Ex no: 8

Implement SVM/Decision tree classification techniques

Aim:

To implement SVM/Decision tree classification techniques in R programming.

Procedure:

a) SVM:

1. Install and load the e1071 package for SVM functionality.
2. Load the iris dataset into the environment.
3. Inspect the first few rows of the dataset to understand its structure.
4. Split the dataset into training (70%) and testing (30%) sets using random sampling.
5. Set a random seed for reproducibility of the data split.
6. Fit the Support Vector Machine (SVM) model to the training data using the `Species` variable as the target and all other features as predictors, with a radial kernel.
7. Print the summary of the fitted SVM model to view model parameters and support vectors.
8. Predict the species of the test data using the trained SVM model.
9. Create a confusion matrix by comparing the predicted and actual species values in the test set.
10. Calculate and display the accuracy of the SVM model by dividing the sum of correctly predicted instances by the total number of instances in the test set.

b) Decision Tree:

1. Install and load the rpart package for decision tree functionality.

2. Load the iris dataset into the environment.

3. Split the dataset into training (70%) and testing (30%) sets using random sampling.

4. Set a random seed to ensure reproducibility of the data split.

5. Fit a decision tree model to the training data using the `Species` variable as the target and all other features as predictors.

6. Print the summary of the decision tree model to view its structure and performance.

7. Visualize the fitted decision tree using `plot()` and label the nodes with `text()`.

8. Predict the species of the test data using the trained decision tree model.

9. Create a confusion matrix by comparing the predicted and actual species values in the test set.

10. Calculate the model's accuracy by dividing the sum of correct predictions by the total number of predictions and displaying the result as a percentage.

Program:

a) SVM:

```
# 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")
```

b) Decision Tree:

```
# Load the dataset
data(mtcars)
# Convert 'am' to a factor (categorical variable)
mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manual"))
# Fit a logistic regression model
logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)
# Print the summary of the model
print(summary(logistic_model))
# Predict probabilities for the logistic model
predicted_probs <- predict(logistic_model, type = "response")
# Display the predicted probabilities
print(predicted_probs)
# Plotting the data and logistic regression curve
plot(mtcars$mpg, as.numeric(mtcars$am) - 1,
    main = "Logistic Regression: Transmission vs. MPG",
    xlab = "Miles Per Gallon (mpg)",
    ylab = "Probability of Manual Transmission",
    pch = 19, col = "blue")
# Add the logistic regression curve
curve(predict(logistic_model, data.frame(mpg = x), type = "response"),
     add = TRUE, col = "red", lwd = 2)
```

Output:

a) SVM:

Console Terminal Background Jobs

R 4.4.1 · ~/

```
> # Load the iris dataset
> data(iris)

> # Inspect the first few rows of the dataset
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa

> # 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)

Call:
svm(formula = Species ~ ., data = train_data, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  45

 ( 7 18 20 )


Number of Classes:  3

Levels:
 setosa versicolor virginica
```

```
> # 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)
            Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13

> # Calculate accuracy
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)

> cat("Accuracy:", accuracy * 100, "%\n")
Accuracy: 97.77778 %
```

Environment | History | Connections | Tutorial

Import Dataset | 196 MiB | List

R | Global Environment

Data	
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
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…
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

b) Decision Tree:

Console Terminal Background Jobs

R 4.4.1 · ~/

```
> 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)
Call:
rpart(formula = Species ~ ., data = train_data, method = "class")
  n= 105

          CP nsplit  rel error    xerror       xstd
1 0.5294118      0 1.00000000 1.2058824 0.06232572
2 0.3970588      1 0.47058824 0.5441176 0.07198662
3 0.0100000      2 0.07352941 0.1176471 0.03997857

Variable importance
 Petal.Width Petal.Length Sepal.Length  Sepal.Width
          34           32           21           13
```

```
Node number 1: 105 observations,    complexity param=0.5294118
  predicted class=virginica   expected loss=0.647619  P(node) =1
    class counts:    36    32    37
   probabilities: 0.343 0.305 0.352
  left son=2 (36 obs) right son=3 (69 obs)
  Primary splits:
      Petal.Length < 2.45 to the left,  improve=35.54783, (0 missing)
      Petal.Width  < 0.8  to the left,  improve=35.54783, (0 missing)
      Sepal.Length < 5.45 to the left,  improve=24.79179, (0 missing)
      Sepal.Width  < 3.25 to the right, improve=12.34670, (0 missing)
  Surrogate splits:
      Petal.Width  < 0.8  to the left,  agree=1.000, adj=1.000, (0 split)
      Sepal.Length < 5.45 to the left,  agree=0.924, adj=0.778, (0 split)
      Sepal.Width  < 3.25 to the right, agree=0.819, adj=0.472, (0 split)

Node number 2: 36 observations
  predicted class=setosa      expected loss=0  P(node) =0.3428571
    class counts:    36     0     0
   probabilities: 1.000 0.000 0.000
```

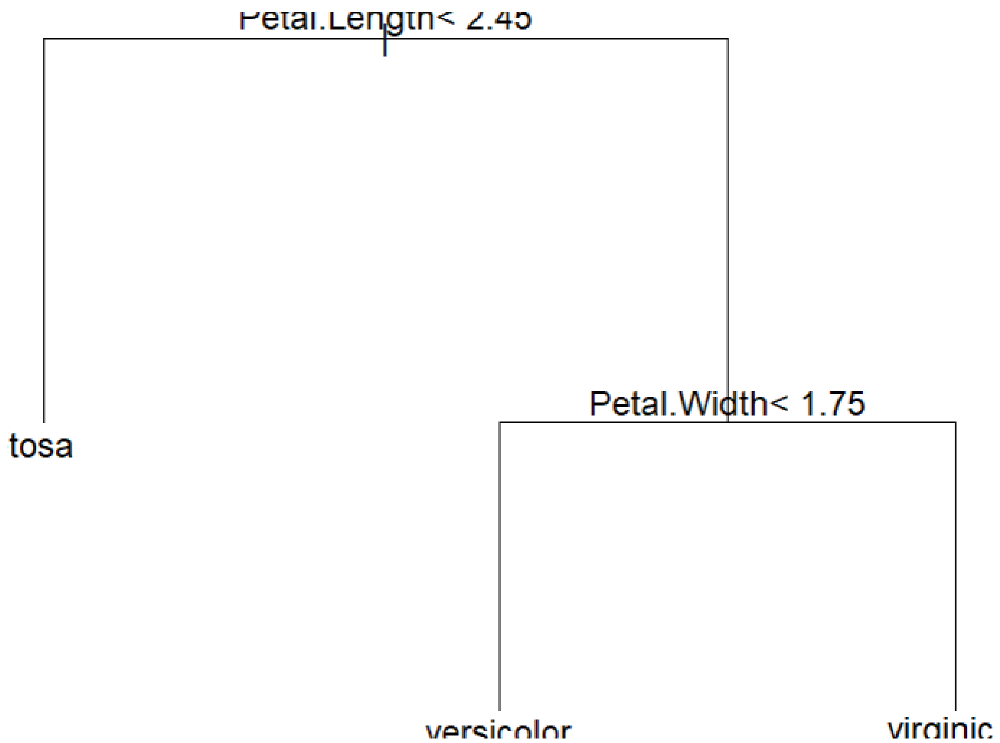
```
Node number 3: 69 observations,    complexity param=0.3970588
  predicted class=virginica   expected loss=0.4637681  P(node) =0.6571429
    class counts:     0    32    37
   probabilities: 0.000 0.464 0.536
  left son=6 (35 obs) right son=7 (34 obs)
  Primary splits:
      Petal.Width  < 1.75 to 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


> # Plot the Decision Tree
> plot(tree_model)
```

Environment | History | Connections | Tutorial

Import Dataset | 225 MiB | List

R | Global Environment

Data	
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 18 0 0 1 12
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…
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



Result:

Thus the implementation of SVM/Decision tree classification techniques using R programming has been executed successfully.