

Implement SVM/Decision tree classification technique

AIM:

To Implement SVM and Decision tree classification techniques using R programming in R Studio.

a) SVM IN R

```
# Install and load the e1071 package (if not already installed)
install.packages("e1071") 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")
```

OUTPUT:

```

SVM.R
Source on Save
Package e1071 required but is not installed. Install Don't Show Again
1 # Install and load the e1071 package (if not already installed)
2 install.packages("e1071")
3 library(e1071)
4 # Load the iris dataset
5 data(iris)
6 # Inspect the first few rows of the dataset
7 head(iris)
8 # Split the data into training (70%) and testing (30%) sets
9 set.seed(123) # For reproducibility
10 sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
11 train_data <- iris[sample_indices, ]
12 test_data <- iris[-sample_indices, ]
13 # Fit the SVM model
14 svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
15 # Print the summary of the model
16 summary(svm_model)
17 # Predict the test set
18 predictions <- predict(svm_model, newdata = test_data)
19 # Evaluate the model's performance
20 confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
21 print(confusion_matrix)
22 # Calculate accuracy
23 accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
24 cat("Accuracy:", accuracy * 100, "%\n")

```

package 'proxy' successfully unpacked and MD5 sums checked
 package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in

C:\Users\Jayar\AppData\Local\Temp\RtmpsHAtXR\downloaded_packages

	Actual		
Predicted	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	0
virginica	0	1	13

Accuracy: 97.77778 %

b) Decision tree in R

```
# Install and load the rpart package (if not already installed)
install.packages("rpart") 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)

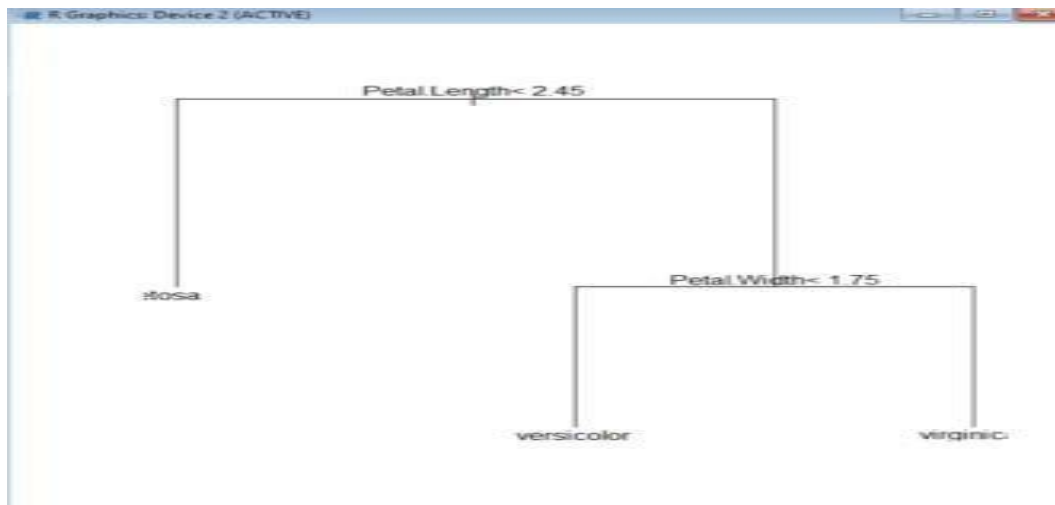
# Calculate accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix) cat("Accuracy:",
accuracy * 100, "%\n")
```

OUTPUT:

```

1 # Install and load the rpart package (if not already installed)
2 install.packages("rpart")
3 library(rpart)
4 # Load the iris dataset
5 data(iris)
6 # Split the data into training (70%) and testing (30%) sets
7 set.seed(123) # For reproducibility
8 sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
9 train_data <- iris[sample_indices, ]
10 test_data <- iris[-sample_indices, ]
11 # Fit the Decision Tree model
12 tree_model <- rpart(Species ~ ., data = train_data, method = "class")
13 # Print the summary of the model
14 summary(tree_model)
15 # Plot the Decision Tree
16 plot(tree_model)
17 text(tree_model, pretty = 0)
18 # Predict the test set
19 predictions <- predict(tree_model, newdata = test_data, type = "class")
20 # Evaluate the model's performance
21 confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
22 print(confusion_matrix)
23 # Calculate accuracy
24 accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
25 cat("Accuracy:", accuracy * 100, "%\n")

```

**RESULT:**

Thus, the Implementation SVM/Decision tree classification techniques using R programming in R Studio.