semana3_actividad1_A01793919

October 4, 2022

1 Semana 3 - Actividad 1

1.1 Alumno: Luis José Navarrete Baduy

1.2 Matrícula: A01793919

1.3 Profesor: Jobish Vallikavungal Devassia

1.4 Fecha: 4 de octubre del 2022

1.4.1 Parte 1: Fundamentos de bases de datos

El propósito de un científico de datos es poder conocer y encontrar relaciones entre los datos para tomar mejores decisiones y aumentar la productividad, sin embargo, los datos que generan las empresas pueden ser de múltiples aplicaciones o bases de datos de tipo transaccional. Estas bases de datos generan gran cantidad de información en tiempo real es que muy difícil realizar el análisis, para esto existe otro tipo de base de datos que utilizan los científicos de datos que son de tipo analítico comúnmente conocido como data warehouse.

Los data warehouse tienen la característica de utilizar principalmente datos estructurados y en algunos casos semi-estructurados (snowflake) y en estos se puede tener un registro histórico de los datos de interés, al igual que existen diferentes capas en el data warehouse como el landing o raw donde se reciben los datos y se verifica que tenga el formato adecuado o se tenga que realizar un procesamiento previo a la fase de staging, el científico de datos puede conectarse a la base de datos y realizar solicitudes (queries) o de igual forma se pueden crear data marts para cada lineal especifica del negocio que desea analizar algún tipo de información esto sirve para limpiar los accesos de algunos usuarios a la información de la base de datos.

1.4.2 Parte 2: Selección y limpieza de los Datos en Python

Se importan las librerías para realizar el preprocesamiento

[33]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Se carga el dataset del repositorio y se elimina la columna id

[34]:
```
data_url = 'https://raw.githubusercontent.com/PosgradoMNA/
 ↪Actividades_Aprendizaje-/main/default%20of%20credit%20card%20clients.csv'
```

```
df = pd.read_csv(data_url)
df.drop('ID', axis=1, inplace=True) # remover columna id
df
```

[34]:

```
           X1   X2   X3   X4    X5   X6   X7   X8   X9  X10  …      X15  \
0       20000  2.0  2.0  1.0  24.0  2.0  2.0 -1.0 -1.0 -2.0  …      0.0
1      120000  2.0  2.0  2.0  26.0 -1.0  2.0  0.0  0.0  0.0  …   3272.0
2       90000  2.0  2.0  2.0  34.0  0.0  0.0  0.0  0.0  0.0  …  14331.0
3       50000  2.0  2.0  1.0  37.0  0.0  0.0  0.0  0.0  0.0  …  28314.0
4       50000  1.0  2.0  1.0  57.0 -1.0  0.0 -1.0  0.0  0.0  …  20940.0
…         …   …   …   …    …   …   …   …   …   …   …       …
29995  220000  1.0  3.0  1.0  39.0  0.0  0.0  0.0  0.0  0.0  …  88004.0
29996  150000  1.0  3.0  2.0  43.0 -1.0 -1.0 -1.0 -1.0  0.0  …   8979.0
29997   30000  1.0  2.0  2.0  37.0  4.0  3.0  2.0 -1.0  0.0  …  20878.0
29998   80000  1.0  3.0  1.0  41.0  1.0 -1.0  0.0  0.0  0.0  …  52774.0
29999   50000  1.0  2.0  1.0  46.0  0.0  0.0  0.0  0.0  0.0  …  36535.0

           X16      X17      X18      X19      X20     X21      X22     X23  \
0          0.0      0.0      0.0    689.0      0.0     0.0      0.0     0.0
1       3455.0   3261.0      0.0   1000.0   1000.0  1000.0      0.0  2000.0
2      14948.0  15549.0   1518.0   1500.0   1000.0  1000.0   1000.0  5000.0
3      28959.0  29547.0   2000.0   2019.0   1200.0  1100.0   1069.0  1000.0
4      19146.0  19131.0   2000.0  36681.0  10000.0  9000.0    689.0   679.0
…        …        …        …        …        …       …        …
29995  31237.0  15980.0   8500.0  20000.0   5003.0  3047.0   5000.0  1000.0
29996   5190.0      0.0   1837.0   3526.0   8998.0   129.0      0.0     0.0
29997  20582.0  19357.0      0.0      0.0  22000.0  4200.0   2000.0  3100.0
29998  11855.0  48944.0  85900.0   3409.0   1178.0  1926.0  52964.0  1804.0
29999  32428.0  15313.0   2078.0   1800.0   1430.0  1000.0   1000.0  1000.0

         Y
0      1.0
1      1.0
2      0.0
3      0.0
4      0.0
…    …
29995  0.0
29996  0.0
29997  1.0
29998  1.0
29999  1.0

[30000 rows x 24 columns]
```

Se crea la función getColumnInfo para verificar el tipo de dato de cada columna y el numero de datos vacíos

[35]:
```
def getColumnInfo(df: pd.DataFrame):
    # Obtenemos el total de registros vacíos en cada columna
    return list(zip(df.columns.values.tolist(),df.isna().sum().to_list(),df.
 ↪dtypes.to_list()))
getColumnInfo(df)
```

[35]:
```
[('X1', 0, dtype('int64')),
 ('X2', 1, dtype('float64')),
 ('X3', 2, dtype('float64')),
 ('X4', 2, dtype('float64')),
 ('X5', 5, dtype('float64')),
 ('X6', 3, dtype('float64')),
 ('X7', 5, dtype('float64')),
 ('X8', 7, dtype('float64')),
 ('X9', 9, dtype('float64')),
 ('X10', 16, dtype('float64')),
 ('X11', 14, dtype('float64')),
 ('X12', 11, dtype('float64')),
 ('X13', 11, dtype('float64')),
 ('X14', 13, dtype('float64')),
 ('X15', 15, dtype('float64')),
 ('X16', 17, dtype('float64')),
 ('X17', 10, dtype('float64')),
 ('X18', 8, dtype('float64')),
 ('X19', 9, dtype('float64')),
 ('X20', 8, dtype('float64')),
 ('X21', 11, dtype('float64')),
 ('X22', 11, dtype('float64')),
 ('X23', 5, dtype('float64')),
 ('Y', 3, dtype('float64'))]
```

Se hace copia del dataframe y se eliminan los registros que tengan la variable Y en null

[36]:
```
data2 = df.copy()
# Remover registros que no tengan valor en la columna 'Y'
data2 = data2.dropna(subset=['Y'])
```

Verificar el tipo de dato de cada columna y el numero de datos vacíos

[37]:
```
getColumnInfo(data2)
```

[37]:
```
[('X1', 0, dtype('int64')),
 ('X2', 1, dtype('float64')),
 ('X3', 2, dtype('float64')),
 ('X4', 2, dtype('float64')),
 ('X5', 5, dtype('float64')),
 ('X6', 3, dtype('float64')),
 ('X7', 5, dtype('float64')),
```

```
 ('X8', 7, dtype('float64')),
 ('X9', 9, dtype('float64')),
 ('X10', 16, dtype('float64')),
 ('X11', 14, dtype('float64')),
 ('X12', 9, dtype('float64')),
 ('X13', 9, dtype('float64')),
 ('X14', 10, dtype('float64')),
 ('X15', 12, dtype('float64')),
 ('X16', 14, dtype('float64')),
 ('X17', 7, dtype('float64')),
 ('X18', 5, dtype('float64')),
 ('X19', 6, dtype('float64')),
 ('X20', 5, dtype('float64')),
 ('X21', 8, dtype('float64')),
 ('X22', 8, dtype('float64')),
 ('X23', 2, dtype('float64')),
 ('Y', 0, dtype('float64'))]
```

Se obtiene información del dataset como numero de registros, medidas de tendencia central, valores máximos, mínimos y cuartíles

[38]:
```
data2.describe()
```

[38]:

	X1	X2	X3	X4	X5	\
count	29997.000000	29996.000000	29995.000000	29995.000000	29992.000000	
mean	167496.072274	1.603781	1.853076	1.551925	35.483862	
std	129748.803871	0.489119	0.790343	0.521968	9.218114	
min	10000.000000	1.000000	0.000000	0.000000	21.000000	
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	
75%	240000.000000	2.000000	2.000000	2.000000	41.000000	
max	1000000.000000	2.000000	6.000000	3.000000	79.000000	

	X6	X7	X8	X9	X10	\
count	29994.000000	29992.000000	29990.000000	29988.000000	29981.000000	
mean	-0.016770	-0.133836	-0.166556	-0.220888	-0.266435	
std	1.123765	1.197187	1.195976	1.169139	1.133275	
min	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	
max	8.000000	8.000000	8.000000	8.000000	8.000000	

	…	X15	X16	X17	X18	\
count	…	29985.000000	29983.000000	29990.000000	29992.000000	
mean	…	43275.652326	40324.493980	38881.135745	5662.945886	
std	…	64345.500073	60809.984983	59561.312967	16564.165089	
min	…	-170000.000000	-81334.000000	-339603.000000	0.000000	

```
25%    …    2329.000000    1763.500000    1256.250000    1000.000000
50%    …   19052.000000   18107.000000   17081.000000    2100.000000
75%    …   54560.000000   50213.000000   49208.250000    5006.000000
max    …  891586.000000  927171.000000  961664.000000  873552.000000

                X19            X20            X21            X22  \
count  2.999100e+04   29992.000000   29989.000000   29989.000000
mean   5.922489e+03    5225.623400    4827.252526    4800.297209
std    2.304418e+04   17608.422625   15668.751975   15280.842069
min    0.000000e+00       0.000000       0.000000       0.000000
25%    8.355000e+02     390.000000     296.000000     251.000000
50%    2.009000e+03    1800.000000    1500.000000    1500.000000
75%    5.000000e+03    4505.500000    4014.000000    4033.000000
max    1.684259e+06  896040.000000  621000.000000  426529.000000

                X23            Y
count   29995.000000  29997.000000
mean     5216.259977      0.221189
std     17778.848359      0.415054
min         0.000000      0.000000
25%       118.000000      0.000000
50%      1500.000000      0.000000
75%      4000.000000      0.000000
max    528666.000000      1.000000

[8 rows x 24 columns]
```

Se muestra el número de registros y columnas de la variable data2

[39]:
```
data2.shape
```

[39]:
```
(29997, 24)
```

Se hace una copia del dataframe data2 para ver cuantos registros quedan si se hace un dropna de todas las columans

[40]:
```
data3 = data2.copy()
data3.dropna(inplace=True)
data3.shape
```

[40]:
```
(29958, 24)
```

Verificar el tipo de dato de cada columna y el numero de datos vacíos

[41]:
```
getColumnInfo(data3)
```

[41]:
```
[('X1', 0, dtype('int64')),
 ('X2', 0, dtype('float64')),
 ('X3', 0, dtype('float64')),
```

```
 ('X4', 0, dtype('float64')),
 ('X5', 0, dtype('float64')),
 ('X6', 0, dtype('float64')),
 ('X7', 0, dtype('float64')),
 ('X8', 0, dtype('float64')),
 ('X9', 0, dtype('float64')),
 ('X10', 0, dtype('float64')),
 ('X11', 0, dtype('float64')),
 ('X12', 0, dtype('float64')),
 ('X13', 0, dtype('float64')),
 ('X14', 0, dtype('float64')),
 ('X15', 0, dtype('float64')),
 ('X16', 0, dtype('float64')),
 ('X17', 0, dtype('float64')),
 ('X18', 0, dtype('float64')),
 ('X19', 0, dtype('float64')),
 ('X20', 0, dtype('float64')),
 ('X21', 0, dtype('float64')),
 ('X22', 0, dtype('float64')),
 ('X23', 0, dtype('float64')),
 ('Y', 0, dtype('float64'))]
```

Se cambia de float a int los datos de la variable de salida implementando una función lambda

[42]:
```
# La variable Y se tiene que cambiar el tipo de dato de float a integer
data2['Y'] = data2['Y'].apply(lambda x: int(x))
```

Se crea la función converColumns para cambiar el tipo de dato en las variables que son de tipo flotante

[64]:
```
def convertColumns(df: pd.DataFrame, columns: list) -> pd.DataFrame:
    """
    Esta funcion no crea una copia del dataset que se pasa como argumento
    """
    for i in columns:
        df[i] = df[i].apply(lambda x: int(x))
    return df
```

Se considera eliminar los registros que no tengan valor en las variables X15 a la X23, sería bueno eliminar por el treshhold porque algunos registros tienen muchos datos vacíos, los registros restantes que tengan valores Na se puede considerar hacer imputación de medias para no afectar tanto al análisis, un buen thresh para empezar sería 15

[59]:
```
data4 = data2.copy()
data4.dropna(thresh=14, inplace=True)
data4.shape
```

[59]:
```
(29991, 24)
```

Se realiza la imputación de la mediana en todos los registros que estén vacíos

[60]:
```
# Realizando la imputación con el valor de la mediana de cada variable
data4 = data4.fillna(value=data4.median()).copy()
```

Convertir las columnas a tipo entero ya que si se desean obtener columnas dummies puede ser un problema tener tipo de dato float

[65]:
```
columns_2_int = ['X2', 'X3', 'X4', 'X5','X6', 'X7', 'X8', 'X9', 'X10', 'X11']
convertColumns(data4, columns_2_int)
```

[65]:

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	…	X15	X16	\
0	20000	2	2	1	24	2	2	-1	-1	-2	…	0.0	0.0	
1	120000	2	2	2	26	-1	2	0	0	0	…	3272.0	3455.0	
2	90000	2	2	2	34	0	0	0	0	0	…	14331.0	14948.0	
3	50000	2	2	1	37	0	0	0	0	0	…	28314.0	28959.0	
4	50000	1	2	1	57	-1	0	-1	0	0	…	20940.0	19146.0	
…	…	..	..	..	..	..	..	..	..	…	…	…	…	
29995	220000	1	3	1	39	0	0	0	0	0	…	88004.0	31237.0	
29996	150000	1	3	2	43	-1	-1	-1	-1	0	…	8979.0	5190.0	
29997	30000	1	2	2	37	4	3	2	-1	0	…	20878.0	20582.0	
29998	80000	1	3	1	41	1	-1	0	0	0	…	52774.0	11855.0	
29999	50000	1	2	1	46	0	0	0	0	0	…	36535.0	32428.0	

	X17	X18	X19	X20	X21	X22	X23	Y
0	0.0	0.0	689.0	0.0	0.0	0.0	0.0	1
1	3261.0	0.0	1000.0	1000.0	1000.0	0.0	2000.0	1
2	15549.0	1518.0	1500.0	1000.0	1000.0	1000.0	5000.0	0
3	29547.0	2000.0	2019.0	1200.0	1100.0	1069.0	1000.0	0
4	19131.0	2000.0	36681.0	10000.0	9000.0	689.0	679.0	0
…	…	…	…	…	…	…	..	
29995	15980.0	8500.0	20000.0	5003.0	3047.0	5000.0	1000.0	0
29996	0.0	1837.0	3526.0	8998.0	129.0	0.0	0.0	0
29997	19357.0	0.0	0.0	22000.0	4200.0	2000.0	3100.0	1
29998	48944.0	85900.0	3409.0	1178.0	1926.0	52964.0	1804.0	1
29999	15313.0	2078.0	1800.0	1430.0	1000.0	1000.0	1000.0	1

[29991 rows x 24 columns]

Verificar el tipo de dato de cada columna y el numero de datos vacíos

[66]:
```
getColumnInfo(data4)
```

[66]:
```
[('X1', 0, dtype('int64')),
 ('X2', 0, dtype('int64')),
 ('X3', 0, dtype('int64')),
 ('X4', 0, dtype('int64')),
 ('X5', 0, dtype('int64')),
 ('X6', 0, dtype('int64')),
```

```
('X7', 0, dtype('int64')),
('X8', 0, dtype('int64')),
('X9', 0, dtype('int64')),
('X10', 0, dtype('int64')),
('X11', 0, dtype('int64')),
('X12', 0, dtype('float64')),
('X13', 0, dtype('float64')),
('X14', 0, dtype('float64')),
('X15', 0, dtype('float64')),
('X16', 0, dtype('float64')),
('X17', 0, dtype('float64')),
('X18', 0, dtype('float64')),
('X19', 0, dtype('float64')),
('X20', 0, dtype('float64')),
('X21', 0, dtype('float64')),
('X22', 0, dtype('float64')),
('X23', 0, dtype('float64')),
('Y', 0, dtype('int64'))]
```

1.4.3 Parte 3: Preparación de los datos

¿Qué datos considero mas importantes? ¿Por qué?

Todas las variables representan información útil, sin embargo, las que considero importantes son las variables del historial del pago (X6 - X11), el importe del estado de cuenta mes por mes (X12 - X17), el importe del pago anterior (X18 - X23) y la variable de salida Y.

Entre las variables a descartar son el ID porque no proporciona información relevante y en caso de utilizar modelos de aprendizaje supervisado es recomendado descartar las variables de gérno, sexo, estado marital y educación para evitar sesgos y entrenar modelo discriminatorios.

¿Se eliminaron o reemplazaron datos nulos? ¿Qué se hizo y por qué?

Primero se identificaron la cantidad de valores vacíos en cada columna y al ver que existían datos nulos en la columna Y al ser solamente dos se consideró eliminarlos para no meter ruido al algoritmo ya que al no saber el valor esperado esto se puede considerar como un valor atípico dentro de la categoría que se le asigne, sin embargo, también hay que considerar que solamente eran tres registros y puede que el error no se significativo.

Se eliminaron 6 registros gracias al umbral de 15 con el dropna para conservar la mayor cantidad de combinaciones posibles. En el caso de eliminar todos los datos nulos en cada registro se perdieron 39 registros.

Se realizo la imputación con el valor de la mediana ya que algunas variables X15 a la X23 su valor medio es muy grande debido a la escala de los datos por lo que si se imputa con la media la distribución puede ser mas abierta y pasar como datos de otra categoría.

¿Es necesario ordenar los datos para el análisis? Sí / No / ¿Por qué?

Bueno esto va a depender del análisis que se vaya a realizar, considero que para realizar un análisis exploratorio no es necesario, ya que en caso de visualizar los datos mediante un boxplot o por un histograma la librería matplotlib se encarga calcular las medidas de tendencia central y los cuartiles.

Al igual que si se busca dividir los datos para implementar un sistema de aprendizaje automático se tomarán los registros de manera aleatoria.

¿Existen problemas de formato que deban solucionar antes del proceso de modelado? Sí / No / Por qué.

En el caso de las variables categóricas como X2, X3, X4, X5, X6, X7, X8, X9, X10, X11 y Y se tuvo que convertir a tipo de dato int ya que por defecto estaban como float, esto podría causar errores al momento de utilizar algún otro método como get_dummies de pandas. Para las demas variables de tipo float se converaron como estaban en la fuente.

¿Qué ajustes se realizaron en el proceso de limpieza de datos (agregar, integrar, eliminar, modificar registros (filas), cambiar atributos (columnas)?

- Se eliminaron los registros que no tuvieran la variables de salida Y, también se removieron registros que no tuvieran al menos 14 valores
- Se cambiaron los tipos de datos de las variables categóricas y de la variable de salida a Int porque estaban en float
- Se realizó imputación de datos con el valor de la mediana a las columnas X15 a la X23 ya que el valor medio era muy alto y podía causar ruido en los registros.