Experiment 2.2

1. Aim:

To perform classification using Bayesian classification algorithm using R.

2. Objective:

To use Naïve bayes Algorithm to perform effective classification in R.

3. Script and Output:

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- It is mainly used in text classification that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

- The formula for Bayes' theorem is given as:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

CODE-

```
# Using Naive Bayes algorithm
#Functions for latent class analysis, support vector machines,
#shortest path computation, bagged clustering, naive Bayes classifier

 install.packages("e1071")

#Contains several basic utility functions including: moving (rolling, running) window statistic functions, read/write

install.packages("caTools")

#Misc functions for training and plotting classification and regression models.

install.packages("caret")

# Installing library for the given experiment

library(e1071)

library(caTools)

 library(caret)

library(rpart)

library(rpart.plot)

orange_data = Orange

str(orange_data)

summary(orange_data)

# Splitting data into train and test data

spl = sample.split(orange_data, SplitRatio = 0.7)

dataTrain = subset(orange_data, spl==TRUE)

dataTest = subset(orange_data, spl==FALSE)

dataTrain

dataTest


# Feature Scaling

train_scale <- scale(dataTrain[, 2:3])

test_scale <- scale(dataTest[, 2:3])
```

 # Fitting Naive Bayes Model to training dataset
#The use of set.seed is to make sure that we get the same results for randomization.

set.seed(120)

#If we randomly select some observations for any task in R or in
#any statistical software it results
#in different values all the time and this happens because of randomization

classifier_cl <- naiveBayes(age ~ ., data = dataTrain)

classifier_cl

 # Predicting on test data'

y_pred <- predict(classifier_cl, newdata = dataTest)

 # summarize accuracy

table_matrix <- table(dataTest$age, y_pred)

print(table_matrix)

#table(predictions, orange$age)

accuracy_Test <- sum(diag(table_matrix)) / sum(table_matrix)

cat("Test Accuracy is: ", accuracy_Test)

 # Confusion Matrix

cm <- table(dataTest$age, y_pred)

cm

 # Model Evaluation

confusionMatrix(cm)

```
# Structure
str(Orange)
#Performing Naive Bayes on Dataset
#Using Naive Bayes algorithm on the dataset which includes 11 persons and 6 varia
# Installing Packages
install.packages("e1071")
install.packages("caTools")
install.packages("caret")
install.packages("RWeka")

# Loading package
library(e1071)
library(caTools)
library(caret)

library(RWeka)
library(rpart)
library(rpart.plot)
library(partykit)

orange_data = Orange
str(orange_data)
summary(orange_data)
```

Output:

```
> orange_data = Orange
> str(orange_data)
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':      35 obs.
of  3 variables:
 $ Tree         : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<..: 2 2 2 2 2 2 2 4 4 4 ...
 $ age          : num  118 484 664 1004 1231 ...
 $ circumference: num  30 58 87 115 120 142 145 33 69 111 ...
 - attr(*, "formula")=Class 'formula'  language circumference ~ age | Tree
  .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
  ..$ x: chr "Time since December 31, 1968"
  ..$ y: chr "Trunk circumference"
 - attr(*, "units")=List of 2
  ..$ x: chr "(days)"
  ..$ y: chr "(mm)"
> summary(orange_data)
 Tree       age          circumference
 3:7   Min.   : 118.0   Min.   : 30.0
 1:7   1st Qu.: 484.0   1st Qu.: 65.5
 5:7   Median :1004.0   Median :115.0
 2:7   Mean   : 922.1   Mean   :115.9
 4:7   3rd Qu.:1372.0   3rd Qu.:161.5
       Max.   :1582.0   Max.   :214.0
```

Code:

```
# Splitting data into train and test data
spl = sample.split(orange_data, SplitRatio = 0.7)
dataTrain = subset(orange_data, spl==TRUE)
dataTest = subset(orange_data, spl==FALSE)
dataTrain
dataTest

# Feature Scaling
train_scale <- scale(dataTrain[, 2:3])
test_scale <- scale(dataTest[, 2:3])
```

Output:

```
> spl = sample.split(orange_data, SplitRatio = 0.7)
> dataTrain = subset(orange_data, spl==TRUE)
> dataTest = subset(orange_data, spl==FALSE)
> dataTrain
Grouped Data: circumference ~ age | Tree
   Tree  age circumference
2     1  484            58
3     1  664            87
5     1 1231           120
6     1 1372           142
8     2  118            33
9     2  484            69
11    2 1004           156
12    2 1231           172
```

Code:

```
# Feature Scaling
train_scale <- scale(dataTrain[, 2:3])
test_scale <- scale(dataTest[, 2:3])

# Fitting Naive Bayes Model to training dataset
set.seed(120)          # Setting Seed
classifier_cl <- naiveBayes(age ~ ., data = dataTrain)
classifier_cl
```

Output:

```
> train_scale <- scale(dataTrain[, 2:3])
> test_scale <- scale(dataTest[, 2:3])
> # Fitting Naive Bayes Model to training dataset
> set.seed(120)        # Setting Seed
> classifier_cl <- naiveBayes(age ~ ., data = dataTrain)
> classifier_cl

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
      118       484       664      1004      1231      1372      1582 
0.1304348 0.1739130 0.1304348 0.1304348 0.1739130 0.1304348 0.1304348 

Conditional probabilities:
      Tree
Y              3         1         5         2         4
  118  0.3333333 0.0000000 0.3333333 0.3333333 0.0000000
  484  0.0000000 0.2500000 0.2500000 0.2500000 0.2500000
  664  0.3333333 0.3333333 0.0000000 0.0000000 0.3333333
  1004 0.3333333 0.0000000 0.3333333 0.3333333 0.0000000
  1231 0.0000000 0.2500000 0.2500000 0.2500000 0.2500000
  1372 0.3333333 0.3333333 0.0000000 0.0000000 0.3333333
  1582 0.3333333 0.0000000 0.3333333 0.3333333 0.0000000

      circumference
Y          [,1]      [,2]
  118  31.00000  1.732051
  484  59.50000  8.346656
```

Code:

```
# Predicting on test data'
y_pred <- predict(classifier_cl, newdata = dataTest)

# summarize accuracy
table_matrix <- table(dataTest$age, y_pred)
print(table_matrix)
#table(predictions, iris$Species)
accuracy_Test <- sum(diag(table_matrix)) / sum(table_matrix)
cat("Test Accuracy is: ", accuracy_Test)

# Confusion Matrix
cm <- table(dataTest$age, y_pred)
cm
# Model Evaluation
confusionMatrix(cm)
```

Output:

```
> # Predicting on test data'
> y_pred <- predict(classifier_cl, newdata = dataTest)
> # summarize accuracy
> table_matrix <- table(dataTest$age, y_pred)
> print(table_matrix)
      y_pred
       118 484 664 1004 1231 1372 1582
  118    2   0   0    0    0    0    0
  484    0   0   1    0    0    0    0
  664    0   0   0    2    0    0    0
  1004   0   0   1    0    1    0    0
  1231   0   0   0    1    0    0    0
  1372   0   0   0    0    0    0    2
  1582   0   0   0    0    1    1    0
> #table(predictions, iris$Species)
> accuracy_Test <- sum(diag(table_matrix)) / sum(table_matrix)
> cat("Test Accuracy is: ", accuracy_Test)
Test Accuracy is:  0.1666667
```