Experiment: 3.1

Aim: Implementation of Classification algorithm in R Programming.

Software Required: R Programming, VS Code

Description: The experiment involves using R programming to implement a classification algorithm for business intelligence purposes. Participants will learn how to preprocess data, train a classification model, evaluate its performance, and use it for predicting future outcomes.

Steps:

1.Import and Preprocess Data:

a. Launch R programming environment.

b. Load the required libraries and import the dataset for classification.

c. Perform data preprocessing tasks such as handling missing values, data transformation, and feature scaling.

2.Split the Dataset:

a. Split the dataset into a training set and a testing/validation set.

b. Allocate a significant portion of the data for training the classification model.

3.Train the Classification Model:

a. Select an appropriate classification algorithm, such as Decision Trees, Naive Bayes, or Random Forests.

b. Use the training set to train the classification model.

c. Configure and fine-tune the algorithm's parameters to optimize performance.

4.Evaluate Model Performance:

a. Apply the trained model to the testing/validation set.

b. Calculate evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's performance.

c. Use visualizations like confusion matrices and ROC curves to analyze the model's predictive power.

5.Make Predictions:

a. Use the trained classification model to make predictions on new, unseen data.

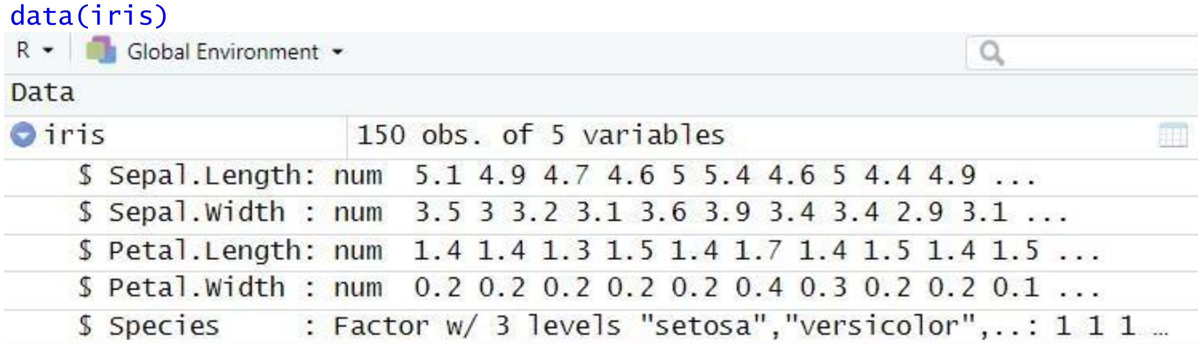
b. Apply the preprocessing steps performed earlier on the new data.

c. Generate predictions and interpret the results.

Outcome:

Step 1: Data Preparation

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



Check the structure and summary of the dataset

```
> str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> summary(iris)
```

```
  Sepal.Length    Sepal.Width     Petal.Length
 Min.   :4.300   Min.   :2.000   Min.   :1.000
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
 Median :5.800   Median :3.000   Median :4.350
 Mean   :5.843   Mean   :3.057   Mean   :3.758
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
 Max.   :7.900   Max.   :4.400   Max.   :6.900
  Petal.Width          Species
 Min.   :0.100   setosa    :50
 1st Qu.:0.300   versicolor:50
 Median :1.300   virginica :50
 Mean   :1.199
 3rd Qu.:1.800
 Max.   :2.500
```

Step 2: Data Splitting Split the dataset into a training set (70%) and a testing set (30%)

```
> set.seed(123)  # For reproducibility
> sample_indices <- sample(1:nrow(iris), 0.7 * nrow(iris))
> training_data <- iris[sample_indices, ]
> testing_data <- iris[-sample_indices, ]
```

```
testing_data              45 obs. of 5 variables
    $ Sepal.Length: num  5.1 4.9 4.7 5 5.4 5.1 5.7 5.2 5.2 5.2 ...
    $ Sepal.Width : num  3.5 3 3.2 3.6 3.7 3.5 3.8 3.5 3.4 4.1 ...
    $ Petal.Length: num  1.4 1.4 1.3 1.4 1.5 1.4 1.7 1.5 1.4 1.5 ...
    $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.3 0.3 0.2 0.2 0.1 ...
    $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
training_data          105 obs. of 5 variables
  $ Sepal.Length: num  4.3 5 7.7 4.4 5.9 6.5 5.5 5.5 5.8 6.1 ...
  $ Sepal.Width : num  3 3.3 3.8 3.2 3 3 2.5 2.6 2.7 3 ...
  $ Petal.Length: num  1.1 1.4 6.7 1.3 5.1 5.2 4 4.4 5.1 4.6 ...
  $ Petal.Width : num  0.1 0.2 2.2 0.2 1.8 2 1.3 1.2 1.9 1.4 ...
  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 3 1 3 3 2 2 3 2 ...
Values
 sample_indices          int [1:105] 14 50 118 43 150 148 90 91 143 92 ...
```

Step 3: Model Building Train a logistic regression model Train a multiclass logistic regression model

```
> model <- multinom(Species ~ ., data = training_data)
```

```
# weights:  18 (10 variable)
initial  value 115.354290
iter  10 value 14.037979
iter  20 value 3.342288
iter  30 value 2.503699
iter  40 value 2.171547
iter  50 value 2.099460
iter  60 value 1.828506
iter  70 value 0.904367
iter  80 value 0.669147
iter  90 value 0.622003
iter 100 value 0.609416
final  value 0.609416
stopped after 100 iterations
```

Step 4: Model Evaluation Make predictions for the testing data

```
predicted_classes <- predict(model, newdata = testing_data, type = "class")
```

Create a confusion matrix to evaluate the model

```
> confusion_matrix <- table(observed = testing_data$Species, predicted = predicted_
classes)
> print(confusion_matrix)
```

	predicted		
observed	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	1
virginica	0	0	13

Calculate accuracy

```
> accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix) > cat("Accuracy:", accuracy, "\n")
```

```
Accuracy: 0.9777778
```

Values	
accuracy	0.977777777777778
best_lambda	0.00182026400656745
confusion_matrix	'table' int [1:3, 1:3] 14 0 0 0 17 0 0 1 13
predicted_classes	Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 …
predictions	num [1:45, 1:3, 1] 0.998 0.994 0.998 0.999 0.998 ...
sample_indices	int [1:105] 14 50 118 43 150 148 90 91 143 92 ...

Learning Outcomes

1. Gain proficiency in using R programming for implementing classification algorithms.
2. Understand the importance of data preprocessing in preparing data for classification tasks.
3. Learn different classification algorithms and their application in business intelligence.
4. Develop skills in training and fine-tuning classification models using R.
5. Acquire knowledge of evaluating classification model performance using various metrics and visualizations.
6. Learn how to use trained classification models to make predictions on new data.