# Combined all classifier model

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]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCa
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 [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
X
```

Out[5]:

```
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 [6]:

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

Out[6]:

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

In [7]:

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

In [8]:

```python
# 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 [9]:

```python
X_encoded
```

Out[9]:

```
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 [10]:

```python
#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 [11]:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

In [33]:

```
# Initialize a dictionary to store the accuracy scores
accuracy_scores = {}

# Define the classifier models
models = {
    'Random Forest': RandomForestClassifier(),
    'Logistic Regression': LogisticRegression(),
    'Support Vector Machine': SVC(),
    'K-Nearest Neighbors': KNeighborsClassifier(),
    'Decision Tree': DecisionTreeClassifier(),
    'Naive Bayes': GaussianNB()
}

# Iterate over each model
for model_name, model in models.items():
    # Fit the model on the training data
    model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = model.predict(X_test)

    # Calculate the accuracy score
    accuracy = accuracy_score(y_test, y_pred)

    # Store the accuracy score in the dictionary
    accuracy_scores[model_name] = accuracy

# Sort the accuracy scores in descending order
sorted_scores = sorted(accuracy_scores.items(), key=lambda x: x[1], reverse=True)

# Print the ranking and accuracy scores of the models
print("Model Rankings based on Accuracy Score:")
for rank, (model_name, accuracy) in enumerate(sorted_scores, start=1):
    print(f"Rank {rank}: {model_name} - Accuracy: {accuracy:.4f}")

# Print all the accuracy scores
print("\nAll Accuracy Scores:")
for model_name, accuracy in accuracy_scores.items():
    print(f"{model_name}: {accuracy:.4f}")
```

```
Model Rankings based on Accuracy Score:
Rank 1: Random Forest - Accuracy: 0.8650
Rank 2: Decision Tree - Accuracy: 0.8005
Rank 3: Support Vector Machine - Accuracy: 0.7975
Rank 4: Logistic Regression - Accuracy: 0.7890
Rank 5: Naive Bayes - Accuracy: 0.7850
Rank 6: K-Nearest Neighbors - Accuracy: 0.7645

All Accuracy Scores:
Random Forest: 0.8650
Logistic Regression: 0.7890
Support Vector Machine: 0.7975
K-Nearest Neighbors: 0.7645
Decision Tree: 0.8005
Naive Bayes: 0.7850
```

In [38]:

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

In [56]:

```
# Train the Random Forest model
rd = RandomForestClassifier()
rd.fit(X_train, y_train)
```

Out[56]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [57]:

```
y_prob = rd.predict_proba(X_test)[:, 1]

```

In [58]:

```
# Compute the False Positive Rate (FPR), True Positive Rate (TPR), and thresholds
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
```

In [60]:

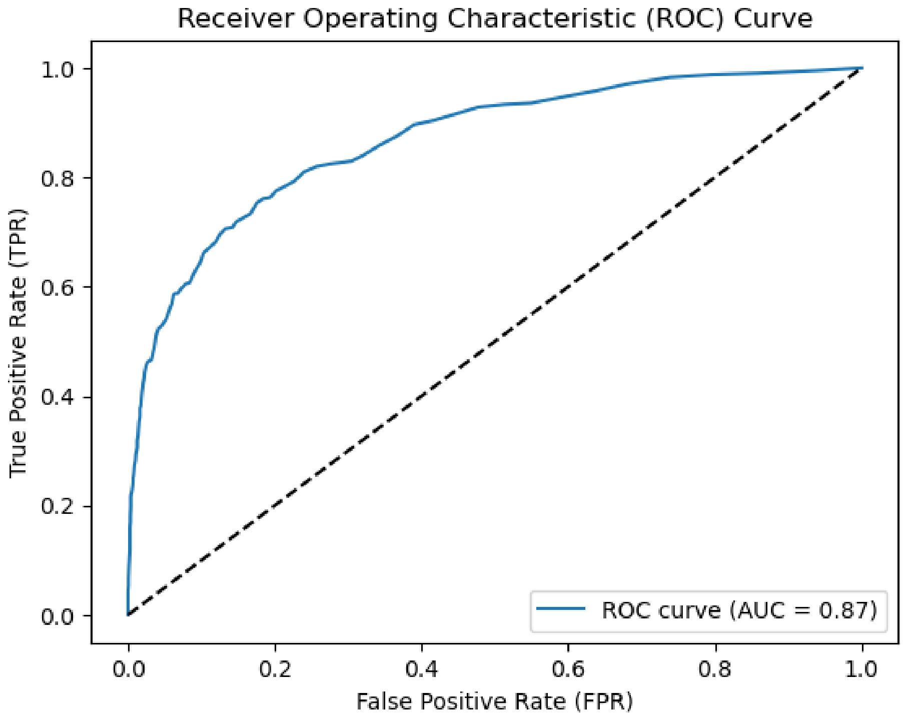
```
auc = roc_auc_score(y_test, y_prob)
auc
```

Out[60]:

```
0.8677077286272687
```

In [61]:

```
# 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 = 0.87, the model demonstrates reasonable discriminative ability, but
there is still room for improvement. It correctly ranks 87% 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.
```