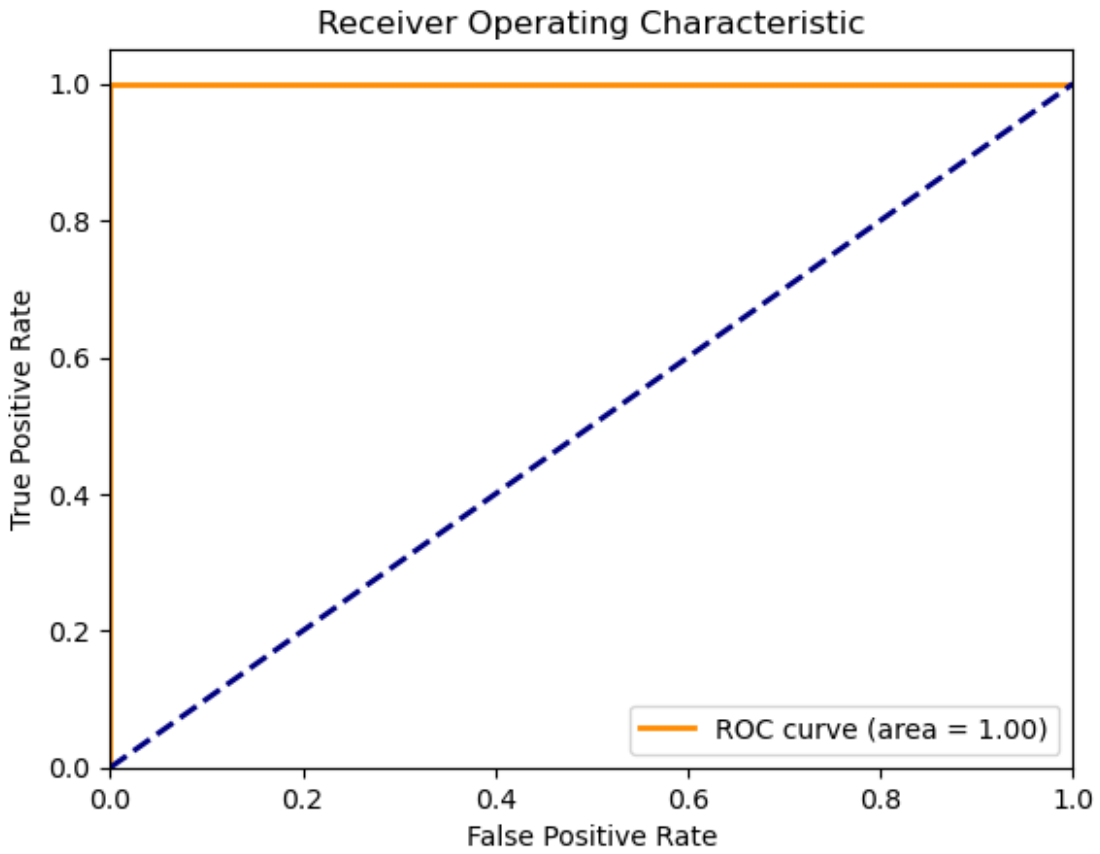
- Name: Deep Salunkhe
- Roll No.:21102A0014
- SEM-7 ML Lab3 Github Link

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, classification_report,
roc_curve, auc
import matplotlib.pyplot as plt

# Load the Iris dataset
iris_url = "iris.data"
iris_columns = ['sepal_length', 'sepal_width', 'petal_length',
'petal_width', 'species']
iris_data = pd.read_csv(iris_url, header=None, names=iris_columns)

# Explore the dataset
print(iris_data.head())

# Split the dataset into training and testing sets
X = iris_data.drop('species', axis=1)
y = iris_data['species']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Construct a Decision Tree classifier using the Gini Index
clf = DecisionTreeClassifier(criterion='gini')
clf.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = clf.predict(X_test)

# Evaluate the classifier's performance using the specified metrics
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

# Classification Report
class_report = classification_report(y_test, y_pred)
print("Classification Report:\n", class_report)

# ROC Curve and AUC
y_prob = clf.predict_proba(X_test)
fpr, tpr, _ = roc_curve(y_test, y_prob[:, 1],
pos_label=clf.classes_[1])
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()
```

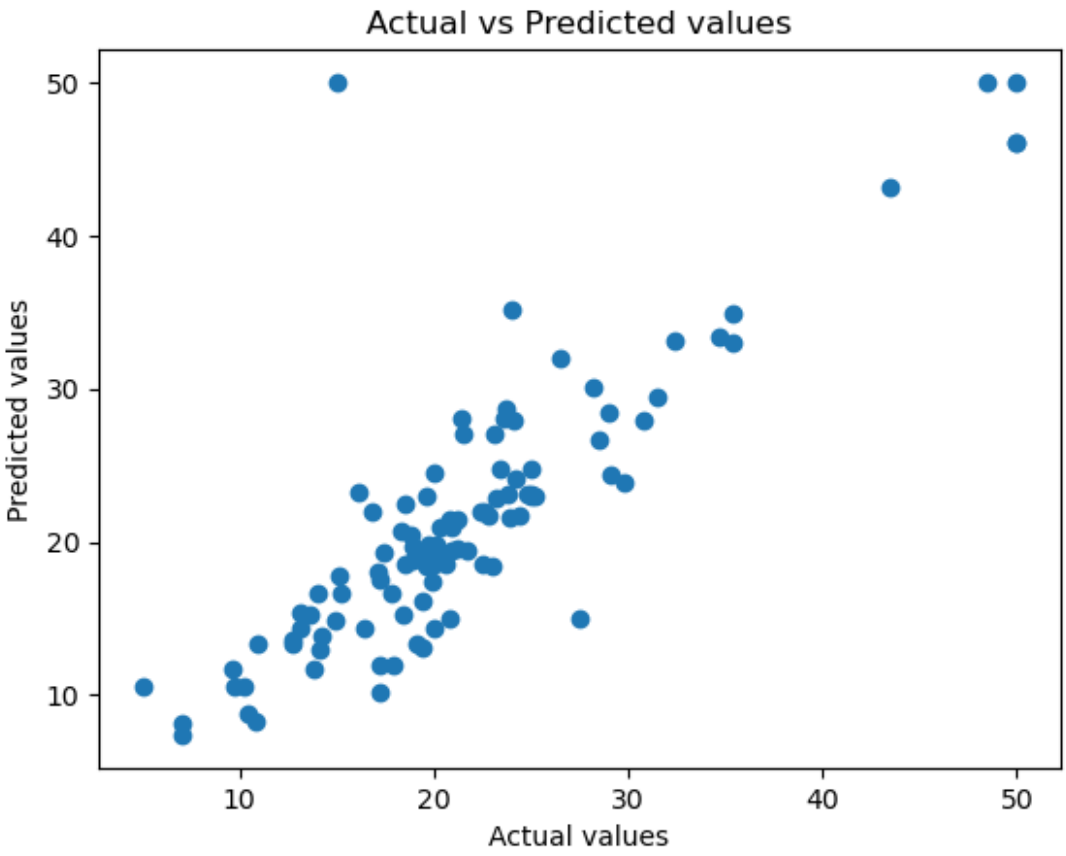
```
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa
Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
Classification Report:
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```




```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt

# Load the dataset
# Assuming the provided image represents a similar structure to the
Boston Housing dataset.
column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
housing_data = pd.read_csv('housing.csv', names=column_names,
delim_whitespace=True)

# Explore the dataset
print(housing_data.head())

# Split the dataset into training and testing sets
X = housing_data.drop('MEDV', axis=1)
y = housing_data['MEDV']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```python
# Construct a Decision Tree regressor using mean squared error
regr = DecisionTreeRegressor(criterion='squared_error')
regr.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = regr.predict(X_test)

# Evaluate the regressor's performance using the specified metrics
# Mean Squared Error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)

# Plot actual vs. predicted values
plt.figure()
plt.scatter(y_test, y_pred)
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.title('Actual vs Predicted values')
plt.show()
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX
\
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0

1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0

2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0

3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0

4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0


   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
Mean Squared Error: 23.55705882352941
```


Actual vs Predicted values
Predicted values
Actual values
10
20
30
40
50