PART A

Aim: Create two vectors in R for numeric data and display there addition,subtraction,multiplication,division.

```
vac1 <-c(10,20,30,40)

vac2 <-c(5,10,15,20)

cat("vector1 = ",vac1, "\n")

cat("vector2 = ",vac2,"\n")

addition <- vac1+vac2

subtraction <-vac1-vac2

mul <-vac1*vac2

div <-vac1/vac2

cat("addition : ",addition, "\n")

cat("subtraction : ",subtraction, "\n")

cat("multiplication : ",mul, "\n")

cat("division :",div, "\n")
```

Aim: input a vector for 10 student and display name in sorted order.

```
name_vector=character(10)

for(i in 1:10) {

  name=readline(prompt=(paste0("enter name of student", i,": ")))

  name_vector[i]=name

}

cat(name_vector,"\n")

sortedname=sort(name_vector)

cat(sortedname)
```

Aim: Create a list in a data structure that has components of mixed data types.

```
my_list <- list(name="gaurav",age=19,speaks=c("english","hindi"))
print(my_list)
```

Aim: Create a code to display Fibonacci series.

```
n <-10
a <-0
b <-1
for (i in 3:n) {
c <-a+b
 cat(c," ")
 a <-b
 b <-c
}
```

Aim : Implement decision tree on credit card issue dataset (import from kaggale)

```
install.packages("party")
installed.packages("rtools")
library(party)
iris_1 <-iris[sample(150),]
train=iris[1:100,]
test=iris[101:150,]
tree=ctree(Species~Petal.Length+Petal.Width,data = train)
plot(tree)
p= predict(tree, test)
```

Result : The Decision tree has been successfully executed.

```
test$Species) accuracy = (21 + 16 + 10) / 50 * 100 accuracy
```

AIM. Implement the KNN algorithm on the Brest cancer dataset.

```
if (!requireNamespace("e1071")) install.packages("e1071")

library(e1071)

data(iris)

set.seed(123)

sample_index <- sample(nrow(iris), size = 0.7 * nrow(iris), replace = FALSE)

train_data <- iris[sample_index, ]

test_data <- iris[-sample_index, ]

model <- naiveBayes(Species ~ ., data = train_data)

predictions <- predict(model, newdata = test_data)

conf_matrix <- table(predictions, test_data$Species)

print(conf_matrix)

accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)

print(paste("Accuracy:", round(accuracy, 4)))

plot(train_data)
```

Aim : Implement the Naïve Bayes algorithm on the iris dataset.

```
install.packages("e1071")

library(e1071)

library(ggplot2)

library(reshape2)
```

```
data(iris)

set.seed(123)
sample_index <- sample(1:nrow(iris), 0.8 * nrow(iris))
train_data <- iris[sample_index, ]
test_data <- iris[-sample_index, ]

naive_bayes_model <- naiveBayes(Species ~ ., data = train_data)

predictions <- predict(naive_bayes_model, test_data)

conf_matrix <- table(predictions, test_data$Species)

print("Confusion Matrix:")
print(conf_matrix)

accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
print(paste("Accuracy:", round(accuracy, 4)))
```

AIM : input two matrix and show the addition of both matrices

```
mat1 <- matrix(numeric(), nrow = 3, ncol = 3)
cat("Enter elements of the first 3x3 matrix:\n")
```

```
for (i in 1:3) {
  for (j in 1:3) {
    mat1[i, j] <- as.numeric(readline(prompt = paste("Enter element [", i, ",", j, "]: ")))
  }
}

mat2 <- matrix(numeric(), nrow = 3, ncol = 3)
cat("Enter elements of the second 3x3 matrix:\n")
for (i in 1:3) {
  for (j in 1:3) {
    mat2[i, j] <- as.numeric(readline(prompt = paste("Enter element [", i, ",", j, "]: ")))
  }
}

cat("\nFirst Matrix:\n")
print(mat1)
cat("\nSecond Matrix:\n")
print(mat2)
add_mat <- mat1 + mat2

cat("\nAddition Matrix:\n")
print(add_mat)
```

PART B

1.implement random forest algorithm on iris data set

```
nstall.packages("randomForest")
library(randomForest)
iris_1 <- iris[sample(150),]
View(iris_1)
train <- iris_1[1:100,]
test <- iris_1[101:150,]
model <- randomForest(Species~.,data=train)
plot(model)
p=predict(model,test)
p
```

2. implement K mean clustering in your own dataset CREATE DATASET using vector

```
installed.packages("ggplot")
library(ggplot2)
set.seed(123)
mydata <- data.frame(x=runif(100),y=runif(100),z=runif(100))
kmeans_result <- kmeans(mydata,3)
centers <- kmeans_result$centers
cluster_assignments <- kmeans_result$cluster
print(centers)
ggplot(mydata, aes(x,y,color=factor(cluster_assignments)))+geom_point()
```

3.implement linar regression on iris dataset

```
library(datasets)
data("iris")
```

```
library(ggplot2)
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point() +
  geom_smooth(method = "lm", color = "blue")
```

4.implement SVM algorithm using WDBC dataset

```
# Load necessary packages
library(e1071)
library(caret)

# Load dataset
install.packages("e1071")
install.packages("caret")
library(e1o71)
library(caret)

data("WDBC")
df <- WDBC

# Preprocess data
df$diagnosis <- factor(df$diagnosis, levels = c("B", "M"))

# Split the data into training and testing sets
set.seed(123)
train_index <- createDataPartition(df$diagnosis, p = 0.7, list = FALSE)
train_data <- df[train_index, ]
test_data <- df[-train_index, ]
```

```
# Train SVM model
svm_model <- svm(diagnosis ~ ., data = train_data, kernel = "radial")

# Make predictions
predictions <- predict(svm_model, newdata = test_data)

# Evaluate model
confusionMatrix(predictions, test_data$diagnosis)
```

5.Implement logistic regression on the IRIS dataset

```
install.packages("caTools")
install.packages("ROCR")
library(caTools)
library(ROCR)

# Load dataset and fix incorrect function call
dataset <- mtcars

# Split the dataset into training and testing sets
set.seed(123)
split <- sample.split(dataset$vs, SplitRatio = 0.7)
train_reg <- subset(dataset, split == TRUE)
test_reg <- subset(dataset, split == FALSE)

# Train logistic regression model
logistic_model <- glm(vs ~ wt + disp, data = train_reg, family = binomial)
```

```
# Print summary of the logistic model
summary(logistic_model)

# Make predictions on the test set
predict_reg <- predict(logistic_model, test_reg, type = "response")
predict_reg <- ifelse(predict_reg > 0.5, 1, 0)

# Evaluate model
confusion_matrix <- table(test_reg$vs, predict_reg)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste('Accuracy =', accuracy))

# Create ROC curve
ROCPred <- prediction(predict_reg, test_reg$vs)
ROCPer <- performance(ROCPred, measure = "tpr", x.measure = "fpr")
auc <- performance(ROCPred, measure = "auc")
auc <- auc@y.values[[1]]
plot(ROCPer, colorize = TRUE, print.cutoffs.at = seq(0.1, by = 0.1), main = "ROC CURVE")
abline(a = 0, b = 1)
auc <- round(auc, 4)
legend(0.6, 0.4, legend = paste("AUC =", auc), cex = 1, title = "AUC")
```

6. AIM: Implement Aprori algorithm

```
install.packages("arules")
install.packages("arulesViz")
install.packages("RColorBrewer")
library(arules)
library(arulesViz)
library(RColorBrewer)
data("Groceries")
rules <- apriori(Groceries, parameter = list(supp = 0.01, conf = 0.2))
inspect(rules[1:10])
itemFrequencyPlot(
  Groceries,
  topN = 20,
  col = brewer.pal(8, 'Pastel2'),
  main = 'Relative Item Frequency Plot',
  type = "relative",
  ylab = "Item Frequency (Relative)"
)
```

7.import IRIS data set and display first three columns

```
data(iris)
print(iris)
iris_subset <- iris[, c(1, 3, ncol(iris))]
print(iris_subset)
```

8. import IRIS data set and display first ,3RD,LAST columns

```
 data(iris)
print(iris)
iris[, c(1, 3, ncol(iris))]
print(iris_subset)
```