Semana 3 - Actividad 1

Limpieza de datos

Lázaro Lara Martínez. Matricula A01793198

José Mtanous Treviño. Matricula A00169781

Ciencia y Analítica de datos.

Profesor Titular. Jobish Vallikavungal Devassia

Profesor Tutor. Mtro. Mario Alberto Solano Saldaña

01/Octubre/2022

Fundamentos de bases de datos y para ciencia de datos.

¿Qué es una base de datos?:

Es un sistema capaz de almacenar gran cantidad de datos que tienen relación entre sí y están estructurados. Sus datos pueden ser consultados rápidamente utilizando un lenguaje estructurado de consultas llamado SQL. En una base de datos existen diferentes tipos de objetos como son tablas las cuales contienen columnas, filas y celdas. Los registros en una tabla son los renglones los cuales contienen diferentes campos o columnas, las celdas contienen un dato de dicha columna y renglón. También existen vistas las cuales pueden unir lógicamente diferentes tablas, índices para hacer más rápidas las consultas, triggers para ejecutar acciones antes o después de insertar, actualizar o borrar datos y constraints para restringir el contenido de los datos.

Las bases de datos utilizan el paradigma de ETL, en donde los datos se depuran y organizan antes de cargarlos, este proceso puede ser costoso para fuentes de datos muy grandes.

Existen distintas aplicaciones para las bases de datos como bancos, aerolíneas, universidades, ventas, tiendas en línea, RH etc. Las Bases de datos más utilizadas son, Oracle MySQL, PostgreSQL y MS SQL Server.

¿Qué es un Data Warehouse?

Es un sistema de almacenamiento de datos que permite a las empresas comprender y utilizar sus datos para tomar decisiones estratégicas, generalmente los Data Warehouse contienen datos no estructurados o con estructura mínima (particiones), a diferencia de las bases de datos tradicionales, esta arquitectura de almacenamiento utiliza el paradigma ELT, en donde los datos se cargan sin depurarse, incluso muchas veces en un formato ‘crudo’ raw. El consumidor de los datos

es el encargado de depurarlos y transformarlos en algo que haga sentido para el contexto en que se usarán.

Estos sistemas de Data Warehouse generalmente se implementan en hardware de propósito general y están diseñados para ser redundantes y ejecutar tareas en paralelo (MapReduce), la estructura de datos que almacenan están optimizadas para lectura y es muy costoso actualizar de

Selección y limpieza de los Datos en Python

Se importan las librerías Pandas y numpy

```
import pandas as pd
import numpy as np
```

Primero obtenemos los datos frescos desde el archivo csv Y mostramos los primeros 5 renglones.

```
myDataSetUrl = 'https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/d
ndf = pd.read_csv(myDataSetUrl, index_col=0)
ndf.head(5)
```

ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	
1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	...	0.0	0.0	0.0	
2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	0.0	...	3272.0	3455.0	3261.0	
3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1
4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2
5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	0.0	...	20940.0	19146.0	19131.0	2

5 rows × 24 columns

```
# Describimos como vienen los datos.
ndf.describe()
```

	X1	X2	X3	X4	X5	X
count	30000.000000	29999.000000	29998.000000	29998.000000	29995.000000	29997.0000
mean	167484.322667	1.603753	1.853057	1.551903	35.484214	-0.0166
std	129747.661567	0.489125	0.790320	0.521968	9.218024	1.1238
min	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.0000
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.0000
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.0000
75%	240000.000000	2.000000	2.000000	2.000000	41.000000	0.0000

```
# Copiamos el dataframe original para no sobre escribirlo.

df = ndf.copy()
```

```
# Obtenemos las llaves
df.keys()
```

```
Index(['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10', 'X11',
       'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20', 'X21',
       'X22', 'X23', 'Y'],
      dtype='object')
```

```
# Varias formas de checar los valores NaN, NA y NULL
#df.isnull()
df.isnull().sum()
# df.isnull().values.any()
# df.isnull().values.sum()
# df.isnull().any()
```

```
X1      0
X2      1
X3      2
X4      2
X5      5
X6      3
X7      5
X8      7
X9      9
X10    16
X11    14
X12    11
X13    11
X14    13
X15    15
X16    17
X17    10
X18     8
X19     9
```

```
X20     8
X21    11
X22    11
X23     5
Y       3
dtype: int64
```

```
#Con esta instrucción obtenemos la información del dataset y vemos que todos los datos son nu
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30000 entries, 1 to 30000
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X1      30000 non-null  int64  
 1   X2      29999 non-null  float64
 2   X3      29998 non-null  float64
 3   X4      29998 non-null  float64
 4   X5      29995 non-null  float64
 5   X6      29997 non-null  float64
 6   X7      29995 non-null  float64
 7   X8      29993 non-null  float64
 8   X9      29991 non-null  float64
 9   X10     29984 non-null  float64
 10  X11     29986 non-null  float64
 11  X12     29989 non-null  float64
 12  X13     29989 non-null  float64
 13  X14     29987 non-null  float64
 14  X15     29985 non-null  float64
 15  X16     29983 non-null  float64
 16  X17     29990 non-null  float64
 17  X18     29992 non-null  float64
 18  X19     29991 non-null  float64
 19  X20     29992 non-null  float64
 20  X21     29989 non-null  float64
 21  X22     29989 non-null  float64
 22  X23     29995 non-null  float64
 23  Y       29997 non-null  float64
dtypes: float64(23), int64(1)
memory usage: 5.7 MB
```

```
#Existen datos nulos?

df.isnull().values.any()
```

```
True
```

```
#Que campos tienen datos nulos?
df.isnull().any()
```

```
X1     False
X2      True
```

```
X3      True
X4      True
X5      True
X6      True
X7      True
X8      True
X9      True
X10     True
X11     True
X12     True
X13     True
X14     True
X15     True
X16     True
X17     True
X18     True
X19     True
X20     True
X21     True
X22     True
X23     True
Y       True
dtype: bool
```

```
df.count()
```

```
X1     30000
X2     29999
X3     29998
X4     29998
X5     29995
X6     29997
X7     29995
X8     29993
X9     29991
X10    29984
X11    29986
X12    29989
X13    29989
X14    29987
X15    29985
X16    29983
X17    29990
X18    29992
X19    29991
X20    29992
X21    29989
X22    29989
X23    29995
Y      29997
dtype: int64
```

```
df.dropna(inplace = True)
```

```
df.count()
```

```
X1     29958
X2     29958
X3     29958
X4     29958
X5     29958
X6     29958
X7     29958
X8     29958
X9     29958
X10    29958
X11    29958
X12    29958
X13    29958
X14    29958
X15    29958
X16    29958
X17    29958
X18    29958
X19    29958
X20    29958
X21    29958
X22    29958
X23    29958
Y      29958
dtype: int64
```

```
#Localizamos los valores nulos del Dadaframe original y comparamos que en el nuevo ya se borr
null_columns=ndf.columns[ndf.isnull().any()]
ndf[null_columns].isnull().sum()
print(ndf[ndf.isnull().any(axis=1)][null_columns].head())
```

```
            X2   X3   X4    X5   X6   X7   X8   X9  X10  X11  ...      X15      X16  \
ID                                                             ...
19   2.0  1.0  1.0  49.0  1.0 -2.0 -2.0 -2.0 -2.0 -2.0  ...      NaN      NaN
39   1.0  1.0  2.0  25.0  1.0 -1.0 -1.0 -2.0 -2.0 -2.0  ...      0.0      0.0
50   1.0  1.0  2.0  24.0  0.0  0.0  0.0  0.0  NaN  0.0  ...  19865.0  20480.0
65   2.0  2.0  1.0  51.0 -1.0 -1.0 -2.0 -2.0 -1.0 -1.0  ...      0.0   2353.0
161  1.0  1.0  2.0  41.0  2.0  2.0  2.0  NaN  2.0  0.0  ...  28168.0  27579.0

          X17     X18     X19     X20     X21     X22     X23    Y
ID
19        NaN     0.0     0.0     0.0     0.0     0.0     0.0  0.0
39        NaN   780.0     0.0     0.0     0.0     0.0     0.0  1.0
50   20063.0  1318.0  1315.0   704.0   928.0   912.0  1069.0  0.0
65       0.0     0.0     NaN     0.0  2353.0     0.0     0.0  0.0
161  28321.0  3500.0     0.0  2200.0     NaN  1200.0  1250.0  0.0

[5 rows x 23 columns]
```

```
null_columns=df.columns[df.isnull().any()]
df[null_columns].isnull().sum()
```

```
print(df[df.isnull().any(axis=1)][null_columns].head())
```

```
Empty DataFrame
Columns: []
Index: []
```

```
#De nuevo mostramos los primeros 5 datos.
df.head(5)
```

ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	...	X15	X16	X17	
1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	-2.0	...	0.0	0.0	0.0	
2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	0.0	...	3272.0	3455.0	3261.0	
3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1
4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2
5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	0.0	...	20940.0	19146.0	19131.0	2

5 rows × 24 columns

```
# y la info resultante.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29958 entries, 1 to 30000
Data columns (total 24 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X1      29958 non-null  int64  
 1   X2      29958 non-null  float64
 2   X3      29958 non-null  float64
 3   X4      29958 non-null  float64
 4   X5      29958 non-null  float64
 5   X6      29958 non-null  float64
 6   X7      29958 non-null  float64
 7   X8      29958 non-null  float64
 8   X9      29958 non-null  float64
 9   X10     29958 non-null  float64
 10  X11     29958 non-null  float64
 11  X12     29958 non-null  float64
 12  X13     29958 non-null  float64
 13  X14     29958 non-null  float64
 14  X15     29958 non-null  float64
 15  X16     29958 non-null  float64
 16  X17     29958 non-null  float64
 17  X18     29958 non-null  float64
 18  X19     29958 non-null  float64
 19  X20     29958 non-null  float64
 20  X21     29958 non-null  float64
 21  X22     29958 non-null  float64
```

```
 22  X23     29958 non-null  float64
 23  Y       29958 non-null  float64
dtypes: float64(23), int64(1)
memory usage: 5.7 MB
```

Decidimos borrar los datos nulos porque no tiene sentido desde el punto de vista del negocio asignar valores de media, mediana o moda diferentes clientes además, se borraron solo 42 registros del total de 30,000.

```
## Obtener los Valores diferentes en los datos de Género para corroborar que sea consistente.
# X2: Gender (1 = male; 2 = female). Todo se ve bien.

df.X2.unique()
```

```
array([2., 1.])
```

```
## Obtener los Valores diferentes en los datos de Género para corroborar que sea consistente.
# X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
# Podemos ver que existen valores fuera de lo esperado. 0, 5 y 6. El cero puede considerarse

df.X3.unique()
```

```
array([2., 1., 3., 5., 4., 6., 0.])
```

```
## Obtener los Valores diferentes en los datos de Género para corroborar que sea consistente.
# X4: Marital status (1 = married; 2 = single; 3 = others). Existe el valor 0 que no se encue

df.X4.unique()
```

```
array([1., 2., 3., 0.])
```

Sección 3

1. ¿Qué datos considero mas importantes?

   Para este tipo de análisis todos los datos del archivo son necesarios, es primordial saber X1 el monto del crédito, X6 – X11 el historial de pagos, X12 – X17 monto del estado de cuenta, y X18 – X23 cantidad de los últimos pagos.

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

   Decidimos borrar los datos nulos porque no tiene sentido desde el punto de vista del negocio asignar valores de media, mediana o moda diferentes clientes. Además, se borraron solo 42 registros del total de 30,000, apoximadamente el 0.14%

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

   Desde el punto de vista estadístico no consideramos necesario ordenarlos, debido a que cada registro es independiente del siguiente, no es necesario comparara un registro contra el siguiente. Desde el punto de vista de procesamiento tampoco consideramos que es necesario ordenarlos ya que el tamaño del conjunto de datos es muy pequeño y *cabe* en memoria, si fuera un conjunto enorme en donde tuvieramos que paralelizar el procesamiento podríamos pensar en ordenar los datos o partisionarlos, sin embargo para este conjunto no es necesario.

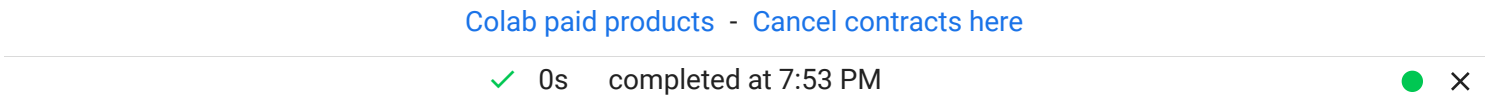
4. ¿Existen problemas de formato que deban solucionar antes del proceso de modelado? Sí / No / Por qué.

   No existen problemas de formato, los datos categóricos X2, X3 y X4 contienen datos numéricos como se indica en la información de los atributos. El único problema que veo en los datos categóricos es unos cuantos ceros que pueden considerarse nulos, pero no puede asignarse un valor arbitrario, ya que es el nivel de educación del cliente, o su estatus marital. En Educación existen valores no esperados, 5 y 6 que no vienen descritos en la información del atributo. Seria cuestión de investigar cuál es el significado de dichos valores.

   Todos los demás datos hablan de montos y son numéricos, los datos negativos tienen su significado en el negocio como por ejemplo X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005; . . .;X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.

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

   Eliminamos los datos que tenían nulos, debido a que no se pienso que sea correcto asignar valores como media a datos financieros de diferentes personas, no fue necesario hacer otro ajuste.


Colab paid products - Cancel contracts here
0s completed at 7:53 PM