1 Implementation of R program – basic as Arithmetic operations

```
vec1 <- c(0, 2)
vec2 <- c(2, 3)
cat ("Addition of vectors :", vec1 + vec2, "\n")
cat ("Subtraction of vectors :", vec1 - vec2, "\n")
cat ("Multiplication of vectors :", vec1 * vec2, "\n")
cat ("Division of vectors :", vec1 / vec2, "\n")
cat ("Modulo of vectors :", vec1 %% vec2, "\n")
cat ("Power operator :", vec1 ^ vec2)
```

Output

```
Addition of vectors : 2 5
Subtraction of vectors : -2 -1
Multiplication of vectors : 0 6
Division of vectors : 0 0.6666667
Modulo of vectors : 0 2
Power operator : 0 8
```

2 Implementation of R program – basic as Relational Operation

```
vec1 <- c(0, 2)
vec2 <- c(2, 3)
cat ("Vector1 less than Vector2 :", vec1 < vec2, "\n")
cat ("Vector1 less than equal to Vector2 :", vec1 <= vec2, "\n")
cat ("Vector1 greater than Vector2 :", vec1 > vec2, "\n")
cat ("Vector1 greater than equal to Vector2 :", vec1 >= vec2, "\n")
cat ("Vector1 not equal to Vector2 :", vec1 != vec2, "\n")
```

Output

```
Vector1 less than Vector2 : TRUE TRUE
Vector1 less than equal to Vector2 : TRUE TRUE
Vector1 greater than Vector2 : FALSE FALSE
Vector1 greater than equal to Vector2 : FALSE FALSE
Vector1 not equal to Vector2 : TRUE TRUE
```

3 Implementation of data types in R

```
x = 5.6
print(class(x))
y = 5L
print(class(y))
x = 4
y = 3
z = x > y
print(z)
print(class(z))
x = 4 + 3i
print(class(x))
char = "DATA SCIENCE"
```

```
    print(char)
    print(class(char))
```

Output

```
[1] "numeric"
[1] "integer"
[1] TRUE
[1] "logical"
[1] "complex"
[1] "DATA SCIENCE"
[1] "character"
```

4 control statements

```
x <- 100
if(x > 10){
print(paste(x, "is greater than 10"))
}
x <- 5
if(x > 10){
print(paste(x, "is greater than 10"))
}else{
print(paste(x, "is less than 10"))
}
x <- 0
if (x > 0) {
  print("x is a positive number")
} else if (x < 0) {
  print("x is a negative number")
} else {
  print("x is zero")
}
```

Output

```
[1] "100 is greater than 10"
[1] "5 is less than 10"
[1] "x is zero"
```

5 Looping statements

```
fruits <- list("apple", "banana", "cherry")
for (x in fruits) {
  print(x)
}
val = 1
while (val <= 5)
{
   # statements
   print(val)
```

```
    val = val + 1
}
val = 1
repeat
{
print(val)
val = val + 1
if(val > 5)
{
break
} }
```

Output

```
[1] "apple"
[1] "banana"
[1] "cherry"
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

6 **write R program to perform decision tree**

```
# Load required packages
library(rpart)
library(rpart.plot)
# Load dataset (Iris dataset in this example)
data(iris)
# Split data into training and testing sets
trainIndex <- sample(1:nrow(iris), 0.7*nrow(iris))
train <- iris[trainIndex, ]
test <- iris[-trainIndex, ]
# Train decision tree model
treeModel <- rpart(Species ~ ., data=train, method="class")
# Visualize decision tree
rpart.plot(treeModel)
# Make predictions on test set
predictions <- predict(treeModel, test, type="class")
# Evaluate accuracy of predictions
accuracy <- sum(predictions == test$Species) / nrow(test)
cat ("Accuracy:", accuracy)
```

Output

```
Accuracy: 0.9333333
```

7 **write R program to perform native bayes**

```
# Load the e1071 package for Naive Bayes classifier
library(e1071)
# Load the iris dataset
data(iris)
# Split the data into training and testing sets
trainIndex <- sample(1:nrow(iris), 100)
trainData <- iris[trainIndex, ]
testData <- iris[-trainIndex, ]
# Train a Naive Bayes classifier using the training data
nb_model <- naiveBayes(Species ~ ., data = trainData)
# Predict the class labels for the test data using the trained model
nb_pred <- predict(nb_model, testData)
# Calculate the accuracy of the model
nb_accuracy <- sum(nb_pred == testData$Species) / nrow(testData)
# Print the accuracy of the model
cat("Naive Bayes accuracy:", nb_accuracy, "\n")
```

Output

Naive Bayes accuracy: 0.94

8 K means clustering

```
install.packages("ClusterR")

install.packages("cluster")

library(ClusterR)

library(cluster)

iris_1 <- iris[, -5]

set.seed(240) # Setting seed

kmeans.re <- kmeans(iris_1, centers = 3, nstart = 20)

kmeans.re

kmeans.re$cluster

cm <- table(iris$Species, kmeans.re$cluster)

cm

plot(iris_1[c("Sepal.Length", "Sepal.Width")])

plot(iris_1[c("Sepal.Length", "Sepal.Width")],

        col = kmeans.re$cluster)

plot(iris_1[c("Sepal.Length", "Sepal.Width")],
```

```r
           col = kmeans.re$cluster,
           main = "K-means with 3 clusters")
kmeans.re$centers
kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")]
points(kmeans.re$centers[, c("Sepal.Length", "Sepal.Width")],
col = 1:3, pch = 8, cex = 3)
y_kmeans <- kmeans.re$cluster
clusplot(iris_1[, c("Sepal.Length", "Sepal.Width")],
y_kmeans,lines = 0,shade = TRUE,color = TRUE,labels = 2,plotchar = FALSE,
    span = TRUE,main = paste("Cluster iris"),
               xlab = 'Sepal.Length',
               ylab = 'Sepal.Width')
```

output

9 write a R program to perform kNN

```r
# Load necessary packages
library(class)
# Load dataset
data(iris)
# Split dataset into training and testing set
set.seed(123)
train.index <- sample(1:nrow(iris), 0.7*nrow(iris))
train.data <- iris[train.index, 1:4]
train.labels <- iris[train.index, 5]
test.data <- iris[-train.index, 1:4]
test.labels <- iris[-train.index, 5]
# Run kNN algorithm
k <- 3 # set k value
pred <- knn(train = train.data, test = test.data, cl = train.labels, k = k)
# Compute accuracy
accuracy <- sum(pred == test.labels)/length(test.labels)
print(paste0("Accuracy: ", accuracy))
```

output

[1] "Accuracy: 0.977777777777778"

10 Random Forest

```
# Load the randomForest package
library(randomForest)
# Load the dataset
data(iris)
# Split the data into training and testing sets
set.seed(123)
train_idx <- sample(nrow(iris), 0.7 * nrow(iris))
train_data <- iris[train_idx, ]
test_data <- iris[-train_idx, ]
# Train the model using the randomForest function
rf_model <- randomForest(Species ~ ., data = train_data, ntree = 500)
# Make predictions on the testing set
predictions <- predict(rf_model, test_data)
# Print the confusion matrix and accuracy
table(predictions, test_data$Species)
accuracy <- mean(predictions == test_data$Species)
print(paste0("Accuracy: ", accuracy))
```

11 Charts in R

```
# Load built-in dataset
data(iris)
# Scatter plot
plot(iris$Sepal.Length, iris$Sepal.Width, xlab="Sepal Length", ylab="Sepal Width", main="Scatter Plot of Sepal Length vs. Sepal Width")
# Histogram
hist(iris$Sepal.Length, breaks=10, col="blue", xlab="Sepal Length", main="Histogram of Sepal Length")
# Boxplot
boxplot(iris$Sepal.Length, main="Boxplot of Sepal Length", ylab="Sepal Length")
# Bar chart
```

```
barplot(table(iris$Species), main="Bar Chart of Iris Species", xlab="Species", ylab="Count")

# Line chart

plot(1:10, type="n", xlab="X-axis", ylab="Y-axis", main="Line Chart")

for (i in 1:3) {

  lines(1:10, rnorm(10)+i, col=i)

}

# Pie chart

pie(table(iris$Species), main="Pie Chart of Iris Species")
```