In [1]:
```python
import pandas as pd
import scipy
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [2]:
```python
#display first 10 rows
df = pd.read_csv('Churn_Modelling.csv')
df.head(10)
```

Out[2]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	1	0	149756.71	1
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	1	1	10062.80	0
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	1	0	119346.88	1
8	9	15792365	He	501	France	Male	44	4	142051.07	2	0	1	74940.50	0
9	10	15592389	H?	684	France	Male	27	2	134603.88	1	1	1	71725.73	0

In [3]:
```python
num_rows, num_columns = df.shape

print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

```
Number of Rows: 10000
Number of Columns: 14
```

In [4]:
```python
df.columns
```

Out[4]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [5]:
```python
#removes duplicate values
df.drop_duplicates
```

Out[5]:
```
<bound method DataFrame.drop_duplicates of       RowNumber  CustomerId    Surname  CreditScore Geography  Gender  Age  \
0             1    15634602   Hargrave          619    France  Female   42
1             2    15647311       Hill          608     Spain  Female   41
2             3    15619304       Onio          502    France  Female   42
3             4    15701354       Boni          699    France  Female   39
4             5    15737888   Mitchell          850     Spain  Female   43
...         ...         ...        ...          ...       ...     ...  ...
9995       9996    15606229   Obijiaku          771    France    Male   39
9996       9997    15569892  Johnstone          516    France    Male   35
9997       9998    15584532        Liu          709    France  Female   36
9998       9999    15682355  Sabbatini          772   Germany    Male   42
9999      10000    15628319     Walker          792    France  Female   28

      Tenure    Balance  NumOfProducts  HasCrCard  IsActiveMember  \
0          2       0.00              1          1               1
1          1   83807.86              1          0               1
2          8  159660.80              3          1               0
3          1       0.00              2          0               0
4          2  125510.82              1          1               1
...      ...        ...            ...        ...             ...
9995       5       0.00              2          1               0
9996      10   57369.61              1          1               1
9997       7       0.00              1          0               1
9998       3   75075.31              2          1               0
9999       4  130142.79              1          1               0

      EstimatedSalary  Exited
0           101348.88       1
1           112542.58       0
2           113931.57       1
3            93826.63       0
4            79084.10       0
...               ...     ...
9995         96270.64       0
9996        101699.77       0
9997         42085.58       1
9998         92888.52       1
9999         38190.78       0

[10000 rows x 14 columns]>
```

In [6]:
```python
df
```

Out[6]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

In [7]:
```python
#checks for missing value
df.isna()
```

Out[7]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	False	False	False	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False	False	False	False	False	False	False

10000 rows × 14 columns

In [8]:
```python
#fills the missing value with 0
df.fillna(0)
```

Out[8]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 14 columns

In [9]:
```python
#removes unnecesssary characters from surname
df["Surname"].str.strip("123.,?/_")
```

Out[9]:
```
0        Hargrave
1            Hill
2            Onio
3            Boni
4        Mitchell
           ...
9995     Obijiaku
9996    Johnstone
9997          Liu
9998    Sabbatini
9999       Walker
Name: Surname, Length: 10000, dtype: object
```

In [10]:
```python
#replaces '1s' and '0s' to 'yes' and 'no'
df['HasCrCard']=df['HasCrCard'].replace({0: 'no', 1: 'yes'})
df['IsActiveMember']=df['IsActiveMember'].replace({0: 'no', 1: 'yes'})
df['Exited']=df['Exited'].replace({0: 'no', 1: 'yes'})
```

In [11]:
```python
df
```

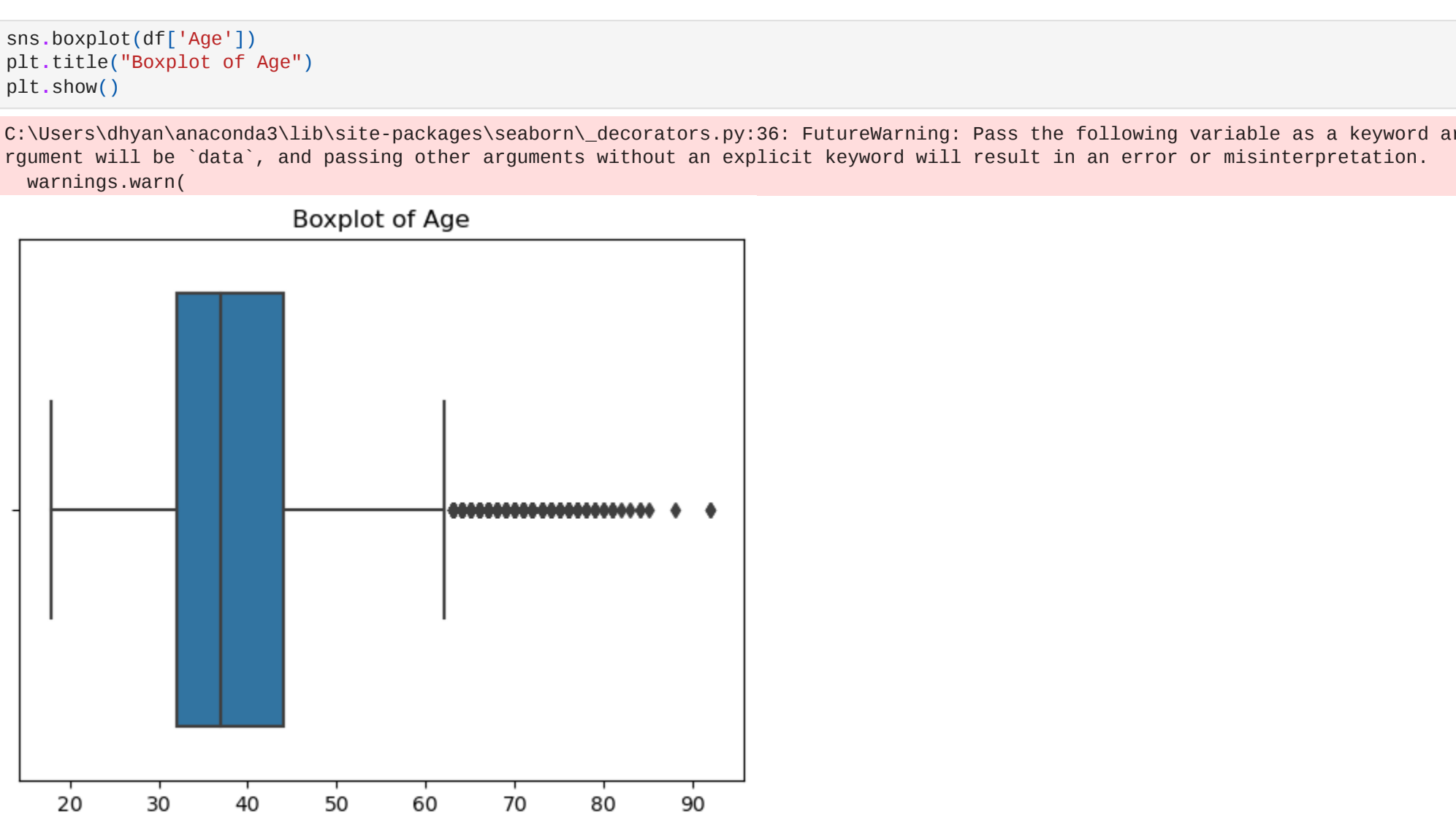
Out[11]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	yes	yes	101348.88	yes
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	no	yes	112542.58	no
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	yes	no	113931.57	yes
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	no	no	93826.63	no
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	yes	yes	79084.10	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.00	2	yes	no	96270.64	no
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.61	1	yes	yes	101699.77	no
9997	9998	15584532	Liu	709	France	Female	36	7	0.00	1	no	yes	42085.58	yes
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.31	2	yes	no	92888.52	yes
9999	10000	15628319	Walker	792	France	Female	28	4	130142.79	1	yes	no	38190.78	no

10000 rows × 14 columns

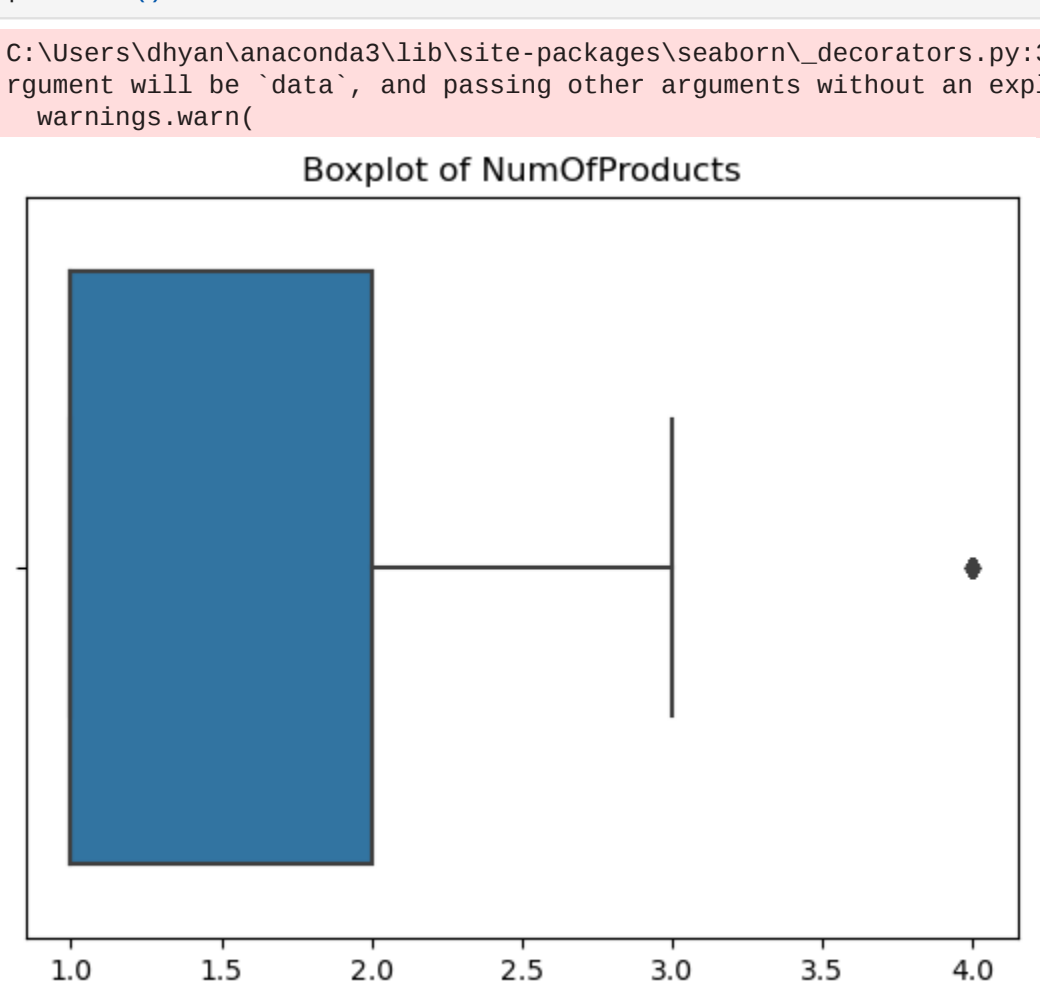
In [12]:
```python
#histogram
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
ax1.hist(df['Age'], bins=10, edgecolor='red')
ax2.hist(df['HasCrCard'], bins=5, edgecolor='green')
```

Out[12]:
```
(array([7055.,    0.,    0.,    0., 2945.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. ]),
 <BarContainer object of 5 artists>)
```



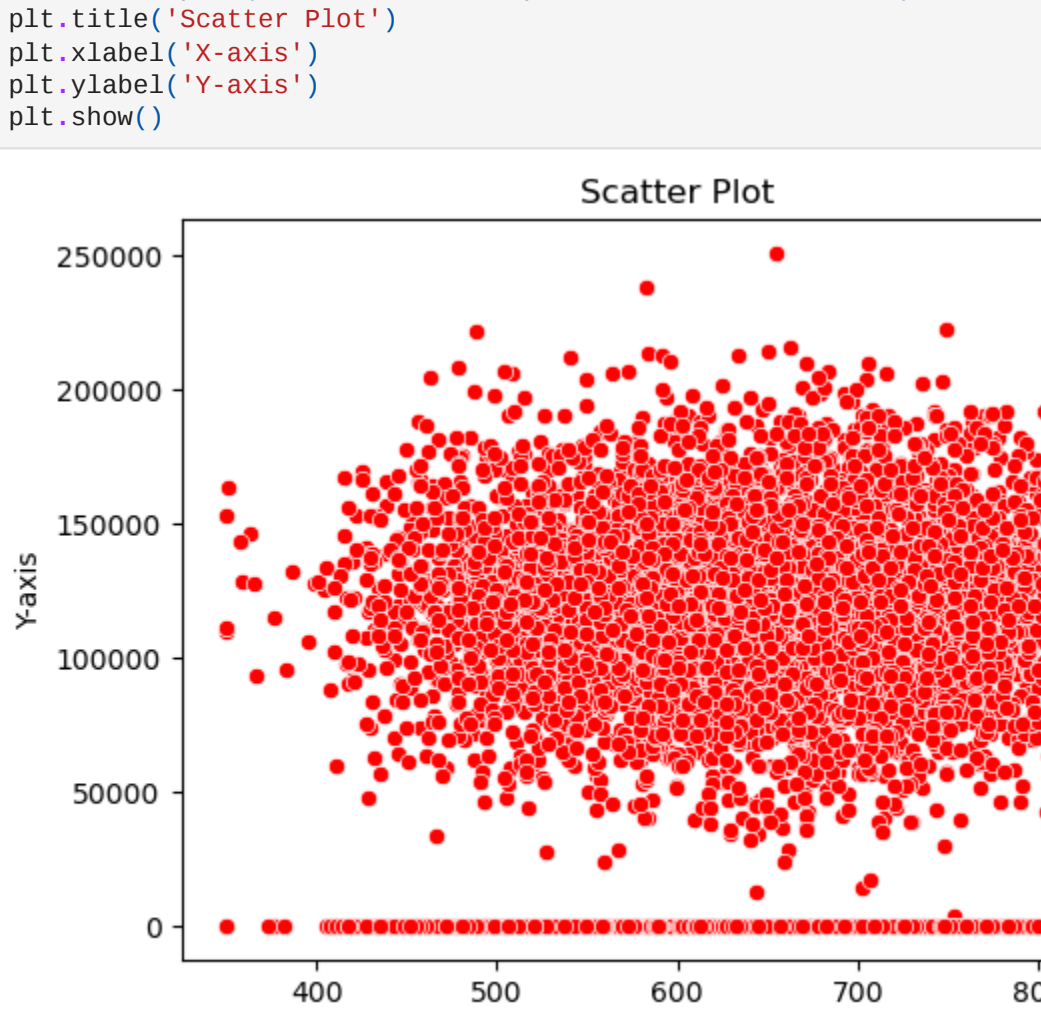
In [13]:
```python
sns.boxplot(df['Age'])
plt.title("Boxplot of Age")
plt.show()
```

```
C:\Users\dhyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

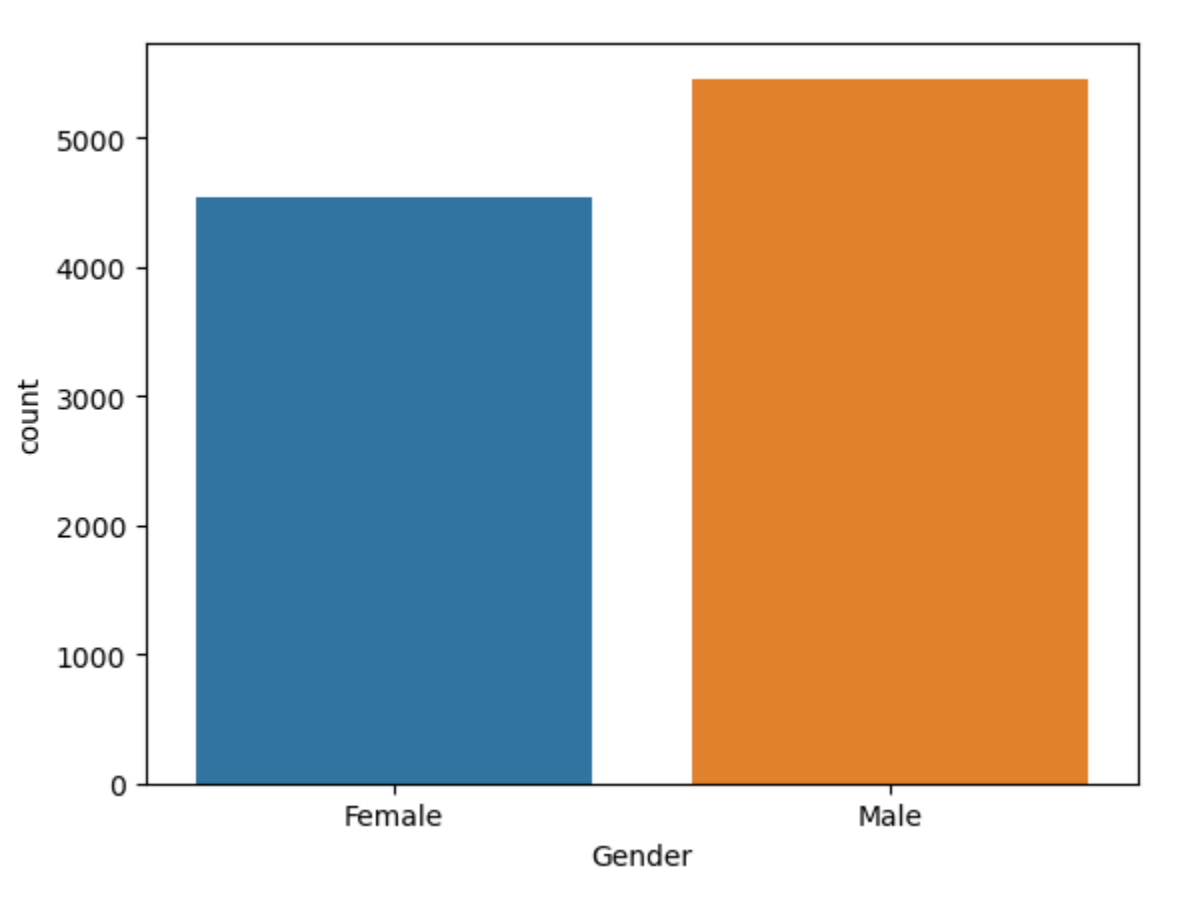


In [14]:
```python
sns.boxplot(df['NumOfProducts'])
plt.title("Boxplot of NumOfProducts")
plt.show()
```

```
C:\Users\dhyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

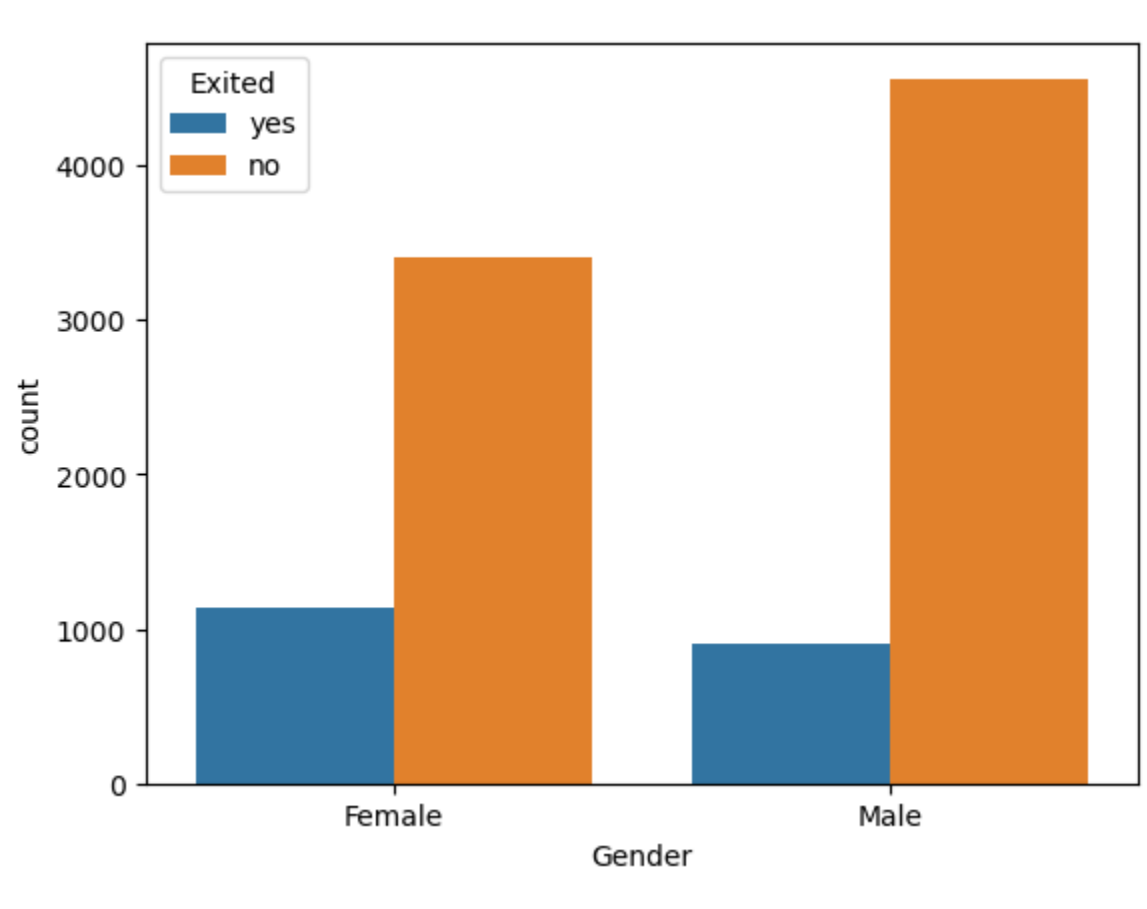


In [15]:
```python
#scatterplot for creditscore and balance
sns.scatterplot(x=df.CreditScore,y=df.Balance,color="red")
plt.title('Scatter Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



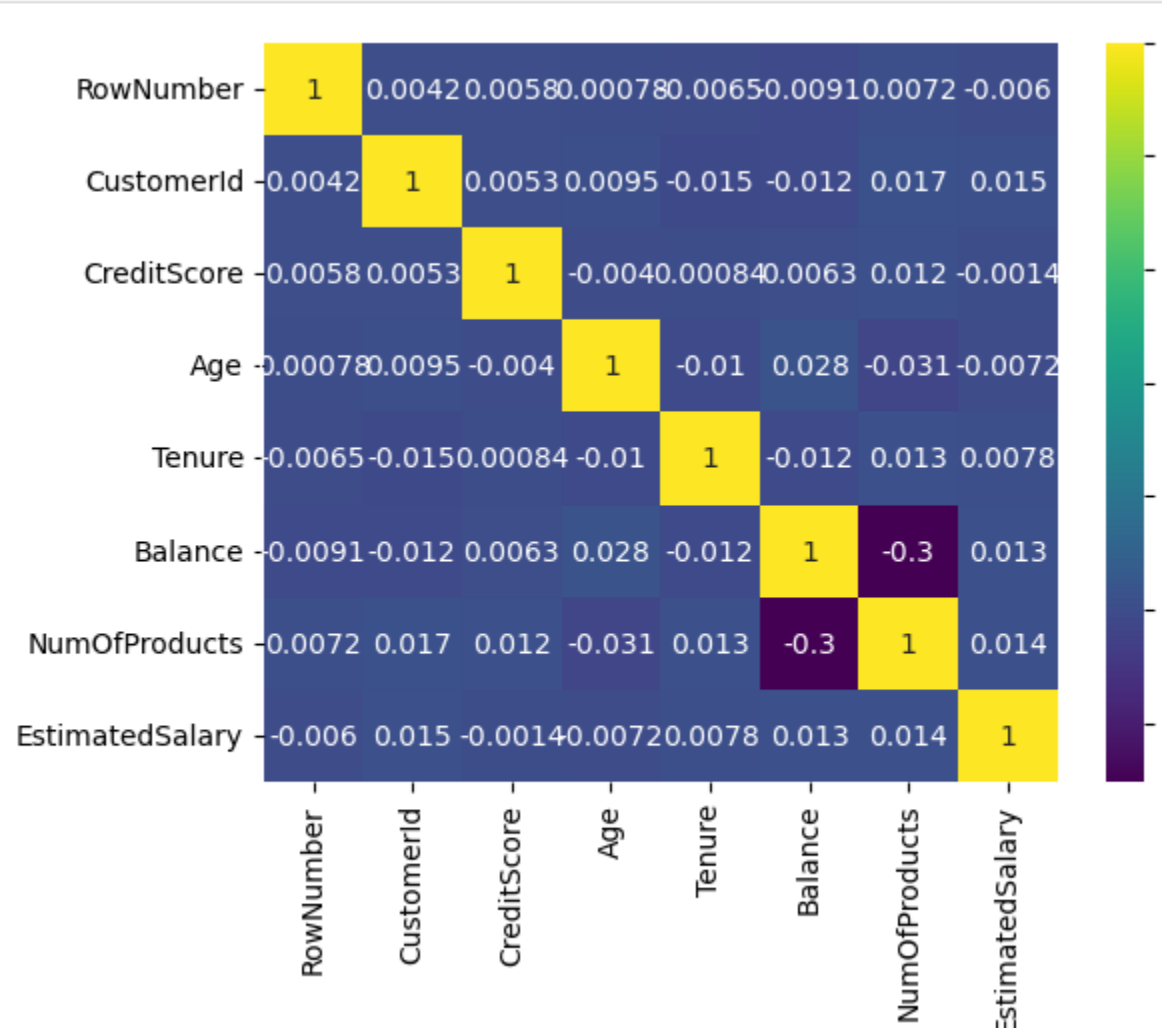
In [16]:
```python
sns.countplot(x='Gender',data=df)
```

Out[16]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```

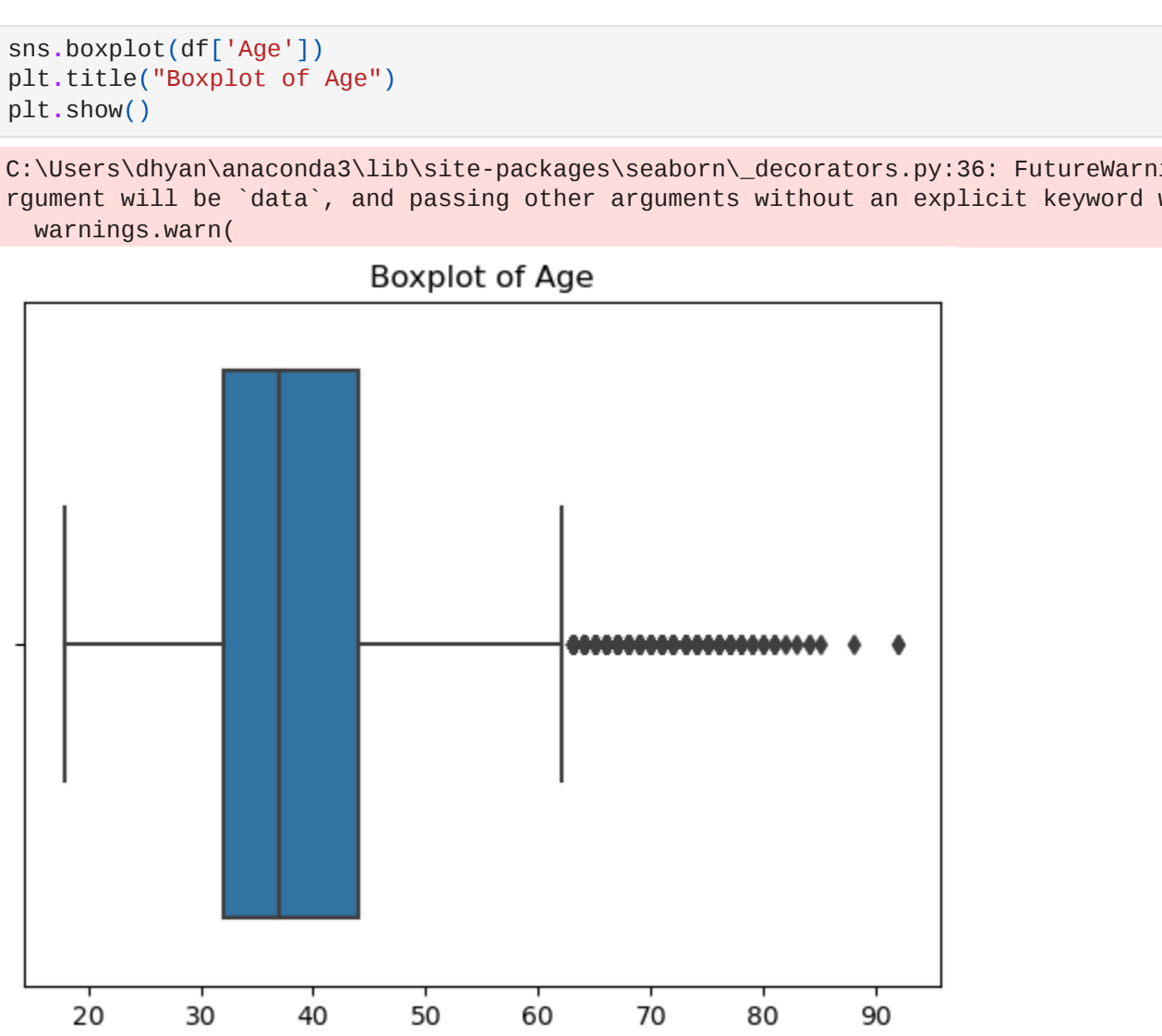


In [17]:
```python
sns.countplot(x='Gender',hue='Exited',data=df)
```

Out[17]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```

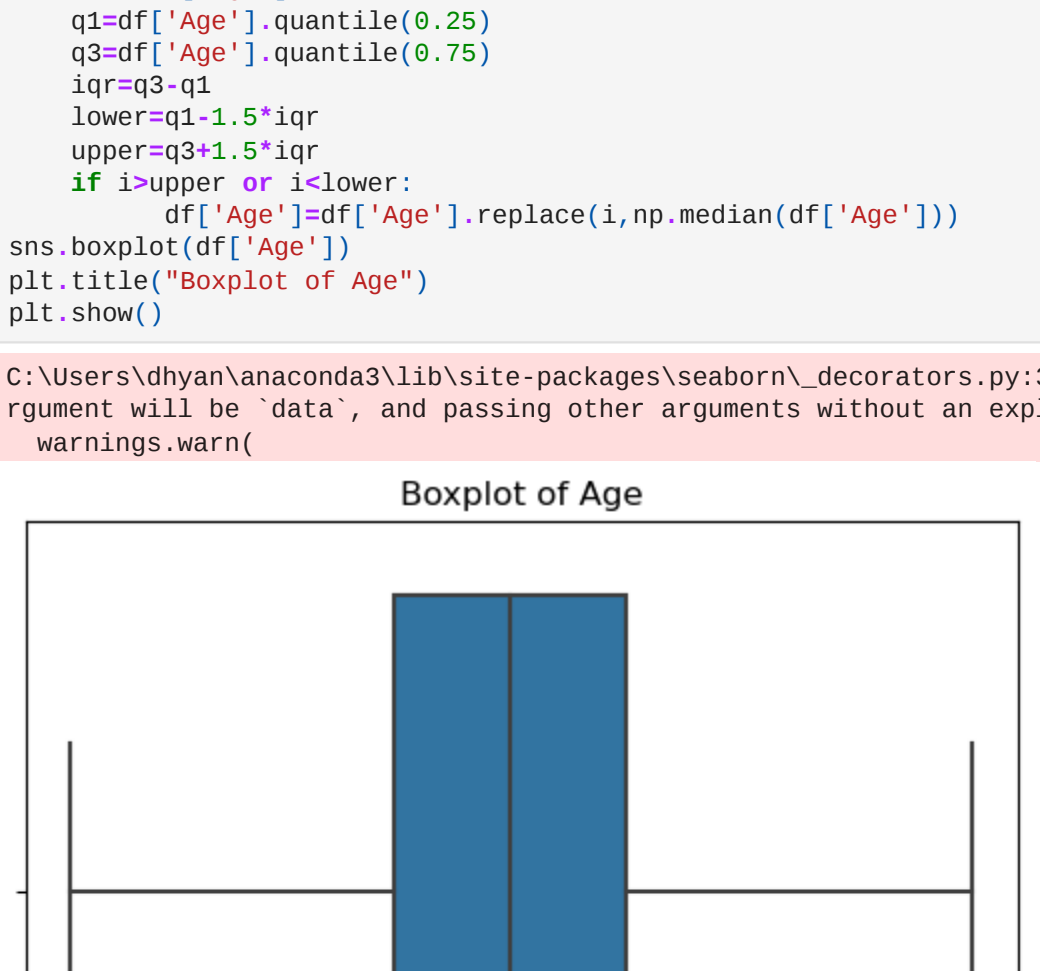


In [18]:
```python
#correlation between the variables with heatmap
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='viridis')
plt.show()
```



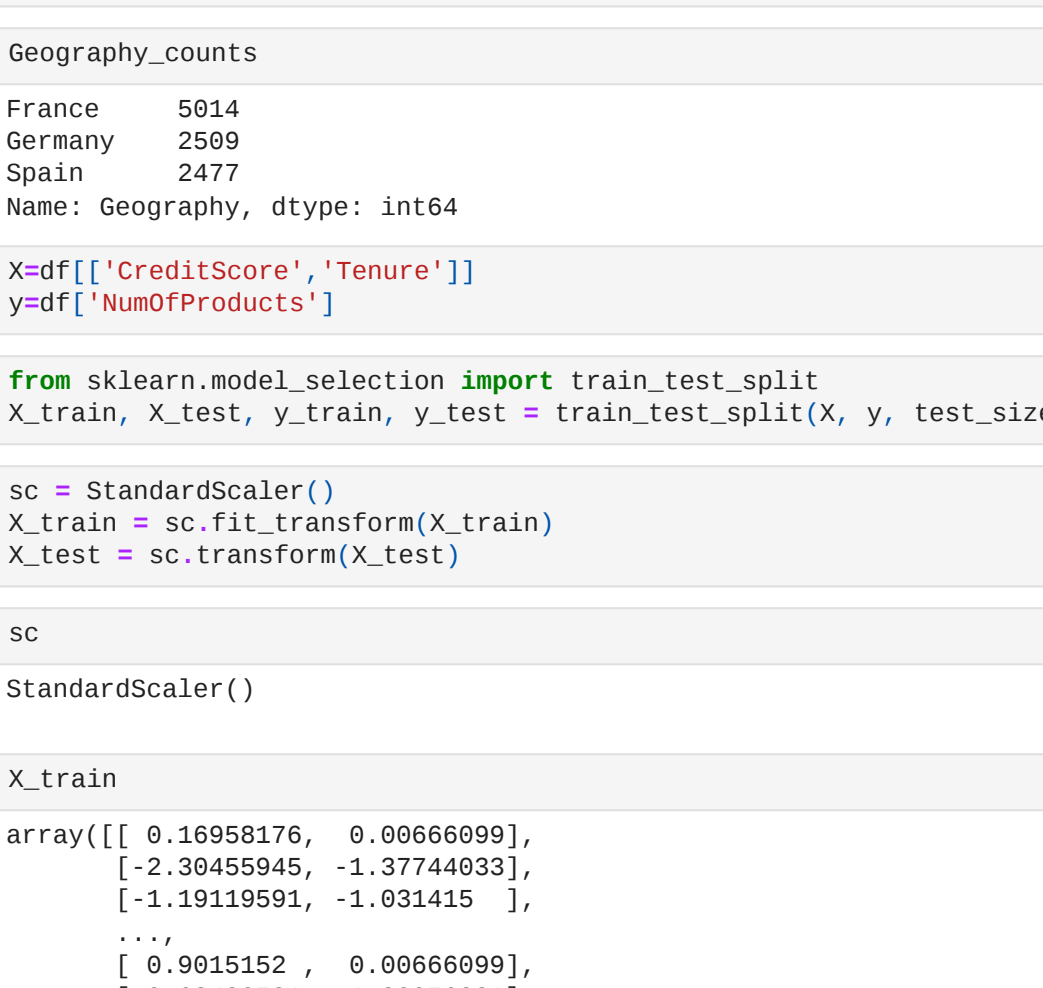
In [19]:
```python
sns.boxplot(df['Age'])
plt.title("Boxplot of Age")
plt.show()
```

```
C:\Users\dhyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [20]:
```python
#outliers of age
for i in df['Age']:
    q1=df['Age'].quantile(0.25)
    q3=df['Age'].quantile(0.75)
    iqr=q3-q1
    lower=q1-1.5*iqr
    upper=q3+1.5*iqr
    if i>upper or i<lower:
          df['Age']=df['Age'].replace(i,np.median(df['Age']))
sns.boxplot(df['Age'])
plt.title("Boxplot of Age")
plt.show()
```

```
C:\Users\dhyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [21]:
```python
Geography_counts = df['Geography'].value_counts()
```

In [22]:
```python
Geography_counts
```

Out[22]:
```
France     5014
Germany    2509
Spain      2477
Name: Geography, dtype: int64
```

In [33]:
```python
X=df[['CreditScore','Tenure']]
y=df['NumOfProducts']
```

In [34]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

In [35]:
```python
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [42]:
```python
sc
```

Out[42]:
```
StandardScaler()
```

In [36]:
```python
X_train
```

Out[36]:
```
array([[ 0.16958176,  0.00666099],
       [-2.30455945, -1.37744033],
       [-1.19119591, -1.031415  ],
       ...,
       [ 0.9015152 ,  0.00666099],
       [-0.62420521,  1.39076231],
       [-0.28401079, -1.37744033]])
```

In [37]:
```python
X_test
```

Out[37]:
```
array([[-0.55204276,  1.04473698],
       [-1.31490297, -1.031415  ],
       [ 0.57162971,  1.04473698],
       ...,
       [-0.74791227, -1.37744033],
       [-0.00566991, -0.33936434],
       [-0.79945688,  1.04473698]])
```

In [38]:
```python
pca = PCA(n_components = 2)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
```

In [39]:
```python
X_train
```

Out[39]:
```
array([[-0.11520238, -0.12462244],
       [ 0.65557222,  2.60356702],
       [ 0.11298216,  1.57162325],
       ...,
       [-0.63275749, -0.64217754],
       [ 1.4247972 , -0.54203772],
       [-0.77317144,  1.17482336]])
```

In [40]:
```python
X_test
```

Out[40]:
```
array([[ 1.12909378, -0.34838742],
       [ 0.20045626,  1.65909735],
       [ 0.33453736, -1.14294385],
       ...,
       [-0.44514356,  1.50285124],
       [-0.2359576 ,  0.24397606],
       [ 1.30404198, -0.17343922]])
```

In [41]:
```python
pca
```

Out[41]:
```
PCA(n_components=2)
```

In [43]:
```python
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

Out[43]:
```
LogisticRegression(random_state=0)
```

In [44]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[976  69   0   0]
 [841  45   0   0]
 [ 48   2   0   0]
 [ 18   1   0   0]]
```

Out[44]:
```
0.5105
```

In [ ]: