Assignment No. 3

In [3]:
```
#Importing the Libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [4]:
```
df = pd.read_csv("Churn_Modelling.csv")
```

In [5]:
```
#Preprocesssing
df.isnull()
```

Out[5]:

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

10000 rows × 14 columns

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [8]:
```
df.dtypes
```

Out[8]:
```
RowNumber          int64
CustomerId         int64
Surname           object
CreditScore        int64
Geography         object
Gender            object
Age                int64
Tenure             int64
Balance          float64
NumOfProducts      int64
HasCrCard          int64
IsActiveMember     int64
EstimatedSalary  float64
Exited             int64
dtype: object
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [10]:
```
#Dropping the unnecessary columns
df= df.drop(['RowNumber', 'CustomerId', 'Surname'], axis =1)
```

In [11]:
```
df.head()
```

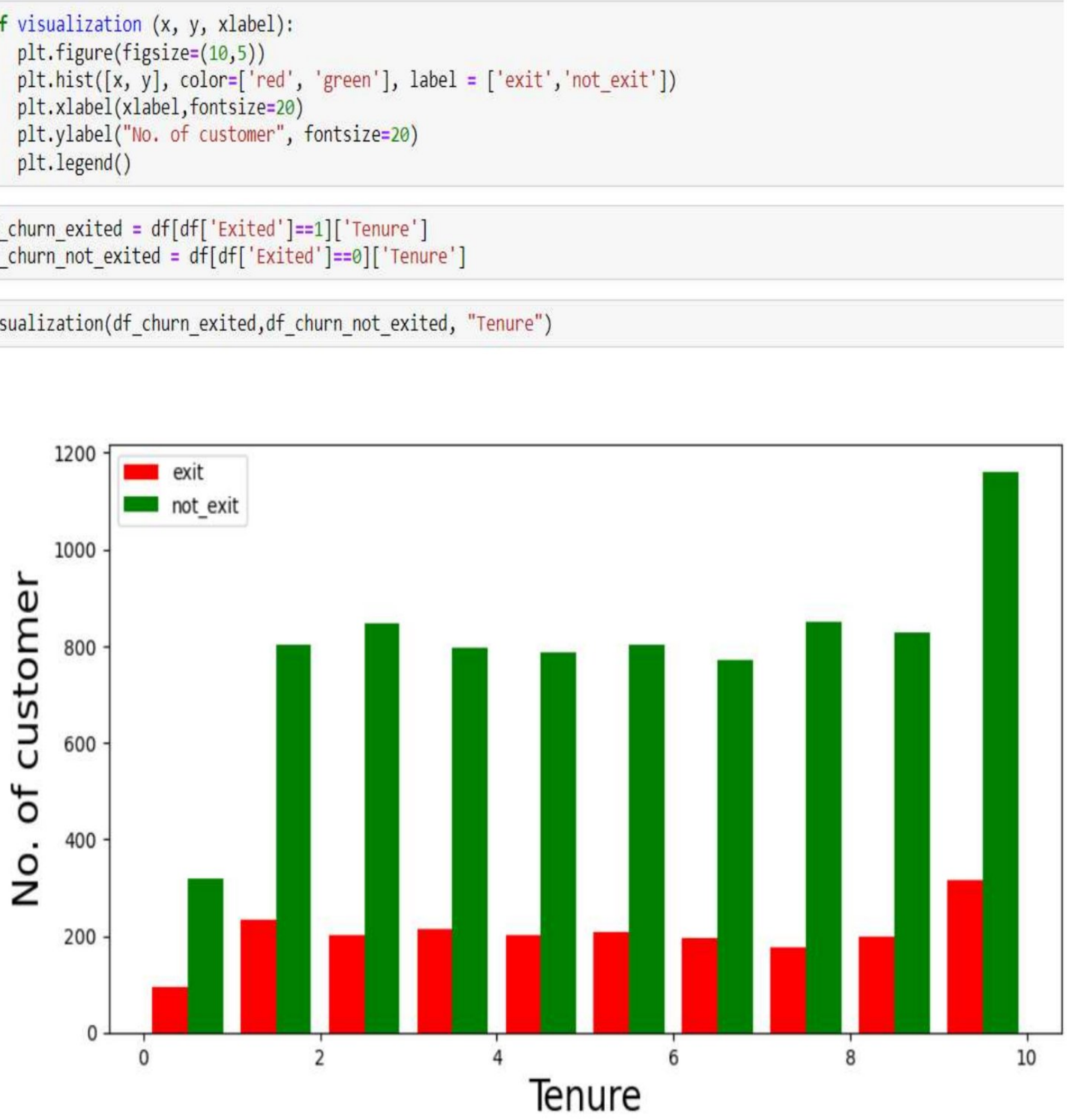
Out[11]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

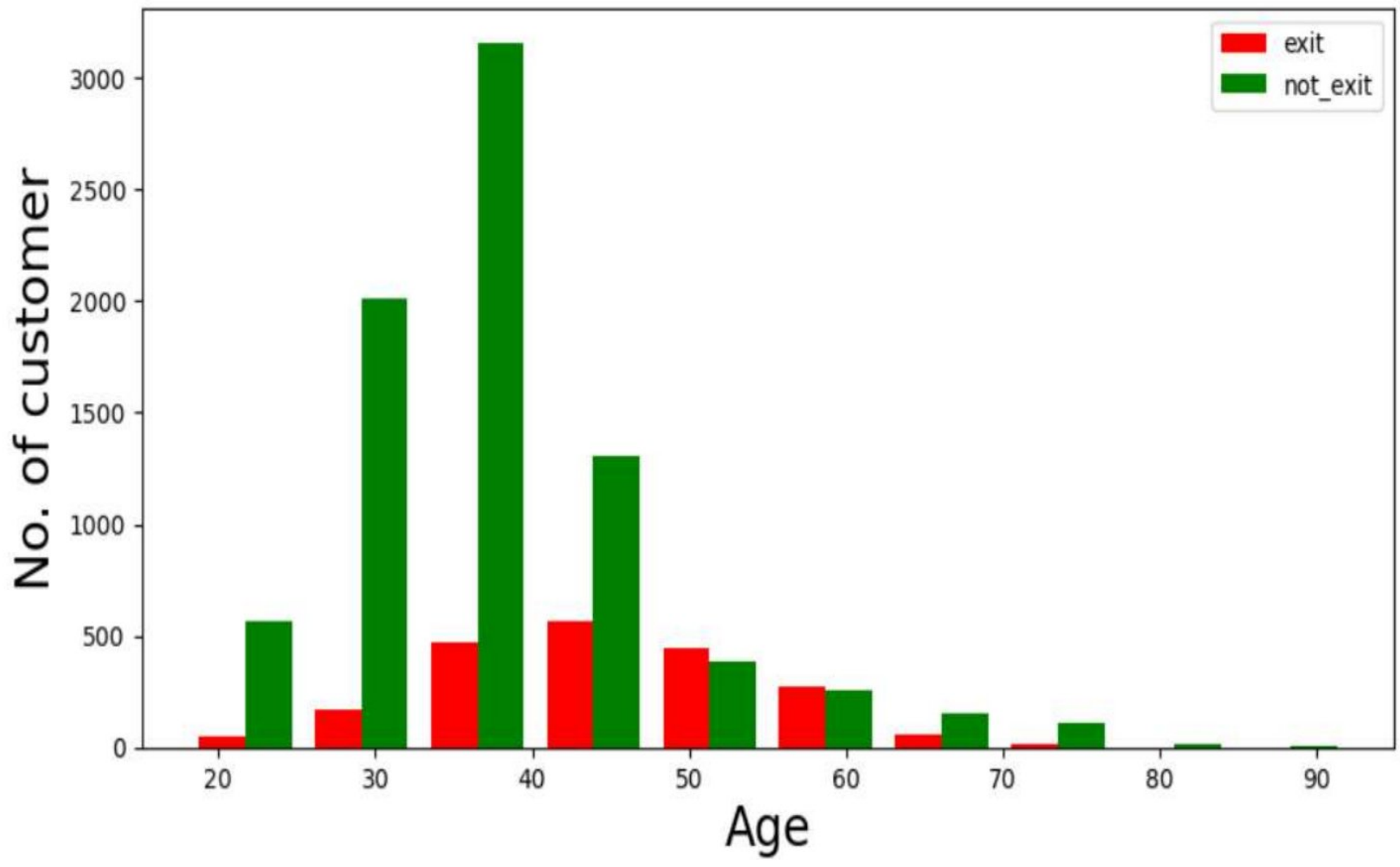
In [12]:
```
def visualization (x, y, xlabel):
    plt.figure(figsize=(10,5))
    plt.hist([x, y], color=['red', 'green'], label = ['exit','not_exit'])
    plt.xlabel(xlabel,fontsize=20)
    plt.ylabel("No. of customer", fontsize=20)
    plt.legend()
```

In [13]:
```
df_churn_exited = df[df['Exited']==1]['Tenure']
df_churn_not_exited = df[df['Exited']==0]['Tenure']
```

In [14]:
```
visualization(df_churn_exited,df_churn_not_exited, "Tenure")
```



In [15]:
```
df_churn_exited2 = df[df['Exited']==1]['Age']
df_churn_not_exited2 = df[df['Exited']==0]['Age']
visualization(df_churn_exited2,df_churn_not_exited2, "Age")
```



In [16]:
```
X =df[['CreditScore','Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited']]
states = pd.get_dummies(df['Geography'],drop_first =True)
gender = pd.get_dummies(df['Gender'],drop_first = True)
```

In [17]:
```
df =pd.concat([df,gender,states],axis=1)
```

In [18]:
```
#Splitting the training and testing Dataset
df.head()
```

Out[18]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Male	Germany	Spain
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1	0	0	0
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	0	0	1
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	0	0	0
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0	0	0	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	0	0	1

In [19]:
```
X= df[['CreditScore','Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary','Male','Germany','Spain']]
y=df['Exited']
```

In [20]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30)
```

In [21]:
```
#Normalizing the values with mean as 0 and Standard Deviation as 1
from sklearn.preprocessing import StandardScaler
sc =StandardScaler()
X_train =sc.fit_transform(X_train)
X_test =sc.transform(X_test)
```

In [22]:
```
X_train
```

Out[22]:
```
array([[ 0.38628108,  0.48664962,  1.04235774, ...,  0.90244757,
         1.72155741, -0.57339125],
       [ 0.80193718,  0.67749795, -0.68454375, ...,  0.90244757,
         1.72155741, -0.57339125],
       [ 0.81232858, -0.18131955, -0.68454375, ...,  0.90244757,
        -0.58086939, -0.57339125],
       ...,
       [-0.48659671,  1.15461879,  1.04235774, ...,  0.90244757,
        -0.58086939, -0.57339125],
       [-0.67364195,  0.48664962,  0.69697744, ...,  0.90244757,
         1.72155741, -0.57339125],
       [ 1.84107741, -0.37216788, -1.72068464, ..., -1.10809762,
        -0.58086939,  1.74400987]])
```

In [23]:
```
X_test
```

Out[23]:
```
array([[-0.33072568,  0.29580128, -1.02992404, ..., -1.10809762,
        -0.58086939, -0.57339125],
       [-0.39307409,  0.29580128,  1.73311833, ..., -1.10809762,
        -0.58086939, -0.57339125],
       [ 1.30072449,  1.25004295,  1.04235774, ...,  0.90244757,
        -0.58086939, -0.57339125],
       ...,
       [ 0.27197566, -0.75386455, -0.68454375, ...,  0.90244757,
        -0.58086939, -0.57339125],
       [ 0.83311138,  0.00952878,  0.35159714, ..., -1.10809762,
        -0.58086939,  1.74400987],
       [-0.73599037, -0.18131955,  0.00621685, ...,  0.90244757,
        -0.58086939,  1.74400987]])
```

In [24]:
```
from sklearn.neural_network import MLPClassifier
ann = MLPClassifier(hidden_layer_sizes=(100,100,100),random_state = 0,max_iter =100, activation ='relu')
ann.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:692: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (100) reached and the optimization hasn't converged yet.
  warnings.warn(
```

Out[24]:
```
MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100, random_state=0)
```

In [25]:
```
y_pred =ann.predict(X_test)
```

In [26]:
```
y_pred
```

Out[26]:
```
array([1, 0, 0, ..., 0, 1, 0], dtype=int64)
```

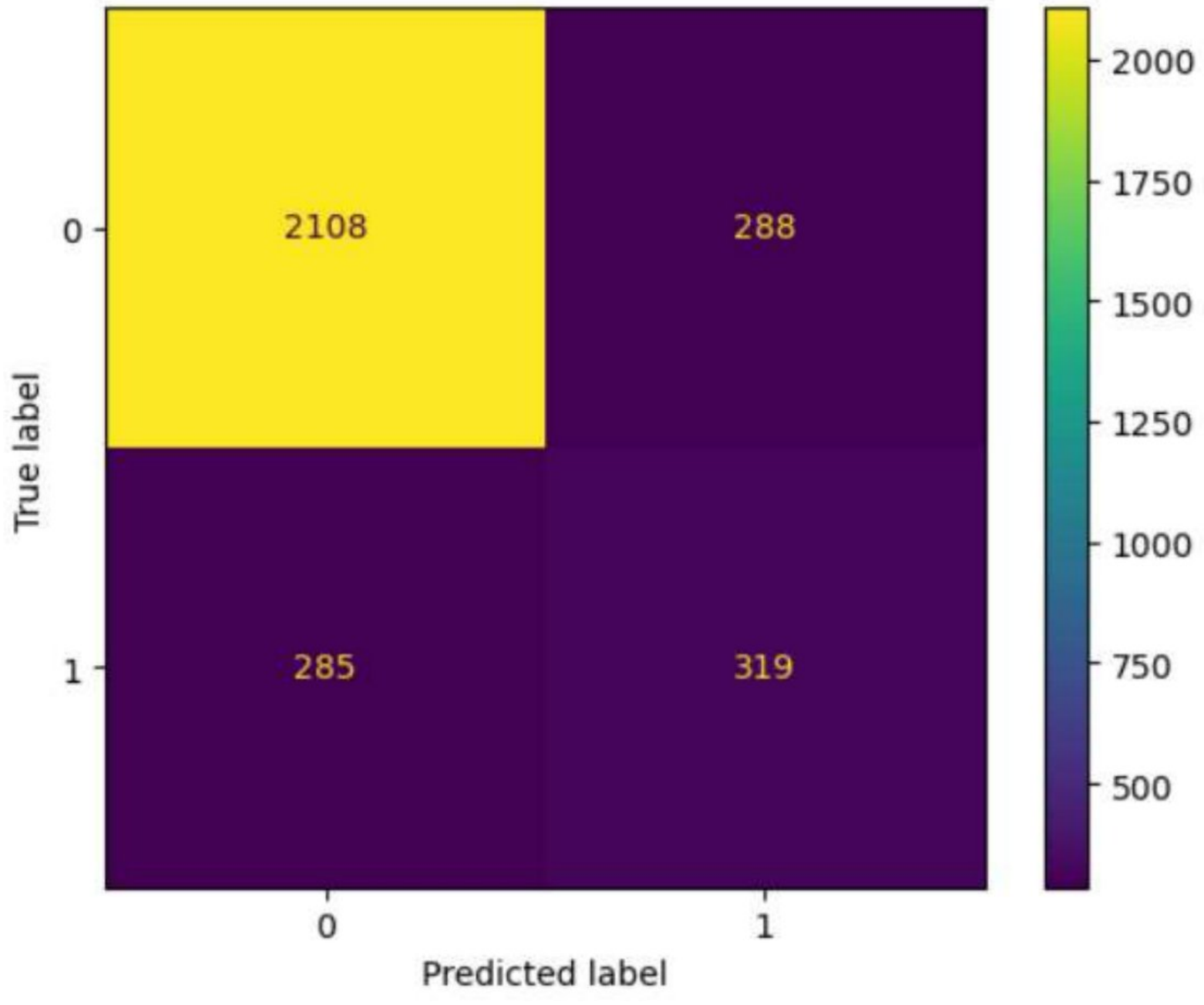
In [30]:
```
from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score, classification_report
```

In [31]:
```
y_test.value_counts()
```

Out[31]:
```
0    2396
1     604
Name: Exited, dtype: int64
```

In [32]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[32]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2a7aa29cc70>
```



In [39]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.809
```

In [40]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.88      0.88      2396
           1       0.53      0.53      0.53       604

    accuracy                           0.81      3000
   macro avg       0.70      0.70      0.70      3000
weighted avg       0.81      0.81      0.81      3000
```