Rappi DS Challenge

Leonardo Pacheco

14 de mayo de 2021

1. Análisis del conjunto de datos

El conjunto de datos está compuesto por 26975 registros con los siguientes atributos:

- ID_USER: data type int64
- genero: data type object M, F, no definido
- monto: data type Float64
- fecha: data type timestamp desde Enero 2 hasta enereo 30 del 2020
- hora: data type int64
- dispositivo: data type object
- establecimiento: data type object, Restaurante, Abarrotes, Super, MPago, Farmacia
- ciudad: data type object Toluca, Guadalajara, Merida, Monterrey
- tipo_tc: data type object Física, Virtual
- linea_tc: data type int64
- interes_tc: data type int64
- status_txn: data type object Aceptada, En Proceso o Rechazada
- is_prime: data type bool

- dcto: data type Float64
- cashback: data type Float64
- fraude: data type bool

1.1. Distrubución de los datos por día de la semana

El día de la semana en el que más se usa la tarjeta es el viernes, y el que menos es el miércoles. Lo anterior se ve reflejado en la figura 1

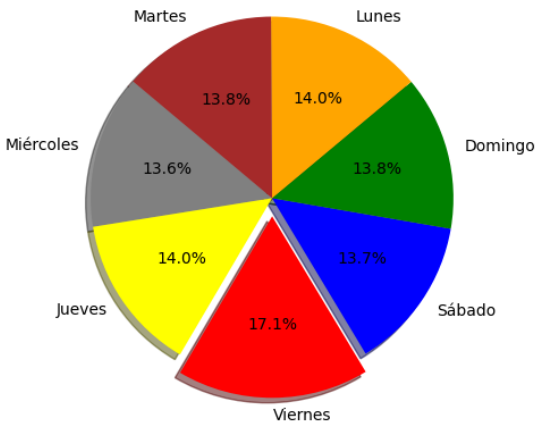

Figura 1: Días de la semana

1.2. Distrubución de los datos por genero

La mayoría de los clientes que usaron la tarjeta y su genero fue registrado, son hombres. Aproximadamente el 10 % de los datos no tiene genero definido. Lo anterior se puede ver en la figura 2

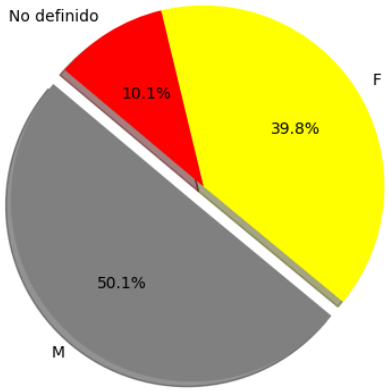

Figura 2: Genero

1.3. Distribución de los datos por legalidad

De todos los datos únicamente el 3 % ha sido clasificado como fraude; esto es, 810 casos han sido fraudes positivos. Ver figura 3

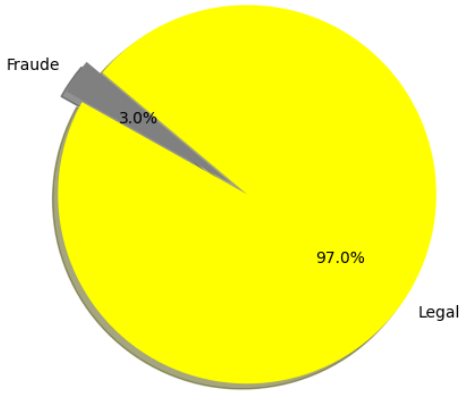

Figura 3: Fraude

1.4. Distribución de los datos por establecimiento

En el diagrama de la figura 4 se observa que la mayoría de los clientes han usado sus tarjetas en restaurantes y donde menos la han usado es en las framacias, aunque en realidad no hay mucha variación.

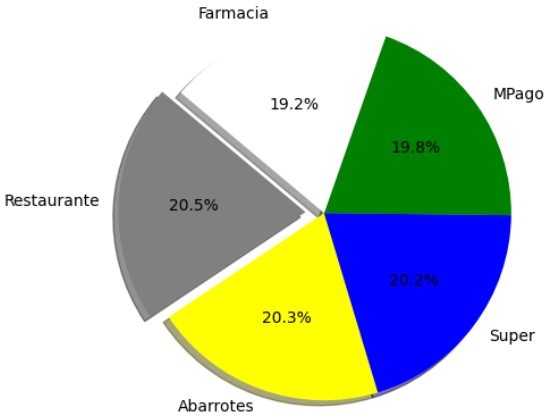

Figura 4: Establecimiento

1.5. Registros faltantes

En la figura 5 podemos ver que los únicos atributos que no tienen algunos valores son los de establecimiento y ciudad.

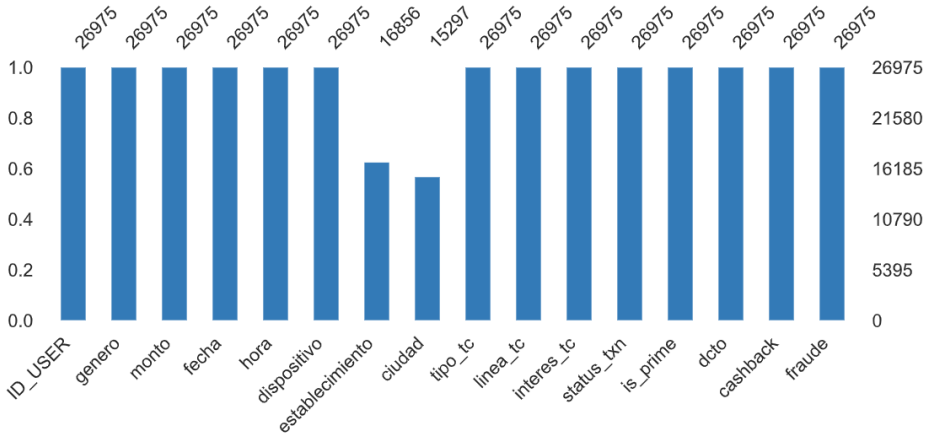

Figura 5: Registros faltantes

1.6. Correlación de las variables

En la figura 6 se representa la crorelación de las variables dada por el coeficioente de Pearson. Se puede observar como es natural, que hay una alta correlación entre monto-cashback, monto-descuento y descuento-cashback. Además se puede ver que no hay una relación lineal tan fuerte entre las variables no mencionadas arriba.

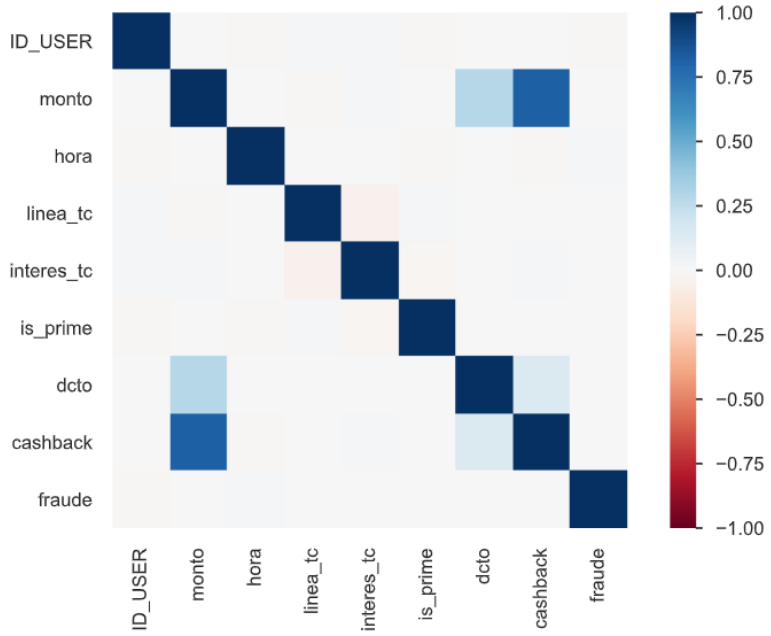

Figura 6: Correlación de Pearson

Por otra parte, en la figura 7 se puede ver que hay relación entre tipo_tc-cashback, linea_tc- interes_tc, linea_tc-is_prime

2. Visualización de los datos en 2D

En la figura 8 se muestra el resultado de hacer un PCA con dos componentes. Para manejar las variables categóricas se ha usado la técnica de Traget Ecoding.

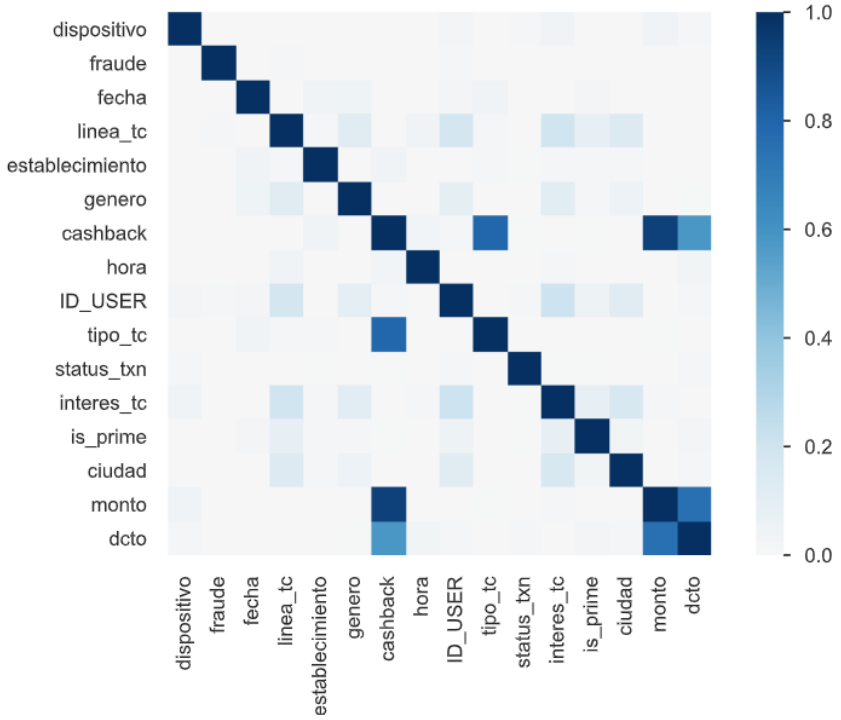

Figura 7: Correlación φk

En la figura 9 aparecen los clientes representados en 2D, se han sumado las variables numéricas (incluyendo las catégoricas transformadas) y se ha hecho PCA.

3. Clasificasión de los clientes

3.1. K-Means

Para escoger el número de clusters se usa la figura 10, el punto de codo, parece corresponder a 7 clusters.
En la figura 11 se ve el número de clientes por cluster, resultado de ejecutar K-Means

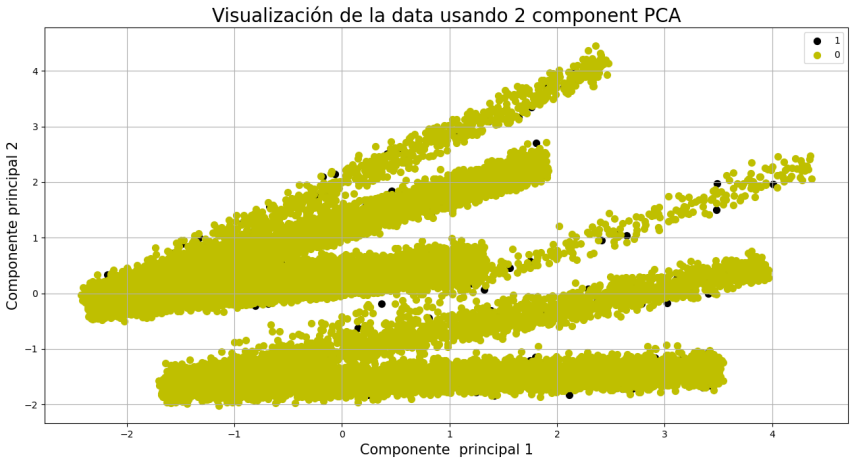

Figura 8: Visualización

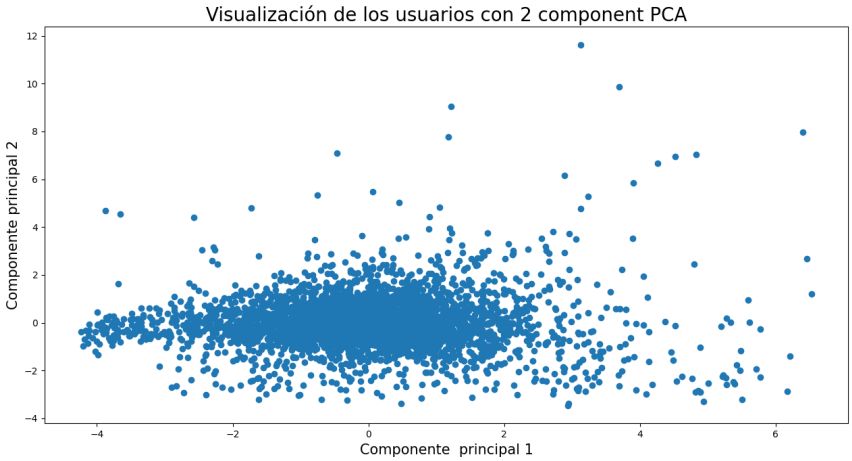

Figura 9: Clientes

3.2. Clustering jerárquico usando dendogrma

Basado en la figura 12 se observa que al cortar a la altura de 60 se obtienen 5 clusters.

En la figura 13 se ven los resultados de correr el clustering jerárquico.

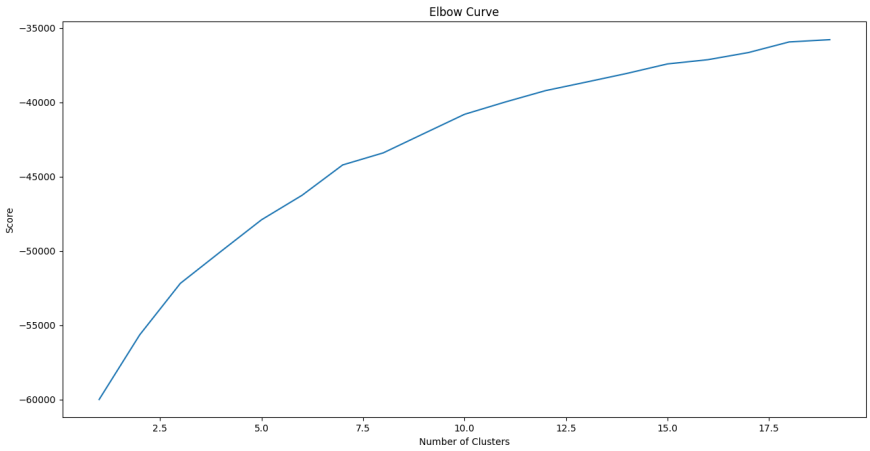

Figura 10: Número de clusters

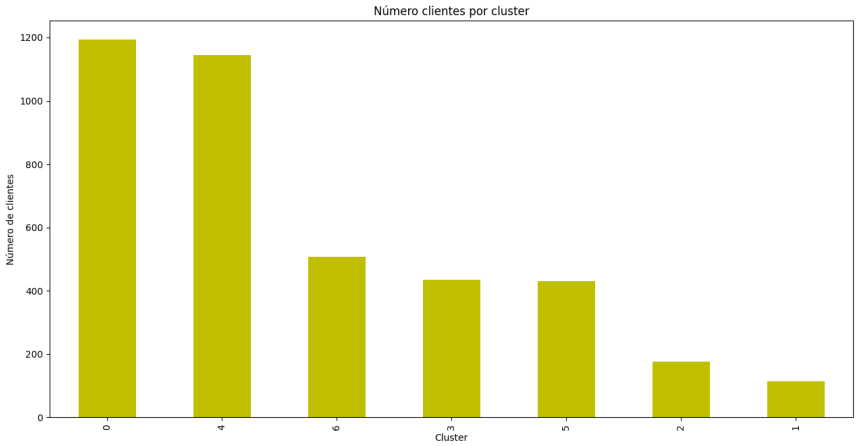

Figura 11: Clusters

Para ver exactamente los resultados de los dos tipos de Clustering realizados, se puede correr el archivo CustumerClassification.py.

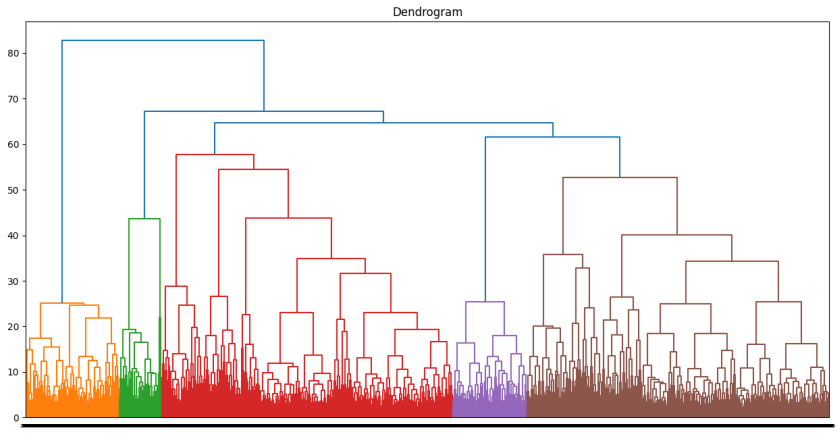

Figura 12: Dendograma

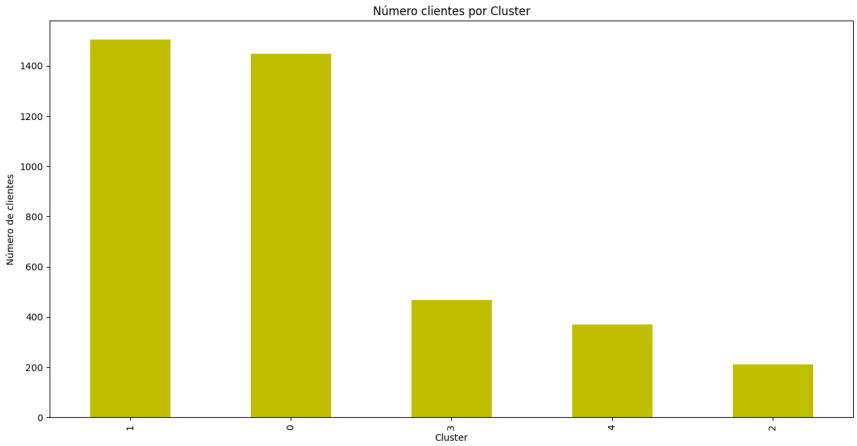

Figura 13: Clustering jerárquico

4. Modelo de detección de Fraude

El modelo de detección de fraude seleccionado usa el clasificador BalancedRamdomForest y Synthetic Minority Oversampling Technique (SMOTE) de imblearn. La siguiente es la matriz de confusión obtenida al correr el modelo

con un conjunto de prueba correspondiente al 20 % de la muestra.

$$C = \begin{bmatrix} 5977 & 1858 \\ 190 & 68 \end{bmatrix} \tag{1}$$

Donde

$$\begin{aligned} C_{11} &= \text{Legales predichos correctamente} \\ C_{12} &= \text{Legales predichos como fraudes} \\ C_{21} &= \text{Fraudes predichos como legales} \\ C_{22} &= \text{Fraudes predichos correctamente} \end{aligned}$$

El modelo es capáz de predecir

$$C_{22} = 68$$

fraudes acertadamente, sin embargo deja pasar

$$C_{21} = 190$$

fraudes como legales. Un modelo bueno de detección de fraude, debería cumplir con:

$$C_{21} < C_{22}, \tag{2}$$

por lo anteriror, hay que mejorar el modelo.

Al ejecutar .feature_importances_ para el modelo, se obtiene que:

$$\begin{aligned} \text{genero} &\rightarrow 0{,}332119 \\ \text{status\_txn} &\rightarrow 0{,}206891 \\ \text{ciudad} &\rightarrow 0{,}143411 \\ \text{dispositivo} &\rightarrow 0{,}108265, \end{aligned}$$

Los otros predictores tienen valores menores a estos cuatro, entonces podemos concluir que las variables que más determinan el comportamiento del modelo son las listadas arriba. Todos los datos mencionados arriba se pueden obtener al ejecutar el archivo FraudstersDetector.py

El trade-off de usar este modelo para la predicción de fraudes es que aunque detecta algo de fraude de manera acertada, deja pasar muchos casos de fraude. Además se puede afectar a muchos clientes dado que

$$C_{12}$$

da un número tan alto. Idealmente se debería buscar un modelo en el que C_{22} sea mayor que C_{12} y C_{21}; esto es , un modelo que beneficie al cliente y a la empresa.