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

In [2]:
```
df = pd.read_csv('Churn_Modelling.csv')
```

In [3]:
```
df.shape
```

Out[3]:
```
(10000, 14)
```

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

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

In [5]:
```
df.head()
```

Out[5]:

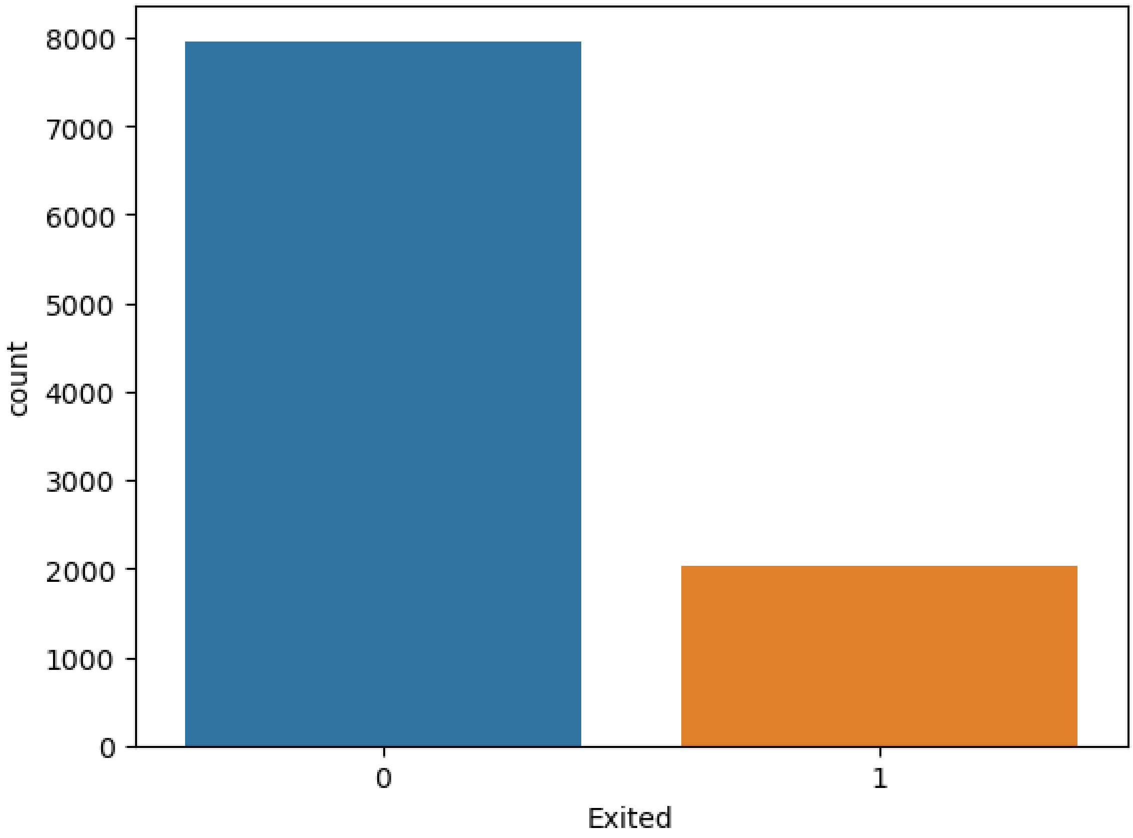
	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	E
0	1	15634602	Hargrave	619	France	Female	42	2	
1	2	15647311	Hill	608	Spain	Female	41	1	83
2	3	15619304	Onio	502	France	Female	42	8	159
3	4	15701354	Boni	699	France	Female	39	1	
4	5	15737888	Mitchell	850	Spain	Female	43	2	125

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

# output data
y = df['Exited']
```

In [7]:
```
sns.countplot(x = y)
```

Out[7]:
```
<Axes: xlabel='Exited', ylabel='count'>
```



In [8]:
```
y.value_counts()
```

Out[8]:
```
0    7963
1    2037
Name: Exited, dtype: int64
```

In [9]:
```
# normalise
# standardization
from sklearn.preprocessing import StandardScaler as ss
scaler = ss()
```

In [10]:
```
x_scaled = scaler.fit_transform(x)
```

In [11]:
```
x_scaled
```

Out[11]:
```
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 [12]:
```
# cross validation
from sklearn.model_selection import train_test_split as tts
```

In [13]:
```
xtrain,xtest,ytrain,ytest = tts(x_scaled, y, test_size=0.25)
```

In [14]:
```
x_scaled.shape
```

Out[14]:
```
(10000, 8)
```

In [16]:
```
xtest.shape
```

Out[16]:
```
(2500, 8)
```

In [17]:
```
xtrain.shape
```

Out[17]:
```
(7500, 8)
```

In [18]:
```
from sklearn.neural_network import MLPClassifier as mlpc
```

In [19]:
```
ann = mlpc(hidden_layer_sizes=(100, 100, 100),
          max_iter=100, activation='relu')
```

In [20]:
```
# train
ann.fit(xtrain, ytrain)
```

```
C:\Users\sadek\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:684: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

Out[20]:
```
▾ MLPClassifier
MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100)
```

In [21]:
```
y_pred = ann.predict(xtest)
```

In [22]:
```
y_pred
```

Out[22]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

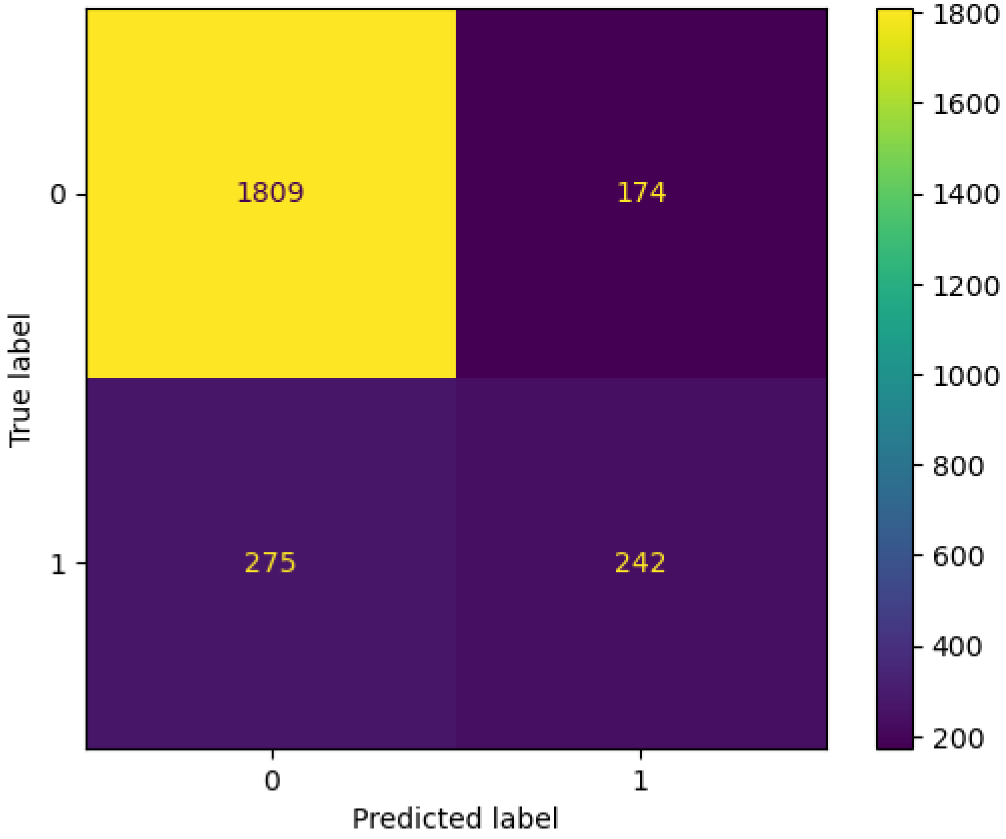
In [23]:
```
from sklearn.metrics import ConfusionMatrixDisplay as matrix
from sklearn.metrics import classification_report as report
from sklearn.metrics import accuracy_score as score
```

In [24]:
```
ytest.value_counts()
```

Out[24]:
```
0    1983
1     517
Name: Exited, dtype: int64
```

In [25]:
```
matrix.from_predictions(ytest, y_pred)
```

```
Out[25]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2453faf7e20>
```



```
In [26]: score(ytest, y_pred)
```

```
Out[26]: 0.8204
```

```
In [27]: print(report(ytest, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      1983
           1       0.58      0.47      0.52       517

    accuracy                           0.82      2500
   macro avg       0.72      0.69      0.70      2500
weighted avg       0.81      0.82      0.81      2500
```

```
In [28]: !pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\sadek\anaconda3\lib\site
-packages (0.10.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\sadek\anaconda3\lib
\site-packages (from imbalanced-learn) (2.2.0)
Requirement already satisfied: joblib>=1.1.1 in c:\users\sadek\anaconda3\lib\site-pa
ckages (from imbalanced-learn) (1.1.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\sadek\anaconda3\lib\site-pa
ckages (from imbalanced-learn) (1.23.5)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\sadek\anaconda3\lib\s
ite-packages (from imbalanced-learn) (1.2.1)
Requirement already satisfied: scipy>=1.3.2 in c:\users\sadek\anaconda3\lib\site-pac
kages (from imbalanced-learn) (1.10.0)
```

In [30]:
```
from imblearn.over_sampling import RandomOverSampler as rs
```

In [31]:
```
ros = rs()
```

In [32]:
```
x_res, y_res = ros.fit_resample(x, y)
```

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

Out[33]:
```
1    7963
0    7963
Name: Exited, dtype: int64
```

In [34]:
```
# normalize
from sklearn.preprocessing import StandardScaler as ss
```

In [35]:
```
scaler = ss()
```

In [38]:
```
x_scaled = scaler.fit_transform(x_res)
```

In [39]:
```
x_scaled
```

Out[39]:
```
array([[-0.29941334,  0.07851204, -1.02461935, ...,  0.64646199,
         1.08538632,  0.0183093 ],
       [-0.41136471, -0.01571424, -1.37009269, ..., -1.54688135,
         1.08538632,  0.21233705],
       [-1.49016885,  0.07851204,  1.0482207 , ...,  0.64646199,
        -0.92133094,  0.23641332],
       ...,
       [ 1.2068415 ,  0.36119087,  0.70274735, ...,  0.64646199,
        -0.92133094, -1.27686147],
       [ 0.27052093, -0.76952445, -1.71556604, ...,  0.64646199,
        -0.92133094, -0.61495554],
       [ 1.18648671,  0.17273831, -0.33367267, ..., -1.54688135,
         1.08538632,  1.2755192 ]])
```

In [40]:
```
x_scaled.shape
```

Out[40]:
```
(15926, 8)
```

In [41]:
```
# cross validation
```

```
In [66]: xtrain,xtest,ytrain,ytest = tts(x_scaled, y_res, test_size=0.3)
```

```
In [67]: x_res.shape
```

```
Out[67]: (15926, 8)
```

```
In [68]: xtest.shape
```

```
Out[68]: (3186, 8)
```

```
In [69]: xtrain.shape
```

```
Out[69]: (12740, 8)
```

```
In [70]: ann.fit(xtrain, ytrain)
```

```
C:\Users\sadek\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:684: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

```
Out[70]: ▾ MLPClassifier
         MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100)
```

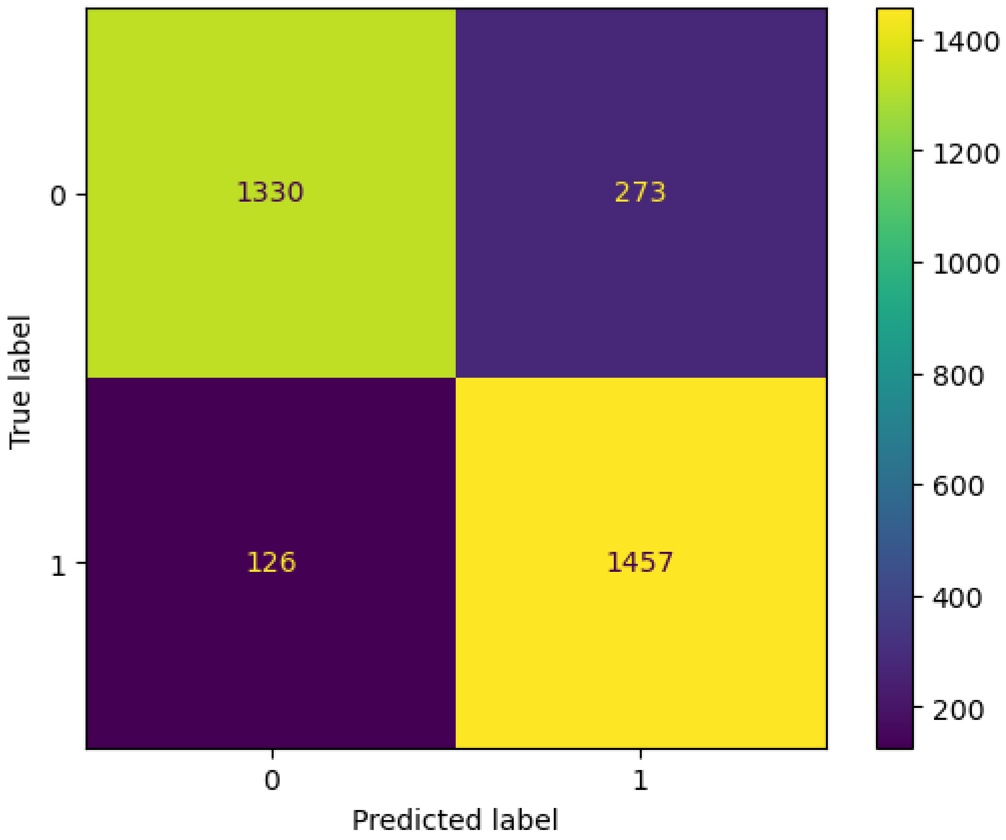
```
In [71]: y_pred = ann.predict(xtest)
```

```
In [72]: ytest.value_counts()
```

```
Out[72]: 0    1603
         1    1583
         Name: Exited, dtype: int64
```

```
In [73]: matrix.from_predictions(ytest, y_pred)
```

```
Out[73]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x245454b1ea0>
```



In [74]:
```
print(score(ytest, y_pred))
```

```
0.8747645951035782
```

In [75]:
```
print(report(ytest, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      0.83      0.87      1603
           1       0.84      0.92      0.88      1583

    accuracy                           0.87      3186
   macro avg       0.88      0.88      0.87      3186
weighted avg       0.88      0.87      0.87      3186
```

In [76]:
```
from imblearn.under_sampling import RandomUnderSampler as rs
```

In [77]:
```
rus = rs()
```

In [78]:
```
x_res, y_res = rus.fit_resample(x, y)
```

In [79]:
```
x_scaled = scaler.fit_transform(x_res)
```

In [80]:
```
xtrain,xtest,ytrain,yest = tts(x_scaled, y_res, test_size=0.2)
```

In [86]:
```
ann.fit(xtrain,ytrain)
```

```
C:\Users\sadek\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_percep
tron.py:684: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reac
hed and the optimization hasn't converged yet.
  warnings.warn(
```

Out[86]:

```
▾ MLPClassifier
MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100)
```

In [87]:

```python
y_pred = ann.predict(xtest)
```

In [88]:

```python
matrix.from_predictions(ytest, y_pred)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[88], line 1
----> 1 matrix.from_predictions(ytest, y_pred)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_plot\confusion_matrix.py:463, in
ConfusionMatrixDisplay.from_predictions(cls, y_true, y_pred, labels, sample_weight,
normalize, display_labels, include_values, xticks_rotation, values_format, cmap, ax,
colorbar, im_kw, text_kw)
    460     else:
    461         display_labels = labels
--> 463 cm = confusion_matrix(
    464     y_true,
    465     y_pred,
    466     sample_weight=sample_weight,
    467     labels=labels,
    468     normalize=normalize,
    469 )
    471 disp = cls(confusion_matrix=cm, display_labels=display_labels)
    473 return disp.plot(
    474     include_values=include_values,
    475     cmap=cmap,
   (...)
    481     text_kw=text_kw,
    482 )

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:317, in confusion_matrix(y_true, y_pred, labels, sample_weight, normalize)
    232 def confusion_matrix(
    233     y_true, y_pred, *, labels=None, sample_weight=None, normalize=None
    234 ):
    235     """Compute confusion matrix to evaluate the accuracy of a classification.
    236 
    237     By definition a confusion matrix :math:`C` is such that :math:`C_{i, j}`
   (...)
    315     (0, 2, 1, 1)
    316     """
--> 317     y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    318     if y_type not in ("binary", "multiclass"):
    319         raise ValueError("%s is not supported" % y_type)

File ~\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:86, in _check_targets(y_true, y_pred)
     59 def _check_targets(y_true, y_pred):
     60     """Check that y_true and y_pred belong to the same classification task.
     61 
     62     This converts multiclass or binary types to a common shape, and raises a
   (...)
     84     y_pred : array or indicator matrix
     85     """
---> 86     check_consistent_length(y_true, y_pred)
     87     type_true = type_of_target(y_true, input_name="y_true")
     88     type_pred = type_of_target(y_pred, input_name="y_pred")

File ~\anaconda3\lib\site-packages\sklearn\utils\validation.py:397, in check_consist
```

```
ent_length(*arrays)
    395 uniques = np.unique(lengths)
    396 if len(uniques) > 1:
--> 397     raise ValueError(
    398         "Found input variables with inconsistent numbers of samples: %r"
    399         % [int(l) for l in lengths]
    400     )

ValueError: Found input variables with inconsistent numbers of samples: [3186, 815]
```

In [84]:

```
y_res.value_counts()
```

Out[84]:

```
0    2037
1    2037
Name: Exited, dtype: int64
```

In [ ]: