Semana 3-Actividad 1

Ciencia y Analitica de datos

Alumno: Valeria Alejandra Ramirez Herrera

Matricula: A01240716

Profesor: Jobish Vallikavungal

Fecha: 04/Octubre/2022

Parte 1 Fundamentos de bases de datos

1. Fundamentos de bases de datos y para ciencia de datos

   Una base de datos es una coleccion de datos/informacion relacionados a un tema que son almacenados electronicamente. Los datos recolectados son de suma importancia para la ciencia de datos ya quelas personas que se dedican a esto pasan la mayo parte de su recolectando datos y preparandolos, para despues comenza en el desarrollo de su modelo y desplegarlo, y al final revisar constantemente su performance. Asi que teniendo ya una base de datos basados en su problema espefico es de mucha ayuda. Basado en la presentacion del profesor Jobish, algunas caracteristicas de las bases de datos pueden ser: que son un repositorio de datos compartidos, los datos son gestionados por un agente de control y son almacenados en una forma estandarizada.
2. Fundamentos de almacenes de datos (Data Warehouse) para ciencia de datos. Un almacen de datos es una base datos muy grande que contiene informacion de muchas fientas y las pone en un lugar para su uso. Para la ciencia de datos, los almacenes de datos son igual de utiles ya que estan disenados dar soporte con las aplicaciones de Inteligencia de Negocios, en especial el analisis interno de los datos. Este siendo uno de los problemas para los cientificos de datos, ya que con tanta informacion siendo importada, la tranformacion de los datos se puede volver complicada por si hay datos corruptos o con outliers.

Parte 2: Selección y limpieza de los Datos en Python

Revisa detenidamente la página Ejercicio guiado para: Selección y limpieza de los Datos en PythonEnlaces 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 lineas del codigo en celdas del Text) como funciona tu

```
import pandas as pd
import numpy as np


input = 'https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/c


df=pd.read_csv(input) #Aquí se encuentran los datos originales
df
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	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	326
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	1554
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	2954
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	1913
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	1598
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0	
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0	1935
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0	4894
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	1531

30000 rows × 25 columns

```
 df.isnull().values.any() #Se hace verificación si falta algún dato
```

```
True
```

```
df.isnull().any() #En cuales columnas falta al menos 1 dato
```

```
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
```

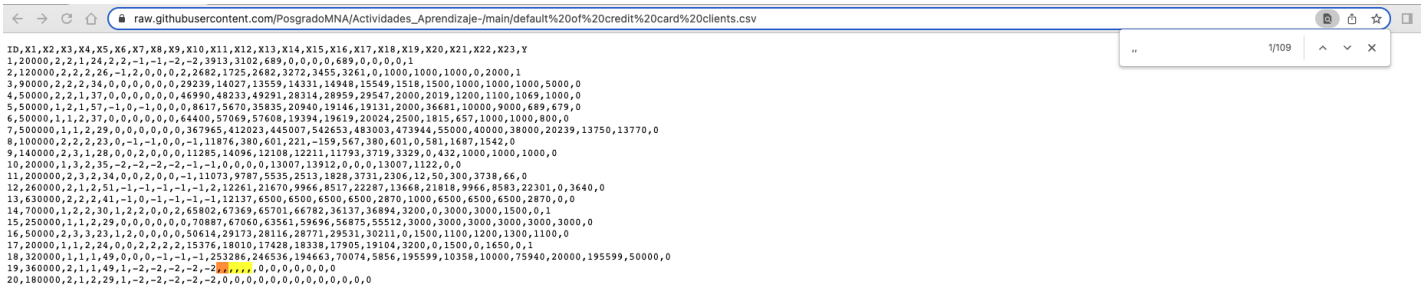
```
ndf=df.copy() #Datos generales del dataset
ndf.describe()
```

	ID	X1	X2	X3	X4	X
count	30000.000000	30000.000000	29999.000000	29998.000000	29998.000000	29995.00000
mean	15000.500000	167484.322667	1.603753	1.853057	1.551903	35.4842
std	8660.398374	129747.661567	0.489125	0.790320	0.521968	9.2180
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.00000
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.00000
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.00000
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.00000
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.00000

8 rows × 25 columns

Solucion 1 - se eliminan todos los renglones si tienen al menos un dato perdido

En este universo de datos de 30,000 renglones x 23 columnas, se tiene un total de 690,00 datos donde solo 109 de ellos son datos perdidos, lo cual representa solo el 0.016% del total de datos.




```
ndf=df #nueva variable
ndf.dropna(inplace=True) #Quita los renglores que tienen al menos 1 dato perdido, en e
#un total de 42 renglones fueron removidos (0.14%)

ndf
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	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	326
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	1554
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	2954
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	1913
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0	1598
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0	
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0	1935
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0	4894
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0	1531

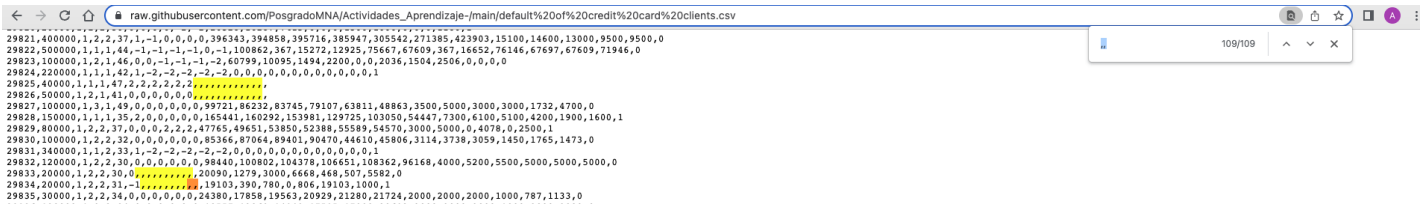
29958 rows × 25 columns

```
ndf.isnull().values.any() #Se comprueba que se hizo la limpieza y no falta ningún dato
```

```
False
```

Solución 2 - Se eliminan renglones si al menos 1 dato en columna está perdido

Este método no funcionaría de manera eficiente ya que, como se muestra en la siguiente imagen, la mayoría de las columnas serían eliminadas y se perderían muchos datos importantes para los renglones que sí contienen datos.



Solución 3 - Llenar espacio vaciós en las columnas, dependiendo del tipo de dato (categórico o continuo)

- Columnas con datos categóricos: X2, X3, X4, X6, X7, X8, X9, X10, X11
- Columnas con datos continuos: X1, X5, X12, X13, X14, X15, X16, X17, X18, X19, X20, X21, X22, X23

A la vez que se hace ajuste para los datos categóricos y entren en el rango establecido.

```
ndf = df.copy()
ndf['X2'].fillna(value= ndf['X2'].mode(), inplace = True)
ndf['X3'].fillna(value= ndf['X3'].mode(), inplace = True)
ndf['X4'].fillna(value= ndf['X4'].mode(), inplace = True)
ndf['X6'].fillna(value= ndf['X6'].mode(), inplace = True)
ndf['X7'].fillna(value= ndf['X7'].mode(), inplace = True)
ndf['X8'].fillna(value= ndf['X8'].mode(), inplace = True)
ndf['X9'].fillna(value= ndf['X9'].mode(), inplace = True)
ndf['X10'].fillna(value= ndf['X10'].mode(), inplace = True)
ndf['X11'].fillna(value= ndf['X11'].mode(), inplace = True)


ndf['X1'].fillna(value= ndf['X1'].mean(), inplace = True)
ndf['X5'].fillna(value= ndf['X5'].mean(), inplace = True)
ndf['X12'].fillna(value= ndf['X12'].mean(), inplace = True)
ndf['X13'].fillna(value= ndf['X13'].mean(), inplace = True)
ndf['X14'].fillna(value= ndf['X14'].mean(), inplace = True)
ndf['X15'].fillna(value= ndf['X15'].mean(), inplace = True)
ndf['X16'].fillna(value= ndf['X16'].mean(), inplace = True)
ndf['X17'].fillna(value= ndf['X17'].mean(), inplace = True)
ndf['X18'].fillna(value= ndf['X18'].mean(), inplace = True)
ndf['X19'].fillna(value= ndf['X19'].mean(), inplace = True)
ndf['X20'].fillna(value= ndf['X20'].mean(), inplace = True)
ndf['X21'].fillna(value= ndf['X21'].mean(), inplace = True)
ndf['X22'].fillna(value= ndf['X22'].mean(), inplace = True)
ndf['X23'].fillna(value= ndf['X23'].mean(), inplace = True)

#Se hace el reemplazo/limpieza de valores que no entran en el rango de los datos categ
#Columnas con datos categóricos: X2, X3, X4, X6, X7, X8, X9, X10, X11

ndf=ndf.replace({'X3':0},4) #Todo lo que tenga un 0 para la columna 3, se cambia a 4=d
ndf=ndf.replace({'X3':5},4) #Todo lo que tenga un 5 para la columna 3, se cambia a 4=d
```

```
ndf=ndf.replace({'X3':6},4) #Todo lo que tenga un 6 para la columna 3, se cambia a 4=c

ndf=ndf.replace({'X4':0},3) #Todo lo que tenga un 0 para la columna 4, se cambia a 3=c

ndf=ndf.replace({'X6':-2},-1) #Todo lo que tenga un -2 para la columna 6, se cambia a
ndf=ndf.replace({'X6':0},-1) #Todo lo que tenga un 0 para la columna 6, se cambia a -1
ndf=ndf.replace({'X7':-2},-1) #Todo lo que tenga un -2 para la columna 7, se cambia a
ndf=ndf.replace({'X7':0},-1) #Todo lo que tenga un 0 para la columna 7, se cambia a -1
ndf=ndf.replace({'X8':-2},-1) #Todo lo que tenga un -2 para la columna 8, se cambia a
ndf=ndf.replace({'X8':0},-1) #Todo lo que tenga un 0 para la columna 8, se cambia a -1
ndf=ndf.replace({'X9':-2},-1) #Todo lo que tenga un -2 para la columna 9, se cambia a
ndf=ndf.replace({'X9':0},-1) #Todo lo que tenga un 0 para la columna 9, se cambia a -1
ndf=ndf.replace({'X10':-2},-1) #Todo lo que tenga un -2 para la columna 10, se cambia
ndf=ndf.replace({'X10':0},-1) #Todo lo que tenga un 0 para la columna 10, se cambia a
ndf=ndf.replace({'X10':-2},-1) #Todo lo que tenga un -2 para la columna 11, se cambia
ndf=ndf.replace({'X10':0},-1) #Todo lo que tenga un 0 para la columna 11, se cambia a


#Se cambian los nombres de las columnas
ndf.rename(columns = {'X1' : 'Credit Amount (NT dollar)', 'X2': 'Gender', 'X3': 'Educa
ndf
```

	ID	Credit Amount (NT dollar)	Gender	Education	Marital Status	Age	Past payment Sep 2005	Past payment Aug 2005	Past payment Jul 2005
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0
1	2	120000	2.0	2.0	2.0	26.0	-1.0	2.0	-1.0
2	3	90000	2.0	2.0	2.0	34.0	-1.0	-1.0	-1.0
3	4	50000	2.0	2.0	1.0	37.0	-1.0	-1.0	-1.0
4	5	50000	1.0	2.0	1.0	57.0	-1.0	-1.0	-1.0
...	...	...	...	...	...	...	...	...	...
29995	29996	220000	1.0	3.0	1.0	39.0	-1.0	-1.0	-1.0
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	-1.0
29999	30000	50000	1.0	2.0	1.0	46.0	-1.0	-1.0	-1.0

29958 rows × 25 columns

```
ndf.isnull().values.any() #Se comprueba que se hizo la limpieza y no falta ningún dato
```

```
False
```

Parte 3: Preparación de los datos

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

- **¿Qué datos considero mas importantes? ¿Por qué?** Considero que los datos más importantes corresponden a la escolaridad (X3), estado marital (X4) y el historial de pagos (X6-X11). Porque para otorgar un crédito influye si la persona ya tiene un grado escolar Universitario/Posgrado, si la persona tiene o no familia, no porque el estar soltero sea problema sino porque el mantener a personas adicionales implica responsabilidad al igual que el historial de pagos, ya que mientras los pagos estén a tiempo significa que el grado de responsabilidad de la persona es alto.
- **¿Se eliminaron o reemplazaron datos nulos? ¿Qué se hizo y por qué?** Se hicieron dos posibles soluciones ya que ambas podían aplica Solución 1: Se removieron los renglones que no contenian al menos un dato, ya que en este universo de datos de 30,000 renglones x 23 columnas, se tiene un total de 690,00 datos donde solo 109 de ellos son datos perdidos, lo cual representa solo el 0.016% del total y no se ve muy significativo. Por otro lado se tuvo otra solución que fue ajustar datos faltantes dependiendo del tipo de datos, a los categóricos faltantes se les aplicó la función de Moda y a los datos continuos les apliqué la media para evitar datos faltantes y como se menciona en la solución anterior es mínimo pero se manteniene dentro
- **¿Es necesario ordenar los datos para el análisis? Sí / No / ¿Por qué?** Sí es necesario e importante el ordenar los datos ya que muchas veces al no hacerlo pudieramos tener inconsistencias en los datos del dataset original como números faltantes, números negativos, datos duplicados, etc. En nuestro caso podrían no corresponder las variables categóricas al rango de valor dado, etc, y podríamos llegar a tener errores o que el sistema no funcione como queremos.
- **¿Existen problemas de formato que deban solucionar antes del proceso de modelado? Sí / No / Por qué.** Sí, si hay errores de formato, como se mencionó en la pregunta anterior, se presentaron problemas de valores nulos que podrían ser eliminados ya que no representan mucha porción del data set, esto en cuestión de renglones ya que si hablamos de columnas cambia totalmente porque se eliminarían demasiadas columnas dejando el data set tan incompleto que no podríamos trabajar con él. Se tienen también datos que están fuera del rango.

- **¿Qué ajustes se realizaron en el proceso de limpieza de datos (agregar, integrar, eliminar, modificar registros (filas), cambiar atributos (columnas)?** Como se mencionó anterioremente, se dividió las columnas del dataset en datos categóricos y datos continuos. Los datos categóricos se les puso la función de sacar la moda para aquellos datos que no tuvieran valor, mientras que a los datos continuos nulos se sacó la media para evitar la eliminación de los renglones/columnas. Y para aquellos valores de los datos categóricos que se salian del rango establecido se aplicó un reemplazo de datos, la mayoría de ellos si no estaban dentro del rango podían aplicar con el valor para Others. Y al final se hizo el cambio de los atributos de las columnas respecto al nombre de las mismas.

Semana 3 - Actividad 1

- Nombre: Evelyn Aylin Rendon Medina
- Matricula: A01748750

- Materia: Ciencia y analítica de datos (Gpo 10)
- Profesor: Jobish Vallikavungal Devassia
- Fecha de entrega: 4 octubre 2022

Parte 1

Fundamentos de bases de datos y para ciencia de datos. Una base de datos se puede entender como una recopilación de información, la cual posee determinado orden o estructura (datos estructurados) y su contenido tiene relación o está asociado a una actividad determinada. Dicha base puede contener tablas, la cual está formada por filas (observaciones) y columnas (categorías), donde la intersección entre tales elementos se llama 'celda'. De igual forma, es importante mencionar que cuando hablamos de datos, se hace referencia a los registros contenidos, es decir, la observación de una categoría es un dato o registro y el conjunto de ellos crea nuestra base de datos. El personal especializado para trabajar con bases de datos recibe el nombre de 'científico de datos' y es aquel individuo que requiere comprender los datos de los cuales dispone, con la finalidad de analizarlos y fundamentar la toma de decisiones. Para ello requiere un amplio conocimiento teórico y habilidad al utilizar las tecnologías disponibles que le permiten manipular los datos para obtener información, como encontrar relaciones o predecir comportamientos. Cabe destacar que los datos son una parte fundamental del modelo y determinarán en gran medida su rendimiento, por lo que tener un conjunto de datos apropiado es también un aspecto vital, de forma que la persona especializada que lleve a cabo dicha tarea pasará alrededor del 80% del tiempo recopilándolos y preparándolos. Existe un enfoque de la base de datos, donde se tiene un repositorio de dichos registros con cierto estándar en su almacén para lo cual es necesario un sistema de administración de datos (DBMS) y algunos lenguajes para manipular tales datos son el DML (query language) y SQL (Structured Query Language).

Fundamentos de almacenes de datos (Data Warehouse) para ciencia de datos. Un almacén de datos (DW) conceptualmente es un depósito central cuyo análisis (de los datos) nos ayuda a tomar mejores decisiones, pues tienen conocimiento que las respalda. Se podría decir que es una base de datos de gran tamaño y su contenido se genera a través de la unificación de datos provenientes de distintas fuentes. Dicho en otras palabras, este tipo de almacenamiento es capaz

de contener en sí varias bases de datos y su diseño se orienta en hacer posible su análisis, por lo que deberá contener datos estructurados. Es preciso destacar que una base de datos está relacionada con una única actividad o un particular propósito, que es lo que la diferencia de un almacén de datos (data warehouse). De igual forma, un lago de datos es un repositorio capaz de contener datos de diversos tipos (estructurados, no estructurados y semiestructurados) y dicha característica lo diferencia del almacén (DW), pues el último requiere que sean estructurados para que sea posible consultar sus datos. Otro concepto relacionado que es importante destacar es la llave primaria (primary key), que es un identificador único que se genera para cada observación, tal como es nuestro CURP. Cuando en una tabla se hace referencia a una llave proveniente de otra tabla, se le llama llave foránea (foreign key)

Parte 2

Carga del archivo

Se obtiene la base de datos y se guarda como un dataframe a través de la librería de pandas

```
baseDatos='https://raw.githubusercontent.com/PosgradoMNA/Actividades_Aprendizaje-/main/defaul
import pandas as pd
import numpy as np
from statistics import mode
df=pd.read_csv(baseDatos)
df.head()
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X1
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	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.(
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0	15549.(
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0	29547.(
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0	19131.(

5 rows × 25 columns

Se crea una copia del dataframe para hacer modificaciones

```
cdf=df.copy()
cdf
```

	ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16
0	1	20000	2.0	2.0	1.0	24.0	2.0	2.0	-1.0	-1.0	...	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
2	3	90000	2.0	2.0	2.0	34.0	0.0	0.0	0.0	0.0	...	14331.0	14948.0
3	4	50000	2.0	2.0	1.0	37.0	0.0	0.0	0.0	0.0	...	28314.0	28959.0
4	5	50000	1.0	2.0	1.0	57.0	-1.0	0.0	-1.0	0.0	...	20940.0	19146.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
29995	29996	220000	1.0	3.0	1.0	39.0	0.0	0.0	0.0	0.0	...	88004.0	31237.0
29996	29997	150000	1.0	3.0	2.0	43.0	-1.0	-1.0	-1.0	-1.0	...	8979.0	5190.0
29997	29998	30000	1.0	2.0	2.0	37.0	4.0	3.0	2.0	-1.0	...	20878.0	20582.0
29998	29999	80000	1.0	3.0	1.0	41.0	1.0	-1.0	0.0	0.0	...	52774.0	11855.0
29999	30000	50000	1.0	2.0	1.0	46.0	0.0	0.0	0.0	0.0	...	36535.0	32428.0

30000 rows × 25 columns

Estandarización general

Se verifica el tipo de datos en cada columna del dataframe

```
cdf.dtypes
```

```
ID       int64
X1       int64
X2     float64
X3     float64
X4     float64
X5     float64
X6     float64
X7     float64
X8     float64
X9     float64
X10    float64
X11    float64
X12    float64
X13    float64
X14    float64
X15    float64
X16    float64
X17    float64
X18    float64
X19    float64
```

```
X20    float64
X21    float64
X22    float64
X23    float64
Y      float64
dtype: object
```

Las comprobaciones que se realizarán en estos datos serán únicamente para conocer mejor este conjunto específico, pero se contemplan correciones bajo la consideración que no sabemos cómo serían los datos futuros

Se comprueba que X1 tenga valores numéricos <=0 debido a que es el monto de crédito otorgado y no podría ser negativo. En caso que existieran esos valores, se considerarán iguales a cero.

```
x1=cdf[cdf['X1']<=0]
print(x1)
cdf[cdf['X1']<=0]=0 #Cambio de valores conforme el estándar comentado
```

```
           ID  X1   X2   X3   X4   X5   X6   X7   X8   X9  ...  X15  X16  X17  \
0          -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
9          -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
18         -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
19         -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
23         -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
...        ..  ..  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
29984      -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
29985      -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
29986      -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
29989      -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0
29992      -1  -1 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0  ... -1.0 -1.0 -1.0

           X18  X19  X20  X21  X22  X23    Y
0         -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
9         -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
18        -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
19        -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
23        -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
...        ...  ...  ...  ...  ...  ...  ...
29984     -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
29985     -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
29986     -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
29989     -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0
29992     -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0

[6451 rows x 25 columns]
```

Se comprueba si hay elementos en X2 cuyos valores no correspondan a las categorías de género disponibles. En caso que existieran esos valores, se sustituirán por uno de los dos géneros al azar,

puesto que el modelo no está particularmente enfocado en el género y no se opta por alguno de los dos arbitrariamenete para evitar algún sesgo.

```
x2=cdf[(cdf['X2']!=1) & (cdf['X2']!=2)]
print(x2)
cdf[(cdf['X2']!=1) & (cdf['X2']!=2)]=np.random.randint(1,3) #Cambio de valores conforme el es
```

```
           ID  X1   X2   X3   X4   X5   X6   X7   X8   X9  ...  X15  X16  X17  \
0           0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
9           0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
18          0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
19          0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
23          0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
...        ..  ..  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
29984       0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
29985       0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
29986       0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
29989       0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0
29992       0   0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0

       X18  X19  X20  X21  X22  X23    Y
0      0.0  0.0  0.0  0.0  0.0  0.0  0.0
9      0.0  0.0  0.0  0.0  0.0  0.0  0.0
18     0.0  0.0  0.0  0.0  0.0  0.0  0.0
19     0.0  0.0  0.0  0.0  0.0  0.0  0.0
23     0.0  0.0  0.0  0.0  0.0  0.0  0.0
...    ...  ...  ...  ...  ...  ...  ...
29984  0.0  0.0  0.0  0.0  0.0  0.0  0.0
29985  0.0  0.0  0.0  0.0  0.0  0.0  0.0
29986  0.0  0.0  0.0  0.0  0.0  0.0  0.0
29989  0.0  0.0  0.0  0.0  0.0  0.0  0.0
29992  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[6852 rows x 25 columns]
```

Se comprueba si hay elementos en X3 cuyos valores no correspondan a las categorías de educación disponibles. En caso que existieran esos valores, se les asignará la categoría 'otra' porque es una no específica y al no contar con el dato correcto, sería la opción más lógica.

```
x3=cdf[(cdf['X3']!=1) & (cdf['X3']!=2) & (cdf['X3']!=3) & (cdf['X3']!=4)]
print(x3)
cdf[(cdf['X3']!=1) & (cdf['X3']!=2) & (cdf['X3']!=3) & (cdf['X3']!=4)]=4 #Cambio de valores c
```

```
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
```

Se comprueba si hay elementos en X4 cuyos valores no correspondan a las categorías de estado civil disponibles. En caso que existieran esos valores, se les asignará la categoría 'otra' porque es una no específica y al no contar con el dato correcto, sería la opción más lógica.

```
x4=cdf[(cdf['X4']!=1) & (cdf['X4']!=2) & (cdf['X4']!=3)]
print(x4)
cdf[(cdf['X4']!=1) & (cdf['X4']!=2) & (cdf['X4']!=3)]=3 #Cambio de valores conforme el estánc
```

```
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
```

Se comprueba si hay elementos en X5 cuyos valores no correspondan con una edad lógica (<18 = ilógico). En caso que existieran esos valores, se les asignará edad=18.

```
x5=cdf[cdf['X5']<18]
print(x5)
cdf[cdf['X5']<18]=18 #Cambio de valores conforme el estándar comentado
```

```
          ID  X1   X2   X3   X4   X5   X6   X7   X8   X9  ...  X15  X16  X17  \
0          1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
9          1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
18         1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
19         1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
23         1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
...       ..  ..  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...  ...
29984      1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
29985      1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
29986      1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
29989      1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0
29992      1   1  1.0  1.0  1.0  1.0  1.0  1.0  1.0  1.0  ...  1.0  1.0  1.0

       X18  X19  X20  X21  X22  X23    Y
0      1.0  1.0  1.0  1.0  1.0  1.0  1.0
9      1.0  1.0  1.0  1.0  1.0  1.0  1.0
18     1.0  1.0  1.0  1.0  1.0  1.0  1.0
19     1.0  1.0  1.0  1.0  1.0  1.0  1.0
23     1.0  1.0  1.0  1.0  1.0  1.0  1.0
...    ...  ...  ...  ...  ...  ...  ...
29984  1.0  1.0  1.0  1.0  1.0  1.0  1.0
29985  1.0  1.0  1.0  1.0  1.0  1.0  1.0
29986  1.0  1.0  1.0  1.0  1.0  1.0  1.0
29989  1.0  1.0  1.0  1.0  1.0  1.0  1.0
29992  1.0  1.0  1.0  1.0  1.0  1.0  1.0

[6852 rows x 25 columns]
```

Se comprueba si hay elementos en X6:X11 (sep-abr 2005) cuyos valores no correspondan con un historial de pago lógico (<-1). De momento solo se hace para conocer nuestros datos, en la siguiente sección se tomarán medidas.

```
r6=cdf.columns.get_loc('X6') #Se busca el índice para la columna llamada 'X6'
r11=cdf.columns.get_loc('X11') #Se busca el índice para la columna llamada 'X11'
for i in range(r6,r11+1):
  x6_x11=cdf[cdf.iloc[:,i]<-1]
  print(x6_x11)
```

```
    Empty DataFrame
    Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
    Index: []

    [0 rows x 25 columns]
    Empty DataFrame
    Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
    Index: []

    [0 rows x 25 columns]
    Empty DataFrame
    Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
    Index: []

    [0 rows x 25 columns]
    Empty DataFrame
    Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
    Index: []

    [0 rows x 25 columns]
    Empty DataFrame
    Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
    Index: []

    [0 rows x 25 columns]
    Empty DataFrame
    Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
    Index: []

    [0 rows x 25 columns]
```

En caso que existieran esos valores (historial de pago no lógico), se les asignará la categoría 'paga debidamente' (-1), puesto que si son números negativos más grandes que -1, podría significar un error al capturar los datos.

```
for i in range(r6,r11+1): #Se crea un bucle para modificar los valores del índice de la colum
  cdf[cdf.iloc[:,i]<-1] =-1 #cdf[cdf['X6']<-1] =-1
```

Comprobamos que fueron corregidos

```
x6=cdf[cdf['X6']<-1]
print(x6)
x11=cdf[cdf['X11']<-1]
print(x11)
```

```
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
```

Se comprueba si hay elementos en X11:X17 cuyos valores no correspondan con un monto de deuda lógico (<-1 = ilógico). De momento solo se hace para conocer nuestros datos, en la siguiente sección se tomarán medidas.

```
r12=cdf.columns.get_loc('X12') #Se busca el índice para la columna llamada 'X12'
r17=cdf.columns.get_loc('X17') #Se busca el índice para la columna llamada 'X17'
for·i·in·range(r12,r17+1):
··x12_x17=cdf[cdf.iloc[:,i]<-1]·
··print(x12_x17)
```

```
        X12      X13      X14      X15      X16      X17
276   -10.0  22535.0   2728.0      0.0   2654.0      0.0  22545.0      0.0
564  3346.0   2698.0      1.0   1693.0   6940.0      7.0   1230.0      3.0
977  1056.0    390.0    662.0      0.0  61411.0   1056.0      0.0   1998.0
1039 18495.0  18804.0   8769.0      0.0  18650.0    659.0   1000.0   1000.0
...      ...      ...      ...      ...      ...      ...      ...      ...
28555 2093.0   5773.0  53400.0      0.0  51566.0   2093.0   5773.0   2984.0
28634 6383.0    817.0   4923.0      0.0   6383.0      2.0      0.0   1809.0
28754 11632.0   7864.0   1245.0      0.0  12019.0   1000.0    157.0   7851.0
29602 14766.0  14687.0   3000.0     18.0  15300.0   5000.0  15132.0   6500.0
29888 112738.0 115746.0   4400.0      0.0 112736.0   5000.0   5300.0   5000.0

        Y
72    0.0
276   1.0
564   0.0
977   0.0
1039  0.0
...   ...
28555 0.0
28634 0.0
28754 0.0
```

```
29602  0.0
29888  0.0

[103 rows x 25 columns]
          ID      X1   X2   X3   X4    X5   X6   X7   X8   X9  ...     X15  \
194      195   50000  1.0  2.0  1.0  36.0  0.0  0.0  0.0  0.0  ...   -14.0
757      758  230000  2.0  2.0  1.0  30.0  2.0  0.0 -1.0 -1.0  ...  -828.0
1000    1001  100000  1.0  2.0  1.0  29.0  0.0  0.0  0.0  0.0  ... -2618.0
1337    1338  290000  2.0  1.0  2.0  31.0 -1.0 -1.0 -1.0 -1.0  ...  -150.0
2914    2915  320000  2.0  2.0  2.0  50.0 -1.0  0.0 -1.0 -1.0  ...    -6.0
...      ...     ...  ...  ...  ...   ...  ...  ...  ...  ...  ...     ...
28157  28158  130000  2.0  3.0  1.0  31.0  0.0 -1.0 -1.0 -1.0  ... -1018.0
28236  28237  400000  2.0  3.0  1.0  40.0  0.0 -1.0 -1.0 -1.0  ...  -329.0
28799  28800  200000  2.0  3.0  2.0  42.0  1.0 -1.0 -1.0 -1.0  ... -3903.0
29353  29354  390000  1.0  2.0  2.0  28.0  0.0  0.0  0.0  0.0  ... -1391.0
29606  29607   90000  1.0  2.0  1.0  45.0 -1.0  3.0  2.0  2.0  ... -1170.0

            X16       X17      X18      X19      X20       X21      X22  \
194        72.0     658.0   2000.0   1000.0   2000.0     500.0   1000.0
757      6153.0   12663.0   1222.0   1837.0      0.0    7863.0  12675.0
1000    95748.0  101299.0   3320.0   5000.0      0.0  100000.0   7186.0
1337    66675.0   68071.0      0.0    480.0      4.0   69001.0   2500.0
2914     1252.0       0.0   3000.0    342.0      2.0    1500.0      0.0
...         ...       ...      ...      ...      ...       ...      ...
28157   89445.0   90179.0   4065.0   6704.0  21018.0   92610.0   3272.0
28236   65691.0   23657.0  35729.0  72031.0      0.0   66020.0  23776.0
28799   11989.0   11839.0  10000.0   2509.0      0.0   24000.0      0.0
29353  249742.0  253914.0   9124.0   3181.0    895.0  253009.0  10009.0
29606     390.0     390.0      0.0      0.0      0.0    1560.0    390.0

           X23    Y
194    20011.0  0.0
757     2959.0  1.0
1000       0.0  0.0
1337    3000.0  0.0
2914   10701.0  0.0
...        ...  ...
```

En caso que existieran esos valores (monto de deuda lógico), se transformarán a positivos, considerando que se deban a un error de captura.

```
for i in range(r12,r17+1):
  cdf[cdf.iloc[:,i]<-1]*-1
```

Se comprueba si hay elementos en X18:X23 cuyos valores no correspondan con un monto de pago anterior lógico (<-1 = ilógico). De momento solo se hace para conocer nuestros datos, en la siguiente sección se tomarán medidas.

```
r18=cdf.columns.get_loc('X18') #Se busca el índice para la columna llamada 'X18'
r23=cdf.columns.get_loc('X23') #Se busca el índice para la columna llamada 'X23'
for i in range(r18,r23+1):
```

```
  x18_x23=cdf[cdf.iloc[:,i]<-1]
  print(x18_x23)
```

```
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
Empty DataFrame
Columns: [ID, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X1
Index: []

[0 rows x 25 columns]
```

En caso que existieran esos valores (de pago anterior no lógicos), se transformarán a positivos, considerando que se deban a un error de captura.

```
for i in range(r18,r23+1):
  cdf[cdf.iloc[:,i]<-1]*-1
```

Tratamiento de datos faltantes

Verificamos si hay datos faltantes, dónde y cuántos

```
print(cdf.isnull().values.any())
print("")
print(cdf.isnull().any())
```

```
print("")
print(cdf.isnull().sum())
```

```
True

ID     False
X1     False
X2     False
X3     False
X4     False
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

ID      0
X1      0
X2      0
X3      0
X4      0
X5      2
X6      1
X7      3
X8      5
X9      7
X10    12
X11    10
X12     7
X13     7
X14     9
X15    11
X16    12
X17     6
X18     6
X19     6
X20     6
X21     7
X22     8
X23     5
```

```
Y       3
dtype: int64
```

Eliminamos las filas donde falten todos los elementos

```
cdf.dropna(how='all', inplace = True)
```

Eliminamos las filas donde haya a partir de 10 elementos faltantes

```
for i in range(10,14):
  cdf.dropna(thresh=i, inplace = True)
```

Se imputan los valores faltantes en X5 con la edad más frecuente.

```
#cdf[cdf['X5'].isnull()]
modaX5=mode(cdf['X5']) ##modaX5=cdf.X5.mode()
print(modaX5)
cdf['X5'].fillna(modaX5, inplace=True)
```

```
18.0
```

Los valores NaN de X6:X11, son sustituidos por el valor más común de esa columna (moda).

```
for i in range(r6,r11+1):
  cdf.iloc[:,i].fillna(mode(cdf.iloc[:,i]), inplace=True) #modaX6=mode(cdf['X6'])
  print('moda x'+str(i),':',mode(cdf.iloc[:,i])) #print('x6',modaX6)
```

```
moda x6 : 0.0
moda x7 : 0.0
moda x8 : 0.0
moda x9 : 0.0
moda x10 : 0.0
moda x11 : 0.0
```

Los datos faltantes en X12:X23 se obtienen a través de la imputación media.

```
for i in range(r12,r23+1):
  cdf.iloc[:,i].fillna(mode(cdf.iloc[:,i]), inplace=True) #modaX6=mode(cdf['X6'])
  print('median x'+str(i),':',cdf.iloc[:,i].median()) #ndf.Length.median()
```

```
median x12 : 17989.0
median x13 : 18111.0
median x14 : 18163.0
median x15 : 17434.0
```

```
median x16 : 16338.0
median x17 : 14816.0
median x18 : 1831.0
median x19 : 1760.0
median x20 : 1400.0
median x21 : 1011.0
median x22 : 1017.0
median x23 : 1000.0
```

Debido a que Y representa la probabilidad real de incumplimiento como variable de respuesta lo considero un dato clave y donde haya elementos NaN contemplo más viable eliminarlos que imputarlos.

```
cdf[cdf['Y'].isnull()]
cdf['Y'].dropna(inplace=True)
cdf[cdf['Y'].isnull()]
```

ID	X1	X2	X3	X4	X5	X6	X7	X8	X9	...	X15	X16	X17	X18	X19	X20	X21	X22	X

0 rows × 25 columns

Haz doble clic (o pulsa Intro) para editar

```
print(cdf.isnull().sum())
```

```
ID     0
X1     0
X2     0
X3     0
X4     0
X5     0
X6     0
X7     0
X8     0
X9     0
X10    0
X11    0
X12    0
X13    0
X14    0
X15    0
X16    0
X17    0
X18    0
X19    0
X20    0
X21    0
X22    0
X23    0
```

```
Y      0
dtype: int64
```

Parte 3

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

Considero medianamente importante el grado de educación porque puede darnos una idea relativa sobre los ingresos de individuo. Considero importantes las variables que contienen la deuda (X12:X17) y los montos pagados anteriormente (X18:X23) puesto que nos proporciona un panorama general del flujo monetario que los individuos producen, contemplando cuánto se pagó y cuánto se debe. En general, los datos que considero más importantes son los relativos al historial de pagos pasados (X6:X11), si pagaba a tiempo o con cuánta demora, pues podría determinar el compromiso en cuanto al pago, como también Y, que es la predicción.

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

Se eliminaron y también reemplazaron los datos nulos o faltantes. Se eliminaron en los casos donde todas las columnas para una misma fila eran solamente ese tipo de valores, como también cuando tenían a partir de 10 datos faltantes, pues al no contar con básicamente la mitad de la información, no podía ser un registro representativo para el modelo. No se eliminaron todas las filas con valores nulos, con la finalidad de practicar y experimentar la limpieza de datos, porque en realidad eliminar todas las filas con al menos un valor nulo habría sido viable pues no poseen un peso porcentual muy grande o representativo para la muestra. De igual forma, cabe destacar que en este conjunto de datos particular no se encontró ningún caso donde no hubiese ningún dato para una misma fila, pero se mantiene el comando contemplando que podría haber datos futuros que sí apliquen. Los demás datos que no correspondían a los dos casos mencionados anteriormente se reemplazaron, para no perder un gran volumen e información y poder representar de mejor manera el modelo.

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

Ordenar en el sentido ascendente o descendente no parece ser necesario, pues el modelo o las transformaciones consideradas no lo requieren, mas sí hay que “poner los datos en orden” por así decirlo. Me refiero a estandarizar o asegurarnos que los valores (conforme las descripciones dadas de sus variables) son correctos, como se menciona más a detalle en el siguiente punto.

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 correspondan a las categorías existentes, que la edad asignada sea razonable (mayoría de edad) y que el monto de la deuda, pagos anteriores e historial de pago no posean valores negativos. Debía realizarse porque si los valores se salían de los estándares que en la misma descripción de las variables de definía, simplemente eran datos inexistentes o inválidos en cuanto a su clasificación, lo cual, en el conjunto de datos sí sucedía y debía corregirse, además de prevenir que sucediera en datos futuros. Un ejemplo de tales correcciones es, en las categóricas, si había la opción 'otra', se optó por ella (al reasignar categorías inválidas), pues es de alguna forma aquella que no está clara y parecía ser por tanto la nueva.

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

Como se menciona anteriormente, algunas filas donde no se contaba con toda o casi la mayor parte de la información fueron eliminadas, puesto que no resultaban representativas al carecer de tantos datos. Sin embargo, también se contemplaron otras opciones, como la imputación media para preservar la distribución de los datos y no sacrificar su demás información al eliminarlos. Asimismo, se asignaron las categorías alternativas (otra/o) cuando la capturada era inválida, se consideró un género aleatorio para sustituir los valores ilógicos (puesto que no se considera muy importante dicha variable, además que elegir solo un género podría causar sesgo y otros métodos generar valores no binarios), la moda en hábitos de pago (puesto que sí es posible observar ese tipo de tendencia), multiplicación por -1 para los elementos negativos (donde no era lógico que tuviesen tales valores) y eliminación de datos que se consideran muy importantes faltantes (como Y).

Productos de pago de Colab - Cancelar contratos

✓ 0 s completado a las 21:45