RAJALAKSHMI ENGINEERING COLLEGE

An AUTONOMOUS Institution

Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENTOFARTIFICIAL INTELLIGENCE AND MACHINE LEAR NING

AI19542 – DATA SCIENCE USING R

LAB MANUAL

THIRDYEAR

FIFTH SEMESTER

2024 -2025

ODDSEMESTER

Ex No:1 Da te:	BasicsofR–datatypes,vectors,factors,listanddata fra mes

AIM:

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

AL GO RIT HM :

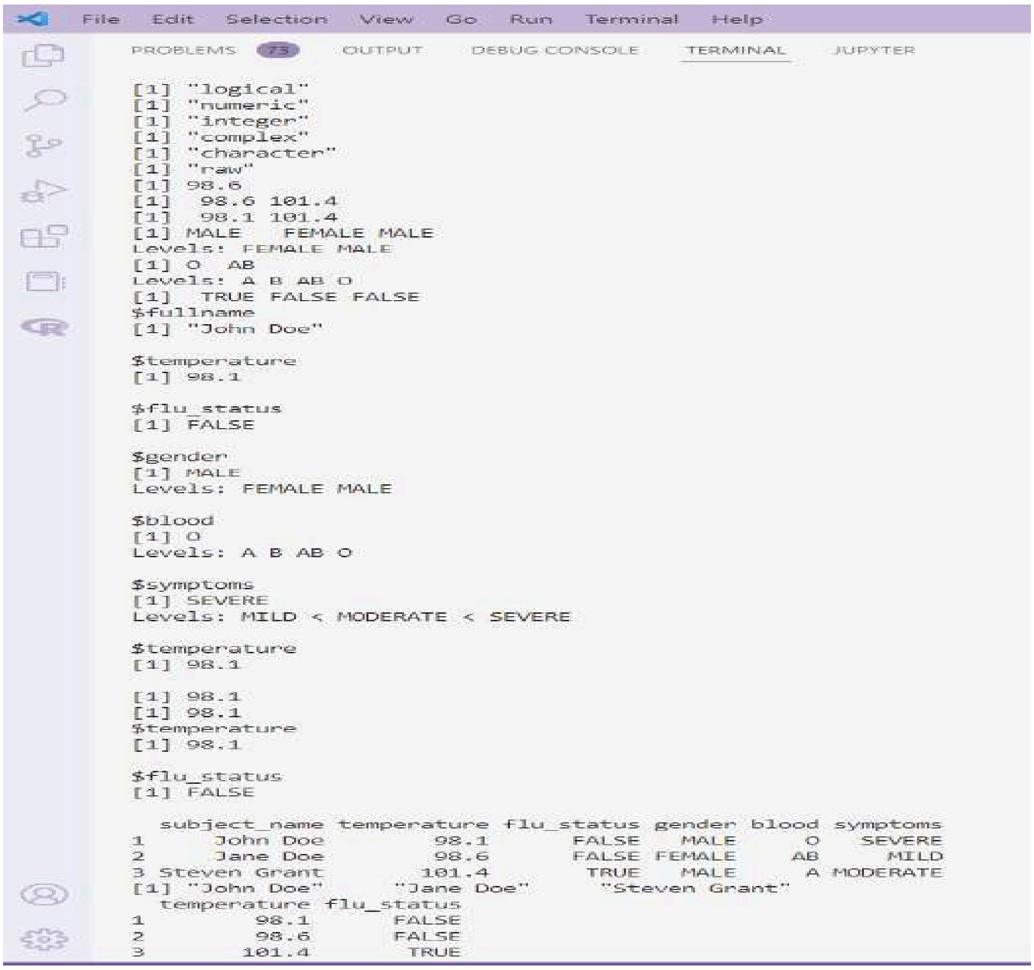
1. Start
2. Assignvaluesinlogical,numerical,character,complexandcharacterinrawformtoa variable v.
3. Print the classofv.
4. AssignavectorforsubjectNames,temperatureandflu_statusforthreepatientsusing c()functionandaccess theelements.
5. Createafactorusingfactor()withduplicatevaluesandassignlevelwithdistinct values.
6. Displaythespecificelementand checkforcertainvaluesinfactor.
7. Createalistusinglist()fromthepatientdetailsandaccessthemultiple elements.
8. Createadataframeusingdata.frame()withmultiplevectorsasfeatures.Accessthe el ement s.
9. Createamatrixusing matrix() withdifferent allocationsand access the elements.
10. Stop.

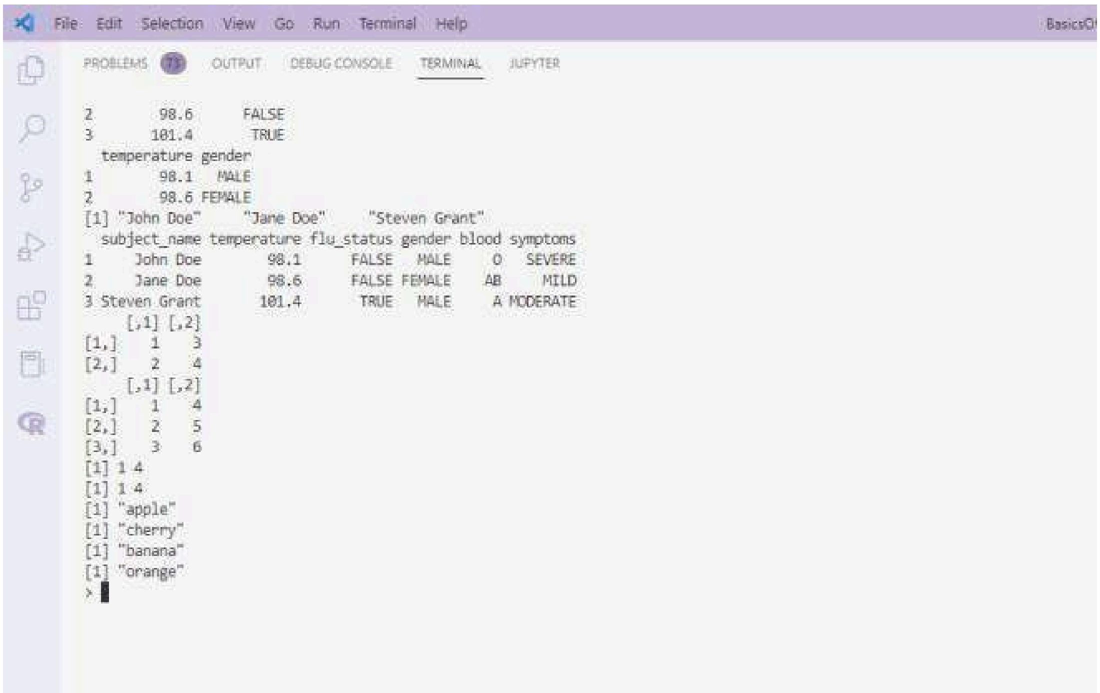
PROGRAM:

```
#DataTypes
v <-T RU E
print(class(v))
v<-23.5
print(class(v))
v<-2L
print(class(v))
v<-2+5i
print(class(v))
v<-"TRUE"
pri nt(cl ass(v) )
v<-charToRaw("Hello")
pri nt(cl ass(v) )

#Vectors
subject_name<-c("JohnDoe","Jane Doe","StevenGrant")
te mperat ure<-c(98.1,98.6,101.4)
flu_status<-c(FALSE,FALSE,TRUE)
te mperat ure[2]
temperature[2:3]
te mper atur e[-2]
#Fact ors
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"),
        le ve ls= c(" MIL D" , "MO DE RAT E ", "S E VE R E "),
        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])
subjec t1
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_statu
s")]
pt_dat a[c(1,2) ,c (2,4)]
pt_data[,1] pt_da ta[ ,]

#Matrices    m<-matrix(c(1,2,3,4),ncol=2)    pri    nt(m    )    m<-
matrix(c(1,2,3,4,5,6),nrow=3)
pri nt(m )
pri nt(m [1,])
pri nt(m [1,])
thismatrix<-matrix(c("apple","banana","cherry","orange"),nrow   =2,ncol=2)
for(rows in1:nrow(thismatrix)){


 for (columnsin1:ncol(thismatrix)){
  pri nt(t hism atri x[rows, columns] )
 }
}
```

OU TP UT:





Re su lt :

ThustheRScriptprogramtoimplementvariousdatatypes,vectors,factors, listsanddata framesisexecutedsuccessfullyandtheoutputisverified.

Ex no: 2 Dat e:	Diagnosis of Breast Cancer using KNN.

Aim:

ToimplementaRprogramtopredictanddiagnoseBreastCancerusingKNN algorithm.

Al gorit hm :

1. Start
2. Read the csv file from the directory and store it in bcd variable.
3. Dropthe first column id.
4. Change the diagnosis feature withcategorical values BandM ina factor
5. Normalizethe dataset.
6. Splitthedatasetfortraining andtesting,withdiagnosisastheresponsevariable and the restasthepredictorvariables.
7. Importthe library“class”for knnclassification.
8. Predicttheknnmodelusingknn()with5clusterswiththecorrespondingtrainingand testing data.
9. Displaythe confusion matrixand accuracyofthe 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"))
norm ali ze<-funct ion(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]
li brary( class)
y_pred<-knn(train=x_train,test=x_test,cl=y_train,k=5)
tb l =ta bl e(x =y _ test , y=y _ pred )
tbl
accuracy = sum(diag(tbl))
```

OU TP UT:

PROBLEMS 37 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
'data.frame':   569 obs. of  32 variables:
 $ id               : int  87139402 8910251 905520 868871 9012568 906539 925291 87880 862989 89827 ...
 $ diagnosis        : chr  "B" "B" "B" "B" ...
 $ radius_mean      : num  12.3 10.6 11 11.3 15.2 ...
 $ texture_mean     : num  12.4 18.9 16.8 13.4 13.2 ...
 $ perimeter_mean   : num  78.8 69.3 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  0.00805 0.00747 0.00516 0.01127 0.00501 ...
 $ compactness_se   : num  0.0118 0.03581 0.00936 0.03498 0.01485 ...
 $ 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  549 425 471 434 819 ...
 $ 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"
> 
```

Re su lt :

Thus theR Scriptprogramto implementdiagnosisofBreastCancerusing K-Nearest Neighbour algorithm is executed successfully and the output is verified.

Ex No: 3 Dat e:	Filtering Mobilephonespamusing NaïveBayes

AIM:

ToimplementaRprogramtoFilterMobilephonespamusing NaïveBayes.

ALGORITHM :

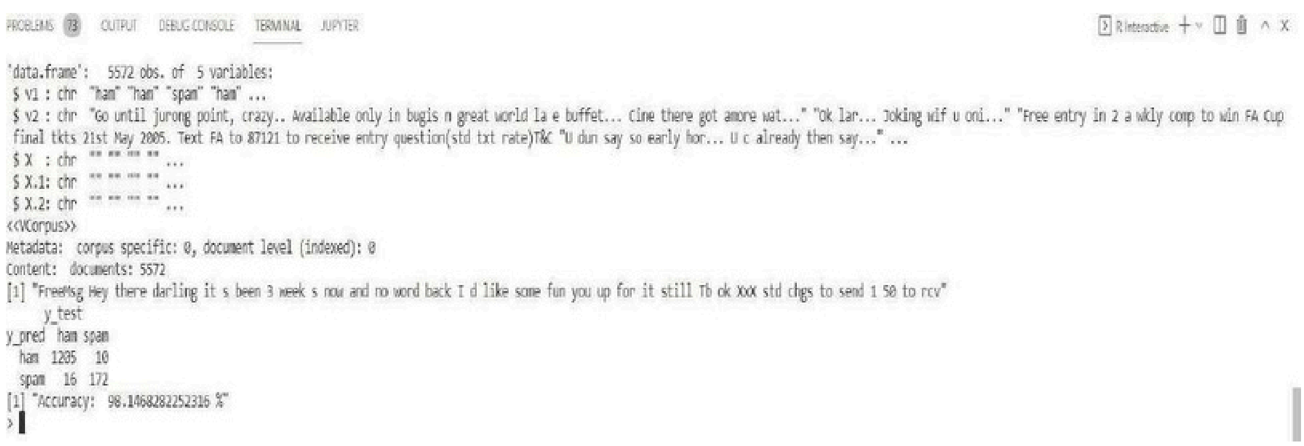
1. Start
2. Importthecsvfileandstorethedataframein“Sms”.Haveaglimpseatthestructure of the data frame.
3. Remove the unneccesary columns which is from column 3 to 5.
4. Convertthe labelsasfactors.
5. Remove special characters from the dataset and retain only alpha numeric characters usingalnumin str_replace_all()from“stringr”package.
6. Create a volatile corpus VCorpus() for text mining from the source object of “v2” which is extracted using VectorSource() .
7. CreateaDocumentTermMatrix()tosplittheSMSmessageintoindividual C omponents.
8. Createtrainingandtesting dataset withthe split ratio 0.75.
9. Find thefrequenttermswhichappearforatleast5timesinDocumentTermMatrixin 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))
```

```r
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,"%"))
```

OU TP UT:



RE S ULT:

ThustheR programtoimplementfilteringofMobilephonespamusingNaïve Bayes is executed successfully and the output is verified.

Ex No:4 Dat e:	Risky Bank Loans using Decision Trees

AIM:

ToimplementaRprogramtofindRiskyBankloansusing DecisionTree.

ALGORITHM :

1. Start
2. Import the dataset credit.csv and display the structure of the dataset.
3. Displaythetable to find the rangeofvaluesand findthe missing values.
4. Factorise thedefault columnand set seedof123.
5. Splitthedatasetfortrainingandtestingintheratioof0.8,with“default”asthe responsevariable,andthe restaspredictorvariables.
6. Importthe libraryC5.0for implementingdecisiontree.
7. Trainthedecisiontree modelusingC5.0functionforthetrainingdataset.
8. Testthemodeltopredictusingpredict().Printthe confusionmatrix.
9. Print the accuracyofthe decisiontreemodel.
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)

st r(tr ain_sam ple)

x_train <- credit[train_sample, -17]

x_test <- credit[-train_sample, -17]

y_train <- credit[train_sample, 17]

y_test <- credit[-train_sample, 17]

li brary( 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)

pri nt(past e(" Accuar acy: ",a cc*100,"% ") )
```

OU TP UT:

```
> credit <- read.csv("E:\\Academic Docs\\Semester-5\\Data Science using R\\cre$
> str(credit)
'data.frame':   1000 obs. of  17 variables:
 $ checking_balance    : chr  "< 0 DM" "1 - 200 DM" "unknown" "< 0 DM" ...
 $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
 $ credit_history      : chr  "critical" "good" "critical" "good" ...
 $ purpose             : chr  "furniture/appliances" "furniture/appliances" "education" "furniture/appliances" ...
 $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
 $ savings_balance     : chr  "unknown" "< 100 DM" "< 100 DM" "< 100 DM" ...
 $ employment_duration : chr  "> 7 years" "1 - 4 years" "4 - 7 years" "4 - 7 years" ...
 $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
 $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
 $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
 $ other_credit        : chr  "none" "none" "none" "none" ...
 $ housing             : chr  "own" "own" "own" "other" ...
 $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
 $ job                 : chr  "skilled" "skilled" "unskilled" "skilled" ...
 $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
 $ phone               : chr  "yes" "no" "no" "no" ...
 $ default             : chr  "no" "yes" "no" "no" ...
> tb <- table(credit$savings_balance)
> summary(credit$amount)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    250    1366    2320    3271    3972   18424
> credit$default <- factor(credit$default)
> set.seed(123)
> train_sample <- sample(1000, 900)
> str(train_sample)
 int [1:900] 415 463 179 526 195 938 818 118 299 229 ...
> 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)

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


C5.0 [Release 2.07 GPL Edition]         Wed Oct 26 11:59:18 2022
-------------------------------

Class specified by attribute `outcome'
```

```
Decision tree:

checking_balance in {unknown,> 200 DM}: no (412/54)
checking_balance in {< 0 DM,1 - 200 DM}:
:...credit_history in {very good,perfect}:
    :...housing = rent: yes (16/1)
    :   housing = other:
    :   :...employment_duration in {1 - 4 years,4 - 7 years,
    :   :   :                           > 7 years}: yes (10)
    :   :   employment_duration in {< 1 year,unemployed}:
    :   :   :...job = skilled: yes (1)
    :   :       job in {management,unskilled,unemployed}: no (3)
    :   housing = own:
    :   :...purpose in {business,education}: no (9/2)
    :       purpose in {renovations,car0}: yes (3)
    :       purpose = car:
    :       :...months_loan_duration <= 18: yes (5)
    :       :   months_loan_duration > 18:
    :       :   :...job in {skilled,management,unemployed}: no (3)
    :       :       job = unskilled: yes (1)
    :       purpose = furniture/appliances:
    :       :...other_credit = store: no (4)
    :           other_credit in {none,bank}:
    :           :...existing_loans_count <= 1: yes (6)
    :               existing_loans_count > 1: no (2)
    credit_history in {good,critical,poor}:
    :...months_loan_duration <= 15:
        :...purpose in {business,car0}: no (10)
        :   purpose = furniture/appliances:
        :   :...savings_balance in {unknown,< 100 DM,500 - 1000 DM,
        :   :   :                           > 1000 DM}: no (88/14)
        :   :   savings_balance = 100 - 500 DM: yes (3)
        :   purpose = education:
        :   :...savings_balance = unknown: no (3)
        :   :   savings_balance in {< 100 DM,100 - 500 DM,500 - 1000 DM,
        :   :                           > 1000 DM}: yes (7/1)
        :   purpose = renovations:
        :   :...other_credit in {none,bank}: no (5/1)
        :   :   other_credit = store: yes (1)
        :   purpose = car:
        :   :...credit_history in {critical,poor}: no (27/4)
        :       credit_history = good:
        :       :...existing_loans_count > 1: yes (2)
```

```
:   :   :...checking_balance = < 0 DM: yes (4)
:   :       checking_balance = 1 - 200 DM: no (3/1)
:   purpose = furniture/appliances:
:   :...savings_balance in {100 - 500 DM,
:       :                   500 - 1000 DM}: yes (6)
:       savings_balance = < 100 DM:
:       :...months_loan_duration <= 22: yes (12/1)
:           months_loan_duration > 22:
:           :...amount <= 2325: yes (3)
:               amount > 2325: no (6)
employment_duration = 4 - 7 years:
:...savings_balance in {100 - 500 DM,
:   :                   500 - 1000 DM}: no (8)
:   savings_balance = < 100 DM:
:   :...job in {management,unskilled,
:       :       unemployed}: no (6)
:       job = skilled:
:       :...dependents > 1: no (3/1)
:           dependents <= 1:
:           :...months_loan_duration <= 22: no (3)
:               months_loan_duration > 22: yes (8)
employment_duration = > 7 years:
:...other_credit = store: no (2)
    other_credit = bank:
    :...job in {skilled,unemployed}: yes (6)
    :   job in {management,unskilled}: no (4/1)
    other_credit = none:
    :...purpose = business: yes (2)
        purpose in {education,renovations,
        :           car0}: no (1)
        purpose = furniture/appliances:
        :...job in {skilled,unskilled,unemployed}: no (9)
        :   job = management: yes (2)
        purpose = car:
        :...amount <= 6999: no (2/1)
            amount > 6999: [S1]

SubTree [S1]

checking_balance = < 0 DM: no (1)
checking_balance = 1 - 200 DM: yes (3)
```

```
Evaluation on training data (900 cases):

	    Decision Tree
	  ----------------
	  Size      Errors

	    69   99(11.0%)   <<


	   (a)   (b)    <-classified as
	  ----  ----
	   625    30    (a): class no
	    69   176    (b): class yes


	Attribute usage:

	100.00% checking_balance
	 54.22% credit_history
	 48.22% months_loan_duration
	 42.22% savings_balance
	 31.00% purpose
	 22.33% employment_duration
	  9.22% years_at_residence
	  8.78% housing
	  8.44% job
	  6.11% other_credit
```

```
	  5.78% amount
	  4.89% existing_loans_count
	  4.22% phone
	  2.89% percent_of_income
	  1.56% dependents
	  0.78% age


Time: 0.0 secs

> y_pred <- predict(model,x_test)
> cm <- table(y_pred,y_test)
> print(cm)
      y_test
y_pred no yes
   no  55  20
   yes 10  15
> acc<-sum(diag(cm))/sum(cm)
> print(paste("Accuaracy: ",acc*100,"%"))
[1] "Accuaracy:  70 %"
>
```

RESULT:

Thus theRprogramtofindRiskyBankloansusingDecisionTreeisexecuted successfully and the output is verified.

Ex No: 5 Dat e:	MedicalExpense withLinear Regression.

AIM:

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

AL GO RIT HM :

1. Start
2. Loadthe Insurance dataset and analyse the structureofthedataset.
3. Getthesummarystatistics.Checkwhetherthedistributionisright-skewed orleft skewed by comapring the mean and median. Verify the same using histogram.
4. Checkthe distributionof“region”usingtable.
5. Createacorrelation matrixof“age”, “bmi”, “children”, “expenses”.
6. Todetermine thepatternofthedataset,use scatterplotusingpairs()for“age”,“bmi”, “children”, “expenses”.
7. Todisplaya more informative scatterplot usepairs.panel() from“psych” library.
8. Fit the linear regression model using lm() with expenses as the dependent variable.
9. Evaluatethe modelperformance using summary().
10. Toimprovethemodelperformance,square theagevariableasage2andbmi30is1if bmi>=30else 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)

st r(ins ur ance)

summary(insurance$expenses)
hist (i nsurance$expenses)

ta bl e(i nsurance$regi on)

cor( insurance[ c("a ge","bm i" ,"chil dren" ," expenses" )])

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

library(psych)

pairs.panels(insurance[c("age","bmi","children","expenses")])8

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

summary(ins_model)

insurance$age2 <- insurance$age^2

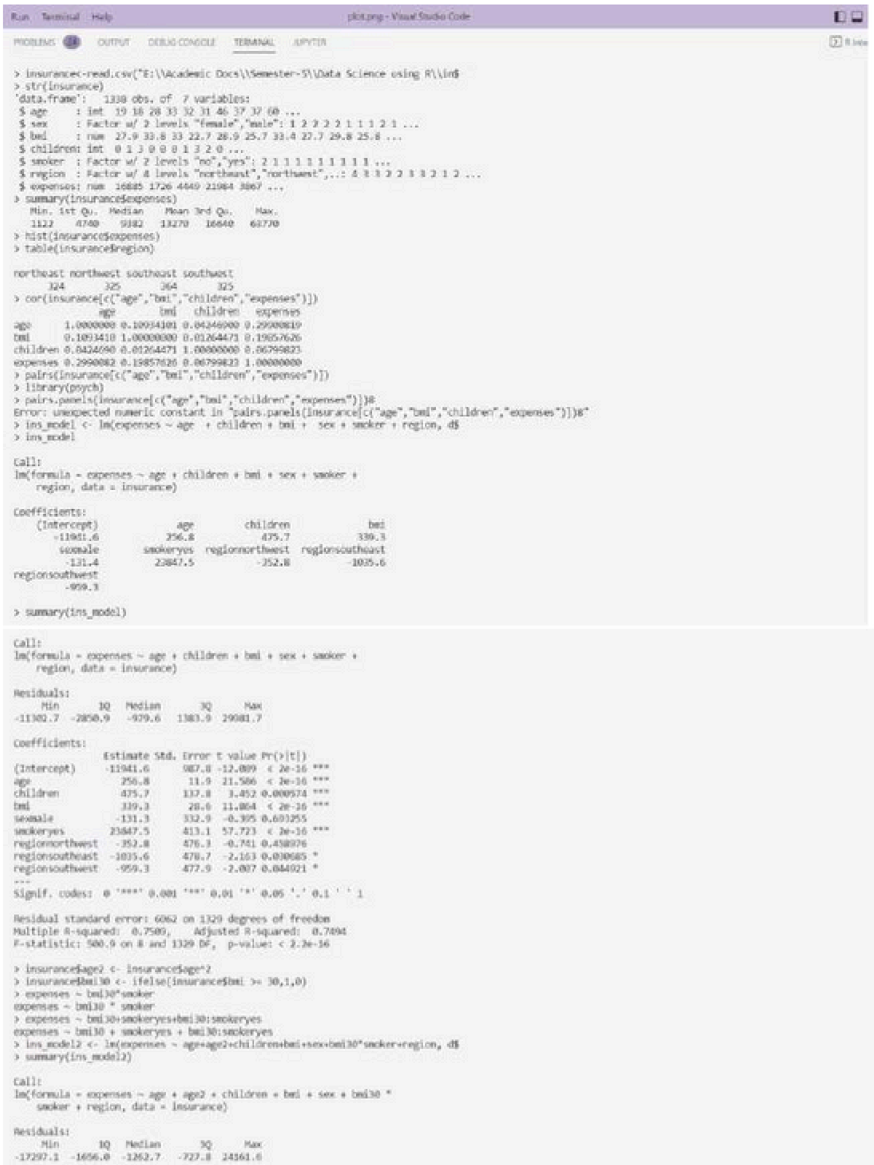
insurance$bmi30 <-ifelse(insurance$bmi>= 30,1,0)

expenses ~bmi30*smoker

expenses ~ bmi30+smokeryes+bmi30:smokeryes

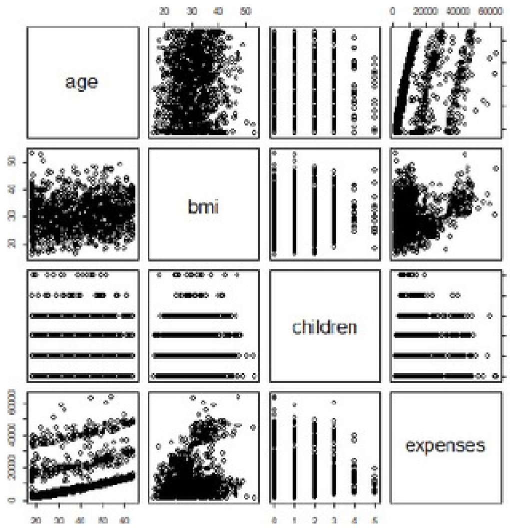
ins_model2<-lm(expenses ~age+age2+children+bmi+sex+bmi30*smoker+region,
data =i nsurance)
summary(ins_model2)

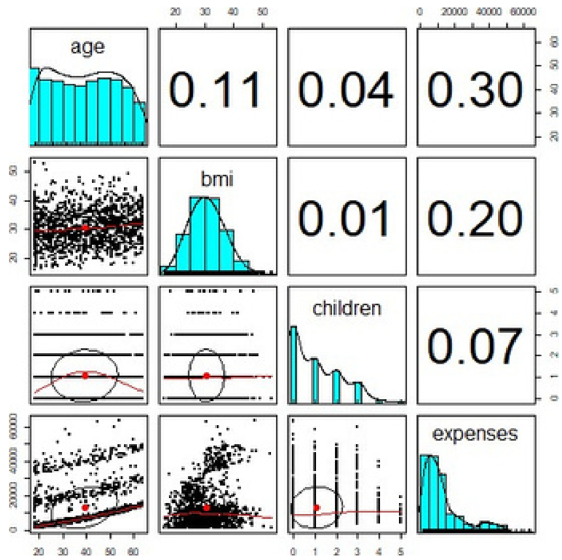
OU TP UT:



```
> insurance<-read.csv("E:\\Academic Docs\\Semester-5\\Data Science using R\\in$
> str(insurance)
'data.frame':   1338 obs. of  7 variables:
 $ age     : int  19 18 28 33 32 31 46 37 37 60 ...
 $ sex     : Factor w/ 2 levels "female","male": 1 2 2 2 2 1 1 1 2 1 ...
 $ bmi     : num  27.9 33.8 33 22.7 28.9 25.7 33.4 27.7 29.8 25.8 ...
 $ children: int  0 1 3 0 0 0 1 3 2 0 ...
 $ smoker  : Factor w/ 2 levels "no","yes": 2 1 1 1 1 1 1 1 1 1 ...
 $ region  : Factor w/ 4 levels "northeast","northwest",..: 4 3 3 2 2 3 3 2 1 2 ...
 $ expenses: num  16885 1726 4449 21984 3867 ...
> summary(insurance$expenses)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1122    4740    9382   13270   16640   63770
> hist(insurance$expenses)
> table(insurance$region)

northeast northwest southeast southwest
      324       325       364       325
> cor(insurance[c("age","bmi","children","expenses")])
               age        bmi   children   expenses
age      1.0000000 0.10934101 0.04246900 0.29900819
bmi      0.1093410 1.00000000 0.01264471 0.19857626
children 0.0424690 0.01264471 1.00000000 0.06799823
expenses 0.2990082 0.19857626 0.06799823 1.00000000
> pairs(insurance[c("age","bmi","children","expenses")])
> library(psych)
> pairs.panels(insurance[c("age","bmi","children","expenses")])8
Error: unexpected numeric constant in "pairs.panels(insurance[c("age","bmi","children","expenses")])8"
> ins_model <- lm(expenses ~ age  + children + bmi +  sex + smoker + region, d$
> ins_model

Call:
lm(formula = expenses ~ age + children + bmi + sex + smoker +
    region, data = insurance)

Coefficients:
    (Intercept)              age         children              bmi
       -11941.6            256.8            475.7            339.3
        sexmale        smokeryes  regionnorthwest  regionsoutheast
         -131.4          23847.5           -352.8          -1035.6
regionsouthwest
         -959.3

> summary(ins_model)

Call:
lm(formula = expenses ~ age + children + bmi + sex + smoker +
    region, data = insurance)

Residuals:
     Min       1Q   Median       3Q      Max
-11302.7  -2850.9   -979.6   1383.9  29981.7

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -11941.6      987.8 -12.089  < 2e-16 ***
age                256.8       11.9  21.586  < 2e-16 ***
children           475.7      137.8   3.452 0.000574 ***
bmi                339.3       28.6  11.864  < 2e-16 ***
sexmale           -131.3      332.9  -0.395 0.693255
smokeryes        23847.5      413.1  57.723  < 2e-16 ***
regionnorthwest   -352.8      476.3  -0.741 0.458976
regionsoutheast  -1035.6      478.7  -2.163 0.030685 *
regionsouthwest   -959.3      477.9  -2.007 0.044921 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6062 on 1329 degrees of freedom
Multiple R-squared:  0.7509,    Adjusted R-squared:  0.7494
F-statistic: 500.9 on 8 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 * smoker
> expenses ~ bmi30+smokeryes+bmi30:smokeryes
expenses ~ bmi30 + smokeryes + bmi30:smokeryes
> ins_model2 <- lm(expenses ~ age+age2+children+bmi+sex+bmi30*smoker+region, d$
> summary(ins_model2)

Call:
lm(formula = expenses ~ age + age2 + children + bmi + sex + bmi30 *
    smoker + region, data = insurance)

Residuals:
     Min       1Q   Median       3Q      Max
-17297.1  -1656.0  -1262.7   -727.8  24161.6
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

    smoker + region, data = insurance)

Residuals:
     Min       1Q   Median       3Q      Max
-17297.1  -1656.0  -1262.7   -727.8  24161.6

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        139.0053  1363.1359   0.102 0.918792
age                -32.6181    59.8250  -0.545 0.585600
age2                 3.7307     0.7463   4.999 6.54e-07 ***
children           678.6017   105.8855   6.409 2.03e-10 ***
bmi                119.7715    34.2796   3.494 0.000492 ***
sexmale           -496.7690   244.3713  -2.033 0.042267 *
bmi30             -997.9355   422.9607  -2.359 0.018449 *
smokeryes        13404.5952   439.9591  30.468  < 2e-16 ***
regionnorthwest   -279.1661   349.2826  -0.799 0.424285
regionsoutheast   -828.0345   351.6484  -2.355 0.018682 *
regionsouthwest  -1222.1619   350.5314  -3.487 0.000505 ***
bmi30:smokeryes  19810.1534   604.6769  32.762  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4445 on 1326 degrees of freedom
Multiple R-squared:  0.8664,    Adjusted R-squared:  0.8653
F-statistic: 781.7 on 11 and 1326 DF,  p-value: < 2.2e-16

>
```





RESULT:

ThustheRprogramtopredictmedicalexpensesusinglinearregressionis executed successfully and the output is verified.

Ex No: 6 Dat e:	Modeling strength of concrete.

AIM:

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

AL GO RIT HM :

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 >= 30) and re-evaluate.
8. Stop

PROGRAM:

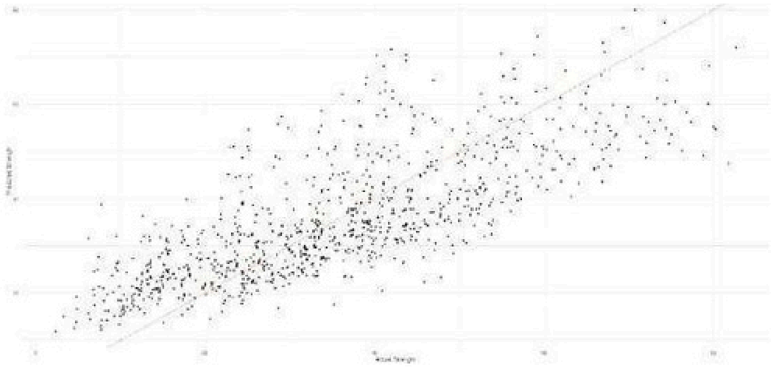
```
li brary( caret )

li brary( ggpl ot2)

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

head(data)

sum( is.na(dat a))

set .seed(123)

trainIndex <- createDataPartition(data$CompressiveStrength, p =

0.8, list = FALSE)

trainData <- data[trainIndex,]

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

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

sum mar y(mode l)

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()
```

OU TP U T:

```
> 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 0 0 ...
 $ water           : num  162 162 228 228 192 228 228 228 228 228 ...
 $ superplastic    : num  2.5 2.5 0 0 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 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          0.109746   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 theR Scriptprogramto implementModeling strength of concreteisexecuted successfully and the output is verified.

Ex No: 7 Dat e:	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.

AL GO RIT HM :

1. Start
2. Load Data: Load the transaction dataset (assume each transaction is a list of items purcha sed).
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 al gorit hm.
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 com binat ions .
8. Stop

PROGRAM:

```
if(!require(arules)) install.packages("arules", dependencies=TRUE)
li brar y(arul es)
dat a(" Groc eri es")
sum mar y(Gr oceri es)
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 =
mi n_confi dence))
sum mar y(fr equent_i te mset s)
inspect(frequent_itemsets[1:10])
rules <- apriori(Groceries, parameter = list(supp = min_support, conf = min_confidence,
target = "rules"))
sum mar y(rul es)
inspect(sort(rules, by = "confidence")[1:10]) # Display top 10 rules by confidence
if(!require(arulesViz)) install.packages("arulesViz", dependencies=TRUE)
li brar y(arul esViz )
plot(rules, method = "graph", control = list(type = "items"))
```

OU TP U T:

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

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

RE S ULT :

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

Ex No: 8 Dat e:	Finding Teen Segments of Market.

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.

AL GO RIT HM :

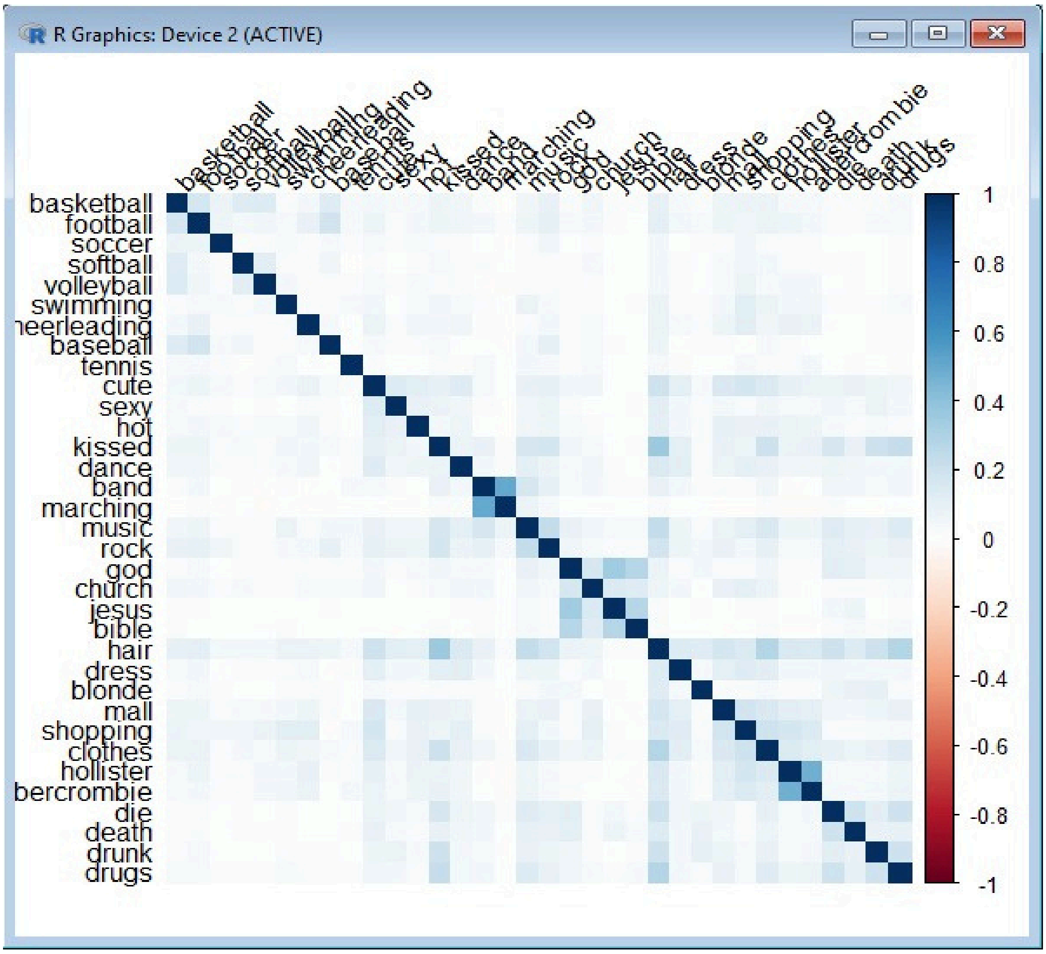
1. START: Collect raw data from sources relevant to the teen market (e.g., social media data, survey responses).
2. PREPROCESSING: Clean the data (e.g., remove missing values, correct errors).
3. SELECT FEATURES: Choose features that help in segmentation (e.g., age, purchase patterns, interests).
4. 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 segm ent.
8. STOP: Develop targeted marketing strategies based on the insights from the segm enta ti on.
9. This approach allows businesses to better understand the teen market and tailor their products or marketing campaigns accordingly.

PROGRAM:

```
li brar y(dplyr )
li brar y(ggpl ot2)
li brar y(corr plot )
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)
}
ma in('pat h_to_your _fi le .csv')
```

OUTPUT:



RESULT:

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

Ex No: 9 Dat e:	Tuning stock models for better performance.

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.

AL GO RIT HM :

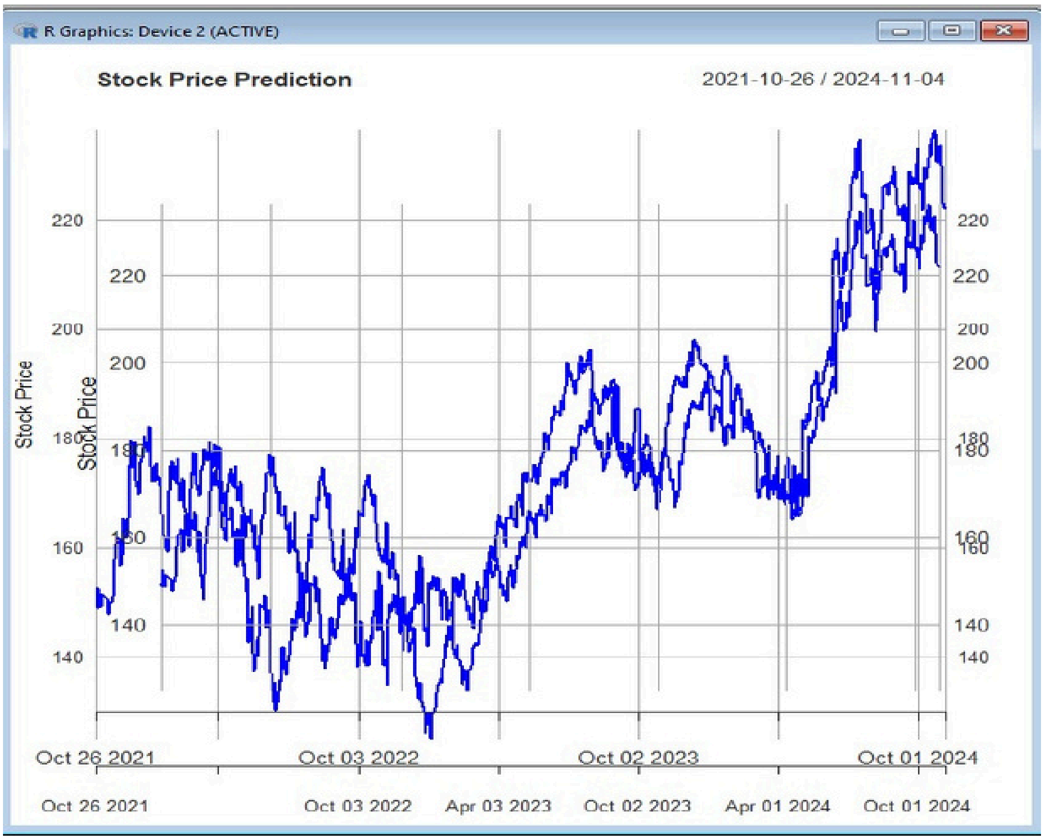
1. Start
2. Data Collection: Gather historical stock data (e.g., price, volume, market sentiment, technical indicators).
3. Data Preprocessing: Clean the data by handling missing values, normalizing features, and creating relevant indicators (e.g., moving averages, RSI).
4. Feature Engineering: Create 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:

```
li brary( random For est)
li brary( Metr ics)
data <- read.csv("C:/Users/AI_LAB/Desktop/77/stock.csv")
if (is.null(data)) {
 stop("Data not loaded. Please check the file path.")
}
st r(dat a)
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)
```

OU TP U T:



RE S ULT:

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