Ex.No – 8 Roll No – 210701190

Implemention of 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)

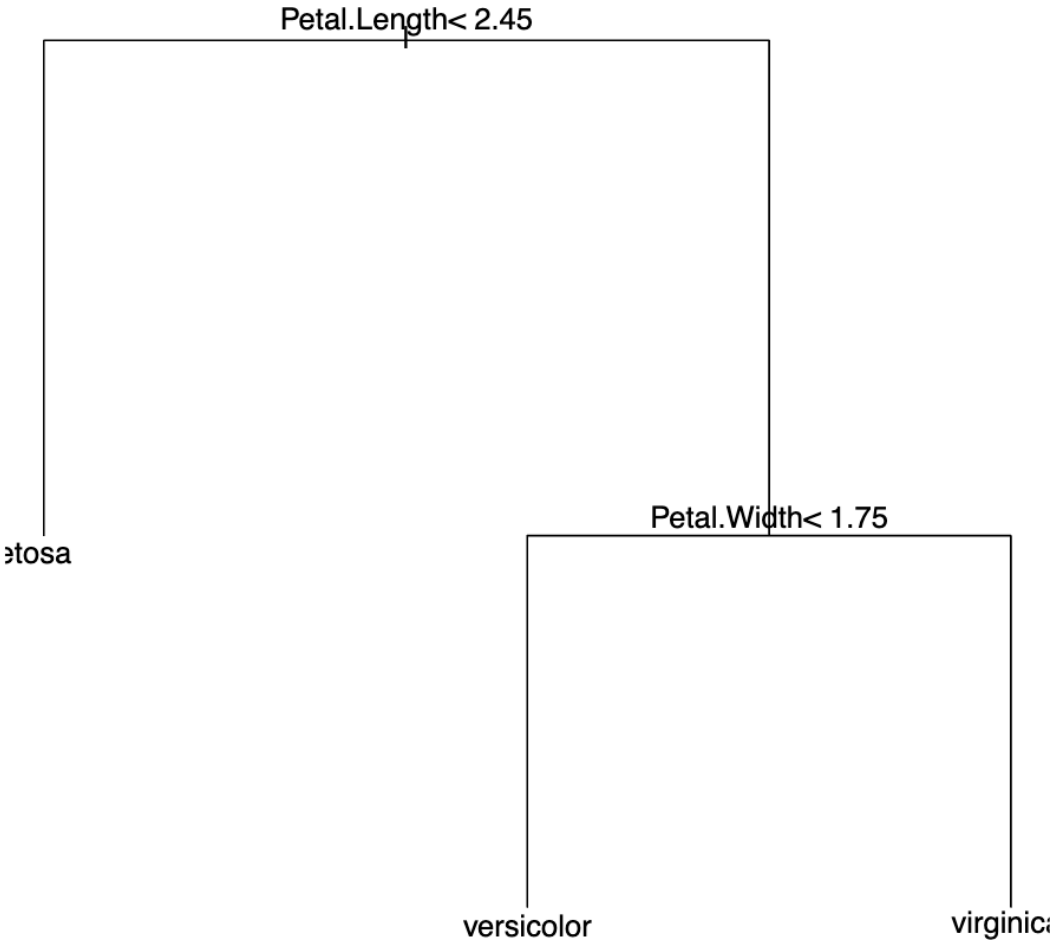
Calculate accuracy

accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix) cat("Accuracy:",

accuracy * 100, "%\n")

OUTPUT:

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

Call:
svm(formula = Species ~ ., data = train_data, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  45

 ( 7 18 20 )


Number of Classes:  3

Levels:
 setosa versicolor virginica



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


Petal.Length< 2.45
etosa
Petal.Width< 1.75
versicolor
virginica

RESULT:

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