

Implement SVM/Decision tree classification techniques

a) SVM IN 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")
```

The screenshot displays the RStudio interface. The top pane shows an R script with the following code:

```

1 # Load the iris dataset
2 data(iris)
3 # Inspect the first few rows of the dataset
4 head(iris)
5 # Split the data into training (70%) and testing (30%) sets
6 set.seed(123) # For reproducibility
7 sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
8 train_data <- iris[sample_indices, ]
9 test_data <- iris[-sample_indices, ]
10 # Fit the SVM model
11 svm_model <- svm(Species ~ ., data = train_data, kernel = "radial")
12 # Print the summary of the model
13 summary(svm_model)
14 # Predict the test set
15 predictions <- predict(svm_model, newdata = test_data)
16 # Evaluate the model's performance
17 confusion_matrix <- table(Predicted = predictions, Actual = test_data$Species)
18 print(confusion_matrix)
19 # Calculate accuracy
20 accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
21 cat("Accuracy:", accuracy * 100, "%\n")
22 |
23

```

The bottom pane shows the console output, which includes the installation of the 'e1071' package and the execution of the script:

```

R 4.4.1 ~ /
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.4/e1071_1.7-14.zip'
Content type 'application/zip' length 671816 bytes (656 KB)
downloaded 656 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\asus\AppData\Local\Temp\RtmpGnq5\downloaded_packages
      Actual
Predicted  setosa versicolor virginica
setosa      14          0          0
versicolor  0          17          0
virginica   0           1         13
Accuracy: 97.7778 %
> |

```

Below the console output, the same text is repeated, likely representing a second run or a continuation of the process:

```

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\asus\AppData\Local\Temp\RtmpGnq5\downloaded_packages
      Actual
Predicted  setosa versicolor virginica
setosa      14          0          0
versicolor  0          17          0
virginica   0           1         13
Accuracy: 97.7778 %

```

b) DECISION TREE IN R**CODE:**

```
# 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")

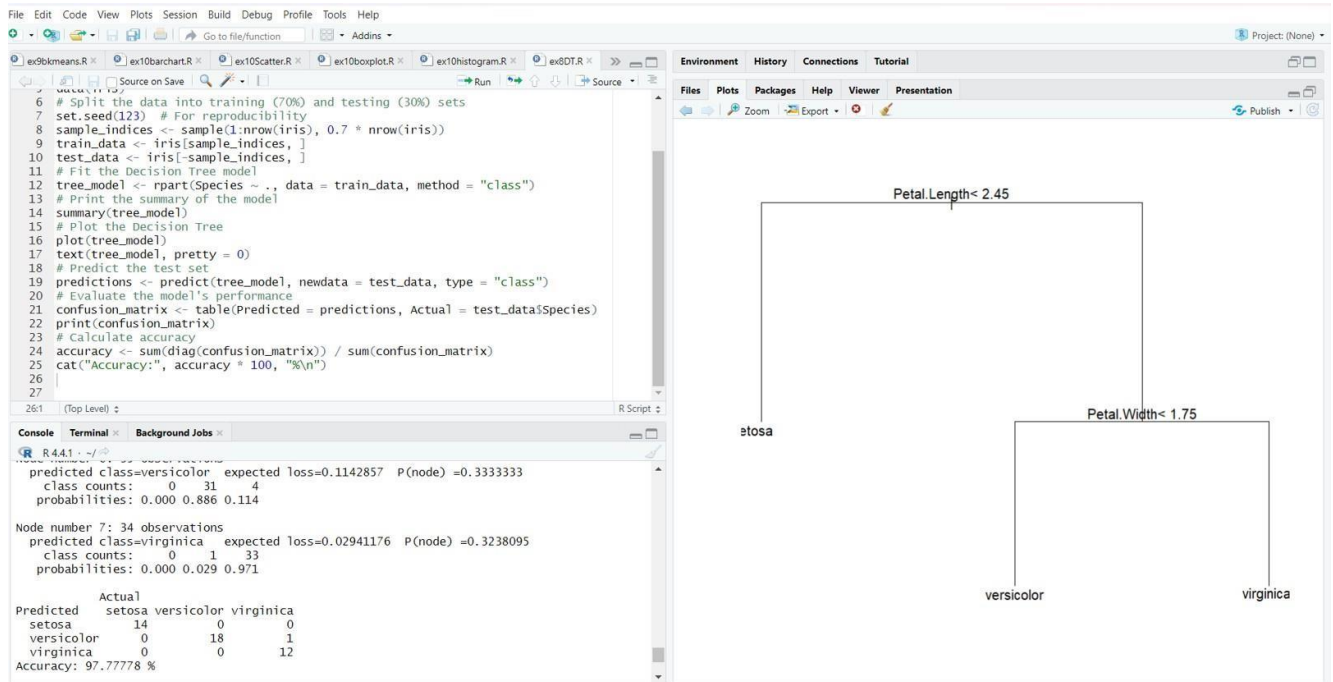
# Print the summary of the model
summary(tree_model)

# 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:**RESULT:**

SVM and Decision tree classification techniques are implemented Successfully.