In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [3]:

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

In [4]:

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

In [5]:

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

In [37]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [38]:

```
classifier=DecisionTreeClassifier()
```

In [39]:

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

Out[39]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

In [40]:

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

In [41]:

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

```
[[62  6]
 [ 4 28]]
```

In [45]:

```
from sklearn.metrics import accuracy_score
# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

Out[45]:

```
0.9
```

In [42]:

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

Out[42]:

```
1.0
```

In [43]:

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

Out[43]:

```
0.9
```

In [46]:

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

In [47]:

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

In [48]:

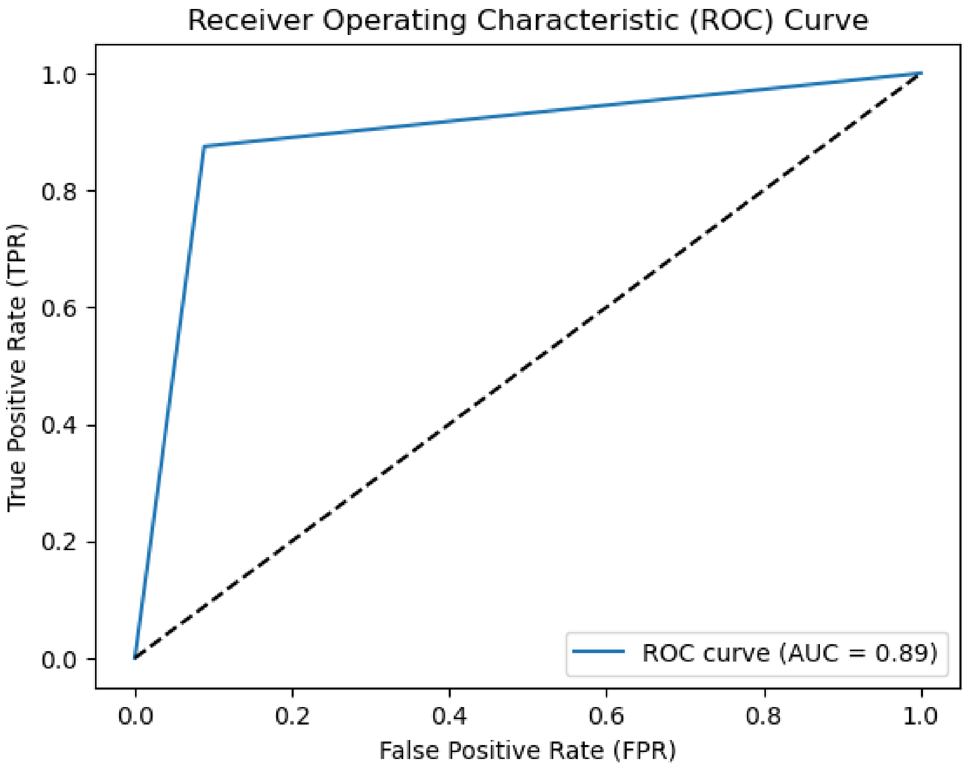
```
auc = roc_auc_score(y_test, y_pred)
auc
```

Out[48]:

```
0.8933823529411764
```

In [49]:

```
# 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 = 0.89, the model demonstrates reasonable discriminative ability, but there is still room for improvement. It correctly ranks 89% 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 [50]:

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

```
Accuracy: 86.00 %
Standard Deviation: 5.33 %
```

In [51]:

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

In [35]:

```
param_grid = {
    'penalty': ['l2'],  # Regularization penalty term
    'C': [0.1, 1.0, 10.0],  # Inverse of regularization strength
    'solver': ['liblinear', 'lbfgs', 'saga'],  # Solver algorithm
    'max_iter': [100, 200, 300],  # Maximum number of iterations
    'fit_intercept': [True, False],  # Include intercept term
    'class_weight': [None, 'balanced']  # Class weights
}
```

In [55]:

```
# Create the GridSearchCV object
grid_search = GridSearchCV(estimator=classifier, param_grid=param_grid, cv=10)
grid_search = GridSearchCV(classifier, param_grid, cv=10)
grid_search = grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print("Best Accuracy: {:.2f} %".format(best_accuracy*100))
print("Best Parameters:", best_parameters)

```

```
Best Accuracy: 89.33 %
Best Parameters: {'criterion': 'entropy', 'max_depth': 5, 'max_features': None, 'mi
n_samples_leaf': 4, 'min_samples_split': 5}
```

In [ ]:

```

```