Assignment 9

Title: Model Evaluation

Name: Pranjal Rane

NUID: 002756852

In [1]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import (accuracy_score, precision_score, recall_score,
                             classification_report, confusion_matrix,
                             roc_curve, auc, precision_recall_curve,
                             average_precision_score)
from sklearn.preprocessing import StandardScaler
```

Fetching Data (Wine Quality Dataset)

In [2]:

```
!wget https://archive.ics.uci.edu/static/public/186/wine+quality.zip
```

```
--2024-03-24 19:38:23--  https://archive.ics.uci.edu/static/public/186/wine+quality.zip
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified
Saving to: 'wine+quality.zip'

wine+quality.zip        [ <=>                ]  89.21K   484KB/s    in 0.2s

2024-03-24 19:38:24 (484 KB/s) - 'wine+quality.zip' saved [91353]
```

In [3]:

```
!unzip -o wine+quality.zip
```

```
Archive:  wine+quality.zip
  inflating: winequality-red.csv
  inflating: winequality-white.csv
  inflating: winequality.names
```

In [4]:

```
data_red_wine = pd.read_csv('winequality-red.csv', sep=';')
data_white_wine = pd.read_csv('winequality-white.csv', sep=';')
```

In [5]:

```
data_red_wine['wineType'] = 1
data_white_wine['wineType'] = 0
```

In [6]:

```
df = pd.concat([data_red_wine, data_white_wine], ignore_index=True)
df.sample(5)
```

Out[6]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	wineType
2938	8.4	0.58	0.27	12.15	0.033	37.0	116.0	0.99590	2.99	0.39	10.8	6	0
2106	6.0	0.24	0.27	1.90	0.048	40.0	170.0	0.99380	3.64	0.54	10.0	7	0
5748	5.8	0.24	0.28	1.40	0.038	40.0	76.0	0.98711	3.10	0.29	13.9	7	0
1490	7.1	0.22	0.49	1.80	0.039	8.0	18.0	0.99344	3.39	0.56	12.4	6	1
6215	7.3	0.28	0.37	1.20	0.039	26.0	99.0	0.99198	3.01	0.62	10.8	5	0

In [7]:

```
features = [i for i in df.columns]
features.remove('wineType')
print(features)
```

```
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free
sulfur dioxide', 'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'qualit
y']
```

Split the dataset into training set and test set (80, 20).

In [8]:

```
x = df[features]
y = df['wineType']
y = y.astype(int)

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, stratify=y, rand
om_state=42)
```

In [9]:

```
sc_x = StandardScaler()
x_train = sc_x.fit_transform(x_train)
x_test = sc_x.transform(x_test)
```

Helper Function to Calculate Evaluation Metrics

In [10]:

```
def evaluation_metrics(decision_tree):
    y_pred = decision_tree.predict(x_test)

    # 1. Accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)

    # 2. Precision and Recall
    precision = precision_score(y_test, y_pred, average='binary')
    recall = recall_score(y_test, y_pred, average='binary')
    print("Precision:", precision)
    print("Recall:", recall)

    # 3. Classification Report
    report = classification_report(y_test, y_pred)
    print("Classification Report:\n", report)

    # 4. Confusion Matrix
    conf_matrix = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:\n", conf_matrix)

    # 5. ROC Curve
    y_prob = decision_tree.predict_proba(x_test)[:, 1]  # Get probabilities for the posit
ive class
```

```
    fpr, tpr, thresholds = roc_curve(y_test, y_prob)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_
auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic')
    plt.legend(loc="lower right")
    plt.show()

    # 6. Precision/Recall Curve
    precision, recall, _ = precision_recall_curve(y_test, decision_tree.predict_proba(x_
test)[:, 1])
    average_precision = average_precision_score(y_test, y_pred)

    # Plot the precision-recall curve
    plt.figure()
    plt.step(recall, precision, where='post', label=f'Average precision (AP)={average_pr
ecision:.2f}')
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.ylim([0.0, 1.05])
    plt.xlim([0.0, 1.0])
    plt.title('2-class Precision-Recall curve')
    plt.legend(loc="best")
    plt.show()
```

Using scikit-learn's DecisionTreeClassifier, train a supervised learning model that can be used to generate predictions for your data. Report on the six evaluation metrics listed in objective

In [11]:

```
decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(x_train, y_train)
```

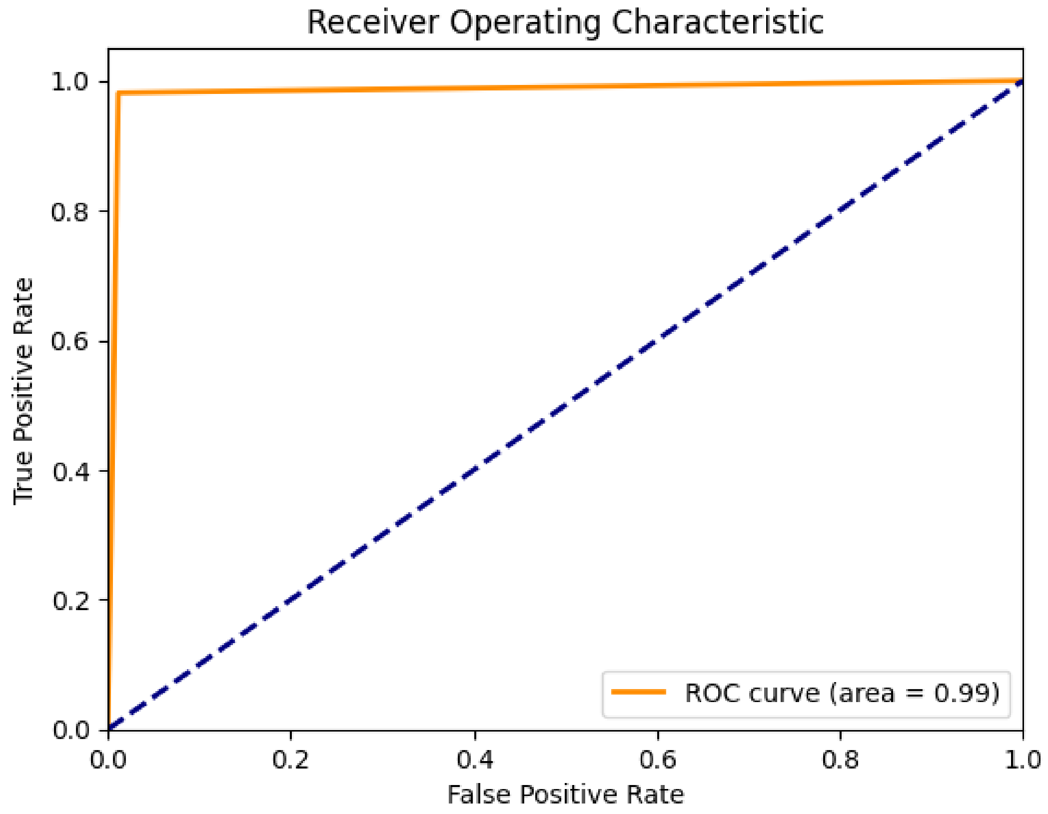
Out[11]:

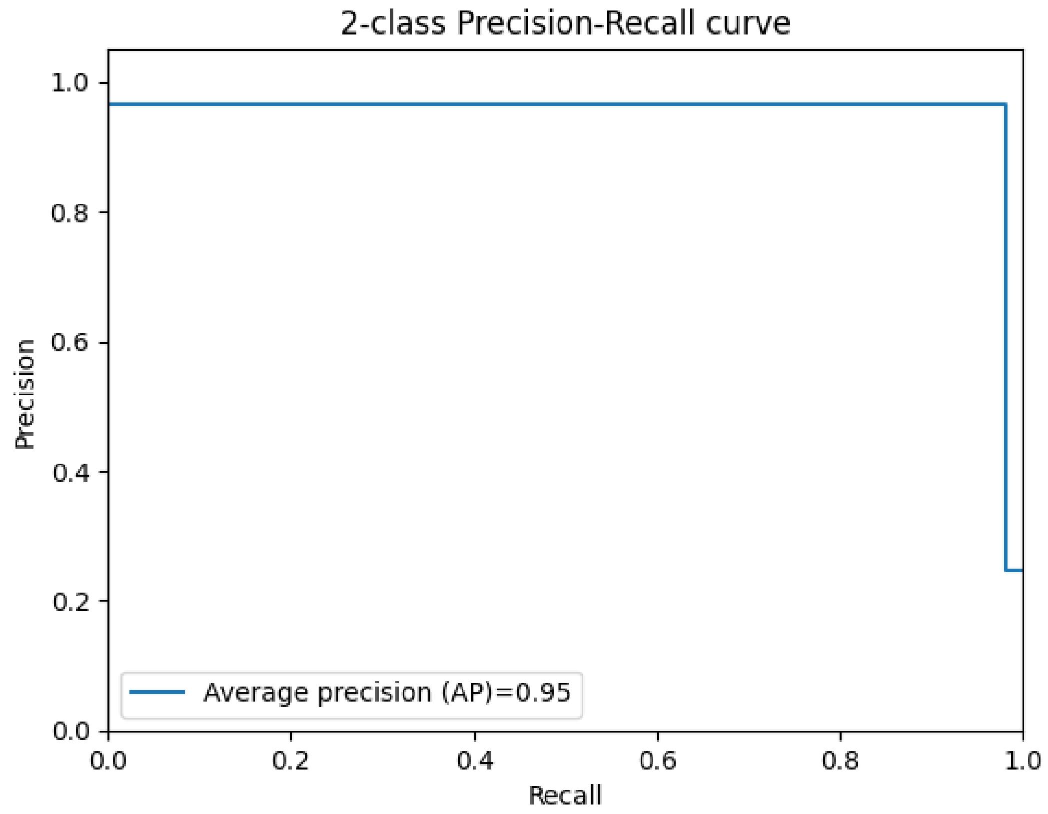
```
▼ DecisionTreeClassifier  i ?

DecisionTreeClassifier(random_state=42)
```

In [12]:

```
print ("For the given Decision Tree : ")
evaluation_metrics(decision_tree)
```

```
For the given Decision Tree :
Accuracy: 0.9869230769230769
Precision: 0.9661538461538461
Recall: 0.98125
Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.97      0.98      0.97       320

    accuracy                           0.99      1300
   macro avg       0.98      0.99      0.98      1300
weighted avg       0.99      0.99      0.99      1300

Confusion Matrix:
 [[969  11]
 [  6 314]]
```





Similarly as in previous step, train another Decision Tree Classifier - but in this case set the maximum depth of the tree to 1 (max_depth = 1). Use the same training and test set as you used for the Decision Tree in the previous step. Report on the six evaluation metrics listed in objective

In [13]:

```
decision_tree2 = DecisionTreeClassifier(random_state=42, max_depth = 1)
decision_tree2.fit(x_train, y_train)
```

Out[13]:

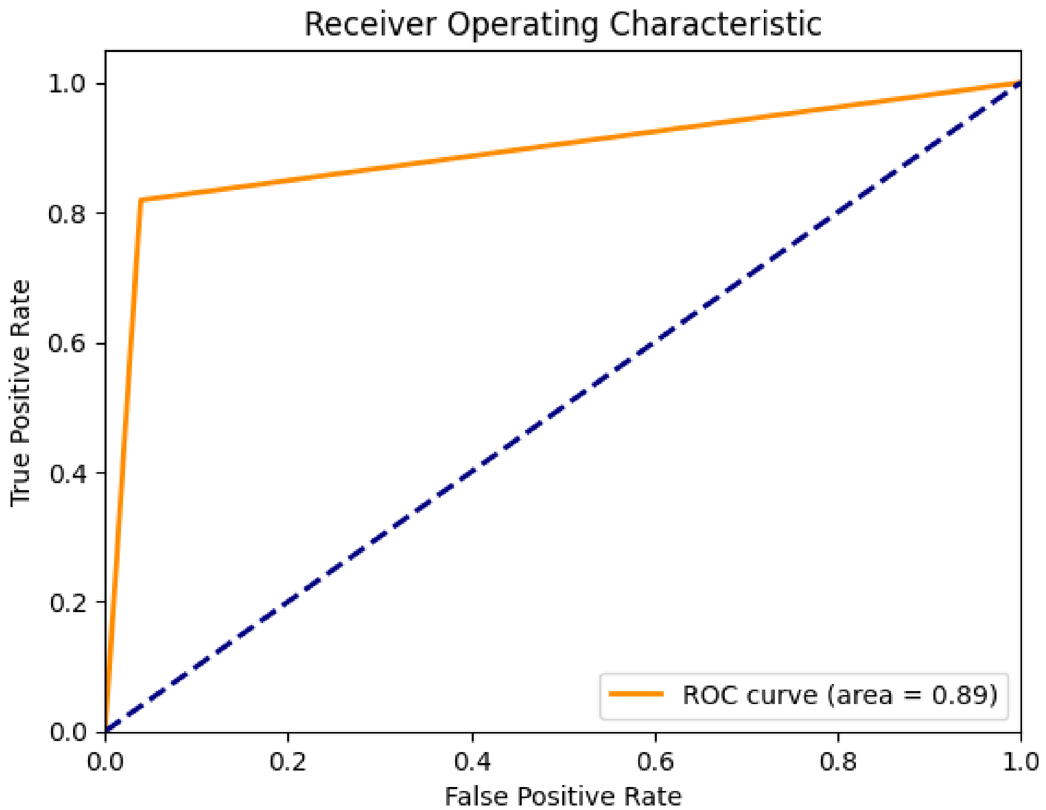
▾ DecisionTreeClassifier i ?

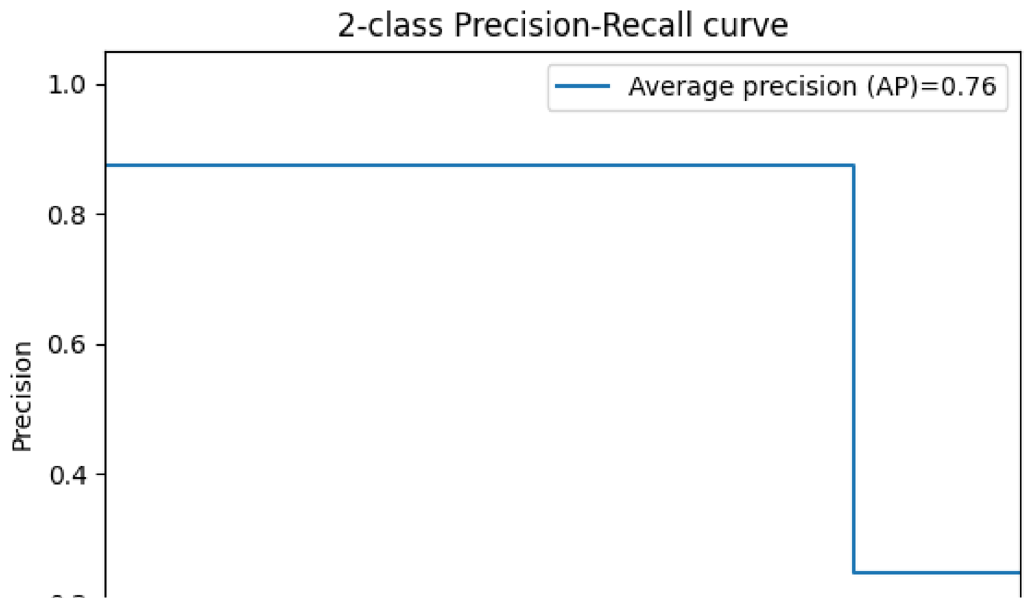
DecisionTreeClassifier(max_depth=1, random_state=42)

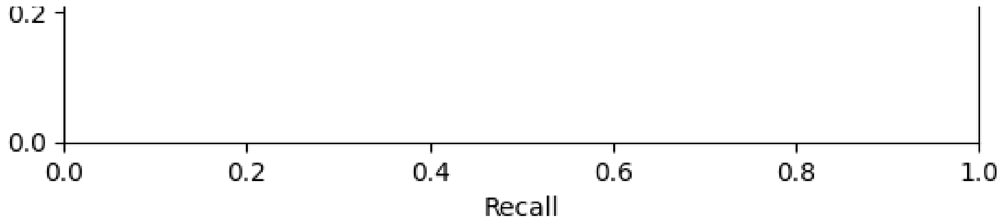
In [14]:

```
print ("For the given Decision Tree with max_depth = 1: ")
evaluation_metrics(decision_tree2)
```

```
For the given Decision Tree with max_depth = 1:
Accuracy: 0.9261538461538461
Precision: 0.8733333333333333
Recall: 0.81875
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.96      0.95       980
           1       0.87      0.82      0.85       320

    accuracy                           0.93      1300
   macro avg       0.91      0.89      0.90      1300
weighted avg       0.93      0.93      0.93      1300

Confusion Matrix:
 [[942  38]
 [ 58 262]]
```






0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Recall