UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

MAESTRÍA EN CIENCAS DE DATOS

APRENDIZAJE AUTOMÁTICO

MAESTRO: JOSÉ ALBERTO BENAVIDES VÁZQUEZ

TAREA #3

ALUMNO: EDWIN MARTÍN ROMERO SILVA

MATRÍCULA: 1731276

Tarea 3

1)Crear funciones para calcular las medidas de tendencia central y de dispersión con Python.

Creé 5 funciones: Media, Mediana, Moda, Varianza, Desviación.

```
def media(dataframe, columna):
    promedio = dataframe[columna].mean()
    return promedio

promedio = media(df, "Loan Amount")
promedio
```

```
16848.902776336658
```

```
def mediana(df, columna):
    df = df.sort_values(columna)
    df = df.reset_index(drop = True)

    largo = len(df[columna])
    if largo % 2 == 0:
        posicion_1 = (largo/2)-1
        posicion_2 = (largo/2)
        mediana = (df.loc[posicion_1, columna] + df.loc[posicion_2, columna])/2
    else:
        posicion = ((largo + 1)/2) - 1
        mediana = df.loc[posicion, columna]
    return mediana

mediana = mediana(df, "Loan Amount")
mediana
```

```
16073
```

```
def moda(df, columna):
    conteo = df.groupby(columna)[[columna]].count()
    conteo.columns = ['Conteo']
    conteo = conteo.sort_values('Conteo', ascending = False).reset_index(drop = False)
    mod = conteo.loc[0, 'Conteo']
    return mod

moda = moda(df, "Term")
moda
```

```
43780
```

```
def varianza(df, columna):
    x_barra = df[columna].mean()
    n = len(df[columna])
    df['aux'] = (df[columna] - x_barra)**2
    varianza = df['aux'].sum()/n
    return varianza

varianza = varianza(df, "Term")
varianza
```

```
11.071696477311754
```

```
def desviacion_estandar(df, columna):
    x_barra = df[columna].mean()
    n = len(df[columna])
    df['aux'] = (df[columna] - x_barra)**2
    varianza = df['aux'].sum()/n
    varianza = varianza ** (1/2)
    return varianza

desviacion_estandar = desviacion_estandar(df, "Term")
desviacion_estandar
```

```
3.3274158858356966
```

2)Comprueba si tus variables de interés son conjuntos de datos paramétricos o no paramétricos

Comprobé con el Shapiro Test cuales variables se distribuyen de forma normal, utilicé un ciclo debido a la gran cantidad de variables y realicé el ejercicio únicamente en las variables tipo float e int.

```
df_numericas = df.select_dtypes(include = ['float64', 'int64']).columns

for column in df_numericas:
    p_value = stats.shapiro(df[column])[1]
    if p_value > 0.05:
        print(f'{column}: Paramétrica')
    else:
        print(f'{column}: No paramétrica')
```

En los resultados de este análisis obtuve que ninguna de las variables se distribuye Normal, sin embargo, en los siguientes análisis podremos conocer su distribución a través de histogramas.

```
Loan Amount: No paramétrica
Funded Amount: No paramétrica
Funded Amount Investor: No paramétrica
Term: No paramétrica
Interest Rate: No paramétrica
Home Ownership: No paramétrica
Debit to Income: No paramétrica
Delinquency - two years: No paramétrica
Inquires - six months: No paramétrica
Open Account: No paramétrica
Public Record: No paramétrica
Revolving Balance: No paramétrica
Revolving Utilities: No paramétrica
Total Accounts: No paramétrica
```

```
Total Received Interest: No paramétrica
Total Received Late Fee: No paramétrica
Recoveries: No paramétrica
Collection Recovery Fee: No paramétrica
Collection 12 months Medical: No paramétrica
Last week Pay: No paramétrica
Total Collection Amount: No paramétrica
Total Current Balance: No paramétrica
Total Revolving Credit Limit: No paramétrica
Loan Status: No paramétrica
Dif_monto_sol_monto_fin: No paramétrica
Dif_monto_sol_monto_aut: No paramétrica
Dif_monto_fin_monto_aut: No paramétrica
```

3)Calcula estadísticos descriptivos

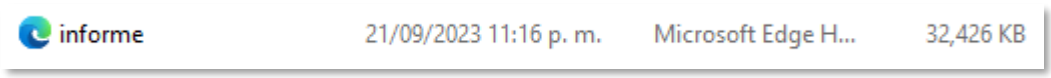
Las funciones del punto 1 nos ayudan mucho a practicar Python en caso de que lo tengamos un poco olvidado, pero una forma muy eficiente de calcular estadística descriptiva es la función describe(), ya que nos genera todos estos valores (a excepción de la moda) para todas las variables del dataframe.

```
df.describe()
```

	Loan Amount	Funded Amount	Funded Amount Investor	Term	Interest Rate	Home Ownership	Debit to Income	Delinquency - two years
count	67463.000000	67463.000000	67463.000000	67463.000000	67463.000000	67463.000000	67463.000000	67463.000000
mean	16848.902776	15770.599114	14621.799323	58.173814	11.846258	80541.502522	23.299241	0.327127
std	8367.865726	8150.992662	6785.345170	3.327441	3.718629	45029.120366	8.451824	0.800888
min	1014.000000	1014.000000	1114.590204	36.000000	5.320006	14573.537170	0.675299	0.000000
25%	10012.000000	9266.500000	9831.684984	58.000000	9.297147	51689.843335	16.756416	0.000000
50%	16073.000000	13042.000000	12793.682170	59.000000	11.377696	69335.832680	22.656658	0.000000
75%	22106.000000	21793.000000	17807.594120	59.000000	14.193533	94623.322785	30.048400	0.000000
max	35000.000000	34999.000000	34999.746430	59.000000	27.182348	406561.536400	39.629862	8.000000

Además, la librería pandas_profiling, nos permite exportar un informe con información importante de cada variable, como el promedio, los valores únicos, la cantidad de nulos, el valor mínimo, el valor máximo, la cantidad de 0s e infinitos.

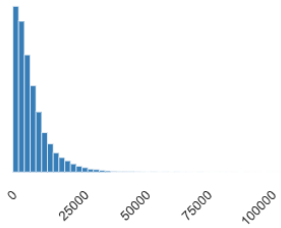
```python
import pandas as pd
from pandas_profiling import ProfileReport

df2 = pd.DataFrame(df)

# Crea un informe de perfil
profile = ProfileReport(df2)

# Genera el informe
profile.to_file("informe.html")
```



El reporte es muy extenso y detallado, solo mostraré unas cuantas variables para no alargar mucho este documento.

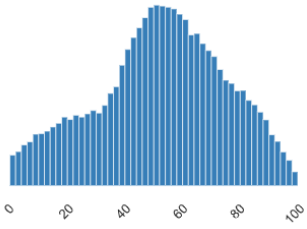
Revolving Balance
Real number (ℝ)

Distinct	20582	Minimum	0
Distinct (%)	30.5%	Maximum	116933
Missing	0	Zeros	7
Missing (%)	0.0%	Zeros (%)	< 0.1%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	7699.3424	Memory size	527.2 KiB



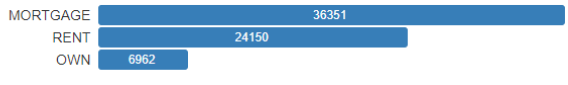
Revolving Utilities
Real number (ℝ)

Distinct	67458	Minimum	0.00517236
Distinct (%)	> 99.9%	Maximum	100.88005
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	52.889443	Memory size	527.2 KiB



RELACION_VIVIENDA
Categorical

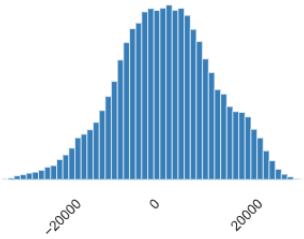
Distinct	3
Distinct (%)	< 0.1%
Missing	0
Missing (%)	0.0%
Memory size	527.2 KiB



Dif_monto_sol_monto_aut
Real number (ℝ)

HIGH_CORRELATION UNIQUE

Distinct	67463	**Minimum**	-33237.261
Distinct (%)	100.0%	**Maximum**	32638.059
Missing	0	**Zeros**	1
Missing (%)	0.0%	**Zeros (%)**	< 0.1%
Infinite	0	**Negative**	28337
Infinite (%)	0.0%	**Negative (%)**	42.0%
Mean	2227.1035	**Memory size**	527.2 KiB

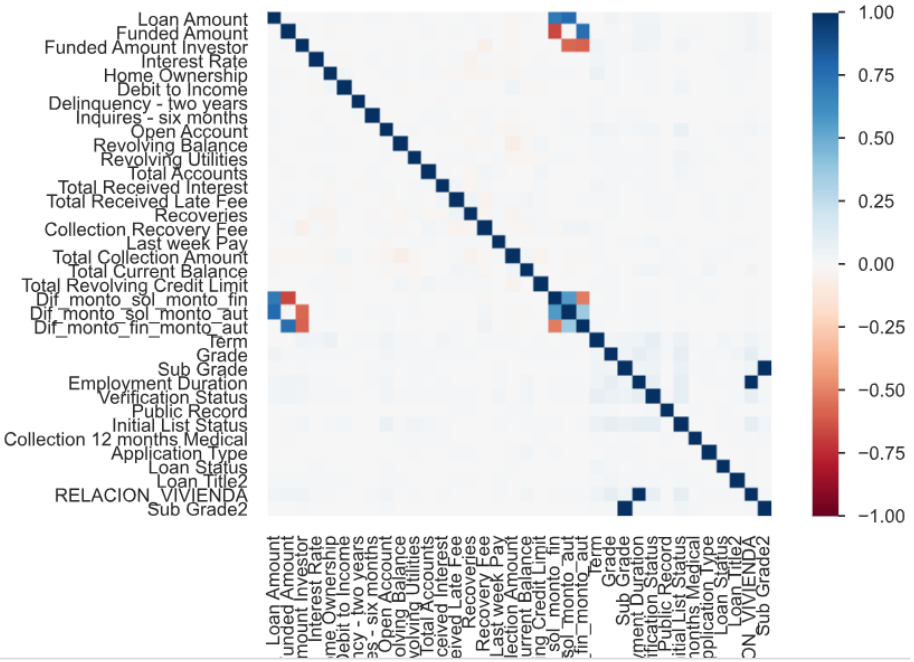


4)Haz una matriz de correlación y escribe algunas interpretaciones

Puedo generar la matriz de correlación con valores entre -1 y 1, pero al ser demasiadas variables los números no se aprecian muy bien, por lo que decidí sustituir los coeficientes de correlación por un mapa de calor que resalta los valores cercanos a |1|.

En mi experiencia, esta es una matriz de correlación fuera de lo común, ya que parece que las variables predictoras no están correlacionadas entre ellas. En realidad, esto es muy positivo, solo que no es muy común. Probablemente la persona que creó la base, previamente limpio la misma de variables linealmente correlacionadas.

Otra cosa que noto de esta matriz es que la variable target **LOAN STATUS** no esta correlacionada con ninguna variable, lo cual es malo, quiere decir que individualmente ninguna de estas variables es capaz de predecir el target.



5)Realiza algunas pruebas de hipótesis a partir de las conclusiones de la matriz de correlación

Esta base de datos y sus variables se enfocan en predecir la variable **LOAN STATUS** (BUENO/MALO).

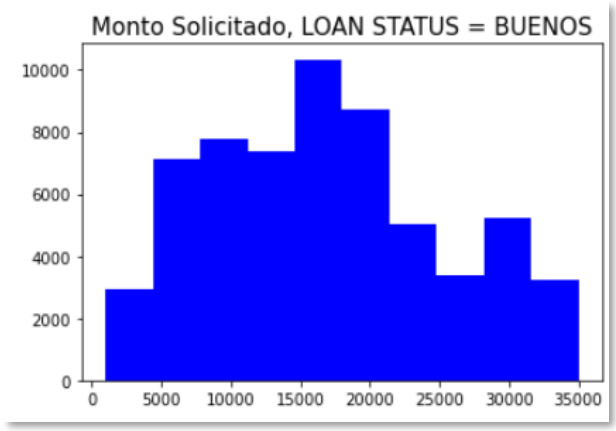
Puedo realizar una prueba para comprobar que el promedio de la variable **LOAN AMOUNT** (por ejemplo) en la población con LOAN STATUS = BUENO, es significativamente diferente al promedio de la misma variable en la población con LOAN STATUS = MALO.

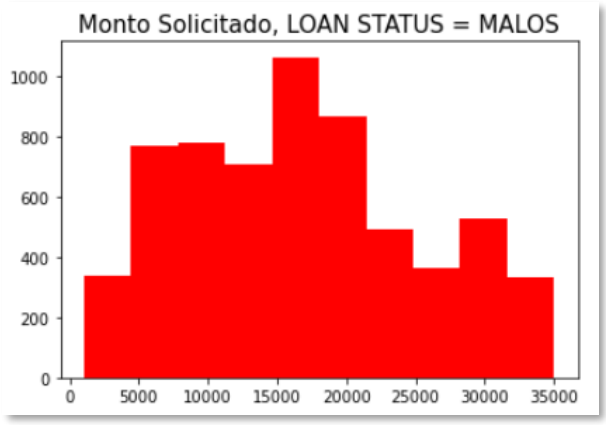
Al tratarse de únicamente 2 poblaciones (Buenos y Malos) se puede realizar con una prueba t.

Probé esta prueba con las variables LOAN AMOUNT y DIF_MON_SOL_MON_FIN. El resultado en ambas es que la diferencia entre las medias de ambas poblaciones no es significante, lo cual también podría indicar que no son variables capaces de predecir LOAN STATUS individualmente.

```python
# Supongamos que tienes dos arreglos de datos para cada grupo
loan_status_1 = df[df['Loan Status'] == 1]['Dif_monto_sol_monto_fin']
loan_status_0 = df[df['Loan Status'] == 0]['Dif_monto_sol_monto_fin']

print(loan_status_1.mean())
print(loan_status_0.mean())

# Realiza la prueba t de Student para comparar los dos grupos
t_statistic, p_value = stats.ttest_ind(loan_status_1, loan_status_0)

# Comprueba si el p-value es menor que un nivel de significancia dado (por ejemplo, 0.05)
alpha = 0.05
if p_value < alpha:
    print("El promedio de la población con LOAN STATUS = MALO es diferente de la población con LOAN STA
else:
    print("No hay evidencia suficiente para concluir que los promedios son diferentes.")
```

```
926.2509213267105
1093.8039920290091
No hay evidencia suficiente para concluir que los promedios son diferentes.
```





```python
import scipy.stats as stats

# Supongamos que tienes dos arreglos de datos para cada grupo
loan_status_1 = df[df['Loan Status'] == 1]['Loan Amount']
loan_status_0 = df[df['Loan Status'] == 0]['Loan Amount']

print(loan_status_1.mean())
print(loan_status_0.mean())

# Realiza la prueba t de Student para comparar los dos grupos
t_statistic, p_value = stats.ttest_ind(loan_status_1, loan_status_0)

# Comprueba si el p-value es menor que un nivel de significancia dado (por ejemplo, 0.05)
alpha = 0.05
if p_value < alpha:
    print("El promedio de la población con LOAN STATUS = MALO es diferente de la población con LOAN STA
else:
    print("No hay evidencia suficiente para concluir que los promedios son diferentes.")
```

```
16731.67441115206
16860.853092025744
No hay evidencia suficiente para concluir que los promedios son diferentes.
```