ShriDattaMadhira_Assignment9

November 10, 2021

[2]:
```
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()

X = data.data
y = data.target

col_names = data.feature_names
```

[3]:
```
X[:5]
```

[3]:
```
array([[1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01, 2.776e-01,
        3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00, 9.053e-01,
        8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02, 1.587e-02,
        3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02, 2.019e+03,
        1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01, 1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, 1.326e+03, 8.474e-02, 7.864e-02,
        8.690e-02, 7.017e-02, 1.812e-01, 5.667e-02, 5.435e-01, 7.339e-01,
        3.398e+00, 7.408e+01, 5.225e-03, 1.308e-02, 1.860e-02, 1.340e-02,
        1.389e-02, 3.532e-03, 2.499e+01, 2.341e+01, 1.588e+02, 1.956e+03,
        1.238e-01, 1.866e-01, 2.416e-01, 1.860e-01, 2.750e-01, 8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, 1.203e+03, 1.096e-01, 1.599e-01,
        1.974e-01, 1.279e-01, 2.069e-01, 5.999e-02, 7.456e-01, 7.869e-01,
        4.585e+00, 9.403e+01, 6.150e-03, 4.006e-02, 3.832e-02, 2.058e-02,
        2.250e-02, 4.571e-03, 2.357e+01, 2.553e+01, 1.525e+02, 1.709e+03,
        1.444e-01, 4.245e-01, 4.504e-01, 2.430e-01, 3.613e-01, 8.758e-02],
       [1.142e+01, 2.038e+01, 7.758e+01, 3.861e+02, 1.425e-01, 2.839e-01,
        2.414e-01, 1.052e-01, 2.597e-01, 9.744e-02, 4.956e-01, 1.156e+00,
        3.445e+00, 2.723e+01, 9.110e-03, 7.458e-02, 5.661e-02, 1.867e-02,
        5.963e-02, 9.208e-03, 1.491e+01, 2.650e+01, 9.887e+01, 5.677e+02,
        2.098e-01, 8.663e-01, 6.869e-01, 2.575e-01, 6.638e-01, 1.730e-01],
       [2.029e+01, 1.434e+01, 1.351e+02, 1.297e+03, 1.003e-01, 1.328e-01,
        1.980e-01, 1.043e-01, 1.809e-01, 5.883e-02, 7.572e-01, 7.813e-01,
        5.438e+00, 9.444e+01, 1.149e-02, 2.461e-02, 5.688e-02, 1.885e-02,
        1.756e-02, 5.115e-03, 2.254e+01, 1.667e+01, 1.522e+02, 1.575e+03,
        1.374e-01, 2.050e-01, 4.000e-01, 1.625e-01, 2.364e-01, 7.678e-02]])
```

[4]:
```
y[:5]
```

[4]:
```
array([0, 0, 0, 0, 0])
```

[5]:
```
col_names
```

[5]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

0.0.1 Q1. Split the dataset into training set and test set (80, 20).

[6]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.80,␣
↪test_size=0.20)
```

0.0.2 Q2. Using scikit-learn's DecisionTreeClassifier, train a supervised learning model that can be used to generate predictions for your data.

[7]:
```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier().fit(X_train, y_train)
predict = dtc.predict(X_test)

Td = dtc.get_depth()
print("Depth of the classifier: ", Td)
print("Number of leaves in the classifier: ", dtc.get_n_leaves())
feature_importances = dtc.tree_.compute_feature_importances(normalize=False)
print("Feature importance for the classifier: ", str(feature_importances))
```

```
Depth of the classifier:  7
Number of leaves in the classifier:  21
Feature importance for the classifier:  [0.         0.01098901 0.         0.
0.0043956  0.
 0.         0.0364979  0.003663   0.         0.         0.
 0.00832591 0.00211506 0.00068266 0.         0.         0.01333839
 0.         0.00424791 0.         0.00544723 0.33716141 0.
 0.01072882 0.         0.0043061  0.02503976 0.         0.        ]
```

0.0.3 Q3. 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.

[8]:
```
dtc1 = DecisionTreeClassifier(max_depth=1).fit(X_train, y_train)
predict1 = dtc1.predict(X_test)

Td = dtc1.get_depth()
print("Depth of the classifier: ", Td)
print("Number of leaves in the classifier: ", dtc1.get_n_leaves())
feature_importances1 = dtc1.tree_.compute_feature_importances(normalize=False)
print("Feature importance for the classifier: ", str(feature_importances1))
```

```
Depth of the classifier:  1
Number of leaves in the classifier:  2
Feature importance for the classifier:  [0.         0.         0.         0.
0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.33166691 0.
 0.         0.         0.         0.         0.         ]
```

0.0.4 Q4. Report on the six evaluation metrics listed in objective for both the models, and compare their results.

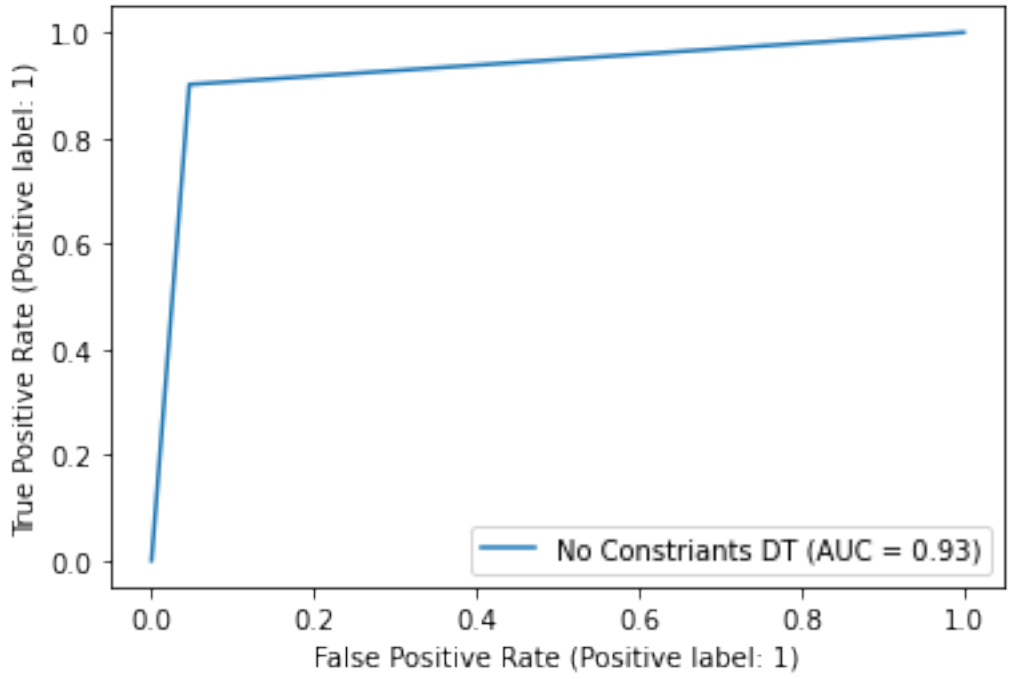
0.0.5 Model 1 - Decision Tree with no depth restriction

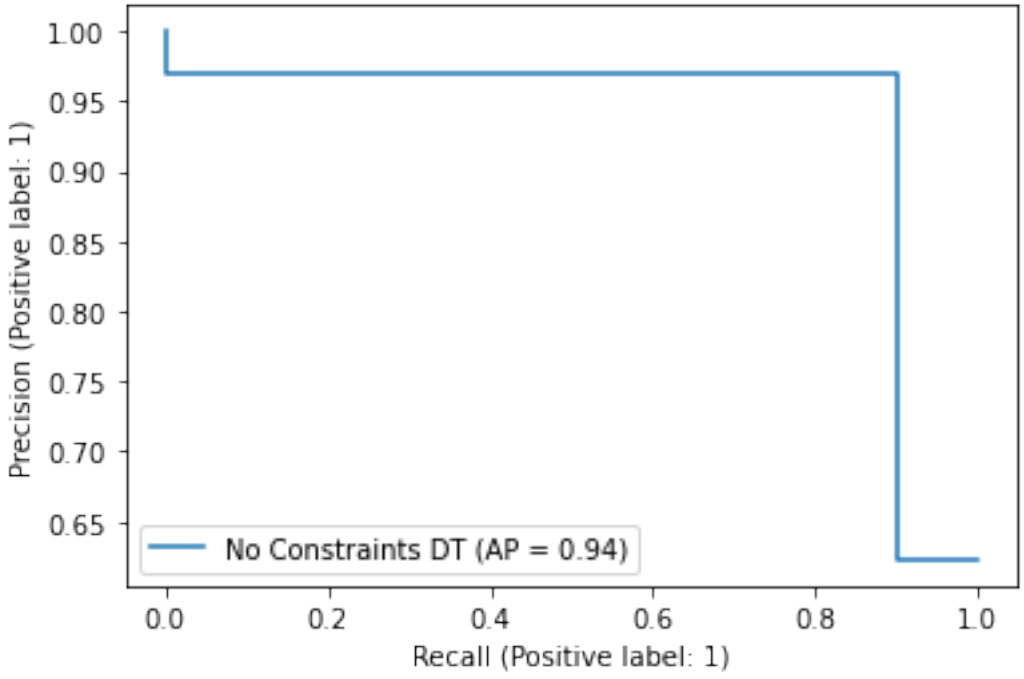
[9]:
```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
from sklearn.metrics import precision_score, recall_score,␣
↪PrecisionRecallDisplay
from sklearn.metrics import RocCurveDisplay
```

[14]:
```
# Accuracy of model on test data
print("Accuracy score on test data for this model: ", accuracy_score(y_test,␣
↪predict), "\n")

# Precision and Recall values
print("Precision for this model: ", precision_score(y_test, predict), "\n")
print("Recall for this model: ", recall_score(y_test, predict), "\n")

# Classification Report
print("Classification Report for this model:\n", classification_report(y_test,␣
↪predict), "\n")

# Confusion Matrix
```

```
print("Confusion Matrix for this model:\n", confusion_matrix(y_test, predict),␣
 ↪"\n")

# ROC Curve
roc_display = RocCurveDisplay.from_predictions(y_test, predict, name="No␣
 ↪Constriants DT")


# Precision Recall Curve
pr_display = PrecisionRecallDisplay.from_predictions(y_test, predict,
                                                     name="No Constraints DT")
```

```
Accuracy score on test data for this model:  0.9210526315789473

Precision for this model:  0.9696969696969697

Recall for this model:  0.9014084507042254

Classification Report for this model:
              precision    recall  f1-score   support

           0       0.85      0.95      0.90        43
           1       0.97      0.90      0.93        71

    accuracy                           0.92       114
   macro avg       0.91      0.93      0.92       114
weighted avg       0.93      0.92      0.92       114


Confusion Matrix for this model:
 [[41  2]
 [ 7 64]]
```


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
No Constriants DT (AUC = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0

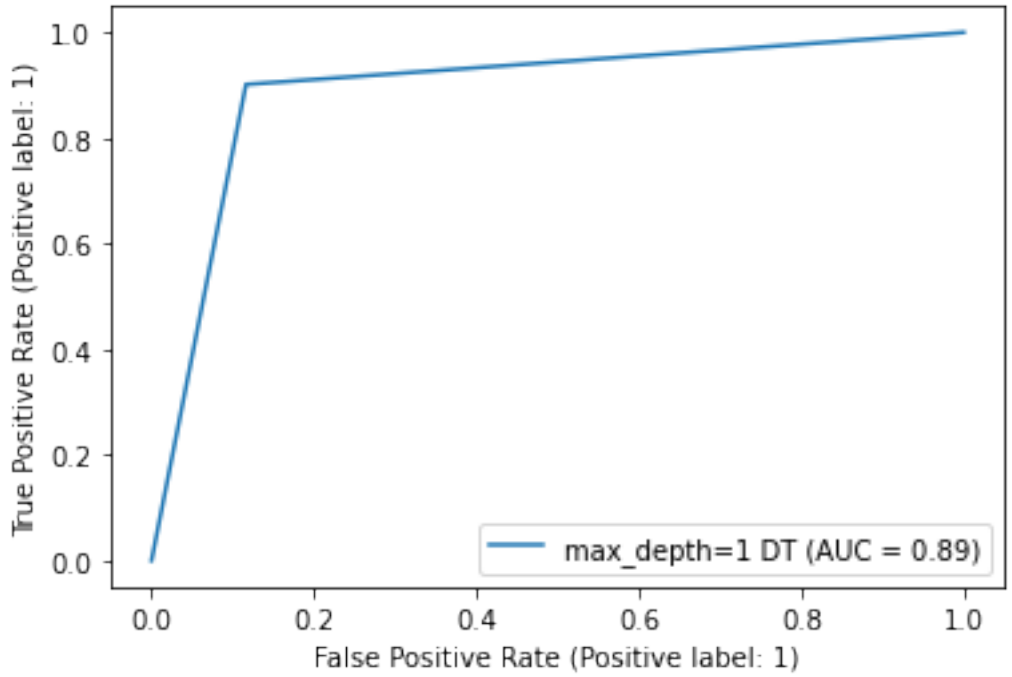

Precision (Positive label: 1)
Recall (Positive label: 1)
No Constraints DT (AP = 0.94)
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.0
0.2
0.4
0.6
0.8
1.0

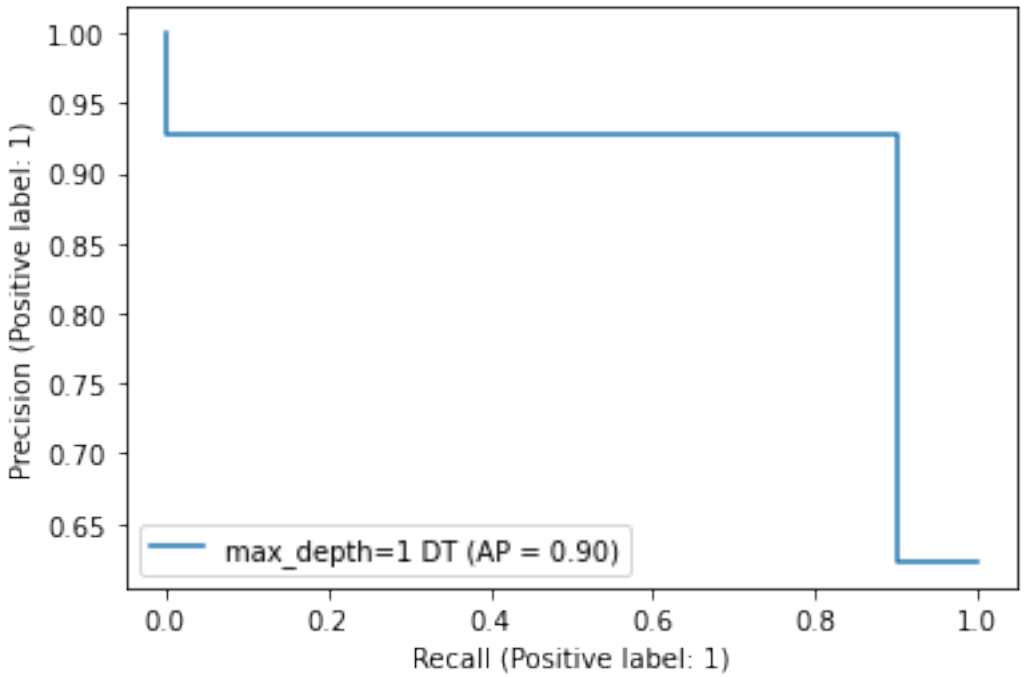
0.0.6 Model 2 - Decision Tree Classifier with max_depth = 1

[13]:
```
# Accuracy of model on test data
print("Accuracy score on test data for this model: ", accuracy_score(y_test,␣
 ↪predict1), "\n")

# Precision and Recall values
print("Precision for this model: ", precision_score(y_test, predict1), "\n")
print("Recall for this model: ", recall_score(y_test, predict1), "\n")

# Classification Report
print("Classification Report for this model:\n", classification_report(y_test,␣
 ↪predict1), "\n")

# Confusion Matrix
print("Confusion Matrix for this model:\n", confusion_matrix(y_test, predict1),␣
 ↪"\n")

# ROC Curve
print("ROC Curve and Precision Recall Curve\n")
roc_display = RocCurveDisplay.from_predictions(y_test, predict1,␣
 ↪name="max_depth=1 DT")

# Precision Recall Curve
pr_display = PrecisionRecallDisplay.from_predictions(y_test, predict1,
                                                     name="max_depth=1 DT")
```

```
Accuracy score on test data for this model:  0.8947368421052632

Precision for this model:  0.927536231884058

Recall for this model:  0.9014084507042254

Classification Report for this model:
               precision    recall  f1-score   support

           0       0.84      0.88      0.86        43
           1       0.93      0.90      0.91        71

    accuracy                           0.89       114
   macro avg       0.89      0.89      0.89       114
weighted avg       0.90      0.89      0.90       114


Confusion Matrix for this model:
 [[38  5]
 [ 7 64]]
```

ROC Curve and Precision Recall Curve





0.0.7 Comparing both the models

The dataset we took is a binary classification dataset with two classes as target classes. Coming to the comparision of results between the two models, we can see that, although the accuracy for depth 1 decision tree is less, there isn't much difference between both the models. The same goes for precision and recall.

But from the ROC curve, we can see that the number of False Positive rates reduced from depth 1 model to full range model. More data is classified correctly.