In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import RobustScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
```

In [4]:
```
df = pd.read_csv('churn modelling.csv', index_col=0)
df.head()
```

Out[4]:

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

In [5]:
```
df.drop(['CustomerId', 'Surname'], axis=1, inplace=True)
```

In [6]:
```
df.shape
```

Out[6]:
```
(10000, 11)
```

In [7]:
```
df.isna().sum()
```

Out[7]:
```
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 [8]:
```
X = df.drop('Exited', 1)
y = df.Exited
y.value_counts()
```

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

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
X.columns
```

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

In [10]:
```
num_cols = ['CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'EstimatedSalary']
cat_cols = ['HasCrCard', 'IsActiveMember', 'Geography', 'Gender']
```

In [11]:
```
ct = ColumnTransformer([
    ('s1', RobustScaler(), num_cols),
    ('s2', OneHotEncoder(sparse=False, handle_unknown='ignore'), cat_cols)
])
```

In [12]:
```
p = Pipeline([
    ('ct', ct),
    ('mod', LogisticRegression(random_state=0))
])
```

In [13]:
```
p.fit(X_train, y_train)
```

Out[13]:
```
Pipeline(steps=[('ct',
                 ColumnTransformer(transformers=[('s1', RobustScaler(),
                                                  ['CreditScore', 'Age',
                                                   'Tenure', 'Balance',
                                                   'NumOfProducts',
                                                   'EstimatedSalary']),
                                                 ('s2',
                                                  OneHotEncoder(handle_unknown='ignore',
                                                                sparse=False),
                                                  ['HasCrCard',
                                                   'IsActiveMember',
                                                   'Geography', 'Gender'])])),
                ('mod', LogisticRegression(random_state=0))])
```

In [14]:
```
preds = p.predict(X_test)
preds[:15]
```

Out[14]:
```
array([1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0], dtype=int64)
```

In [15]:
```
np.array(y_test)[:15]
```

Out[15]:
```
array([1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0], dtype=int64)
```

In [16]:
```
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
confusion_matrix(y_true=y_test, y_pred=preds)
```

Out[16]:
```
array([[1530,   63],
       [ 319,   88]], dtype=int64)
```

In [17]:
```
p.classes_
```

Out[17]:
```
array([0, 1], dtype=int64)
```

In [18]:
```
confusion_matrix(y_test, preds, labels=(1,0))
```

Out[18]:
```
array([[  88,  319],
       [  63, 1530]], dtype=int64)
```

In [19]:
```
confusion_matrix(y_test, preds, labels=(1,0)).ravel()
```

Out[19]:
```
array([  88,  319,   63, 1530], dtype=int64)
```

In [38]:
```
accuracy_score(y_test, preds)
```

Out[38]:
```
0.809
```

In [39]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,\
fbeta_score, matthews_corrcoef
precision_score(y_test, preds)
```

Out[39]:
```
0.5827814569536424
```

In [40]:
```
tp, fn, fp, tn = confusion_matrix(y_test, preds, labels=(1,0)).ravel()
precision = tp/(tp+fp)
precision
```

Out[40]:
```
0.5827814569536424
```

In [41]:
```
recall_score(y_test, preds)
```

Out[41]:
```
0.21621621621621623
```

In [42]:
```
# harmonic mean of precision and recall
f1_score(y_test, preds)
```

Out[42]:
```
0.31541218637992835
```

In [ ]:
```

```

In [ ]:
```

```