

Maestría en Inteligencia Artificial Aplicada

Ciencia y Analítica de Datos

Semana 3 Actividad 1

Curso: Ciencia y analítica de datos (Gpo 10)

Prof. Jobish Vallikavungal

1 de octubre de 2022

Equipo: 88

Nombre del estudiante:

Dalina Aidee Villa Ocelotl (A01793258)

Miguel Guillermo Galindo Orozco (A01793695)

▼ ***Limpieza de una base de Datos***

▼ **Parte 1: Fundamentos de bases de datos**

- ***Fundamentos de bases de datos en ciencia de datos:***

La ciencia de datos es parte de la evaluación del tratamiento integral de los datos. Dentro del proceso de conocimiento se describe el proceso como parte de la aplicación a resolución de problemas de la vida real con ***datos***. Actualmente se tienen una gran cantidad de datos que pueden venir desde distintas fuentes y además ser de distintos tipos de datos, estos después se transforman en ***información*** a través de aplicación de cambios a esos datos, normalizarlos e incluso tratar de describirlos a través de estadística descriptiva. Este proceso debe incluir la validación propia de la naturaleza de los datos ya que en la vida real siempre puede haber datos faltantes, o incluso datos que pudieran no tener sentido, desarrollando el proceso ETL. Por lo que, conocer el contexto de aplicación de la investigación es fundamental convirtiéndose en aplicación del método científico de manera multidisciplinaria. Seguido de eso se definen supuestos de acuerdo a la información por lo que se puede ofrecer una solución con estadística más avanzada como inferencia

estadística o modelos predictivos que pueden entrenarse con algoritmos de aprendizaje automático. Dentro de este proceso (evaluación, explicación, despliegue productivo y supervisión) se obtienen **insights o puntos clave** y además modelos productivos que ayuden a la toma de decisiones. Finalmente, este proceso de conocimiento nos permite ser más **sabios** respecto de los fenómenos que se monitorean inicialmente a través de datos.

- **Fundamentos de Almacenes de datos (Data Warehouse) para Ciencia de Datos**

Los modelos predictivos requieren de datos consistentes, claros y concisos para aplicar sus conocimientos estadísticos e informáticos que puedan considerarse robustos y precisos. Los Data Warehouse se convierten en los repositorios de información que cumpla con dichos supuestos para utilizarlos, al existir gran cantidad de datos se requiere de establecer una estructura a los datos que la tengan, corregir alguna o incluso establecer una para datos no estructurados que de alguna manera permita el fácil acceso a explotación de los mismos. De acuerdo a la forma del negocio o lugar donde se cree, se puede establecer los lineamientos para estandarizar la información e incluso plantear el correcto gobierno de los datos. Deben garantizar contener todos los datos, la parte del procesamiento, almacenamiento correcto, pero además la agilidad para poder explotarlos y garantizar la seguridad de la información.

- **Referencias:**

*Guía de visualización de datos para principiantes: definición, ejemplos y recursos de aprendizaje. (s/f). Tableau. Recuperado el 1 de octubre de 2022, de <https://www.tableau.com/es-mx/learn/articles/data-visualization> *¿Qué es la ciencia de datos? (s/f). Oracle.com. Recuperado el 1 de octubre de 2022, de <https://www.oracle.com/mx/what-is-data-science/> * <https://newoutlook.it/download/python/hands-on-data-science.pdf> *Data Warehouse: todo lo que necesitas saber sobre almacenamiento de datos. (s/f). Powerdata.Es. Recuperado el 1 de octubre de 2022, de <https://www.powerdata.es/data-warehouse> *texto en cursiva*

▼ **Insights**

- Una **base de datos** es una colección de información organizada de forma lógica cuyo objetivo es almacenar la descripción de un sistema particular del mundo real.

- **Data warehouse** (o almacén de datos) es un repositorio centralizado de las bases de datos, el cual es gestionado por un agente de control, almacenadas de una forma conveniente para el consumo y que se puede analizar para la óptima toma de decisiones.
- El objetivo de un **científico de datos es generar valor a través de los datos**, dónde para lograr este objetivo sigue el siguiente esquema:
 - Definición de problema.
 - Recolección y preparación de los datos.
 - Desarrollo de modelo.
 - Implementación del desarrollo.
 - Medición y continua mejora.
- Dentro del trabajo de un científico de datos es esencial **recolectar y limpiar los datos** ya que es a través de este proceso que permite obtener patrones en la información y que estos ayuden a la toma de decisiones. De no realizarse, la información obtenida podría no transmitir ningún insight que permita sacar conclusiones de provecho.
- Entre las principales tareas de limpieza de una base de datos se encuentran el **tratamiento de missing values (valores ausentes) y outliers (datos atípicos)**.
- Dentro de los valores ausentes, es importante entender la variable y las razones del dato ausente, ya que estos pueden ser por un error en el reporte que se presente de manera aleatoria, o que realmente la falta del dato esté aportando información para entender el patrón de comportamiento. Algunas herramientas para la limpieza de datos ausentes son: imputación de media o mediana, asignación por aproximación, o eliminación de registros.
- De igual forma, es importante entender los datos atípicos para su tratamiento, puede tratarse de un error en la captura de los datos, o de un comportamiento específico de la información. El tratamiento de estos puede ser a través de agrupar la información, disminuyendo el impacto de los mismos, o acotando su valor a un rango inter-cuartil que haga sentido dentro de la distribución de los datos. Finalmente, es importante considerar qué algoritmo se quiere entrenar para desarrollar el modelo, ya que algunos de ellos no necesariamente necesitan un tratamiento exhaustivo de outliers, por ejemplo un ensambles de árboles.

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

Documentación de funcionamiento del código

La intención del presente código es la limpieza de la base de datos vía replicar algunas de las técnicas y soluciones a problemáticas de los datos aprendidas en clase. Adicionalmente, se utilizarán técnicas que creo pertinentes aplicar desde mi experiencia con el trato de los datos.

La forma de realizar la limpieza será la siguiente:

1. Conocer el contenido general de los datos, el tipo de dato, y su distribución.
2. Aplicar limpieza y transformaciones necesarias a las variables derivado del conocimiento adquirido en el paso anterior. El tratamiento seleccionado será de acuerdo a la información que quiera transmitir cada variable. En caso de ser necesaria alguna transformación extra, se especifica en cada paso.

Nota: Se asume que la presente base de datos se utilizará para el desarrollo de un modelo, y que el contenido actual sólo representa el set de entrenamiento por lo que no es necesario particionar la base, ni existirá problemática de filtrado de información.

▼ Librerías a emplear

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

▼ Lectura de Data

```
df = pd.read_csv('https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendiza
```

Visualización de la data que acabamos de leer

```
df.head()
```

	ID	x1	x2	x3	x4	x5	x6	x7	x8	x9	...	x15	x16	x17	x18
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
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	3261.0
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	15149.0
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	20047.0
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	20047.0

5 rows x 25 columns

Conocer el tamaño de los datos que vamos a observar

```
df.shape # Código para conocer la estructura de la data

(30000, 25)
```

El data frame contiene 30,000 registros con 25 columnas

▼ Análisis general de los datos

Descripción breve y rápida de los datos

```
df.describe() ### Descripción del Dataframe
```

	ID	X1	X2	X3	X4	X5
count	30000.000000	30000.000000	29999.000000	29998.000000	29998.000000	29995.000000
mean	15000.500000	167484.322667	1.603753	1.853057	1.551903	35.484211
std	8660.398374	129747.661567	0.489125	0.790320	0.521968	9.218021
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000

8 rows x 25 columns

Conocer el total de missing values por columna

```
### Iteración por columna para conocer el total de missing values
for i in df.columns:
    print('Columna ' + str(i) + ' tiene ' + str(df[i].isna().sum()) + ' valores nulos, c
```

```
Columna ID tiene 0 valores nulos, que representa 0.0% del total
Columna X1 tiene 0 valores nulos, que representa 0.0% del total
Columna X2 tiene 1 valores nulos, que representa 0.0% del total
Columna X3 tiene 2 valores nulos, que representa 0.01% del total
Columna X4 tiene 2 valores nulos, que representa 0.01% del total
Columna X5 tiene 5 valores nulos, que representa 0.02% del total
Columna X6 tiene 3 valores nulos, que representa 0.01% del total
Columna X7 tiene 5 valores nulos, que representa 0.02% del total
Columna X8 tiene 7 valores nulos, que representa 0.02% del total
```

```

Columna X9 tiene 9 valores nulos, que representa 0.03% del total
Columna X10 tiene 16 valores nulos, que representa 0.05% del total
Columna X11 tiene 14 valores nulos, que representa 0.05% del total
Columna X12 tiene 11 valores nulos, que representa 0.04% del total
Columna X13 tiene 11 valores nulos, que representa 0.04% del total
Columna X14 tiene 13 valores nulos, que representa 0.04% del total
Columna X15 tiene 15 valores nulos, que representa 0.05% del total
Columna X16 tiene 17 valores nulos, que representa 0.06% del total
Columna X17 tiene 10 valores nulos, que representa 0.03% del total
Columna X18 tiene 8 valores nulos, que representa 0.03% del total
Columna X19 tiene 9 valores nulos, que representa 0.03% del total
Columna X20 tiene 8 valores nulos, que representa 0.03% del total
Columna X21 tiene 11 valores nulos, que representa 0.04% del total
Columna X22 tiene 11 valores nulos, que representa 0.04% del total
Columna X23 tiene 5 valores nulos, que representa 0.02% del total
Columna Y tiene 3 valores nulos, que representa 0.01% del total

```

Conocer el tipo de dato, valores únicos y algunos ejemplos del dato por columna:

```

## Iteración por columna para conocer el tipo de dato, valores únicos, y ejemplos del
for i in df.columns:
    print('Columna ' + str(i) + ' es un dato tipo ' + str(df.dtypes[i]) + ' con ' + str(

```

```

Columna ID es un dato tipo int64 con 30000 valores únicos. Algunos ejemplos son:
Columna X1 es un dato tipo int64 con 81 valores únicos. Algunos ejemplos son: [
Columna X2 es un dato tipo float64 con 2 valores únicos. Algunos ejemplos son: [
Columna X3 es un dato tipo float64 con 7 valores únicos. Algunos ejemplos son: [
Columna X4 es un dato tipo float64 con 4 valores únicos. Algunos ejemplos son: [
Columna X5 es un dato tipo float64 con 56 valores únicos. Algunos ejemplos son:
Columna X6 es un dato tipo float64 con 11 valores únicos. Algunos ejemplos son:
Columna X7 es un dato tipo float64 con 11 valores únicos. Algunos ejemplos son:
Columna X8 es un dato tipo float64 con 11 valores únicos. Algunos ejemplos son:
Columna X9 es un dato tipo float64 con 11 valores únicos. Algunos ejemplos son:
Columna X10 es un dato tipo float64 con 10 valores únicos. Algunos ejemplos son:
Columna X11 es un dato tipo float64 con 10 valores únicos. Algunos ejemplos son:
Columna X12 es un dato tipo float64 con 22718 valores únicos. Algunos ejemplos s
Columna X13 es un dato tipo float64 con 22339 valores únicos. Algunos ejemplos s
Columna X14 es un dato tipo float64 con 22020 valores únicos. Algunos ejemplos s
Columna X15 es un dato tipo float64 con 21541 valores únicos. Algunos ejemplos s
Columna X16 es un dato tipo float64 con 21000 valores únicos. Algunos ejemplos s
Columna X17 es un dato tipo float64 con 20600 valores únicos. Algunos ejemplos s
Columna X18 es un dato tipo float64 con 7941 valores únicos. Algunos ejemplos so
Columna X19 es un dato tipo float64 con 7899 valores únicos. Algunos ejemplos so
Columna X20 es un dato tipo float64 con 7517 valores únicos. Algunos ejemplos so
Columna X21 es un dato tipo float64 con 6936 valores únicos. Algunos ejemplos so
Columna X22 es un dato tipo float64 con 6895 valores únicos. Algunos ejemplos so
Columna X23 es un dato tipo float64 con 6939 valores únicos. Algunos ejemplos so
Columna Y es un dato tipo float64 con 2 valores únicos. Algunos ejemplos son: [1

```

▼ Análisis por columna

En esta sección se analiza por columnas o conjuntos de columna dependiendo de la información que transmitan las variables de acuerdo a la definición de las mismas. La definición de las variables es tomada del conjunto de datos que se encuentra en el mismo repositorio del ejercicio.

▼ Columna ID

Identificador único del registro.

Esta columna es la única que se trata por separado al no considerarse dentro de alguna de las definiciones anteriores.

Es una variable con el mismo número de valores únicos que el total de registros en el dataset. Van de 1 a 30,000.

No tiene missing values.

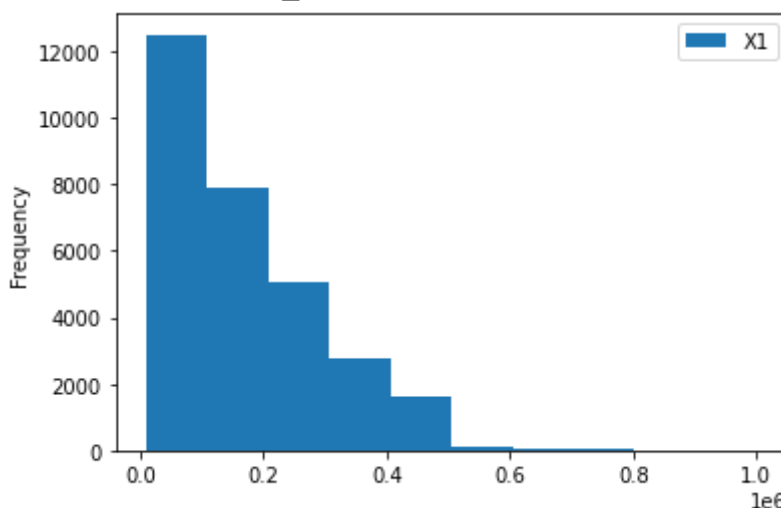
Sólo se decide modificar el tipo de columna de int a str para no tener problemas en lectura de la base en el futuro, o por cruces con otras bases de datos

```
df['ID'] = df['ID'].astype(str) ## Cambiar el tipo de dato a string
```

▼ Columna X1 -- Cantidad proporcionada en el crédito

```
df[['X1']].plot.hist() ## Plot de la distribución
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa96819e390>



No es necesario hacer transformación ya que el tipo de dato es int, no contiene missing values, y no presenta valores negativos.

▼ Columna X2 -- Género

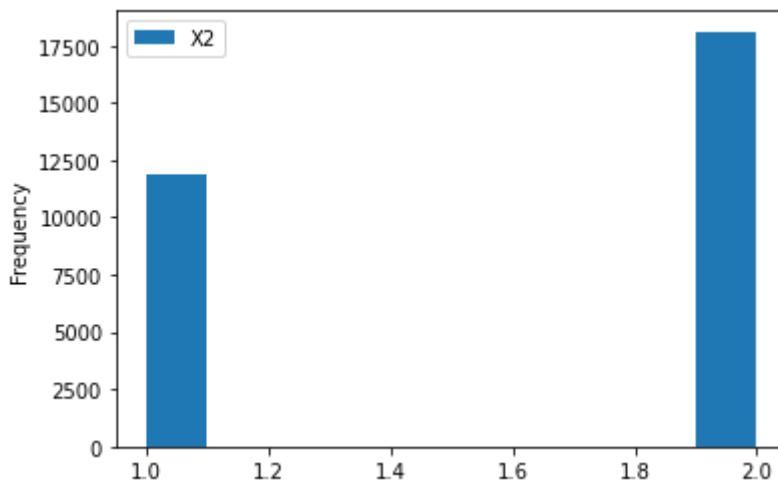
Columna categórica sobre el género de la persona.

Dónde 1 = Hombre, 2 = Mujer.

El tratamiento de missing values es asignando la moda.

```
df[['X2']].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa967c4ff90>



```
df[['X2']].groupby('X2').size().sort_values(ascending=False) ### observar valores más
```

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

```
valor = df['X2'].mode()[0] ## Guardar la moda
```

```
df['X2'] = df['X2'].fillna(value=valor) ## Asignación de missing values a la moda
```

▼ Columna X3 -- Educación

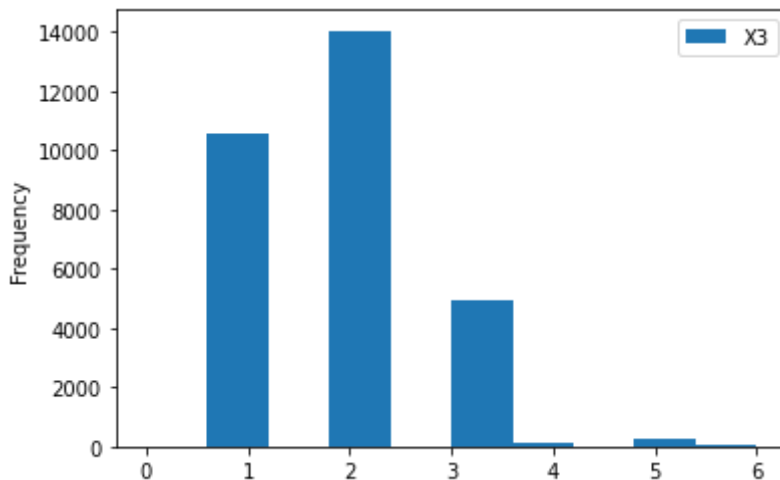
Columna categórica sobre el nivel de educación de la persona.

(1 = graduate school; 2 = university; 3 = high school; 4 = others).

El tratamiento de missing values (2 valores nulos) es asignando la moda.


```
df[['X3']].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa967b8f250>



```
df[['X3']].groupby('X3').size().sort_values(ascending=False) ### observar valores más
```

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

```
valor = df['X3'].mode()[0] ## Guardar la moda
```

```
df['X3'] = df['X3'].fillna(value=valor) ## Asignación de missing values a la moda
```

Adicionalmente se observa que existen categorías no definidas (5,6,0)

Se asignan dichos valores a la categoría 4 (otros)

```
df['X3'] = np.where((df['X3']==0) | (df['X3']==5) | (df['X3']==6), 4, df['X3']) ## Fur
```

▼ Columna X4 -- Estado Civil

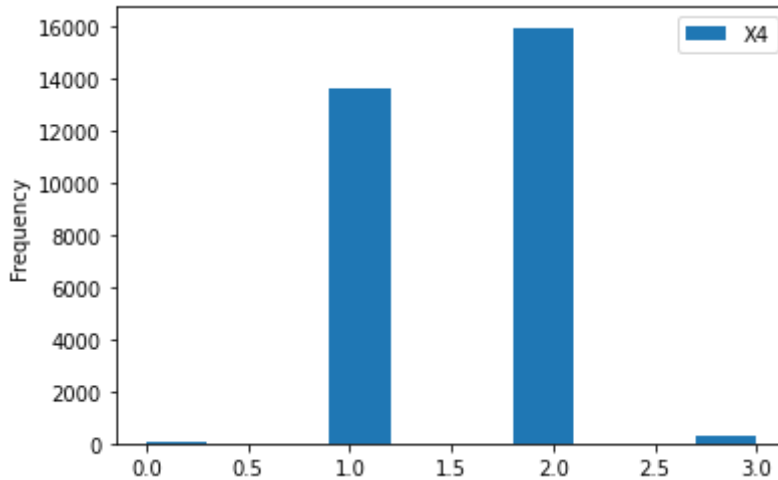
Columna categórica sobre el nivel de educación de la persona.

(1 = married; 2 = single; 3 = others)

El tratamiento de missing values (solo 2 valores nulos) es asignando la moda.

```
df[['X4']].plot.hist() ## Plot de la distribución
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa967b2cbd0>



```
df[['X4']].groupby('X4').size().sort_values(ascending=False) ### observar valores más
```

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

```
valor = df['X4'].mode()[0] ## Guardar la moda
```

```
df['X4'] = df['X4'].fillna(value=valor) ## Asignación de missing values a la moda
```

Adicionalmente se observa que existen categorías no definidas (0)

Se asignan dichos valores a la categoría 3 (otros)

```
df['X4'] = np.where((df['X4']==0) , 3, df['X4']) ## Función np.where que evalúa condic
```

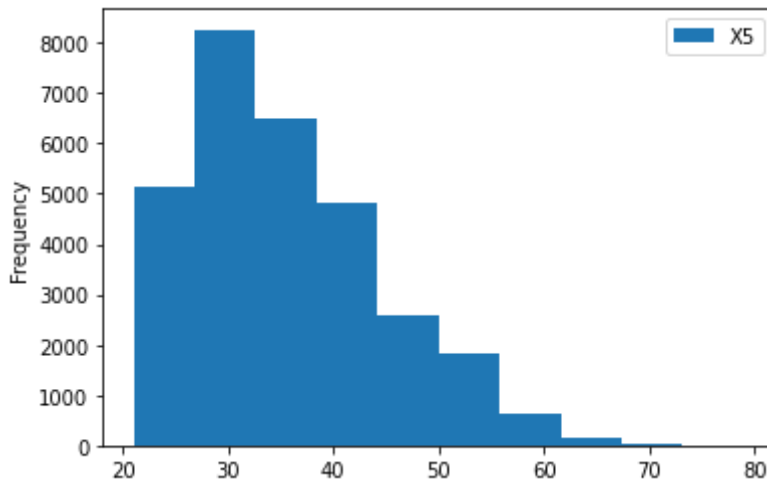
▼ Columna X5 -- Edad

Columna numérica de la edad de la persona.


Recordando la distribución.

```
df[['X5']].plot.hist() ## Plot de distribución
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa967ac1450>
```



```
df[['X5']].describe() # Descripción de la distribución de la variable
```

	X5 
count	29995.000000
mean	35.484214
std	9.218024
min	21.000000
25%	28.000000
50%	34.000000
75%	41.000000
max	79.000000

Observamos que la media y mediana no presentan diferencias significativas. Sólo son 5 valores nulos (0.002%) decide utilizar la media, si el % de missing values fuese mayor a 5% se tomarían otras medidas.

```
valor = df['X5'].mean() ## Almacenar el valor de la media
```

```
df['X5'] = df['X5'].fillna(value=valor) ## Asignación a valores nulos
```

▼ Columna X6 a X11 -- Histórico de pagos

Estatus de pago de los últimos 6 meses. Consideramos que es adecuado ya que para probabilidad de incumplimiento usualmente 6 meses es el mínimo de tiempo para comprobar el pago sostenido de un cliente.

Donde X6 = Sep 2005, X7=Ago 2005, ..., X11= Abr 2005.

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.

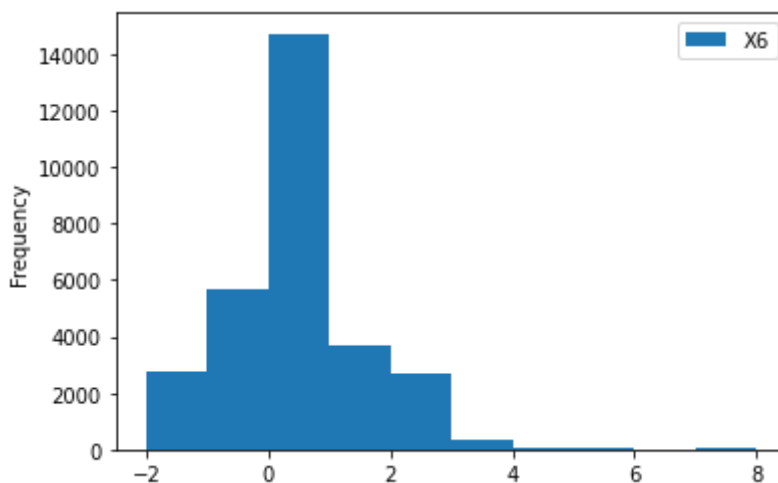
Las siguientes líneas de código desglozan el comportamiento por mes del estatus de pago

```
df[['X6']].groupby('X6').size().sort_values(ascending=False) ### observar valores más
```

```
X6
0.0    14736
-1.0    5684
1.0     3688
-2.0    2759
2.0     2667
3.0      322
4.0       76
5.0       26
8.0       19
6.0       11
7.0        9
dtype: int64
```

```
df[['X6']].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa9679da990>
```



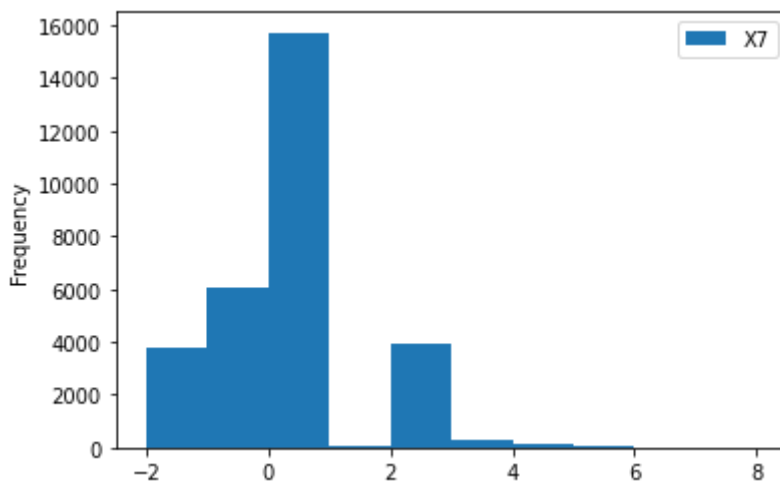
La mayoría de los datos están concentrados en categorías fuera de la descripción de la variable

```
df[['X7']].groupby('X7').size().sort_values(ascending=False) ### observar valores más
```

```
X7
0.0    15728
-1.0    6047
2.0    3927
-2.0    3782
3.0     326
4.0      99
1.0      28
5.0      25
7.0      20
6.0      12
8.0       1
dtype: int64
```

```
df[['X7']].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa9679637d0>
```



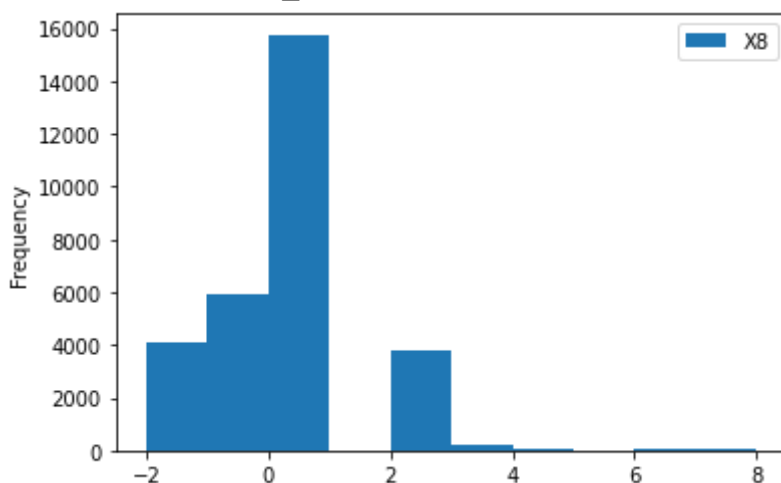
La mayoría de los datos están concentrados en categorías fuera de la descripción de la variable

```
df[['X8']].groupby('X8').size().sort_values(ascending=False) ### observar valores más
```

```
X8
0.0    15761
-1.0    5935
-2.0    4085
2.0    3819
3.0     240
4.0      76
7.0      27
6.0      23
5.0      21
1.0       4
8.0       2
dtype: int64
```

```
df[['X8']].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa9678d73d0>



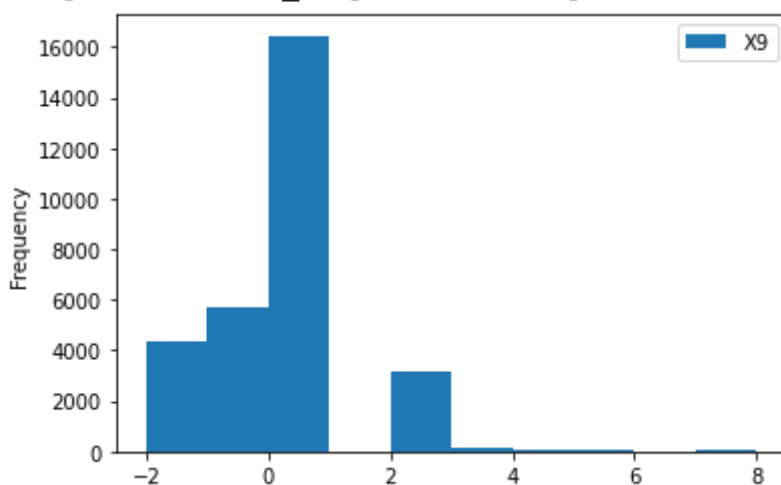
La mayoría de los datos están concentrados en categorías fuera de la descripción de la variable

```
df[['X9']].groupby('X9').size().sort_values(ascending=False) ### observar valores más
```

```
X9
0.0    16450
-1.0     5685
-2.0     4348
2.0      3157
3.0       180
4.0        69
7.0         58
5.0         35
6.0          5
1.0          2
8.0          2
dtype: int64
```

```
df[['X9']].plot.hist()
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa96787b810>



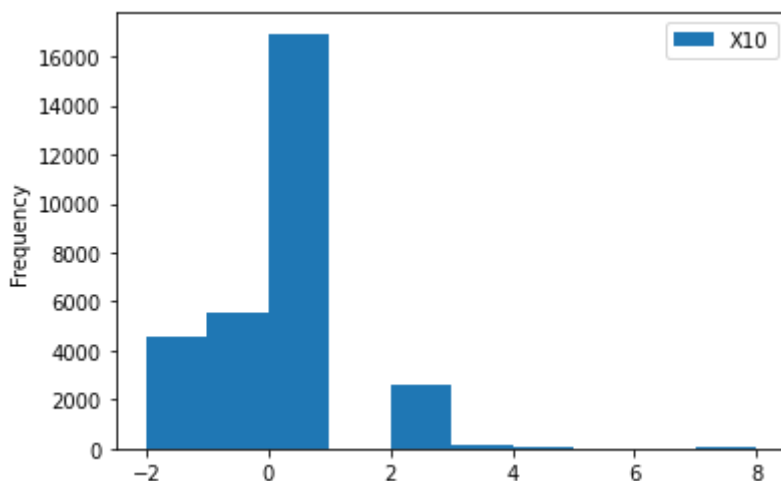
La mayoría de los datos están concentrados en categorías fuera de la descripción de la variable

```
df[['X10']].groupby('X10').size().sort_values(ascending=False) ### observar valores má
```

```
X10
0.0    16937
-1.0    5535
-2.0    4546
2.0     2624
3.0      178
4.0       84
7.0       58
5.0       17
6.0        4
8.0         1
dtype: int64
```

```
df[['X10']].plot.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa9678893d0>
```



La mayoría de los datos están concentrados en categorías fuera de la descripción de la variable

```
df[['X11']].groupby('X11').size().sort_values(ascending=False) ### observar valores má
```

```
X11
0.0    16278
-1.0    5735
-2.0    4895
2.0     2765
3.0      184
4.0       49
7.0       46
6.0       19
5.0       13
```

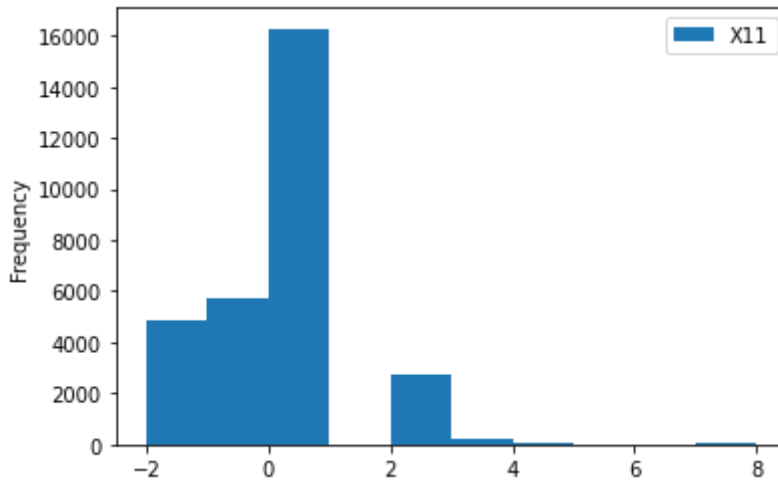
```

8.0      2
dtype: int64

df[['X11']].plot.hist()

```

<matplotlib.axes._subplots.AxesSubplot at 0x7fa96771f350>



La mayoría de los datos están concentrados en categorías fuera de la descripción de la variable

Con lo anterior, observamos que existen categóricas (0,-2) no incluidas en la definición. Por el contexto del problema se interpretan como status corriente en pagos.

Entonces se asigna el valor de -1

```

df['X6'] = np.where((df['X6']==0) | (df['X6']==-2), -1, df['X6'])
df['X7'] = np.where((df['X7']==0) | (df['X7']==-2), -1, df['X7'])
df['X8'] = np.where((df['X8']==0) | (df['X8']==-2), -1, df['X8'])
df['X9'] = np.where((df['X9']==0) | (df['X9']==-2), -1, df['X9'])
df['X10'] = np.where((df['X10']==0) | (df['X10']==-2), -1, df['X10'])
df['X11'] = np.where((df['X11']==0) | (df['X11']==-2), -1, df['X11'])

```

Por las definiciones anteriores, donde la variable te habla sobre el estatus de pago en dicho mes, si existe un missing value, se asigna el valor del mes anterior observado como mejor aproximación del valor faltante

```

df['X10'] = np.where(df['X10'].isna(), df['X11'], df['X10']) ## Si no existe información de
df['X9'] = np.where(df['X9'].isna(), df['X10'], df['X9']) ## Si no existe información de
df['X8'] = np.where(df['X8'].isna(), df['X9'], df['X8']) ## Si no existe información de
df['X7'] = np.where(df['X7'].isna(), df['X8'], df['X7']) ## Si no existe información de
df['X6'] = np.where(df['X6'].isna(), df['X7'], df['X6']) ## Si no existe información de

```

```

### Observar valores nulos por columna de las columnas que estamos analizando
for i in df[['X6', 'X7', 'X8', 'X9', 'X10', 'X11']].columns:
    print('Columna ' + str(i) + ' tiene ' + str(df[i].isna().sum()) + ' valores nulos, c

```



```

Columna X6 tiene 3 valores nulos, que representa 0.01% del total
Columna X7 tiene 5 valores nulos, que representa 0.02% del total
Columna X8 tiene 5 valores nulos, que representa 0.02% del total
Columna X9 tiene 5 valores nulos, que representa 0.02% del total
Columna X10 tiene 14 valores nulos, que representa 0.05% del total
Columna X11 tiene 14 valores nulos, que representa 0.05% del total

```

Aún siguen existiendo missing values en las columnas seleccionadas, por lo que se decide asignar una nueva categoría (0) para aquellos que después del tratamiento sigan presentando missing values.

Se decide una nueva categoría porque a la hora de que nuestro modelo enfrente datos desconocidos, es muy posible que siga afrotando valores faltantes.

```

df['X6'] = np.where(df['X6'].isna(),0,df['X6'])
df['X7'] = np.where(df['X7'].isna(),0,df['X7'])
df['X8'] = np.where(df['X8'].isna(),0,df['X8'])
df['X9'] = np.where(df['X9'].isna(),0,df['X9'])
df['X10'] = np.where(df['X10'].isna(),0,df['X10'])
df['X11'] = np.where(df['X11'].isna(),0,df['X11'])

## Evaluación de valores nulos por columna actual
for i in df[['X6','X7','X8','X9','X10','X11']].columns:
    print('Columna ' + str(i) + ' tiene ' + str(df[i].isna().sum()) + ' valores nulos, c

Columna X6 tiene 0 valores nulos, que representa 0.0% del total
Columna X7 tiene 0 valores nulos, que representa 0.0% del total
Columna X8 tiene 0 valores nulos, que representa 0.0% del total
Columna X9 tiene 0 valores nulos, que representa 0.0% del total
Columna X10 tiene 0 valores nulos, que representa 0.0% del total
Columna X11 tiene 0 valores nulos, que representa 0.0% del total

```

▼ Columnas X12 a X17 -- Cantidad adeudada por mes

Variables sobre la cantidad adeuda por mes.

Donde X12 = Cantidad adeudada a Sep 2005, X13= Cantidad adeudada a Ago 2005, ..., X17= Cantidad adeudada a Abr 2005.

Del descriptivo inicial, observamos que existen valores negativos en el set de variables.

Es por esto que se decide asignar a (-1) con la finalidad de disminuir la variabilidad de la muestra, e identificar todos los registros con dicho problema.

```
## Asignación de valor (-1) a valores negativos.
```

```
df['X12'] = np.where(df['X12']<-1, -1, df['X12'])
df['X13'] = np.where(df['X13']<-1, -1, df['X13'])
df['X14'] = np.where(df['X14']<-1, -1, df['X14'])
df['X15'] = np.where(df['X15']<-1, -1, df['X15'])
df['X16'] = np.where(df['X16']<-1, -1, df['X16'])
df['X17'] = np.where(df['X17']<-1, -1, df['X17'])
```

El tratamiento de missing values será asignar el promedio de saldos (cantidad adeuda por mes) de los meses observables entre Abril 2005 a Septiembre 2005, porque en ese periodo el cliente pudo presentar variabilidad del monto de deuda, si contemplamos solo el máximo, mínimo o anterior, podríamos sesgar el comportamiento de la persona a uno no observable.

Finalmente, el promedio propuesto es por cliente y no por toda la población.

```
## Asignación de valor para tratamiento de missing values por el promedio observado de
df['X12'] = np.where(df['X12'].isna(), df.loc[:, ['X12','X13','X14','X15','X16','X17']
df['X13'] = np.where(df['X13'].isna(), df.loc[:, ['X12','X13','X14','X15','X16','X17']
df['X14'] = np.where(df['X14'].isna(), df.loc[:, ['X12','X13','X14','X15','X16','X17']
df['X15'] = np.where(df['X15'].isna(), df.loc[:, ['X12','X13','X14','X15','X16','X17']
df['X16'] = np.where(df['X16'].isna(), df.loc[:, ['X12','X13','X14','X15','X16','X17']
df['X17'] = np.where(df['X17'].isna(), df.loc[:, ['X12','X13','X14','X15','X16','X17']
```

```
## Valores nulos por columna de las variables tratadas en esta sección
for i in df[['X12','X13','X14','X15','X16','X17']].columns:
    print('Columna ' + str(i) + ' tiene ' + str(df[i].isna().sum()) + ' valores nulos, c
```

```
Columna X12 tiene 8 valores nulos, que representa 0.03% del total
Columna X13 tiene 8 valores nulos, que representa 0.03% del total
Columna X14 tiene 8 valores nulos, que representa 0.03% del total
Columna X15 tiene 8 valores nulos, que representa 0.03% del total
Columna X16 tiene 8 valores nulos, que representa 0.03% del total
Columna X17 tiene 8 valores nulos, que representa 0.03% del total
```

Observamos que siguen existiendo valores nulos, y esto puede seguir sucediendo cuando el modelo se enfrente a data no conocida con missing values.

Se decide asignar el valor de la mediana a los datos faltantes de cada periodo. Elegí la mediana en lugar de la media porque al ser saldos puede haber un mal reporte u outliers que suban el promedio fuera de una respuesta real.

```
## Asignación de valores para tratamiento de missing values
valor12 = df['X12'].median()
valor13 = df['X13'].median()
valor14 = df['X14'].median()
valor15 = df['X15'].median()
valor16 = df['X16'].median()
valor17 = df['X17'].median()
```

```
valor11 = df['X11'].median()
```

```
## Asignación de valores
```

```
df['X12'] = np.where(df['X12'].isna(), valor12, df['X12'])
df['X13'] = np.where(df['X13'].isna(), valor13, df['X13'])
df['X14'] = np.where(df['X14'].isna(), valor14, df['X14'])
df['X15'] = np.where(df['X15'].isna(), valor15, df['X15'])
df['X16'] = np.where(df['X16'].isna(), valor16, df['X16'])
df['X17'] = np.where(df['X17'].isna(), valor17, df['X17'])
```

▼ Columnas X18 a X23 -- Cantidad pagada por mes

Variables sobre la cantidad pagada por mes.

Donde X12 = Cantidad pagada a Sep 2005, X13= Cantidad pagada a Ago 2005, ..., X17= Cantidad pagada a Abr 2005.

Con el descriptivo inicial, notamos que no existen valores negativos por lo que no es necesario realizar alguna transformación.

El tratamiento de missing values será asignar el promedio de pagos (cantidad pagada por mes) de los meses observables entre Abril 2005 a Septiembre 2005 de la persona, porque en ese periodo el cliente pudo presentar variabilidad del monto de pago, si contemplamos solo el máximo, mínimo o anterior, podríamos sesgar el comportamiento de la persona a uno no observable.

Finalmente, el promedio propuesto es por cliente y no por toda la población.

```
## Asignación de valor para tratamiento de missing values por el promedio observado de
df['X18'] = np.where(df['X18'].isna(), df.loc[:, ['X18', 'X19', 'X20', 'X21', 'X22', 'X23']
df['X19'] = np.where(df['X19'].isna(), df.loc[:, ['X18', 'X19', 'X20', 'X21', 'X22', 'X23']
df['X20'] = np.where(df['X20'].isna(), df.loc[:, ['X18', 'X19', 'X20', 'X21', 'X22', 'X23']
df['X21'] = np.where(df['X21'].isna(), df.loc[:, ['X18', 'X19', 'X20', 'X21', 'X22', 'X23']
df['X22'] = np.where(df['X22'].isna(), df.loc[:, ['X18', 'X19', 'X20', 'X21', 'X22', 'X23']
df['X23'] = np.where(df['X23'].isna(), df.loc[:, ['X18', 'X19', 'X20', 'X21', 'X22', 'X23']
```

```
## Valores nulos por columna
```

```
for i in df[['X18', 'X19', 'X20', 'X21', 'X22', 'X23']].columns:
    print('Columna ' + str(i) + ' tiene ' + str(df[i].isna().sum()) + ' valores nulos, c
```

```
Columna X18 tiene 3 valores nulos, que representa 0.01% del total
Columna X19 tiene 3 valores nulos, que representa 0.01% del total
Columna X20 tiene 3 valores nulos, que representa 0.01% del total
Columna X21 tiene 3 valores nulos, que representa 0.01% del total
Columna X22 tiene 3 valores nulos, que representa 0.01% del total
Columna X23 tiene 3 valores nulos, que representa 0.01% del total
```

Observamos que siguen existiendo valores nulos, y esto puede seguir sucediendo cuando el modelo se enfrente a data no conocida con missing values.

Dado que solo se observan 3 datos nulos, se decide asignar el valor de la media a los datos faltantes de cada periodo como mejor aproximación dada su simplicidad.

```
## Valor para ser asignado a missing values
valor18 = df['X18'].mean()
valor19 = df['X19'].mean()
valor20 = df['X20'].mean()
valor21 = df['X21'].mean()
valor22 = df['X22'].mean()
valor23 = df['X23'].mean()

## Asignación de valor a missing values
df['X18'] = np.where(df['X18'].isna(), valor18, df['X18'])
df['X19'] = np.where(df['X19'].isna(), valor19, df['X19'])
df['X20'] = np.where(df['X20'].isna(), valor20, df['X20'])
df['X21'] = np.where(df['X21'].isna(), valor21, df['X21'])
df['X22'] = np.where(df['X22'].isna(), valor22, df['X22'])
df['X23'] = np.where(df['X23'].isna(), valor23, df['X23'])
```

▼ Columna Y -- Flag de comportamiento

Se entiende que la variable Y al sólo tomar los valores 0 y 1 se toma como la clasificación de comportamiento de default para entrenar el modelo.

Sólo se tienen 3 valores nulos. Se decide quitar por completo dichos registros, ya que no sería pertinente asignar un comportamiento sin conocer el resultado, y sesgar la relación de las variables que si observamos.

```
## Filtrar por sólo registros no nulos en variables "Y"
df = df[df['Y'].notna()]
```

Adicionalmente se transforma el tipo de dato de float a int para su uso en la solución de un problema de clasificación.

```
## cambiar el tipo de dato
df['Y'] = df['Y'].astype(int)
```

Resultado de datos faltantes en la muestra

```
## Valores nulos por columna
for i in df.columns:
    print('Columna ' + str(i) + ' tiene ' + str(df[i].isna().sum()) + ' valores nulos, c
```

Columna ID tiene 0 valores nulos, que representa 0.0% del total
Columna X1 tiene 0 valores nulos, que representa 0.0% del total
Columna X2 tiene 0 valores nulos, que representa 0.0% del total
Columna X3 tiene 0 valores nulos, que representa 0.0% del total
Columna X4 tiene 0 valores nulos, que representa 0.0% del total
Columna X5 tiene 0 valores nulos, que representa 0.0% del total
Columna X6 tiene 0 valores nulos, que representa 0.0% del total
Columna X7 tiene 0 valores nulos, que representa 0.0% del total
Columna X8 tiene 0 valores nulos, que representa 0.0% del total
Columna X9 tiene 0 valores nulos, que representa 0.0% del total
Columna X10 tiene 0 valores nulos, que representa 0.0% del total
Columna X11 tiene 0 valores nulos, que representa 0.0% del total
Columna X12 tiene 0 valores nulos, que representa 0.0% del total
Columna X13 tiene 0 valores nulos, que representa 0.0% del total
Columna X14 tiene 0 valores nulos, que representa 0.0% del total
Columna X15 tiene 0 valores nulos, que representa 0.0% del total
Columna X16 tiene 0 valores nulos, que representa 0.0% del total
Columna X17 tiene 0 valores nulos, que representa 0.0% del total
Columna X18 tiene 0 valores nulos, que representa 0.0% del total
Columna X19 tiene 0 valores nulos, que representa 0.0% del total
Columna X20 tiene 0 valores nulos, que representa 0.0% del total
Columna X21 tiene 0 valores nulos, que representa 0.0% del total
Columna X22 tiene 0 valores nulos, que representa 0.0% del total
Columna X23 tiene 0 valores nulos, que representa 0.0% del total
Columna Y tiene 0 valores nulos, que representa 0.0% del total

Nuevo tamaño del dataframe

```
df.shape # Tamaño del dataframe
```

```
(29997, 25)
```

```
## Se nota la falta de los últimos 3 registros eliminados por no tener bandera de comp
```

Sugerencia: Dependiendo del algoritmo a usar se decidirá si es pertinente hacer transformaciones de las variables, o realizar tratamientos extras para outliers. Esto depende del algoritmo a usar debido a que aquellos como Regresión Logística, si se ven afectados por la distribución o por outliers. Sin embargo, algoritmos como ensambles de árboles no se verían afectados. No se realiza mayor investigación al momento porque sale del propósito de dicha práctica.

▼ Asignación de nombre a variables

```
df.columns = ['ID','CreditAmount','Gender','Education','MaritalStatus','Age','Status_Sep',
              'Status_Ago',
              'Status_Jul',
              'Status_Jun',
              'Status_May',
              'Status_Abr',
              'Saldo_Sep',
              'Saldo_Ago',
              'Saldo_Jul',
              'Saldo_Jun',
              'Saldo_May',
              'Saldo_Abr',
              'Pagos_Sep',
              'Pagos_Ago',
              'Pagos_Jul',
              'Pagos_Jun',
              'Pagos_May',
              'Pagos_Abr',
              'Y']
```

▼ Parte 3: Preparación de los datos

Respuesta a las preguntas planteadas

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

Considero que los datos más importantes son aquellos que te hablan sobre las características del crédito, y su comportamiento en el tiempo. Es decir, las variables sobre monto de crédito, saldo en últimos meses, estatus y monto de pago en los periodos anteriores. Las considero más importantes sobre las otras variables sociodemográficas ya que están relacionadas directamente con el propósito del modelo, que es medir la probabilidad de default de una persona en el ambiente crediticio. Por otro lado, dependiendo de cómo se quiera utilizar el modelo, las variables sociodemográficas pueden ayudar a segmentar a la población, y crear modelos más precisos por segmento. Dentro de esto, es importante resaltar la edad de la persona, ya que de esta depende la etapa de vida del cliente, el ingreso, y con ello adquirir mayor responsabilidad crediticia.

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

Se reemplazaron los valores nulos dependiendo del tipo de datos y la información que quiere transmitir. A modo de resumen expongo que, para las variables categóricas como género o nivel de educación se asignó la moda como criterio de limpieza. Por otro lado, la variable histórica del estatus de pago, se decidió asignar el último estatus observado, por ejemplo si no se tiene el dato de Septiembre se asigna el observado de Agosto, y así sucesivamente. De continuar con valores nulos en dicha variable, se asignó el valor "cero" con el fin de identificar los datos sin registro en dichas variables, y que funcione como tratamiento cuando el modelo se encuentre con información no conocida (test). Para los datos que tenían valores con etiquetas distintas a las categorías por definición de la variable se hicieron ajustes por lo que se sobre escribió en categoría "otros" aquellos valores que se desconocía su tipo y pudo deberse a un error de la obtención de la información. Por otro lado, las variables de histórico de Saldos y de Pago, se utilizó el promedio de la persona en los periodos observados, a forma de que el valor asignado esté dentro del rango de comportamiento de la persona. De seguir presentando valores nulos, se asignó la mediana de la población para dicha variable. Finalmente, se decidieron eliminar los registros con valores nulos en "Y" flag de comportamiento, ya que no se tenía una forma de asignar sin sesgar la relación de las variables.

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

No, pienso que lo necesario es conocer la variable en cuanto a la información que almacena y su distribución para entender qué tratamiento se le debe realizar. Más allá de ordenar la información, se trata de ir a fondo al entendimiento de la descripción de la variable en conjunto con lo observado, y de esta manera encontrar la mejor forma de utilizar los datos para que el modelo tenga el mejor desempeño. Sin embargo, lo más importante si es la limpieza de la información para tratar de detectar anomalías y garantizar la consistencia de los resultados obtenidos.

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

Sí existen problemas que debemos solucionar antes de proseguir con el modelado, tales como verificar que el monto de crédito otorgado contenga valores positivos, que los datos capturados para las variables de género, educación y estado civil concuerden con la definición original, la edad corresponda a mayores de 18 años y que el monto de la deuda, pagos anteriores e historial de pago no posean valores negativos porque se saldrían del contexto sobre el que son aplicados. Debía realizarse porque si los valores se salían de los estándares que en la misma descripción de las variables contemplaba simplemente eran datos inexistentes o inválidos en cuanto a su clasificación, lo cual, en el conjunto de datos sí sucedía y debía corregirse, además se debe incluir la excepción de manejo de datos que pudieran no tener el valor de la definición.

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

Adicional a la limpieza de missing values expuesta, se encontró que en algunas variables categóricas como nivel de educación o estatus de pago en meses anteriores, se encontraban clasificaciones no descritas en las definiciones, donde se entiende había un error de llenado, por lo que a dichas clasificaciones se les asignó un único nivel de "otros" para el caso de nivel de educación, y al corriente (-1) para el estatus de pago. Por otro lado, la imputación media para preservar la distribución de los datos y no sacrificar su demás información al eliminarlos. Asimismo, se encontró que existían valores negativos en monto de deuda por lo que se les asignó el valor de -1 para disminuir su efecto y lograr identificarlos. Para la variable target "y" es muy importante no contar con datos faltantes ya que al ser sobre la que se entrenaría es muy

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

✓ 0 s se ejecutó 19:07

