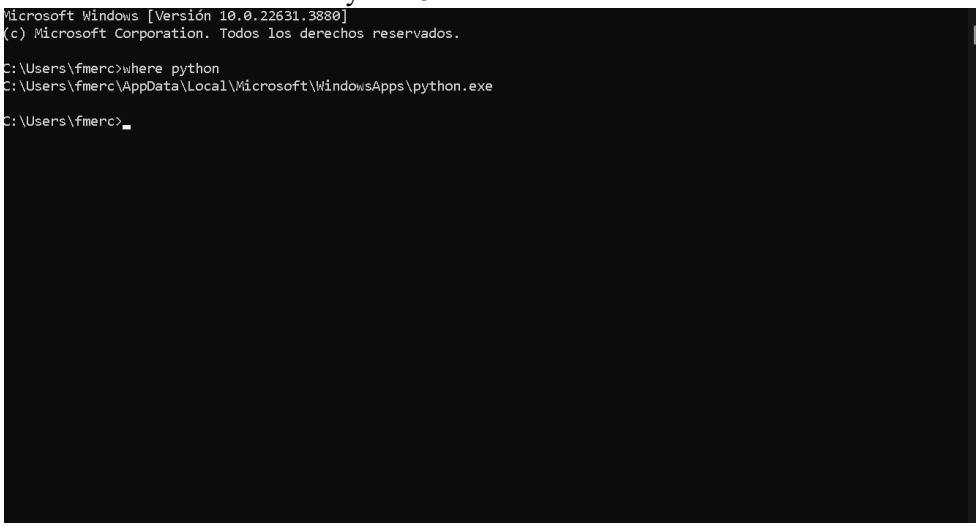
Taller 1 – Python y datos: Analítica Computacional de Datos

Nombre: Isabela Castillo Mercado

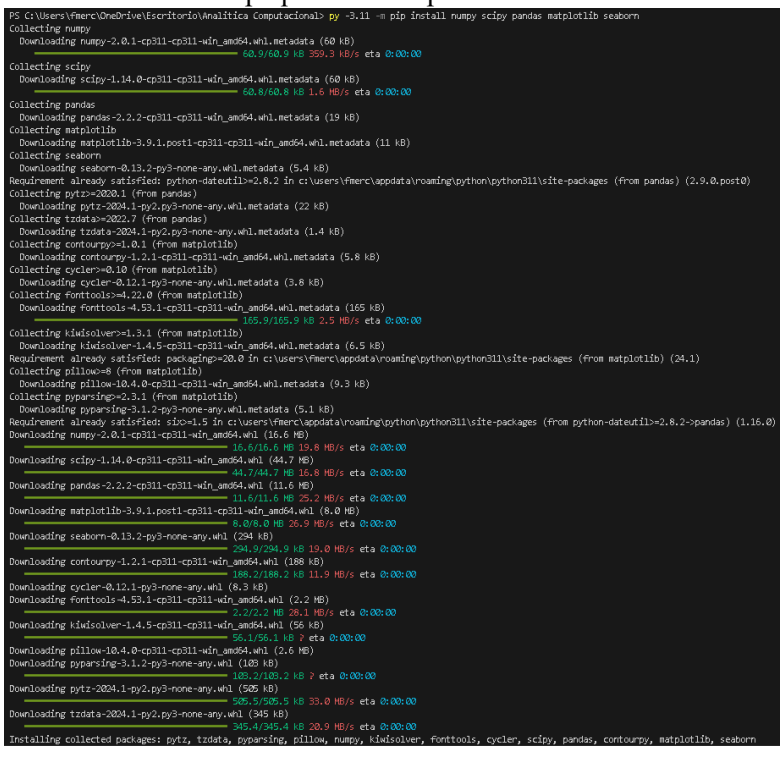
Código: 201813093

1. Se realizo la Instalación de Python 3.11



(*) Aclaración: Mi usuario en este pc es fmerc.

2. Dentro de Visual Studio Code realizamos todo lo mencionado em el enunciado del taller y la instalación de los paquetes correspondientes.



Como se puede observar en la salida del terminal, tuve que especificar la versión python la que instalaba los paquetes, debido a que ya hacia uso de pyehon en Visual Studio Code y tenia la versión 3.92

```
Location: C:\Users\fmerc\AppData\Local\Programs\Python\Python311\Lib\site-packages
Requires: numpy
Required-by: statsmodels
---
Name: pandas
Version: 2.2.2
Summary: Powerful data structures for data analysis, time series, and statistics
Home-page: https://pandas.pydata.org
Author:
Author-email: The Pandas Development Team <pandas-dev@python.org>
License: BSD 3-Clause License

Copyright (c) 2008-2011, AQR Capital Management, LLC, Lambda Foundry, Inc. and PyData Development Team
All rights reserved.
```

```
---
Name: matplotlib
Version: 3.9.1.post1
Summary: Python plotting package
Home-page:
Author: John D. Hunter, Michael Droettboom
Author-email: Unknown <matplotlib-users@python.org>
License: License agreement for matplotlib versions 1.3.0 and later
============================================================
```

```
Name: seaborn
Version: 0.13.2
Summary: Statistical data visualization
Home-page:
Author:
Author-email: Michael Waskom <mwaskom@gmail.com>
License:
Location: C:\Users\fmerc\AppData\Local\Programs\Python\Python311\Lib\site-packages
Requires: matplotlib, numpy, pandas
Required-by:
PS C:\Users\fmerc\OneDrive\Escritorio\Analitica Computacional>
```

```
Name: pandas
Version: 2.2.2
Summary: Powerful data structures for data analysis, time series, and statistics
Home-page: https://pandas.pydata.org
Author:
Author-email: The Pandas Development Team <pandas-dev@python.org>
License: BSD 3-Clause License

Copyright (c) 2008-2011, AQR Capital Management, LLC, Lambda Foundry, Inc. and PyData Development Team
All rights reserved.
```

```
Name: scipy
Version: 1.11.4
Summary: SciPy: Scientific Library for Python
Home-page: https://www.scipy.org
Author:
Author-email:
License: BSD
Location: c:\users\fmerc\anaconda3\lib\site-packages
Requires: numpy
Required-by: cvxpy, datashader, ecos, gensim, osqp, pyportfolioopt, qdldl, scikit-image, scikit-learn, scikit-plot, scs, shap, statsmodels, xgboost
```

2. **Exploración de Datos en Python**

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

Python

Se importan las librerías numpy como np y pandas como pd para su uso en el resto del cuaderno.

```python
df = pd.read_csv('.\\BankChurn.csv')
```

Python

Se lee e importa el archivo CSV como un DataFrame utilizando la función read_csv de la librería pandas (pd).

```python
df.shape
```

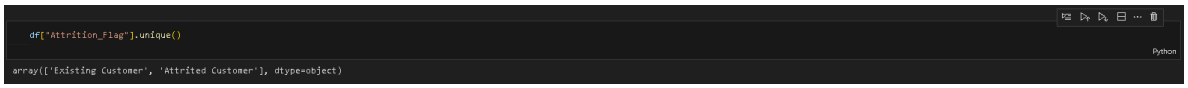
Python

Muestra la dimensión de la data frame

```
df.head()
```

	Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	A
0	Existing Customer	45	M	3	High School	Married	$60K - $80K	Blue	39	5	1	3	12691.0	777	
1	Existing Customer	49	F	5	Graduate	Single	Less than $40K	Blue	44	6	1	2	8256.0	864	
2	Existing Customer	51	M	3	Graduate	Married	$80K - $120K	Blue	36	4	1	0	3418.0	0	
3	Existing Customer	40	F	4	High School	Unknown	Less than $40K	Blue	34	3	4	1	3313.0	2517	
4	Existing Customer	40	M	3	Uneducated	Married	$60K - $80K	Blue	21	5	1	0	4716.0	0	

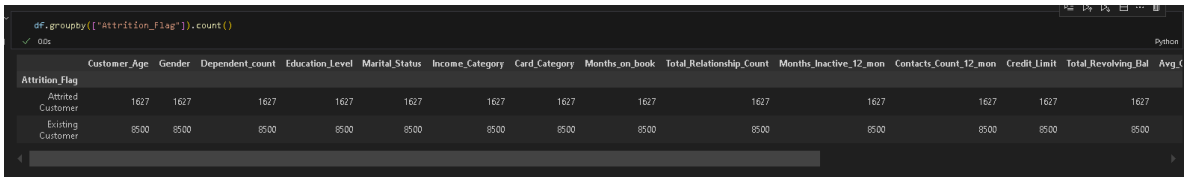
La función head muestra las primeras 5 filas con todas las columnas del DataFrame, comenzando con el índice cero, ya que Python utiliza indexación basada en cero.



```
df["Attrition_Flag"].unique()
```

```
array(['Existing Customer', 'Attrited Customer'], dtype=object)
```

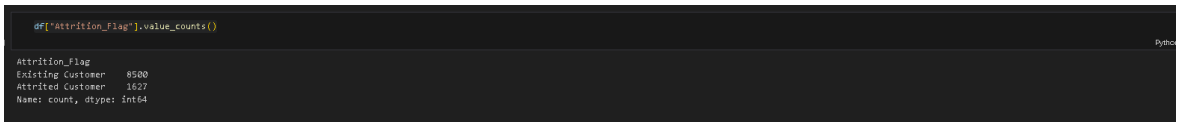
La función unique en Python muestra los valores únicos de una columna, eliminando duplicados. Dado que la columna Attrition_Flag son 2 categorías, te devolverá solo los valores únicos.



```
df.groupby(["Attrition_Flag"]).count()
```

Attrition_Flag	Customer_Age	Gender	Dependent_count	Education_Level	Marital_Status	Income_Category	Card_Category	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg_
Attrited Customer	1627	1627	1627	1627	1627	1627	1627	1627	1627	1627	1627	1627	1627	
Existing Customer	8500	8500	8500	8500	8500	8500	8500	8500	8500	8500	8500	8500	8500	

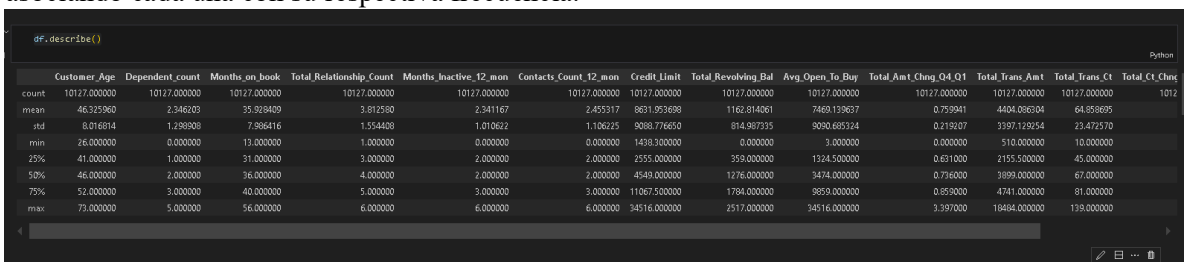
La función groupby agrupa por categoría y cuenta la cantidad de filas que tiene cada categoría en la columna Attrition_flag.



```
df["Attrition_Flag"].value_counts()
```

```
Attrition_Flag
Existing Customer    8500
Attrited Customer    1627
Name: count, dtype: int64
```

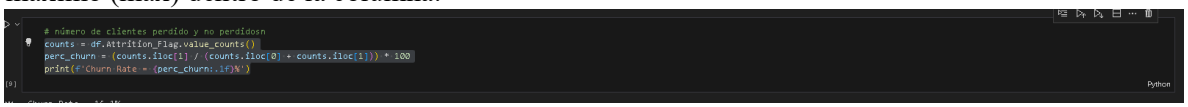
La función value_counts muestra el número de observaciones por categoría en Attrition_Flag, asociando cada una con su respectiva frecuencia.



```
df.describe()
```

	Customer_Age	Dependent_count	Months_on_book	Total_Relationship_Count	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit	Total_Revolving_Bal	Avg_Open_To_Buy	Total_Amt_Chng_Q4_Q1	Total_Trans_Amt	Total_Trans_Ct	Total_Ct_Chng
count	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	10127.000000	1012
mean	46.325960	2.346203	35.928409	3.812580	2.341167	2.455317	8631.953698	1162.814061	7469.139637	0.759941	4404.086304	64.858695	
std	8.016814	1.298908	7.986416	1.554408	1.010622	1.106225	9088.776650	814.987335	9090.685324	0.219207	3397.129254	23.472570	
min	26.000000	0.000000	13.000000	1.000000	0.000000	0.000000	1438.300000	0.000000	3.000000	0.000000	510.000000	10.000000	
25%	41.000000	1.000000	31.000000	3.000000	2.000000	2.000000	2555.000000	359.000000	1324.500000	0.631000	2155.500000	45.000000	
50%	46.000000	2.000000	36.000000	4.000000	2.000000	2.000000	4549.000000	1276.000000	3474.000000	0.736000	3899.000000	67.000000	
75%	52.000000	3.000000	40.000000	5.000000	3.000000	3.000000	11067.500000	1784.000000	9859.000000	0.859000	4741.000000	81.000000	
max	73.000000	5.000000	56.000000	6.000000	6.000000	6.000000	34516.000000	2517.000000	34516.000000	3.397000	18484.000000	139.000000	

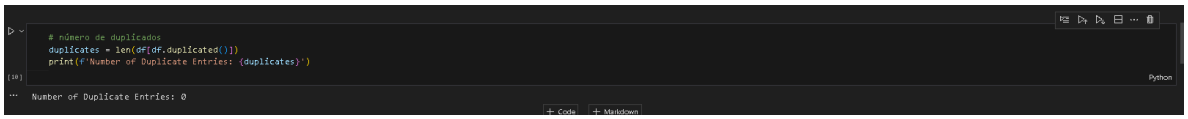
La función describe proporciona un análisis descriptivo de cada columna con atributos cuantitativos, mostrando el número de observaciones (count), el promedio (mean), la desviación estándar (std), el valor mínimo (min), los cuartiles del 25%, 50% y 75%, y el valor máximo (max) dentro de la columna.



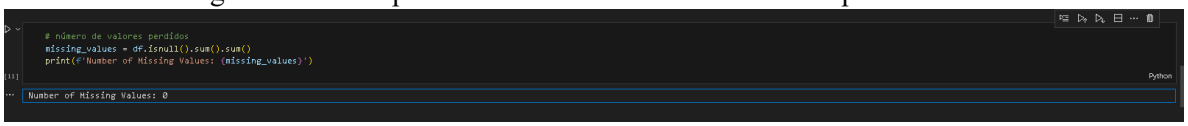
```
# número de clientes perdído y no perdídosn
counts = df.Attrition_Flag.value_counts()
perc_churn = (counts.iloc[1] / (counts.iloc[0] + counts.iloc[1])) * 100
print(f'Churn Rate = {perc_churn:.1f}%')
```

La función calcula la tasa de abandono como el porcentaje de Attrited Customer sobre el total de clientes (la suma de Existing Customer y Attrited Customer), multiplicado por 100. Utiliza iloc para seleccionar valores específicos en el DataFrame count (almacena el conteo por categoría de la columna Attrition_Flag utilizando la función value_counts), donde Existing Customer está en la fila 1 y Attrited Customer en la fila 0. Finalmente, imprime el valor

calculado en perc_churn junto con el texto "Churn Rate", formateado como porcentaje con un decimal.

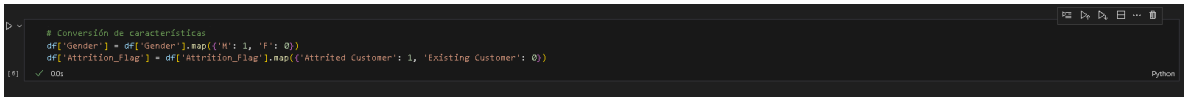


Calcula el número de duplicados (Calcula el largo de un arreglo) que se encuentran en toda la data frame de igual manera imprimimos el resultado con la función `print`

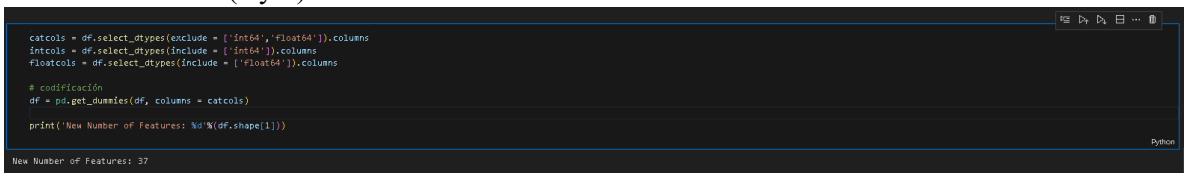


La función is_null() identifica los valores nulos en cada celda del DataFrame, asignando un valor booleano: True para nulos (representado internamente como 1) y False para no nulos (representado como 0). Primero, se suman los valores nulos por columna, y luego se suman estas sumas para obtener el total de valores faltantes en todo el DataFrame. Finalmente, este total se imprime junto con un texto descriptivo usando print.

Esta función cuenta y muestra la cantidad de columnas, filas y la frecuencia de cada tipo de dato en el DataFrame df.

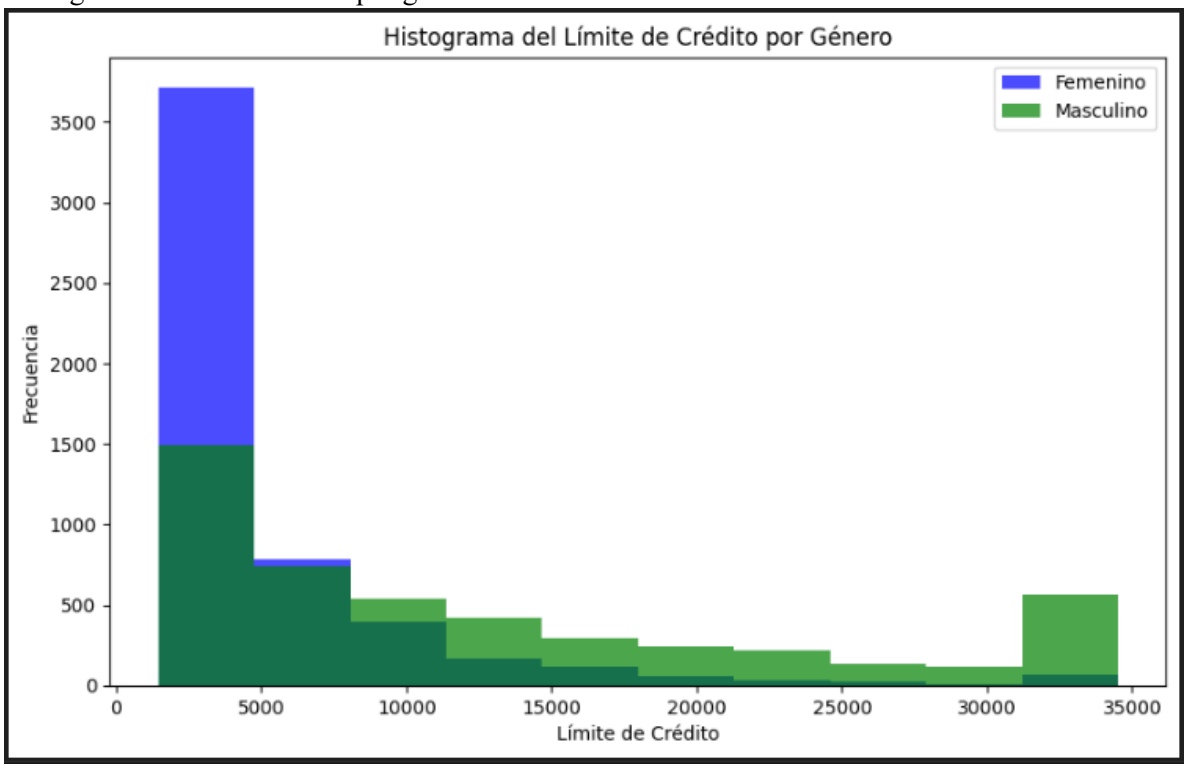


Esta función convierte los valores categóricos de las columnas Gender y Attrition_Flag en valores numéricos (1 y 0) en el DataFrame df.



Esta función identifica las columnas categóricas, enteras y de punto flotante en df, aplica codificación one-hot (get_dummies) a las categóricas, y luego imprime el nuevo número de columnas del DataFrame.

Histograma limite de Credito por genero



Realice un histograma, segmentado por genero para tener una noción de si la concentración del límite de créditos se ve impactado por el género. Si bien podemos observar que en los valores más menores de crédito se concentran mas hombres que mujeres, de lo contrario en valores más altos de crédito se concentran más mujeres que hombres.

El análisis, no tiene un sustento estadístico, por lo que simplemente es una noción cualitativa de la grafica

El análisis de la BBDD BikePrices.csv se encuentra como archivo de soporte BikePrices.ipynb

3. **Números Aleatorios y Bondad de Ajuste en Python**

Explicación de cada celda de codigo del Notebook de data-gen.ipynb

Generación de muestras de una variable aleatoria normal usando numpy.

```
import numpy as np

mu = 3
sigma = 0.5
n=1000
vals = np.random.normal(loc=mu, scale=sigma, size=n)
print(vals)
```

```
[3.28682338 4.26756866 3.71678232 3.81757272 4.21016255 3.68898727
 2.85211509 2.49886586 3.02548503 1.97671357 3.5793983  2.84402638
 3.2307679  1.84425534 2.99674606 3.55157335 3.93517619 3.18622204
 3.58353701 2.73869864 2.39174033 3.80224303 3.7718596  2.98042721
 2.83348933 2.42641157 3.5765898  3.07559501 4.40624612 2.63191767
 3.10976763 3.2487942  2.53322702 3.20548954 2.73756619 3.85998572
```

La función de numpy genera un arreglo de 1000 números (n), ajustado a una normal con media 3 y desviación estándar de 0.5. y después se imprime el arreglo

```
import pandas as pd

df = pd.DataFrame(vals)
print(df.describe())
```

```
                0
count  1000.000000
mean      3.023062
std       0.492066
min       1.281704
25%       2.688698
50%       3.022621
75%       3.351697
max       4.747871
```

Realiza un analisis descriptivo de el arreglo vals que se genero

```
import matplotlib.pyplot as plt
count, bins, ignored = plt.hist(x=vals, bins=30)
plt.title('Histograma de tiempos de servicio')
plt.xlabel('Tiempos de servicio')
plt.ylabel('Frecuencia')
plt.show()

print(bins)
print(count)
```

La función realiza un histograma, con particiones de 30, y el comando plt llama las función de matplotlib para asignar titulo, eje x y eje y.El Histograma replica la forma de una Normal.

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

# Normal (mu=0, sigma=1)
x= np.arange(-4,4,0.001)
plt.plot(x, norm.pdf(x))
plt.title('Función de densidad normal estándar')
plt.xlabel('Valores')
plt.ylabel('Densidad')
plt.show()
```

Esta función genera y visualiza la función de densidad de probabilidad de una distribución normal estándar (media 0, desviación estándar 1) usando matplotlib.

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
# Normal (mu=3, sigma=1)
x= np.arange(0,6,0.001)
plt.plot(x, norm.pdf(x,loc=3))
plt.title('Función de densidad normal(mu=3, sigma=1)')
plt.xlabel('Valores')
plt.ylabel('Densidad')
plt.show()
```

Esta función genera y visualiza la función de densidad de probabilidad de una distribución normal con media 3 y desviación estándar 1 usando matplotlib.

```python
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

# Normal (mu=0, sigma=1)
x= np.arange(0,6,0.001)
plt.plot(x, norm.pdf(x,loc=3,scale=0.5))
plt.title('Función de densidad normal(mu=3, sigma=0.5)')
plt.xlabel('Valores')
plt.ylabel('Densidad')
plt.show()
```

Esta función genera y visualiza la función de densidad de probabilidad de una distribución normal con media 3 y desviación estándar 0.5 usando matplotlib.

QQ plot usando statsmodels.

```python
import statsmodels.api as sm
import scipy.stats as stats
fig = sm.qqplot(data = vals, dist=stats.norm, loc=3, scale=0.5, line="45")
plt.show()
```

Este código genera un gráfico Q-Q (cuantiles-cuantiles) para comparar la distribución de los datos en vals con una distribución normal con media 3 y desviación estándar 0.5, utilizando statsmodels.

```python
import statsmodels.api as sm
import scipy.stats as stats
fig = sm.qqplot(data = vals, dist=stats.norm, loc=3, scale=1, line="45")
plt.show()
```

Este código genera un gráfico Q-Q (cuantiles-cuantiles) para comparar la distribución de los datos en vals con una distribución normal con media 3 y desviación estándar 1, utilizando statsmodels.

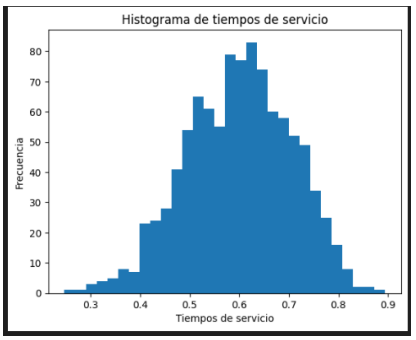
```python
import statsmodels.api as sm
import scipy.stats as stats
fig = sm.qqplot(data = vals, dist=stats.norm, loc=0, scale=1, line="45")
plt.show()
```

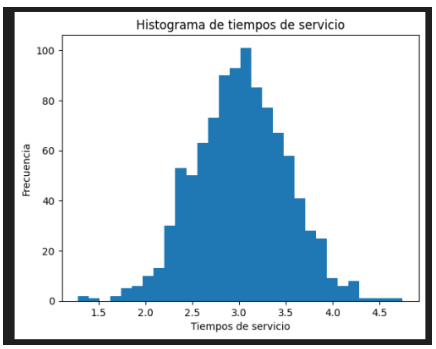
Este código genera un gráfico Q-Q (cuantiles-cuantiles) para comparar la distribución de los datos en vals con una distribución normal con media 0 y desviación estándar 1, utilizando statsmodels.

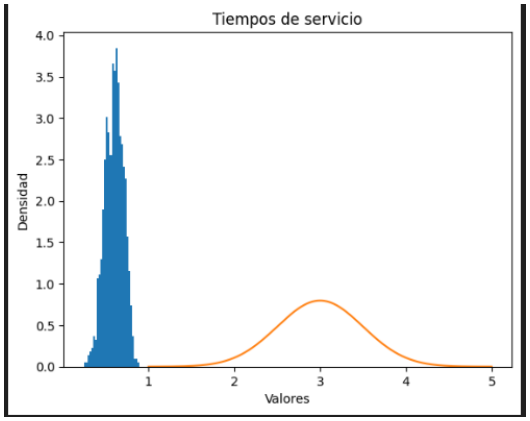
```python
import matplotlib.pyplot as plt
# histograma
count, bins, ignored = plt.hist(x=vals, bins=30, density=True)

# densidad normal(3, 0.5)
x= np.arange(1,5,0.001)
plt.plot(x, norm.pdf(x,loc=3,scale=0.5))

plt.title('Tiempos de servicio')
plt.xlabel('Valores')
plt.ylabel('Densidad')
plt.show()
```

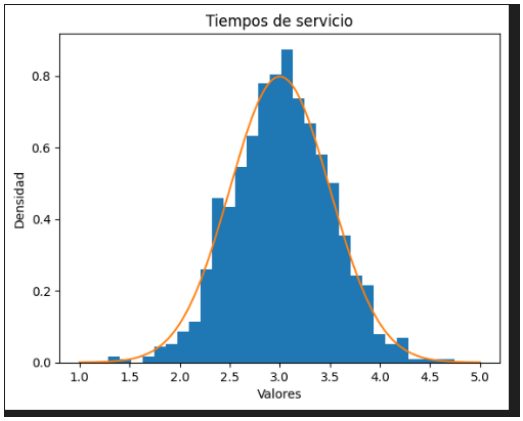
Este código crea un histograma de los datos en vals con 30 intervalos y superpone la función de densidad de una distribución normal con media 3 y desviación estándar 0.5, y luego muestra el gráfico con matplotlib.

Cambio a distribución beta









Vemos que en el cambio de distribución la grafica un poco con tipo de rezago hacia la derecha, haciendo una contraste de estas graficas podemos observar que una grafica se ajusta a la normal y se observa como la distribución beta no se justa a la normal