

SVM AND DECISION TREE IMPLEMENTATION

AIM:

To implement Support Vector Machine and Decision Tree classification techniques using R.

Support Vector Machine:

PROGRAM:

```
# 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
```

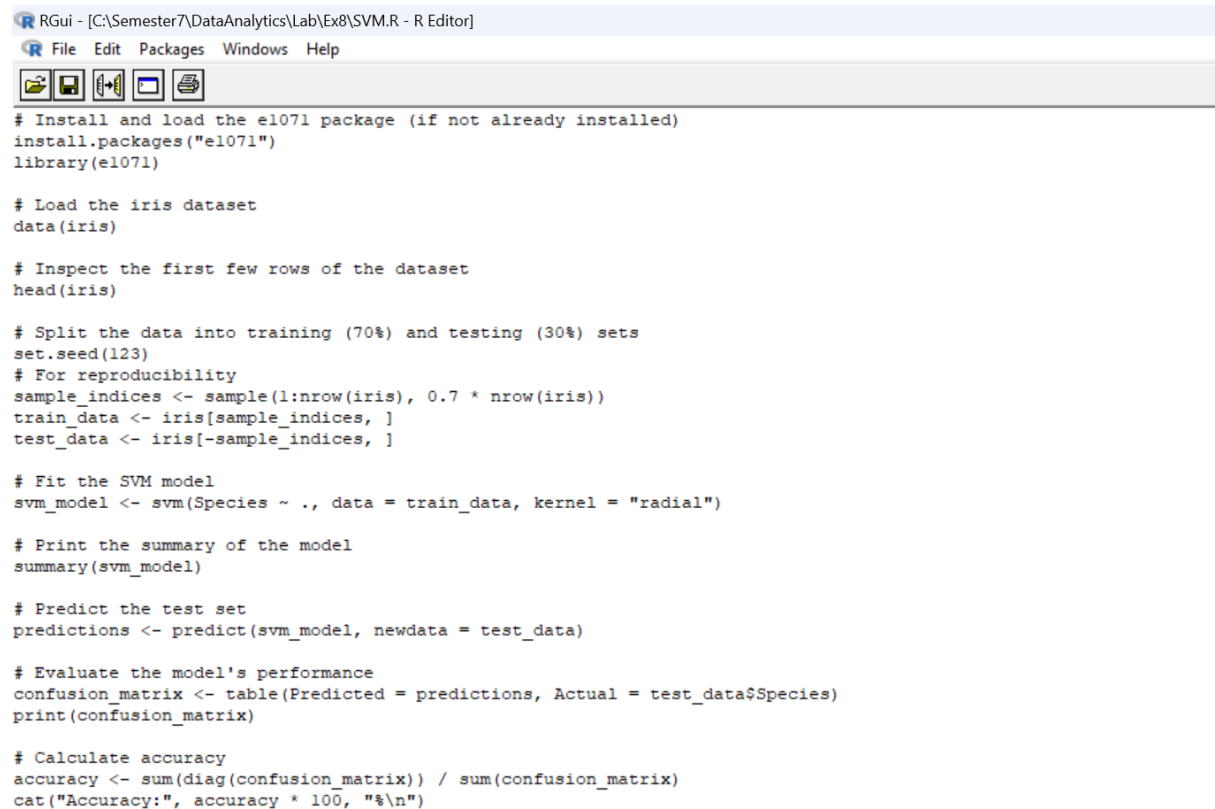
```
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
```

```
print(confusion_matrix)
```

```
# Calculate accuracy
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
```

```
cat("Accuracy:", accuracy * 100, "%\n")
```



The screenshot shows the RGui application window. The title bar reads "RGui - [C:\Semester7\DataAnalytics\Lab\Ex8\SVM.R - R Editor]". The menu bar includes "File", "Edit", "Packages", "Windows", and "Help". Below the menu bar is a toolbar with icons for file operations and execution. The main text area contains the following R code:

```
# 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:

```

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

Decision Tree:

PROGRAM:

```
# 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
```

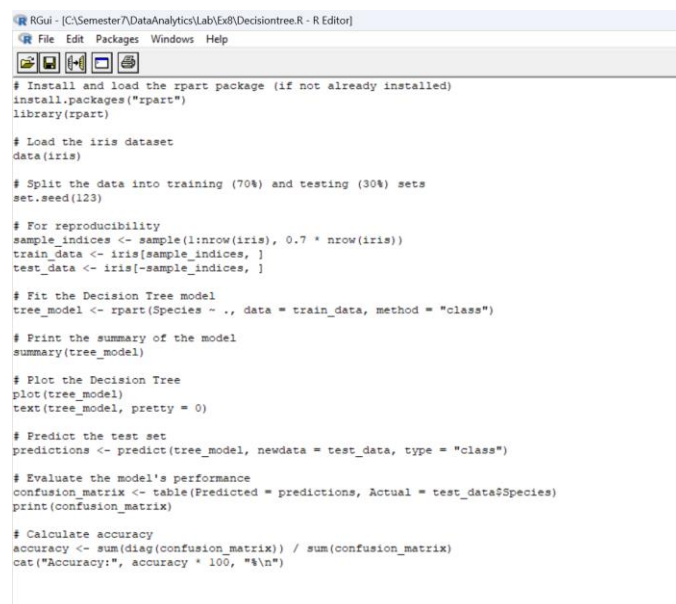
```
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
```

```
print(confusion_matrix)
```

```
# Calculate accuracy
```

```
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
```

```
cat("Accuracy:", accuracy * 100, "%\n")
```

A screenshot of an R GUI window titled "RGui - [C:\Semester7\DataAnalytics\Lab\Ex8\Decisiontree.R - R Editor]". The window has a menu bar with "File", "Edit", "Packages", "Windows", and "Help". Below the menu bar is a toolbar with icons for file operations and running code. The main text area contains R code that performs the following steps: 1. Install and load the 'rpart' package. 2. Load the 'iris' dataset. 3. Split the data into training (70%) and testing (30%) sets using 'set.seed(123)'. 4. Fit a decision tree model using 'rpart' with 'Species' as the response variable. 5. Print the summary of the model. 6. Plot the decision tree. 7. Predict the test set. 8. Evaluate the model's performance by creating a confusion matrix and printing it. 9. Calculate the accuracy and print it as a percentage. The code is as follows:

```
# 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")

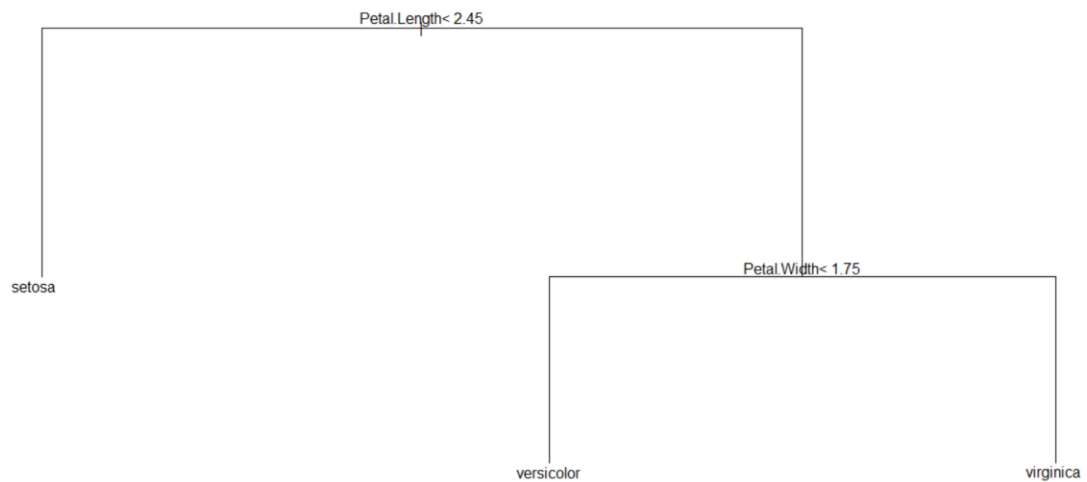
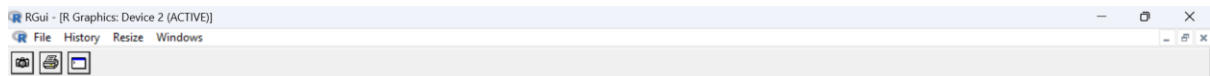
# Evaluate the model's performance
confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
print(confusion_matrix)

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

OUTPUT:

```

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



RESULT:

Thus, support vector machine and decision tree classification algorithms are successfully implemented using R.