

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.

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:

Environment History Connections Tutorial

Import Dataset 145 MiB

R - Global Environment

Data

Object	Description
data	7 obs. of 2 variables
iris	150 obs. of 5 variables
linear_model	List of 12
logistic_model	List of 30
mtcars	32 obs. of 11 variables
svm_model	List of 31
test_data	45 obs. of 5 variables
train_data	105 obs. of 5 variables
tree_model	List of 14

Values

accuracy	0.977777777777778
confusion_matrix	'table' int [1:3, 1:3] 14 0 0 0 17 1 0 0 13
heights	num [1:7] 150 160 165 170 175 180 185
predicted_probs	Named num [1:32] 0.461 0.461 0.598 0.492 0.297 ...
predictions	Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 ...
sample_indices	int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
weights	num [1:7] 55 60 62 68 70 75 80

```

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
8.1 (Top Level)

```

Console Terminal Background Jobs

R 4.4.1 - ~/

Number of Fisher Scoring iterations: 5

Model	Accuracy
Mazda RX4	0.46109512
Mazda RX4 Wag	0.46109512
Datsun 710	0.59789839
Hornet 4 Drive	0.49171990
Hornet Sportabout	0.29690087
Merc 230	0.59789839
Merc 240	0.59789839
Merc 450SE	0.21552479
Merc 450SL	0.12601104
Cadillac Fleetwood	0.03197098
Lincoln Continental	0.03197098
Chrysler Imperial	0.11005178
Fiat 128	0.96591395
Honda Civic	0.93878132
Toyota Corolla	0.97821971
Toyota Corona	0.49939484
Dodge Challenger	0.13650937
AMC Javelin	0.12601104
Camaro Z28	0.07446438
Pontiac Firebird	0.32991148
Fiat X1-9	0.85549212
Porsche 914-2	0.79886349
Lotus Europa	0.93878132
Ford Pantera L	0.14773451
Ferrari Dino	0.36468861
Maserati Bora	0.11940215
Volvo 142E	0.49171990

Predicted Actual

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

Accuracy: 97.77778 %

Files Plots Packages Help Viewer Presentation

Decision tree:

Environment History Connections Tutorial

Import Dataset 151 MiB

R - Global Environment

Data

Object	Description
data	7 obs. of 2 variables
iris	150 obs. of 5 variables
linear_model	List of 12
logistic_model	List of 30
mtcars	32 obs. of 11 variables
svm_model	List of 31

Files Plots Packages Help Viewer Presentation

Zoom Export Publish

```

1 # Load the Iris dataset
2 data(iris)
3 # Split the data into training (70%) and testing (30%) sets
4 set.seed(123) # For reproducibility
5 sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
6 train_data <- iris[sample_indices, ]
7 test_data <- iris[-sample_indices, ]
8 # Fit the Decision Tree model
9 tree_model <- rpart(Species ~ ., data = train_data, method = "class")
10 # Print the summary of the model
11 summary(tree_model)
12 # Plot the Decision Tree
13 plot(tree_model)
14 text(tree_model, pretty = 0)
15 # Predict the test set
16 predictions <- predict(tree_model, newdata = test_data, type = "class")
17
10.1 (Top Level)

```

Console Terminal Background Jobs

R 4.4.1 - ~/

Primary splits:

- Petal.Width < 1.75 to the left, improve=25.291950, (0 missing)
- Petal.Length < 4.75 to the left, improve=25.187810, (0 missing)
- Sepal.Length < 6.15 to the left, improve= 5.974246, (0 missing)
- Sepal.Width < 2.45 to the left, improve= 2.411006, (0 missing)

Surrogate splits:

- Petal.Length < 4.75 to the left, agree=0.913, adj=0.824, (0 split)
- Sepal.Length < 6.15 to the left, agree=0.696, adj=0.382, (0 split)
- Sepal.Width < 2.65 to the left, agree=0.638, adj=0.265, (0 split)

Node number 6: 35 observations
 predicted class=versicolor expected loss=0.1142857 P(node)=0.3333333
 class counts: 0 31 4
 probabilities: 0.000 0.886 0.114

Node number 7: 34 observations
 predicted class=virginica expected loss=0.02941176 P(node)=0.3238095
 class counts: 0 1 33
 probabilities: 0.000 0.029 0.971

Actual

Predicted \ Actual	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	18	1
virginica	0	0	12

Accuracy: 97.77778 %

>

```

graph TD
    Root["Petal.Length < 4.75"]
    Root --> Setosa["setosa"]
    Root --> Node6["Petal.Width < 1.75"]
    Node6 --> Versicolor["versicolor"]
    Node6 --> Virginica["virginica"]

```

RESULT:

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