Informe Examen Transversal Machine Learning

Introducción

Fase 1 - Entendimiento del caso

Este informe presenta los resultados del análisis de datos sobre el banco monopoly,podemos apreciar en la siguiente imagen el contexto del negocio, una diccionario con las distintas variables que se usaran a lo largo de la exploración de datos, limpieza de datos y fases posteriores ademas de los objetivos generales y específicos propuestos.

Resumen del contexto

El cliente referido como "Banco monopoly" me contrato como analista de datos, Logrando asi identificar distintos aspectos en el contexto de negocios,identificando que se necesita una estrategia para atraer potenciales nuevos clientes al banco para ello, Es importante analizar el uso de los productos financieros. Finalmente, podemos decir que la base de datos tiene 574 variables (columnas) y 51.124 registros (filas).
También es importante destacar que cada fila contiene registros únicos de clientes , todos los datos tratados corresponden a una base que comprende un total de 12 meses de datos de los clientes banco monopoly de forma mensual por cada cliente. Ademas para este caso se usara la metodologia CRISP-DM para una mejor organizacion ya que se dividen en distintas fases y mejor entendimiento del caso.

Dato	Descripción
region	region de residencia
renta	renta de cliente
sexo	sexo
Edad	Edad
Adicional	indicador de tenecia de TC adicionales
Antiguedad	Antiguedad de clientes (meses)
consumo	indicador de credito consumo
debito	indicador de tenecia de TD
ctacte	indicador de cuenta corriente
cuentas	numero de cuentas que tiene el cliente
hipotecario	indicador de credito hipotecario
TC	numero de TC que tiene el cliente

Objetivos generales

- Identificar grupos de clientes con comportamientos similares para personalizar productos y servicios.
- identificar que grupo de clientes que realiza mayor compra de productos y servicios.
- Identificar los productos financieros que generan más ingresos o que tienen mayor aceptación entre los clientes.

Objetivos especificos

- Clasificar a los clientes según su nivel de riesgo de crédito en base a indicador de cuenta corriente
- Predecir el monto total de compras anuales de un cliente en función de su edad.

Fase 2 - Exploración de los datos

En la exploración de los datos no centramos en conocer en dataframe y sus distintas columnas y filas junto con sus distintos tipos de datos que contienen

```
[4] df.columns

Index(['Id', 'Subsegmento', 'Sexo', 'Region', 'Edad', 'Renta', 'Antiguedad',
       'Internauta', 'Adicional', 'Dualidad',
       ...
       'PagoNac_T01', 'PagoInt_T01', 'EeccNac_T01', 'EeccInt_T01', 'UsoL1_T01',
       'UsoL2_T01', 'UsoLI_T01', 'IndRev_T01', 'target', 'Unnamed: 574'],
      dtype='object', length=575)

[5] df.shape

(51124, 575)

[6] df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51124 entries, 0 to 51123
Columns: 575 entries, Id to Unnamed: 574
dtypes: float64(509), int64(53), object(13)
memory usage: 224.3+ MB
```

podemos ver en el siguiente codigo los distintos datos con los rangos intercuartiles ademas de el valor min y el valor maximo.

```
[7] df.describe()
```

	Id	Subsegmento	Region	Edad	Renta	Antiguedad	Internauta	Adicional	Dualidad	Monoproducto	...	ColMx_T01	PagoNac_T01	PagoInt_T01	EeccNac_T01	EeccInt_T01	
count	51124.000000	51124.000000	51071.000000	51124.000000	3.775900e+04	51124.000000	51124.000000	51124.000000	51124.000000	51124.000000	...	5.112400e+04	5.112400e+04	5.112400e+04	5.112400e+04	51124.000000	5.1
mean	25562.500000	182.024274	10.828220	38.702879	6.630771e+05	38.896154	0.684199	0.256181	0.381347	0.063141	...	5.237914e+03	7.637553e+04	1.734930e+03	1.939488e+05	7.323155	1.8
std	14758.371918	29.276596	3.392703	13.302573	4.092795e+05	35.672549	0.464839	0.436527	0.485722	0.243218	...	4.852871e+04	1.490256e+05	4.235368e+04	2.884980e+05	108.161194	2.8
min	1.000000	151.000000	1.000000	9.000000	1.000000e+00	6.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000e+00	0.000000e+00	0.000000e+00	-1.861866e+06	-7886.760000	-3.7
25%	12781.750000	160.000000	9.000000	28.000000	4.199990e+05	14.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000e+00	5.000000e+03	0.000000e+00	3.198100e+04	0.000000	1.7
50%	25562.500000	170.000000	13.000000	35.000000	5.670120e+05	25.000000	1.000000	0.000000	0.000000	0.000000	...	0.000000e+00	3.400150e+04	0.000000e+00	9.223050e+04	0.000000	8.1
75%	38343.250000	210.000000	13.000000	46.000000	8.149035e+05	54.000000	1.000000	1.000000	1.000000	0.000000	...	0.000000e+00	9.200000e+04	0.000000e+00	2.359780e+05	0.000000	2.3
max	51124.000000	959.000000	13.000000	104.000000	1.308933e+07	324.000000	1.000000	1.000000	1.000000	1.000000	...	2.072818e+06	8.697782e+06	4.219680e+06	6.911556e+06	3929.810000	6.9

8 rows × 562 columns

En esta imagen se observa las primeras 5 filas y las ultimas 5 filas del dataframe ademas de los distintos datos que contienen cada una de las columnas

```
df.head()
```

	Id	Subsegmento	Sexo	Region	Edad	Renta	Antiguedad	Internauta	Adicional	Dualidad	...	PagoNac_T01	PagoInt_T01	EeccNac_T01	EeccInt_T01	UsoL1_T01	UsoL2_T01	UsoLI_T01	IndRev_T01	target	Unnamed: 574
0	1	160	M	13.0	43	NaN	130	1	1	0	...	33000	0.0	1099866.0	0.0	1099866.0	15080	0.0	R	0	NaN
1	2	160	H	13.0	46	143640.0	69	1	0	0	...	300000	0.0	214592.0	0.0	214592.0	83596	0.0	R	0	NaN
2	3	170	H	13.0	45	929106.0	24	1	1	0	...	216676	0.0	0.0	0.0	7400.0	0	0.0	T	0	NaN
3	4	151	H	13.0	46	172447.0	134	0	1	0	...	60000	0.0	272762.0	0.0	272762.0	10591	0.0	R	0	NaN
4	5	170	H	13.0	46	805250.0	116	0	1	1	...	272925	0.0	249562.0	0.0	75339.0	377782	0.0	R	0	NaN

5 rows × 575 columns

+ Código + Texto

```
df.tail()
```

	Id	Subsegmento	Sexo	Region	Edad	Renta	Antiguedad	Internauta	Adicional	Dualidad	...	PagoNac_T01	PagoInt_T01	EeccNac_T01	EeccInt_T01	UsoL1_T01	UsoL2_T01	UsoLI_T01	IndRev_T01	target	Unnamed: 574
51119	51120	160	H	13.0	51	364978.0	57	1	1	1	...	300000	0.0	478320.0	0.0	478320.0	12668	0.0	R	0	NaN
51120	51121	170	H	13.0	51	625376.0	39	1	0	0	...	166098	0.0	166098.0	0.0	0.0	572363	0.0	R	0	NaN
51121	51122	160	H	13.0	47	806220.0	153	1	1	0	...	18891	0.0	9652.0	0.0	9652.0	16241	0.0	R	0	NaN
51122	51123	160	M	13.0	47	NaN	11	1	0	0	...	26528	0.0	24638.0	0.0	24638.0	84982	0.0	R	0	NaN
51123	51124	170	H	13.0	51	840878.0	75	1	1	0	...	12360	0.0	12360.0	0.0	18500.0	0	0.0	R	0	NaN

5 rows × 575 columns

En estas imágenes se puede apreciar el análisis de esta dos variables las cuales son especificadas en los objetivos específicos

Analisis de la variable Edad

```
[10] df['Edad'].mean()
```

```
38.70287927392223
```

```
df['Edad'].median()
```

```
35.0
```

```
[12] df['Edad'].mode()
```

	Edad
0	27

dtype: int64

Analisis de la variable Ctacte

```
[70] df['Ctacte'].mean()
```

```
0.9258415938757484
```

```
[71] df['Ctacte'].median()
```

```
1.0
```

```
[72] df['Ctacte'].mode()
```

	Ctacte
0	1

dtype: int64

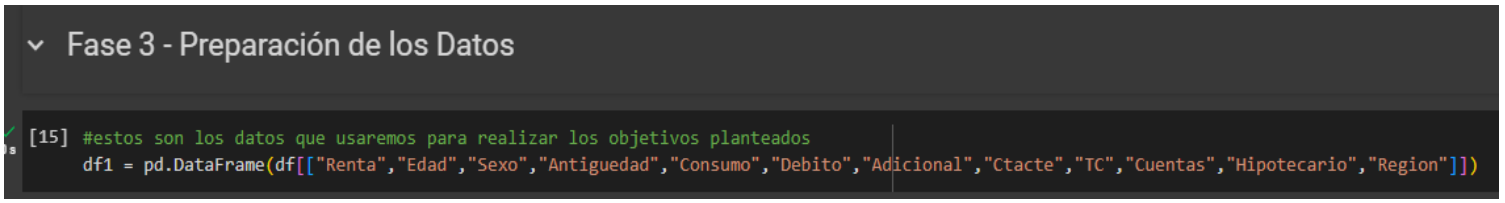
Análisis de datos

Fase 3-Preparación de datos

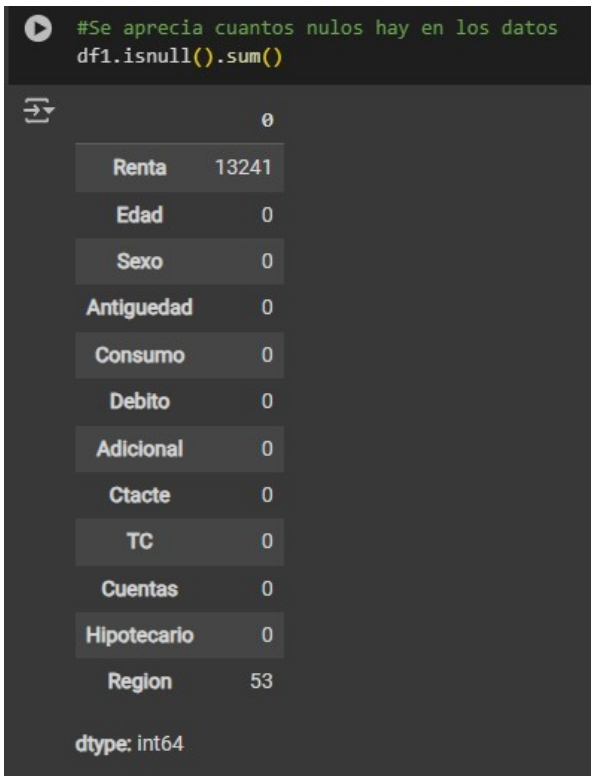
Los datos fueron limpiados y preparados utilizando las siguientes técnicas:

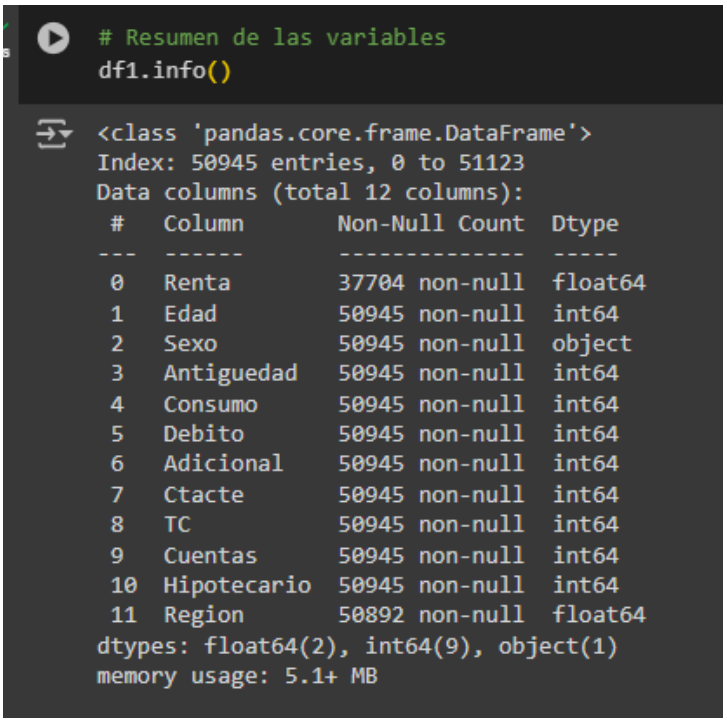
* Detección de valores faltantes
* Eliminación de datos duplicados
* Normalización de los datos
* Manejo de valores atípicos

Podemos apreciar en la siguiente imagen que en la tercera fase se realiza una copia con las variables a utilizar descartando así las que no nos sirven



Siguiendo con la preparación de datos verificamos si hay datos faltantes en las distintas variables en nuestro caso hay datos faltantes en renta y región,ademas se puede apreciar de que tipo de datos son los distintos datos.





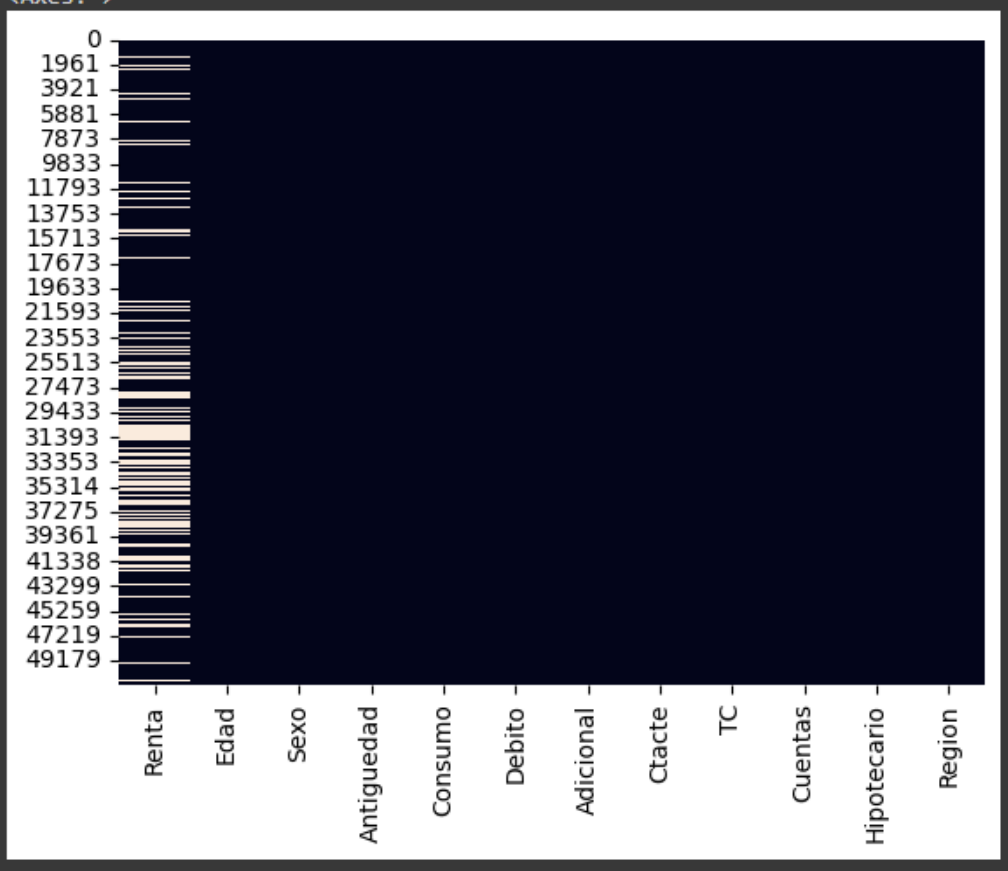
La imagen nos muestra una descripción de las distintas variables como la media, su valor mínimo,su valor Máximo,

```
[18] # Estadísticas descriptivas
     df1.describe()
```

	Renta	Edad	Antiguedad	Consumo	Debito	Adicional	Ctacte	TC	Cuentas	Hipotecario	Region
count	3.770400e+04	50945.000000	50945.000000	50945.000000	50945.000000	50945.000000	50945.000000	50945.000000	50945.000000	50945.000000	50892.000000
mean	6.632504e+05	38.545255	38.682756	0.000883	0.876396	0.255766	0.925842	1.733045	1.408323	0.138012	10.827694
std	4.091703e+05	13.051162	35.410446	0.029708	0.329132	0.436295	0.262031	0.877816	0.550705	0.344916	3.393353
min	1.000000e+00	19.000000	6.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000
25%	4.200000e+05	28.000000	14.000000	0.000000	1.000000	0.000000	1.000000	1.000000	1.000000	0.000000	9.000000
50%	5.672410e+05	35.000000	25.000000	0.000000	1.000000	0.000000	1.000000	2.000000	1.000000	0.000000	13.000000
75%	8.150000e+05	46.000000	54.000000	0.000000	1.000000	1.000000	1.000000	2.000000	2.000000	0.000000	13.000000
max	1.308933e+07	80.000000	324.000000	1.000000	1.000000	1.000000	1.000000	12.000000	5.000000	1.000000	13.000000

Se puede apreciar los valores faltantes de forma más clara gracias a que es un gráfico de calor

```
[22] #Identificamos los missing values visualmente
     sns.heatmap(df1.isnull(), cbar=False)
```



Se procesa la variable renta rellenando los valores faltantes ademas de remplazar las comas por puntos para un mayor procesamiento,ademas se rellenan los datos faltantes Con knn imputer el cual ayuda a rellenar de forma mas uniforme los datos.

exploramos los datos de renta

```
[23] df1.Renta.unique()
```

```
array([    nan, 143640., 929106., ..., 625376., 806220., 840878.])
```

```
df1.Renta.info()
```

```
<class 'pandas.core.series.Series'>
Index: 50945 entries, 0 to 51123
Series name: Renta
Non-Null Count  Dtype
--------------  -----
37704 non-null  float64
dtypes: float64(1)
memory usage: 796.0 KB
```

```
[25] df1.Renta.describe()
```

	Renta
count	3.770400e+04
mean	6.632504e+05
std	4.091703e+05
min	1.000000e+00
25%	4.200000e+05
50%	5.672410e+05
75%	8.150000e+05
max	1.308933e+07

dtype: float64

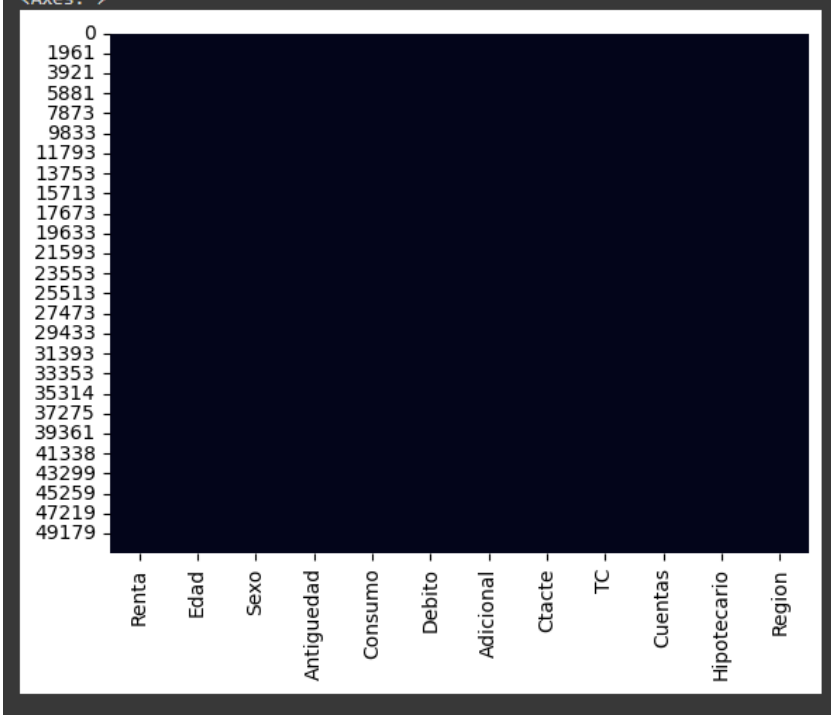
```
[28] ## primero cambiamos las comas por puntos
     df1['Renta'] = df1['Renta'].astype(str).str.replace(',', '.').astype(float)
```

```
[32] #rellenamos los datos faltantes con knn inputer
     knn_imputer = KNNImputer(n_neighbors=3, weights="uniform")
     df1[['Renta']] = knn_imputer.fit_transform(df1[['Renta']])
```

Se observa de forma mas visual el rellena miento de datos con las técnicas antes mencionadas como knn-imputer

```
#comprobamos de forma mas visual los datos faltantes
sns.heatmap(df1.isnull(), cbar=False)
```

```
<Axes: >
```



Se observa que las variables Sexo y Región también contienen datos faltantes pero donde son tan pocos se eliminan sin afectar mucho al dataframe ademas que se verifica por ultima si quedan datos nulos y con ello terminar de limpiar los datos nulos

```
[27] #considerando que los datos faltantes son 53 de region y 1 de sexo los eliminamos
     df1.dropna(subset=['Sexo', 'Region'], inplace=True)
```

```
#verificamos denuevo los datos faltantes y vemos que estan limpios
df1.isna().sum()
```

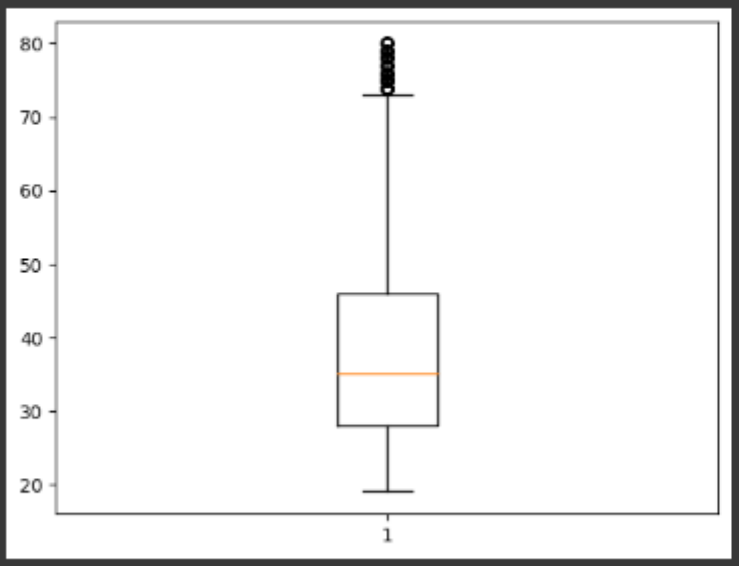
	0
Renta	0
Edad	0
Sexo	0
Antiguedad	0
Consumo	0
Debito	0
Adicional	0
Ctacte	0
TC	0
Cuentas	0
Hipotecario	0
Region	0

dtype: int64

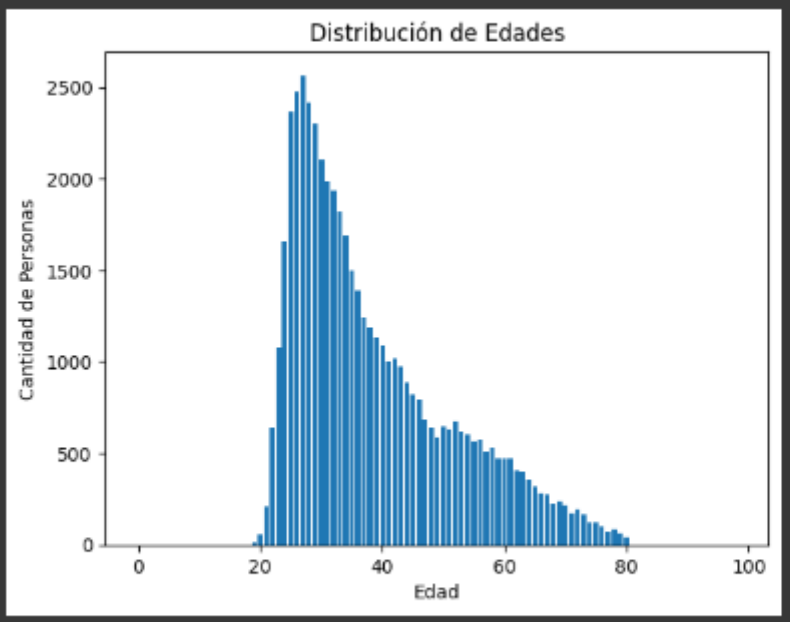
En esta imagen se puede apreciar como se manejan los valor atípicos y eliminación de estos mismos

Manejo de los valores atipicos

```
[65] plt.boxplot(list(df1['Edad']))
     plt.show()
```

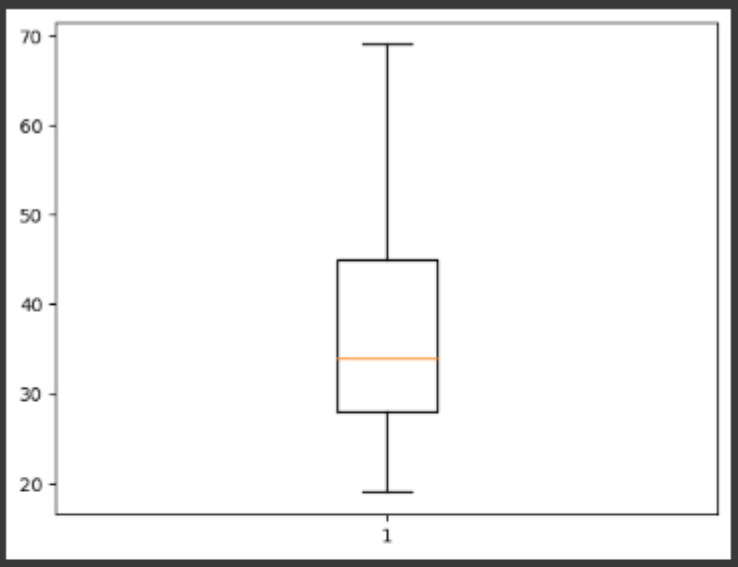



```
# distriución de la variable edad

plt.hist(df1['Edad'], bins=range(0, 100), align='left', rwidth=0.8)
plt.xlabel('Edad')
plt.ylabel('Cantidad de Personas')
plt.title('Distribución de Edades')
plt.show()
```

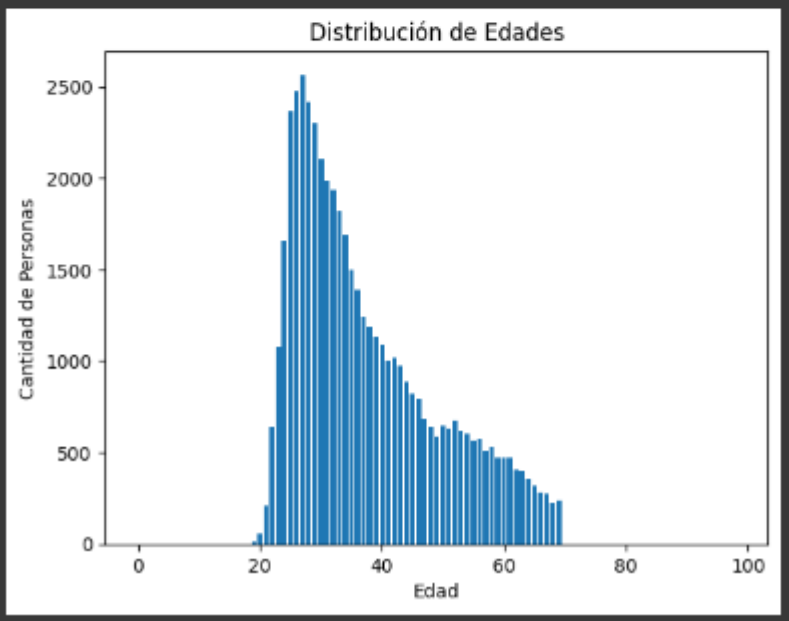



```
[67] # filtramos edaddes mayores a 80 años
     df1 = df1.query('Edad < 70')
```

```
# boxplot despues de quiar ouliers

plt.boxplot(list(df1['Edad']))
plt.show()
```




```
[69] # distribución después de quitar outliers

plt.hist(df1['Edad'], bins=range(0, 100), align='left', rwidth=0.8)
plt.xlabel('Edad')
plt.ylabel('Cantidad de Personas')
plt.title('Distribución de Edades')
plt.show()
```



Análisis de datos

Fase 3 – preparación de datos

Los datos fueron analizados utilizando las siguientes técnicas:

* Visualización de datos
* Análisis de regresión
* Análisis de correlación

Se puede apreciar el análisis de la variable Edad ademas de un histograma mostrando que la mayor cantidad de personas que son parte del banco son de entre 28 y 30 años.

Edad

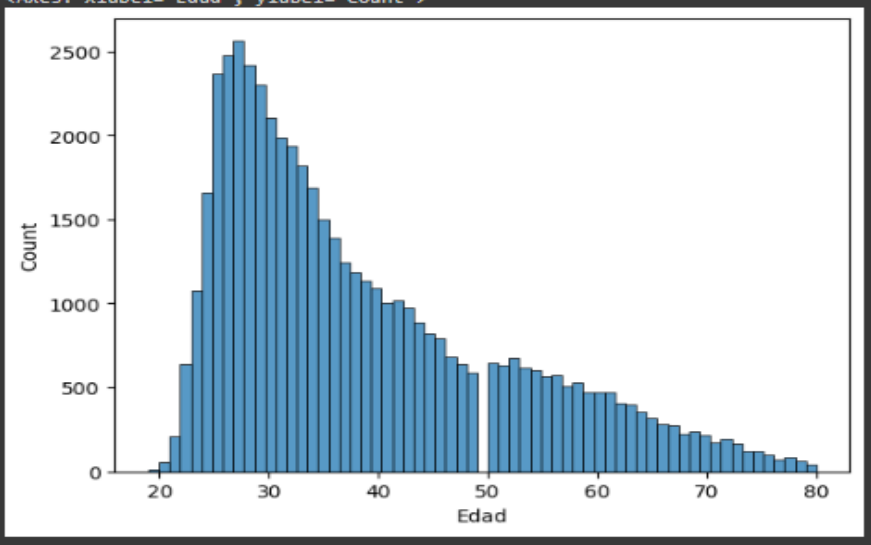
```
df1.Edad.describe()
```

	Edad
count	50892.000000
mean	38.553112
std	13.053192
min	19.000000
25%	28.000000
50%	35.000000
75%	46.000000
max	80.000000

dtype: float64

```
[32] # Histograma
sns.histplot(df1.Edad)
```

```
<Axes: xlabel='Edad', ylabel='Count'>
```



Se puede apreciar el análisis de la variable Sexo ademas de un histograma mostrando que hay mas hombre que mujeres en el banco monopoly

Sexo

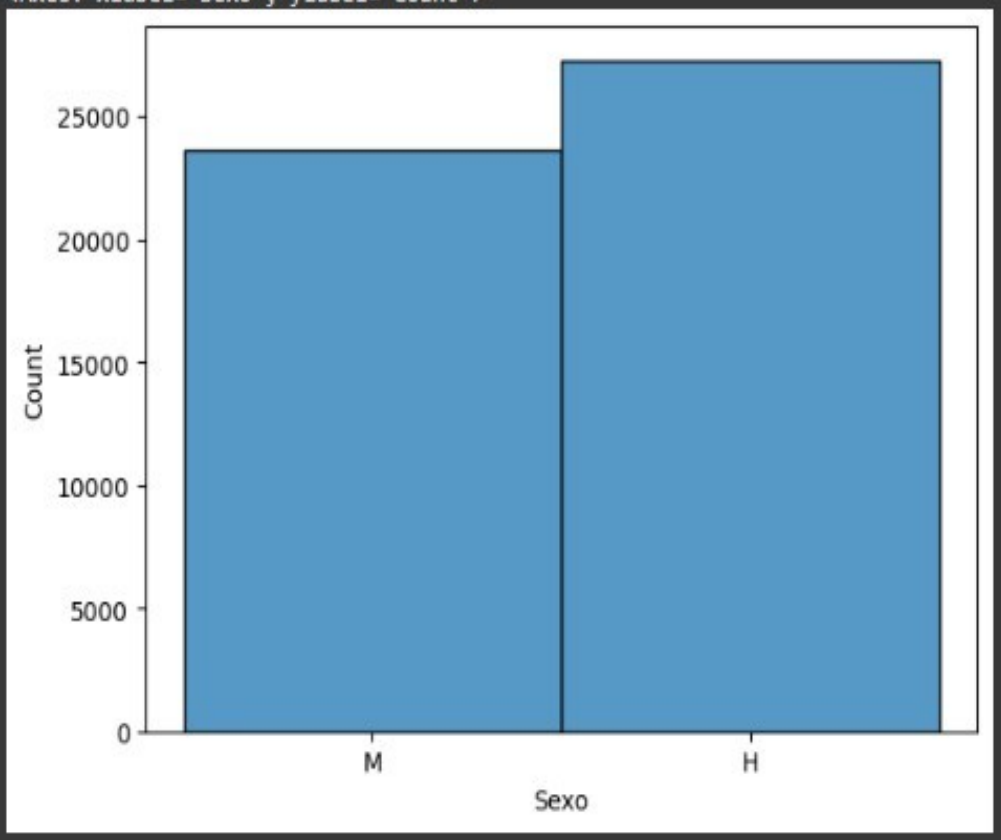
```
df1.Sexo.describe()
```

	Sexo
count	50892
unique	2
top	H
freq	27274

dtype: object

```
[34] # Histograma
sns.histplot(df1.Sexo)
```

<Axes: xlabel='Sexo', ylabel='Count'>



Se puede apreciar el análisis de la variable ctante ademas de un histograma mostrando que es 1 es que poseen una cuenta corriente y 0 que no poseen una cuenta corriente

Ctacte

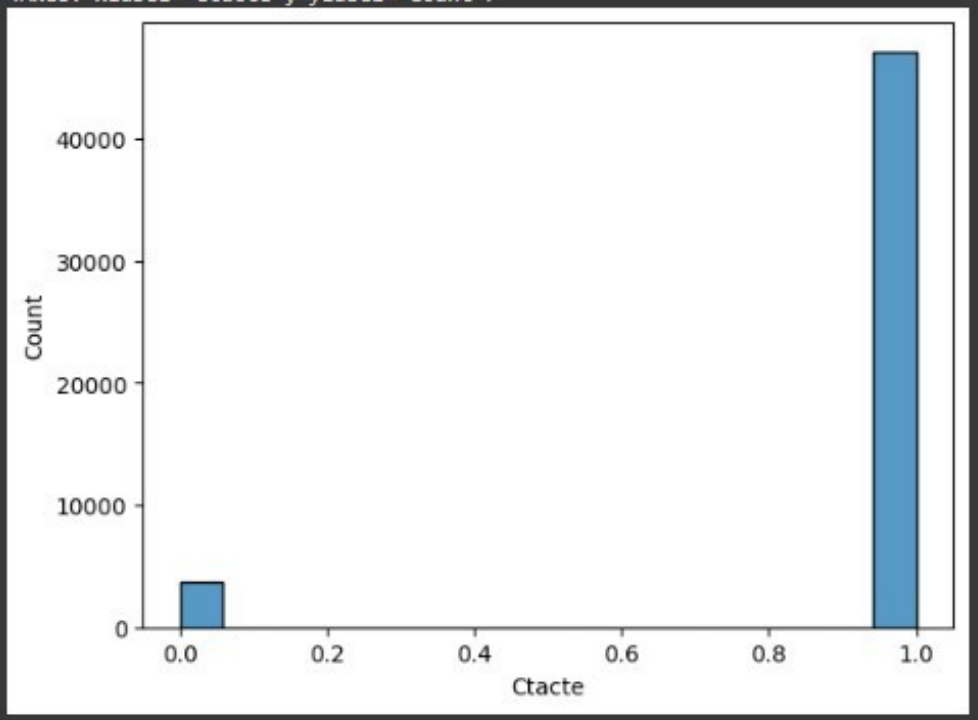
```
df1.Cuentas.describe()
```

	Cuentas
count	50892.000000
mean	1.408473
std	0.550788
min	1.000000
25%	1.000000
50%	1.000000
75%	2.000000
max	5.000000

dtype: float64

```
[42] # histograma
sns.histplot(df1.Ctacte)
```

```
<Axes: xlabel='Ctacte', ylabel='Count'>
```



transformación de la variable sexo a numerica a través de
la tecnica de label encoder para que 1 se hombre y 0 mujer

Transformacion de variables

```
[70] # transformamos la variable sexo a numérica


     label_encoder = preprocessing.LabelEncoder()

     df1['Sexo']= label_encoder.fit_transform(df1['Sexo'])

     df1['Sexo'].unique()
```

```
array([1, 0])
```

```
# Estandarización de datos
scaler = StandardScaler()
df1_standarscaler = scaler.fit_transform(df1)
df1_standarscaler
```

```
array([[-1.15943464e-03,  4.55525024e-01,  1.07305854e+00, ...,
        -7.48165632e-01, -4.05172032e-01,  6.39036242e-01],
       [-1.47065129e+00,  7.08652381e-01, -9.31915609e-01, ...,
        -7.48165632e-01,  2.46808743e+00,  6.39036242e-01],
       [ 7.50697627e-01,  6.24276595e-01, -9.31915609e-01, ...,
        -7.48165632e-01,  2.46808743e+00,  6.39036242e-01],
       ...,
       [ 4.03168029e-01,  7.93028167e-01, -9.31915609e-01, ...,
        -7.48165632e-01, -4.05172032e-01,  6.39036242e-01],
       [-1.15943464e-03,  7.93028167e-01,  1.07305854e+00, ...,
        -7.48165632e-01, -4.05172032e-01,  6.39036242e-01],
       [ 5.01183106e-01,  1.13053131e+00, -9.31915609e-01, ...,
        -7.48165632e-01, -4.05172032e-01,  6.39036242e-01]])
```

```
[72] #Normalizacion de datos
     scaler = Normalizer()
     df1_normalize = scaler.fit_transform(df1)
     df1_normalize
```

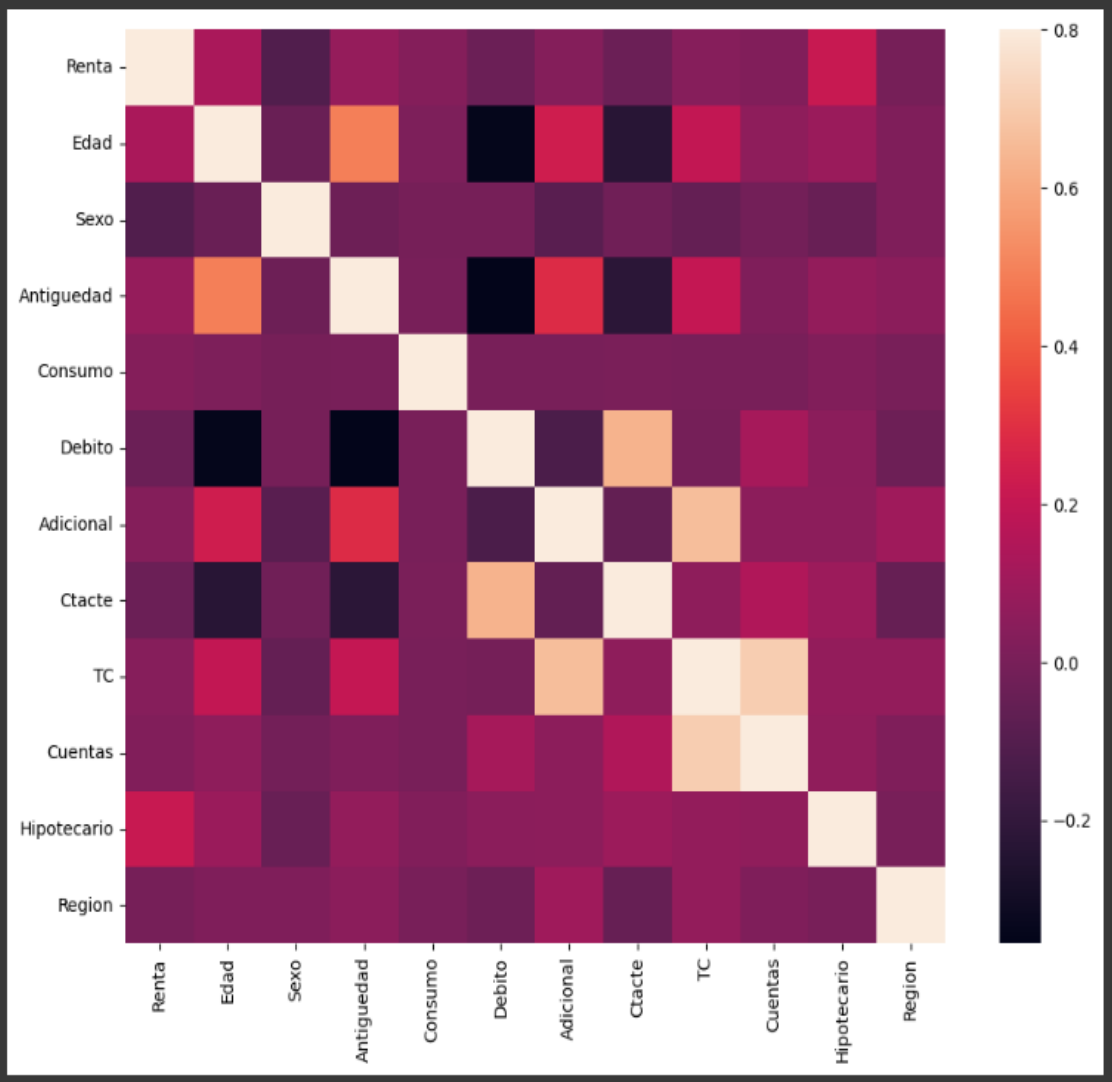
```
array([[9.99999978e-01, 6.48322296e-05, 1.50772627e-06, ...,
        1.50772627e-06, 0.00000000e+00, 1.96004415e-05],
       [9.99999829e-01, 3.20245002e-04, 0.00000000e+00, ...,
        6.96184788e-06, 6.96184788e-06, 9.05040224e-05],
       [9.99999998e-01, 4.84336555e-05, 0.00000000e+00, ...,
        1.07630346e-06, 1.07630346e-06, 1.39919449e-05],
       ...,
       [9.99999980e-01, 5.82967417e-05, 0.00000000e+00, ...,
        1.24035621e-06, 0.00000000e+00, 1.61246307e-05],
       [9.99999997e-01, 7.08631360e-05, 1.50772630e-06, ...,
        1.50772630e-06, 0.00000000e+00, 1.96004419e-05],
       [9.99999994e-01, 6.06508907e-05, 0.00000000e+00, ...,
        1.18923315e-06, 0.00000000e+00, 1.54600310e-05]])
```

```
[74] scaler = MinMaxScaler()
     df1_minmaxscaler =  scaler.fit_transform(df1)
     df1_minmaxscaler
```

```
array([[0.05067101, 0.48      , 1.        , ..., 0.        , 0.        ,
        1.        ],
       [0.01097375, 0.54      , 0.        , ..., 0.        , 1.        ,
        1.        ],
       [0.07098188, 0.52      , 0.        , ..., 0.        , 1.        ,
        1.        ],
       ...,
       [0.06159362, 0.56      , 0.        , ..., 0.        , 0.        ,
        1.        ],
       [0.05067101, 0.56      , 1.        , ..., 0.        , 0.        ,
        1.        ],
```

Se puede apreciar una matriz de correlación el cual ve que tanto tiene relación las variables entre ellas, entre valores positivos y valores negativos.

```
# Matriz de correlación:

corrmat = df1.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```



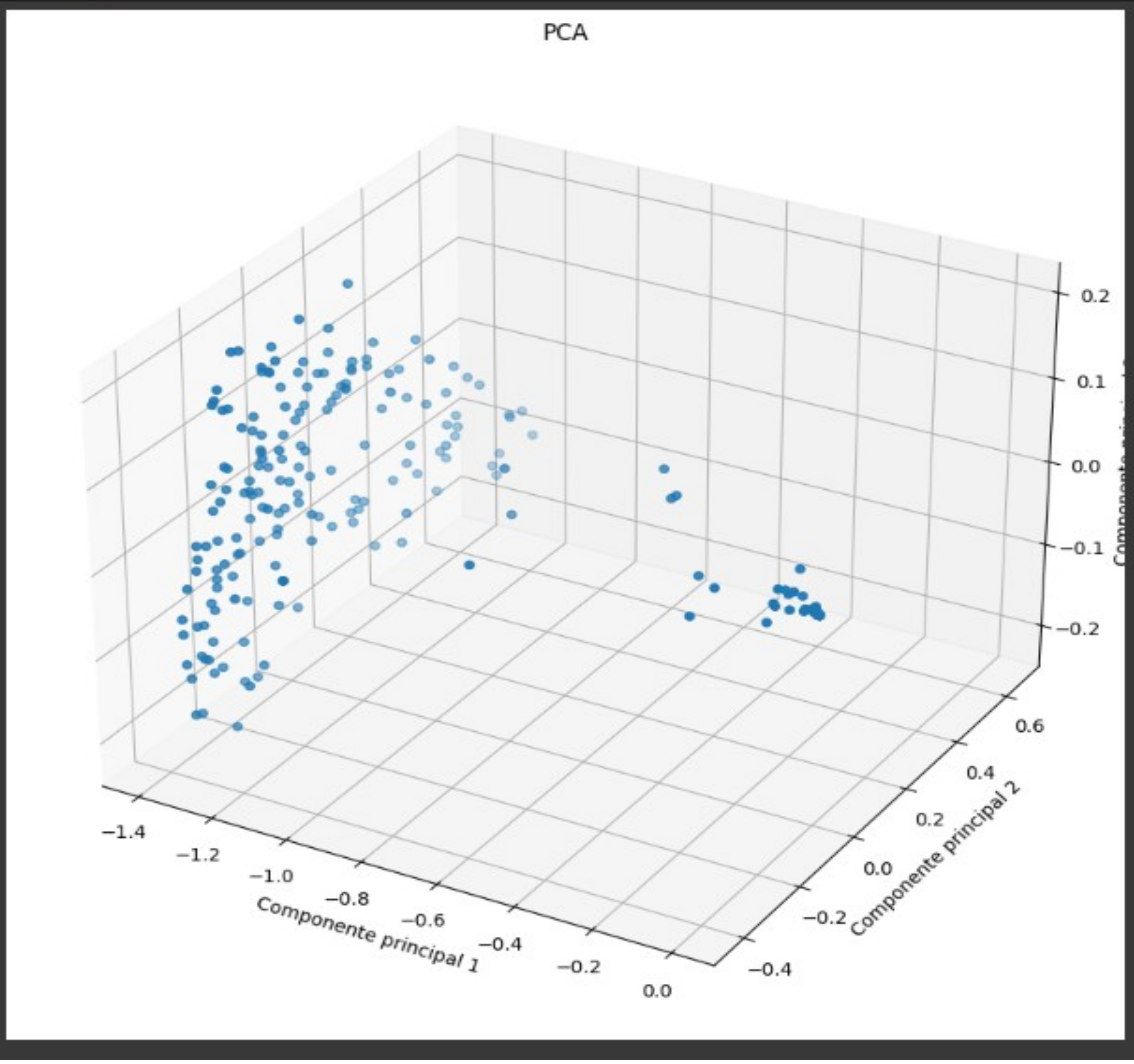
Se aplica el algoritmo de pca para reducir los datos y poder trabajar con ellos correctamente preparándose para la siguiente fase que es el modelado

PCA

Uso de PCA para reducir dimensionalidad para mejorar la estructura de datos y los resultados a la hora del entrenamiento del modelo

```
[67] #se selecciona el número de componentes principales como en este caso tenemos 12 variables vamos a elegir 3 componentes principales para evitar el sobreajuste
     pca = PCA(n_components=3)

[68] df1_pca = pca.fit_transform(df1_normalize)
```

```
fig = plt.figure(figsize=(10, 12))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(df1_pca[:, 0], df1_pca[:, 1], df1_pca[:, 2])
ax.set_xlabel('Componente principal 1')
ax.set_ylabel('Componente principal 2')
ax.set_zlabel('Componente principal 3')
ax.set_title('PCA')

plt.show()
```




```
[73] print(pca.components_)
```

```
[[ 7.04787790e-01 -5.61923173e-01 -7.84105438e-03 -3.97636313e-01
   9.01383651e-10 -1.37593098e-02 -2.72619006e-03 -1.44469576e-02
  -2.57812285e-02 -2.22462095e-02 -1.13945700e-03 -1.66655539e-01]
 [-3.96741493e-02 -5.84792493e-01 -2.78423955e-03  8.02824046e-01
  -7.48694100e-11 -1.54589366e-02  7.45506718e-04 -1.41536344e-02
  -2.11904684e-02 -2.14221009e-02  8.95512149e-04 -1.02766221e-01]
 [ 1.25204525e-01 -1.56480020e-01  3.01259163e-02  2.25624700e-02
  -2.13828920e-10  5.48849414e-02  1.58047609e-02  5.03197160e-02
   9.61826579e-02  6.54150924e-02  4.61553337e-03  9.69059045e-01]]
```

Se utiliza kmeans como visualización y uso de los datos del pca para su separación de los datos y agrupación

K-means

```
[81] # Utiliza los datos del PCA como entrada para k-means
     kmeans = KMeans(n_clusters=3)
     kmeans.fit(df1_pca)
```

```
KMeans
KMeans(n_clusters=3)
```

```
[82] # Obtén los centroides de los clusters y las etiquetas de los datos
     centroides = kmeans.cluster_centers_
     etiquetas = kmeans.labels_
```

```
[ ] # Muestra los datos de k-means por pantalla
    print("Centroides de los clusters:")
    print(centroides)
```

```
Centroides de los clusters:
[[ 4.62648402e-03 -2.32640999e-06  3.76503668e-06]
 [-1.34542773e+00 -1.76109058e-01  2.31687949e-03]
 [-1.31961438e+00  3.03724593e-01 -6.92268573e-03]]
```

```
[84] print("\nEtiquetas de los datos:")
     print(etiquetas)
```

```
Etiquetas de los datos:
[0 0 0 ... 0 0 0]
```

```
# Muestra un gráfico de dispersión de los datos de k-means
plt.scatter(df1_pca[:, 0], df1_pca[:, 1], c=etiquetas)
plt.scatter(centroides[:, 0], centroides[:, 1], c='red', marker='*', s=200)
plt.title("K-means")
plt.show()
```