NEQUI: Prueba Técnica

Heberto Jimenez Estupiñan

10 de agosto de 2023

1. Introducción

El proyecto busca entender y analizar transacciones ocurridas entre 2020 y 2021, con el fin de identificar malas practicas, específicamente una practica conocida como fraccionamiento transaccional. Esta se basa en transacciones múltiples intencionadas con el fin de evitar una sola transacción de un monto superior, en periodos de tiempo inferiores a 24 h.

Para este proyecto, se hace uso de Python, como principal lenguaje, y Pyspark como framework para la exploración y modelado de los datos.

El proyecto fue desarrollado en principalmente dos archivos: report.ipynb, el cual contiene el código relacionado con la exploración de los datos, y Model.ipynb el cual contiene el codigo relacionado con el modelado.

2. Exploratory Data Analysis

2.1. Carga de datos

Los datos están divididos en dos archivos parquet. Con el fin de hacer un mejor análisis, se hizo uso de ambos archivos, estos se cargaron y concatenaron para generar un solo archivo spark, con dimensiones = (21516918,8).

2.2. Limpieza

Después de concatenar, existía la posibilidad de encontrar datos duplicados, además que era importante eliminar transacciones que tuvieran valores nulos. Después de esto, los datos tenían las siguientes dimensiones = (21516907,8). Una diferencia de 11 transacciones, correspondientes a duplicados. Esto quiere decir, que no existían datos nulos.

2.3. Exploración

Una primera exploración de los datos es necesaria para entender las magnitudes y relaciones de los datos. De esta se encontraron algunos datos relevantes como:

- Un usuario puede tener mas de una cuenta. Mientras que hay 3,087,217 usuarios únicos, hay 3,099,711 de cuentas únicas, una diferencia de 12494 cuentas.
- Como se mencionó al principio, los datos correspondes exclusivamente a los años 2020 y 2021.
- Los montos totales de las transacciones en 2021 son 432,914 veces mayores a las de 2020. Podemos decir que los datos están mayoritariamente representados por el año 2021.
- Existen solo 3 merchants para todas las transacciones. Sin embargo, el merchant con id terminado en 7758 contiene el 85 % del total de transacciones de 2021.
- Con subsidiary los montos de las transacciones tienen una distribución uniforme.

- Esta distribución uniforme también se ve en los días de las semana, exceptuando el día domingo, donde los montos caen a la mitad. Respecto a los meses, Diciembre y Enero, los clientes guardan mas su dinero. Esto podría darnos una intuición de que son clientes corporativos, pues clientes naturales pueden transferir mas en estas festividades.

2.4. Transacciones fragmentadas

En este caso, las transacciones las podemos dividir en dos grupos, transacciones que fueron segmentadas y aquellas que no. Para identificar de forma mas eficiente estos grupos, se creó una columna flag Y. Este vector se creó de la siguiente manera:

Apartir de un window function, se agrupan los valores por user_id y se ordenan por transaction_date. Se crea una columna temporal llamada time_diff, que contiene la fecha de transacción de la transacción anterior. Si la resta de transaction_date y time_diff es menor a 24h (86400 segundos), entonces las transacciones de marcan con 1. Y así sucesivamente.

Para este calculo se usó el user_id y no el account_number, pues es posible que clientes, fracciones sus transacciones entre diferentes cuentas.

Una vez identificadas las transacciones fragmentadas, se observa que los clientes con mayores cantidades de transacciones a su vez son los que mas fragmentan las transacciones. En este top 20 de clientes, no se observa un patrón claro respecto a la desviación estándar de las transacciones, pues varían desde los 5 dolares hasta los 858 dolares.

3. Modelo Analítico

3.1. Procesamiento de datos

En esta etapa, se busca transformar los valores actuales de las transacciones, a valores que no solo sean mas fáciles de leer al humano, sino a la maquina también. En este proceso también es importante determinar que datos de una transacción son realmente relevantes a la hora tomar una decisión clave. En este orden de ideas, se busca determinar cuales columnas son prescindibles o que de alguna manera pueden afectar la validez del modelo a seleccionar. En primer lugar el dato relacionado con el identificador de la transacción (_id), no aporta ningún tipo de información relevante a la hora de hacer un análisis pues estos se van creando con cada transacción. En segundo lugar, el identificador del cliente, una vez se utiliza para la clasificación de los datos, puede generar ruedo a la hora de testear los modelos, pues si cada día aparecen clientes nuevos, cambiaría la distribución de los datos, y los modelos perderían relevancia, es por esto, que esta columna también es eliminada. Esto mismo ocurre para el numero de la cuenta.

Para el resto de columnas, se hacen las siguientes transformaciones.

- merchant_id & transaction_type: Estas variables categóricas primero se relacionan con un index, para posteriormente ser modeladas por one-hot-encoding.
- transaction_amount: Estos valores de transacción varían desde los 5.9 hasta 4624.8 dolares. Si hacemos una normalización regular, la cantidad de valores podría genera ruido al modelo. Entonces se propone transformarlas en categorías, pequeñas, medianas o altas. Esta distribución en base en los quantilies 25, 50 y 75.

summary	transaction amount
count	21516907
mean	191.3
stddev	240.8
min	5.9
max	4624.7

- transaction_date: Esta columna se divide en las siguientes columnas; Year, Month, Day, Day of Week and Hour. No se usan los segundo para evitar ruido y complejidad, además que los rangos de legitimidad están basados precisamente en horas.
- subsidiary: Para esta columna se determino usar la frecuencia de sus valores, es posible que esta columna agregue ruido al model, es por esto, que se marca como una posible falla, y de no encontrarse un modelo adecuado, como mitigación se propone remover este feature.

3.2. Selección de modelo

Como se mencionó anteriormente, las transacciones se pueden separar en dos grupos, esto en otras palabras es clasificación, específicamente, clasificación binaria, pues solo se tienen dos clases, Transacciones legítimas y Fraudulentas. Por limitaciones de recursos (tiempo y hardware) se propone emplear y comparar, Random Forest Classifier y Gradient Boosting Classifier (similar a regresión logística), para posteriormente seleccionar el mejor respecto a la validación. Estos modelos fueron seleccionados gracias a su capacidad con datos tabulares, manejo de datos des balanceados y manejo de features como los de fechas.

3.3. Evaluación

Marcar una transacción como fragmentada cuando no lo era, o no detectar una transacción fragmentada, tienen consecuencias similares, determinar cual de los dos lados preferir esta por fuera del scope del proyecto, es por esto que lo ideal, sera un modelo que minimice ambas, es por esto que para la evaluación de los modelos y posterior selección, se determino como métrica la el area bajo la curva ROC o mas conocida como AUC.

El modelo Gradient Boosting Classifier, tomo mas tiempo de lo esperado. Se espera con con mejores recursos se pueda entrenar y evaluar. Para este proyecto entonces, se define como modelo Random Forest Classifier, con un AUC = 0.65. Un modelo con un desempeño aceptable. Con esto podemos concluir que si eliminamos el feature subsidiary podríamos obtener mejores resultados.

3.4. Frecuencia de actualización

Como se menciono en el principio, hay un gran desbalance respecto a los años de las transacciones, no hay datos posteriores a 2021, y no hay información sobre como han actuado los bancos para sancionar a los clientes que hacen este tipo de practicas. Este drifting puede generar un bias en los modelos, y con el tiempo pueden pasar a ser obsoletos. Alimentar el modelo muy seguido, por el otro lado, recarga considerablemente los recursos y en vez de ayudar, puede generar varianza en los modelos. Es entonces que se sugiere, una retroalimentación de los modelos de manera semestral, con datos mejor distribuidos, y que representen de mejor manera el compartamiento de los clientes a lo largo del año, además, como se vio, enero y diciembre son outlayers que deben evadirse.

3.5. Arquitectura

Con el fin de reducir costos y asegurar escalabilidad, servicios en la nube como AWS son sugeridos. El uso de lambdas y EC2 que permitan evaluar y hacer inferencia en los modelos, apartir de cpu, debido que los tiempos de inferencia con cpu y gpu en estos modelos no son considerables, se ayudan a disminuir costos. Estas lambas, permitirían integrarse perfectamente con python, y su flexibilidad perimite incoporarlos en cualquier etapa de la arquitectura ya previamente definida por la empresa.