Ex.No: 8 **Roll No : 210701159**

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

```
# Install and load the e1071 package (if not already installed)
library(e1071)
# Load the iris dataset
data(iris)
# Inspect the first few rows of the dataset
head(iris)
# Split the data into training (70%) and testing (30%) sets
```

```
set.seed(123) # For reproducibility
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
# Fit the SVM model
svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
# Print the summary of the model
summary(svm_model)
# Predict the test set
predictions <- predict(svm_model, newdata = test_data)
# Evaluate the model's performance
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
print(confusion_matrix)
# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy * 100, "%\n")
```

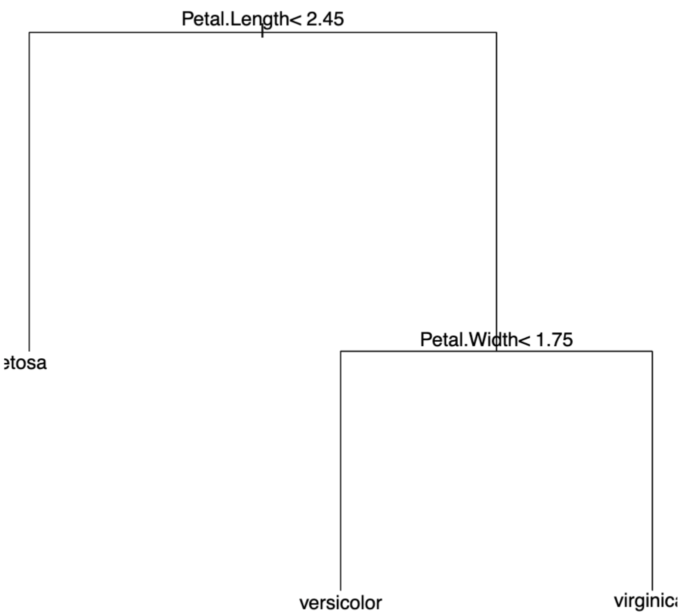
DecisionTree.py

```
# Install and load the rpart package (if not already installed)
library(rpart)
# Load the iris dataset
data(iris)
# Split the data into training (70%) and testing (30%) sets
set.seed(123) # For reproducibility
sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
train_data <- iris[sample_indices, ]
test_data <- iris[-sample_indices, ]
# Fit the Decision Tree model
tree_model <- rpart(Species ~ ., data = train_data, method = "class")
# Print the summary of the model
summary(tree_model)
# Plot the Decision Tree
plot(tree_model)
```

```
text(tree_model, pretty = 0)
# Predict the test set
predictions <- predict(tree_model, newdata = test_data, type = "class")
# Evaluate the model's performance
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
print(confusion_matrix)
# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
cat("Accuracy:", accuracy * 100, "%\n")
```

OUTPUT:

```
            Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
Accuracy: 97.77778 %
>
```



```
Call:
rpart(formula = Species ~ ., data = train_data, method = "class")
  n= 105

          CP nsplit  rel error    xerror       xstd
1 0.5294118      0 1.00000000 1.2058824 0.06232572
2 0.3970588      1 0.47058824 0.5441176 0.07198662
3 0.0100000      2 0.07352941 0.1176471 0.03997857

Variable importance
 Petal.Width Petal.Length Sepal.Length  Sepal.Width
          34           32           21           13

Node number 1: 105 observations,    complexity param=0.5294118
  predicted class=virginica   expected loss=0.647619  P(node) =1
    class counts:    36    32    37
   probabilities: 0.343 0.305 0.352
  left son=2 (36 obs) right son=3 (69 obs)
  Primary splits:
      Petal.Length < 2.45 to the left,  improve=35.54783, (0 missing)
      Petal.Width  < 0.8  to the left,  improve=35.54783, (0 missing)
      Sepal.Length < 5.45 to the left,  improve=24.79179, (0 missing)
      Sepal.Width  < 3.25 to the right, improve=12.34670, (0 missing)
  Surrogate splits:
      Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.000, (0 split)
      Sepal.Length < 5.45 to the left,  agree=0.924, adj=0.778, (0 split)
      Sepal.Width  < 3.25 to the right, agree=0.819, adj=0.472, (0 split)
```

```
Node number 2: 36 observations
  predicted class=setosa      expected loss=0  P(node) =0.3428571
    class counts:    36     0     0
   probabilities: 1.000 0.000 0.000

Node number 3: 69 observations,    complexity param=0.3970588
  predicted class=virginica   expected loss=0.4637681  P(node) =0.6571429
    class counts:     0    32    37
   probabilities: 0.000 0.464 0.536
  left son=6 (35 obs) right son=7 (34 obs)
  Primary splits:
      Petal.Width  < 1.75 to the left,  improve=25.291950, (0 missing)
      Petal.Length < 4.75 to the left,  improve=25.187810, (0 missing)
      Sepal.Length < 6.15 to the left,  improve= 5.974246, (0 missing)
      Sepal.Width  < 2.45 to the left,  improve= 2.411006, (0 missing)
  Surrogate splits:
      Petal.Length < 4.75 to the left,  agree=0.913, adj=0.824, (0 split)
      Sepal.Length < 6.15 to the left,  agree=0.696, adj=0.382, (0 split)
      Sepal.Width  < 2.65 to the left,  agree=0.638, adj=0.265, (0 split)

Node number 6: 35 observations
  predicted class=versicolor  expected loss=0.1142857  P(node) =0.3333333
    class counts:     0    31     4
   probabilities: 0.000 0.886 0.114

Node number 7: 34 observations
  predicted class=virginica   expected loss=0.02941176  P(node) =0.3238095
    class counts:     0     1    33
   probabilities: 0.000 0.029 0.971

            Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         18         1
  virginica       0          0        12
Accuracy: 97.77778 %
```

RESULT:

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