In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv('C:/shubhangi/2023-24/LP-III_ML/Assignment 3/Churn_Modelling.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCr(
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

In [4]:
```
df.shape
```

Out[4]:
```
(10000, 14)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [6]:
```
#input data
x=df[['CreditScore','Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary']]
#output data
y=df['Exited']
```

In [7]:
```
x
```

Out[7]:

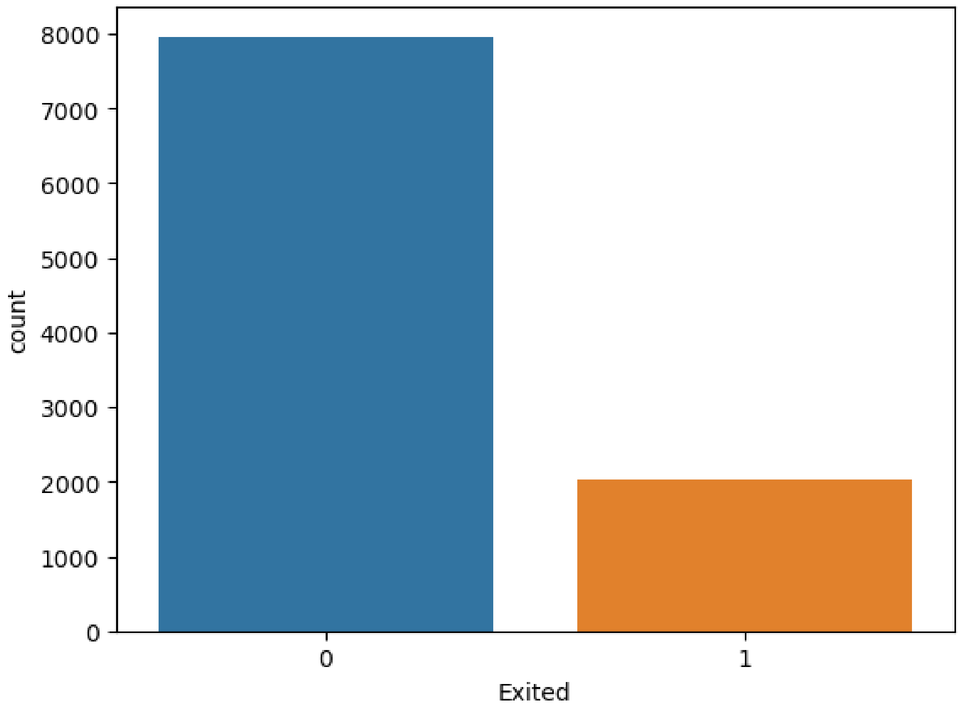
	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	42	2	0.00	1	1	1	101348.88
1	608	41	1	83807.86	1	0	1	112542.58
2	502	42	8	159660.80	3	1	0	113931.57
3	699	39	1	0.00	2	0	0	93826.63
4	850	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...
9995	771	39	5	0.00	2	1	0	96270.64
9996	516	35	10	57369.61	1	1	1	101699.77
9997	709	36	7	0.00	1	0	1	42085.58
9998	772	42	3	75075.31	2	1	0	92888.52
9999	792	28	4	130142.79	1	1	0	38190.78

10000 rows × 8 columns

In [8]:
```
import seaborn as sns
```

In [9]:
```
sns.countplot(x=y)
```

Out[9]:
```
<Axes: xlabel='Exited', ylabel='count'>
```



In [10]:
```
y.value_counts()
```

Out[10]:
```
0    7963
1    2037
Name: Exited, dtype: int64
```

In [11]:
```
pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\programdata\anaconda3\lib\site-packages (0.
10.1)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from
imbalanced-learn) (1.24.3)
Requirement already satisfied: scipy>=1.3.2 in c:\programdata\anaconda3\lib\site-packages (from i
mbalanced-learn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages
(from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from
imbalanced-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages
(from imbalanced-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [12]:
```
from imblearn.over_sampling import RandomOverSampler
```

In [13]:
```
res=RandomOverSampler(random_state=1)
```

In [14]:
```
x_res,y_res=res.fit_resample(x,y)
```

In [18]:
```
y_res.value_counts()
```

Out[18]:
```
1    7963
0    7963
Name: Exited, dtype: int64
```

```
In [19]: from sklearn.preprocessing import StandardScaler
         scaler=StandardScaler()
```

```
In [20]: x_scaled=scaler.fit_transform(x_res)
```

```
In [21]: x_scaled
```

```
Out[21]: array([[-0.30317594,  0.07969247, -1.0276189 , ...,  0.65477112,
                  1.0874469 ,  0.0032301 ],
                [-0.41466059, -0.01443957, -1.37123012, ..., -1.52725124,
                  1.0874469 ,  0.19715802],
                [-1.48896724,  0.07969247,  1.0340484 , ...,  0.65477112,
                 -0.91958513,  0.22122191],
                ...,
                [ 1.03463989,  0.64448473, -0.34039647, ..., -1.52725124,
                  1.0874469 , -1.23001093],
                [ 0.18330254, -0.20270365, -1.0276189 , ...,  0.65477112,
                 -0.91958513, -1.04712788],
                [ 0.75086077,  0.55035268,  1.72127083, ..., -1.52725124,
                 -0.91958513,  0.03840677]])
```

```
In [22]: # cross validation
         from sklearn.model_selection import train_test_split
```

```
In [23]: x_train,x_test,y_train,y_test=train_test_split(x_scaled,y_res,random_state=0,test_size=0.25)
```

```
In [24]: x_scaled.shape
```

```
Out[24]: (15926, 8)
```

```
In [25]: x_train.shape
```

```
Out[25]: (11944, 8)
```

```
In [26]: y_res.shape
```

```
Out[26]: (15926,)
```

```
In [27]: x_test.shape
```

```
Out[27]: (3982, 8)
```

```
In [28]: from sklearn.neural_network import MLPClassifier
```

```
In [29]: ann=MLPClassifier(hidden_layer_sizes=(100,100,100),random_state=100,max_iter=100,activation='relu'
```

```
In [30]: ann.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and the optimization h
asn't converged yet.
  warnings.warn(
```

```
Out[30]: ▾                  MLPClassifier
         MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100,
                       random_state=100)
```

In [31]:
```
y_pred=ann.predict(x_test)
```

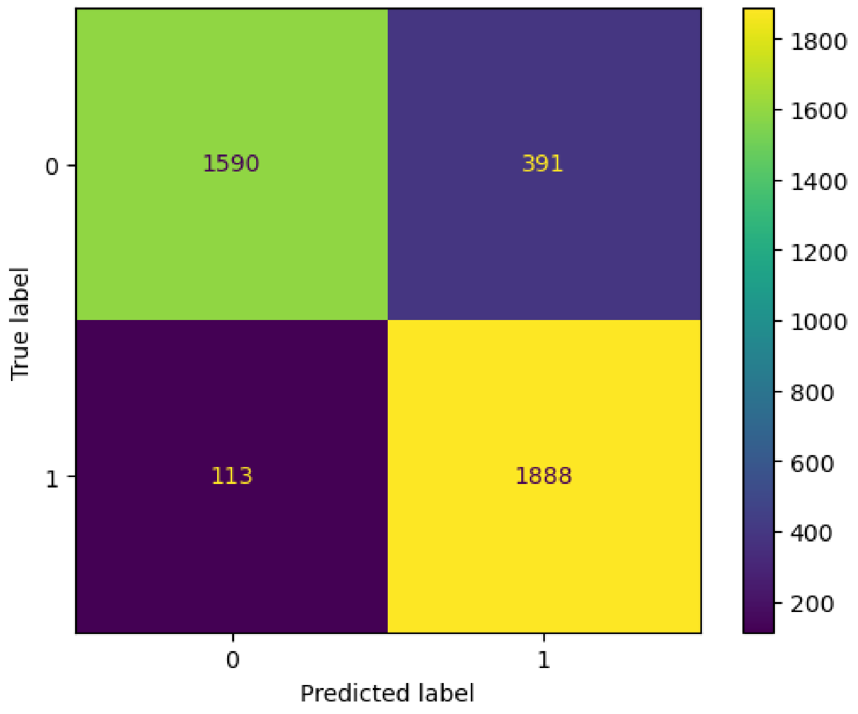
In [32]:
```
from sklearn.metrics import ConfusionMatrixDisplay,classification_report
from sklearn.metrics import accuracy_score
```

In [33]:
```
y_test.value_counts()
```

Out[33]:
```
1    2001
0    1981
Name: Exited, dtype: int64
```

In [34]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[34]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x287eafdd490>
```



In [35]:
```
accuracy_score(y_test,y_pred)
```

Out[35]:
```
0.8734304369663486
```

In [36]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.93      0.80      0.86      1981
           1       0.83      0.94      0.88      2001

    accuracy                           0.87      3982
   macro avg       0.88      0.87      0.87      3982
weighted avg       0.88      0.87      0.87      3982
```

In [37]:
```
print("Ann model Implemented....")
```

```
Ann model Implemented....
```

```
In [ ]:
```

In [39]:
```
import pandas as pd
```

In [40]:
```
df=pd.read_csv('C:/shubhangi/2023-24/LP-III_ML/Assignment 3/Churn_Modelling.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCr
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	

In [4]:
```
df.shape
```

Out[4]:
```
(10000, 14)
```

In [5]:
```
df.columns
```

Out[5]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [6]:
```
#input data
x=df[['CreditScore','Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary']]
#output data
y=df['Exited']
```

In [7]:
```
x
```

Out[7]:

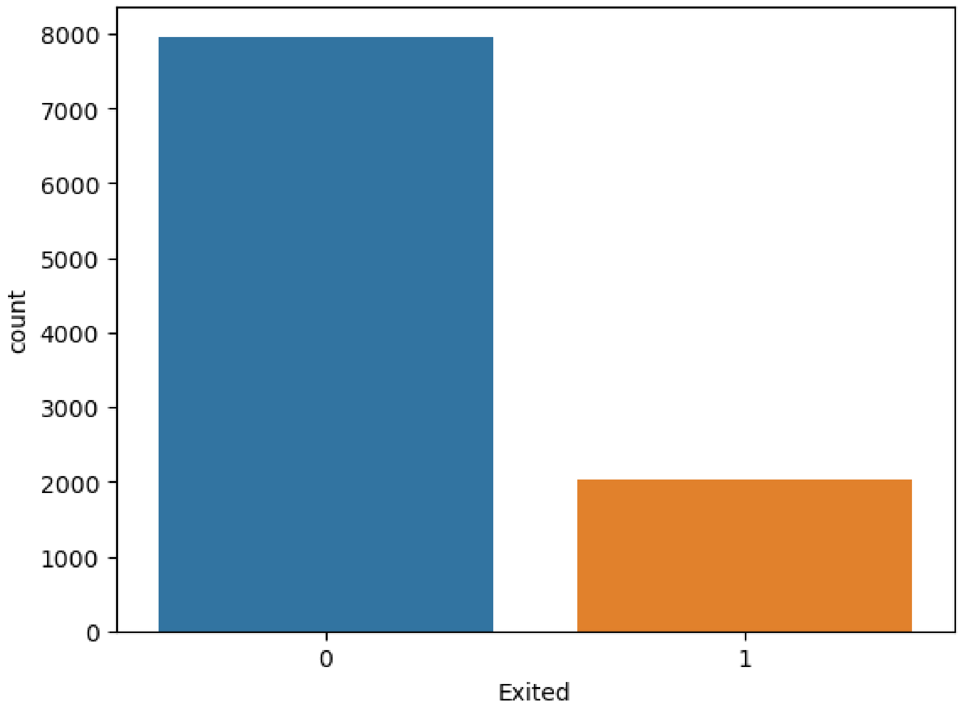
	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	619	42	2	0.00	1	1	1	101348.88
1	608	41	1	83807.86	1	0	1	112542.58
2	502	42	8	159660.80	3	1	0	113931.57
3	699	39	1	0.00	2	0	0	93826.63
4	850	43	2	125510.82	1	1	1	79084.10
...	...	...	...	...	...	...	...	...
9995	771	39	5	0.00	2	1	0	96270.64
9996	516	35	10	57369.61	1	1	1	101699.77
9997	709	36	7	0.00	1	0	1	42085.58
9998	772	42	3	75075.31	2	1	0	92888.52
9999	792	28	4	130142.79	1	1	0	38190.78

10000 rows × 8 columns

In [8]:
```
import seaborn as sns
```

In [9]:
```
sns.countplot(x=y)
```

Out[9]:
```
<Axes: xlabel='Exited', ylabel='count'>
```



In [12]:
```
y.value_counts()
```

Out[12]:
```
0    7963
1    2037
Name: Exited, dtype: int64
```

In [13]:
```
#normalize
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

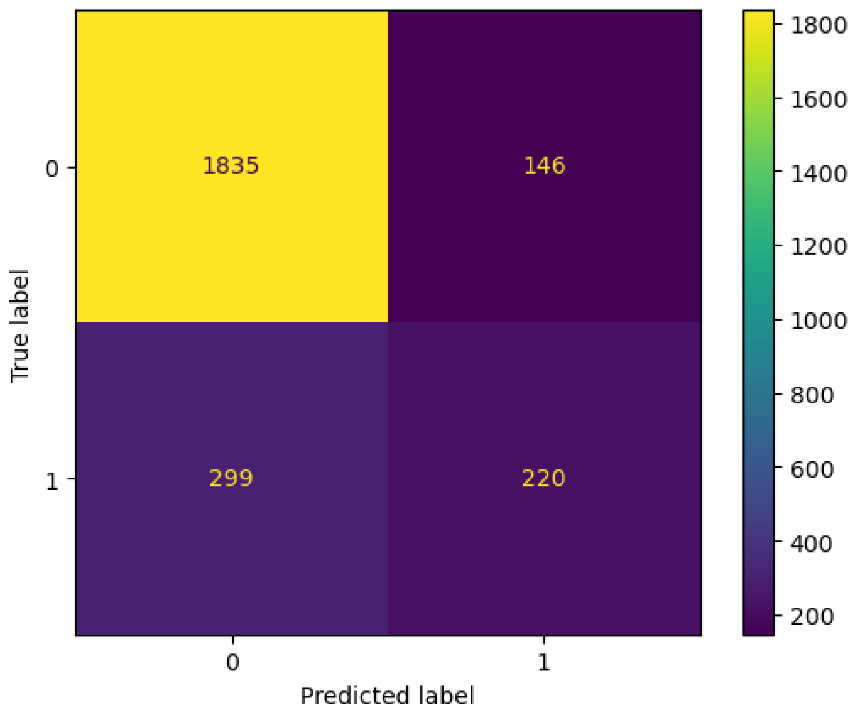
In [18]:
```
x_scaled=scaler.fit_transform(x)
```

In [19]:
```
x_scaled
```

Out[19]:
```
array([[-0.32622142,  0.29351742, -1.04175968, ...,  0.64609167,
         0.97024255,  0.02188649],
       [-0.44003595,  0.19816383, -1.38753759, ..., -1.54776799,
         0.97024255,  0.21653375],
       [-1.53679418,  0.29351742,  1.03290776, ...,  0.64609167,
        -1.03067011,  0.2406869 ],
       ...,
       [ 0.60498839, -0.27860412,  0.68712986, ..., -1.54776799,
         0.97024255, -1.00864308],
       [ 1.25683526,  0.29351742, -0.69598177, ...,  0.64609167,
        -1.03067011, -0.12523071],
       [ 1.46377078, -1.04143285, -0.35020386, ...,  0.64609167,
        -1.03067011, -1.07636976]])
```

```
In [21]: # cross validation
         from sklearn.model_selection import train_test_split

In [22]: x_train,x_test,y_train,y_test=train_test_split(x_scaled,y)

In [23]: x_scaled.shape

Out[23]: (10000, 8)

In [24]: x_train.shape

Out[24]: (7500, 8)

In [25]: x_test.shape

Out[25]: (2500, 8)

In [27]: from sklearn.neural_network import MLPClassifier

In [30]: ann=MLPClassifier(hidden_layer_sizes=(100,100,100),random_state=100,max_iter=100,activation='relu'

In [31]: ann.fit(x_train,y_train)

         C:\ProgramData\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686:
         ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached and the optimization h
         asn't converged yet.
           warnings.warn(

Out[31]: ▾                              MLPClassifier
         MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100,
                       random_state=100)

In [32]: y_pred=ann.predict(x_test)

In [33]: from sklearn.metrics import ConfusionMatrixDisplay,classification_report
         from sklearn.metrics import accuracy_score

In [34]: y_test.value_counts()

Out[34]: 0    1981
         1     519
         Name: Exited, dtype: int64
```

```
In [35]: ConfusionMatrixDisplay.from_predictions(y_test,y_pred)

Out[35]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2091e555190>
```



```
In [37]: accuracy_score(y_test,y_pred)

Out[37]: 0.822
```

```
In [38]: print(classification_report(y_test,y_pred))

              precision    recall  f1-score   support

           0       0.86      0.93      0.89      1981
           1       0.60      0.42      0.50       519

    accuracy                           0.82      2500
   macro avg       0.73      0.68      0.69      2500
weighted avg       0.81      0.82      0.81      2500
```

```
In [ ]:
```