

AI19542 - DATA SCIENCE USING R



**RAJALAKSHMI
ENGINEERING COLLEGE**
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE
LEARNING**

AI19542 – DATA SCIENCE USING R

THIRD YEAR

FIFTH SEMESTER

2024 - 2025

ODD SEMESTER

Ex No:1

Date:

Basics of R – data types, vectors, factors, list and data frames

AIM:

To implement and understand the basics of R programming with its data types, vectors, factors, list and data frames.

ALGORITHM:

1. Start
2. Assign values in logical, numerical, character, complex and character in raw form to a variable v.
3. Print the class of v.
4. Assign a vector for subject Names, temperature and flu_status for three patients using c() function and access the elements.
5. Create a factor using factor() with duplicate values and assign level with distinct values.
6. Display the specific element and check for certain values in factor.
7. Create a list using list() from the patient details and access the multiple elements.
8. Create a data frame using data.frame() with multiple vectors as features. Access the elements.
9. Create a matrix using matrix() with different allocations and access the elements.
10. Stop.

PROGRAM:

```
#Data Types
v<-TRUE
print(class(v))
v<-23.5
print(class(v))
v<-2L
print(class(v))
v<-2+5i
print(class(v))
v<-"TRUE"
print(class(v))
v<-charToRaw("Hello")
print(class(v))

#Vectors
subject_name<-c("John Doe","Jane Doe","Steven Grant")
temperature<-c(98.1,98.6,101.4)
flu_status<-c(FALSE,FALSE,TRUE)
temperature[2]
temperature[2:3]
temperature[-2]

#Factors
gender<-factor(c("MALE","FEMALE","MALE"))
gender
blood<-factor(c("O","AB","A"),levels=c("A","B","AB","O"))
```

```

blood[1:2]
symptoms<-factor(c("SEVERE","MILD","MODERATE"),
  levels=c("MILD","MODERATE","SEVERE"),
  ordered=TRUE)
symptoms>"MODERATE"

#Lists
subject1<-list(fullname=subject_name[1],
  temperature=temperature[1],
  flu_status=flu_status[1],
  gender=gender[1],
  blood=blood[1],
  symptoms=symptoms[1])
subject1
subject1[2]
subject1[[2]]
subject1$temperature
subject1[c("temperature","flu_status")]

#Data Frames
pt_data<-data.frame(subject_name, temperature, flu_status,
  gender,blood,symptoms)
pt_data
pt_data$subject_name
pt_data[c("temperature","flu_status")]
pt_data[c(1,2),c(2,4)]
pt_data[,1]
pt_data[,]

#Matrices
m<-matrix(c(1,2,3,4),ncol=2)
print(m)
m<-matrix(c(1,2,3,4,5,6),nrow=3)
print(m)
print(m[1,])
print(m[1,])
thismatrix <- matrix(c("apple", "banana", "cherry","orange"), nrow = 2, ncol = 2)
for (rows in 1:nrow(thismatrix)) {
  for (columns in 1:ncol(thismatrix)) {
    print(thismatrix[rows, columns])
  }
}

```

OUTPUT:

```
File Edit Selection View Go Run Terminal Help
PROBLEMS 73 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

[1] "logical"
[1] "numeric"
[1] "integer"
[1] "complex"
[1] "character"
[1] "raw"
[1] 98.6
[1] 98.6 101.4
[1] 98.1 101.4
[1] MALE FEMALE MALE
Levels: FEMALE MALE
[1] O AB
Levels: A B AB O
[1] TRUE FALSE FALSE
$fullname
[1] "John Doe"

$temperature
[1] 98.1

$flu_status
[1] FALSE

$gender
[1] MALE
Levels: FEMALE MALE

$blood
[1] O
Levels: A B AB O

$symptoms
[1] SEVERE
Levels: MILD < MODERATE < SEVERE

$temperature
[1] 98.1

[1] 98.1
[1] 98.1
$temperature
[1] 98.1

$flu_status
[1] FALSE

  subject_name temperature flu_status gender blood symptoms
1 John Doe      98.1      FALSE MALE      O      SEVERE
2 Jane Doe      98.6      FALSE FEMALE AB      MILD
3 Steven Grant  101.4      TRUE  MALE      A      MODERATE
[1] "John Doe"      "Jane Doe"      "Steven Grant"

  temperature flu_status
1 98.1      FALSE
2 98.6      FALSE
3 101.4      TRUE
```

```
File Edit Selection View Go Run Terminal Help BasicsO
PROBLEMS 73 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

2 98.6 FALSE
3 101.4 TRUE
temperature gender
1 98.1 MALE
2 98.6 FEMALE
[1] "John Doe" "Jane Doe" "Steven Grant"
subject_name temperature flu_status gender blood symptoms
1 John Doe      98.1      FALSE MALE      O      SEVERE
2 Jane Doe      98.6      FALSE FEMALE AB      MILD
3 Steven Grant  101.4      TRUE  MALE      A      MODERATE
[,1] [,2]
[1,] 1 3
[2,] 2 4
[,1] [,2]
[1,] 1 4
[2,] 2 5
[3,] 3 6
[1] 1 4
[1] 1 4
[1] "apple"
[1] "cherry"
[1] "banana"
[1] "orange"
>
```

Result:

Thus the R Script program to implement various data types, vectors, factors, lists and data frames is executed successfully and the output is verified.

Ex no: 2

Diagnosis of Breast Cancer using KNN.

Date:

Aim:

To implement a R program to predict and diagnose Breast Cancer using KNN algorithm.

Algorithm:

1. Start
2. Read the csv file from the directory and store it in bcd variable.
3. Drop the first column id.
4. Change the diagnosis feature with categorical values B and M in a factor
5. Normalize the dataset.
6. Split the dataset for training and testing, with diagnosis as the response variable and the rest as the predictor variables.
7. Import the library "class" for knn classification.
8. Predict the knn model using knn() with 5 clusters with the corresponding training and testing data.
9. Display the confusion matrix and accuracy of the knn model.
10. Stop

PROGRAM:

```
bcd<-read.csv("../input/breast-cancer-dataset/Breast_Cancer.csv", stringsAsFactors=FALSE)
bcd<-bcd[-1]

bcd$diagnosis<-factor(bcd$diagnosis, levels=c("B","M"), labels=c("Benign","Malignant"))

normalize<-function(x){
  return (x-min(x)) / (max(x)- min(x))
}

bcd_n <- as.data.frame(lapply(bcd[2:31], normalize))
x_train <- bcd_n[1:469,]
x_test <- bcd_n[470:569,]
y_train <- bcd[1:469,1]
y_test <- bcd[470:569,1]

library(class)

y_pred<-knn(train=x_train,test=x_test,cl=y_train,k=5)
tbl=table(x=y_test,y=y_pred)

tbl

accuracy = sum(diag(tbl))
```

OUTPUT:

```
PROBLEMS 11 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

'data.frame':  569 obs. of  32 variables:
 $ id          : int  87139402 8910251 905520 068871 9012568 906539 925291 07880 862989 89627 ...
 $ diagnosis   : chr  "B" "B" "B" "B" ...
 $ radius_mean : num  12.3 18.6 11 11.3 15.2 ...
 $ texture_mean : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean : num  78.8 89.3 70.9 73 97.7 ...
 $ area_mean   : num  464 346 373 385 712 ...
 $ smoothness_mean : num  0.1028 0.0969 0.1077 0.1164 0.0796 ...
 $ compactness_mean : num  0.0698 0.1147 0.078 0.1136 0.0693 ...
 $ concavity_mean : num  0.0399 0.0639 0.0305 0.0464 0.0339 ...
 $ points_mean  : num  0.037 0.0264 0.0248 0.048 0.0266 ...
 $ symmetry_mean : num  0.196 0.192 0.171 0.177 0.172 ...
 $ dimension_mean : num  0.0595 0.0649 0.0634 0.0607 0.0554 ...
 $ radius_se    : num  0.236 0.451 0.197 0.338 0.178 ...
 $ texture_se   : num  0.666 1.197 1.387 1.343 0.412 ...
 $ perimeter_se : num  1.67 3.43 1.34 1.85 1.34 ...
 $ area_se      : num  17.4 27.1 13.5 26.3 17.7 ...
 $ smoothness_se : num  0.00885 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se : num  0.0118 0.03581 0.00936 0.03408 0.01405 ...
 $ concavity_se  : num  0.0168 0.0335 0.0106 0.0219 0.0155 ...
 $ points_se     : num  0.01241 0.01365 0.00748 0.01965 0.00915 ...
 $ symmetry_se   : num  0.0192 0.035 0.0172 0.0158 0.0165 ...
 $ dimension_se  : num  0.00225 0.00332 0.0022 0.00344 0.00177 ...
 $ radius_worst : num  13.5 11.9 12.4 11.9 16.2 ...
 $ texture_worst : num  15.6 22.9 26.4 15.8 15.7 ...
 $ perimeter_worst : num  87 78.3 79.9 76.5 104.5 ...
 $ area_worst    : num  548 425 471 434 819 ...
 $ smoothness_worst : num  0.139 0.121 0.137 0.137 0.113 ...
 $ compactness_worst : num  0.127 0.252 0.148 0.182 0.174 ...
 $ concavity_worst : num  0.1242 0.1916 0.1067 0.0867 0.1362 ...
 $ points_worst  : num  0.0939 0.0793 0.0743 0.0861 0.0818 ...
 $ symmetry_worst : num  0.283 0.294 0.3 0.21 0.249 ...
 $ dimension_worst : num  0.0677 0.0759 0.0788 0.0678 0.0677 ...

      y
x      Benign Malignant
Benign    61         0
Malignant   4         35
[1] "Accuracy 96"
> []
```

Result:

Thus the R Script program to implement diagnosis of Breast Cancer using K-Nearest Neighbour algorithm is executed successfully and the output is verified.

Ex No: 3

Date:

Filtering Mobile phone spam using Naïve Bayes

AIM:

To implement a R program to Filter Mobile phone spam using Naïve Bayes.

ALGORITHM:

1. Start
2. Import the csv file and store the dataframe in "Sms". Have a glimpse at the structure of the data frame.
3. Remove the unnecessary columns which is from column 3 to 5.
4. Convert the labels as factors.
5. Remove special characters from the dataset and retain only alpha numeric characters using `alnum` in `str_replace_all()` from "stringr" package.
6. Create a volatile corpus `VCorpus()` for text mining from the source object of "v2" which is extracted using `VectorSource()`.
7. Create a `DocumentTermMatrix()` to split the SMS message into individual Components.
8. Create training and testing dataset with the split ratio 0.75.
9. Find the frequent terms which appear for atleast 5 times in `DocumentTermMatrix` in training and testing dataset respectively.
10. Train the model using `naiveBayes()` from `e1071` library.
11. Evaluate the model Performance.
12. Print the confusion matrix and Accuracy of the model.
13. Stop.

PROGRAM:

```
sms <- read.csv("../input/spam-ham-dataset/spam.csv", stringsAsFactors=FALSE)
str(sms)
sms <- sms[-3:-5]
sms$v1 <- factor(sms$v1)
library(stringr)
sms$v2 = str_replace_all(sms$v2, "[^[:alnum:]]", " ") %>% str_replace_all(., "[
]+", " ")
library(tm)
sms_corpus <- VCorpus(VectorSource(sms$v2))
```

```

print(sms_corpus)
print(as.character(sms_corpus[[6]]))
sms_dtm <- DocumentTermMatrix(sms_corpus, control = list
(tolower=TRUE,removeNumbers=TRUE,stopwords=TRUE,removePunctuations=TRUE,stemmi
ng=TRUE))
x_train <- sms_dtm[1:4169, ]
x_test <- sms_dtm[4170:5572, ]
y_train <- sms[1:4169, ]$v1
y_test <- sms[4170:5572, ]$v1
sms_freq_word_train <- findFreqTerms(x_train, 5)
sms_freq_word_test <- findFreqTerms(x_test, 5)
x_train<- x_train[ , sms_freq_word_train]
x_test <- x_test[ , sms_freq_word_test]
convert_counts <- function(x) {x <- ifelse(x > 0, "Yes", "No")}
x_train <- apply(x_train, MARGIN = 2,convert_counts)
x_test <- apply(x_test, MARGIN = 2,convert_counts)
library(e1071)
model <- naiveBayes(x_train, y_train,laplace=1)
y_pred <- predict(model, x_test)
cm = table(y_pred, y_test)
print(cm)
acc = sum(diag(cm))/sum(cm)
print(paste("Accuracy: ",acc*100,"%"))

```

OUTPUT:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
[1] "Accuracy: 98.346828252336 %"
>

```

RESULT:

Thus the R program to implement filtering of Mobile phone spam using Naïve Bayes is executed successfully and the output is verified.

Ex No:4

Risky Bank Loans using Decision Trees

Date:

AIM:

To implement a R program to find Risky Bank loans using Decision Tree.

ALGORITHM:

1. Start
2. Import the dataset credit.csv and display the structure of the dataset.
3. Display the table to find the range of values and find the missing values.
4. Factorise the default column and set seed of 123.
5. Split the dataset for training and testing in the ratio of 0.8, with “default” as the response variable, and the rest as predictor variables.
6. Import the library C5.0 for implementing decision tree.
7. Train the decision tree model using C5.0 function for the training dataset.
8. Test the model to predict using predict(). Print the confusion matrix.
9. Print the accuracy of the decision tree model.
10. Stop

PROGRAM:

```
credit <- read.csv("credit.csv")

str(credit)

table(credit$savings_balance)

summary(credit$amount)

credit$default <- factor(credit$default)

set.seed(123)

train_sample <- sample(1000, 800)

str(train_sample)

x_train <- credit[train_sample, -17]

x_test <- credit[-train_sample, -17]

y_train <- credit[train_sample, 17]

y_test <- credit[-train_sample, 17]

library(C50)

model <- C5.0(x_train,y_train)
```

```
summary(model)
```

```
y_pred <- predict(model,x_test)
```

```
cm = table(y_pred,y_test)
```

```
print(cm)
```

```
acc=sum(diag(cm))/sum(cm)
```

```
print(paste("Accuracy: ",acc*100,"%"))
```

OUTPUT:

```
Run: Terminal - RStudio
RStudio - RStudio Desktop - RStudio Desktop
RStudio - RStudio Desktop - RStudio Desktop

> credit <- read.csv("E:\\Academics\\Semester-V\\Data Science using R\\credit
> str(credit)
'data.frame':   1000 obs. of  17 variables:
 $ checking_balance : chr  "x 0 00" "x 100 00" "unknown" "x 0 00" ...
 $ months_loan_duration: int  4 40 12 40 24 36 24 36 12 36 ...
 $ credit_history : chr   "critical" "good" "critical" "good" ...
 $ purpose : chr   "furniture/appliances" "furniture/appliances" "education" "furniture/appliances" ...
 $ sex : chr   "male" "female" "male" "female" "male" "female" ...
 $ savings_balance : chr   "unknown" "x 100 00" "x 100 00" "x 100 00" ...
 $ employment_duration : chr  "> 7 years" "1 - 4 years" "2 - 7 years" "2 - 7 years" ...
 $ percent_of_income : int   4 2 2 2 2 2 2 2 2 ...
 $ years_at_residence : int   4 2 1 4 4 4 2 4 2 ...
 $ age : chr   40 32 30 40 45 45 45 45 45 ...
 $ other_credit : chr   "none" "none" "none" "none" ...
 $ housing : chr   "rent" "own" "own" "other" ...
 $ existing_loan_count : int   2 1 1 2 2 1 1 1 2 ...
 $ job : chr   "skilled" "skilled" "unskilled" "skilled" ...
 $ dependencies : chr   1 1 0 2 2 2 1 1 1 ...
 $ phone : chr   "yes" "no" "no" "no" ...
 $ default : chr   "no" "yes" "no" "no" ...
> cm <- table(credit$checking_balance)
> summary(cm)
Min. 1st Qu. Median Mean 3rd Qu. Max.
 200    1700    2100    2071    3072    10404
> credit$default <- factor(credit$default)
> set.seed(123)
> train_sample <- sample(1000, 800)
> str(train_sample)
int [1:800] 435 463 129 526 395 308 818 110 299 225 ...
> x_train <- credit[train_sample, 1:17]
> x_test <- credit[-train_sample, 1:17]
> y_train <- credit[train_sample, 17]
> y_test <- credit[-train_sample, 17]
> library(C50)
> model <- C5.0(x_train,y_train)
> summary(model)

Call:
C5.0.default(x = x_train, y = y_train)

C5.0 [Maximum size 999, 1024 nodes]      Wed Oct 26 11:00:08 2022

Class specified by attribute 'outcome'
```

```
Decision tree:
checking_balance in {unknown,x 00 00} no (101/54)
checking_balance in {x 0 00,x 100 00} yes (99)
...credit_history in {very good,perfect}
...housing < rent} yes (50/1)
  housing < other}
  1 ...employment_duration in {1 - 4 years,4 - 7 years,
  2 1 ...employment_duration in {> 7 years} yes (30)
  3 employment_duration in {> 1 year,unemployed}
  4 ...job < skilled} yes (1)
  5 job < management,unskilled,unemployed} no (2)
  housing < own}
  1 ...purpose in {business,education} no (3/2)
  2 purpose in {recreation,car} yes (1)
  3 purpose < car}
  4 months_loan_duration < 18} yes (1)
  5 months_loan_duration < 18}
  6 ...job in {skilled,management,unemployed} no (1)
  7 job < unskilled} yes (1)
  8 purpose < furniture/appliances}
  9 ...other_credit < other} no (4)
  10 other_credit in {none,none}
  11 ...existing_loan_count <= 1} yes (4)
  12 existing_loan_count <= 1} no (2)
credit_history in {good,critical,good}
...months_loan_duration <= 18}
  1 ...purpose in {business,car} no (14)
  2 purpose < furniture/appliances}
  3 ...savings_balance in {unknown,x 100 00,500 - 1000 00,
  4 x 1000 00} no (40/24)
  5 savings_balance < 100 - 500 00} yes (1)
  6 purpose < education}
  7 ...savings_balance < unknown} no (1)
  8 savings_balance in {x 100 00,100 - 500 00,500 - 1000 00,
  9 x 1000 00} yes (7/3)
  10 purpose < recreation}
  11 ...other_credit in {none,none} no (5/1)
  12 other_credit < other} yes (1)
  13 purpose < car}
  14 ...credit_history in {critical,good} no (17/4)
  15 credit_history < good}
  16 ...existing_loan_count <= 1} yes (2)
```

```

> if (...checking_balance == 0 (0): yes (0)
> if (...checking_balance == 1 - 200 (0): no (1/1)
> purpose = furniture/appliances:
> ...savings_balance in (100 - 500 (0):
> | 500 - 1000 (0): yes (0)
> | savings_balance < 100 (0):
> | ...months_loan_duration < 22: yes (14/1)
> | | months_loan_duration > 22:
> | | ...amount < 1250: yes (3)
> | | | amount > 1250: no (0)
> employment_duration = 4 - 7 years:
> ...savings_balance in (100 - 500 (0):
> | savings_balance < 100 (0):
> | ...job in (management,unskilled,
> | | unemployed): no (0)
> | | job = skilled:
> | | | dependents < 1: no (1/1)
> | | | ...months_loan_duration < 22: no (0)
> | | | months_loan_duration > 22: yes (0)
> employment_duration = > 7 years:
> ...other_credit = other: no (2)
> | other_credit = bank:
> | ...job in (skilled,unemployed): yes (0)
> | | job in (management,unskilled): no (4/1)
> | other_credit = none:
> | ...purpose = business: yes (2)
> | | purpose in (education,renovation,
> | | | car): no (1)
> | | purpose = furniture/appliances:
> | | ...job in (skilled,unskilled,unemployed): no (0)
> | | | job = management: yes (1)
> | | purpose = car:
> | | ...amount < 8000: no (7/1)
> | | | amount > 8000: (1)

summary (x)

checking_balance = 0 (0): no (1)
checking_balance = 1 - 200 (0): yes (1)

```

evaluation on training data (900 cases):

```

Decision tree
Size Errors
28 99(11.0%) or

(4) (0) <-classified as
825 99 (0): class no
89 170 (0): class yes

```

Attribute usage:

```

100.00% checking_balance
54.22% credit_history
44.22% months_loan_duration
42.22% savings_balance
31.66% purpose
33.33% employment_duration
9.25% years_at_residence
8.75% housing
6.44% job
6.33% other_credit

```

```

3.75% amount

```

```

4.44% existing_loan_cost
4.22% phone
9.44% percent_of_income
3.96% dependents
8.75% age

```

Time: 0.0 sec

```

> y_pred <- predict(model,x_test)
> m <- table(y_pred,y_test)
> print(m)
y_test
y_pred no yes
no 23 20
yes 19 15
> acc <- sum(diag(m))/sum(m)
> print(paste("Accuracy: ",acc*100,"%"))
[1] "Accuracy: 90.5%"
>

```

RESULT:

Thus the R program to find Risky Bank loans using Decision Tree is executed successfully and the output is verified.

Ex No: 5

Date:

Medical Expense with Linear Regression.

AIM:

To implement a R program to predict Medical Expense using Linear Regression

ALGORITHM:

1. Start
2. Load the Insurance dataset and analyse the structure of the dataset.
3. Get the summary statistics. Check whether the distribution is right-skewed or left skewed by comparing the mean and median. Verify the same using histogram.
4. Check the distribution of "region" using table.
5. Create a correlation matrix of "age", "bmi", "children", "expenses".
6. To determine the pattern of the dataset, use scatterplot using pairs() for "age", "bmi", "children", "expenses".
7. To display a more informative scatterplot use pairs.panel() from "psych" library.
8. Fit the linear regression model using lm() with expenses as the dependent variable.
9. Evaluate the model performance using summary().
10. To improve the model performance, square the age variable as age2 and bmi30 is 1 if bmi >= 30 else 0.
11. Train the model with age + age2 + bmi30 as also as the independent variables.
12. Evaluate the model performance for model2 using summary().
13. Stop.

PROGRAM:

```
insurance<-read.csv("insurance.csv",stringsAsFactors = TRUE)

str(insurance)

summary(insurance$expenses)

hist(insurance$expenses)

table(insurance$region)

cor(insurance[c("age","bmi","children","expenses")])

pairs(insurance[c("age","bmi","children","expenses")])

library(psych)

pairs.panels(insurance[c("age","bmi","children","expenses")])

ins_model <- lm(expenses ~ age + children + bmi + sex + smoker + region, data =
insurance)

ins_model
```

```
summary(ins_model)
```

```
insurance$age2 <- insurance$age^2
```

```
insurance$bmi30 <- ifelse(insurance$bmi >= 30,1,0)
```

```
expenses ~ bmi30*smoker
```

```
expenses ~ bmi30+smokeryes+bmi30:smokeryes
```

```
ins_model2 <- lm(expenses ~ age+age2+children+bmi+sex+bmi30*smoker+region,  
data=insurance)
```

```
summary(ins_model2)
```

OUTPUT:

```
Summary: read.csv("C:/Users/Devi/Desktop/FA/Data Science using R/108  
> str(insurance)  
'data.frame': 1338 obs. of  7 variables:  
 $ age      : int  36 36 36 36 32 34 36 37 39 40 ...  
 $ sex      : factor w/ 2 levels "female","male": 1 1 1 1 1 1 1 1 1 1 ...  
 $ bmi      : num  27.5 33.0 33.0 33.0 29.0 29.7 33.4 37.7 29.0 29.0 ...  
 $ children : int  0 1 0 0 0 1 1 1 0 ...  
 $ smoker   : factor w/ 4 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...  
 $ region   : factor w/ 4 levels "northwest","northeast",...: 4 1 1 1 1 1 1 1 1 ...  
 $ expenses : num  3085 3726 4449 2384 3807 ...  
# summary(insurance$expenses)  
# Min. 1st Qu. Median      Mean 3rd Qu.      Max.        
# 1122    2348    3982    3129.9    4449    6379  
# first(insurance$expenses)  
# table(insurance$region)  
  
northwest  northeast  southeast  southwest  
    524         525         584         525  
# cor(insurance[,c("age","bmi","children","expenses")])  
#           age      bmi      children      expenses  
age      1.0000000  0.1891101  0.0140081  0.2990619  
bmi      0.1891101  1.0000000  0.0134471  0.1349709  
children 0.0140081  0.0134471  1.0000000  0.0679823  
expenses 0.2990619  0.1349709  0.0679823  1.0000000  
# pairs(insurance[,c("age","bmi","children","expenses")])  
# library(ggplot2)  
# pairs.gpairs(insurance[,c("age","bmi","children","expenses")])  
Error: unexpected numeric constant in 'pairs.gpairs(insurance[,c("age","bmi","children","expenses")])$'  
# ins_model <- lm(expenses ~ age + children + bmi + sex + smoker + region, data = insurance)  
# ins_model  
  
Call:  
lm(formula = expenses ~ age + children + bmi + sex + smoker +  
    region, data = insurance)  
  
Coefficients:  
(Intercept)      age      children      bmi  
 13007.4      296.8      475.7      475.7  
sexmale      smokeryes  regionnorthwest  regionnortheast  
  -111.4      2384.5      -312.9      -1825.6  
regionseasouthwest  
   -305.5  
  
# summary(ins_model)  
  
Call:  
lm(formula = expenses ~ age + children + bmi + sex + smoker +  
    region, data = insurance)  
  
Residuals:  
      Min       1Q   Median       3Q      Max  
-1150.7   -486.9    -97.6    381.9    2081.7  
  
Coefficients:  
            (Intercept)      age      children      bmi  
            13007.4      296.8      475.7      475.7  
sexmale      smokeryes  regionnorthwest  regionnortheast  
            -111.4      2384.5      -312.9      -1825.6  
regionseasouthwest  
             -305.5  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 500 on 1329 degrees of freedom  
Multiple R-squared:  0.7980,    Adjusted R-squared:  0.7884  
F-statistic: 506.0 on 6 and 1329 Df,    p-value: < 2.2e-16  
  
# insurance$age2 <- insurance$age^2  
# insurance$bmi30 <- ifelse(insurance$bmi >= 30,1,0)  
# expenses ~ bmi30*smoker  
# expenses ~ bmi30+smokeryes+bmi30:smokeryes  
# expenses ~ bmi30+smokeryes+bmi30:smokeryes  
# ins_model2 <- lm(expenses ~ age+age2+children+bmi+sex+bmi30*smoker+region, data = insurance)  
# summary(ins_model2)  
  
Call:  
lm(formula = expenses ~ age + age2 + children + bmi + sex + bmi30 *  
    smoker + region, data = insurance)  
  
Residuals:  
      Min       1Q   Median       3Q      Max  
-1229.1   -486.8    -102.7    377.8    2435.6
```

```

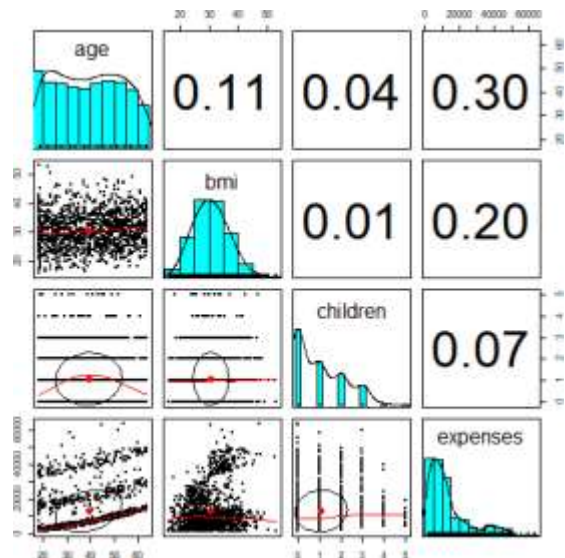
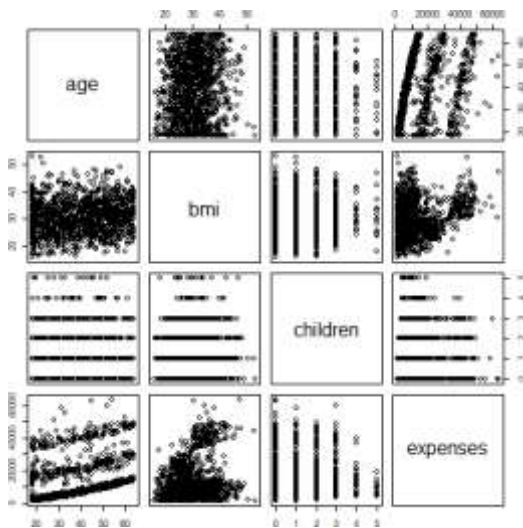
model <- reglm(data = insurance)

Residuals:
    Min:    -10.1    Median:    -0.4    Q1:    -2.8    Q3:    2.8    Max:   10.1

Coefficients:
(Intercept)  239.0512 150.1194  0.102  0.048752
age          -22.0381  50.8298  0.545  0.985098
age2         -1.7387   0.7803  0.888  0.34e-07 ***
children     870.8817  805.8075  0.488  2.01e-16 ***
bmi          159.7725   34.2798  1.484  0.000002 ***
smokeable   -496.7090  344.3713  1.053  0.042007 *
bmi^2       -607.0375  422.8687  1.319  0.018880 *
smokeables  13488.5812  430.8951  30.888  1.26e-16 ***
regionnorthwest  276.1841  340.8826  0.799  0.424285
regionnorthwest  -826.0343  551.5484  1.555  0.020842 *
regionnorthwest  1222.1028  750.5134  1.607  0.000005 ***
bmi^2smokeables 18404.3334  384.6706  32.762  1.24e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4445 on 1336 degrees of freedom
Multiple R-squared:  0.8968,    Adjusted R-squared:  0.8953
F-statistic: 701.7 on 11 and 1336 DF,  p-value: < 2.2e-16

```



RESULT:

Thus the R program to predict medical expenses using linear regression is executed successfully and the output is verified.

Ex No: 6

Modeling strength of concrete.

Date:

AIM:

To build a predictive model for the compressive strength of concrete based on its composition and age using linear regression in R.

ALGORITHM:

1. Start
2. Load the Insurance dataset and check its structure.
3. Get summary statistics and check skewness using mean, median, and histogram.
4. Check the distribution of "region" using a table.
5. Create a correlation matrix for "age," "bmi," "children," and "expenses."
6. Use scatterplots to examine relationships among "age," "bmi," "children," and "expenses."
7. Fit an initial linear model with "expenses" as the target, then improve by adding `age2` (age squared) and `bmi30` (1 if bmi \geq 30) and re-evaluate.
8. Stop

PROGRAM:

```
library(caret)

library(ggplot2)

data <- read.csv("concrete.csv")

head(data)

sum(is.na(data))

set.seed(123)

trainIndex <- createDataPartition(data$CompressiveStrength, p =
0.8, list = FALSE)

trainData <- data[trainIndex,]

testData <- data[-trainIndex,]
```

```
model <- lm(CompressiveStrength ~ ., data = trainData)
```

```
summary(model)
```

```
predictions <- predict(model, newdata = testData)
```

```
mae <- mean(abs(predictions - testData$CompressiveStrength))
```

```
print(paste("Mean Absolute Error:", round(mae, 2)))
```

```
ggplot() +
```

```
  geom_point(aes(x = testData$CompressiveStrength, y = predictions), color = 'blue') +
```

```
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
```

```
  labs(title = "Predicted vs Actual Compressive Strength",
```

```
        x = "Actual Strength",
```

```
        y = "Predicted Strength") +
```

```
  theme_minimal()
```

OUTPUT:

```
> str(concrete)
'data.frame': 1030 obs. of 10 variables:
 $ cement      : num  540 540 332 332 199 ...
 $ slag        : num   0 0 142 142 132 ...
 $ ash         : num   0 0 0 0 0 0 0 0 ...
 $ water       : num  162 162 228 228 192 228 228 228 ...
 $ superplastic : num   2.5 2.5 0 0 0 0 0 0 ...
 $ coarseagg   : num  1040 1055 932 932 978 ...
 $ fineagg     : num   676 676 594 594 826 ...
 $ age         : int   28 28 270 365 360 90 365 28 28 ...
 $ strength    : num   80 61.9 40.3 41 44.3 ...
 $ Predicted_Strength: num  55.1 54.7 57.6 68 59.4 ...

> summary(model)
Call:
lm(formula = strength ~ cement + slag + water + superplastic +
    coarseagg + fineagg + age, data = concrete)

Residuals:
    Min     1Q   Median     3Q     Max
-30.901  -7.239   0.441   6.899  34.408

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 121.611036  17.015934   7.147 1.69e-12 ***
cement       0.067636   0.004135  16.357 < 2e-16 ***
slag         0.042550   0.005192   8.196 7.39e-16 ***
water       -0.323265   0.032336  -9.997 < 2e-16 ***
superplastic 0.371641   0.094876   3.917 9.56e-05 ***
coarseagg   -0.027502   0.006913  -3.978 7.44e-05 ***
fineagg     -0.038549   0.006777  -5.688 1.68e-08 ***
age          0.000000   0.000000   0.000 1.00e+00
```



```
age      0.109746  0.005514 19.903 < 2e-16 ***
```

```
---
```

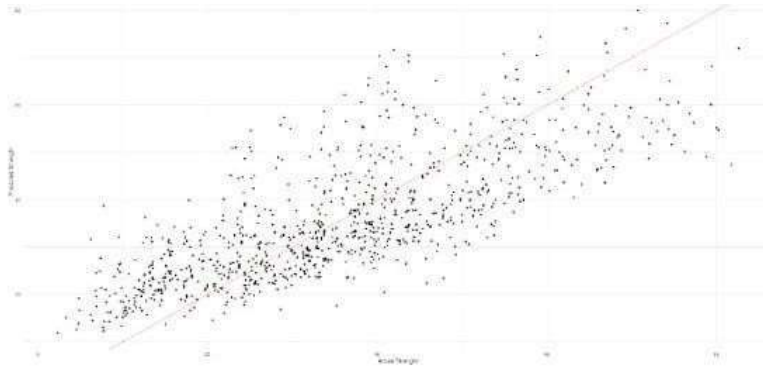
```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 10.64 on 1022 degrees of freedom
```

```
Multiple R-squared:  0.5971,    Adjusted R-squared:  0.5944
```

```
F-statistic: 216.4 on 7 and 1022 DF,  p-value: < 2.2e-16
```

```
> ggplot(concrete, aes(x = strength, y = Predicted_Strength)) +  
+   geom_point() +  
+   geom_abline(slope = 1, intercept = 0, color = "red") +  
+   labs(title = "Actual vs Predicted Concrete Strength",  
+        x = "Actual Strength",  
+        y = "Predicted Strength") +  
+   theme_minimal()
```



```
> cat("R-squared:", r_squared, "\n")  
R-squared: 0.5971302
```

```
> cat("RMSE:", rmse, "\n")  
RMSE: 10.59832
```

RESULT:

Thus the R Script program to implement Modeling strength of concrete is executed successfully and the output is verified.

Ex No: 7

Date:

Identification of frequently Purchased groceries with Apriori algorithm.

AIM:

To identify frequent itemsets of grocery items that are commonly purchased together using the Apriori algorithm. This will help in understanding customer buying patterns and optimizing store layout or inventory.

ALGORITHM:

1. Start
2. Load Data: Load the transaction dataset (assume each transaction is a list of items purchased).
3. Data Preprocessing: Convert the data into a transactional format suitable for association rule mining.
4. Set Parameters: Define minimum support and confidence levels for the Apriori algorithm.
5. Apply Apriori Algorithm: Use the Apriori algorithm to find frequent itemsets.
6. Generate Association Rules: Extract association rules from the frequent itemsets based on support and confidence thresholds.
7. Analyze Results: Sort and filter rules to identify the most frequently purchased item combinations.
8. Stop

PROGRAM:

```
if(!require(arules)) install.packages("arules", dependencies=TRUE)

library(arules)

data("Groceries")

summary(Groceries)

min_support <- 0.01 # Example: at least 1% of transactions
min_confidence <- 0.5 # Example: at least 50% confidence

frequent_itemsets <- apriori(Groceries, parameter = list(supp = min_support, conf =
min_confidence))

summary(frequent_itemsets)

inspect(frequent_itemsets[1:10])

rules <- apriori(Groceries, parameter = list(supp = min_support, conf = min_confidence,
target = "rules"))

summary(rules)

inspect(sort(rules, by = "confidence")[1:10]) # Display top 10 rules by confidence

if(!require(arulesViz)) install.packages("arulesViz", dependencies=TRUE)

library(arulesViz)

plot(rules, method = "graph", control = list(type = "items"))
```

OUTPUT:

Summary of the Groceries Dataset

transactions as itemMatrix in sparse format with 9835 rows (elements/itemsets/transactions) and 169 columns (items) and a density of 0.02609146

most frequent items:

| whole milk | other vegetables | rolls/buns | soda | yogurt | (Other) |
|------------|------------------|------------|------|--------|---------|
| 2513 | 1903 | 1809 | 1715 | 1372 | 34055 |

Frequent Itemsets:

set of 50 itemsets

example of first 10 itemsets (sorted by support):

| items | support |
|----------------------------------|------------|
| 1 {whole milk} | 0.25551601 |
| 2 {other vegetables} | 0.19349263 |
| 3 {rolls/buns} | 0.18393493 |
| 4 {soda} | 0.17437722 |
| 5 {yogurt} | 0.13950178 |
| 6 {whole milk, other vegetables} | 0.0751 |
| 7 {whole milk, yogurt} | 0.0561 |

Association Rules (Top 10 by Confidence):

set of 10 rules

example of first 10 rules (sorted by confidence):

| lhs | rhs | support | confidence | lift |
|------------------------|-----------------|---------|------------|------|
| [1] {yogurt} | => {whole milk} | 0.0561 | 0.4032 | 1.57 |
| [2] {rolls/buns} | => {whole milk} | 0.0567 | 0.3084 | 1.21 |
| [3] {soda} | => {whole milk} | 0.0569 | 0.3058 | 1.20 |
| [4] {tropical fruit} | => {whole milk} | 0.0519 | 0.2674 | 1.03 |
| [5] {other vegetables} | => {whole milk} | 0.0751 | 0.3926 | 1.53 |

RESULT:

Thus the R program to Identification of frequently Purchased groceries with Apriori algorithm is executed successfully and the output is verified.

Ex No: 8

Finding Teen Segments of Market.

Date:

AIM:

The aim of this process is to identify and segment the teen demographic in a market based on behavior, preferences, or other relevant characteristics for targeted marketing or product development.

ALGORITHM:

1. **START:** Collect raw data from sources relevant to the teen market (e.g., social media data, survey responses).
2. **PREPROCESSING:** Clean the data (e.g., remove missing values, correct errors).
3. **SELECT FEATURES:** Choose features that help in segmentation (e.g., age, purchase patterns, interests).
4. **APPLY CLUSTERING ALGORITHM:** Run clustering algorithms (e.g., K-Means or DBSCAN) to create market segments.
5. **EVALUATE MODEL:** Evaluate the clustering performance using a scoring metric (e.g., silhouette score).
6. **VISUALIZE DATA:** Visualize the segmented data to understand different groups.
7. **EXTRACT INSIGHTS:** Identify unique patterns and preferences within each segment.
8. **STOP:** Develop targeted marketing strategies based on the insights from the segmentation.
9. This approach allows businesses to better understand the teen market and tailor their products or marketing campaigns accordingly.

PROGRAM:

```
library(dplyr)
library(ggplot2)
library(corrplot)
load_data <- function(file_path) {
  df <- read.csv(file_path)
  return(df)
}
preprocess_data <- function(df) {
  # Check for missing values
  print(colSums(is.na(df)))
  df[is.na(df)] <- 0 # Fill missing values with 0
  return(df)
}
```

```

analyze_segments <- function(df) {
  # Example: Segment by gender
  gender_counts <- table(df$gender)
  print("Gender Distribution:")
  print(gender_counts)

  interest_features <- c('basketball', 'football', 'soccer', 'softball', 'volleyball',
                        'swimming', 'cheerleading', 'baseball', 'tennis',
                        'cute', 'sexy', 'hot', 'kissed', 'dance',
                        'band', 'marching', 'music', 'rock', 'god',
                        'church', 'jesus', 'bible', 'hair', 'dress',
                        'blonde', 'mall', 'shopping', 'clothes',
                        'hollister', 'abercrombie', 'die', 'death',
                        'drunk', 'drugs')

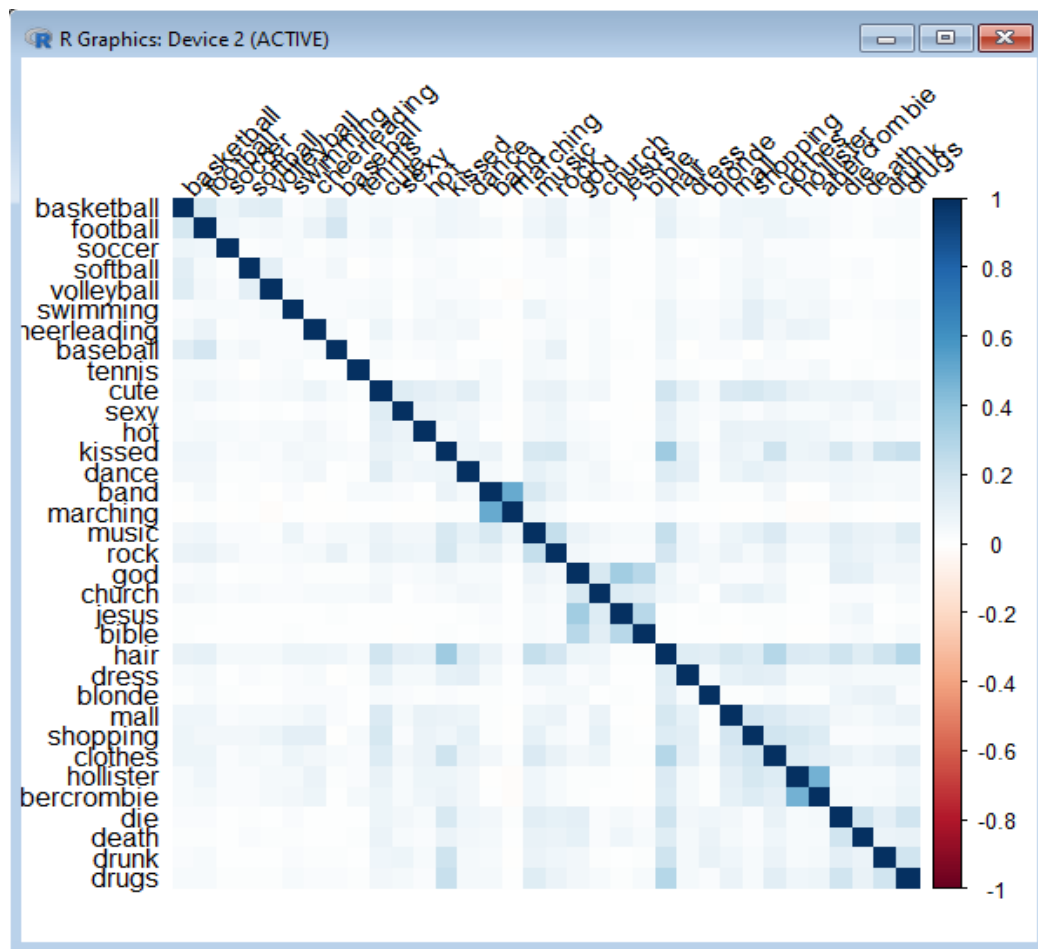
  corr_matrix <- cor(df[interest_features])
  corrplot(corr_matrix, method = "color", tl.col = "black", tl.srt = 45)
}

main <- function(file_path) {
  df <- load_data(file_path)
  df <- preprocess_data(df)
  analyze_segments(df)
}

main('path_to_your_file.csv')

```

OUTPUT:



RESULT:

Thus the R program to Finding Teen Segments of Market is executed successfully and the output is verified.

Ex No: 9

Tuning stock models for better performance.

Date:

AIM:

The aim is to enhance the predictive performance of stock market models by optimizing hyperparameters, improving data features, and using techniques like cross-validation and model selection to better forecast stock prices or trends.

ALGORITHM:

1. Start
2. Data Collection: Gather historical stock data (e.g., price, volume, market sentiment, technical indicators).
3. Data Preprocessing: Clean the data by handling missing values, normalizing features, and creating relevant indicators (e.g., moving averages, RSI).
4. Feature Engineering: Create new features based on existing data to improve model predictions (e.g., lagged values, percentage changes, or volatility).
5. Model Selection: Choose an appropriate model (e.g., Linear Regression, Decision Trees, Random Forest, LSTM for time series).
6. Hyperparameter Tuning: Tune the hyperparameters of the model using techniques like Grid Search or Random Search to optimize performance.
7. Cross-Validation: Implement cross-validation (e.g., k-fold) to ensure that the model generalizes well on unseen data.
8. Model Evaluation: Evaluate the model's performance using metrics like RMSE, MAE, or accuracy, and compare the results with different models.
9. Model Refinement: Refine the model by adjusting hyperparameters further, adding/removing features, or testing different algorithms to achieve better results
10. End.

PROGRAM:

```
library(randomForest)
library(Metrics)

data <- read.csv("C:/Users/AI_LAB/Desktop/77/stock.csv")
if (is.null(data)) {
  stop("Data not loaded. Please check the file path.")
}
str(data)

data$Closing.Volume <- as.numeric(as.character(data$Closing.Volume)) # Update based on
your target variable
data <- na.omit(data)
```

```

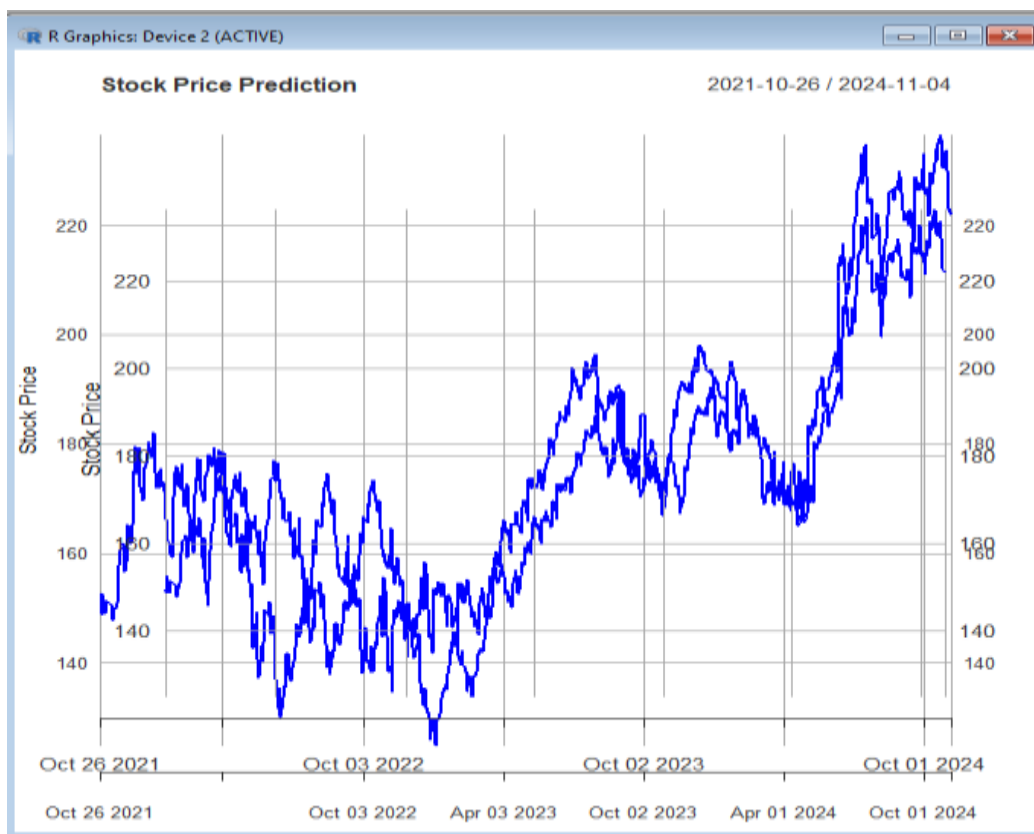
set.seed(123)
train_index <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
rf_model <- randomForest(Closing.Volume ~ ., data = train_data, ntree = 100)
predictions <- predict(rf_model, newdata = test_data)

actuals <- test_data$Closing.Volume
mae <- mean(abs(predictions - actuals))
rmse <- sqrt(mean((predictions - actuals)^2))
cat("Mean Absolute Error:", mae, "\n")
cat("Root Mean Squared Error:", rmse, "\n")

plot(test_data$Date, actuals, type = 'l', col = 'blue', ylim = range(c(actuals, predictions)),
      xlab = 'Date', ylab = 'Closing Price', main = 'Actual vs Predicted Closing Prices')
lines(test_data$Date, predictions, col = 'red')
legend("topright", legend = c("Actual", "Predicted"), col = c("blue", "red"), lty = 1)

```

OUTPUT:



RESULT:

Thus the R program to Tuning stock models for better performance is executed successfully and the output is verified.