INDEX

Week No.	Name of the Experiment	Page	Date of Experiment	Date of Submission	Faculty Sign
	PART A: DATA MINING				
1	Demonstration of preprocessing on dataset student.arff	1 – 7	18/7/25	25/7/25	
2	Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm.	8 – 13	25/7/25	1/8/25	
3	Demonstration of classification rule process on dataset employee.arff using j48 algorithm.	14 – 22	1/8/25	8/8/25	
4	Demonstration of classification rule process on dataset employee.arff using id3 algorithm.	23 – 28	8/08/25	22/8/25	
5	Demonstration of classification rule process on dataset employee.arff using naïve bayes algorithm.	29 – 35	22/08/25	29/8/25	
6	Demonstration of clustering rule process on dataset iris.arff using simple k-means.	36 – 41	29/8/25	19/9/25	
7	Demonstration of clustering rule process on dataset student.arff using hierarchical clustering.	42 – 48	19/9/25	19/9/25	
	PART A: DATA ANALYTICS				
8	a) Write R program to find R-Mean, Median & Mode with the sample data. b) Write R program to find Analysis and Covariance with the sample data and visualize the regression graphically.	49 – 61	26/9/25	26/9/25	

9	Write R program to find the following Regressions with the sample data and visualize the regressions graphically. a. Linear Regression b. Multiple Regression c. Logistic Regression d. Poisson Regression.	62 – 75	10/10/25	10/10/25	10/10
10	Write R program to find a) Time Series Analysis with the sample data and visualize the regression graphically. b) Non Linear Least Square with the sample data and visualize the regression graphically. c) Decision Tree with the sample data and visualize the regression graphically.	76 – 85	17/10/25	17/10/25	
11.	Write R program to find the following Distribution with the sample data and visualize the linear regression graphically. a) Normal Distribution b) Binomial Distribution	86 – 94	24/10/25	24/10/25	
12.	Write R program to do the following tests with the sample data and visualize the results graphically. a) X^2 test b) t-test c) F-test	95 – 106	31/10/25	31/10/25	

Scanned with OKEN Scanner

OUTPUT

Relation Declaration

@ relation student

Attribute Declaration

@ attribute age { <30, 30-40, >40}

@ attribute income { low, medium, high}

@ attribute student {yes, no}

@ attribute credit-rating {fair, excellent, poor}

@ attribute marks real

@ attribute buys pc{yes, no}

Instances

@ data

<30, high, no, fair, 80, no

30-40, high, no, fair, 65, yes

>40, medium, no, excellent, 90, no

<30, low, yes, fair, 85, yes

>40, medium, no, excellent, 90, yes

30-40, high, no, fair, 77, no

30-40, medium, yes, excellent, 78, no

>40, medium, no, excellent, 88, yes

30-40, low, yes, fair, 66, no

<30, low, yes, fair, 65, no

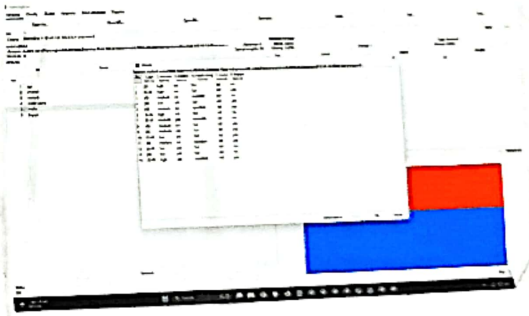
30-40, high, yes, excellent, 85, yes

filters

1) Discritize

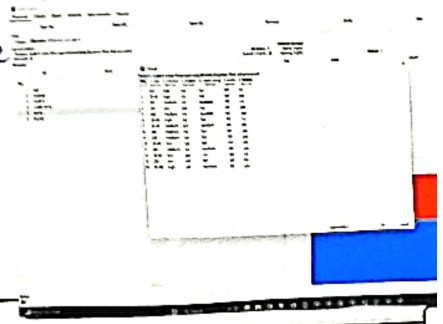
choose → weka → filters → supervised → attribute → descritize.

the process of converting a real value into an odinal attribute (Nominal or bins)



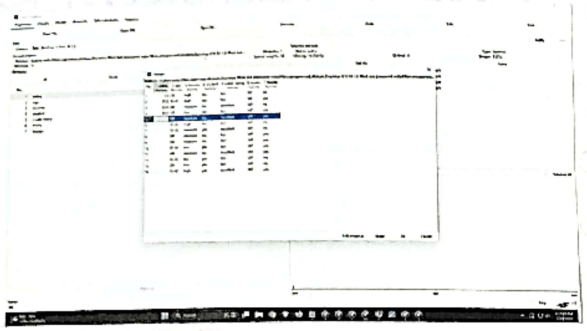
2) Discritize

choose → weka → filters → unsupervised → attribute → descritize



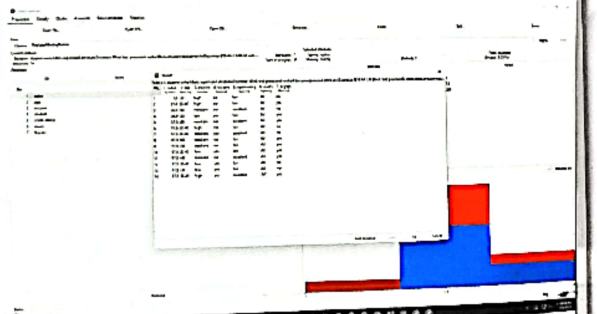
Scanned with OKEN Scanner

3) Add choose → weka → filters → unsupervised → attribute → add (No visualization).



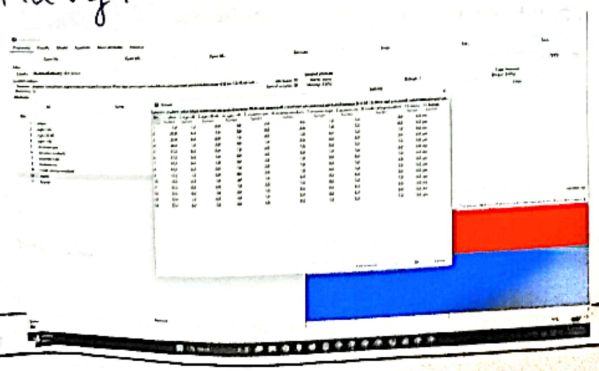
4) Replace missing value

choose → weka → filters → unsupervised → attribute → replace missing value



5) Nominal to binary

Choose → weka → unsupervised → attribute → nominal to binary.



VIVA QUESTIONS

1. What is the main goal of Data Mining?

Ans. To extract useful patterns, knowledge or insi. from large amount of data.

2. What are the Two Types of Data Mining Tasks?

Ans. Descriptive Tasks
Predictive Tasks

3. What are the Data Mining Functionalities?

Ans. Classification, digression, clustering, association rule mining, summarization.

4. What are the Features of WEKA?

Ans. Open source and free tool, user-friendly GUI supports data preprocessing

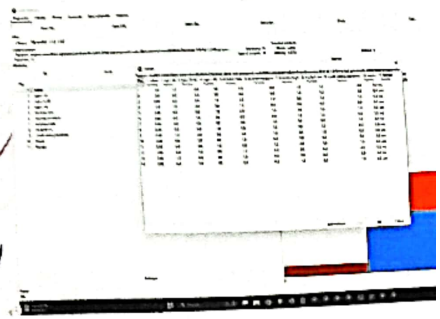
5. Navigate the options available in the WEKA.

Ans. 1) Explorer 2) Experimenter 3) knowledge flow, 4) Simple CLI 5) Workbench

6. What is ARFF file format?

Ans. Attribute Relation file format is a text file forma used in WEKA to represent datasets.

Normalization – The process of changing numeric values (pre-defined values) to (0 to 1)
(or)
converting large numerical value to (0 to 1) range



Scanned with OKEN Scanner

Week-2

Date: 25/7/24

AIM

This experiment illustrates some of the basic elements of asscociation rule mining using WEKA. The sample dataset used for this example is contactlenses.arff

Step-1: Open the data file in Weka Explorer. It is presumed that the required data fields have been discretized. In this example it is age attribute.

Step-2: Clicking on the associate tab will bring up the interface for association rule algorithm.

Step-3: We will use apriori algorithm. This is the default algorithm.

Step-4: Inorder to change the parameters for the run (example support, confidence etc) we click on the text box immediately to the right of the choose button.

DESCRIPTION

The Apriori Algorithm: Finding Frequent Itemsets Using Candidate Generation

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses *prior knowledge* of frequent itemset properties. Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are used to explore (*k*+1)-itemsets.

Apriori property: All nonempty subsets of a frequent itemset must also be frequent.

It has two steps.

1) Join step

2) Prune step

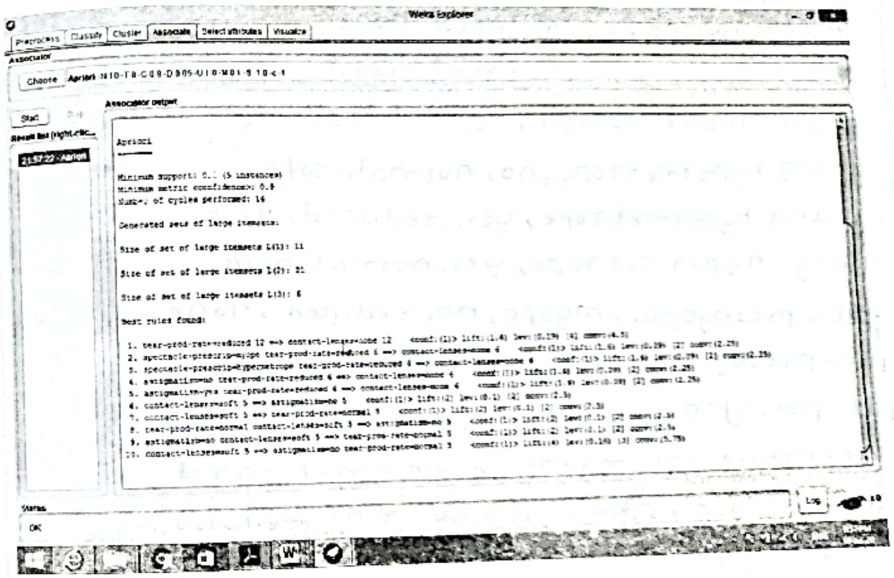
PROCEDURE

Step-1: Load appropriate dataset into weka

Step-2: Select associate tab and select apriori algorithm

1) Join step

2) Prune step



OUTPUT

```
@attr
@relation contact_lenses
@attribute age                    {young, pre-presbyopic,}
@attribute spectacle-prescrip     {myope, hypermetrope}
@attribute astigmatism            {no, yes}
@attribute tear-prod-rate         {reduced, normal}
@attribute contact-lenses         {soft, hard, none}
```

```
@data
%
%.24 instances
%
young, myope, no, reduced, none
young, myope, no, normal, soft
young, myope, yes, reduced, none
young, myope, yes, normal, hard
young, hypermetrope, no, reduced, none
young, hypermetrope, no, normal, soft
young, hypermetrope, yes, reduced, none
young, hypermetrope, yes, normal, hard
pre-presbyopic, myope, no, reduced, none
pre-presbyopic, myope, no, normal, soft
pre-presbyopic, myope, yes, reduced, none
pre-presbyopic, myope, yes, normal, hard
pre-presbyopic, hypermetrope, no, reduced, none
pre-presbyopic, hypermetrope, no, normal, soft
pre-presbyopic, hypermetrope, yes, reduced, none
pre-presbyopic, hypermetrope, yes, normal, none
presbyopic, myope, no, reduced, none
presbyopic, myope, no, normal, none
presbyopic, myope, yes, reduced, none
presbyopic, myope, yes, normal, hard
presbyopic, hypermetrope, no, reduced, none
presbyopic, hypermetrope, no, normal, soft
presbyopic, hypermetrope, yes, reduced, none
presbyopic, hypermetrope, yes, normal, none
```


Scanned with OKEN Scanner

Week-3

Date: 18/08/

AIM

This experiment illustrates the use of j-48 classifier in weka. The sample data set used in this experiment is "student" data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment:

Step-1: We begin the experiment by loading the data (student.arff)into weka.

Step-2: Next we select the "classify" tab and click "choose" button t o select the "j48"classifier.

Step-3: Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values. The default version does perform some pruning but does not perform error pruning.

Step-4: Under the "text" options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don't have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: We now click "start" to generate the model .the Ascii version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: Note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Step-7: Now weka also lets us a view a graphical version of the classification tree. This can be done by right clicking the last result set and selecting "visualize tree" from the pop-up menu.

Step-8: We will use our model to classify the new instances.

Step-9: In the main panel under "text" options click the "supplied test set" radio button and then click the "set" button. This wills pop-up a window which will allow you to open the file containing test instances.

DESCRIPTION

Classification is a **data mining** function that assigns items in a collection to target categories or classes. The goal of **classification** is to accurately predict the target class for each case in the **data**. For example, a **classification** model could be used to identify loan applicants as low, medium, or high credit risks.

Entropy characterizes the (im) purity of an arbitrary collection of examples

Information Gain is the expected reduction in entropy caused by partitioning the examples according to a given attribute

If we have a set with k different values in it, we can calculate the entropy as follows:

$$\text{entropy(Set)} = I(\text{Set}) = -\sum_{i=1}^{k} P(\text{valuei}).\log2(P(\text{valuei}))$$

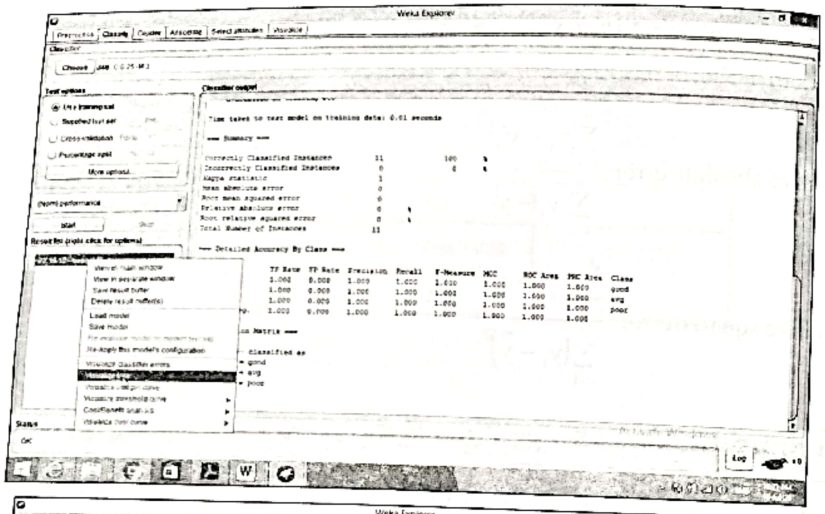
Where P(valuei) is the probability of getting the ith value when randomly selecting one from the set. So, for the set R = {a,a,a,b,b,b,b,b}

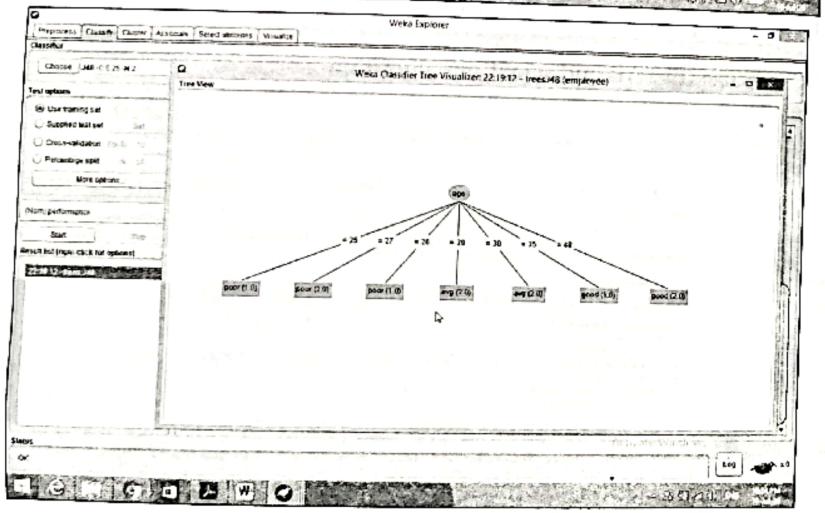
$$\text{entropy (R)} = I(R) = -[(3/8)\log2(3/8) + (5/8)\log2(5/8)]$$

Kappa statistic is a measure of how closely the instances classified by the machine learning classifier matched the **data** labeled as ground truth, controlling for the accuracy of a random classifier as measured by the expected accuracy.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**.

Scanned with OKEN Scanner





OUTPUT

@relation employee
@attribute age {25, 27, 28, 29, 30, 35, 48}
@attribute salary {10K, 15K, 17K, 20K, 25K, 30K, 32K, 35K}
@attribute performance {good, avg, poor}

@data
%
25, 10K, poor
27, 17K, poor
27, 17K, poor
28, 17K, poor
29, 20K, avg
30, 25K, avg
29, 25K, avg
30, 25K, avg
35, 32K, good
48, 32K, good
48, 32K, good
%

Scanned with OKEN Scanner

Week-4

Date: 8/03/25

AIM

This experiment illustrates the use of ID3 classifier in weka.the sample data set used in this experiment is "employee"data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment

Step-1: We begin the experiment by loading the data (employee.arff) into weka.

Step-2: Next we select the "classify" tab and click "choose" button to select the "j48"classifier.

Step-3: Now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values the default version does perform some pruning but does not perform error pruning.

Step-4: Under the "text "options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don't have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: We now click "start" to generate the model .the ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: Note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Scanned with OKEN Scanner

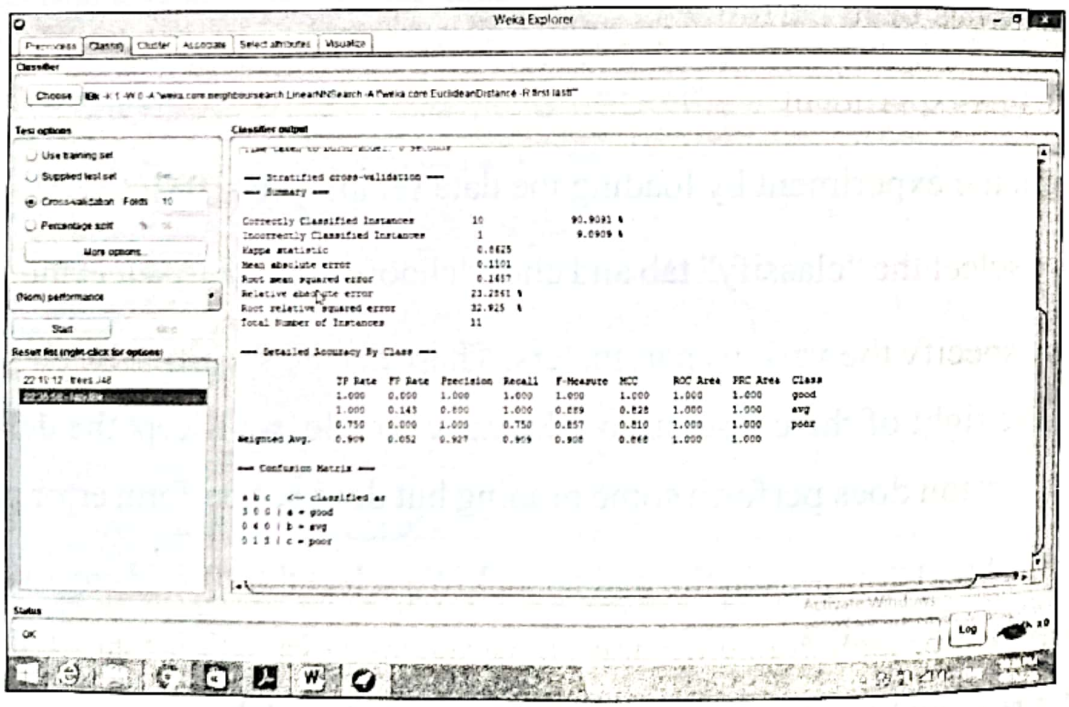
Step-7: ... done by right clicking the last result set and selecting "visualize tree" from the pop-... menu.

Step-8: We will use our model to classify the new instances.

Step-9: In the main panel under "text "options click the "supplied test set" radio button and the click the "set" button. This wills pop-up a window which will allow you to open the fi... containing test instances.

DESCRIPTION

Classification is a **data mining** function that assigns items in a collection to target categories or classes. The goal of **classification** is to accurately predict the target class for each case in the **data**. For example, a **classification** model could be used to identify loan applicants as low, medium, or high credit risks.



OUTPUT

Scanned with OKEN Scanner

Week-5

Date: 22/08/25

AIM

This experiment illustrates the use of naïve bayes classifier in weka. The sample data set used in this experiment is "employee"data available at arff format. This document assumes that appropriate data pre processing has been performed.

Steps involved in this experiment:

Step-1: We begin the experiment by loading the data (employee.arff) into weka.

Step-2: next we select the "classify" tab and click "choose" button to select the "id3"classifier.

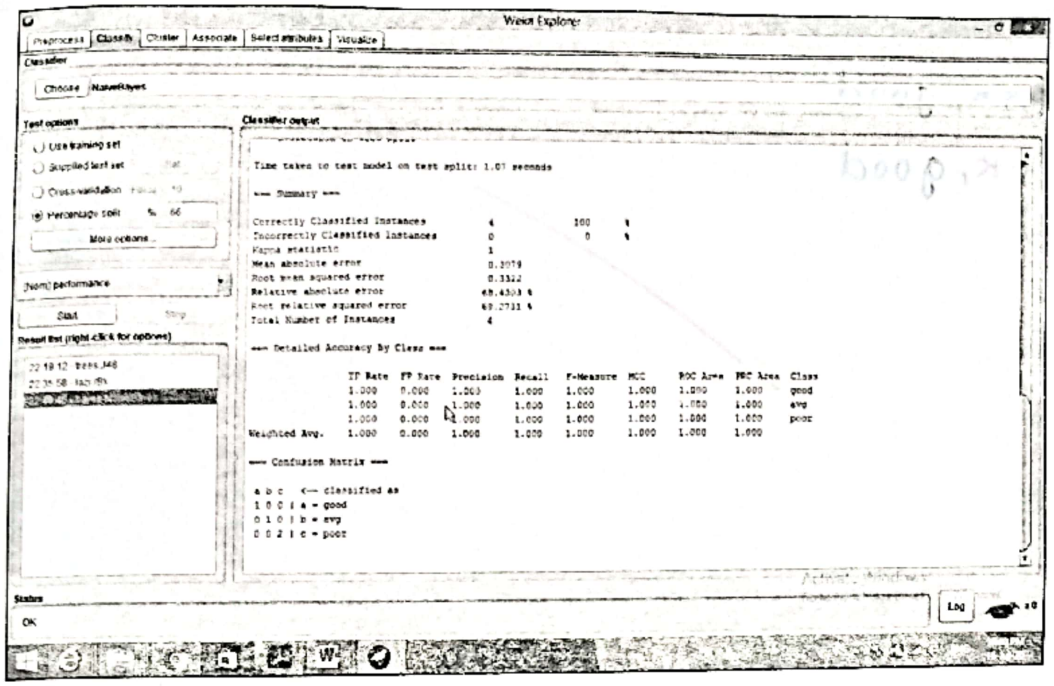
Step-3: now we specify the various parameters. These can be specified by clicking in the text box to the right of the chose button. In this example, we accept the default values his default version does perform some pruning but does not perform error pruning.

Step-4: under the "text "options in the main panel. We select the 10-fold cross validation as our evaluation approach. Since we don't have separate evaluation data set, this is necessary to get a reasonable idea of accuracy of generated model.

Step-5: we now click"start"to generate the model .the ASCII version of the tree as well as evaluation statistic will appear in the right panel when the model construction is complete.

Step-6: note that the classification accuracy of model is about 69%.this indicates that we may find more work. (Either in preprocessing or in selecting current parameters for the classification)

Scanned with OKEN Scanner



OUTPUT

dataset

@relation employee

@attribute age {25, 27, 28, 29, 30, 35, 38}

@attribute salary {10K, 15K, 17K, 20K, 25K, 30K, 32K, 35K}

@attribute performance {good, avg, poor}

@data

25, 10K, poor
27, 15K, poor
27, 17K, poor
28, 17K, poor
29, 20K, avg
30, 35K, avg
29, 25K, avg
30, 20K, avg
35, 32K, good

Scanned with OKEN Scanner

OUTPUT

Dataset (iris)

```
@RELATION iris
@ATTRIBUTE sepallength REAL
@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE petallength REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class {Iris-setosa, Iris-versicolor}
                 Iris-verginica
@DATA
5.1,3.5,1.4,0.2, Iris-setosa
4.9,3.0,1.4,0.2, Iris-setosa
4.7,3.2,1.3,0.2, Iris-setosa
4.6,3.1,1.5,0.2, Iris-setosa
5.7,2.9,4.2,1.3, Iris-versicolor
6.2,2.9,4.3,1.3, Iris-versicolor
5.1,2.5,3.0,1.1, Iris-versicolor
5.7,2.8,4.1,1.3, Iris-versicolor
6.3,3.3,6.0,2.5, Iris-virginica
6.3,2.5,5.0,1.9, Iris-virginica
6.5,3.0,5.2,2.0, Iris-virginica
6.5,3.0,5.2,2.0, Iris-virginica
6.2,3.4,5.4,2.3, Iris-virginica
5.9,3.0,5.1,1.8, Iris-virginica
%
%
%
%
```


Scanned with OKEN Scanner

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode

- Use training set
- Supplied test set
- Percentage split
- Classes to clusters evaluation
- (Nom) class
- Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

09:21:22 - SimpleKMeans

09:21:06 - SimpleKMeans

Clusterer output

```
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 7
Within cluster sum of squared errors: 62.1436882815797

Initial starting points (random):

Cluster 0: 6.1,2.6,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor

Missing values globally replaced with mean/mode

Final cluster centroids:
                                        Cluster#
Attribute               Full Data           0           1
                          (150.0)     (100.0)      (50.0)
==========================================================
sepallength                5.8433      6.262       5.006
sepalwidth                 3.054       2.872       3.418
petallength                3.7587      4.906       1.464
petalwidth                 1.1987      1.676       0.244
class               Iris-setosa Iris-versicolor  Iris-setosa


Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      100 ( 67%)
1       50 ( 33%)
```

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm

More

Capabilities

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	2
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

Open... Save... OK Cancel

Status

OK

Preprocess Classify Cluster Associate Select attributes Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode

- Use training set
- Supplied test set
- Percentage split
- Classes to clusters evaluation
- (Nom) class
- Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

09:22:22 - SimpleKMeans

09:21:06 - SimpleKMeans

Clusterer output

```
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Clustering model (full training set) ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 6.6136.9274690336

Initial starting points (random):

Cluster 0: 6.1,2.6,4.7,1.4,Iris-versicolor
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor
Cluster 2: 6.6,3.0,4.4,1.4,Iris-virginica
Cluster 3: 6.5,4.2,1.2,0.2,Iris-setosa

Missing values globally replaced with mean/mode

Final cluster centroids:
                                        Cluster#
Attribute               Full Data           0           1           2           3
                          (150.0)      (26.0)      (36.0)      (36.0)      (52.0)
==========================================================================================
sepallength                5.8433      6.4962      5.5361      6.5806      5.006
sepalwidth                 3.054       2.9769      2.5361      2.978       3.418
petallength                3.7587      4.6       3.9167      5.852       1.464
petalwidth                 1.1987      1.4423      1.2         2.026       0.244
class               Iris-setosa Iris-versicolor Iris-versicolor Iris-virginica Iris-setosa


Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       26 ( 17%)
1       36 ( 24%)
2       36 ( 24%)
3       52 ( 35%)
```

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

About

Cluster data using the k means algorithm

More

Capabilities

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose EuclideanDistance -R first-l
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	4
numExecutionSlots	1
preserveInstancesOrder	False
reduceNumberOfDistanceCalcsViaCanopies	False
seed	10

Open... Save... OK Cancel

Status

OK

Scanned with OKEN Scanner

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Clusterer

Choose HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

Cluster mode

- Use training set
- Supplied test set
- Percentage split
- Classes to clusters evaluation
- Store clusters for visualization

Ignore attributes

Start

Result list (right-click for options)

09:22:24 - SimpleKMeans

09:23:06 - SimpleKMeans

Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     student
Instances:    10
Attributes:   4
              CGPA
              Attendence
              StudyHours
              JobOffer
Test mode:    evaluate on training data

=== Clustering model (full training set) ===

Cluster 0
((0.0:0.17521,0.0:0.17521):0.15424,((0.0:0.18534,0.0:0.18534):0.02922,0.0:0.21457):0.11459)

Cluster 1
(((1.0:0.13514,1.0:0.13514):0.19432,(1.0:0.27027,1.0:0.27027):0.05918):0,1.0:0.32945)


Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       5 ( 50%)
1       5 ( 50%)
```

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Clusterer

Choose SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

Cluster mode

- Use training set
- Supplied test set
- Percentage split
- Classes to clusters evaluation
- Store clusters for visualization

Ignore attributes

Start

Result list (right-click for options)

09:22:24 - SimpleKMeans

09:23:06 - SimpleKMeans

Clusterer output

```
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Clustering model (full trai
```

Weka Clusterer Visualize: 09:23:06 - SimpleKMeans (iris)

X: Instance_number (Num) | Y: sepallength (Num)

Colour: Cluster (Nom) | Select Instance

Clear | Open | Save | Jitter

Plot: iris_clustered

7.9 / 6.1 / 4.3 — 0 / 74.5 / 149

Class colour

```
kMeans
======

Number of iterations: 4
Within cluster sum of squared e

Initial starting points (random

Cluster 0: 6.1,2.9,4.7,1.4,Iris
Cluster 1: 6.2,2.9,4.3,1.3,Iris
Cluster 2: 6.9,3.1,5.1,2.3,Iris
Cluster 3: 5.8,4.0,1.2,0.2,Iris

Missing values globally replace

Final cluster centroids:

Attribute              Full D
                         (150
==========================
sepallength              5.8
sepalwidth               3.
petallength              3.7
petalwidth               1.2
class                Iris-set

Time taken to build model (full

=== Model and evaluation on training set ===

Clustered Instances

0       24 ( 16%)
1       26 ( 17%)
2       50 ( 33%)
3       50 ( 33%)
```

Status

OK

Scanned with OKEN Scanner

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Clusterer

Choose HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"

Cluster mode

- Use training set
- Supplied test set
- Percentage split
- Classes to clusters evaluation
- (Nom) JobOffer
- Store clusters for visualization

Ignore attributes

Start | Stop

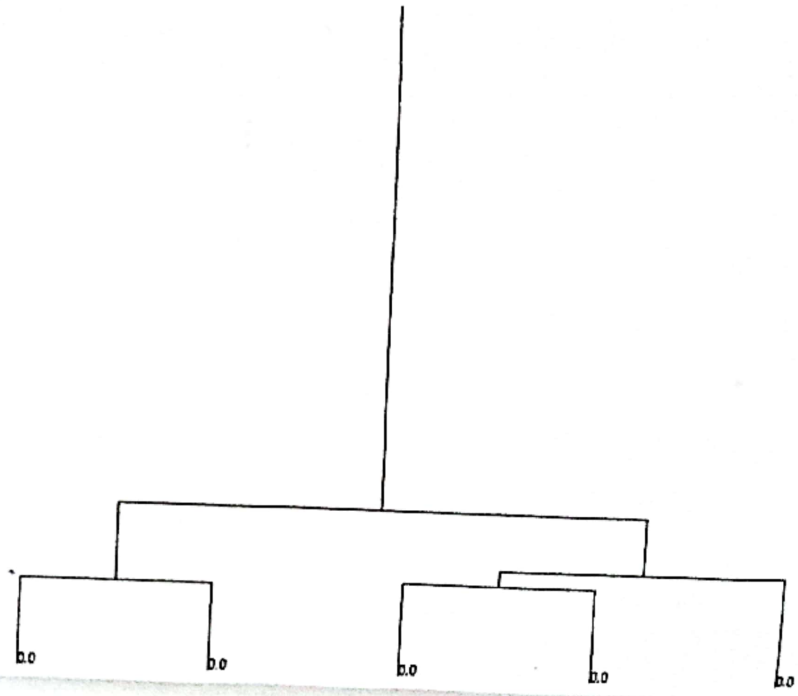
Result list (right-click for options)

09:22:24 - SimpleKMeans
09:23:06 - SimpleKMeans
09:25:49 - HierarchicalClusterer

Clusterer output

```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     student
Instances:    10
Attributes:   4
              CGPA
              Attendence
              StudyHours
              JobOffer
Test mode:    evaluate on tra

=== Clustering model (full tr

Cluster 0
((0.0:0.17521,0.0:0.17521):0.

Cluster 1
(((1.0:0.13514,1.0:0.13514):0

Time taken to build model (fu

=== Model and evaluation on t

Clustered Instances

0       5 ( 50%)
1       5 ( 50%)
```

Weka Cluster Tree Visualizer: 09:25:49 - HierarchicalClusterer (student)



Status
OK

Scanned with OKEN Scanner

- To drop the missing values from the calculation use na.rm = TRUE. which means re the NA values.

Median

The middle most value in a data series is called the median. The median () function is used to calculate this value.

Syntax

The basic syntax for calculating median in R is "

median (x, na.rm = FALSE)

Following is the description of the parameters used "

- x is the input vector.
- **na.rm** is used to remove the missing values from the input vector.

Mode

The mode is the value that has highest number of occurrences in a set of data. Unlike mean a median, mode can have both numeric and character data.

R does not have a standard in-built function to calculate mode. So we create a user function calculate mode of a data set in R. This function takes the vector as input and gives the mo value as output.

SOURCE CODE

Mean:-

```
x <- c(12, 7, 3, 4.2, 18, 2, 54, -21, 8, -5)
result.mean <- mean(x)
print(result.mean)
```

O/P:-

```
> print(result.mean)
[17.8.2]
```


Scanned with OKEN Scanner

mean using trim

```
result.mean <- mean(x, trim=0.3)
print(result.mean)
```

o/p:-

```
> print(result.mean)
[1] 5.55
```

mean with NA

```
x <- c(12,7,3,4.2,18,2,54,-21,8,-5,NA)
result.mean <- mean(x)
print(result.mean)
```

o/p:-

```
> print(result.mean)
[1] NA
```

mean with na.rm = TRUE

```
x <- c(12,7,3,4.2,18,18,2,54,-21,8,-5,NA)
result.mean <- mean(x, na.rm = TRUE)
print(mean)
```

o/p:-

```
> print(result.mean)
[1] 8.22
```


Scanned with OKEN Scanner

```
# median
x <- c (12, 7, 3, 4, 2, 18, 2, 54, -21, 8, -5)
result.median <- median (x)
print (result.median)
```

o/p:-

```
>print (result.median)
 [1] 5.6
```

```
# mode
get mode <- function(v){
    unique v <- unique(v)
    unique v [which.max (tabulate (max (v, unique
}

v <- c(2, 1, 2, 3, 1, 2, 3, 4, 1, 5, 5, 3, 2, 3)
result <- getmode (v)
print(result)
```

o/p:-

```
>print(result)
 [1] 2
```

OUTPUT

```
char v <- c("o", "it", "the", "it", "it")
result <- getmode(char v)
print(result)
```

o/p-

```
>print(result)
[1] "it"
```

Scanned with OKEN Scanner

(b) AIM

Write R program to find Analysis and Covariance with the sample data and visualize the regression graphically.

DESCRIPTION

We use Regression analysis to create models which describe the effect of variation in predictor variables on the response variable. Sometimes, if we have a categorical variable with values like Yes/No or Male/Female etc. The simple regression analysis gives multiple results for each value of the categorical variable. In such scenario, we can study the effect of the categorical variable by using it along with the predictor variable and comparing the regression lines for each level of the categorical variable. Such an analysis is termed as **Analysis of Covariance** also called as **ANCOVA.**

Sample Example

Consider the R built in data set mtcars. In it we observer that the field "am" represents the type of transmission (auto or manual). It is a categorical variable with values 0 and 1. The miles per gallon value(mpg) of a car can also depend on it besides the value of horse power("hp").

We study the effect of the value of "am" on the regression between "mpg" and "hp". It is done by using the **aov()** function followed by the **anova()** function to compare the multiple regressions.

ANCOVA Analysis

We create a regression model taking "hp" as the predictor variable and "mpg" as the response variable taking into account the interaction between "am" and "hp".

Comparing Two Models

Now we can compare the two models to conclude if the interaction of the variables is truly insignificant. For this we use the **anova()** function.

SOURCE CODE

```
input <- mtcars[, c("am", "mpg", "hp")]
print(head(input))
```


Scanned with OKEN Scanner

```
result1 <- aov(mpg~hp*am, data=input)
print(summary(result1))
result2 <- aov(mpg~hp+am, data=input)
print(summary(result2))
print(anova(result1, result2))
boxplot(anova(result1, result2))
```

output:-

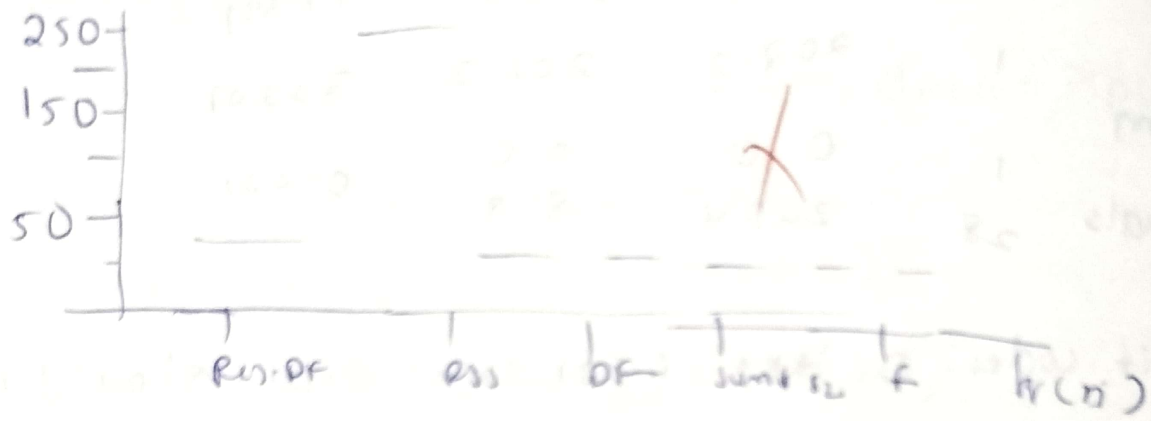
```
> print(head(input))
```

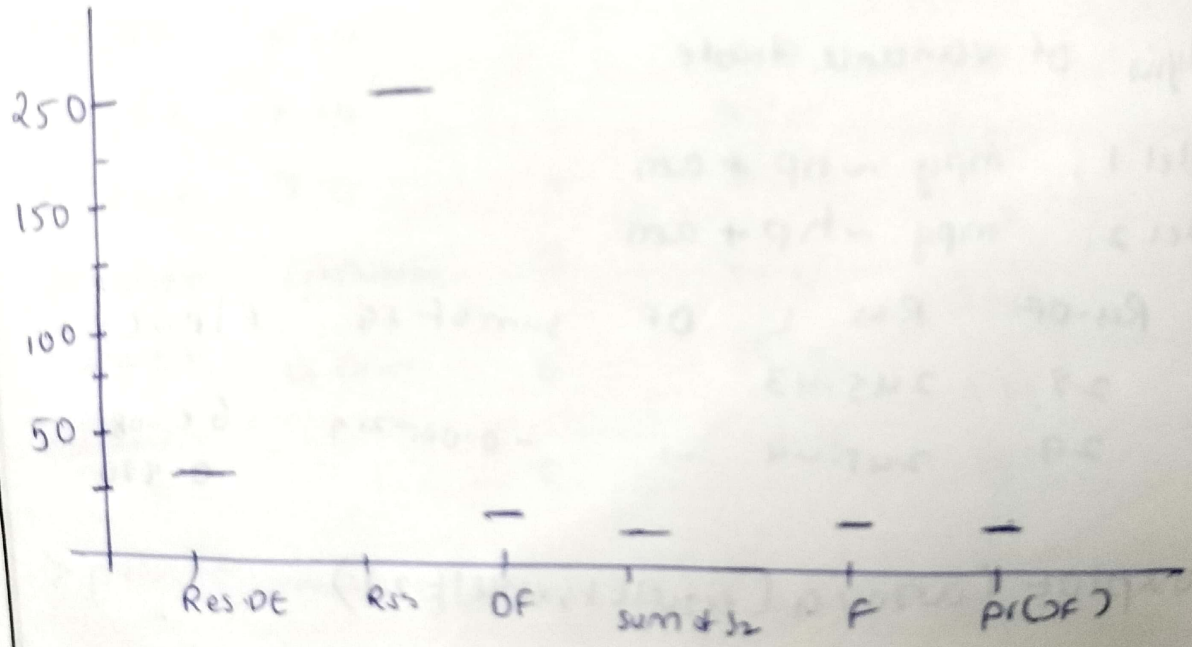
		am	mpg	hp
Mazda	Rx4	1	21.0	110
Mazda	Rx4	1	21.0	110
Datsun	710	1	22.8	93
Hornet	sportabout	0	21.4	110
Hornet	4 Drive	0	18.7	175
Valiant		0	18.1	105

```
> print(summary(result1))
```


Scanned with OKEN Scanner

OUTPUT



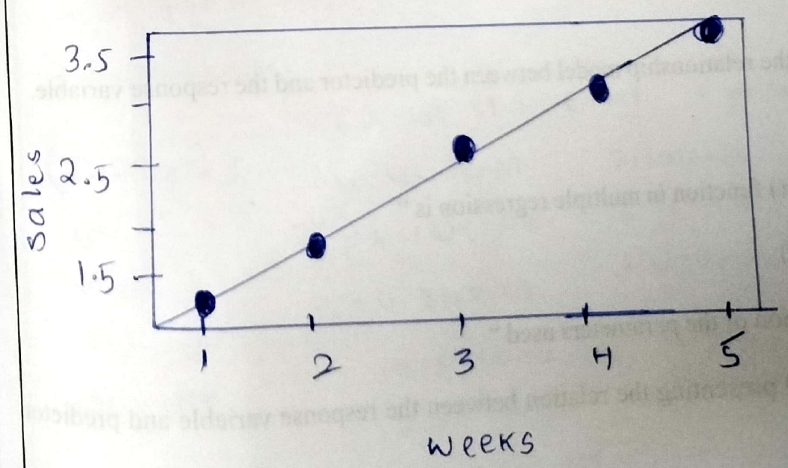


```
x <- c(1,2,3,4,5)
y <- c(1.2, 1.8, 2.6, 3.2, 3.8)
result <- lm(y~x)
print(summary(result))
png(file = "LinearRegression graph.png")
plot(x,y,col = "red", main = "Weekly Sales
                Report", abline(lm(y~x)),
        cex = 1.3, pch = 16, xlab = "weeks", ylab "sales")
dev.off()
```

OUTPUT

```
> print(summary(result))
Call:
lm(formula = y~x)
Residuals:
  6.418e-16  -6.000e-02  8.000e-02  2.00e-01
Coefficients:
            Estimate Std. Error t value Pr(
(Intercept) 0.54000    0.06633   8.141
x           0.66000    0.02000  33.000
---
Residual standard error: 0.0653 on 3 degrees
F-statistic : 1089 on 1 and 3 F ; p-value = 6
> plot( )..
```

weekly sales report



B) MULTIPLE REGRESSION

DESCRIPTION

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

The general mathematical equation for multiple regression is "

y = a + b1x1 + b2x2 +...bnxn

Following is the description of the parameters used "

- **y** is the response variable.

a, b1, b2...bn are the coefficients.

x1, x2, ...xn are the predictor variables.

e create the regression model using the **lm()** function in R. The model determines the value of e coefficients using the input data. Next, we can predict the value of the response variable for given set of predictor variables using these coefficients.

() Function

is function creates the relationship model between the predictor and the response variable.

ntax

e basic syntax for **lm()** function in multiple regression is "

y~x1+x2+x3...,data)

lowing is the description of the parameters used "

formula is a symbol presenting the relation between the response variable and predictor variables.

data is the vector on which the formula will be applied.

SOURCE CODE

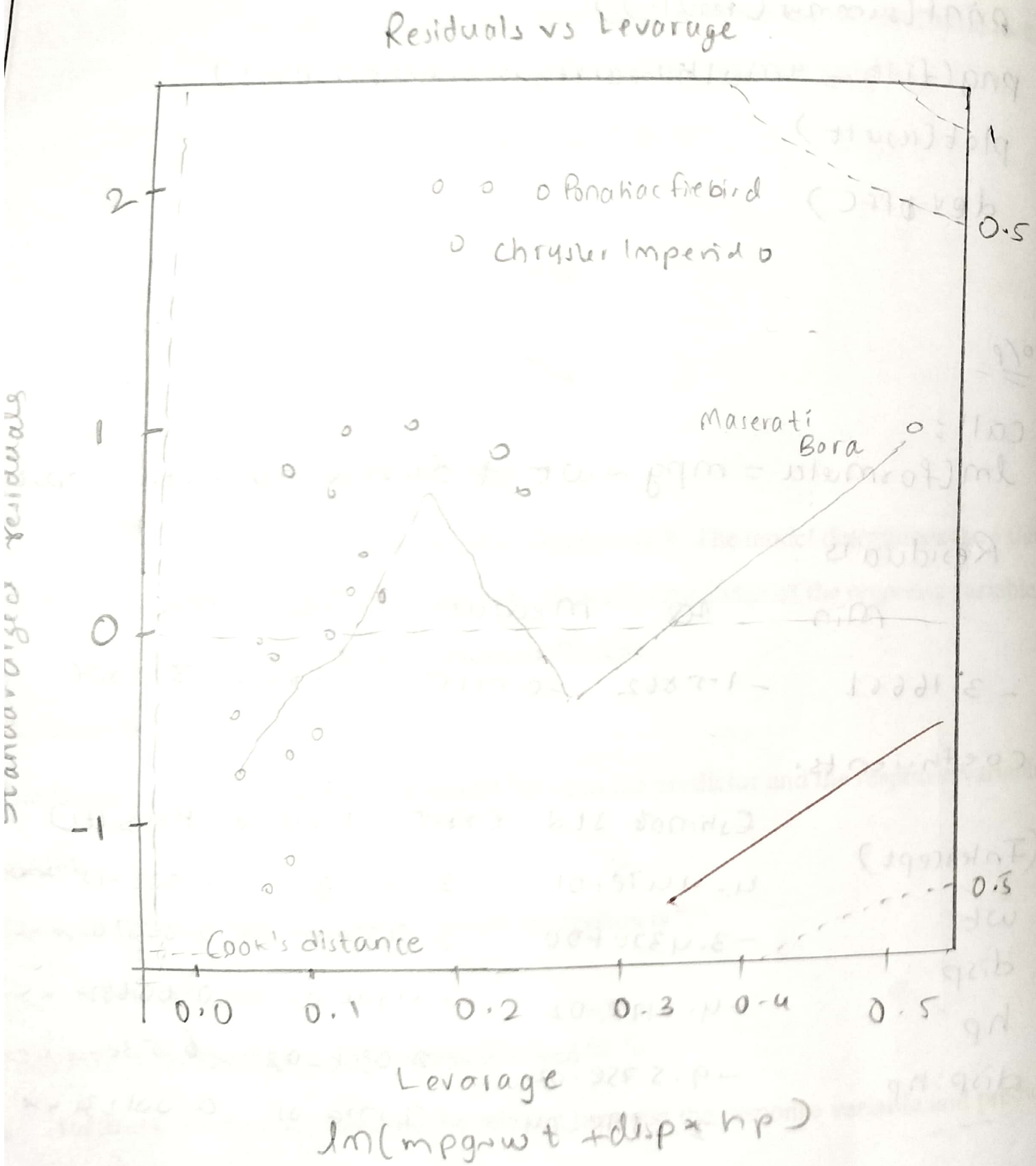
```
input <-mtcars[,c("mpg", "wt", "disp", "hp")]
result <-lm(mpg~wt+disp*hp, data=input
print(summary(result))
png(file = "Multilinear regression graph.png")
plot(result)
dev.off()
```

o/p:

```
Call:
lm(formula = mpg ~ wt + disp * hp, data = i

Residuals:
     Min       1Q   Median       3Q      Max
 -3.1666   -1.5862   -0.5715    1.6613   3.78

Coefficients:
            Estimate Std. Error t value Pr(>
(Intercept) 4.447e+01  2.699e+00  1.30e
wt          -3.433e+00 8.977e-01  0.00
disp        -4.219e-02 1.436e-02  0.0
hp          -9.575e-02 2.030e-02  6.
disp:hp     2.589e-04  7.177e-05  0.

F-statistic: 51.02 on 4 and 27 DF, p-valu
```

Scanned with OKEN Scanner

OUTPUT

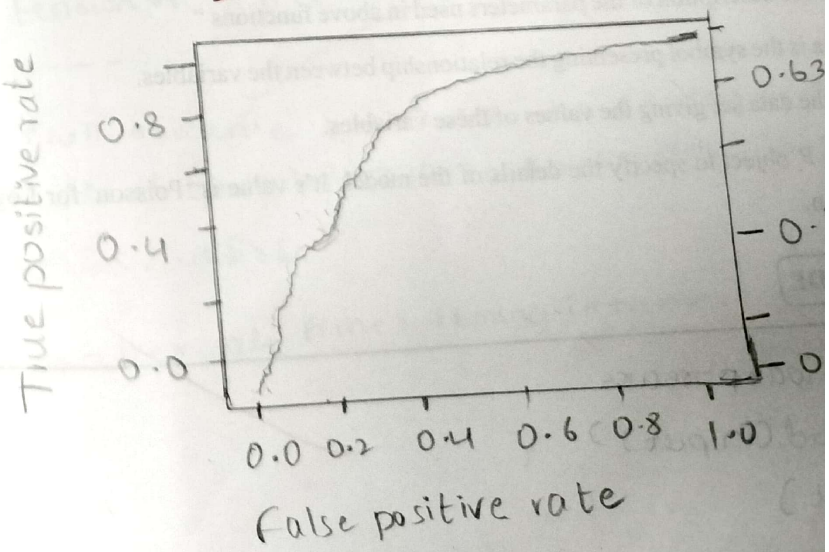

Residuals vs Levarage
Pontiac firebird
Chrysler Imperial
Maserati Bora
Cook's distance
1
0.5
0.5
2
1
0
-1
0.0
0.1
0.2
0.3
0.4
0.5
Levarage
lm(mpg~wt + disp * hp)
Standardized residuals

SOURCE CODE

```
claimants <- read.csv("C:/Users/students/claimants.csv")
sum(is.na(claimants))
claimants<-na.omit(claimants)
logit<-glm(ATTORNEY ~ factor(CLMSEX)
      + factor(CLMINSUR) + factor(SEATBELT)
        +CLMAGE + LOSS, family="binomial",
            data=claimants)
summary(logit)
prob=predict(logit, type =c("response"),claimants)
prob
confusion<-table(prob>0.5, claimants$ATTORNEY)
confusion
Accuracy<-sum(diag(confusion))/sum(confusion)
Accuracy
install.packages("ROCR")
install.packages("PROC")
library(ROCR)
library(pROC)
rocrpred <-prediction(prob, claimants$ATTORNEY)
rocrperf <-performance(rocrpred, 'tpr','fpr')
plot(rocrperf,colorize=T, text.adj=c(-0.2,1.7))
auc<-auc(claimants$ATTORNEY ~ prob)
auc
```


OUTPUT

```
sum > 290
summ(logit)
Call:
glm(formula = " ")
Coeff :-
Number of Fisher Scoring iterations : 6

>prob
        1          2          3
2.970231e-01 5.018179e-01 5.762151e-01 ...
:

>confusion
        0    1
FALSE 390  125
TRUE  198  393

>Accuracy 0.705292
```



D) POISSON REGRESSION

DESCRIPTION

Poisson Regression

Poisson Regression involves regression models in which the response variable is in the form of counts and not fractional numbers. For example, the count of number of births or number of wins in a football match series. Also the values of the response variables follow a Poisson distribution.

The general mathematical equation for Poisson regression is "

log(y) = a + b1x1 + b2x2 + bnxn.....

Following is the description of the parameters used "

- **y** is the response variable.
- **a** and **b** are the numeric coefficients.
- **x** is the predictor variable.

The function used to create the Poisson regression model is the **glm()** function.

Syntax

The basic syntax for **glm()** function in Poisson regression is "glm(formula,data,family)

Following is the description of the parameters used in above functions "

- **formula** is the symbol presenting the relationship between the variables.
- **data** is the data set giving the values of these variables.
- **family** is R object to specify the details of the model. It's value is 'Poisson' for Logistic Regression.

SOURCE CODE

```
input <- warpbreaks
print(head(input))
plot(input)
```



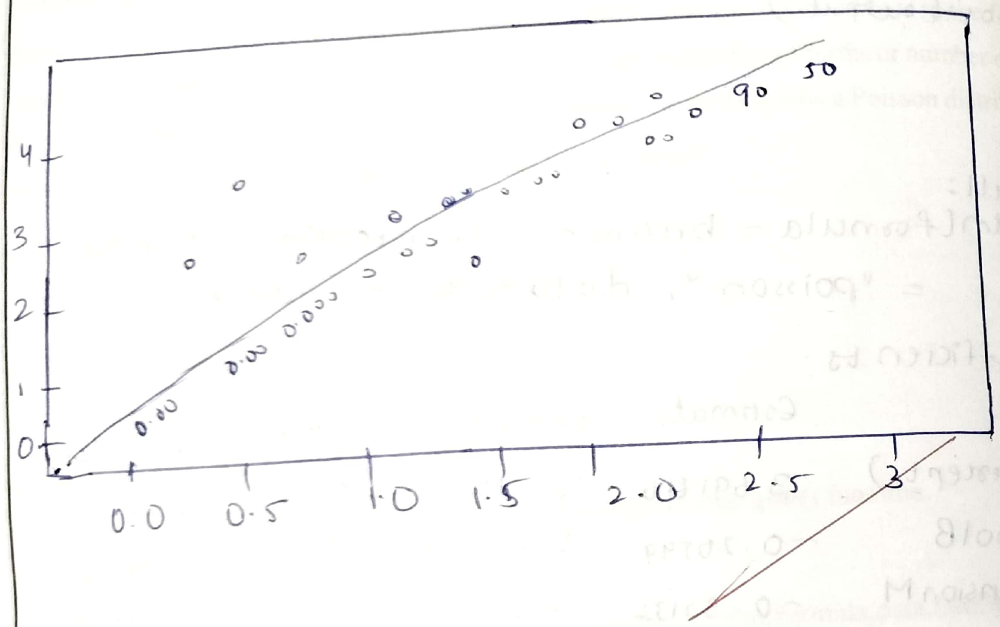
```
output <-glm (formula = breaks ~wool + tension,
        data = warpbreaks, family = "poisson
print(summary(output))
plot(output)
```

O/P:

```
Call:
glm(formula = breaks ~wool + tension, family
    = "poisson", data= warpbreaks)

Coefficients:
            Estimate   Std. Error   z value   Pr(>
(Intercept)  3.69196    0.4541      31.032    <2e
wool B      -0.20599    0.05157     -3.994    6.9
tension M   -0.32132    0.06027     -5.332    9.7
tension H   -0.51849    0.06396     -8.107    5.
----

Null deviance : 297.37  on 53 degrees of
Residual deviance : 210.30 on 50 degree of

AIC: 493.06

Number of Fisher Scoring iterations : 4
```

Scanned with OKEN Scanner

OUTPUT



VIVA QUESTIONS

1. Define regression and its types?

Ans. Regression is a statistical technique used to the relationship b/w dependent variable & indepen

2. What is regression in statistics with an example?

Ans. It is used in statistics to estimate the relationships among variables.

3. What is meant by linear regression?

Ans. Linear regression is the simplest form regression where $y = \beta_0 + \beta_1 x + \epsilon$

4. Explain multiple linear regression with an example?

Ans. MLR is a extension of L.R where 2 o regression used to predict the depend

5. What is the difference between linear regression and multiple regression?

Ans. Number of independent vars, equation use case.

6. What is difference between linear and logistic regression?

Ans. Output type, Output range, equation, function used.

Freddy

Week-10

Date: _____

(a) AIM

Write R program to find Time Series Analysis with the sample data and visualize the regression graphically.

DESCRIPTION

Time Series Analysis

Time series is a series of data points in which each data point is associated with a timestamp. A simple example is the price of a stock in the stock market at different points of time on a given day. Another example is the amount of rainfall in a region at different months of the year. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called **time-series object**. It is also an R data object like a vector or data frame.

The time series object is created by using the **ts()** function.

Syntax

The basic syntax for **ts()** function in time series analysis is "

timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used "

- **data** is a vector or matrix containing the values used in the time series.
- **start** specifies the start time for the first observation in time series.

- **end** specifies the end time for the last observation in time series.
- **frequency** specifies the number of observations per unit time.

Except the parameter "data" all other parameters are optional.

SOURCE CODE

```
snowfall <- c(630.4, 911.5, 683.5, 790, 1170.8,
        860.1, 1330.6, 996.6, 783.2, 982,
        831.8, 1021)
snowfall_timeseries <- ts(snowfall, start
        = c(2023,1), frequency=8)
print(snowfall_timeseries)
png(file = "snowfall.png")
plot(snowfall_timeseries)
dev.off()
```

Scanned with OKEN Scanner

OUTPUT

Time Series:

Start = c(2023,1)

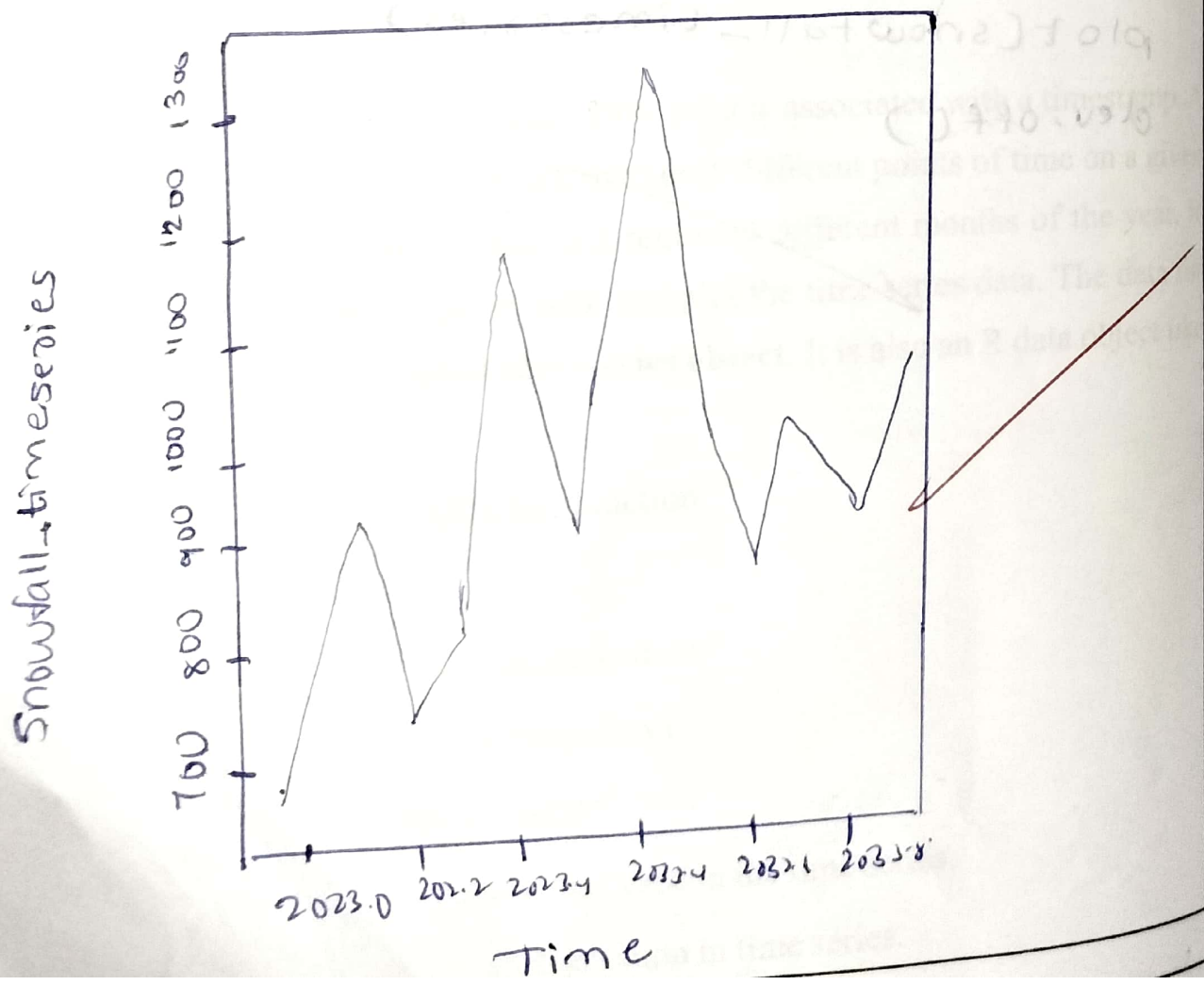
End = c(2024,4)

Frequency = 8

> dev.off()

RstudioGD

2



```
png(file = "nls.png")
plot(xvalues, yvalues)
model <- nls(yvalues ~ b1 * xvalues ^2 + b2,
        start = list(b1=1, b2=3))
model
new.data <- data.frame(xvalues
        = seq(min(xvalues, max(xvalues), len=100))
lines(new.data$xvalues, predict(model,
        newdata = new.data))
dev.off()
print(sum(resid(model)^2))
print(confint(model))
```

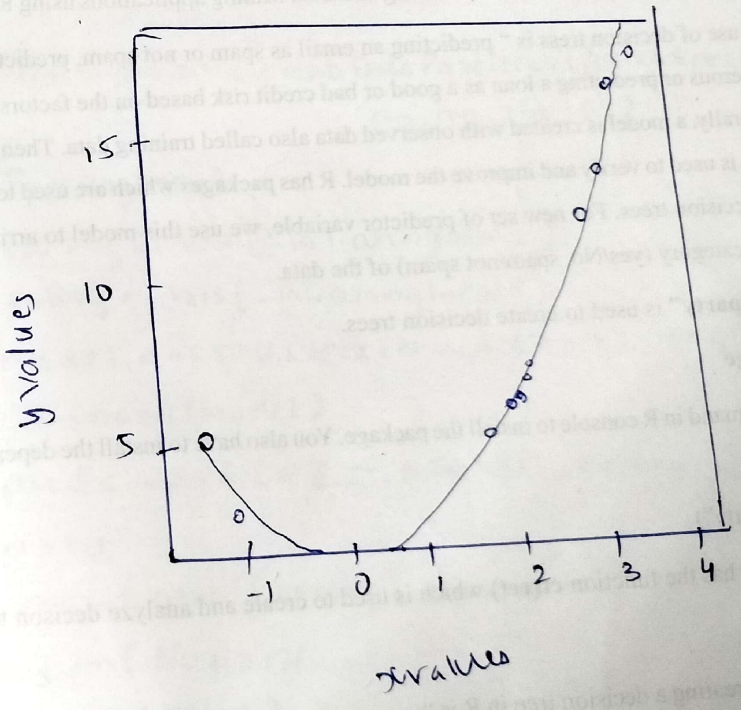
O/P:-

```
>model
Nonlinear regression model
  model: yvalues ~ b1 * xvalues ^2 + b2
  data: parent.frame()
   b1    b2
1.195  1.997
 residual sum of squares: 1.082
Number of iterations to convergence: 1
```

png

OUTPUT

	2.5%	97.5%
b1	1.137708	1.253135
b2	1.1497364	2.498484



(c) AIM

Write R program to find Decision Tree with the sample data and visualize the regression graphically.

DESCRIPTION

Decision tree

Decision tree is a graph to represent choices and their results in form of a tree. The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions. It is mostly used in Machine Learning and Data Mining applications using R.

Examples of use of decision tress is " predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these. Generally, a model is created with observed data also called training data. Then a set of validation data is used to verify and improve the model. R has packages which are used to create and visualize decision trees. For new set of predictor variable, we use this model to arrive at a decision on the category (yes/No, spam/not spam) of the data.

The R package **"party"** is used to create decision trees.

Install R Package

Use the below command in R console to install the package. You also have to install the dependent packages if any.

install.packages("party")

The package "party" has the function **ctree()** which is used to create and analyze decison tree.

Syntax

The basic syntax for creating a decision tree in R is "

ctree(formula, data)

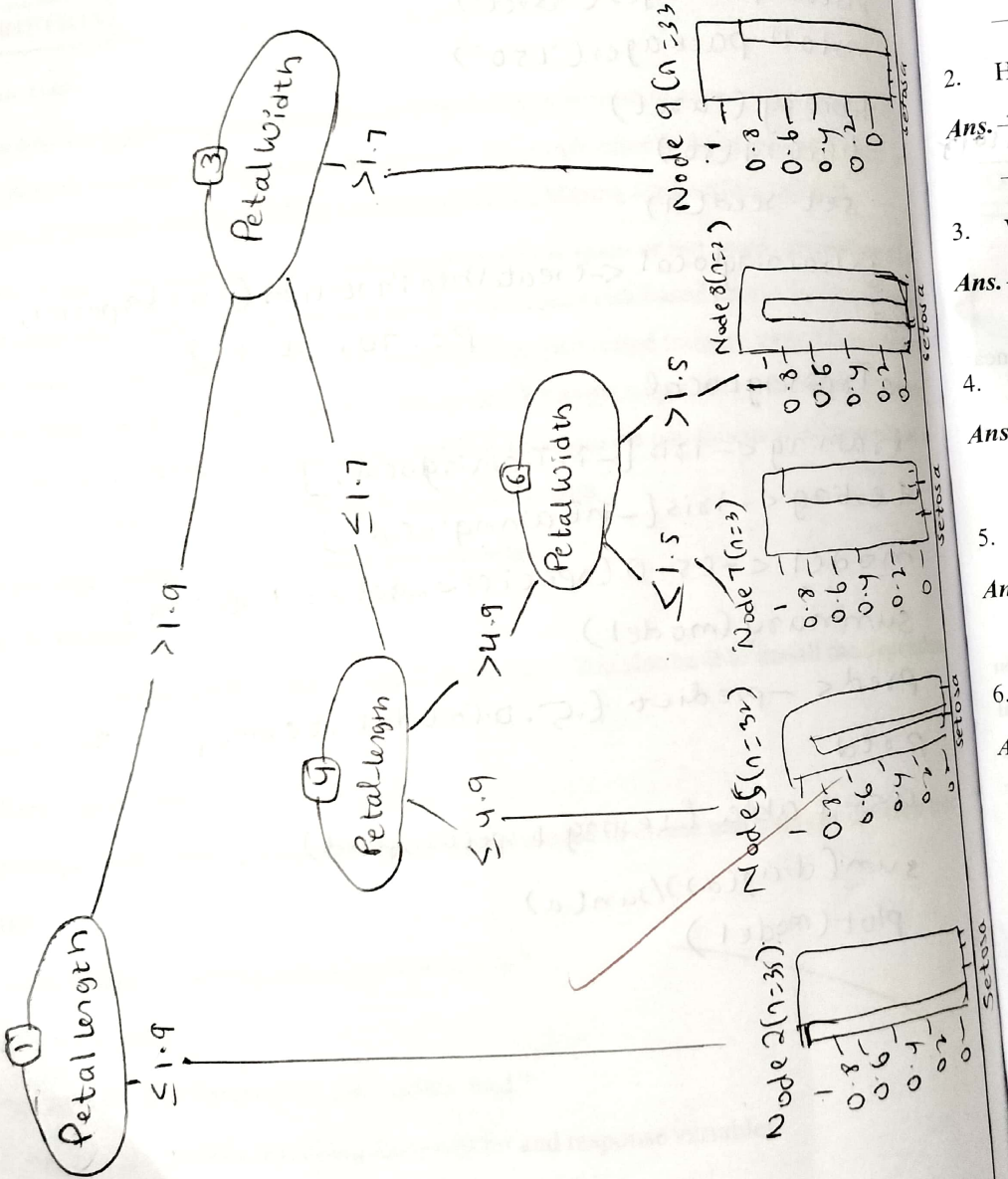
Following is the description of the parameters used "

- **formula** is a formula describing the predictor and response variables.
- **data** is the name of the data set used.

SOURCE CODE

```
data("iris")
install.packages("caret")
install.packages("C50")
library(caret)
library(C50)
set.seed(1)
inTraininglocal <-createDataPartition(iris$species,
                                      p=.70, list=F)
inTraininglocal
training<-iris[-inTraininglocal,]
testing<-iris[-inTraininglocal,]
model <-C5.0(species~.,data = training)
summary(model)
pred<-predict(C5.0(model,testing[,-5]))
pred
a<-table(testing$species,pred)
sum(diag(a))/sum(a)
plot(model)
```

OUTPUT



VIVA QUESTIONS

1. What is time series analysis with an example?

Ans. It is the study of data points collected or recorde specific time intervals to identify terms, patter

2. How do you analyze time series data in R?

Ans. Analyzing time series data: Use ts() func to cre time series object and plot() to visualize it.

3. What is the purpose of time series analysis?

Ans. To identify patterns, trend, seasonal variati data over time for forcasting.

4. What is difference between linear and nonlinear?

Ans. Linear: Relationship b/w variables is straig Non-linear: Relation b/w variables not a stra

5. What is decision tree with an example?

Ans. It is a flowchart like model used for making or predictions by splitting data based on Pe

6. How do you create a decision tree in R?

Ans. Use rpart() func from rpart package.

Scanned with OKEN Scanner

Week-11

Date: 24/10/23

AIM

Write R program to find the following Distribution with the sample data and visualize the linear regression graphically.

A) NORMAL DISTRIBUTION

DESCRIPTION

Normal Distribution

In a random collection of data from independent sources, it is generally observed that the distribution of data is normal. Which means, on plotting a graph with the value of the variable in the horizontal axis and the count of the values in the vertical axis we get a bell shape curve. The center of the curve represents the mean of the data set. In the graph, fifty percent of values lie to the left of the mean and the other fifty percent lie to the right of the graph. This is referred as normal distribution in statistics.

R has four in built functions to generate normal distribution. They are described below.

- dnorm(x, mean, sd)
- pnorm(x, mean, sd)
- qnorm(p, mean, sd)
- rnorm(n, mean, sd)

Following is the description of the parameters used in above functions "

- x is a vector of numbers.
- p is a vector of probabilities.
- n is number of observations(sample size).
- mean is the mean value of the sample data. It's default value is zero.
- sd is the standard deviation. It's default value is 1.

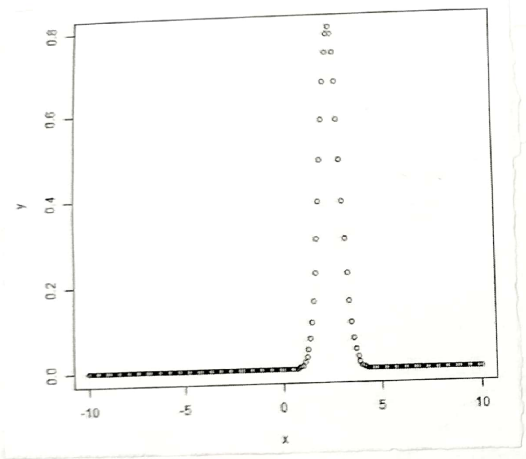
SOURCE CODE

```
x<-seq(-10,10,by=.1)
y<- dnorm(x,mean=2.5,sd=0.5)
png(file="dnorm.png")
plot(x,y)                          -> dnorm
dev.off()

x<-seq(-10, 10,by=.1)
y<- pnorm(x,mean=2.5,sd=2)
png(file="pnorm.png")              -> pnorm
plot(x,y)
dev.off()

x<-seq(0,1,by=0.02)
y<-qnorm(x,mean=2,sd=1)
png(file="qnorm.png")              -> qnorm
plot(x,y)
dev.off()
```

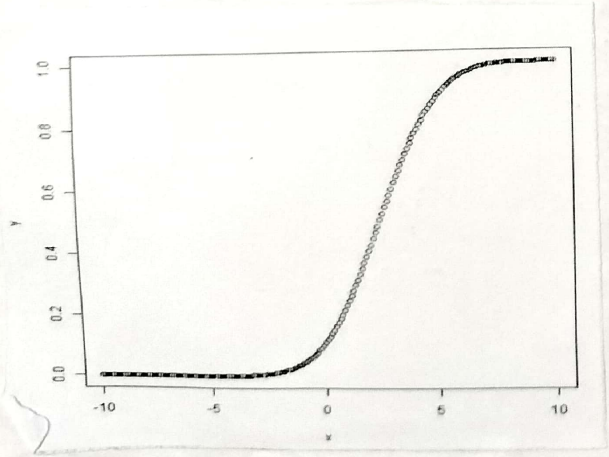
Scanned with OKEN Scanner

```
y<-rnorm(50)
png(file="rnorm.png")
hist(y,main="Normal Distribution")
dev.off()
```

→ rnorm

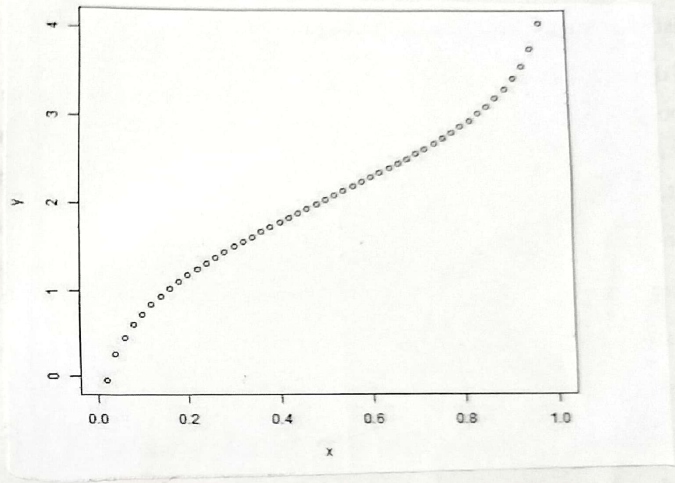


dnorm

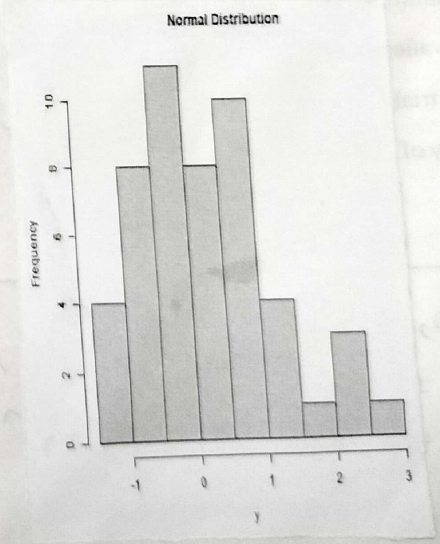


pnorm

OUTPUT



qnorm



rnorm

Scanned with OKEN Scanner

B) BINOMIAL DISTRIBUTION

DESCRIPTION

Binomial Distribution

The binomial distribution model deals with finding the probability of success of an event which has only two possible outcomes in a series of experiments. For example, tossing of a coin always gives a head or a tail. The probability of finding exactly 3 heads in tossing a coin repeatedly for 10 times is estimated during the binomial distribution.

R has four in-built functions to generate binomial distribution. They are described below.

- dbinom(x, size, prob)
- pbinom(x, size, prob)
- qbinom(p, size, prob)
- rbinom(n, size, prob)

Following is the description of the parameters used "

- **x** is a vector of numbers.
- **p** is a vector of probabilities.
- **n** is number of observations.
- **size** is the number of trials.
- **prob** is the probability of success of each trial.

26,27,28 sec. 1/se

SOURCE CODE

```
x <- seq(0,50, by = 1)
y <- dbinom(x,50,0.5)
png(file = "dbinom.png")
plot(x,y)
dev.off()
```

→ dbinom


```
x <- pbinom(26,51,0.35)
plot(x)
print(x)
```

→ pbinom

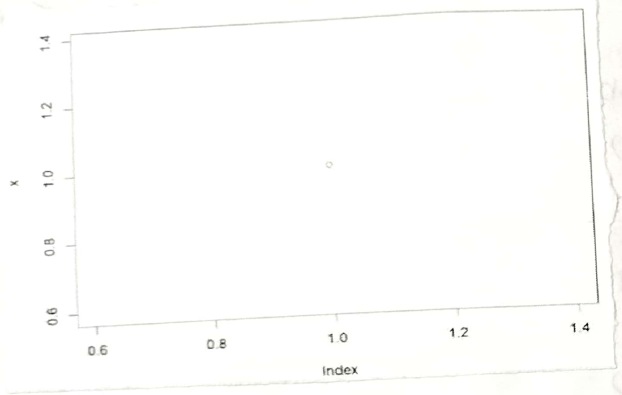
```
x <- dbinom(8,150,.4)
plot(x)
print(x)
```

→ dbinom

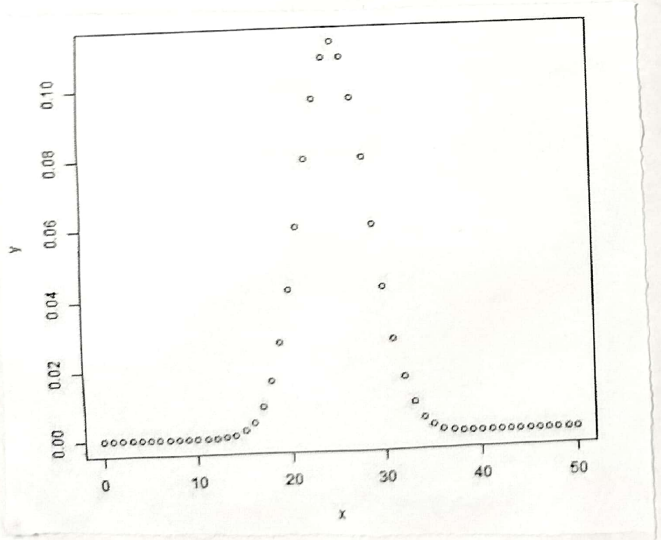
```
x <- qbinom(0.25,30,1/2)
plot(x)
print(x)
```

→ qbinom

Scanned with OKEN Scanner

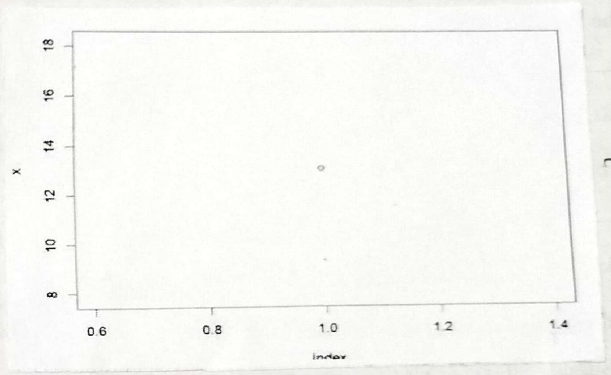


d/binom

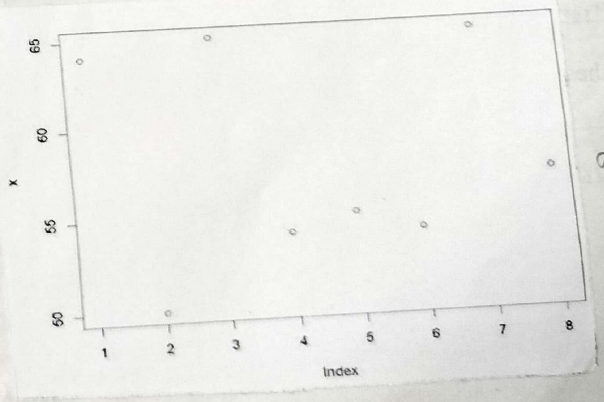


p/binom

OUTPUT



r/binom



q/binom

Scanned with OKEN Scanner

SOURCE CODE

```
library("MASS")
print(str(Cars93))

#loading the mass library
#creating a data frame from the main
  dataset.
car_data <- data.frame(Cars93$AirBags,
                       Cars93$Type)


car_data= table(Cars93$AirBags, Cars93$Type)
print(car_data)
SmoothScatter(car_data)

print(chisq.test(car_data))
```


Pearson's Chi-squared test

data: car_data

X-squared = 33.001, df = 10, p-value = 0.0002723

```
>print(car_data)
```

	Compact	Large	Midsize	Small	Sporty	Van
Driver & Passenger	2	4	7	0	3	0
Driver only	9	7	11	5	8	3
None	5	0	14	16	3	6

Scanned with OKEN Scanner

b)t-test

DESCRIPTION

The T-test is used to measure the means of two groups to check they are equal to each other. Hypothesis testing is significantly used by data researchers to figure out their ideas. In some situation measuring or comparing a single item is not applicable practically, to solve this issue statistical methods were developed which tests hypothesis in the situation for comparison. One such most known hypothesis is T-test also referred to as student T-Test. T-test prefers data with a normal distribution in which data are distributed evenly.

In statistics, the T-test is one of the most common test which is used to determine whether the mean of the two groups is equal to each other. The assumption for the test is that both groups are sampled from a normal distribution with equal fluctuation. The null hypothesis is that the two means are the same, and the alternative is that they are not identical. It is known that under the null hypothesis, we can compute a t-statistic that will follow a t-distribution with n1 + n2 - 2 degrees of freedom.

SOURCE CODE

```
x<-c(12,15,14,10,13,15)
y<-c(14,18,13,16,17,19)
t_test_result<-t.test(x,y)
print(t_test_result)
boxplot(x,y,names=c("Group X","Group Y"),
        main="T-Test Comparison",
        col=c("skyblue","lightgreen"))
```

welch Two Sample t-test

data: x and y

t = -2.4315 , df = 9.7022 , p-value = 0.0365

alternative hypothesis: true difference in means is not equal to 0.

95 percent confidence interval:

-5.76052 -0.23948

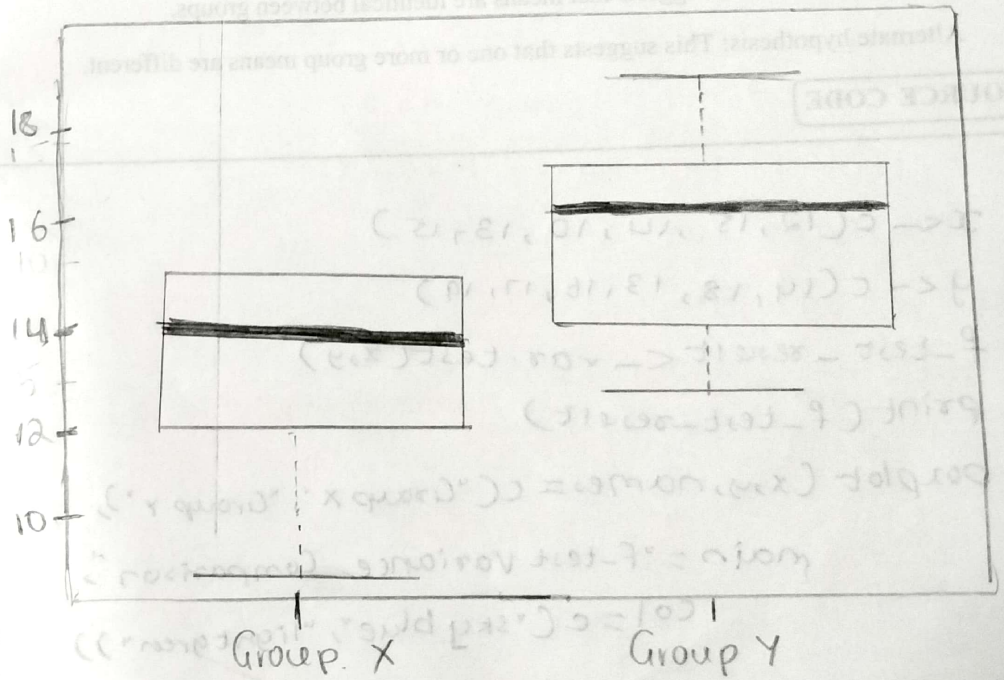
sample estimates:

mean of x mean of y

13.16667 16.16667.

OUTPUT

T- test Comparison



Scanned with OKEN Scanner

c) **F-test**

DESCRIPTION

ANOVA determines whether groups created by levels of an independent variable are statistically different by comparing means of the treatment levels with those of the overall dependent variable. This means that if the mean of any group vastly differs from the overall mean, the null hypothesis is immediately rejected.

An ANOVA test requires us to set up two different hypotheses. These are

- Null hypothesis: This suggests that means are identical between groups.
- Alternate hypothesis: This suggests that one or more group means are different.

SOURCE CODE

```
x<-c(12,15,14,10,13,15)
y<-c(14,18,13,16,17,19)
f_test_result<-var.test(x,y)
print(f_test_result)
boxplot(x,y,names=c("Group x","Group y"),
        main="f-test variance Comparison",
        col=c("sky blue","lightgreen"))
```


Scanned with OKEN Scanner

F-test to compare two variances

data: x and y

F = 0.70186, num df = 5, denom df = 5,
p-value = 0.7071

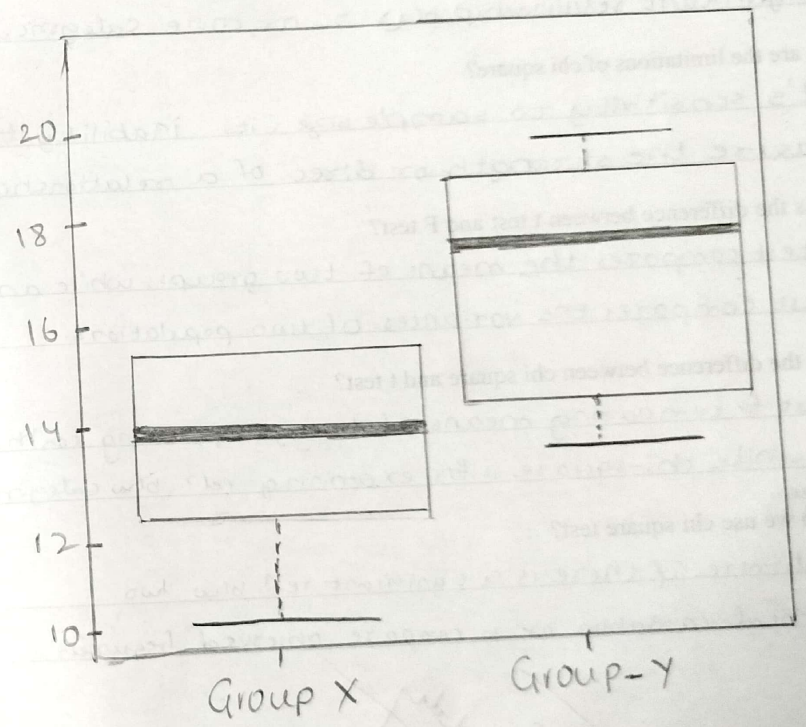
alternative hypothesis:
true ratio of variances is not equal to 1

95 percent confidence interval:
0.09821241 5.01518352

sample estimates:
ratio of variances
0.7018634

OUTPUT

F-test variance comparison



Scanned with OKEN Scanner