

## ▼ Semana 3 - Actividad 1



# Tecnológico de Monterrey

Materia: Ciencia y analítica de datos

Nombre de la actividad: Semana 3 - Actividad 1

Nombre del alumno: Roberto Vega Alanis

Matrícula: A01378921

Fecha: 04 de Octubre de 2022

## ▼ Parte 1: Fundamentos de bases de datos

**Instrucciones:** Revisa la página Lecturas del tema 3: Conceptos de almacenamiento y recuperación de información, y describe tus insights o entendimiento de cada uno de los subtemas: (Alrededor de 200 palabras)

## ▼ Fundamentos de bases de datos y para ciencia de datos

De manera general con base en el contenido de la bibliografía consultada puedo llegar a las siguientes conclusiones.

En primer lugar, las bases de datos son formas organizadas de almacenar datos, sin embargo, esta organización puede ser de 2 tipos (3 en realidad si se consideran los sistemas semi-estructurados): no estructuradas y estructuradas, siendo estos últimos los más comunes aún hoy en día en las industrias. El caso de los datos no estructurados puede incluir, documentos de texto, imágenes, etc. Por otra parte, los datos estructurados son aquellos con una organización tabular: filas y columnas bien definidas.

Un subconjunto particular de este último tipo de datos son los sistemas de bases de datos relacionales (RDBMS por sus siglas en inglés) las cuales tienen identificadores tanto para sus filas (índices) como para sus columnas (encabezados). En pocas palabras son un conjunto de tablas muy bien definidas y relacionadas entre sí. Dichas tablas se les conoce como “entidad” y la conexión entre ellas se logra por medio de columnas comunes, es decir, columnas de diferentes tablas que hacen referencia a la misma información. Dichas relaciones se suelen expresar en diagramas de entidad-relación y pueden mostrar diferentes tipos de relaciones: uno a uno, uno a varios, varios a varios. Cuando se menciona “varios” hace referencia a que los valores de la columna utilizados para la relación pueden aparecer más de una vez (puede haber más de un valor del mismo tipo), mientras que cuando se menciona “uno” hace referencia a que los valores de la columna utilizados para la relación aparecen una sola vez en la misma.

## ▼ Fundamentos de almacenes de datos (Data Warehouse) para ciencia de datos

Con base en el contenido consultado se llegaron a los siguientes puntos referentes a los data warehouses. En primer lugar, se define de manera simple a los data warehouses como una base de datos enorme (un conjunto de “tablas” interrelacionadas entre sí) en el que se conjuntan e interconectan datos de diversas fuentes. Existen varios “esquemas” sobre los cuales pueden ser construidas, siendo un esquema una forma genérica de cómo interconectar todas las fuentes de datos. Muchas veces los data warehouses pueden manejar datos estructurados en forma de bases de datos relacionales, en donde los datos suelen ser conectados por esquemas como el “estrella” en el que básicamente consiste en la unión de una tabla de datos llamada “tabla de hechos” con una cantidad de tablas conocidas como “dimensionales”. Estas últimas, brindan información adicional acerca de un aspecto específico

de la tabla de hechos. Las tablas de hechos por su parte suele registrar “transacciones”, mismas que se pueden traducir con un registro de acciones. Para este tipo de warehouses, es común que se siga un esquema “ETL”, que significa “extraer, transformar, cargar”, en el cual los datos son extraídos de fuentes operacionales, son transformados en forma de datos estructurados y posteriormente son cargados al warehouse. Existe otro tipo particular de warehouse el cual utiliza técnicas diferentes para cargar información cuando el volumen de los datos es excesivo incluso para su procesamiento. Este proceso se le conoce como “ELT” (que significa “extraer, cargar, transformar”), el cual se enfoca en primero extraer los datos de la fuente para enseguida cargarlos a un repositorio en forma de datos en bruto (sin procesar) y, posteriormente, se usa el poder del repositorio para procesar la información. Esta actividad se distribuye en una importante cantidad de repositorios los cuales procesan la información de forma distribuida, aumentando la capacidad de escalar tanto la extracción como el procesamiento de los datos.

## ▼ Parte 2: Selección y limpieza de los Datos en Python

### **Instrucciones:**

Para esta actividad usa el base de datos encontrados por [aquí](#) Enlaces a un sitio externo.: su descripción [aquí](#) Enlaces a un sitio externo..

Revisa detenidamente la página Ejercicio guiado para: Selección y limpieza de los Datos en Python [Enlaces a un sitio externo.](#) En Google Colab (= Jupyter notebook, o bien alguna otra IDE de su interés), escribe tu código para realizar la selección y limpieza de los datos como se indica en el ejercicio. Ejecuta tu código. Explica con tus palabras (documentas las líneas del código en celdas del Text) como funciona tu programa indicando lo que realizaste en la programación, en minimum palabras.

```
# Importar bases de datos
import pandas as pd
import numpy as np
```

```
# Obteniendo link de GitHub con csv de datos
url = 'https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/default%20of%20credit%20card%20clients.csv'
```

```
df = pd.read_csv(url)
df
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	0.0	0.0	0.0	0.0	689.0	0.0	
1	2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	1
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	1
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	1
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	15980.0	8500.0	20000.0	5003.0	3
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0	0.0	1837.0	3526.0	8998.0	
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0	19357.0	0.0	0.0	22000.0	4
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0	48944.0	85900.0	3409.0	1178.0	1
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	15313.0	2078.0	1800.0	1430.0	1

30000 rows × 25 columns



```
# Revisamos contenido general del dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
#   Column  Non-Null Count  Dtype
---  -
#   Column  Non-Null Count  Dtype
```

0	ID	30000	non-null	int64
1	X1	30000	non-null	int64
2	X2	29999	non-null	float64
3	X3	29998	non-null	float64
4	X4	29998	non-null	float64
5	X5	29995	non-null	float64
6	X6	29997	non-null	float64
7	X7	29995	non-null	float64
8	X8	29993	non-null	float64
9	X9	29991	non-null	float64
10	X10	29984	non-null	float64
11	X11	29986	non-null	float64
12	X12	29989	non-null	float64
13	X13	29989	non-null	float64
14	X14	29987	non-null	float64
15	X15	29985	non-null	float64
16	X16	29983	non-null	float64
17	X17	29990	non-null	float64
18	X18	29992	non-null	float64
19	X19	29991	non-null	float64
20	X20	29992	non-null	float64
21	X21	29989	non-null	float64
22	X22	29989	non-null	float64
23	X23	29995	non-null	float64
24	Y	29997	non-null	float64

dtypes: float64(23), int64(2)

memory usage: 5.7 MB

```
# Analizamos la distribución de los datos al todos ser numéricos
df.describe()
```

	ID	X1	X2	X3	X4	X5	X6	X7
<b>count</b>	30000.000000	30000.000000	29999.000000	29998.000000	29998.000000	29995.000000	29997.000000	29995.000000
<b>mean</b>	15000.500000	167484.322667	1.603753	1.853057	1.551903	35.484214	-0.016635	-0.133689
<b>std</b>	8660.398374	129747.661567	0.489125	0.790320	0.521968	9.218024	1.123829	1.197254
<b>min</b>	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000
<b>25%</b>	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000
<b>50%</b>	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000

```
# Verifica si hay registros con al menos un valor vacío
df.isnull().values.any()
```

```
True
```

```
# Revisamos en cuál de las columnas hay valores vacíos
df.isnull().any()
```

```
ID      False
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
```

Se observa que en todas salvo en X1, existen valores vacíos

```
# Código equivalente al anterior de las últimas 2 celdas
```

```
# Algún valor nulo (por lo menos una celda vacía)
```

```
print(df.isna().values.any())
```

```
# Columnas con valores nulos
```

```
print(df.isna().any())
```

```
True
```

```
ID      False
```

```
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
```

```
# Contamos cuantos registros son nulos en cada columna
```

```
df[df.isnull().any(axis = 1)].count()
```

```
ID      42
X1       42
X2       41
X3       40
X4       40
X5       37
X6       39
X7       37
X8       35
X9       33
X10      26
X11      28
X12      31
X13      31
X14      29
X15      27
X16      25
X17      32
X18      34
X19      33
X20      34
X21      31
X22      31
X23      37
Y        39
dtype: int64
```

```
# Proporción de datos nulos por columna (en escala de 0 a 1)
```

```
df[df.isnull().any(axis = 1)].count()/len(df)
```

```
ID      0.001400
```



```
X1      0.001400
X2      0.001367
X3      0.001333
X4      0.001333
X5      0.001233
X6      0.001300
X7      0.001233
X8      0.001167
X9      0.001100
X10     0.000867
X11     0.000933
X12     0.001033
X13     0.001033
X14     0.000967
X15     0.000900
X16     0.000833
X17     0.001067
X18     0.001133
X19     0.001100
X20     0.001133
X21     0.001033
X22     0.001033
X23     0.001233
Y       0.001300
dtype: float64
```

```
# Evaluamos la cantidad de filas que serían afectadas por la eliminación de datos faltantes
print('Se mantiene el ', str((len(df.dropna())/len(df))*100) + '%', ' de los datos')
```

```
Se mantiene el  99.86%  de los datos
```

Al ser 0.13% la cantidad de los registros que contiene al menos un dato faltante, es probable que la información faltante sea por una causa completamente aleatoria (MCAR) y, por lo tanto, la eliminación por lista sea la solución más adecuada.

Sin embargo, antes de proceder a la eliminación de filas, se decidió analizar las columnas categóricas expresadas en valores numéricos discretos

```
# Con base en la metadata, determinamos cuales son variables categóricas
columnas_categ = ['X2','X3', 'X4', 'X5']

# Convertimos variables numéricas a categóricas como corresponde (cambio de tipo de dato)
df_cat = df.copy()
for col in columnas_categ:
    df_cat[col] = df_cat[col].astype('category')
df_cat
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	0.0	0.0	0.0	0.0	689.0	0.0	
1	2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	1
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	1
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	1
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	15980.0	8500.0	20000.0	5003.0	3
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0	0.0	1837.0	3526.0	8998.0	
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0	19357.0	0.0	0.0	22000.0	4
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0	48944.0	85900.0	3409.0	1178.0	1
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	15313.0	2078.0	1800.0	1430.0	1

30000 rows × 25 columns



```
# Revisamos la distribución de los valores categóricos a detalle paar X2
df_cat['X2'].value_counts()
```

```
2.0    18112
1.0    11887
Name: X2, dtype: int64
```

La variable X2 (género) tiene valores correctos y clases ligerameente desbalanceadas

```
# Revisamos la distribución de los valores categóricos a detalle paar X3
df_cat['X3'].value_counts()
```

```
2.0    14030
1.0    10585
3.0     4915
5.0      280
4.0     123
6.0       51
0.0       14
Name: X3, dtype: int64
```

Los valores 0, 5 y 6 son valores inválidos para la columna X3 (Educación), deben tratarse.

```
# Revisamos la distribución de los valores categóricos a detalle paar X4
df_cat['X4'].value_counts()
```

```
2.0    15964
1.0    13657
3.0      323
0.0       54
Name: X4, dtype: int64
```

Los valores de 0 para la variable X4 (Estado civil) no están clasificados por lo que no es correcto asignarlos

```
# Revisamos la distribución de los valores categóricos a detalle paar X5
df_cat['X5'].value_counts()
```



29.0	1605
27.0	1477
28.0	1409
30.0	1395
26.0	1256
31.0	1217
25.0	1186
34.0	1162
32.0	1158
33.0	1146
24.0	1127
35.0	1113
36.0	1108
37.0	1041
39.0	954
38.0	943
23.0	931
40.0	870
41.0	823
42.0	794
44.0	700
43.0	669
45.0	617
46.0	570
22.0	560
47.0	499
48.0	466
49.0	452
50.0	411
51.0	340
53.0	325
52.0	304
54.0	247
55.0	209
56.0	178
57.0	122
58.0	122
59.0	83
60.0	67
21.0	67
61.0	56
62.0	44

```
63.0    31
64.0    31
66.0    25
65.0    24
67.0    16
69.0    15
70.0    10
68.0     5
73.0     4
71.0     3
72.0     3
75.0     3
74.0     1
79.0     1
Name: X5, dtype: int64
```

El rango de valores de X5 (edad) parece correcto. No se requiere procesamiento de esta columna.

Procedemos a la eliminación de registros con valores erróneos al ser minoritarios

```
# Evaluamos el efecto de la eliminación de registros inválidos por las clases categóricas
df_cat[~df_cat['X3'].isin([0.0,5.0,6.0])]['X3'].value_counts()
```

```
2.0    14030
1.0    10585
3.0     4915
4.0     123
0.0       0
5.0       0
6.0       0
Name: X3, dtype: int64
```

```
# Quitamos los registros que pertenecen a las clases inexistentes para columna X3
df_cat = df_cat[~df_cat['X3'].isin([0.0,5.0,6.0])]
```

```
# Evaluamos el efecto de la eliminación de registros inválidos por las clases categóricas
df_cat[~df_cat['X4'].isin([0.0])]['X4'].value_counts()
```

```
2.0    15806
1.0    13475
3.0      318
0.0         0
Name: X4, dtype: int64
```

```
# Quitamos los registros que pertenecen a las clases inexistentes para columna X3
df_cat = df_cat[~df_cat['X4'].isin([0.0])]
```

```
# Evaluamos el tamaño del conjunto de datos limpiado
len(df_cat)
```

```
29601
```

```
# Evaluamos la cantidad de filas que serían afectadas por la eliminación de datos faltantes
print('Se mantiene el ', str((len(df_cat.dropna())/len(df))*100) + '%', ' de los datos')
```

```
Se mantiene el  98.53%  de los datos
```

Con esto se confirma que el efecto de la técnica seleccionada (eliminación por lista) es adecuada

Solución 1: Eliminar datos faltantes (de registros si existe al menos un valor faltante en un valor de cada fila)

```
# Observamos el efecto en la reducción de registros que tendría la eliminación de filas con la condición
len(df_cat.dropna())
```

```
29559
```

Nos percatamos de que las eliminaciones previas son consistentes con la eliminación por lista, diciéndonos que los registros con valores erróneos probablemente contenían además valores nulos. Esto refuerza la teoría de que los datos faltantes sean faltas completamente al azar (MCAR). Con lo que la eliminación por lista se considera como la mejor solución para el presente caso.

```
# Se eliminan las filas que contienen al menos un sólo valor vacío
df_sol_1 = df_cat.dropna()
```

df\_sol\_1

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	0.0	0.0	0.0	0.0	689.0	0.0	
1	2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	0.0	0.0	...	3272.0	3455.0	3261.0	0.0	1000.0	1000.0	1
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1000.0	1
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1200.0	1
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	19131.0	2000.0	36681.0	10000.0	9
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	15980.0	8500.0	20000.0	5003.0	3
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0	0.0	1837.0	3526.0	8998.0	
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0	19357.0	0.0	0.0	22000.0	4
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0	48944.0	85900.0	3409.0	1178.0	1
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	15313.0	2078.0	1800.0	1430.0	1

29559 rows × 25 columns



df\_sol\_1.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29559 entries, 0 to 29999
Data columns (total 25 columns):
#   Column  Non-Null Count  Dtype
#
```

```

---
0  ID      29559 non-null int64
1  X1      29559 non-null int64
2  X2      29559 non-null category
3  X3      29559 non-null category
4  X4      29559 non-null category
5  X5      29559 non-null category
6  X6      29559 non-null float64
7  X7      29559 non-null float64
8  X8      29559 non-null float64
9  X9      29559 non-null float64
10 X10     29559 non-null float64
11 X11     29559 non-null float64
12 X12     29559 non-null float64
13 X13     29559 non-null float64
14 X14     29559 non-null float64
15 X15     29559 non-null float64
16 X16     29559 non-null float64
17 X17     29559 non-null float64
18 X18     29559 non-null float64
19 X19     29559 non-null float64
20 X20     29559 non-null float64
21 X21     29559 non-null float64
22 X22     29559 non-null float64
23 X23     29559 non-null float64
24 Y       29559 non-null float64
dtypes: category(4), float64(19), int64(2)
memory usage: 5.1 MB

```

```
# Se comprueba que ya no se tienen valores nulos
```

```
print(df_sol_1.isna().values.any())
```

```
print(df_sol_1.isnull().values.any())
```

```
False
```

```
False
```

Solución 2: Imputar medida de tendencia central



```
# Imputamos el valor de la mediana
df_sol_2 = df.fillna(df.median())
```

La imputación de valor de la mediana, esta sería adecuada para los valores numéricos no categóricos ya que a partir de lo observado en el análisis estadístico del conjunto de datos (df.describe()), se observa que existen valores con un sesgo considerable en los datos, especialmente en las últimas variables (la mediana es la mitad de la media). De esta manera se evitaría un sesgo en los datos. Sin embargo, al ser mínima la cantidad de datos perdidos en relación al total de los datos se considera como mejor opción la solución 1.

```
# Análisis de valores numéricos "sesgados"
df_sol_1.loc[:, 'X18': 'X23'].describe()
```

	X18	X19	X20	X21	X22	X23
<b>count</b>	29559.000000	2.955900e+04	29559.000000	29559.000000	29559.000000	29559.000000
<b>mean</b>	5650.588315	5.899364e+03	5201.162725	4832.510877	4797.148753	5186.553537
<b>std</b>	16573.889188	2.310455e+04	17591.414381	15721.273274	15251.530482	17668.852466
<b>min</b>	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000
<b>25%</b>	1000.000000	8.270000e+02	390.000000	298.500000	259.000000	138.000000
<b>50%</b>	2100.000000	2.007000e+03	1800.000000	1500.000000	1500.000000	1500.000000
<b>75%</b>	5005.000000	5.000000e+03	4500.000000	4016.000000	4055.500000	4000.000000
<b>max</b>	873552.000000	1.684259e+06	896040.000000	621000.000000	426529.000000	528666.000000



```
# Transformación para reducir el sesgo en la distribución de los datos
```


```
df_sol_1.loc[:, 'X18': 'X23'] = np.sqrt(df_sol_1.loc[:, 'X18': 'X23'])
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1884: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/indexing.html#returning-self.\\_setitem\\_single\\_column\(loc, val, pi\)](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-self._setitem_single_column(loc, val, pi))



```
df_sol_1.loc[:, 'X18': 'X23'].describe()
```

	X18	X19	X20	X21	X22	X23	
<b>count</b>	29559.000000	29559.000000	29559.000000	29559.000000	29559.000000	29559.000000	
<b>mean</b>	55.022671	54.450029	50.152823	47.389427	47.251740	47.500199	
<b>std</b>	51.217017	54.172484	51.826132	50.860993	50.640978	54.133019	
<b>min</b>	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
<b>25%</b>	31.622777	28.757608	19.748418	17.277146	16.093477	11.747340	
<b>50%</b>	45.825757	44.799554	42.426407	38.729833	38.729833	38.729833	
<b>75%</b>	70.746025	70.710678	67.082039	63.371918	63.682808	63.245553	
<b>max</b>	934.640038	1297.790045	946.593894	788.035532	653.091877	727.094217	

Al parecer no hubo efecto notorio en la distribución de los datos.

## ▼ Parte 3: Preparación de los datos

**Instrucciones:** Con base en los resultados de tu libreta de Google Colab de la Parte 2 responde detalladamente las siguientes preguntas:

1. ¿Qué datos considero más importantes? ¿Por qué?

Considero que los datos más importantes dependerán en primer lugar del objetivo del negocio con respecto al uso de los datos. Si consideramos que los datos, según la descripción de la metadata, tienen como objetivo predecir la probabilidad de incumplimiento

de pago de un crédito, entonces la relevancia de los datos dependerá de la magnitud del impacto que tenga cada variable independiente (de entrada) dentro de un modelo generado para dicho fin (en el caso de los atributos). Para saber esto sería necesario realizar un análisis de componentes principales, el cual determinaría cuáles serían los atributos que tienden a explicar mejor el comportamiento de la variable objetivo. Dichos análisis podrían ser un PCA (con aprendizaje no supervisado) o una regresión logística regularizada. Por otra parte, si consideramos los datos en cuanto a su distribución a lo largo de los registros, considero que los datos más importantes

Nuevamente, siendo objetivos, estas suposiciones quedan subordinadas al objetivo del modelo; si el modelo busca detectar anomalías, ciertamente los outliers serían de mayor valor. Si por el contrario, el objetivo principal radica en el desarrollo de un modelo que se generalice mejor a los datos, los datos más importantes serían aquellos que tiendan a alinearse a la distribución propia de la naturaleza de los datos.

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

Con base en el análisis generado, se decidió que la mejor solución para el conjunto de datos sería eliminar los registros con valores nulos. Esta decisión se tomó en primer lugar debido al bajo impacto de la eliminación de los mismos sobre el conjunto de datos. Siendo que originalmente se contaban con 30 000 registros, el total de filas que serían eliminadas sólo sería de alrededor de 39, lo que representaría una pérdida del 0.13% del conjunto de los datos (un valor generalmente considerado despreciable dado el tamaño de la muestra).

Por otra parte, una vez realizada esta evaluación inicial, se procedió a realizar el análisis de las variables categóricas, dicho análisis arrojó 2 conclusiones importantes: • La primera de ellas era que para las variables “X3” y “X4”, existían valores no válidos, es decir, que no estaban etiquetados, mismos que, a pesar de no ser datos nulos debían de ser tratados. • Todos los registros con algún dato nulo estaban contenidos en su totalidad por el conjunto de registros con datos erróneos para alguna de las variables categóricas “X3” o “X4”, con lo que se llega a la conclusión de que los datos faltantes son datos faltantes completamente al azar (MCAR) pues es probable que hayan sido generados por errores en la medición con lo cual se concluye satisfactoriamente que la técnica de eliminación por lista es la adecuada para el presenta caso.

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

Siendo que el conjunto de datos no contiene en sus atributos elementos de una serie de tiempo, es poco probable que los datos adquieran alguna ventaja de este tipo de ordenamiento. Para el caso del análisis de la distribución de los datos y la obtención de sus

medidas de tendencia central (para el caso de valores numéricos) esto sólo sería de valor par obtener la mediana de los mismos, sin embargo, esto se puede obtener de manera directa mediante el método describe(). Adicionalmente, considero que no existiría un elemento o atributo que pudiera utilizarse como elemento de ordenamiento para los datos, al menos desde un punto de análisis numérico. En su lugar, la generación de gráficos de distribución (histogramas, boxplots, etc) brindarían mejor información de los datos.

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

Sí, en primer lugar, es necesario cambiar el tipo de dato de algunas de las variables utilizadas, ya que varias de ellas son consideradas como valores de punto flotante cuando son en realidad variables categóricas. Ese tipo de datos requeriría ya sea un cambio al tipo de dato de la columna a valor “category” o en su defecto sustituirla por un conjunto de columnas resultantes de la aplicación de una transformación de preprocesamiento “OneHotEncoding” (la generación de columnas binarias generadas por cada valor de la categoría de cada columna con variable categórica) con el fin de evitar la generación de una estimación jerárquica entre calores categóricos (donde muy probablemente no exista dicho tipo de relación). Adicionalmente, podría considerarse positivo cambiar el nombre de las columnas del conjunto de datos (por lo menos temporalmente) para facilitar el análisis del mismo. Adicionalmente, previo al modelado (considerando que este sería un modelo de predicción), sería de suma conveniencia realizar un escalamiento de los datos de tal manera que la diferencia en las magnitudes de los datos no genere efectos indeseados en la generación del modelo. Un ejemplo de esto sería que los valores relacionados con valores en dólares tuvieran mayor peso que la edad siendo el único factor el rango de valores utilizado en cada caso.

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

En primer lugar, como se mencionó anteriormente, la primera acción que se realizó como limpieza de los datos fue el cambio de tipo de dato de los atributos categóricos con el fin de evitar que estos fueran tratados como valores numéricos. Adicionalmente, después del análisis detallado en las preguntas anteriores, se procedió a la eliminación de registros cuyas variables categóricas contenían valores inválidos, así como aquellos registros que contenían algún valor nulo (siendo estos últimos subconjunto del primer caso).

Considero además que para las variables de X18 a X23 “Amount of previous payment”, los datos deberían intentar ser alineados, es decir, intentar reducir el sesgo que tienen entre su mediana y su media, pero con el fin de ayudar al cumplimiento de normalidad de

los valores de la variable objetivo, sin embargo, esta acción pareció no tener efecto esperado en la distribución de los datos ya que el sesgo de los mismos indica que dicho desbalanceo podría ser parte de la naturaleza de los mismos.

### Referencias:

- Géron, Aurélien (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly.
- Müller, A., Guido, S: (2016) Introduction to Machine Learning with Python, 1st Edition. O'Reilly

[Productos pagados de Colab](#) - [Cancela los contratos aquí](#)

✓ 0 s se ejecutó 21:17

