

**Ex No 8****Implement SVM/Decision tree classification techniques****AIM:**

To Implement SVM/Decision tree classification techniques using R.

**PROCEDURE:**

- Collect and load the dataset from sources like CSV files or databases.
- Clean and preprocess the data, including handling missing values and encoding categorical variables.
- Split the dataset into training and testing sets to evaluate model performance.
- Normalize or standardize the features, especially for SVM, to ensure consistent scaling.
- Choose the appropriate model: SVM for margin-based classification, Decision Tree for rule-based classification.
- Train the model on the training data using the 'fit' method.
- Make predictions on the testing data using the 'predict' method.
- Evaluate the model using metrics like accuracy, confusion matrix, precision, and recall.
- Visualize the results with plots, such as decision boundaries for SVM or tree structures for Decision Trees.
- Fine-tune the model by adjusting hyperparameters like 'C' for SVM or 'max\_depth' for Decision Trees.

**CODE:****SVM.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")

```

### **Decision Tree.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:

SVM in R:

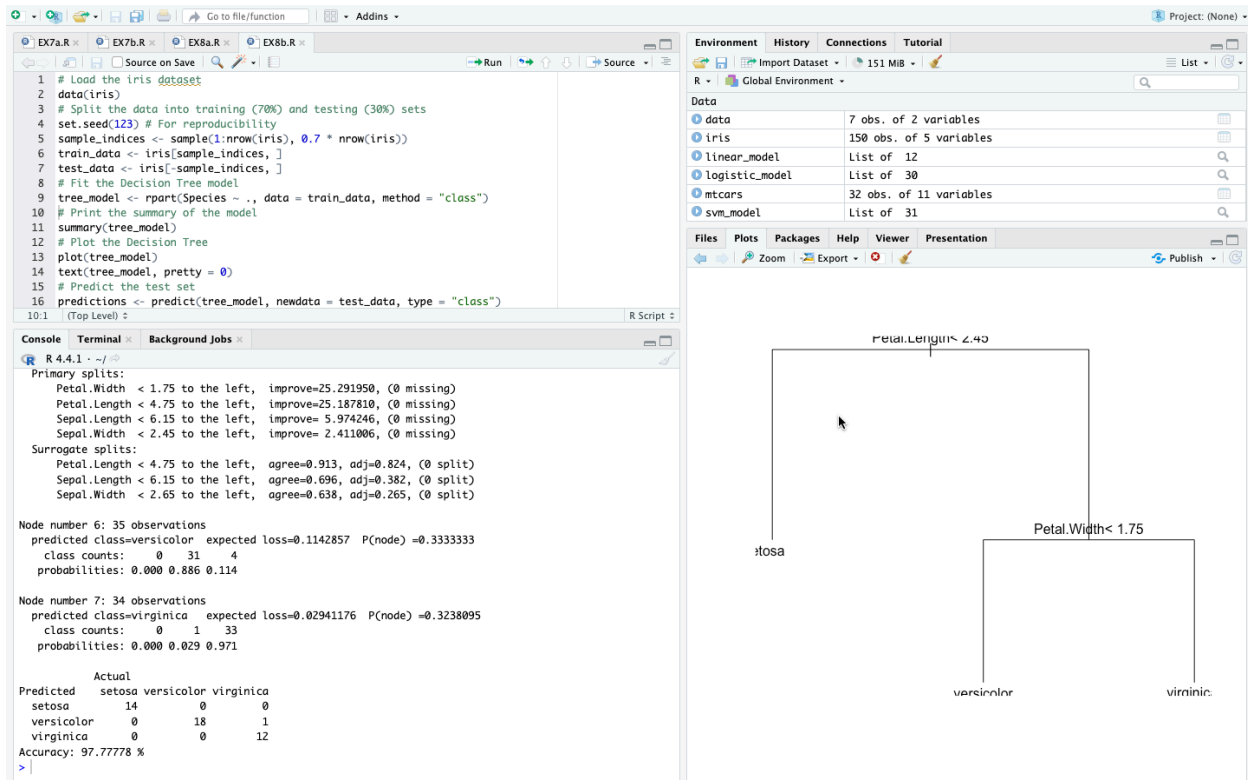
1 data(iris)  
2 # Inspect the first few rows of the dataset  
3 head(iris)  
4 # Split the data into training (70%) and testing (30%) sets  
5 set.seed(123) # For reproducibility  
6 sample\_indices <- sample(1:nrow(iris), 0.7 \* nrow(iris))  
7 train\_data <- iris[sample\_indices, ]  
8 test\_data <- iris[-sample\_indices, ]  
9 # Fit the SVM model  
10 svm\_model <- svm(Species ~ ., data = train\_data, kernel = "radial")  
11 # Print the summary of the model  
12 summary(svm\_model)  
13 # Predict the test set  
14 predictions <- predict(svm\_model, newdata = test\_data)  
15 # Evaluate the model's performance  
16 confusion\_matrix <- table(Predicted = predictions, Actual = test\_data\$Species)  
17  
18 (Top Level) :

Console  
Terminal  
Background Jobs

R 4.4.1 - ~/<br>Number of Fisher Scoring iterations: 5<br><br>Mazda RX4 Mazda RX4 Wag Datsun 710 Hornet 4 Drive<br>0.46109512 0.46109512 0.59789839 0.49171990<br>Hornet Sportabout Valiant Duster 360 Merc 240D<br>0.29690087 0.25993307 0.09858705 0.70846924<br>Merc 230 Merc 280 Merc 280C Merc 450SE<br>0.59789839 0.32991148 0.24260966 0.17246396<br>Merc 450SL Merc 450SLC Cadillac Fleetwood Lincoln Continental<br>0.21552479 0.12601104 0.03197098 0.03197098<br>Chrysler Imperial Fiat 128 Honda Civic Toyota Corolla<br>0.11005178 0.96591395 0.93878132 0.97821971<br>Toyota Corona Dodge Challenger AMC Javelin Camaro Z28<br>0.49939484 0.13650937 0.12601104 0.07446438<br>Pontiac Firebird Fiat X1-9 Porsche 914-2 Lotus Europa<br>0.32991148 0.85549212 0.79886349 0.93878132<br>Ford Pantera L Ferrari Dino Maserati Bora Volvo 142E<br>0.14773451 0.36468861 0.11940215 0.49171990<br><br>> train\_data <- iris[sample\_indices, ]<br>> source("~/EX8a.R")<br>Actual<br>Predicted setosa versicolor virginica<br>setosa 14 0 0<br>versicolor 0 17 0<br>virginica 0 1 13<br>Accuracy: 97.77778 %<br>> |

Environment History Connections Tutorial<br>R - Global Environment<br>Data<br>data 7 obs. of 2 variables<br>iris 150 obs. of 5 variables<br>linear\_model List of 12<br>logistic\_model List of 30<br>mtcars 32 obs. of 11 variables<br>svm\_model List of 31<br>test\_data 45 obs. of 5 variables<br>train\_data 105 obs. of 5 variables<br>tree\_model List of 14<br>Values<br>accuracy 0.977777777777778<br>confusion\_matrix 'table' int [1:3, 1:3] 14 0 0 0 17 1 0 0 13<br>heights num [1:7] 150 160 165 170 175 180 185<br>predicted\_probs Named num [1:32] 0.461 0.461 0.598 0.492 0.297 ...<br>predictions Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 ...<br>sample\_indices int [1:105] 14 50 118 43 150 148 90 91 143 92 ...<br>weights num [1:7] 55 60 62 68 70 75 80

## Decision tree:



## RESULT:

Thus, Implement SVM and Decision tree classification techniques has been successfully executed.