```
# XG_Boost
```

In [1]:

```
#importing all library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score, roc_curve

```

In [2]:

```
df = pd.read_csv("/Users/myyntiimac/Desktop/Churn_Modelling.csv")
df.head()
```

Out[2]:

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

In [3]:

```
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 [5]:

```
X = df.iloc[:, 3:-1].values

```

In [7]:

```
X
```

Out[7]:

```
array([[619, 'France', 'Female', ..., 1, 1, 101348.88],
       [608, 'Spain', 'Female', ..., 0, 1, 112542.58],
       [502, 'France', 'Female', ..., 1, 0, 113931.57],
       ...,
       [709, 'France', 'Female', ..., 0, 1, 42085.58],
       [772, 'Germany', 'Male', ..., 1, 0, 92888.52],
       [792, 'France', 'Female', ..., 1, 0, 38190.78]], dtype=object)
```

In [9]:

```
y = df.iloc[:, -1].values
y
```

Out[9]:

```
array([1, 0, 1, ..., 1, 1, 0])
```

In [10]:

```
#converting gender column ito numerical
le = LabelEncoder()
X[:, 2] = le.fit_transform(X[:, 2])
```

In [11]:

```
X
```

Out[11]:

```
array([[619, 'France', 0, ..., 1, 1, 101348.88],
       [608, 'Spain', 0, ..., 0, 1, 112542.58],
       [502, 'France', 0, ..., 1, 0, 113931.57],
       ...,
       [709, 'France', 0, ..., 0, 1, 42085.58],
       [772, 'Germany', 1, ..., 1, 0, 92888.52],
       [792, 'France', 0, ..., 1, 0, 38190.78]], dtype=object)
```

In [12]:

```
# Assuming 'X' is your input array and 'column_index' is the index of the column you want to one-ho
column_index = 1  # Example column index

# Convert the array to a DataFrame
df = pd.DataFrame(X)

# Perform one-hot encoding using get_dummies
encoded_df = pd.get_dummies(df, columns=[column_index], drop_first=True)

# Extract the values from the encoded DataFrame
X_encoded = encoded_df.values
```

In [13]:

```
X_encoded
```

Out[13]:

```
array([[619, 0, 42, ..., 101348.88, 0, 0],
       [608, 0, 41, ..., 112542.58, 0, 1],
       [502, 0, 42, ..., 113931.57, 0, 0],
       ...,
       [709, 0, 36, ..., 42085.58, 0, 0],
       [772, 1, 42, ..., 92888.52, 1, 0],
       [792, 0, 28, ..., 38190.78, 0, 0]], dtype=object)
```

In [14]:

```
#Split the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size = 0.2, random_state = 0
```

In [71]:

```
from xgboost import XGBClassifier
classifier = XGBClassifier(learning_rate=0.0001,max_depth=15,n_estimators=400)
```

In [72]:

```
classifier.fit(X_train, y_train)
```

Out[72]:

```
▾                                XGBClassifier
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.0001, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=15, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=400, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

In [73]:

```
y_pred = classifier.predict(X_test)
```

In [74]:

```
y_pred
```

Out[74]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

In [75]:

```

cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[1467  128]
 [ 197  208]]
```

In [76]:

```
ac = accuracy_score(y_test, y_pred)
print(ac)
```

```
0.8375
```

In [59]:

```
bias = classifier.score(X_train,y_train)
bias
```

Out[59]:

```
0.865375
```

In [60]:

```
Variance = classifier.score(X_test,y_test)
Variance
```

Out[60]:

```
0.8635
```

In [61]:

```
#ROC AND AUC
from sklearn.metrics import roc_curve, roc_auc_score
```

In [62]:

```
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
fpr, tpr, thresholds
```

Out[62]:

```
(array([0.        , 0.03636364, 1.        ]),
 array([0.       , 0.4691358, 1.       ]),
 array([2, 1, 0]))
```

In [63]:

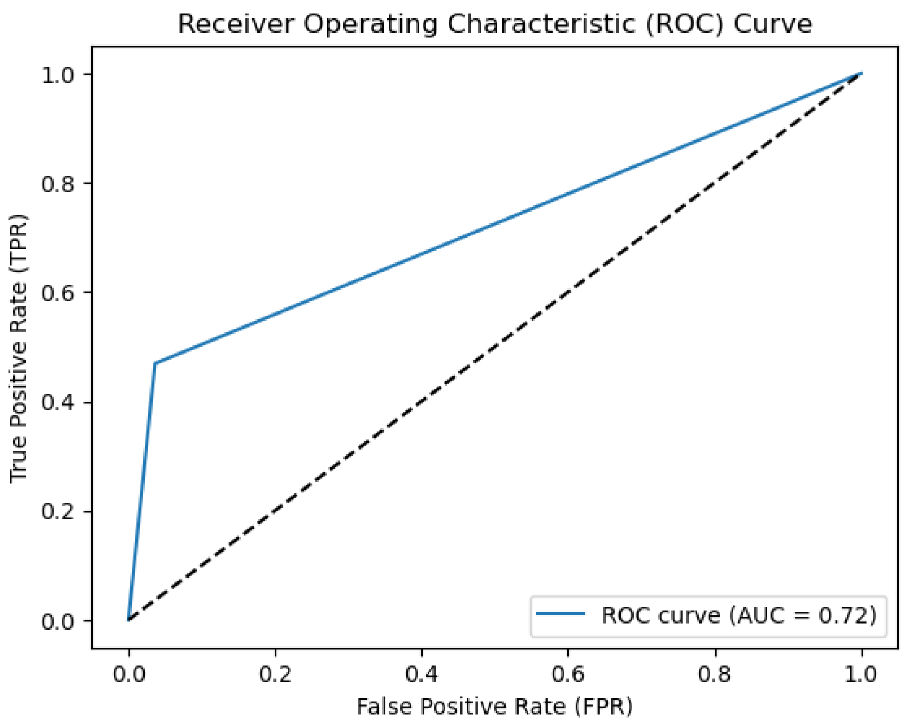
```
auc = roc_auc_score(y_test, y_pred)
auc
```

Out[63]:

```
0.7163860830527496
```

In [64]:

```
# Plotting the ROC curve
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(auc))
plt.plot([0, 1], [0, 1], 'k--')  # Random guess line
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



```
Insight:In the case of AUC = 0.75, the model demonstrates reasonable discriminative ability, but there is still room for improvement. It correctly ranks 75% of the positive samples higher than the negative samples, on average, across different classification thresholds. However, it might misclassify some instances, leading to false positives or false negatives.
```