Ex.No – 8 Roll No – 210701149

Implement SVM and Decision Tree Classification Techniques

AIM:

To implement SVM / Decision Tree Classification Techniques in Python.

PROCEDURES:

1. Collect and load the dataset from sources like CSV files or databases.
2. Clean and preprocess the data, including handling missing values and encoding categorical variables.
3. Split the dataset into training and testing sets to evaluate model performance.
4. Normalize or standardize the features, especially for SVM, to ensure consistent scaling.
5. Choose the appropriate model: SVM for margin-based classification, Decision Tree for rule-based classification.
6. Train the model on the training data using the `fit` method.
7. Make predictions on the testing data using the `predict` method.
8. Evaluate the model using metrics like accuracy, confusion matrix, precision, and recall.
9. Visualize the results with plots, such as decision boundaries for SVM or tree structures for Decision Trees.
10. Fine-tune the model by adjusting hyperparameters like `C` for SVM or `max_depth` for Decision Trees.

CODE:

SVM.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data[:, :2]  # We'll use only the first two features for simplicity
y = iris.target

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Create and train the SVM model
model = SVC(kernel='linear')
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
```

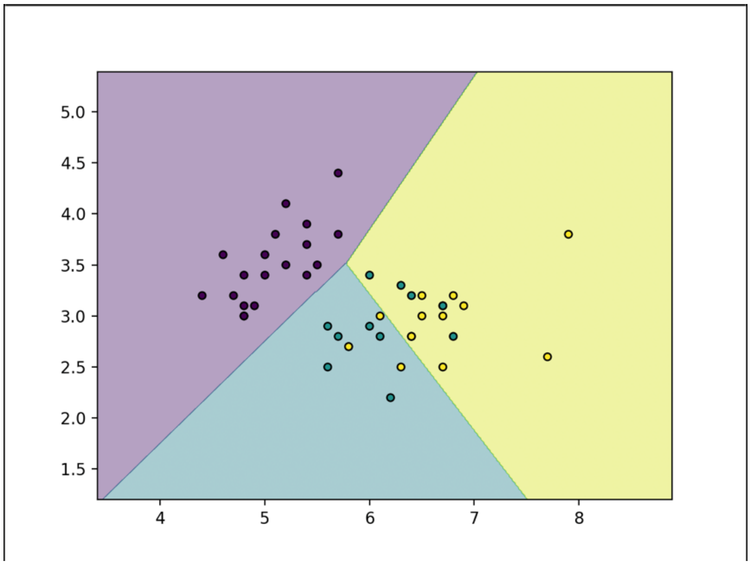
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

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

# Visualize the decision boundary
def plot_decision_boundary(X, y, model):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                        np.arange(y_min, y_max, 0.01))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.4)
    plt.scatter(X[:, 0], X[:, 1], c=y, s=20, edgecolor='k')
    plt.show()

plot_decision_boundary(X_test, y_test, model)
```

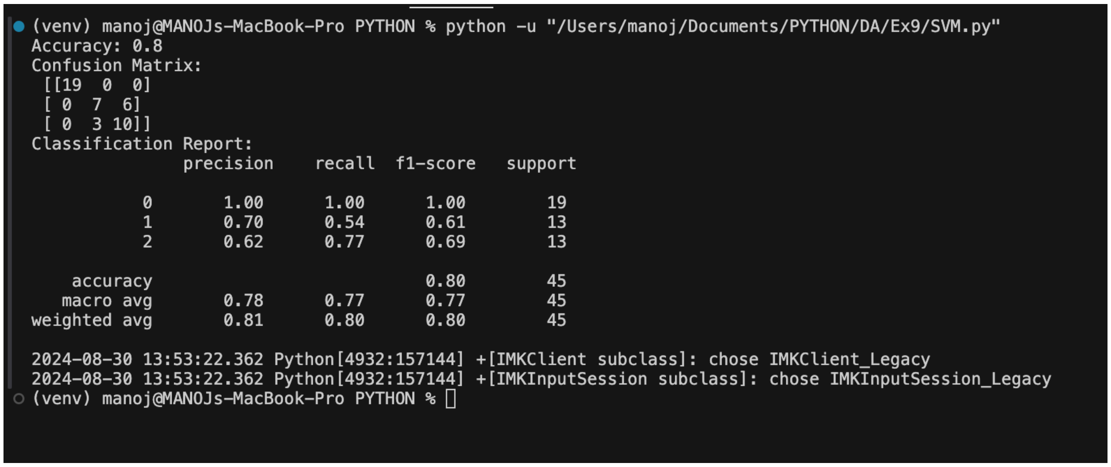
DecisionTree.py

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data[:, :2]  # We'll use only the first two features for simplicity
y = iris.target

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Create and train the Decision Tree model
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)
```

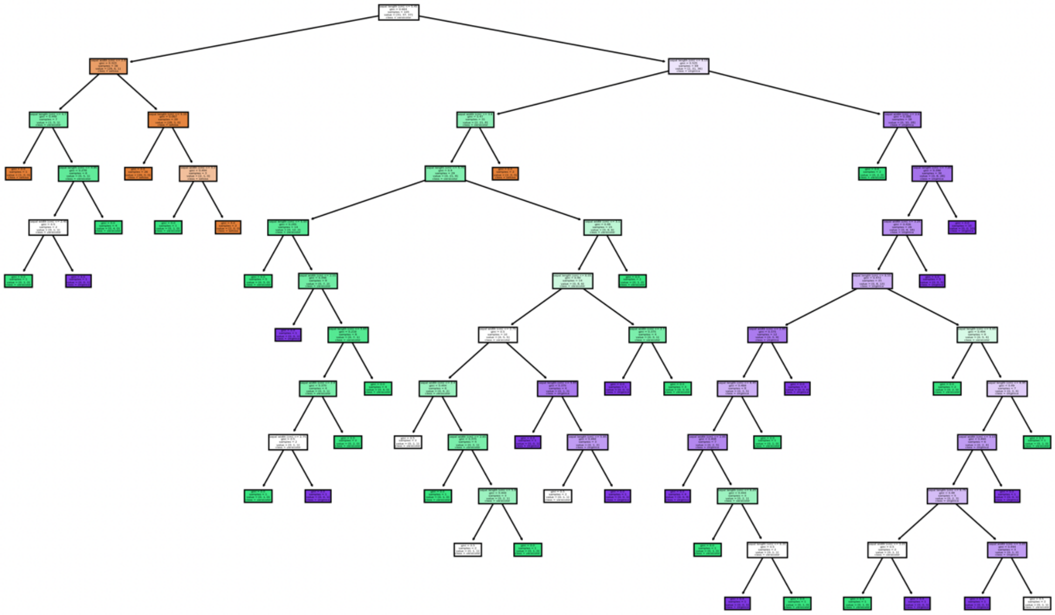
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

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

# Visualize the decision tree
plt.figure(figsize=(10, 8))
plot_tree(model, filled=True, feature_names=iris.feature_names[:2],
class_names=iris.target_names)
plt.show()
```

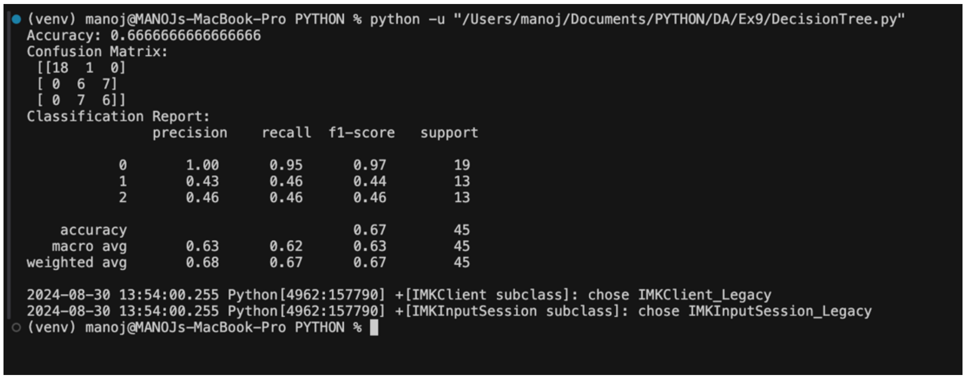
OUTPUT:





```
● (venv) manoj@MANOJs-MacBook-Pro PYTHON % python -u "/Users/manoj/Documents/PYTHON/DA/Ex9/SVM.py"
  Accuracy: 0.8
  Confusion Matrix:
   [[19  0  0]
   [ 0  7  6]
   [ 0  3 10]]
  Classification Report:
                precision    recall  f1-score   support

             0       1.00      1.00      1.00        19
             1       0.70      0.54      0.61        13
             2       0.62      0.77      0.69        13

      accuracy                           0.80        45
     macro avg       0.78      0.77      0.77        45
  weighted avg       0.81      0.80      0.80        45

  2024-08-30 13:53:22.362 Python[4932:157144] +[IMKClient subclass]: chose IMKClient_Legacy
  2024-08-30 13:53:22.362 Python[4932:157144] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
○ (venv) manoj@MANOJs-MacBook-Pro PYTHON % 
```





```
● (venv) manoj@MANOJs-MacBook-Pro PYTHON % python -u "/Users/manoj/Documents/PYTHON/DA/Ex9/DecisionTree.py"
  Accuracy: 0.6666666666666666
  Confusion Matrix:
   [[18  1  0]
   [ 0  6  7]
   [ 0  7  6]]
  Classification Report:
                precision    recall  f1-score   support

             0       1.00      0.95      0.97        19
             1       0.43      0.46      0.44        13
             2       0.46      0.46      0.46        13

      accuracy                           0.67        45
     macro avg       0.63      0.62      0.63        45
  weighted avg       0.68      0.67      0.67        45

  2024-08-30 13:54:00.255 Python[4962:157790] +[IMKClient subclass]: chose IMKClient_Legacy
  2024-08-30 13:54:00.255 Python[4962:157790] +[IMKInputSession subclass]: chose IMKInputSession_Legacy
○ (venv) manoj@MANOJs-MacBook-Pro PYTHON % 
```

RESULT:

Thus, to implement the SVM / Decision Tree Classification Techniques are completed successfully.