

In [1]:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
```

In [2]:

```
1 dataset = pd.read_csv('/Users/myyntiimac/Desktop/Social_Network_Ads.csv')
2 X = dataset.iloc[:, [2, 3]].values
3 y = dataset.iloc[:, -1].values
```

In [3]:

```
1 from sklearn.preprocessing import StandardScaler
2 sc = StandardScaler()
3 X = sc.fit_transform(X)
```

In [4]:

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

In [6]:

```
1 from sklearn.naive_bayes import GaussianNB
```

In [7]:

```
1 Regression=GaussianNB()
```

In [8]:

```
1 Regression.fit(X_train, y_train)
```

Out[8]:

```
▼ GaussianNB
GaussianNB()
```

In [9]:

```
1 y_pred =Regression.predict(X_test)
```

In [10]:

```
1 from sklearn.metrics import confusion_matrix
2 cm = confusion_matrix(y_test, y_pred)
3 print(cm)
```

```
[[65  3]
 [ 7 25]]
```

In [11]:

```
1 bias =Regression.score(X_train, y_train)
2 bias
```

Out[11]:

```
0.8833333333333333
```

In [12]:

```
1 variance = Regression.score(X_test, y_test)
2 variance
```

Out[12]:

0.9

In [13]:

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

In [14]:

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

In [16]:

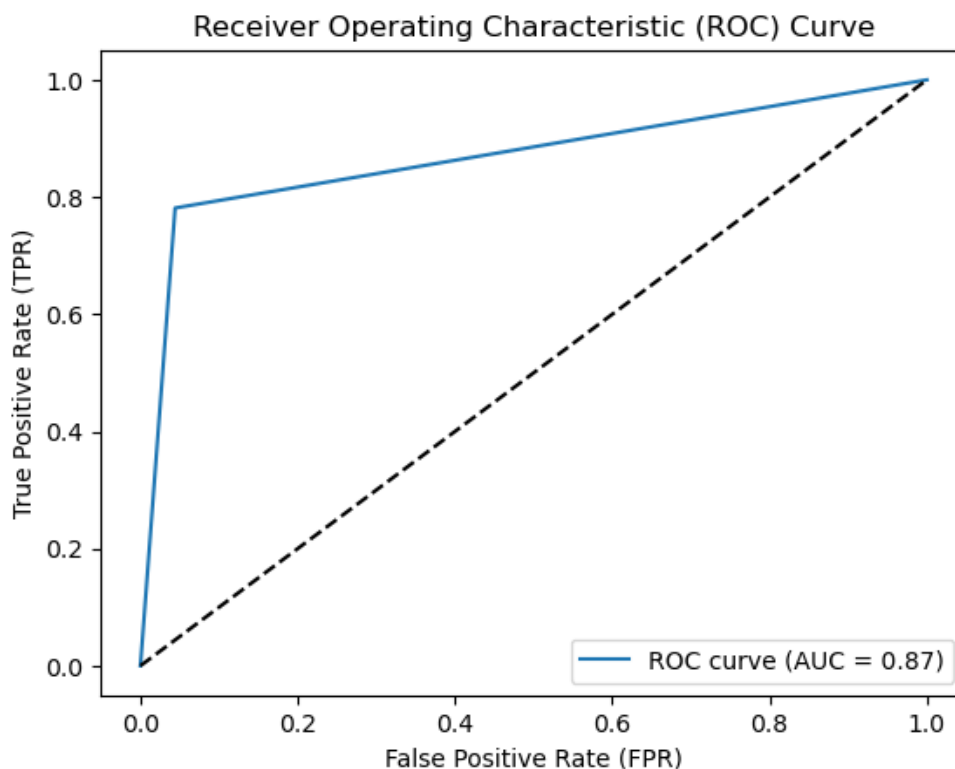
```
1 auc = roc_auc_score(y_test, y_pred)
2 auc
```

Out[16]:

0.8685661764705882

In [17]:

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



```
1 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.
```

In [18]:

```
1 from sklearn.model_selection import cross_val_score
2 accuracies = cross_val_score(estimator = Regression, X = X_train, y = y_train, cv = 10)
3 print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
4 print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

Accuracy: 87.67 %

Standard Deviation: 8.95 %

In [22]:

```
1 from sklearn.model_selection import GridSearchCV
```

In [23]:

```
1 param_grid = {
2     'var_smoothing': [1e-9, 1e-8, 1e-7] # Example grid with different values of var_smoothing
3 }
```

In [24]:

```
1 # Create an object of GridSearchCV
2 grid_search = GridSearchCV(estimator= Regression, param_grid=param_grid, cv=10)
3 grid_search = grid_search.fit(X_train, y_train)
4 best_accuracy = grid_search.best_score_
5 best_parameters = grid_search.best_params_
6 print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
7 print("Best Parameters:", best_parameters)
```

Best Accuracy: 87.67 %

Best Parameters: {'var\_smoothing': 1e-09}

In [ ]:

```
1
```