Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022

Análisis: Abstract data set for Credit card fraud detection

Carga de bibliotecas

En este análisis se usará la biblioteca `SciPy` para realizar un agrupamiento jerárquico y `scikit-learn` para realizar un modelo de detección de anomalías. En ambos casos se persigue detectar de forma automática valores inusuales dentro de un conjunto de datos.

[1]:
```
from scipy.cluster.hierarchy import dendrogram, linkage, cophenet,␣
↪fcluster
from scipy.spatial.distance import pdist
from sklearn import metrics
from sklearn.ensemble import IsolationForest
import copy
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Carga de dataset y resumen de datos

Se usará un *dataset* (Joshi, 2018) el cual corresponde al *dataframe* que se usará durante el análisis.

[2]:
```
df = pd.read_csv("./ds/creditcardcsvpresent.csv")
```

Este dataframe contiene once columnas. Las primeras dos de ellas serán borradas porque una corresponde a un índice de datos y la otra es una columna completamente vacía, por lo tanto irrecuperable.

[3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3075 entries, 0 to 3074
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022

```
 0   Merchant_id                      3075 non-null   int64
 1   Transaction date                 0 non-null      float64
 2   Average Amount/transaction/day   3075 non-null   float64
 3   Transaction_amount               3075 non-null   float64
 4   Is declined                      3075 non-null   object
 5   Total Number of declines/day     3075 non-null   int64
 6   isForeignTransaction             3075 non-null   object
 7   isHighRiskCountry                3075 non-null   object
 8   Daily_chargeback_avg_amt         3075 non-null   int64
 9   6_month_avg_chbk_amt             3075 non-null   float64
 10  6-month_chbk_freq                3075 non-null   int64
 11  isFradulent                      3075 non-null   object
dtypes: float64(4), int64(4), object(4)
memory usage: 288.4+ KB
```

Eliminación de columnas

Primero será necesario guardar la columna objetivo `isFraudulent` en una nueva variable, pues será borrada del dataframe de trabajo debido a que utilizaremos métodos de análisis no supervisados.

[4]:
```
ideal_results = df["isFradulent"]
```

Ahora es posible borrar todas las columnas que no son necesarias para el análisis a realizar.

[5]:
```
df = df.drop(["Merchant_id", "Transaction date", "isFradulent"], axis=1)
```

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3075 entries, 0 to 3074
Data columns (total 9 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Average Amount/transaction/day   3075 non-null   float64
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022

```
 1   Transaction_amount              3075 non-null   float64
 2   Is declined                     3075 non-null   object
 3   Total Number of declines/day    3075 non-null   int64
 4   isForeignTransaction            3075 non-null   object
 5   isHighRiskCountry               3075 non-null   object
 6   Daily_chargeback_avg_amt        3075 non-null   int64
 7   6_month_avg_chbk_amt            3075 non-null   float64
 8   6-month_chbk_freq               3075 non-null   int64
dtypes: float64(3), int64(3), object(3)
memory usage: 216.3+ KB
```

Este dataframe contiene nueve columnas, las cuales no son descritas en la fuente original, por lo que solo es posible intuir su significado, por supuesto, esto podría condicionar la discusión producto del análisis. Es importante hacer énfasis en proporcionar metadatos sobre cualquier conjunto de datos computables: texto, audio, video, dataset, etc.

Seis de esas columnas son de tipo numérico y las tres restantes son categóricas, enseguida se muestra su descripción general.

[7]:
```
df.describe().transpose()
```

[7]:
```
                                count         mean           std       ␣
↪min  \
Average Amount/transaction/day  3075.0   515.026556    291.906978  4.
↪011527
Transaction_amount              3075.0  9876.399210  10135.331016  0.
↪000000
Total Number of declines/day    3075.0     0.957398      2.192391  0.
↪000000
Daily_chargeback_avg_amt        3075.0    55.737561    206.634779  0.
↪000000
6_month_avg_chbk_amt            3075.0    40.022407    155.968840  0.
↪000000
```

```
6-month_chbk_freq                3075.0      0.391870       1.548479   0.
↪000000

                                                  25%            50%             75%␣
↪ \
Average Amount/transaction/day    269.788047   502.549575    765.272803
Transaction_amount               2408.781147  6698.891856  14422.568935
Total Number of declines/day        0.000000     0.000000      0.000000
Daily_chargeback_avg_amt            0.000000     0.000000      0.000000
6_month_avg_chbk_amt                0.000000     0.000000      0.000000
6-month_chbk_freq                   0.000000     0.000000      0.000000

                                      max
Average Amount/transaction/day     2000.0
Transaction_amount               108000.0
Total Number of declines/day         20.0
Daily_chargeback_avg_amt            998.0
6_month_avg_chbk_amt                998.0
6-month_chbk_freq                     9.0
```

[8]:
```
df.describe(include='object').transpose()
```

[8]:

	count	unique	top	freq
Is declined	3075	2	N	3018
isForeignTransaction	3075	2	N	2369
isHighRiskCountry	3075	2	N	2870

[9]:
```
for o in ["Is declined", "isForeignTransaction","isHighRiskCountry"]:
    print("-----")
    print(df[o].value_counts())
```

```
-----
N    3018
Y      57
```

```
Name: Is declined, dtype: int64
-----
N    2369
Y     706
Name: isForeignTransaction, dtype: int64
-----
N    2870
Y     205
Name: isHighRiskCountry, dtype: int64
```

Tratamiento de variables categóricas

Se crean variables separadas, para no usar variables categóricas. La variable categórica `Is declined` que toma valores `Y` o `N` en `df[Ïs declined"]` se puede sustituir por dos variables *dummy*, booleanas, que son `Is declined_Y` e `Is declined_N`. Es posible tomar ambas variables o solo una de ellas, tal y como se hará en este análisis. Posteriormente se borra la variable original y se adjuntas las nuevas variables al dataframe.

[10]:
```
df = pd.get_dummies(df,columns=["Is declined",␣
↪"isForeignTransaction","isHighRiskCountry"],drop_first=True)
```

[11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3075 entries, 0 to 3074
Data columns (total 9 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Average Amount/transaction/day  3075 non-null   float64
 1   Transaction_amount              3075 non-null   float64
 2   Total Number of declines/day    3075 non-null   int64
 3   Daily_chargeback_avg_amt        3075 non-null   int64
 4   6_month_avg_chbk_amt            3075 non-null   float64
 5   6-month_chbk_freq               3075 non-null   int64
```

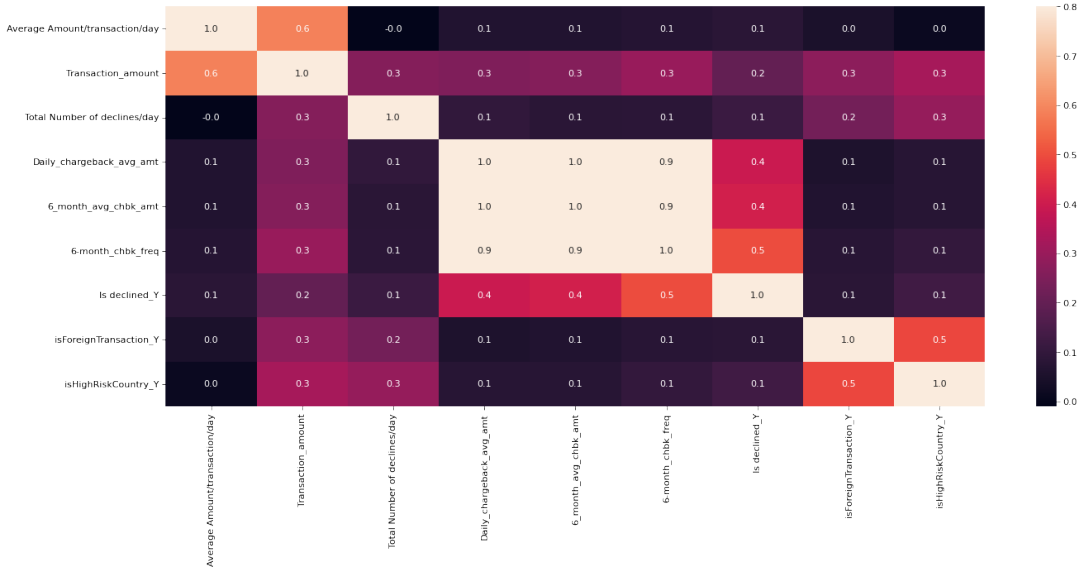
```
 6   Is declined_Y                3075 non-null   uint8
 7   isForeignTransaction_Y       3075 non-null   uint8
 8   isHighRiskCountry_Y          3075 non-null   uint8
dtypes: float64(3), int64(3), uint8(3)
memory usage: 153.3 KB
```

Matriz de correlación

Con la matriz de correlación es posible observar similitudes entre diferentes datos. Es posible observar que la matriz tiene zonas de colores similares, por ejemplo la parte central tiene tres variables que posiblemente sirvan para crear un grupo del cual quizá se construya una categoría, o bien termine por ser un grupo de datos poco relevantes para la clasificación.

[12]:
```
plt.figure(figsize=(20,8),dpi=80)
corrmat = df.corr()
sns.heatmap(corrmat, vmax=.8, fmt='.1f', annot=True)
```

[12]: <AxesSubplot:>



De esta forma, en este momento se hará la siguiente predicción:

- Existen al menos dos grupos de datos: Datos más relevantes para la clasificación y datos

menos relevantes para la clasificación.

- Las columnas `isForeignTransaction_Y`, `isHighRiskCountry_Y`, `Transaction_amount` y `Total Number of declines/day` parecen formar uno de esos grupos.
- Las columnas `Daily_chargeback_avg_amt`, `6_month_avg_chbk_amt`, `6-month_chbk_freq` y `Is declined_Y` forman el segundo grupo.
- No hay evidencia para la columna `Average Amount/transaction/day`.

Está predicción solo se convertiría en una hipótesis si fuera confirmada con una matriz de distancias. En este análisis se procederá directamente a implementar un método de clustering.

Clustering jerárquico

Un *cluster* jerárquico categoriza las entradas en grupos. Es un método no supervisado, por lo tanto no se usarán datos de entrenamiento, sino que todos los datos serán utilizados para crear una clasificación.

[13]:
```
# method=single, complete, average, weighted, centroid, median, ward
Z = linkage(df, "centroid")
```

Es necesario tener una métrica de evaluación del método. Por lo tanto se obtendrá una distancia de correlación *cophenetic* (c) y una matriz de distancias *cophenetic* condensada. Esto nos da una idea de cuán similares son los objetos agrupados. (The SciPy community, 2022)

[14]:
```
c, d = cophenet(Z, pdist(df))
print(c)
```

```
0.8915315812910991
```

La matriz *Z* se compone de cuatro elementos:

- Primer elemento a agrupar
- Segundo elemento a agrupar
- Distancia entre elementos
- Número total de elementos operados

[15]:
```
Z[1000:1010]
```

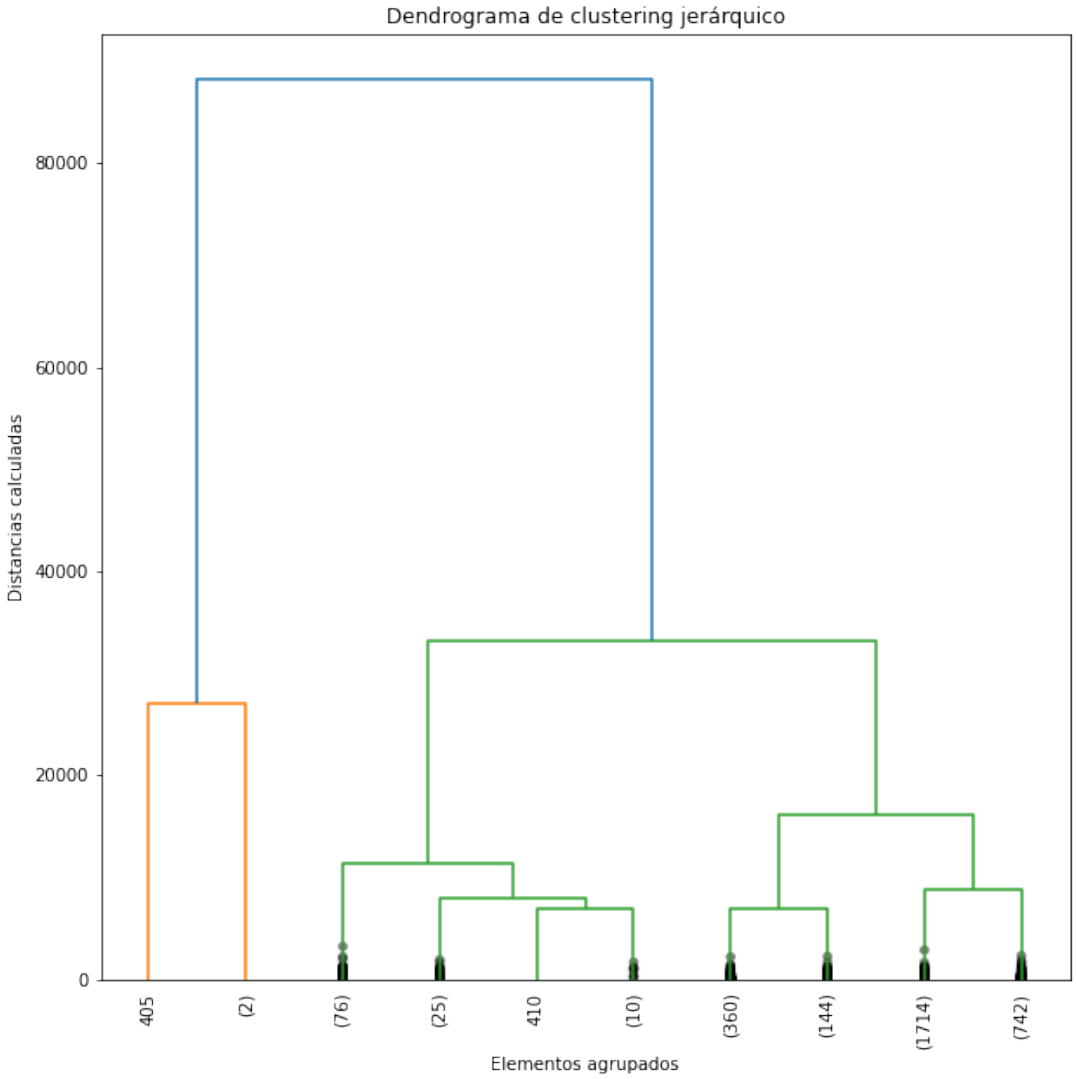
Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022

[15]:
```
array([[2.68400000e+03, 2.96400000e+03, 3.52987493e+01, 2.00000000e+00],
       [5.61000000e+02, 1.98100000e+03, 3.53043469e+01, 2.00000000e+00],
       [2.00800000e+03, 3.52200000e+03, 3.53214863e+01, 4.00000000e+00],
       [3.43400000e+03, 3.61700000e+03, 3.53636471e+01, 7.00000000e+00],
       [1.05500000e+03, 3.01900000e+03, 3.54156518e+01, 2.00000000e+00],
       [6.51000000e+02, 2.57700000e+03, 3.56502443e+01, 2.00000000e+00],
       [1.94100000e+03, 2.53700000e+03, 3.57463121e+01, 2.00000000e+00],
       [1.92100000e+03, 3.78600000e+03, 3.57849989e+01, 3.00000000e+00],
       [1.74000000e+03, 3.66600000e+03, 3.58070741e+01, 6.00000000e+00],
       [1.16400000e+03, 1.94500000e+03, 3.58250572e+01, 2.
↪00000000e+00]])
```

Ahora se mostrará un dendrograma truncado de la matriz Z. En este caso se mostrarán los diez elementos. En el eje de las ordenadas aparecerán las distancias de agrupación, en el eje de las abscisas aparecen dos posibles datos: entre paréntesis el número de elementos incluidos en la hoja, sin paréntesis el índice del elemento que se integra al cluster. Esto nos permite saber el tamaño de los clusters creados y visualizarlos mejor.

[16]:
```
plt.figure(figsize=(10,10))
plt.title("Dendrograma de clustering jerárquico")
plt.ylabel("Distancias calculadas")
plt.xlabel("Elementos agrupados")
dendrogram(Z, leaf_rotation=90.,leaf_font_size=10,␣
↪truncate_mode="lastp", p=10, show_leaf_counts=True,␣
↪show_contracted=True)
plt.savefig('im/dendrograma_jerarquico.png', format='png',␣
↪bbox_inches='tight')
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022



Recuperar clusters

[17]:
```
#clusters = fcluster(Z,40000,criterion="distance")
clusters = fcluster(Z,2,criterion="maxclust")
```

Es momento de observar el comportamiento como clasificador binario.

[18]:
```
print(pd.crosstab(ideal_results, clusters, colnames=["Predicted"],␣
 ↪rownames=["Real"]))
real = copy.deepcopy(ideal_results)
real.replace(to_replace={'N':2,'Y':1},inplace=True)
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022

```
print("Accuracy: ", metrics.accuracy_score(clusters, real))
```

```
Predicted  1     2
Real
N          0  2627
Y          3   445
Accuracy:  0.8552845528455284
```

Como puede observarse, este método falla al detectar aquellos casos en los que hay fraude. Es necesario tomar en cuenta que este cluster solo agrupa los datos, su objetivo directo no es identificar el fraude, por lo que es posible que las agrupaciones correspondan a criterios distintos. Es posible probar la predicción hecha anteriormente sobre la matriz de correlación, para ello se harán dos nuevos clusters con los conjuntos de datos listados entonces.

[19]:
```
Z1 = linkage(df[["isHighRiskCountry_Y","Transaction_amount","Total␣
 ↪Number of declines/day"]], "centroid")
Z2 = linkage(df[["Daily_chargeback_avg_amt","6_month_avg_chbk_amt","6-
month_chbk_freq","Is declined_Y"]], "centroid")
```

Es posible ver que ambos modelos son más favorables al obtener la distancia de correlación *cophenetic*. En el caso del conjunto 2 esta distancia es bastante alentadora aunque la proporción de verdaderos positivos y verdaderos negativos (*accuracy*) podría ser mejor. Es posible considerar al modelo obtenido de `Z2` como favorable.

[20]:
```
c1, d1 = cophenet(Z1,␣
 ↪pdist(df[["isHighRiskCountry_Y","Transaction_amount","Total Number␣
 ↪of declines/day"]]))
c2, d2 = cophenet(Z2,␣
 ↪pdist(df[["Daily_chargeback_avg_amt","6_month_avg_chbk_amt","6-
month_chbk_freq","Is declined_Y"]]))
print("Distancia de correlación cophenetic Z1: ",c1,"\nDistancia de␣
 ↪correlación cophenetic Z2: ",c2)
```

```
Distancia de correlación cophenetic Z1:  0.8930191106120398
Distancia de correlación cophenetic Z2:  0.9799097738300101
```

Asignatura	Datos del alumno	Fecha
Aprendizaje Automático	Bernal Castillo Aldo Alberto Calderón Zetter María Inés Domínguez Espinoza Edgar Uriel	3 de abril de 2022

[21]:
```
clusters2 = fcluster(Z2,2,criterion="maxclust")
print(pd.crosstab(ideal_results, clusters2, colnames=["Predicted"],␣
 ↪rownames=["Real"]))
print("Accuracy: ", metrics.accuracy_score(clusters2, real))
```

```
Predicted    1     2
Real
N           71  2556
Y          134   314
Accuracy:  0.8747967479674796
```

Isolation forest

Es turno de implementar un algoritmo que crea un prototipo de aquello considerable como "normal" en un dataset para luego identificar anomalías.

[22]:
```
ifc=IsolationForest(n_jobs=1,n_estimators=10000)
ifc.fit(df)
anomaly=ifc.predict(df)
```

Es destacable que se usó todo el dataframe para entrenar el modelo, esto ocurre así porque al analizar los valores de una característica, se pretende encontrar valores (o pequeños grupos de valores) que se apartan claramente del resto. (Duboue, 2020)

[23]:
```
real.replace(to_replace={2:1,1:-1},inplace=True)
print(pd.crosstab(ideal_results, anomaly, colnames=["Predicted"],␣
 ↪rownames=["Real"]))
print("Accuracy: ", metrics.accuracy_score(anomaly, real))
```

```
Predicted   -1     1
Real
N          124  2503
Y          422    26
Accuracy:  0.9512195121951219
```

Es posible observar una buena precisión en este método. Ha hecho un mejor trabajo al detectar

los casos en los que efectivamente se espera un fraude.

Reconstrucción del dataset

Ahora se procederá a guardar en un nuevo dataset toda la información resultante de la aplicación de los algoritmos anteriormente expuestos. Dicho dataset quedará almacenado en la carpeta `out` del proyecto.

[24]:
```
df["isFradulent"] = ideal_results
df["Clustering prediction_Y=1"] = clusters2
df["Isolation Forest prediction_N=1"] = anomaly
df.to_csv('out/creditcardcsvpresent_test_complete.csv')
```

Conclusión

En este análisis se examinó un dataset correspondiente a información financiera que en algunos casos corresponden a fraudes. Se realizó una descripción general de los datos, se hizo tratamiento en las variables categóricas y se obtuvo una matriz general de correlaciones. Es dicha matriz se observaron relaciones entre datos que podrían ayudar a formar grupos de entrada para el entrenamiento de un cluster. Esta idea, sin embargo, puede deberse solo a una coincidencia más que a una regla.

Se creó un cluster jerárquico con todos los datos disponibles en el dataframe y se recuperaron los dos grupos más grandes formados. Es posible observar que el cluster simplemente hace grupos, pero no distingue *a priori* aquellas cosas que el analista pudiera estar buscando. El modelo obtenido puede tener numerosas ramas y es responsabilidad del analista saber en que nivel o altura cortarlo. También parece que el algoritmo usado funciona mejor con conjuntos de datos de entrada específicos más pequeños o focalizados.

Posteriormente se implementó un algoritmo de detección de anomalías. El *isolation forest* aísla aquellos datos que se alejan de una norma establecida por el mismo modelo. Su implementación es muy similar a la de otros bosques probados en otras prácticas y se pudo probar que siempre que los parámetros de iteración sean adecuados puede ofrecer resultados bastante confiables.

El algoritmo de *clustering* es mejor detectando grupos de mayor tamaño mientras que el iso-

lation forest tiene mejor precisión cuando se trata preservar los grupos más pequeños, mejor dicho, puede descartar valores atípicos siempre que exista un conjunto histórico que pueda asegurarle cuales observaciones no son válidas dentro de un dominio determinado.

Referencias

Duboue, P. (2020). *The Art of Feature Engineering: Essentials for Machine Learning*. Cambridge University Press.

Joshi, S. (2018). Abstract data set for credit card fraud detection.

The SciPy community (2022). scipy.cluster.hierarchy.cophenet — scipy v1.8.0 manual.