Documentación del Proyecto: Detección de Fraccionamiento Transaccional

Introducción

Este documento proporciona una descripción detallada de nuestro proyecto de detección de fraccionamiento transaccional en NEQUI. El objetivo principal de este proyecto es identificar transacciones sospechosas de fraccionamiento y desarrollar una solución efectiva para mitigar este comportamiento fraudulento.

Descripción de la Solución

1. Alcance del Proyecto

Dado que el problema de fraccionamiento transaccional está ligado a varios factores, nos interesa observar:

- **Frecuencia de las transacciones:** Identificar múltiples transacciones desde hacía un mismo número de cuenta en un corto período de tiempo puede ser un indicio de fraccionamiento transaccional, incluso si cada transacción es pequeña.
- **Patrones de tiempo:** Observar si las transacciones ocurren en un período de tiempo relativamente corto, como dentro de unas pocas horas o días.
- **Número de transacciones:** Analizar si hay un incremento inusual en el número de transacciones en comparación con el comportamiento histórico del número de cuenta.
- **Destino de las transacciones:** Investigar si las transacciones tienen un patrón de destino inusual o si todas se dirigen a una cuenta o cliente específico.

2. Exploración y Evaluación de Datos (EDA)

- **Carga de Datos**: Para la parte inicial de la exploración y visualización los datos se cargaron con PySpark. El resto de procesamiento, se realizó con Pandas.

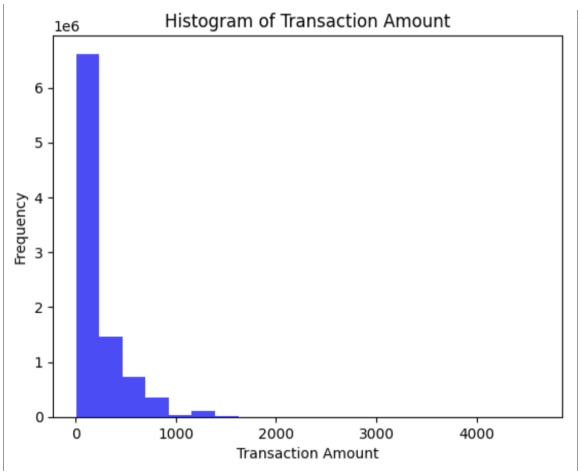
- **Análisis Exploratorio de Datos**:

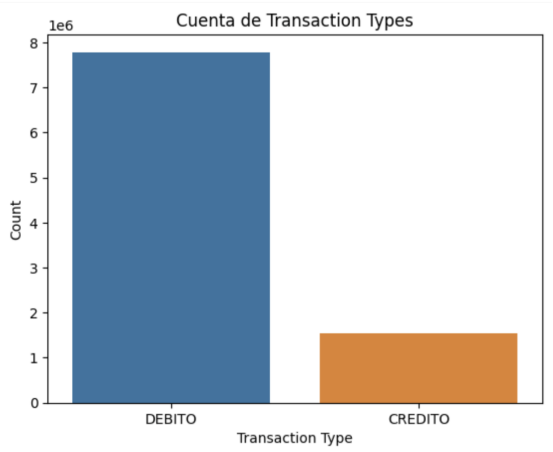
Resumen	monto de transacción
count	9327314
mean	192.73607794843915
stddev	24.55010991860652
min	5.944455
max	4624.786

tipo de transacción	cuenta
DÉBITO	7782400
CRÉDITO	1544914

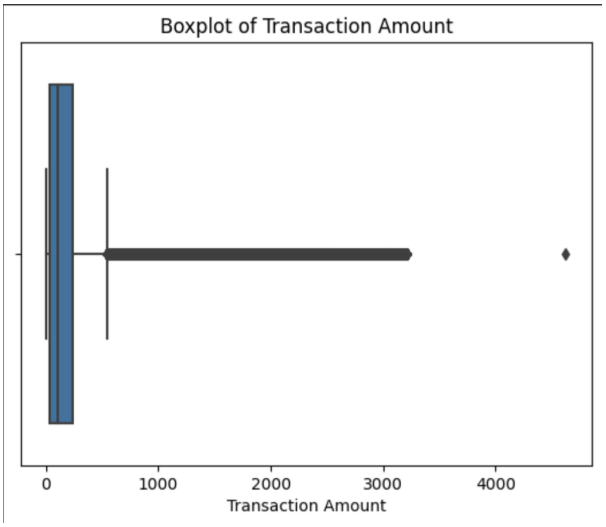
fecha mínima	fecha máxima
2021-01-01 00:01:13	2021-11-30 23:58:41

Visualización de Datos:





Identificación de Valores Atípicos :



- **Selección de Características Relevantes**:

Al estar buscando fraccionamiento transaccional, no consideré relevantes los registros con número de cuenta que habían hecho menos de 4 transferencias en el histórico.

De otro lado también era importante analizar sobre un mismo número de cuentas, cuantas transacciones se hacían en ventanas de 24 horas, y eliminar los registros de usuarios que no habían tenido al menos 4 transacciones en una franja de 24 horas. Con esto habré establecido un conjunto candidato de usuarios que cometen fraccionamiento transaccional.

Es importante resaltar que hay otros enfoques que podrían abarcar más variantes del problema, por ejemplo añadiendo más etiquetas que puedan señalar usuarios como posibles infractores, ejemplos de etiquetas que se podrían implementar:
-. En la exploración de datos noté que este fraude en muchos de los datos del conjunto de datos se da en ventanas de 72 horas. Así que ampliar la ventana sería bueno.
-. Utilizar métodos como (KDE) para ver la densidad y quizás fijar un umbral más acorde a los datos sería de gran utilidad.
-. Si se conociera el monto objetivo a evadir, habría más formas de abordar el problema.
-.Un modelo como HDBSCAN, nos podría ayudar a modelar cómo se agrupan los usuarios y darnos mayor comprensión sobre las características de los usuarios con malas prácticas transaccionales, tanto en general como el resto de los usuarios.
Identificación de Transacciones de Mala Práctica Transaccional (Fraccionamiento Transaccional):
1. El usuario ha realizado más de 4 transacciones en una ventana de 24 horas:
    - Implementar un algoritmo que agrupe las transacciones por usuario y fecha.
    - Contar el número de transacciones para cada usuario en cada ventana de 24 horas.
    - Etiquetar las transacciones de usuarios que superen el límite de 4 transacciones como sospechosas de fraccionamiento.
2. Las múltiples transacciones del usuario tienen un monto bajo en comparación con su historial:
    - - Calcular la media de transacciones históricas de cada usuario.
    - - Comparar el monto de las transacciones actuales con la media del histórico del usuario.

3. El total del monto de las múltiples transacciones es grande en comparación con su historial:
    - - Identificar grupos de transacciones de un mismo usuario que ocurran en la misma ventana de tiempo de 24 horas, en los que al menos 60% de registros, tienen monto de transacción menor que la mediana del monto total de las transacciones para ese usuario, además la suma de los montos que abarca ese 60% debe ser al menos el 20% del movimiento histórico de ese usuario. Estos porcentajes se basan en la ley de Pareto pero con alguna modificación para que se adapte al problema.
    - Las dos condiciones anteriores conformarán el umbral elegido.
    - - Etiquetar las transacciones que caigan por debajo de este umbral como sospechosas.
    - - Etiquetar las transacciones que superen un cierto umbral como sospechosas.

3. Definición del Modelo Analítico

- **Trazado del Flujo de Datos**:
Los siguientes pasos los lleva a cabo el programa 'detectar_fraccionamiento.py' el cual en su interior necesita cargar un csv para su funcionamiento.

- La primera parte del flujo consta de la limpieza del conjunto de datos esta incluye: búsqueda y eliminación de datos faltantes, nulos y duplicados. Se verificó consistencia comprobando que los montos de las transacciones siempre fueran valores positivos y que la identificación de las transacciones fuera siempre única, además se cambió el tipo de dato del monto de transacciones a flotante y se aseguró que la columna fecha de transacciones este en formato datetime64.
- Basado en la selección de características relevantes se filtró el data frame, seleccionando solo usuarios con más de 4 transacciones en su histórico y con más de tres transacciones en el mes.
- Para cada 'account_number' se toma su primer registro y agrupamos los registros que están hasta 24 horas después de él. Luego, tomamos el siguiente registro que no pertenece al grupo anterior y de nuevo tomamos los registros que están hasta 24 horas después de él. Esto se itera hasta terminar de recorrer los elementos de 'account_number'.
- Se marcan las ventanas que contienen 4 o más transacciones en su interior con un un 1 y con el '_id' de la primera transacción en esta ventana. Estas marcas se guardan en las columnas 'varias_transacciones_xdia' y 'misma_ventana'. Las que no cumplen la condición anterior se marcan con ceros.
- Por medio de observación de los datos, se eligió el umbral como la mediana, pues los datos tienen una distribución hacía la izquierda de tipo Pareto y la media sería muy volátil. Los IQR no resultaron una buena elección así que. Dado que los datos son tan variables, cada 'account_number' tiene un umbral asociado, que es la mediana de los montos de sus transacciones.
- Del data frame tomamos los registros cuya columna 'varias_transacciones_xdia' sea igual a 1. Esto nos da las ventanas de 24 horas con más de 4 transacciones por día, en cada una de estas ventanas revisamos dos condiciones: primero que al menos el 60% de estas transacciones sea menor que el umbral, segundo la suma total de esas transacciones es al menos el 20% del movimiento histórico del usuario. Los registros que cumplen esto, los marcamos con un uno en una nueva columna 'posible_fraccionamiento'. Los que no cumplen esta condición son marcados con cero.
- Por último, se guarda un dataframe en formato csv que contiene las columnas iniciales y las creadas en este proceso, en la que la columna 'posible_fraccionamiento' identifica posible fraccionamiento transaccional.

- **Selección del Modelo**: Con estos candidatos elegidos, los siguiente fue aplicar un método no supervisado de detección de anomalías:

- En principio consideré algoritmos de clustering no supervisados. Pues estos agruparán las transacciones similares en grupos o clústeres, y los casos que no se ajustaran bien a ningún clúster podrían considerarse posibles fraccionamientos transaccionales. HDBSCAN para ser exactos, pues es eficaz en la identificación de clústeres de formas y tamaños irregulares, dado que HDBSCAN puede descubrir automáticamente el número de clústeres, pues no conocemos la cantidad de fraccionamientos que se pueden encontrar.

- Después de esto, me incliné por Random Forest un algoritmo supervisado. Es robusto y menos propenso a sobre ajustar los datos en comparación con un solo árbol de decisión. Se puede configurar para manejar conjuntos de datos desequilibrados de manera efectiva, no requiere ajustar tantos hiper parámetros y es adecuado para aplicaciones donde tienes muchas variables para considerar.

- **Frecuencia de Actualización de Datos**: Note que el fraccionamiento transaccional tiende a ocurrir en un período de 24 horas, por lo que es importante actualizar los datos con la misma frecuencia para identificar patrones de manera oportuna.
De otra forma el modelo podría perder transacciones fraudulentas que ocurrieron entre actualizaciones.

4. Criterios para la Entrega

- **Repositorio de GitHub**:
https://github.com/JennViafara/Deteccion-Fraccionamiento-Transaccional-Nequi
-**Drive:**
Este enlace contiene, los dataframes resultantes del modelo('posible-fraccionamiento' marca con uno los registros donde posiblemente hubo fraccionamiento transaccional)
https://drive.google.com/drive/folders/1RjEm-vVvt7iO_3VBme7mP45w0UVYbmjs?usp=sharing

Incorporación del Producto de Datos en un Marco Cooperativo

1. Preparación de Datos:
- Utilizar el producto de datos que fue desarrollado, para etiquetar las transacciones sospechosas de fraccionamiento.
2. Comunicación con Equipos de Fraude:

- Establecer una comunicación efectiva con los equipos y departamentos relevantes en la organización, como el equipo de prevención de fraude, seguridad y otros equipos de análisis de datos. Proporcionarles acceso a los datos etiquetados y explicar el método de detección.
- Programar reuniones regulares para compartir los resultados y descubrimientos de la solución implementada, así como para discutir cualquier actualización o cambio en las reglas y políticas.

2• **Acceso a los Datos Etiquetados:** Proporcionaré acceso a los datos etiquetados por mi solución a otros equipos mediante la creación de una base de datos compartida.

- Implementar un sistema que permita a los equipos consultar y utilizar estos datos de manera eficiente, asegurando que estén disponibles para su análisis.

3. Desarrollo de Reglas de Negocio:

- Trabajar en conjunto con los equipos de negocio y de fraude para definir reglas y políticas claras para abordar las transacciones sospechosas.

4. Integración en el Flujo de Trabajo:
   - Asegurar que el proceso de detección de transacciones sospechosas se integre en el flujo de trabajo existente de la empresa.
   - Con esto implementado, lo ideal sería no solo detectar el fraude una vez cometido. Sino más bien lograr la detección oportuna y prevención de estos fraudes, por medio de alertas que se generen por supuesto por medio de un modelo de predicción.
5. Monitoreo Continuo:
   - Establecer un sistema de monitoreo continuo, esto para alimentar el modelo para detectar y abordar nuevas instancias de fraccionamiento transaccional a medida que ocurren.
6. Actualización y Evaluación Constante:
   - Evaluar regularmente la efectividad de tu modelo y ajustar los umbrales y reglas según sea necesario para mantener la precisión en la detección.

Conclusiones

- Se logró definir un algoritmo eficiente, para la detección de fraccionamiento transaccional, el cual reconoce ciertos tipos de esta mala práctica.
- Se identificaron posibles mejoras para este proceso, tales como:
    - Ampliación del rango de 24 horas. Esto pues en la exploración de datos se hicieron notables cierto tipo de transacciones sospechosas que ocurren a lo largo de días.
    - Modificación de umbrales, siguiendo mecanismos más sofisticados como KDE para ver la densidad.
    - Establecer más parámetros sobre lo que se considera fraccionamiento transaccional, vinculando otras de las columnas, y asignando pesos a estas etiquetas.
    - Explorar la manera en la que se construyen las ventanas de tiempo.
    -
- Al aplicar el modelo que sirve para la detección de anomalías HBDSCAN el resultado que se obtuvo no fue el deseado. Esto se puede deber a la forma de los datos o la selección de hiper parámetros y características, sin embargo al efectuar variaciones de estas, el resultado no cambió.
- Con el algoritmo que se desarrolló en esta prueba, se generaron etiquetas, por esto un modelo supervisado, fue el siguiente paso, al implementar random forest sobre una sección de aproximadamente ⅛ de los dataset, el modelo arrojó las siguientes métricas

  Matriz de Confusión:
  [[49017 35]
  [106 1607]]
  Informe de Clasificación:

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	49052

1.0	0.98	0.94	0.96	1713
accuracy			1.00	50765
macro avg	0.99	0.97	0.98	50765
weighted avg	1.00	1.00	1.00	50765

Estas pruebas indican que el camino promete buenos resultados. Sin embargo, para tener un resultado definitivo es necesario tomar un dataset más exhaustivo y ajustar los umbrales de decisión para controlar aún más el equilibrio entre precisión y recall.

- XGBoost es una buena opción para implementar teniendo en cuenta que en este proyecto ya había elegido parámetros y puesto etiquetas para clasificar fraccionamiento transaccional, una buena opción sería usar algoritmos de clasificación que están diseñados para manejar conjuntos de datos desequilibrados. Elegí este porque tiene buen rendimiento para conjuntos de datos grandes y es compatible con python.
- Isolation Forest es un algoritmo no supervisado que podría ayudar debido a su capacidad para identificar observaciones atípicas en un conjunto de datos, que a diferencia de los algoritmos de clustering no requiere definir previamente el número de clústeres. Esto es beneficioso pues no sabemos cuántos fraccionamientos esperamos encontrar.
- El algoritmo aquí descrito detecta fraude cuando ya ocurrió, pero modificando un poco los parámetros podría ser útil para predecir y emitir alertas antes de que el fraude se lleve a cabo en su totalidad.
- Si bien, estos procesos mencionados logran detectar el fraude una vez ocurrido, lo ideal sería implementar algoritmos para la predicción, que se pueden generar después de comprender más cómo suelen comportarse los usuarios que incurren en el fraccionamiento transaccional.