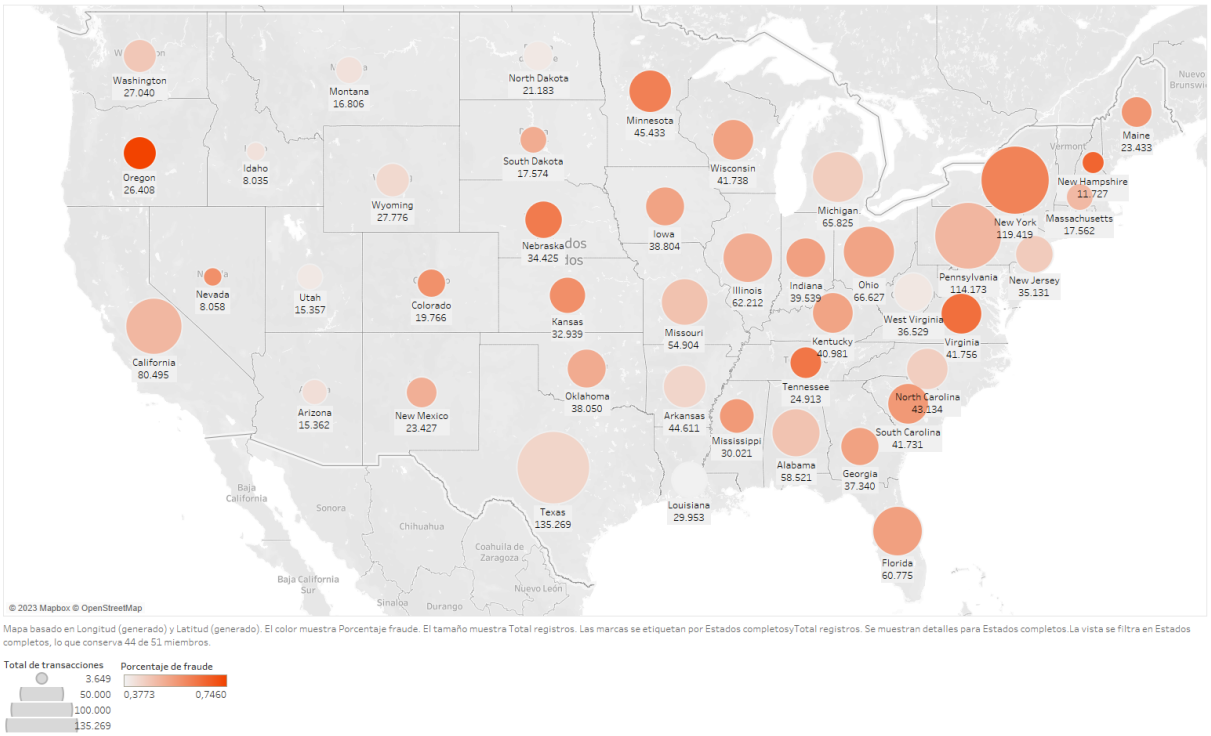
Quizá se podría indicar que en edades avanzadas la probabilidad es (muy) sensiblemente superior a tramos de edad más bajos.

Fraude en función del estado

Por último, vamos a trabajar con los estados de Estados Unidos donde se realizan las transacciones para determinar la probabilidad de fraude de una transacción.

Empecemos mostrando la cantidad de transacciones por estado:

Cantidad de transacciones y porcentaje de fraude



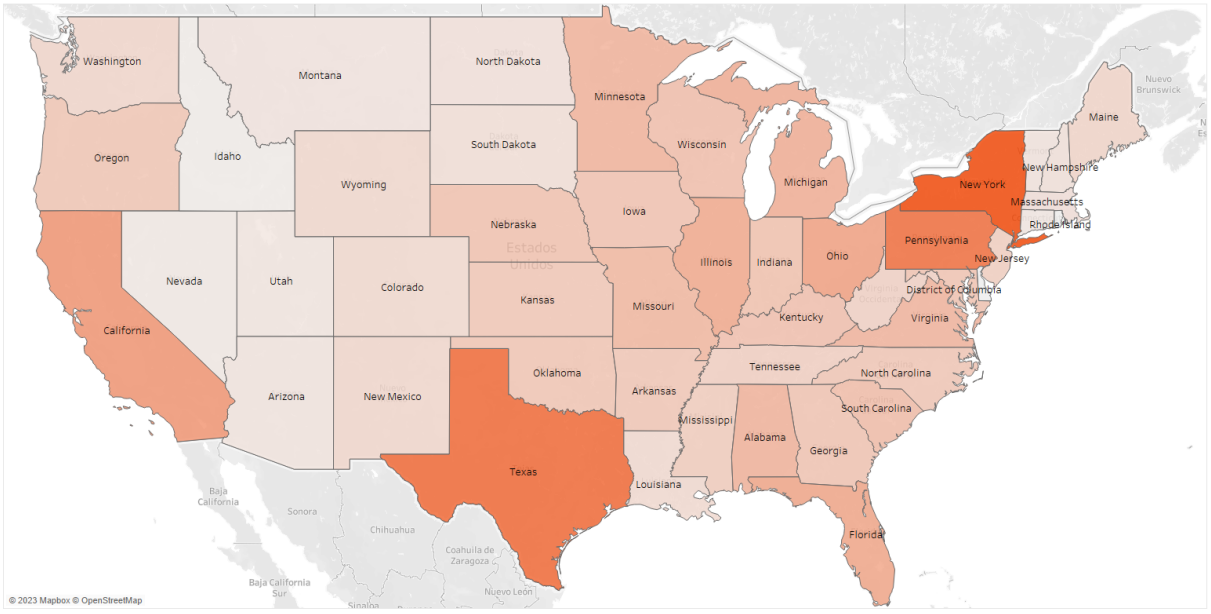
Mapa basado en Longitud (generado) y Latitud (generado). El color muestra Porcentaje fraude. El tamaño muestra Total registros. Las marcas se etiquetan por Estados completosyTotal registros. Se muestran detalles para Estados completos.La vista se filtra en Estados completos, lo que conserva 44 de 51 miembros.

En la gráfica anterior, para facilitar la visualización, se han ocultado estados muy próximos y se han quitado ciertos outliers (Alaska y un par de estados con un porcentaje de fraude prácticamente del 100%). Además, se prescinde de Hawaii por su lejanía en el mapa que dificulta la visualización de la gráfica.

Se puede apreciar que en el espacio del país entre el eje vertical del mismo y la costa este la cantidad de transacciones es superior a la parte contraria. También, por norma general, la cantidad de fraude es mayor en dichos estados.

Mostramos ahora la cantidad de fraude en cada estado respecto del total de fraude:

Porcentaje de fraude en cada estado (sobre el total de fraude)



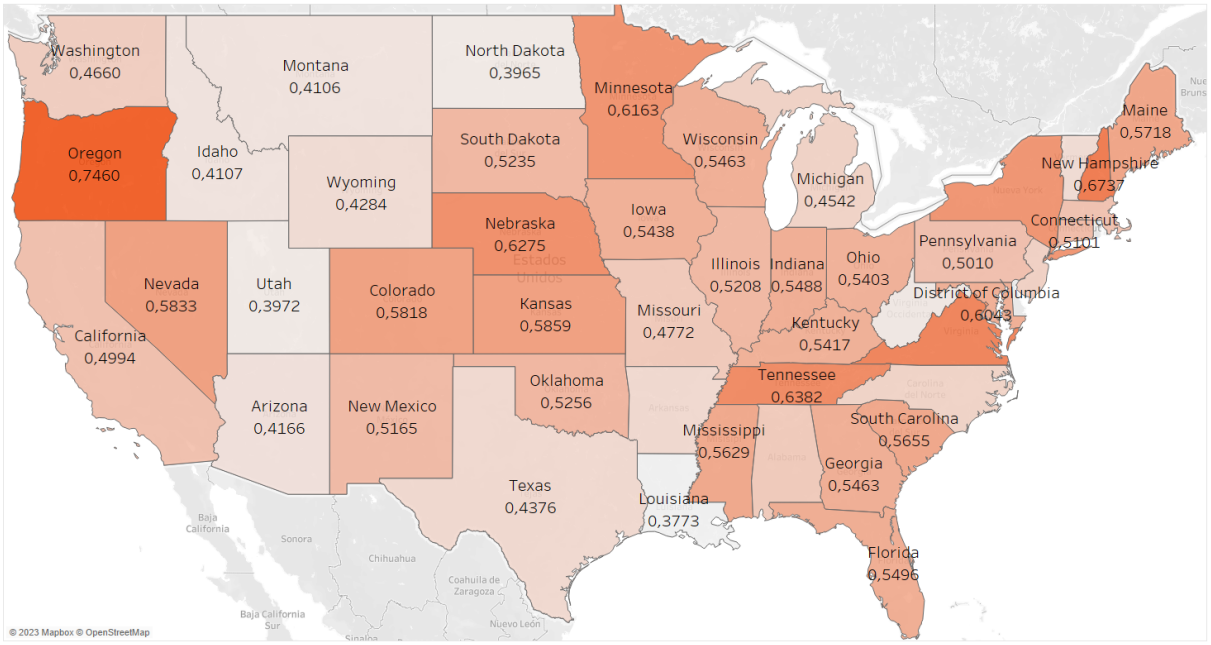
Mapa basado en Longitud (generado) y Latitud (generado). El color muestra Porcentaje de fraudes respecto del total. Las marcas se etiquetan por Estados completos.

Porcentaje d..
0,093 7,564

Lo mostrado es muy similar a lo anteriormente comentado. Hay un porcentaje sensiblemente superior en los estados de la costa este que en los estados de la costa oeste.

Mostremos ahora el porcentaje de fraude en cada estado respecto del total de transacciones de dicho estado:

Porcentaje de fraude en cada estado



Mapa basado en Longitud (generado) y Latitud (generado). El color muestra Porcentaje fraude. Las marcas se etiquetan por Estados completosyPorcentaje fraude.La vista se filtra en Estados completos, lo que excluye Alaska,DelawareyRhode Island.

Porcentaje fr..
0,37730,7460

De nuevo nos encontramos ante un cambio al analizar la frecuencia relativa al número de transacciones por estado. La tendencia que indica que el fraude es mayor en estados de la costa este sigue presente, pero el estado con mayor porcentaje de fraude respecto de sus transacciones es Oregón, en la costa oeste.

En este caso, dada la variabilidad, sí que podríamos indicar que la variable Estado, relacionada con la localización, es relevante, aunque bastante por detrás del gasto de la transacción (la primera variable analizada).

Resto de variables

Analizando las variables de índole económica, porcentajes de paro o fecha, entre otras, no se ha encontrado una relevancia significativa, de forma similar a la variable edad, por lo que se descartan en el estudio.

Existe una variable en concreto, la que indica la profesión de la persona que paga la transacción, que es dificil de categorizar, por la enorme dispersión de profesiones y la dificultad de agruparlas en grupos o sectores profesionales más concretos, por lo que se desecha su análisis.

Análisis DAFO

Al iniciar el proyecto, quisimos llevar a cabo un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) para tener un punto de partida de nuestra situación como equipo de trabajo.

A continuación lo mostramos:

Debilidades	Amenazas	Fortalezas	Oportunidades
Problema complejo, poco conocimiento del mismo Datasets difíciles de encontrar (LOPD)	Plazos de entrega ajustados Poca experiencia en el desarrollo de todas las etapas de un Proyecto Big Data	Equipo Apoyo técnico de los profesores del Bootcamp	Diseñar un nuevo approach, sencillo pero operativo Enriquecer los datasets encontrados con información relevante

Lecciones Aprendidas (*Lessons Learnt*)

¿Qué nos ha aportado el desarrollar este proyecto?

Desarrollar este proyecto nos ha aportado varías cosas, entre ellas ver cómo plantear un proyecto de data desde la primera fase de ideación y así valorar los inconvenientes, soluciones y la viabilidad de nuestra idea. Trabajar en equipo también ha sido muy enriquecedor.

Tener con quién debatir cómo afrontar un problema proporciona una valiosa oportunidad de aprendizaje. Íbamos afrontando cada reto que surgía a lo largo del proyecto.

¿Qué hemos aprendido?

Sobre todo, que no es siempre posible abordar todo el problema. Dado que, según íbamos trabajando, nos encontramos con errores y/o inconvenientes (por ejemplo, numerosas variables categóricas) que requerían volver a valorar el problema.

Por ejemplo, con la Arquitectura, nos hubiera gustado hacer un bucle automatizado y poder reentrenar el modelo. Sin embargo, por habernos tropezado con cosas que nos detenían más de lo esperado, no hemos conseguido reentrenar el modelo. Hemos usado nuevas herramientas, como por ejemplo *Vertex AI* de *GCP*.

También ha sido un aprendizaje el tener que trabajar y coordinarnos como equipo 100% remoto.

¿Qué no volveríamos a hacer de la misma manera?

No volveríamos a escoger datos sintéticos. El hecho de trabajar con datos sensibles que no permitieran obtener un dataset no sintético nos ha condicionado mucho.

Sería interesante, y tendría más valor si pudiéramos volver a entrenar el modelo y también sacar insights para poder evitar y reducir los casos de fraude y proteger a los clientes del banco.

Tuvimos muchísimos problemas con los créditos de prueba de GCP, si hubiéramos iniciado este apartado desde el principio, el tiempo no hubiera corrido tan en nuestra contra.

Y, sobre todo, hemos echado mucho en falta el rol definido de un Project Manager que liderara y gestionara las tareas a lo largo del proyecto.

¿Qué cosas seguiríamos haciendo en el futuro para mejorar el proyecto?

En el futuro, para mejorar el proyecto estaría bien escoger otras variables que pudieran ser más relevantes a la hora de predecir el fraude.

Al trabajar con un dataset sintético y con datos sensibles, han faltado variables que podrían haber aportado más información y, a su vez, más correlación en cuanto al fraude (por ejemplo, fecha de alta de la tarjeta, tipo de tarjeta).

Ahora que hemos visto la manera más sencilla de enviar una notificación correr el modelo, trataríamos de avisar por correo electrónico.

También nos gustaría abordar el reentrenamiento del modelo al entrar nuevos datos. De esta forma conseguiríamos un bucle prácticamente automatizado de detección de fraude

Anexo - Repositorio de Datos, Credenciales

Para desarrollar este proyecto, hemos creado dos cuentas de Google, adjuntamos las credenciales para que los profesores puedan acceder a ambas y consultar los ficheros e información que necesiten. Cualquier problema, contáctenos.

- **Cuenta Google Drive**
  - tfb.bigdata11@gmail.com
  - password: Bootcamp11
- **Cuenta Google Cloud Platform (GCP)**
  - vrg1984cloud2@gmail.com
  - password: Bootcamp11

Todos los csv que hemos usado en el proyecto están en dos ficheros .zip ubicados en una carpeta pública de Google Drive (como alternativa frente a la limitación de espacio que tienen los repositorios de GitHub).

- **Carpeta Data**: data

Drive está organizado en carpetas según los apartados del proyecto, más o menos. Este es el aspecto de la carpeta “Trabajo Final Bootcamp Big Data”: