A Course File

On

DATA MINING AND DATA ANALYTICS LAB

(III- B. Tech. – I– Semester)

Submitted to

DEPARTMENT OF COMPUTER SCIENCE& ENGINEERING

By

Dr. Y.Sucharitha
(Assoc. Professor, Dept. of CSE)

CMR GROUP OF INSTITUTIONS
EXPLORE TO INVENT

CMR INSTITUTE OF TECHNOLOGY

Kandlakoya(V), Medchal Road, Hyderabad – 501 401
Ph. No. 08418-222042, 22106 Fax No. 08418-222106

(2022-23)

CONTENTS

1. SYLLABUS

DATA MINING & ANALYTICS LAB

B.Tech. III Year I Sem. **Course Code: CS-PCC-316**

Course Outcomes: Upon completion of the course, the student will be able to

1. make use of open source data mining and analytic tools
2. examine the interesting insights of Apriori algorithm using WEKA
3. demonstrate the classification and clustering techniques
4. analyze the concepts of data analytics and statistical testing methods
5. compare various kinds of regression techniques

LIST OF EXPERIMENTS

Part-A: Data Mining

1. Demonstration of preprocessing on dataset student.arff
2. Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm.
3. Demonstration of classification rule process on dataset employee.arff using j48 algorithm.
4. Demonstration of classification rule process on dataset employee.arff using id3 algorithm.
5. Demonstration of classification rule process on dataset employee.arff using naïve bayes algorithm.
6. Demonstration of clustering rule process on dataset iris.arff using simple k-means.
7. Demonstration of clustering rule process on dataset student.arff using hierarchical clustering.

Part-B: Data Analytics

1. 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.

2. 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.

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

b) Write R program to find Non Linear Least Square with the sample data and visualize the regression graphically.

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

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

a) Normal Distribution

b) Binomial Distribution

5. Write R program to do the following tests with the sample data and visualize the results graphically.

a) $\chi 2$-test

b) t-test

c) F-test

Micro-Projects: Student must submit a report on one of the following Micro–Projects before commencement of second internal examination.

1. Data Mining Techniques in Healthcare System using WEKA.
2. Credit Scoring Analysis using WEKA.
3. Crime Rate Prediction using K Means.
4. Weather Forecasting using Data Mining.
5. Smart Health Prediction using Data Mining.
6. Movie Success Prediction using Data Mining.
7. Google data analysis using R.
8. IRCTC Reservation system data analysis using R.
9. Facebook data analysis using R.
10. Banking system data analysis using R.

Reference:

1. Data Mining and Analytics Lab Manual, Department of CSE, CMRIT, Hyd.

2. STUDENT ENTRY BEHAVIOR OR PRE-REQUISITES

- Database Management Systems
- Probability & Statistics

3. COURSE OUTCOMES

Course Outcome	Course Outcome Statements
CO - 1	make use of open source data mining and analytic tools
CO – 2	examine the interesting insights of Apriori algorithm using WEKA
CO – 3	demonstrate the classification and clustering techniques
CO – 4	analyze the concepts of data analytics and statistical testing methods
CO – 5	compare various kinds of regression techniques

Co-Po Mapping

POs	PO4	PO5	PO6
CO1	3	3	3
CO2	3	3	3
CO3	3	3	3
CO4	3	3	3
CO5	3	3	3

4. MAPPING OF COURSE WITH PEOS-PSOS-POS

Program Educational Objectives (PEOs)

Sl. No.	PEOs Name	Program Education Objective Statements
1	PEO - 1	Impart profound knowledge in humanities and basic sciences along with core engineering concepts for practical understanding & project development.[PO's: 1,2,3,4,5,7,8,9,10,11 and 12] [PSO's: 1 and 2]
2	PEO – 2	Enrich analytical skills and Industry-based modern technical skills in core and interdisciplinary areas for accomplishing research, higher education, entrepreneurship and to succeed in various engineering positions globally. [PO's: 1,2,3,4,5,6,7,8,9,10 and 12] [PSO's: 1, 2 and 3]
3	PEO – 3	Infuse life-long learning, professional ethics, responsibilities and adaptation to innovation along with effective communication skills with a sense of social awareness. [PO's: 1,2,3,4,5,6,7,8,9,10,11 and 12] [PSO's: 2 and 3]

Program Specific Objectives (PSOs)

Sl. No.	PSOs Name	Program Specific Objective Statements
1	PSO - 1	Use mathematical abstractions and Algorithmic design along with open source programming tools to solve complexities involved in efficient programming.[PO:1,2,3,4 and 5] & [PEO:1 and 2]
2	PSO – 2	Ensure programming & documentation skills for each individual student in relevant subjects i.e., C, C^{++}, Java, DBMS, Web Technologies (Development), Linux, Data Warehousing & Data Mining and on Testing Tools.[PO:1,2,3,4,5,10 and 11] & [PEO:1,2 and 3]
3	PSO – 3	Ensure employability and career development skills through Industry oriented mini & major projects, internship, industry visits, seminars and workshops. [PO:6,7,8,9,10,11 and 12] & [PEO:1,2 and 3]

Program Outcomes (POs)

PO Name	Graduate Attributes	PO Statements
PO1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 2	Problem analysis	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 3	Design/ development of solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 4	Conduct investigations of complex problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 5	Modern tool usage	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]
PO 6	The engineer and society	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. [PEO's: 2 and 3]
PO 7	Environment and sustainability	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. [PEO's: 1,2 and 3]
PO 8	Ethics	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. [PEO's: 1,2 and 3] [PSO's: 2 and 3]
PO 9	Individual and team work	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. [PEO's: 1,2 and 3] [PSO's: 3]
PO 10	Communication	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. [PEO's: 1,2 and 3] [PSO's: 2 and 3]
PO 11	Project management and finance	Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. [PEO's: 1 and 3] [PSO's: 2 and 3]
PO 12	Life-long learning	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. [PEO's: 1,2 and 3] [PSO's: 1,2 and 3]

5. Mapping Of Course Outcomes with PEOs

(CORRELATION - TABLES)

No	Course Outcomes	PEO1	PEO2	PEO3
1	CO - 1	3	3	3
2	CO – 2	3	3	3
3	CO – 3	3	3	3
4	CO – 4	3	3	3
5	CO – 5	3	3	3

6. MAPPING OF COURSE OUTCOMES WITH PSOS
(CORRELATION - TABLES)

No	Course Outcomes	PSO1	PSO2	PSO3	AVERAGE
1	CO - 1	3	3	2	2.67
2	CO – 2	3	3	2	2.67
3	CO – 3	3	3	3	3.00
4	CO – 4	3	2	2	2.33
5	CO – 5	3	3	3	3.00
6	CO - 6	3	3	3	3.00
	Average	3.00	2.83	2.50	2.78

7. DIRECT COURSE ASSESSMENT

(As mentioned in following table of 10 parameters, of which consider only the parameters required for this courses)

No	Description	Targeted Performance	Actual Performance	Remarks	Course Attainment
1	Internal Marks(25)	90% of Students(153 Students) should Secure 60% of Internal Marks i.e., 15 Marks			
2	External Marks(50)	80% of Students(136 Students) should Secure 70% of External Marks i.e., 35 Marks			
3	Clearing of Subject	A minimum of 95% of Students(161 Students) should clear this course in first attempt			
4	Getting First Class	90% of Students(153 Students) should Secure I Class Marks i.e., 45 Marks in my course			
5	Distinction	80% of Students (136 Students) should secure First Class With Distinction i.e., 53 Marks in my course			
6	Outstanding Performance	50% of Students (85 Students) should secure 80% and above Marks. i.e., 60 Marks in my course			

8. INDIRECT COURSE ASSESSMENT

(As mentioned-strong (3), moderate (2), weak (1) & no comment (0))

Mission Statement of CSE

- **Impart fundamentals through state of art technologies for research and career in Computer Science & Engineering.**
- **Create value-based, socially committed professionals for anticipating and satisfying fast changing societal requirements.**
- **Foster continuous self learning abilities through regular interaction with various stake holders for holistic development.**

Correlation of Mission Elements with Mission Statement of CSE Department related to the Course (only Ticking given by faculty)

No	Mission Elements	Strong	Moderate	Weak	No Comment
M-1	Impart Fundamentals	√			
M-2	State Of Art Technologies	√			
M-3	Research & Career Development	√			
M-4	Value based Socially Committed Professional	√			
M-5	Anticipating & Satisfying Industry Trends		√		
M-6	Changing Societal Requirements		√		
M-7	Foster Continuous Learning	√			
M-8	Self Learning Abilities	√			
M-9	Interaction with stakeholders	√			
M-10	Holistic Development	√			

Indirect Course Assessment through Student Satisfaction Survey

(Note for *: Parameters used for course teaching like
a: Classroom teaching b: Simulations c:labs d: Mini_Projects
e: Major Projects f: Conferences g: professional activities
h: Technical Clubs i: Guest Lectures j: Workshops k: Technical Fests l:Tutorials
m:NPTLs n:Digital Library o: Industrial Visits
p: software Tools q: Internship/training r:Technical Seminars
s: NSS t: NSS u: sports etc.
Further assume other parameters if any)

No	Question Based on PEO/ PO/PSO/CO	Parameters (a /b /c…/)*	Strong (3)	Moderate (2)	Weak (1)	No comment (0)
1	Did the course impart fundamentals through interactive learning and contribute to core competence?	a,b,c,d,e,h,i,l,m ,p,q				
2	Did the course provide the required knowledge to foster continuous learning?	a,b,c,d,e,h,i,l,m ,p,q				
3	Whether the syllabus content anticipates & satisfies the industry and societal needs?	a,b,c,d,e,h,i,l,m ,p,q				
4	Whether the course focuses on value based education to be a socially committed professional?	a,b,c,d,e,h,i,l,m ,p,q				
5	Rate the role of the facilitator in mentoring and promoting the self learning abilities to excel academically and professionally?	a,b,c,d,e,h,i,l,m ,p,q				
6	Rate the methodology adopted and techniques used in teaching learning processes?	a,b,c,d,e,h,i,l,m ,p,q				
7	Rate the course in applying sciences & engineering fundamentals in providing research based conclusions with the help of modern tools?	a,b,c,d,e,h,i,l,m ,p,q				
8	Did the course have any scope to design, develop and test a system or component?	a,b,c,d,e,h,i,l,m ,p,q				
9	Rate the scope of this course in addressing cultural, legal, health, environment and safety issues?	a,b,c,d,e,h,i,l,m ,p,q				
10	Scope of applying management fundamentals to demonstrate effective technical project presentations & report writing?	a,b,c,d,e,h,i,l,m ,p,q				
Average						

9. OVERALL COURSE ASSESSMENT

(80% Direct + 20% Indirect, if any)

No	Assessment Type	Weightage	Attainment Level
1	Direct-Assignment, Quiz, Subjective, University Exams, Results, Bench Marks	0.80	
2	Indirect-Surveys-Questionnaire	0.20	
	Overall	1.00	

Course Attainment level:

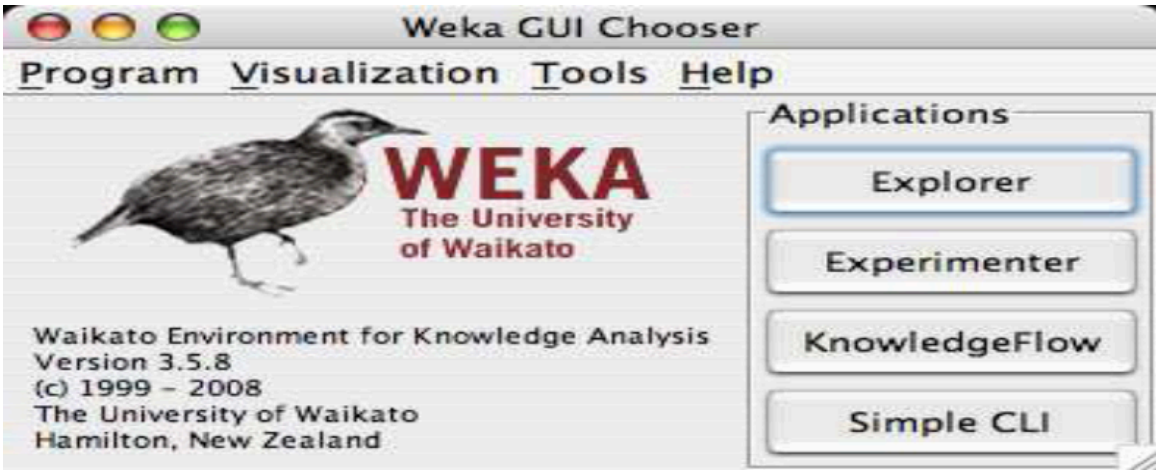
1. Weka Introduction

Weka is created by researchers at the university WIKATO in NewZealand. University of Waikato, Hamilton, New Zealand Alex Seewald (original Command-line primer) David Scuse (original Experimenter tutorial)

- It is java based application.
- It is collection often source, Machine Learning Algorithm.
- The routines (functions) are implemented as classes and logically arranged in packages.
- It comes with an extensive GUI Interface.
- Weka routines can be used standalone via the command line interface.

The Graphical User Interface

The Weka GUI Chooser (class weka.gui.GUIChooser) provides a starting point for launching Weka's main GUI applications and supporting tools. If one prefers a MDI ("multiple document interface") appearance, then this is provided by an alternative launcher called "Main" (class weka.gui.Main). The GUI Chooser consists of four buttons—one for each of the four major Weka applications—and four menus.



The buttons can be used to start the following applications:

- **Explorer An environment** for exploring data with WEKA (the rest of this documentation deals with this application in more detail).
- **Experimenter** An environment for performing experiments and conducting statistical tests between learning schemes.
- **Knowledge Flow** This environment supports essentially the same functions as the Explorer but with a drag-and-drop interface. One advantage is that it supports incremental learning.
- **SimpleCLI Provides** a simple command-line interface that allows direct execution of WEKA commands for operating systems that do not provide their own command line interface.

I. Explorer

The Graphical user interface

1.1 Section Tabs

At the very top of the window, just below the title bar, is a row of tabs. When the Explorer is first started only the first tab is active; the others are greyed out. This is because it is necessary to open (and potentially pre-process) a data set before starting to explore the data.

The tabs are as follows:

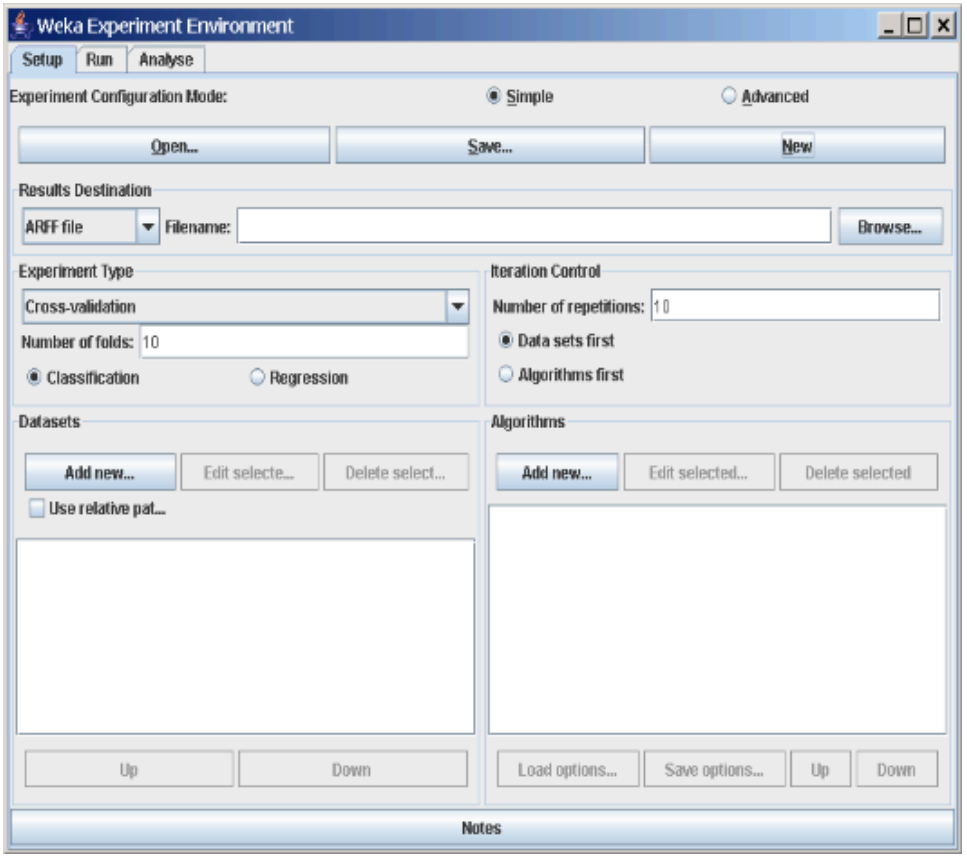
1. **Preprocess.** Choose and modify the data being acted on.
2. **Classify.** Train & test learning schemes that classify or perform regression
3. **Cluster.** Learn clusters for the data.
4. **Associate.** Learn association rules for the data.
5. **Select attributes.** Select the most relevant attributes in the data.
6. **Visualize.** View an interactive 2D plot of the data.

Once the tabs are active, clicking on them flicks between different screens, on which the respective actions can be performed. The bottom area of the window (including the status box, the log button, and the Weka bird) stays visible regardless of which section you are in. The Explorer can be easily extended with custom tabs. The Wiki article "Adding tabs in the Explorer" [7] explains this in detail.

II. Experimenter

2.1 Introduction

The Weka Experiment Environment enables the user to create, run, modify, and analyse experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyse the results to determine if one of the schemes is (statistically) better than the other schemes.



The Experiment Environment can be run from the command line using the Simple CLI. For example, the following commands could be typed into the CLI to run the OneR scheme on the Iris dataset using a basic train and test process. (Note that the commands would be typed on one line into the CLI.) While commands can be typed directly into the CLI, this technique is not particularly convenient and the experiments are not easy to modify. The Experimenter comes in two flavours, either with a simple interface that provides most of the functionality one needs for experiments, or with an interface with full access to the Experimenter's capabilities. You can choose between those two with the Experiment Configuration Mode radio buttons:

• Simple
• Advanced

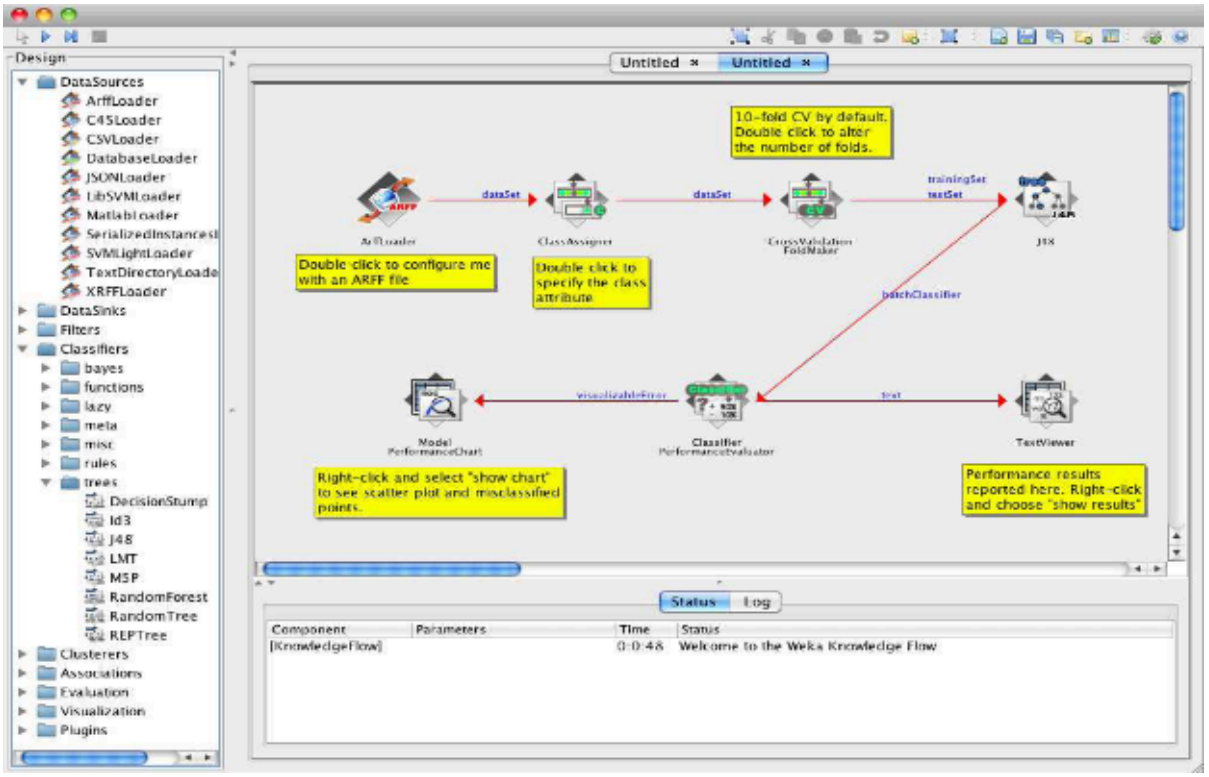
Both setups allow you to setup standard experiments, that are run locally on a single machine, or remote experiments, which are distributed between several hosts. The distribution of experiments cuts down the time the experiments will take until completion, but on the other hand the setup takes more time. The next section covers the standard experiments (both, simple and advanced), followed by the remote experiments and finally the analysing of the results.

III. Knowledge Flow

3.1 Introduction

The Knowledge Flow provides an alternative to the Explorer as a graphical front end to WEKA's core algorithms.

The KnowledgeFlow presents a data-flow inspired interface to WEKA. The user can selectWEKA components from a palette, place them on a layout canvas and connect them together in order to form a knowledge flow for processing and analyzing data. At present, all of WEKA's classifiers, filters, clusterers, associators, loaders and savers are available in the KnowledgeFlow along withsome extra tools.



The Knowledge Flow can handle data either incrementally or in batches (the Explorer handles batch data only). Of course learning from data incremen- tally requires a classifier that can be updated on an instance by instance basis. Currently in WEKA there are ten classifiers that can handle data incrementally.

The Knowledge Flow offers the following features:

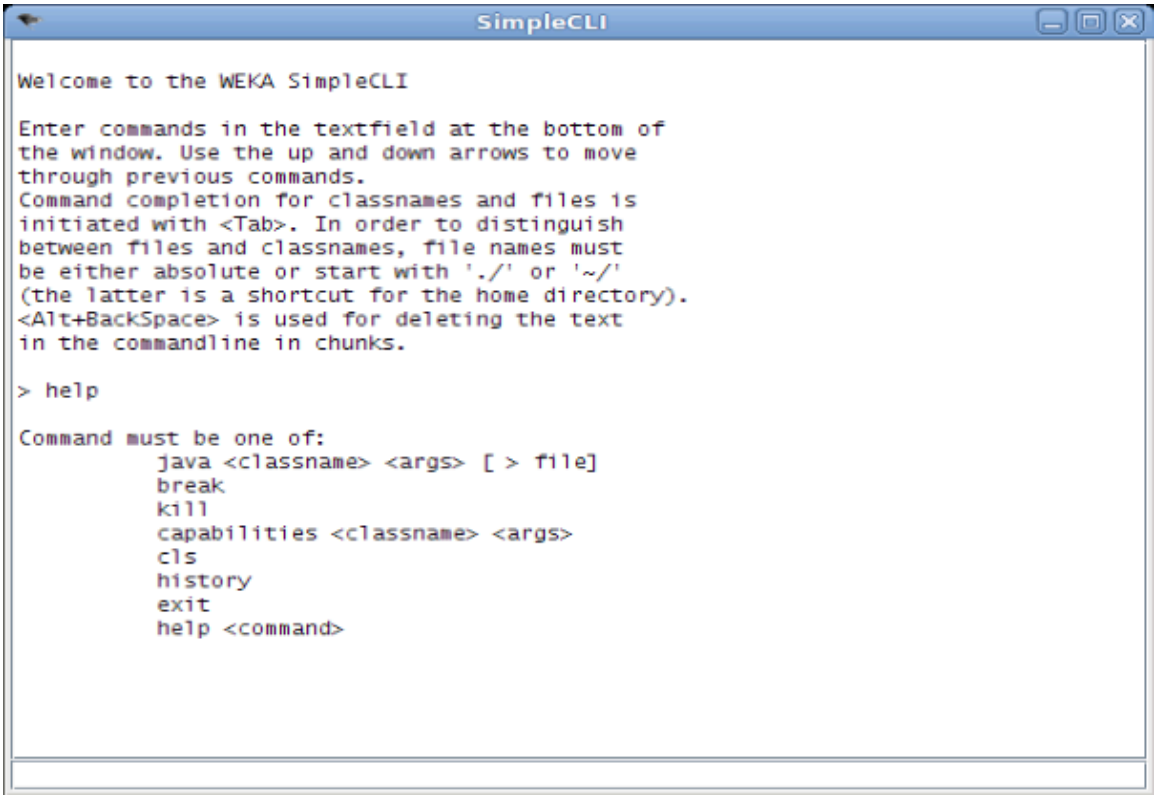
- **intuitive** data flow style layout
- **process** data in batches or incrementally
- **process multiple batches** or streams in parallel (each separate flow executes in its own thread)
- **process multiple streams sequentially** via a user-specified order of execution
- **chain filters** together
- **view models** produced by classifiers for each fold in a cross validation
- **visualize performance** of incremental classifiers during processing

(scrolling plots of classification accuracy, RMS error, predictions etc.)

• **plugin** "perspectives" that add major new functionality (e.g. 3D data visualization, time series forecasting environment etc.)

IV. Simple CLI

The Simple CLI provides full access to all Weka classes, i.e., classifiers, filters, clusterers, etc., but without the hassle of the CLASSPATH (it facilitates the one, with which Weka was started). It offers a simple Weka shell with separated command line and output.



4.1 Commands

The following commands are available in the Simple CLI:

• java <classname> [<args>]

invokes a java class with the given arguments (if any)

• break

stops the current thread, e.g., a running classifier, in a friendly manner

killstops the current thread in an unfriendly fashion

• cls

clears the output area

• capabilities <classname> [<args>]

lists the capabilities of the specified class, e.g., for a classifier with its option:

capabilities weka.classifiers.meta.Bagging -W weka.classifiers.trees.Id3

• exit

exits the Simple CLI

• help [<command>]

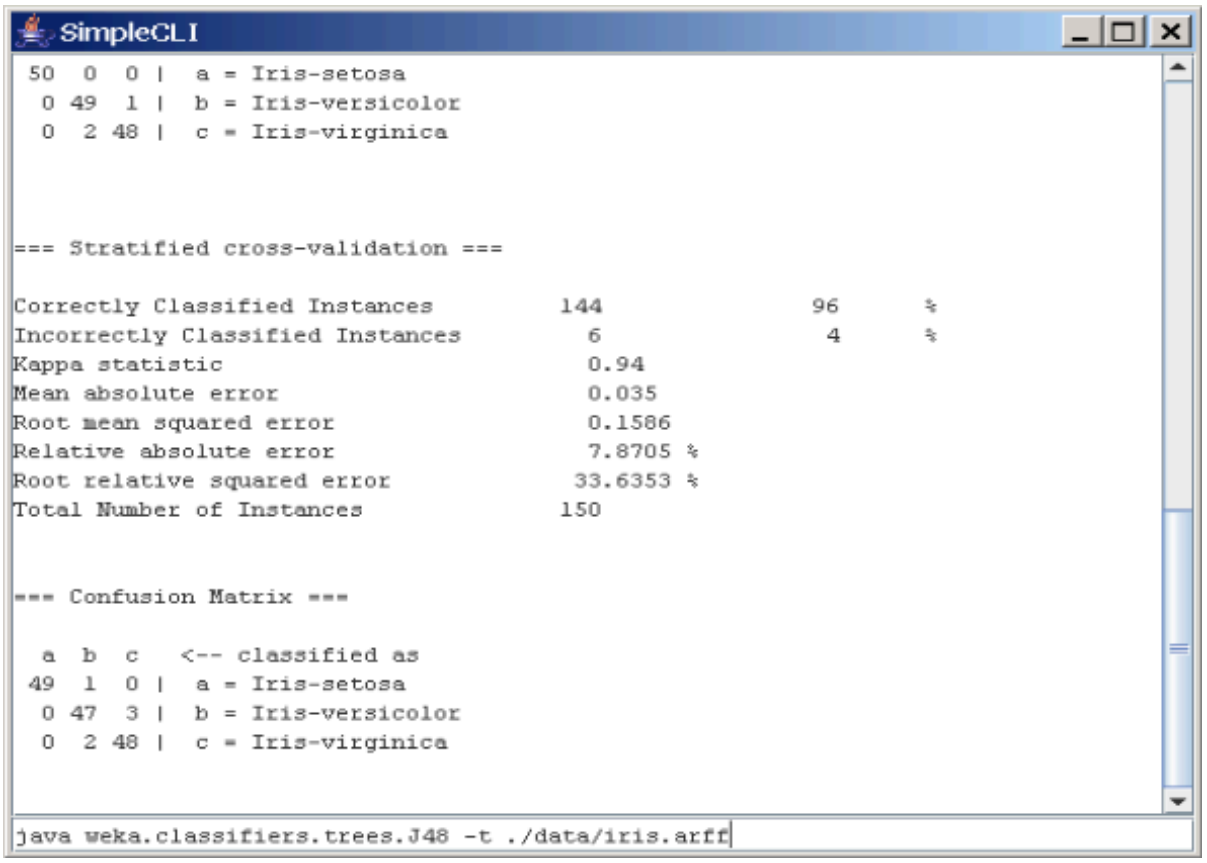
provides an overview of the available commands if without a command name as argument, otherwise more help on the specified command

4.2 Invocation

In order to invoke a Weka class, one has only to prefix the class with ”java”. This command tells the Simple CLI to load a class and execute it with any given parameters. E.g., the J48 classifier can be invoked on the iris dataset with the following command:

java weka.classifiers.trees.J48 -t c:/temp/iris.arff

This results in the following output:



4.3 Command redirection

Starting with this version of Weka one can perform a basic redirection:

java weka.classifiers.trees.J48 -t test.arff > j48.txt

Note: the > must be preceded and followed by a space, otherwise it is not recognized as redirection, but part of another parameter.

4.4 Command completion

Commands starting with java support completion for classnames and filenames via Tab (Alt+BackSpace deletes parts of the command again). In case that there are several matches, Weka lists all possible matches.

• package name completion

java weka.cl<Tab>

results in the following output of possible matches of package names:

Possible matches:

weka.classifiers

weka.clusterers

• classname completion

java weka.classifiers.meta.A<Tab>

lists the following classes

Possible matches:

weka.classifiers.meta.AdaBoostM1

weka.classifiers.meta.AdditiveRegression

weka.classifiers.meta.AttributeSelectedClassifier

• filename completion

In order for Weka to determine whether a the string under the cursor is a classname or a filename, filenames need to be absolute (Unix/Linx: /some/path/file;Windows: C:\Some\Path\file) or relative and starting with a dot (Unix/Linux: ./some/other/path/file;Windows: .\Some\Other\Path\file).

ARFF File Format

An ARFF (= Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes.

ARFF files are not the only format one can load, but all files that can be converted with Weka's "core converters". The following formats are currently supported:

• ARFF (+ compressed)
• C4.5
• CSV
• libsvm
• binary serialized instances
• XRFF (+ compressed)

Overview

ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information. The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types.

An example header on the standard IRIS dataset looks like this:

% 1. Title: Iris Plants Database
%
% 2. Sources:
% (a) Creator: R.A. Fisher
% (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
% (c) Date: July, 1988
%
@RELATION iris
@ATTRIBUTE sepallength NUMERIC
@ATTRIBUTE sepalwidth NUMERIC
@ATTRIBUTE petallength NUMERIC
@ATTRIBUTE petalwidth NUMERIC
@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}
The Data of the ARFF file looks like the following:
@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
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
Lines that begin with a % are comments.
The @RELATION, @ATTRIBUTE and @DATA declarations are case insensitive.

The ARFF Header Section

The ARFF Header section of the file contains the relation declaration and attribute declarations.

The @relation Declaration

The relation name is defined as the first line in the ARFF file. The format is:
@relation <relation-name>

where <relation-name> is a string. The string must be quoted if the name includes spaces.

The @attribute Declarations

Attribute declarations take the form of an ordered sequence of @attribute statements. Each attribute in the data set has its own @attribute statement which uniquely defines the name of that attribute and it's data type. The order the attributes are declared indicates the column position in the data section of the file. For example, if an attribute is the third one

declared then Weka expects that all that attributes values will be found in the third comma delimited column.

The format for the @attribute statement is:

@attribute <attribute-name> <datatype>

where the <attribute-name> must start with an alphabetic character. If spaces are to be included in the name then the entire name must be quoted.

The <datatype> can be any of the four types supported by Weka:

- numeric
- integer is treated as numeric
- real is treated as numeric
- <nominal-specification>
- string
- date [<date-format>]
- relational for multi-instance data (for future use)

where <nominal-specification> and <date-format> are defined below. The keywords numeric, real, integer, string and date are case insensitive.

Numeric attributes

Numeric attributes can be real or integer numbers.

Nominal attributes

Nominal values are defined by providing an <nominal-specification> listing the possible values: <nominal-name1>, <nominal-name2>, <nominal-name3>,

...

For example, the class value of the Iris dataset can be defined as follows:

@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

Values that contain spaces must be quoted.

String attributes

String attributes allow us to create attributes containing arbitrary textual values. This is very useful in text-mining applications, as we can create datasets with string attributes, then writeWeka Filters to manipulate strings (like String- ToWordVectorFilter). String attributes are declared as follows:

@ATTRIBUTE LCC string

Date attributes

Date attribute declarations take the form:

@attribute <name> date [<date-format>]

where <name> is the name for the attribute and <date-format> is an optional string specifying how date values should be parsed and printed (this is the same format used by SimpleDateFormat). The default format string accepts the ISO-8601 combined date and time format: yyyy-MM-dd'T'HH:mm:ss. Dates must be specified in the data section as the corresponding string representations of the date/time (see example below).

Relational attributes

Relational attribute declarations take the form:

@attribute <name> relational
<further attribute definitions>
@end <name>
For the multi-instance dataset MUSK1 the definition would look like this (”...” denotes an omission):
@attribute molecule_name {MUSK-jf78,...,NON-MUSK-199}
@attribute bag relational
@attribute f1 numeric
...
@attribute f166 numeric
@end bag
@attribute class {0,1}
...

The ARFF Data Section
The ARFF Data section of the file contains the data declaration line and the actual instance lines.

The @data Declaration
The @data declaration is a single line denoting the start of the data segment in the file. The format is:
@data
The instance data
Each instance is represented on a single line, with carriage returns denoting the end of the instance. A percent sign (%) introduces a comment, which continues to the end of the line.

Attribute values for each instance are delimited by commas. They must appear in the order that they were declared in the header section (i.e. the data corresponding to the nth @attribute declaration is always the nth field of the attribute).
Missing values are represented by a single question mark, as in:
@data
4.4,?,1.5,?,Iris-setosa

Values of string and nominal attributes are case sensitive, and any that contain space or the comment-delimiter character % must be quoted. (The code suggests that double-quotes are acceptable and that a backslash will escape individual characters.)

An example follows:
@relation LCCvsLCSH
@attribute LCC string
@attribute LCSH string
@data

AG5, ’Encyclopedias and dictionaries.;Twentieth century.’
AS262, ’Science -- Soviet Union -- History.’
AE5, ’Encyclopedias and dictionaries.’
AS281, ’Astronomy, Assyro-Babylonian.;Moon -- Phases.’
AS281, ’Astronomy, Assyro-Babylonian.;Moon -- Tables.’

Dates must be specified in the data section using the string representation specified in the attribute declaration.

For example:
@RELATION Timestamps
@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"
@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
Relational data must be enclosed within double quotes ”. For example an instance of the MUSK1 dataset (”...” denotes an omission):

MUSK-188,"42,...,30",1

Preprocess Tab

1. Loading Data

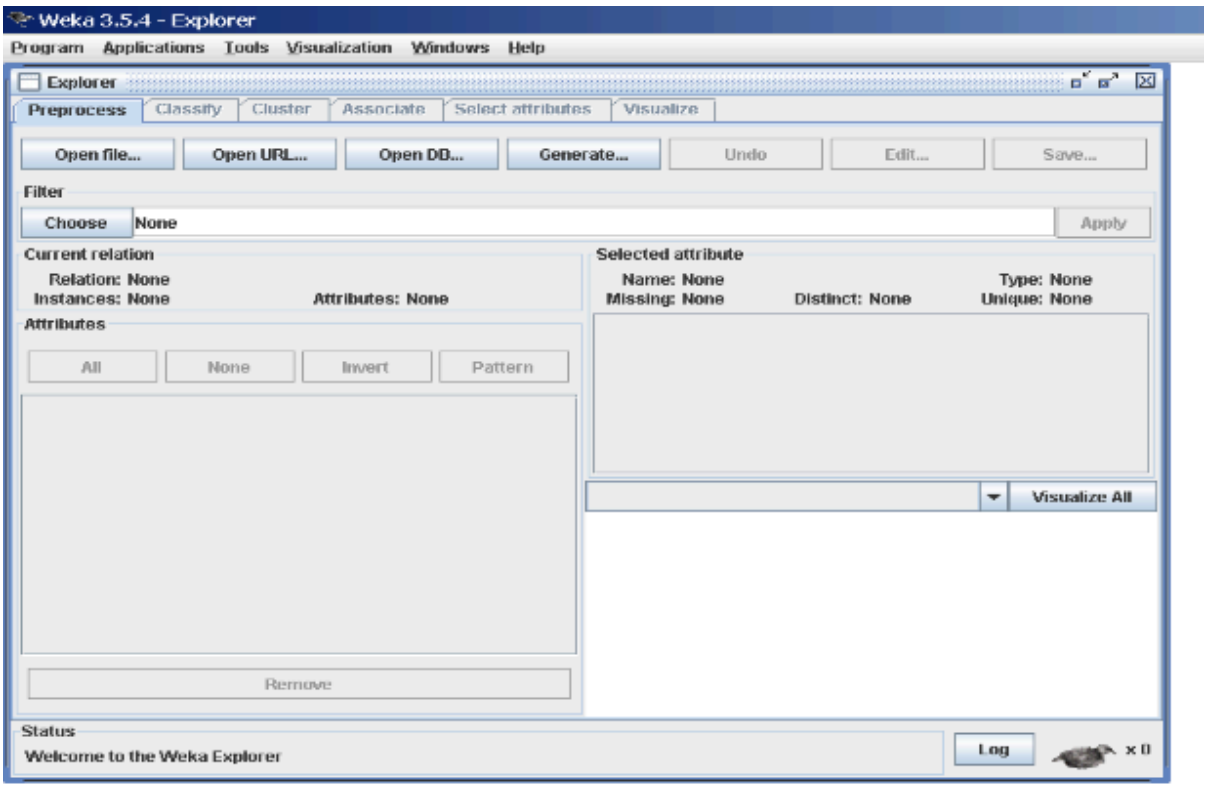
The first four buttons at the top of the preprocess section enable you to load data into WEKA:

1. Open file.... Brings up a dialog box allowing you to browse for the data file on the local file system.

2. Open URL.... Asks for a Uniform Resource Locator address for where the data is stored.

3. Open DB.... Reads data from a database. (Note that to make this work you might have to edit the file in weka/experiment/DatabaseUtils.props.)

4. Generate.... Enables you to generate artificial data from a variety of Data Generators. Using the Open file... button you can read files in a variety of formats: WEKA’s ARFF format, CSV format, C4.5 format, or serialized Instances format. ARFF files typically have a .arff extension, CSV files a .csv extension, C4.5 files a .data and .names extension, and serialized Instances objects a .bsi extension.



The Current Relation: Once some data has been loaded, the Preprocess panel shows a variety of information. The Current relation box (the "current relation" is the currently loaded data, which can be interpreted as a single relational table in database terminology) has three entries:

1. Relation. The name of the relation, as given in the file it was loaded from. Filters (described below) modify the name of a relation.
2. Instances. The number of instances (data points/records) in the data.
3. Attributes. The number of attributes (features) in the data.

Working With Attributes

Below the Current relation box is a box titled Attributes. There are four buttons, and beneath them is a list of the attributes in the current relation.

The list has three columns:
1. No.. A number that identifies the attribute in the order they are specified in the data file.
2. Selection tick boxes. These allow you select which attributes are present in the relation.
3. Name. The name of the attribute, as it was declared in the data file. When you click on different rows in the list of attributes, the fields change in the box to the right titled Selected attribute. This box displays the characteristics of the currently highlighted attribute in the list:

1. Name. The name of the attribute, the same as that given in the attribute list.
2. Type. The type of attribute, most commonly Nominal or Numeric.
3. Missing. The number (and percentage) of instances in the data for which this attribute is missing (unspecified).
4. Distinct. The number of different values that the data contains for this attribute.
5. Unique. The number (and percentage) of instances in the data having a value for this attribute that no other instances have.

Below these statistics is a list showing more information about the values stored in this attribute, which differ depending on its type. If the attribute is nominal, the list consists of each possible value for the attribute along with the number of instances that have that value. If the attribute is numeric, the list gives four statistics describing the distribution of values in the data—the minimum, maximum, mean and standard deviation. And below these statistics there is a coloured histogram, colour-coded according to the attribute chosen as the Class using the box above the histogram. (This box will bring up a drop-down list of available selections when clicked.) Note that only nominal Class attributes will result in a colour-coding. Finally, after pressing the Visualize All button, histograms for all the attributes in the data are shown in a separate window.

Returning to the attribute list, to begin with all the tick boxes are unticked. They can be toggled on/off by clicking on them individually. The four buttons above can also be used to change the selection:

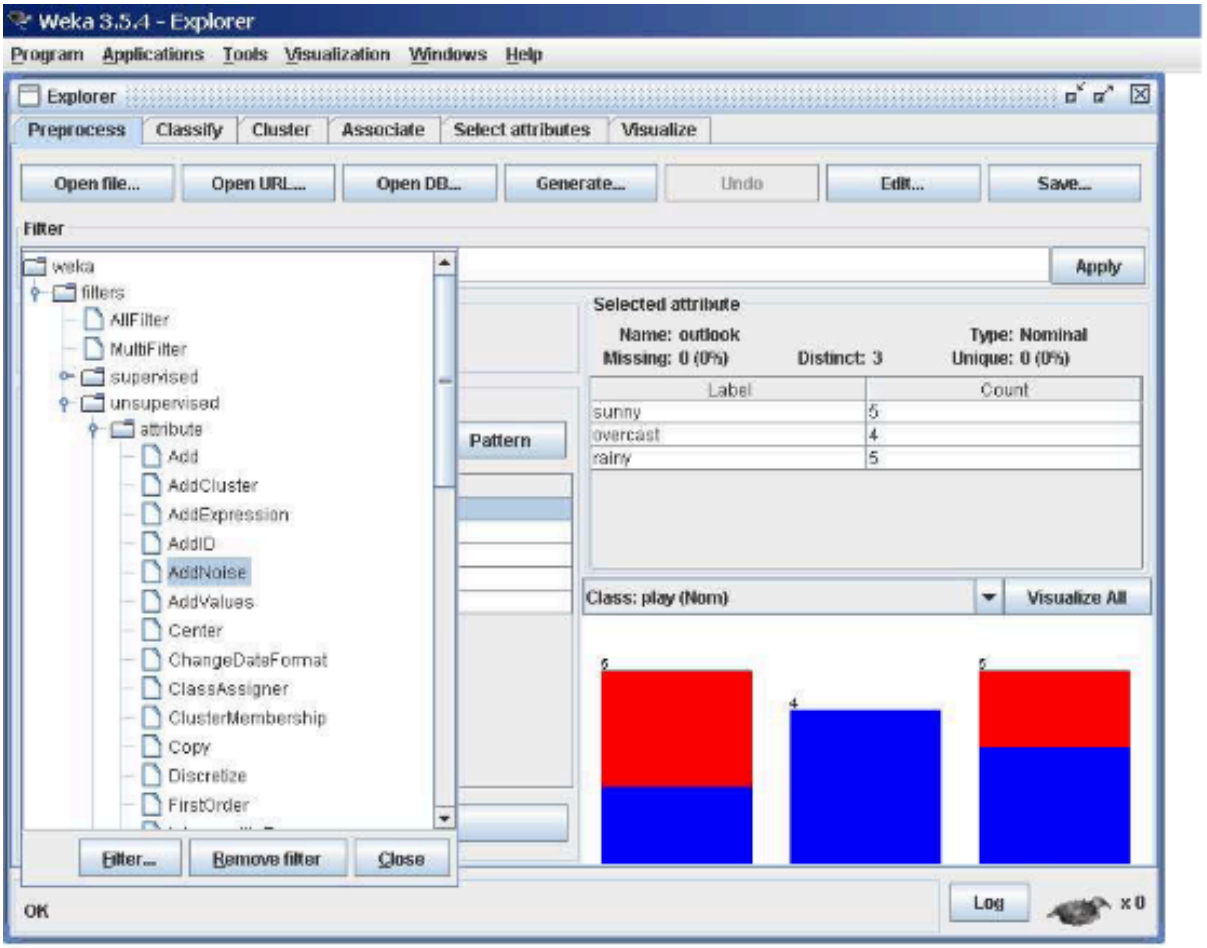
PREPROCESSING
1. All. All boxes are ticked.
2. None. All boxes are cleared (unticked).
3. Invert. Boxes that are ticked become unticked and vice versa.
4. Pattern. Enables the user to select attributes based on a Perl 5 Regular Expression. E.g., .* id selects all attributes which name ends with id.

Once the desired attributes have been selected, they can be removed by clicking the Remove button below the list of attributes. Note that this can be undone by clicking the Undo button, which is located next to the Edit button in the top-right corner of the Preprocess panel.

Working With Filters

The preprocess section allows filters to be defined that transform the data in various ways. The Filter box is used to set up the filters that are required. At the left of the Filter box is a Choose button. By clicking this button it is possible to select one of the filters in WEKA. Once a filter has been selected, its name and options are shown in the field next to the Choose button. Clicking on this box with the left mouse button brings up a GenericObjectEditor dialog box. A click with the right mouse button (or Alt+Shift+left click) brings up a menu

where you can choose, either to display the properties in a GenericObjectEditor dialog box, or to copy the current setup string to the clipboard.



The GenericObjectEditor Dialog Box

The GenericObjectEditor dialog box lets you configure a filter. The same kind of dialog box is used to configure other objects, such as classifiers and clusterers (see below). The fields in the window reflect the available options. Right-clicking (or Alt+Shift+Left-Click) on such a field will bring up a popup menu, listing the following options:

1. Show properties... has the same effect as left-clicking on the field, i.e., a dialog appears allowing you to alter the settings.

2. Copy configuration to clipboard copies the currently displayed configuration string to the system's clipboard and therefore can be used anywhere else in WEKA or in the console. This is rather handy if you have to setup complicated, nested schemes.

3. Enter configuration... is the "receiving" end for configurations that got copied to the clipboard earlier on. In this dialog you can enter a class name followed by options (if the

class supports these). This also allows you to transfer a filter setting from the Preprocess panel to a Filtered Classifier used in the Classify panel.

Left-Clicking on any of these gives an opportunity to alter the filters settings. For example, the setting may take a text string, in which case you type the string into the text field provided. Or it may give a drop-down box listing several states to choose from. Or it may do something else, depending on the information required. Information on the options is provided in a tool tip if you let the mouse pointer hover of the corresponding field. More information on the filter and its options can be obtained by clicking on the More button in the About panel at the top of the GenericObjectEditor window.

Some objects display a brief description of what they do in an About box, along with a More button. Clicking on the More button brings up a window describing what the different options do. Others have an additional button, Capabilities, which lists the types of attributes and classes the object can handle.

At the bottom of the GenericObjectEditor dialog are four buttons. The first two, Open... and Save... allow object configurations to be stored for future use. The Cancel button backs out without remembering any changes that have been made. Once you are happy with the object and settings you have chosen, click OK to return to the main Explorer window.

Applying Filters

Once you have selected and configured a filter, you can apply it to the data by pressing the Apply button at the right end of the Filter panel in the Preprocess panel. The Preprocess panel will then show the transformed data. The change can be undone by pressing the Undo button. You can also use the Edit...button to modify your data manually in a dataset editor. Finally, the Save... button at the top right of the Preprocess panel saves the current version of the relation in file formats that can represent the relation, allowing it to be kept for future use.

Note: Some of the filters behave differently depending on whether a class attribute has been set or not (using the box above the histogram, which will bring up a drop-down list of possible selections when clicked). In particular, the “supervised filters” require a class attribute to be set, and some of the “unsupervised attribute filters” will skip the class attribute if one is set. Note that it is also possible to set Class to None, in which case no class is set.

5. Classification Tab

5.1 Selecting a Classifier

At the top of the classify section is the Classifier box. This box has a text field that gives the name of the currently selected classifier, and its options. Clicking on the text box with the left mouse button brings up a GenericObjectEditor dialog box, just the same as for filters, that you can use to configure the options of the current classifier. With a right click (or Alt+Shift+left click) you can once again copy the setup string to the clipboard or display the

properties in a GenericObjectEditor dialog box. The Choose button allows you to choose one of the classifiers that are available in WEKA.

5.2 Test Options

The result of applying the chosen classifier will be tested according to the options that are set by clicking in the Test options box. There are four test modes:

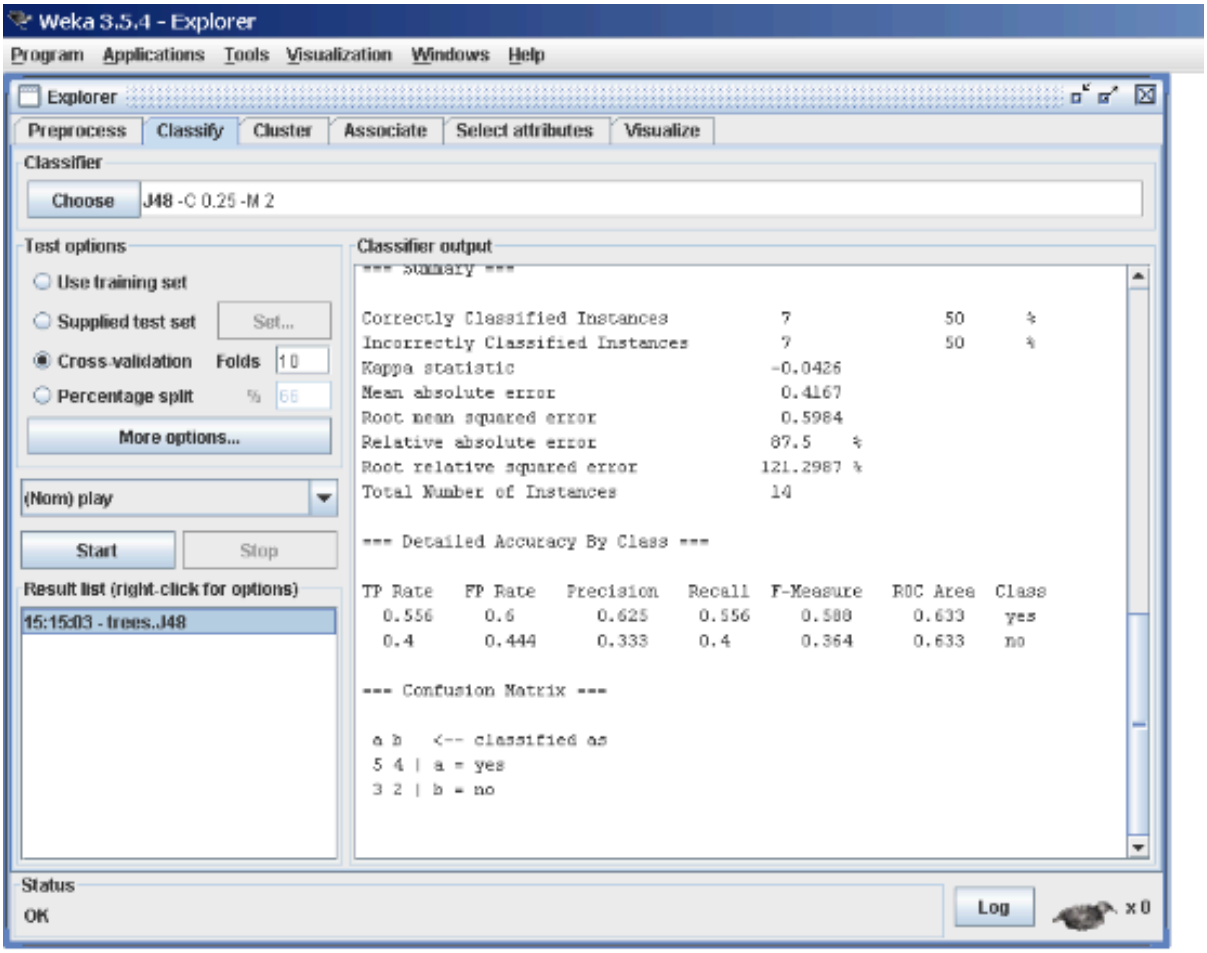
1. Use training set. The classifier is evaluated on how well it predicts the class of the instances it was trained on.

2. Supplied test set. The classifier is evaluated on how well it predicts the class of a set of instances loaded from a file. Clicking the Set... button brings up a dialog allowing you to choose the file to test on.

3. Cross-validation. The classifier is evaluated by cross-validation, using the number of folds that are entered in the Folds text field.

4. Percentage split. The classifier is evaluated on how well it predicts a certain percentage of the data which is held out for testing. The amount of data held out depends on the value entered in the % field.

Note: No matter which evaluation method is used, the model that is output is always the one build from all the training data. Further testing options can be set by clicking on the More options... button:



1. Output model. The classification model on the full training set is output so that it can be viewed, visualized, etc. This option is selected by default.

2. Output per-class stats. The precision/recall and true/false statistics for each class are output. This option is also selected by default.

3. Output entropy evaluation measures. Entropy evaluation measures are included in the output. This option is not selected by default.

4. Output confusion matrix. The confusion matrix of the classifier's predictions is included in the output. This option is selected by default.

5. Store predictions for visualization. The classifier's predictions are remembered so that they can be visualized. This option is selected by default.

6. Output predictions. The predictions on the evaluation data are output.

Note that in the case of a cross-validation the instance numbers do not correspond to the location in the data!

7. Output additional attributes. If additional attributes need to be output alongside the predictions, e.g., an ID attribute for tracking misclassifications, then the index of this attribute can be specified here. The usual Weka ranges are supported,"first" and "last" are therefore valid indices as well (example: "first-3,6,8,12-last").

8. Cost-sensitive evaluation. The errors is evaluated with respect to a cost matrix. The Set... button allows you to specify the cost matrix used.

9. Random seed for xval / % Split. This specifies the random seed used when randomizing the data before it is divided up for evaluation purposes.

10. Preserve order for % Split. This suppresses the randomization of the data before splitting into train and test set.

11. Output source code. If the classifier can output the built model as Java source code, you can specify the class name here. The code will be printed in the "Classifier output" area.

5.3 The Class Attribute

The classifiers in WEKA are designed to be trained to predict a single 'class' attribute, which is the target for prediction. Some classifiers can only learn nominal classes; others can only learn numeric classes (regression problems) still others can learn both.

By default, the class is taken to be the last attribute in the data. If you want to train a classifier to predict a different attribute, click on the box below the Test options box to bring up a drop-down list of attributes to choose from.

5.4 Training a Classifier

Once the classifier, test options and class have all been set, the learning process is started by clicking on the Start button. While the classifier is busy being trained, the little bird moves around. You can stop the training process at any time by clicking on the Stop button. When training is complete, several things happen. The Classifier output area to the right of the display is filled with text describing the results of training and testing. A new entry appears in the Result list box. We look at the result list below; but first we investigate the text that has been output.

5.5 The Classifier Output Text

The text in the Classifier output area has scroll bars allowing you to browse the results. Clicking with the left mouse button into the text area, while holding Alt and Shift, brings up a dialog that enables you to save the displayed output in a variety of formats

(currently, BMP, EPS, JPEG and PNG). Of course, you can also resize the Explorer window to get a larger display area.

The output is

Split into several sections:

1. Run information. A list of information giving the learning scheme options, relation name, instances, attributes and test mode that were involved in the process.

2. Classifier model (full training set). A textual representation of the classification model that was produced on the full training data.

3. The results of the chosen test mode are broken down thus.

4. Summary. A list of statistics summarizing how accurately the classifier was able to predict the true class of the instances under the chosen test mode.

5. Detailed Accuracy By Class. A more detailed per-class break down of the classifier's prediction accuracy.

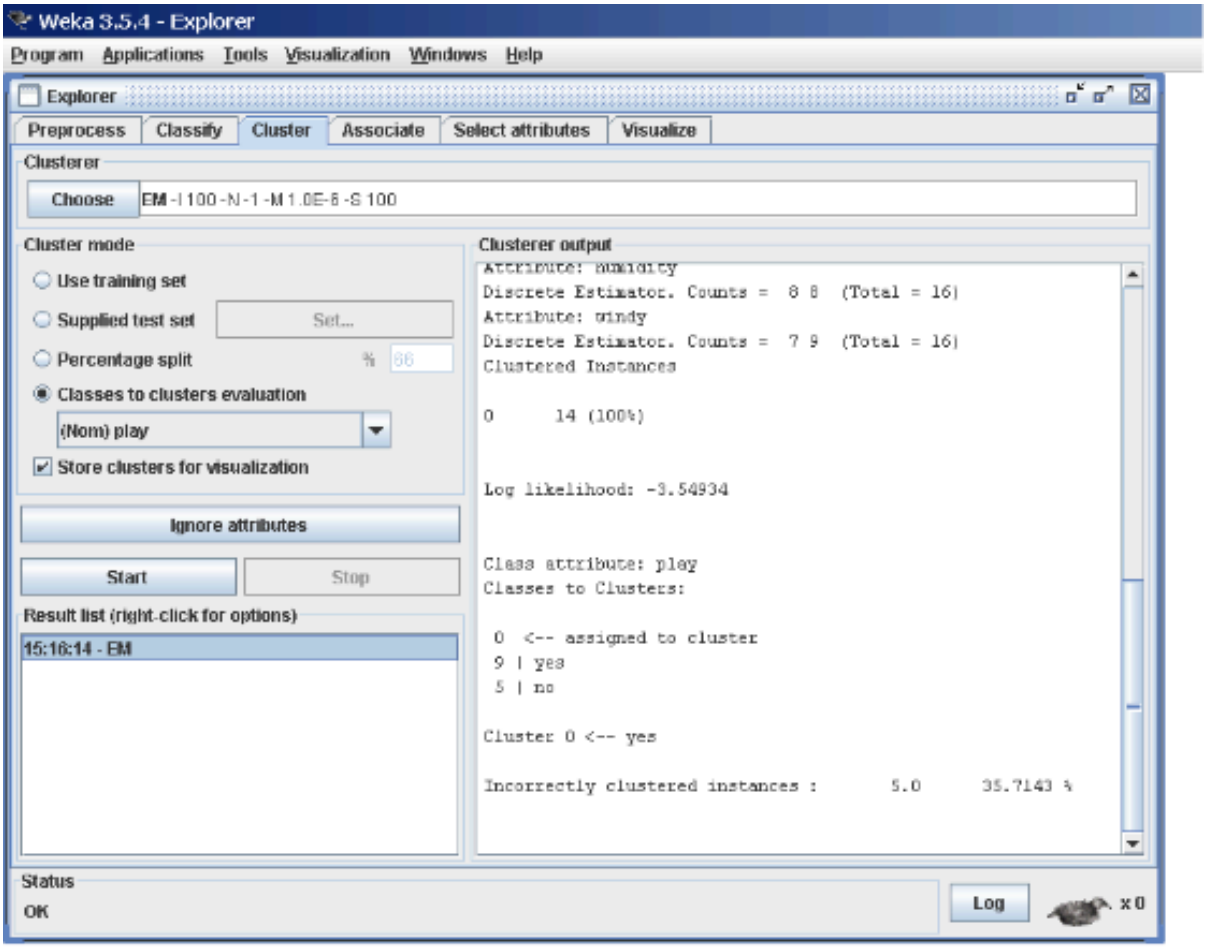
6. Confusion Matrix. Shows how many instances have been assigned to each class. Elements show the number of test examples whose actual class is the row and whose predicted class is the column.

7. Source code (optional). This section lists the Java source code if one chose "Output source code" in the "More options" dialog.

6. Clustering Tab

6.1 Selecting a Clusterer

By now you will be familiar with the process of selecting and configuring objects. Clicking on the clustering scheme listed in the Clusterer box at the top of the window brings up a GenericObjectEditor dialog with which to choose a new clustering scheme.



Cluster Modes

The Cluster mode box is used to choose what to cluster and how to evaluate the results. The first three options are the same as for classification: Use training set, Supplied test set and Percentage split (Section 5.3.1)—except that now the data is assigned to clusters instead of trying to predict a specific class. The fourth mode, Classes to clusters evaluation, compares how well the chosen clusters match up with a pre-assigned class in the data. The drop-down box below this option selects the class, just as in the Classify panel.

An additional option in the Cluster mode box, the Store clusters for visualization tick box, determines whether or not it will be possible to visualize the clusters once training is complete. When dealing with datasets that are so
large that memory becomes a problem it may be helpful to disable this option.

6.2 Ignoring Attributes

Often, some attributes in the data should be ignored when clustering. The Ignore attributes button brings up a small window that allows you to select which attributes are ignored. Clicking on an attribute in the window highlights it, holding down the SHIFT key selects a range of consecutive attributes, and holding down CTRL toggles individual attributes on and off. To cancel the selection, back out with the Cancel button. To activate it, click the Select button. The next time clustering is invoked, the selected attributes are ignored.

6.3 Working with Filters

The Filtered Clusterer meta-clusterer offers the user the possibility to apply filters directly before the clusterer is learned. This approach eliminates the manual application of a filter in the Preprocess panel, since the data gets processed on the fly. Useful if one needs to try out different filter setups.

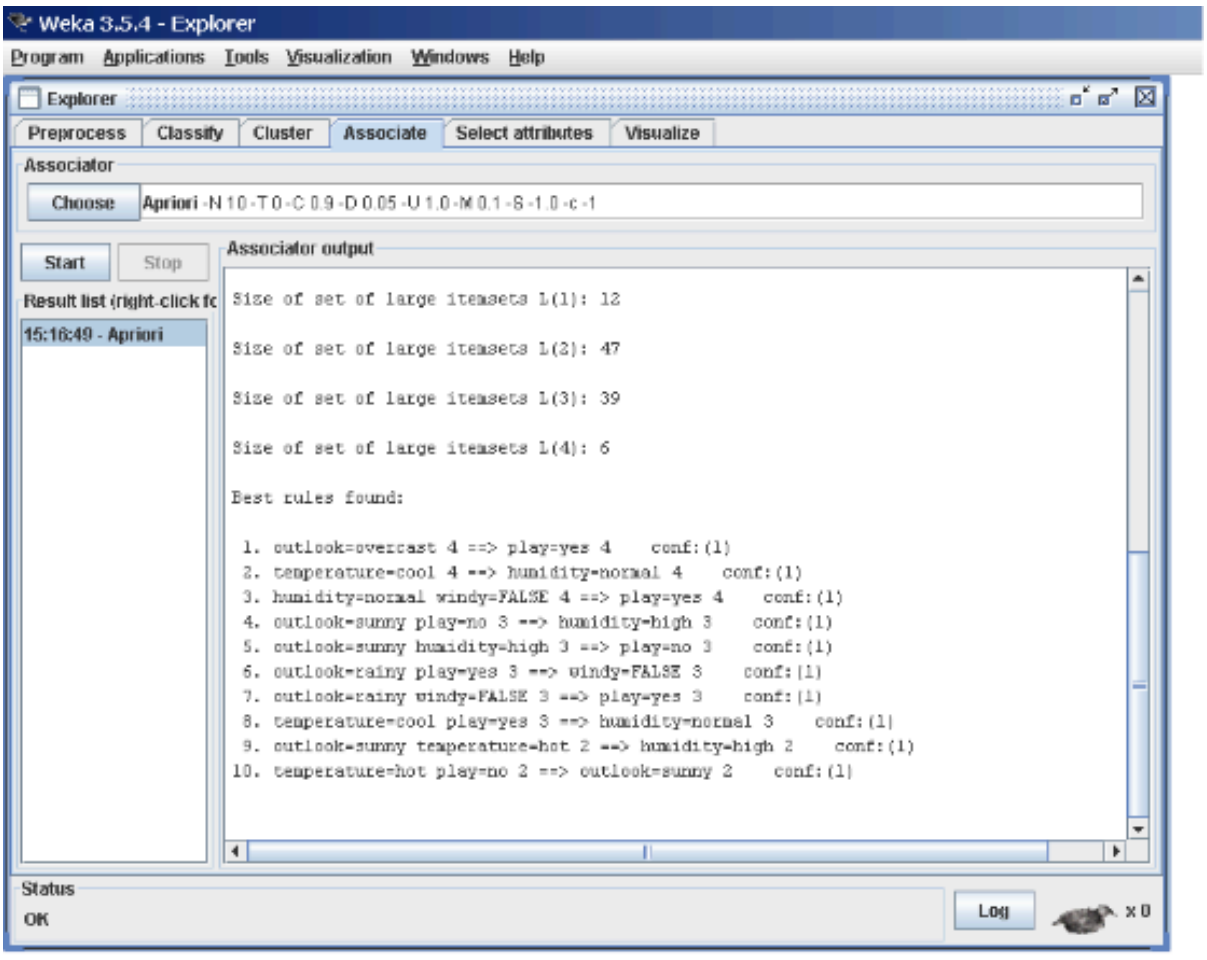
6.4 Learning Clusters

The Cluster section, like the Classify section, has Start/Stop buttons, a result text area and a result list. These all behave just like their classification counterparts. Right-clicking an entry in the result list brings up a similar menu, except that it shows only two visualization options: Visualize cluster assignments and Visualize tree. The latter is grayed out when it is not applicable.

7. Associate Tab

7.1 Setting Up

This panel contains schemes for learning association rules, and the learners are chosen and configured in the same way as the clusterers, filters, and classifiers in the other panels.



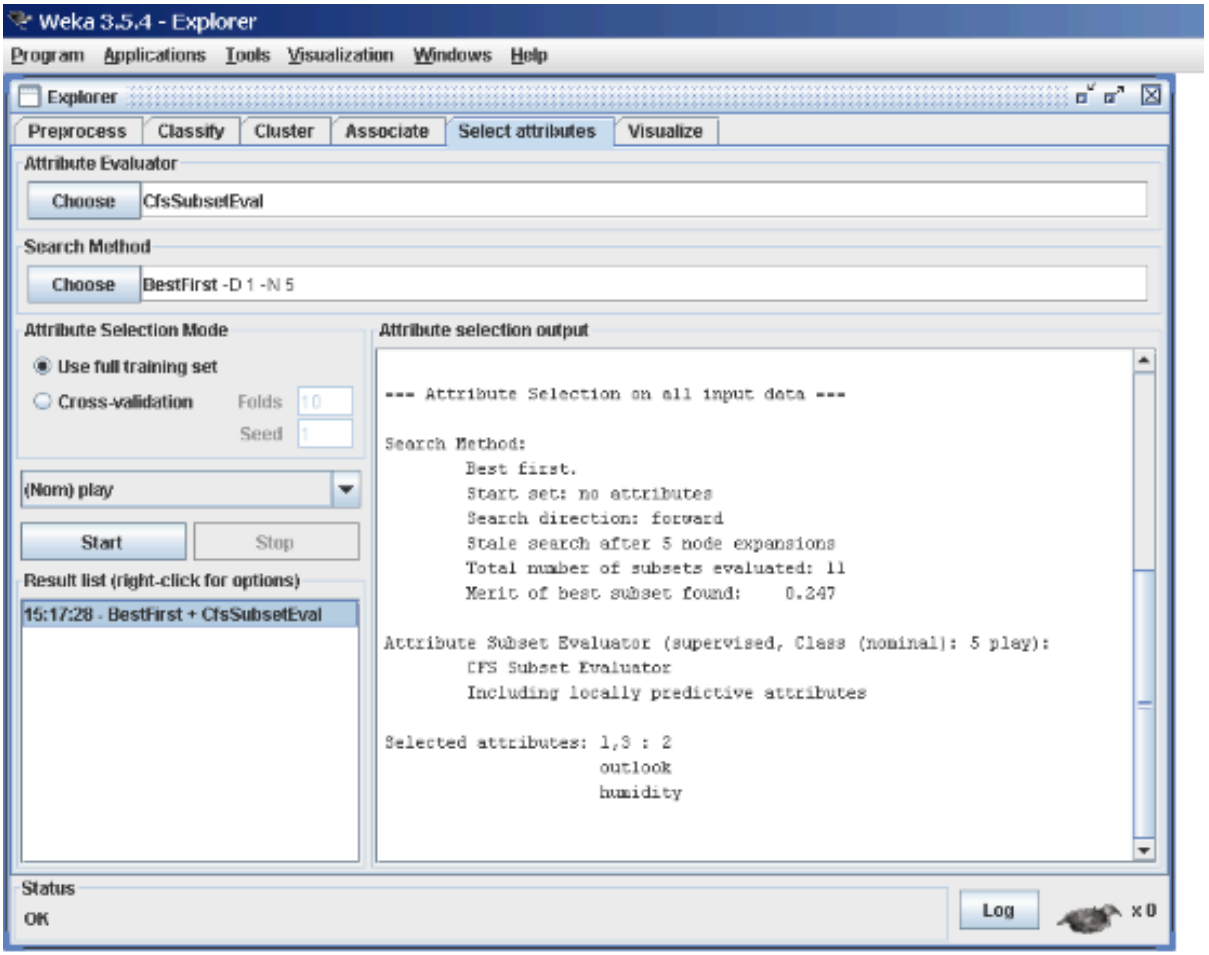
7.2 Learning Associations

Once appropriate parameters for the association rule learner have been set, click the Start button. When complete, right-clicking on an entry in the result list allows the results to be viewed or saved.

8. Selecting Attributes Tab

8.1 Searching and Evaluating

Attribute selection involves searching through all possible combinations of attributes in the data to find which subset of attributes works best for prediction.
To do this, two objects must be set up: an attribute evaluator and a search method. The evaluator determines what method is used to assign a worth to each subset of attributes. The search method determines what style of search is performed.



8.2 Options

The Attribute Selection Mode box has two options:

1. Use full training set. The worth of the attribute subset is determined using the full set of training data.

2. Cross-validation. The worth of the attribute subset is determined by a process of cross-validation. The Fold and Seed fields set the number of folds to use and the random seed used when shuffling the data. As with Classify (Section 5.3.1), there is a drop-down box that can be used to specify which attribute to treat as the class.

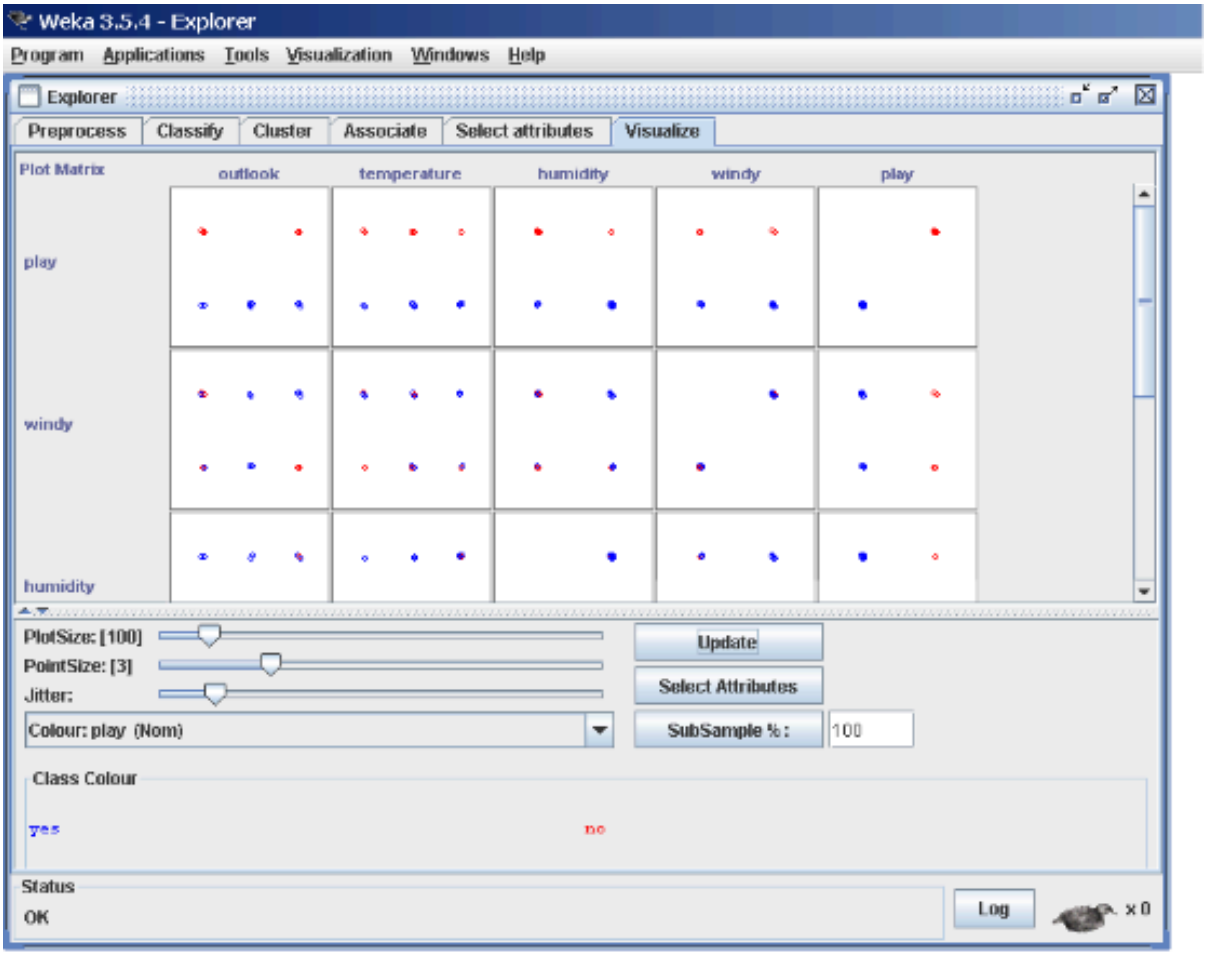
8.3 Performing Selection

Clicking Start starts running the attribute selection process. When it is finished, the results are output into the result area, and an entry is added to the result list. Right-clicking on the result list gives several options. The first three, (View in main window, View in separate window and Save result buffer), are the same as for the classify panel. It is also possible to Visualize reduced data, or if you have used an attribute transformer such as Principal Components, Visualize transformed data. The reduced/transformed data can be saved to a file with the Save reduced data... or Save transformed data... option.

In case one wants to reduce/transform a training and a test at the same time and not use the Attribute Selected Classifier from the classifier panel, it is best to use the Attribute Selection filter (a supervised attribute filter) in batch mode ('-b') from the command line or in the Simple CLI. The batch mode allows one to specify an additional input and output file pair (options -r and -s), that is processed with the filter setup that was determined based on the training data.

9. Visualizing Tab

WEKA's visualization section allows you to visualize 2D plots of the current relation.



9.1 The scatter plot matrix

When you select the Visualize panel, it shows a scatter plot matrix for all the attributes, colour coded according to the currently selected class. It is possible to change the size of each individual 2D plot and the point size, and to randomly jitter the data (to uncover obscured points). It also possible to change the attribute used to colour the plots, to select only a subset of attributes for inclusion in the scatter plot matrix, and to sub sample the data. Note that changes will only come into effect once the Update button has been pressed.

9.2 Selecting an individual 2D scatter plot

When you click on a cell in the scatter plot matrix, this will bring up a separate window with a visualization of the scatter plot you selected. (We described above how to visualize particular results in a separate window—for example, classifier errors—the same visualization controls are used here.)

Data points are plotted in the main area of the window. At the top are two drop-down list buttons for selecting the axes to plot. The one on the left shows which attribute is used for the x-axis; the one on the right shows which is used for the y-axis.

Beneath the x-axis selector is a drop-down list for choosing the colour scheme. This allows you to colour the points based on the attribute selected. Below the plot area, a legend describes what values the colours correspond to. If the values are discrete, you can modify the colour used for each one by clicking on them and making an appropriate selection in the window that pops up.

To the right of the plot area is a series of horizontal strips. Each strip represents an attribute, and the dots within it show the distribution of values of the attribute. These values are randomly scattered vertically to help you see concentrations of points. You can choose what axes are used in the main graph by clicking on these strips. Left-clicking an attribute strip changes the x-axis to that attribute, whereas right-clicking changes the y-axis. The 'X' and 'Y' written beside the strips shows what the current axes are ('B' is used for 'both X and Y').

Above the attribute strips is a slider labelled Jitter, which is a random displacement given to all points in the plot. Dragging it to the right increases the amount of jitter, which is useful for spotting concentrations of points. Without jitter, a million instances at the same point would look no different to just a
single lonely instance.

9.3 Selecting Instances

There may be situations where it is helpful to select a subset of the data using the visualization tool. (A special case of this is the User Classifier in the Classify panel, which lets you build your own classifier by interactively selecting instances.)

Below the y-axis selector button is a drop-down list button for choosing a selection method. A group of data points can be selected in four ways:

1. Select Instance. Clicking on an individual data point brings up a window listing its attributes. If more than one point appears at the same location, more than one set of attributes is shown.

2. Rectangle. You can create a rectangle, by dragging, that selects the points inside it.

3. Polygon. You can build a free-form polygon that selects the points inside it. Left-click to add vertices to the polygon, right-click to complete it. The polygon will always be closed off by connecting the first point to the last.

4. Polyline. You can build a polyline that distinguishes the points on one side from those on the other. Left-click to add vertices to the polyline, right-click to finish. The resulting shape is open (as opposed to a polygon, which is always closed).

Once an area of the plot has been selected using Rectangle, Polygon or Polyline, it turns grey. At this point, clicking the Submit button removes all instances from the plot

except those within the grey selection area. Clicking on the Clear button erases the selected area without affecting the graph.

Once any points have been removed from the graph, the Submit button changes to a Reset button. This button undoes all previous removals and returns you to the original graph with all points included. Finally, clicking the Save button allows you to save the currently visible instances to a new ARFF file.

Part-A: Data Mining

1. Demonstration of preprocessing on dataset student.arff

Aim: This experiment illustrates some of the basic data preprocessing operations that can be performed using WEKA-Explorer. The sample dataset used for this example is the student data available in arff format.

Weka - Preprocessing the Data. The data that is collected from the field contains many unwanted things that leads to wrong analysis. For example, the data may contain null fields, it may contain columns that are irrelevant to the current analysis, and so on.

Step1: Loading the data. We can load the dataset into weka by clicking on open button in preprocessing interface and selecting the appropriate file.
Step2: Once the data is loaded, weka will recognize the attributes and during the scan of the data weka will compute some basic strategies on each attribute. The left panel in the above figure shows the list of recognized attributes while the top panel indicates the names of the base relation or table and the current working relation (which are same initially).
Step3: Clicking on an attribute in the left panel will show the basic statistics on the attributes for the categorical attributes the frequency of each attribute value is shown, while for continuous attributes we can obtain min, max, mean, standard deviation and deviation etc.,
Step4: The visualization in the right button panel in the form of cross-tabulation across two attributes.
Note:we can select another attribute using the dropdown list.
Step5:Selecting or filtering attributes Removing an attribute-When we need to remove an attribute,we can do this by using the attribute filters in weka.In the filter model panel,click on choose button,This will show a popup window with a list of available filters. Scroll down the list and select the “weka.filters.unsupervised.attribute.remove” filters.
Step 6:a)Next click the textbox immediately to the right of the choose button.In the resulting dialog box enter the index of the attribute to be filtered out.

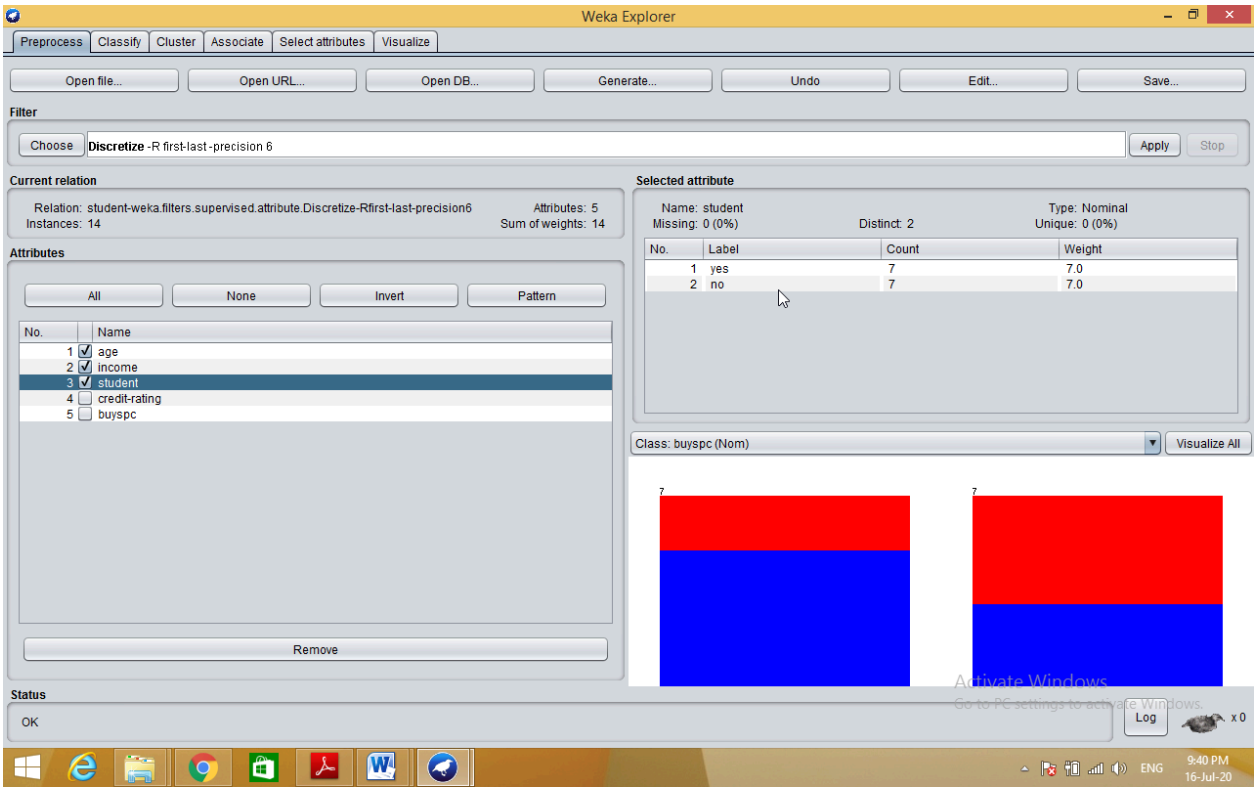
b)Make sure that invert selection option is set to false.The click OK now in the filter box.you will see “Remove-R-7”.

c)Click the apply button to apply filter to this data.This will remove the attribute and create new working relation.

d)Save the new working relation as an arff file by clicking save button on the top(button)panel.(student.arff)

Procedure:

Choose Tab→Weka→filters→Supervised→attribute→Discretize



Dataset student .arff

@relation student
@attribute age {<30,30-40,>40}
@attribute income {low, medium, high}
@attribute student {yes, no}
@attribute credit-rating {fair, excellent}
@attribute buyspc {yes, no}
@data
%
<30, high, no, fair, no
<30, high, no, excellent, no
30-40, high, no, fair, yes
>40, medium, no, fair, yes

>40, low, yes, fair, yes
>40, low, yes, excellent, no
30-40, low, yes, excellent, yes
<30, medium, no, fair, no
<30, low, yes, fair, no
>40, medium, yes, fair, yes
<30, medium, yes, excellent, yes
30-40, medium, no, excellent, yes
30-40, high, yes, fair, yes
>40, medium, no, excellent, no
%

2. Demonstration of Association rule process on dataset contactlenses.arff using apriori algorithm.

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
Step1: 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.
Step2: Clicking on the associate tab will bring up the interface for association rule algorithm.
Step3: We will use apriori algorithm. This is the default algorithm.
Step4: 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.

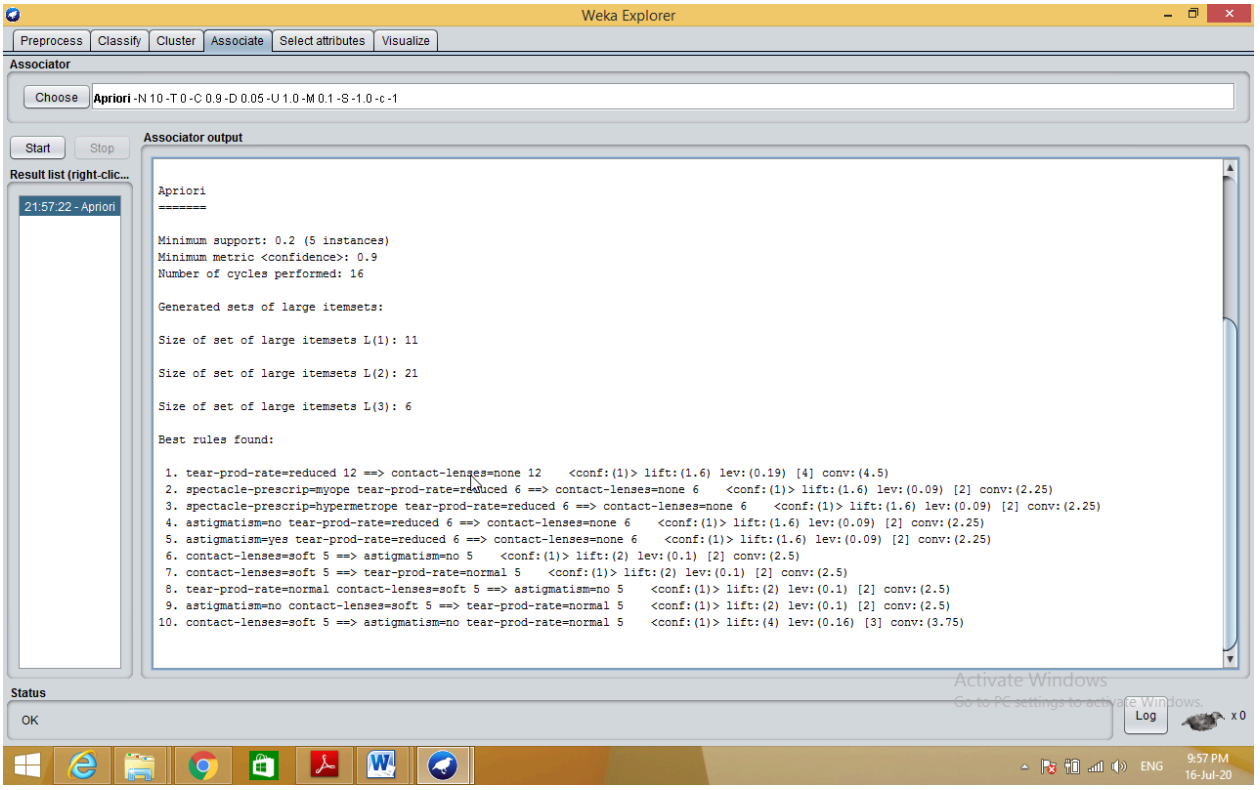
1) Join step

2) Prune step

Procedure:

Step 1: Load appropriate dataset into weka

Step 2: Select associate tab and select apriori algorithm


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Associator
Choose
Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 -c -1
Start
Stop
Result list (right-clic...
21:57:22 - Apriori
Associator output
Apriori
=======
Minimum support: 0.2 (5 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16
Generated sets of large itemsets:
Size of set of large itemsets L(1): 11
Size of set of large itemsets L(2): 21
Size of set of large itemsets L(3): 6
Best rules found:
1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12 <conf:(1)> lift:(1.6) lev:(0.19) [4] conv:(4.5)
2. spectacle-prescrip=myope tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
3. spectacle-prescrip=hypermetrope tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
4. astigmatism=no tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
5. astigmatism=yes tear-prod-rate=reduced 6 ==> contact-lenses=none 6 <conf:(1)> lift:(1.6) lev:(0.09) [2] conv:(2.25)
6. contact-lenses=soft 5 ==> astigmatism=no 5 <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
7. contact-lenses=soft 5 ==> tear-prod-rate=normal 5 <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
8. tear-prod-rate=normal contact-lenses=soft 5 ==> astigmatism=no 5 <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
9. astigmatism=no contact-lenses=soft 5 ==> tear-prod-rate=normal 5 <conf:(1)> lift:(2) lev:(0.1) [2] conv:(2.5)
10. contact-lenses=soft 5 ==> astigmatism=no tear-prod-rate=normal 5 <conf:(1)> lift:(4) lev:(0.16) [3] conv:(3.75)
Activate Windows
Go to PC settings to activate Windows.
Status
OK
Log
x 0
ENG
9:57 PM
16-Jul-20

3. Demonstration of classification rule process on dataset employee.arff using j48 algorithm.

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.
Step2: Next we select the "classify" tab and click "choose" button t o select the "j48"classifier.
Step3: 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.
Step4: 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:

$$entropy(Set) = I(Set) = -\sum_{i=1}^{k} P(value_i) \cdot \log_2(P(value_i))$$

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}

$$entropy\ (R) = I(R) = -\left[\underbrace{\left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right)}_{\text{a-values}} + \underbrace{\left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right)}_{\text{b-values}}\right]$$

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**.

Classifier Accuracy Measures

The ***confusion matrix*** is a useful tool for analyzing how well your classifier can recognize tuples of different classes.

		Predicted class	
		C_1	C_2
Actual class	C_1	true positives	false negatives
	C_2	false positives	true negatives

$$sensitivity = \frac{t\_pos}{pos}$$

$$specificity = \frac{t\_neg}{neg}$$

$$precision = \frac{t\_pos}{(t\_pos + f\_pos)}$$

$$accuracy = sensitivity\frac{pos}{(pos + neg)} + specificity\frac{neg}{(pos + neg)}.$$

F measure (F1 score or **F** score) is a **measure** of a test's accuracy and is defined as the weighted harmonic mean of the precision and recall of the test.

TP Rate: **rate** of true positives (instances correctly classified as a given class)

FP**Rate**: **rate** of false positives (instances falsely classified as a given class)

Precision is the fraction of retrieved data that are relevant to the query:

Recall is the fraction of the relevant data that are successfully retrieved.

Cross-validation: In ***k***-fold **cross-validation**, the initial data are randomly partitioned into *k* mutually exclusive subsets or "folds," *D*1, *D*2: *Dk*, each of approximately equal size.

Predictor Error Measures

$$\text{Absolute error}: \quad |y_i - y_i'|$$

$$\text{Squared error}: \quad (y_i - y_i')^2$$

$$\text{Mean absolute error}: \quad \frac{\sum_{i=1}^{d} |y_i - y_i'|}{d}$$

$$\text{Mean squared error}: \quad \frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d}$$

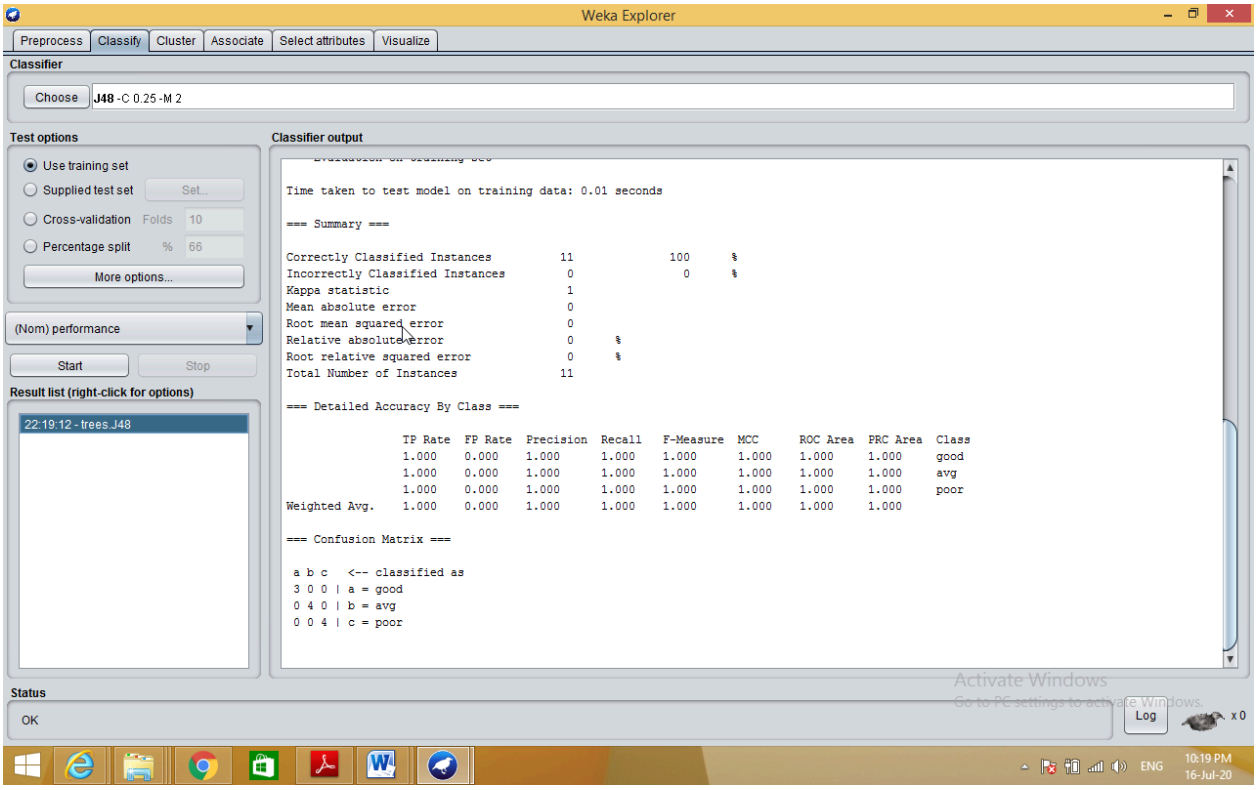
$$\text{Relative absolute error}: \quad \frac{\sum_{i=1}^{d} |y_i - y_i'|}{\sum_{i=1}^{d} |y_i - \bar{y}|}$$

$$\text{Relative squared error}: \quad \frac{\sum_{i=1}^{d} (y_i - y_i')^2}{\sum_{i=1}^{d} (y_i - \bar{y})^2}$$

Procedure:

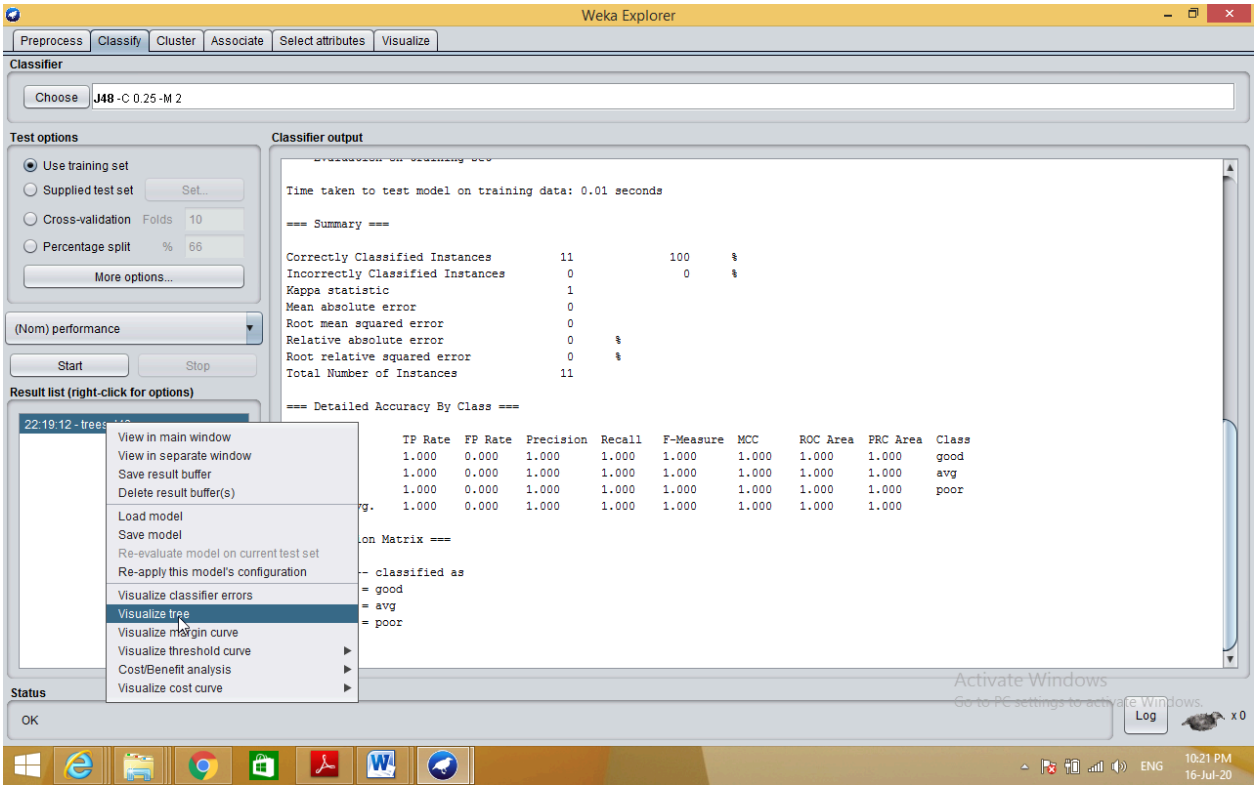
Step 1: Load appropriate dataset into WEKA.

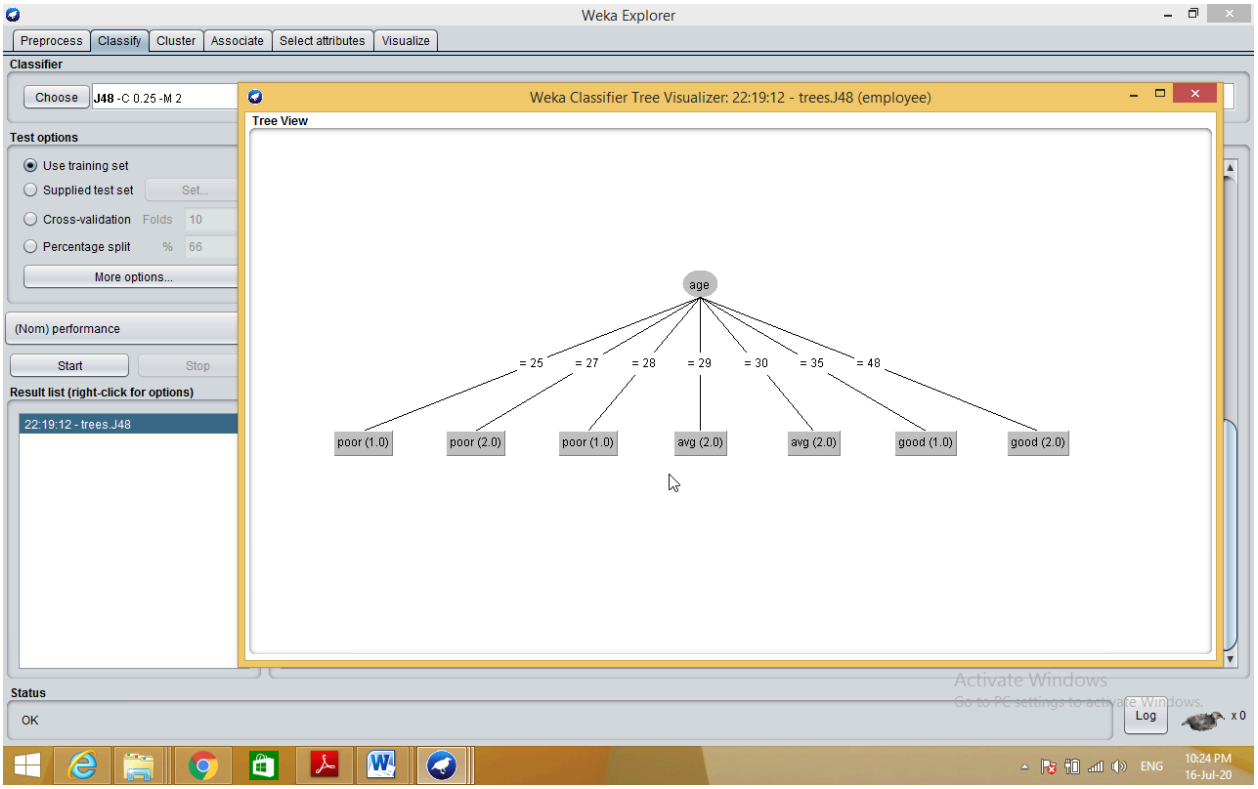
Step 2: Select classify TAB and select J48 algorithm.



Generating Decision Tree:-

Right click on result list□select visualize tree option


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
Classifier output
Time taken to test model on training data: 0.01 seconds
=== Summary ===
Correctly Classified Instances 11 100 %
Incorrectly Classified Instances 0 0 %
Kappa statistic 1
Mean absolute error 0
Root mean squared error 0
Relative absolute error 0 %
Root relative squared error 0 %
Total Number of Instances 11
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 good
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 avg
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 poor
(Nom) performance
Start
Stop
Result list (right-click for options)
22:19:12 - trees
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize classifier errors
Visualize tree
Visualize margin curve
Visualize threshold curve
Cost/Benefit analysis
Visualize cost curve
Status
OK
Log
x 0
Activate Windows
10:21 PM
16-Jul-20


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Weka Classifier Tree Visualizer: 22:19:12 - trees.J48 (employee)
Tree View
age
= 25
= 27
= 28
= 29
= 30
= 35
= 48
poor (1.0)
poor (2.0)
poor (1.0)
avg (2.0)
avg (2.0)
good (1.0)
good (2.0)
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) performance
Start
Stop
Result list (right-click for options)
22:19:12 - trees.J48
Status
OK
Log
x 0
Activate Windows
10:24 PM
16-Jul-20

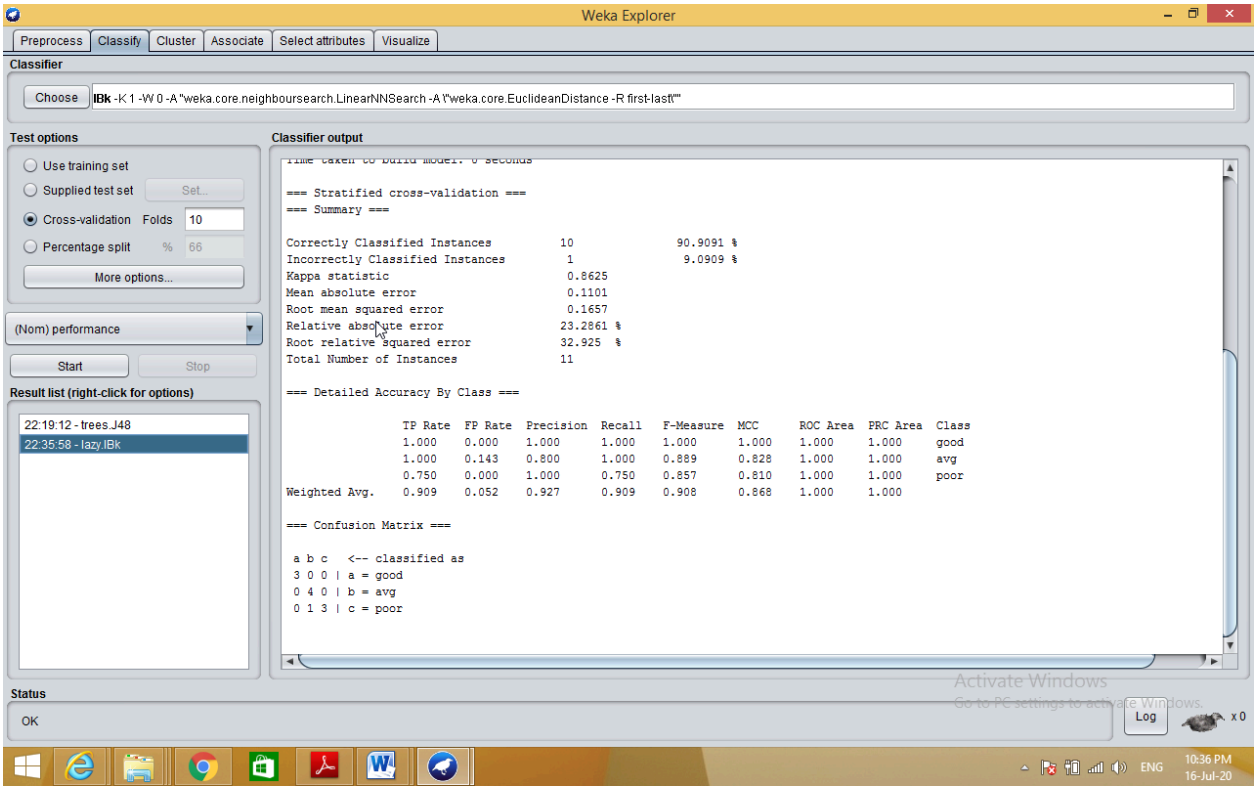
4. Demonstration of classification rule process on dataset employee.arff using id3 algorithm.

Aim: This experiment illustrates the use of j-48 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.
Step2: Next we select the “classify” tab and click “choose” button to select the “j48”classifier.
Step3: 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.
Step4: 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.

Data set employee.arff:

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



5. Demonstration of classification rule process on dataset employee.arff using naïve bayes algorithm.

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:
1. We begin the experiment by loading the data (employee.arff) into weka.
Step2: next we select the "classify" tab and click "choose" button to select the "id3"classifier.
Step3: 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.
Step4: 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 will show pop-up window which will allow you to open the file containing test instances.

Description:
Bayesian classification is based on Baye's Theorem. Bayesian classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Baye's Theorem
Bayes' Theorem is named after Thomas Bayes. There are two types of probabilities

- Posterior Probability [P(H/X)]
- Prior Probability [P(H)]

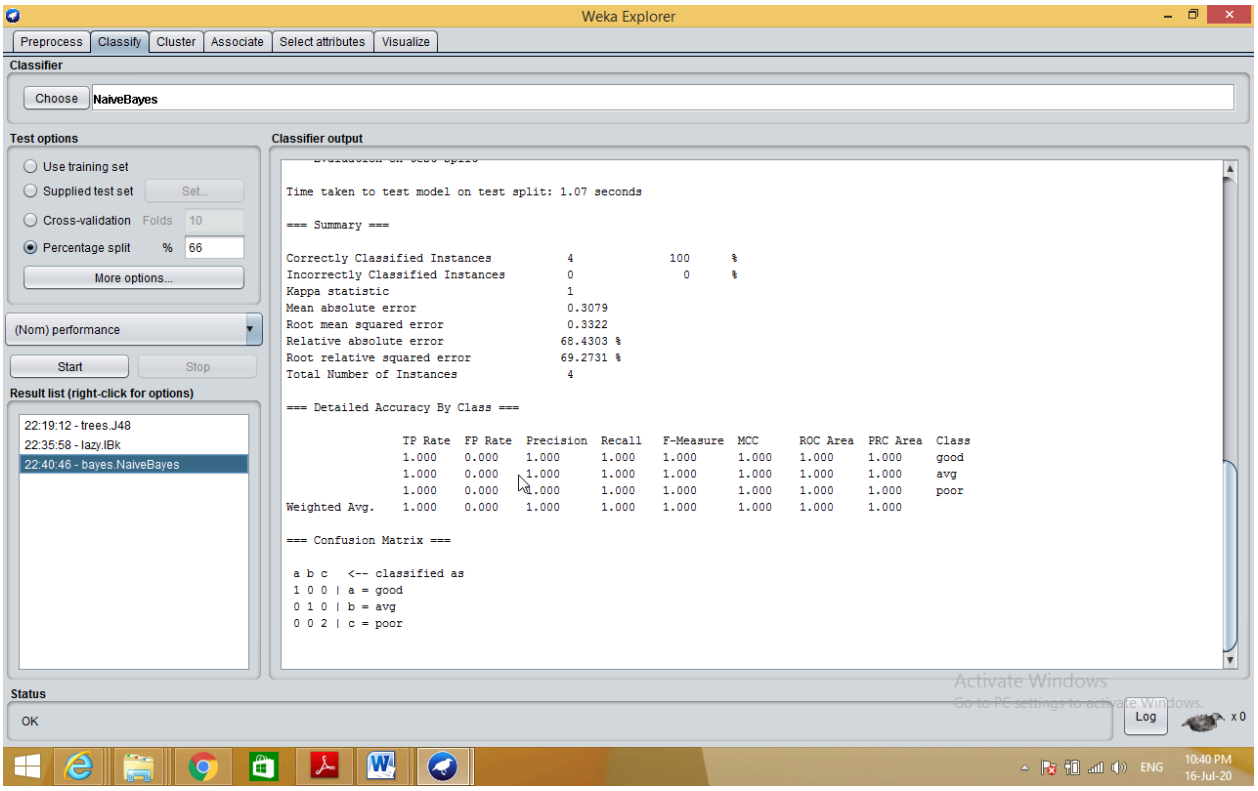
where X is data tuple and H is some hypothesis.
According to Bayes' Theorem,

P(H/X)= P(X/H)P(H) / P(X)

Procedure:

Step 1: Load appropriate dataset into WEKA

Step 2: Go to Classify tab→Bayes→NaiveBayes



6. Demonstration of clustering rule process on dataset iris.arff using simple k-means.

Aim: This experiment illustrates the use of simple k-mean clustering with Weka explorer. The sample data set used for this example is based on the iris data available in ARFF format. This document assumes that appropriate preprocessing has been performed. This iris dataset includes 150 instances.

Steps involved in this Experiment

Step 1: Run the Weka explorer and load the data file iris.arff in preprocessing interface.

Step 2: Inorder to perform clustering select the 'cluster' tab in the explorer and click on the choose button. This step results in a dropdown list of available clustering algorithms.

Step 3 : In this case we select 'simple k-means'.

Step 4: Next click in text button to the right of the choose button to get popup window shown in the screenshots. In this window we enter six on the number of clusters and we leave the value of the seed on as it is. The seed value is used in generating a random number which is used for making the internal assignments of instances of clusters.

Step 5 : Once of the option have been specified. We run the clustering algorithm there we must make sure that they are in the 'cluster mode' panel. The use of training set option is selected and then we click 'start' button. This process and resulting window are shown in the following screenshots.

Step 6 : The result window shows the centroid of each cluster as well as statistics on the number and the percent of instances assigned to different clusters. Here clusters centroid are means vectors for each clusters. This clusters can be used to characterized the cluster.For eg, the centroid of cluster1 shows the class iris.versicolor mean value of the sepal length is 5.4706, sepal width 2.4765, petal width 1.1294, petal length 3.7941.

Step 7: Another way of understanding characterstics of each cluster through visualization ,we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments.

Description:

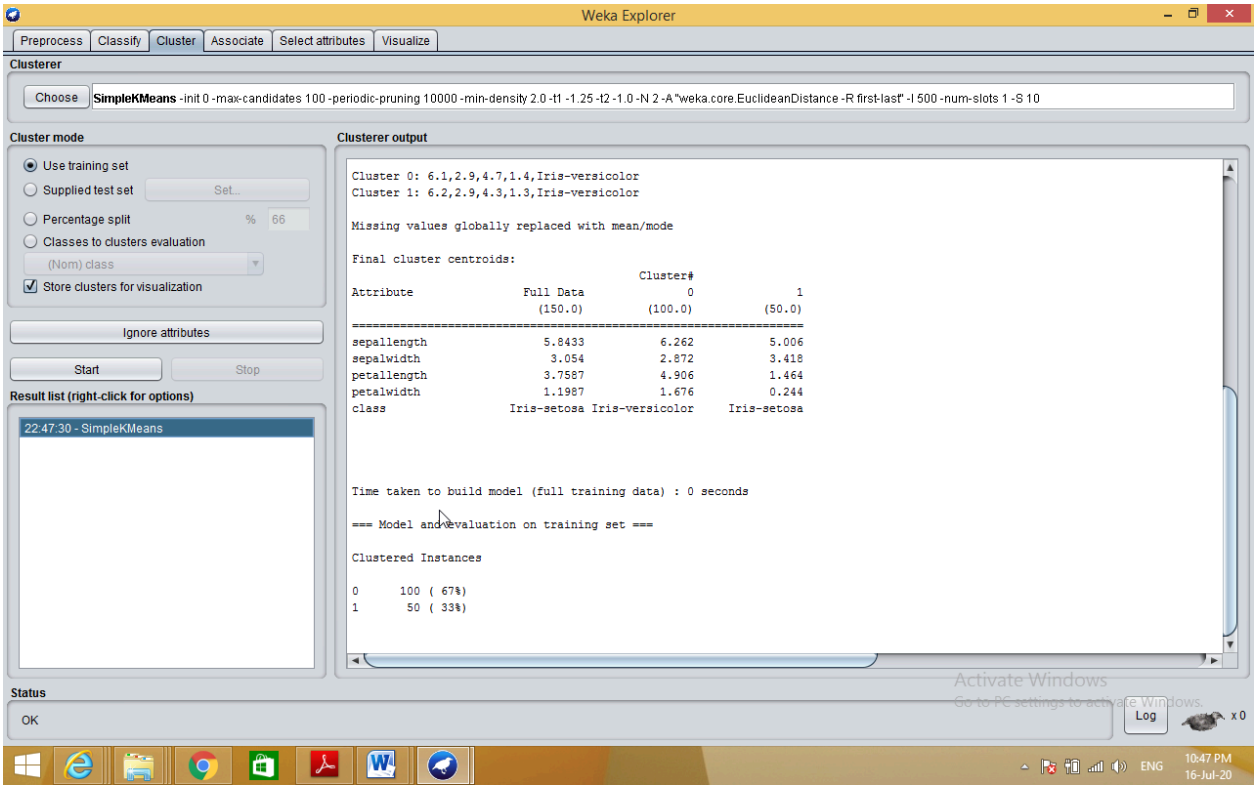
Clustering is a process of partitioning a set of **data** (or objects) into a set of meaningful sub-classes, called **clusters**.

K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean.

Procedure:

Step 1: Load appropriate dataset into WEKA

Step 2: Go to Cluster tab□choose□SimpleKmeans


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 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
22:47:30 - SimpleKMeans
Clusterer output
Cluster 0: 6.1,2.9,4.7,1.4,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%)
Activate Windows
Go to PC settings to activate Windows.
Status
OK
Log
x 0
ENG
10:47 PM
16-Jul-20

7. Demonstration of clustering rule process on dataset student.arff using hierarchical clustering.

Aim: This experiment illustrates the use of hierarchical clustering with Weka explorer. The sample data set used for this example is based on the student data available in ARFF format. This document assumes that appropriate preprocessing has been performed. This istudent dataset includes 14 instances.

Steps involved in this Experiment

Step 1: Run the Weka explorer and load the data file student.arff in preprocessing interface.

Step 2: Inorder to perform clustering select the 'cluster' tab in the explorer and click on the choose button. This step results in a dropdown list of available clustering algorithms.

Step 3 : In this case we select 'hierarchical'.

Step 4: Next click in text button to the right of the choose button to get popup window shown in the screenshots. In this window we enter six on the number of clusters and we leave the value of the seed on as it is. The seed value is used in generating a random number which is used for making the internal assignments of instances of clusters.

Step 5 : Once of the option have been specified. We run the clustering algorithm there we must make sure that they are in the 'cluster mode' panel. The use of training set option is selected and then we click 'start' button. This process and resulting window are shown in the following screenshots.

Step 6 : The result window shows the centroid of each cluster as well as statistics on the number and the percent of instances assigned to different clusters. Here clusters centroid are means vectors for each clusters. This clusters can be used to characterized the cluster.

Step 7: Another way of understanding characteristics of each cluster through visualization ,we can do this, try right clicking the result set on the result. List panel and selecting the visualize cluster assignments. Interpretation of the above visualization From the above visualization, we can understand the distribution of age and instance number in each cluster. For instance, for each cluster is dominated by age. In this case by changing the color dimension to other attributes we can see their distribution with in each of the cluster.

Step 8: We can assure that resulting dataset which included each instance along with its assign cluster. To do so we click the save button in the visualization window and save the result student hierarchical .The top portion of this file is shown in the following figure.

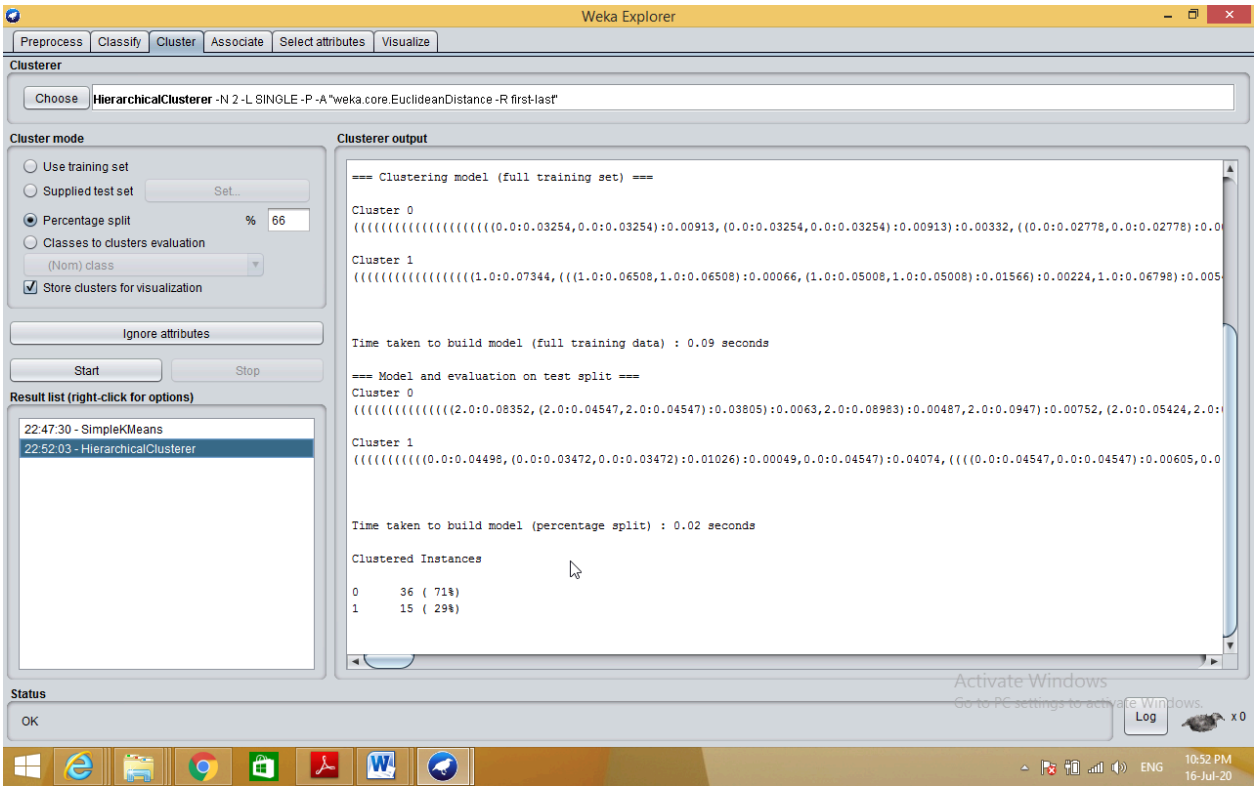
Description:

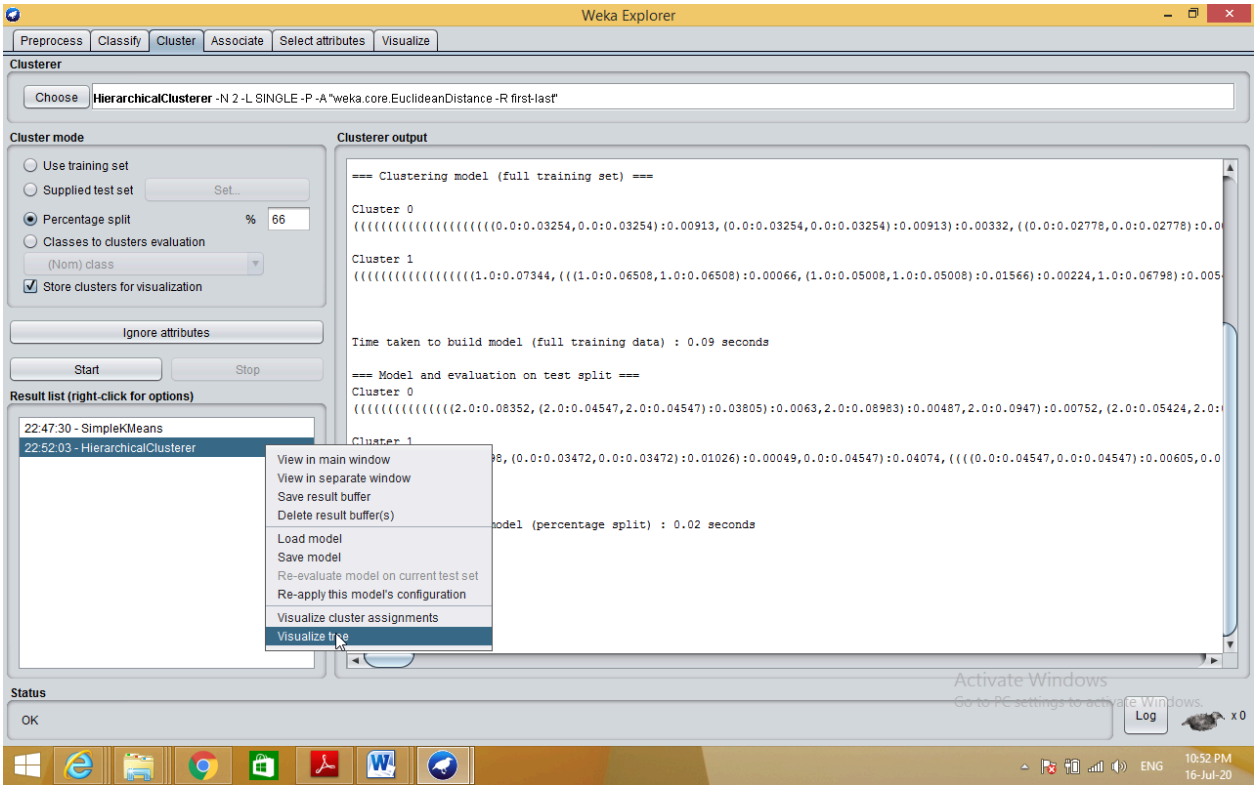
Hierarchical clustering involves creating **clusters** that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a **hierarchy**. There are two types of **hierarchical clustering**, Divisive and Agglomerative.

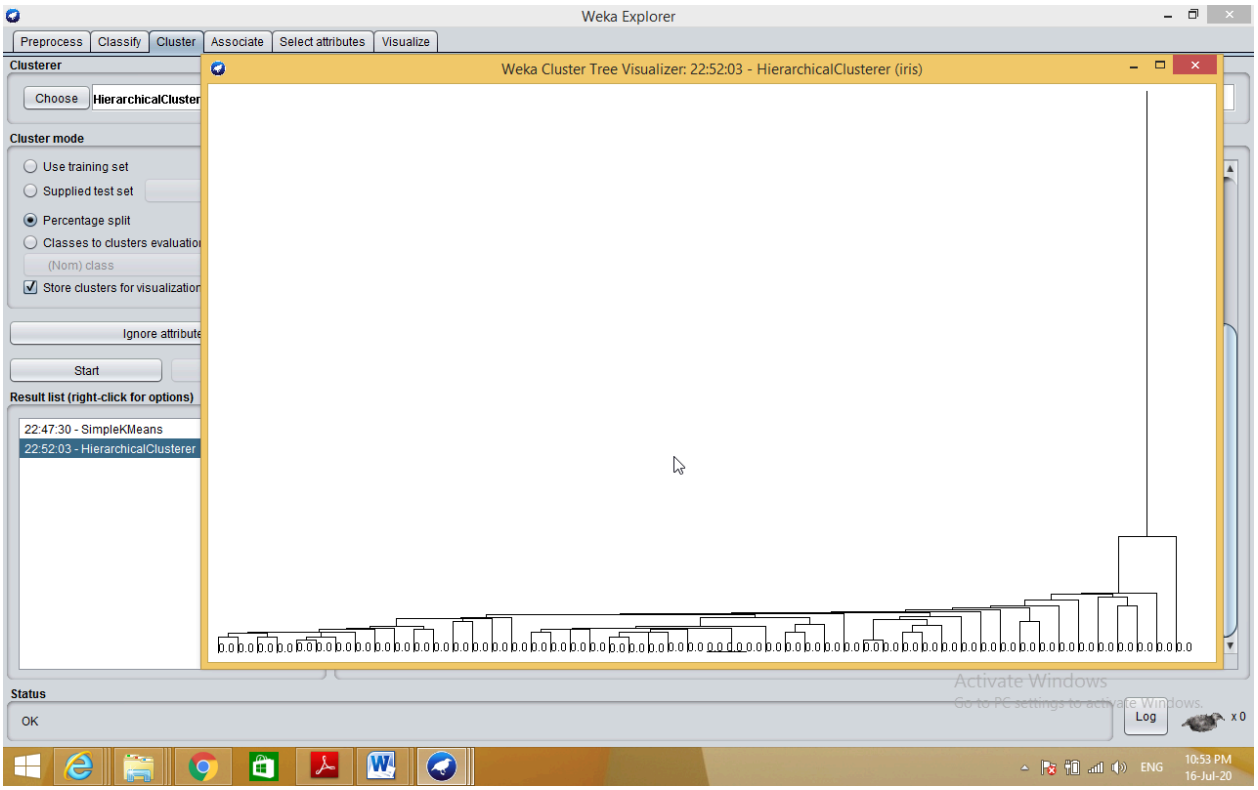
Procedure:

Step 1: Load appropriate dataset into WEKA

Step 2: Go to Cluster tab choose Hierarchical clustering




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
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
22:47:30 - SimpleKMeans
22:52:03 - HierarchicalClusterer
Clusterer output
=== Clustering model (full training set) ===
Cluster 0
Cluster 1
Time taken to build model (full training data) : 0.09 seconds
=== Model and evaluation on test split ===
Cluster 0
Cluster 1
odel (percentage split) : 0.02 seconds
View in main window
View in separate window
Save result buffer
Delete result buffer(s)
Load model
Save model
Re-evaluate model on current test set
Re-apply this model's configuration
Visualize cluster assignments
Visualize tree
Status
OK
Log
x 0
Activate Windows
Go to PC settings to activate Windows.
ENG
10:52 PM
16-Jul-20


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Weka Cluster Tree Visualizer: 22:52:03 - HierarchicalClusterer (iris)
Clusterer
Choose
Cluster mode
Use training set
Supplied test set
Percentage split
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Result list (right-click for options)
22:47:30 - SimpleKMeans
22:52:03 - HierarchicalClusterer
Status
OK
Log
x 0
Activate Windows
Go to PC settings to activate Windows.
ENG
10:53 PM
16-Jul-20

Part-B: Data Analytics
1. a) Write R program to find R-Mean, Median & Mode with the sample data.
Source code:

```
install.packages("datasets")
library(datasets)

airquality <- datasets::airquality
summary(airquality)

summary(airquality)

var(airquality$Solar.R,na.rm = T)

summary(airquality$Temp)

summary(airquality$Wind)

plot(airquality$Wind)
plot(airquality$Temp,airquality$Wind)
plot(airquality)

plot(airquality$Wind, type= "b") # p: points, l: lines,b: both
plot(airquality$Wind, xlab = 'ozone Concentration',
    ylab = 'No of Instances', main = 'Ozone levels in NY city',
    col = 'blue')
```

```
# Horizontal bar plot
barplot(airquality$Ozone, main = 'Ozone Concenteration in air',
        ylab = 'ozone levels', col= 'blue',horiz = F)

#Histogram
hist(airquality$Temp)
hist(airquality$Temp,
     main = 'Solar Radiation values in air',
     xlab = 'Solar rad.', col='blue')

#Single box plot
boxplot(airquality$Temp,main="Boxplot")

# Multiple box plots
boxplot(airquality[,1:4],main='Multiple')
```

Output:

```
summary(airquality)
     Ozone           Solar.R          Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
 NA's   :37       NA's   :7
     Month            Day
 Min.   :5.000   Min.   : 1.0
 1st Qu.:6.000   1st Qu.: 8.0
 Median :7.000   Median :16.0
 Mean   :6.993   Mean   :15.8
 3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :9.000   Max.   :31.0

var(airquality$Solar.R,na.rm = T)
[1] 8110.519

summary(airquality$Temp)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  56.00   72.00   79.00   77.88   85.00   97.00
```

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

Source code:

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


# Get the dataset.
input <- mtcars

# Create the regression model.
result <- aov(mpg~hp*am,data = input)
print(summary(result))


# Create the regression model.
result <- aov(mpg~hp+am,data = input)
print(summary(result))


# Create the regression models.
result1 <- aov(mpg~hp*am,data = input)
result2 <- aov(mpg~hp+am,data = input)

# Compare the two models.
print(anova(result1,result2))
```

Output:

```
result <- aov(mpg~hp*am,data = input)
> print(summary(result))
            Df Sum Sq Mean Sq F value  Pr(>F)
hp           1 678.4  678.4 77.391 1.50e-09 ***
am           1 202.2  202.2 23.072 4.75e-05 ***
hp:am        1  0.0    0.0   0.001   0.981
Residuals   28 245.4   8.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

result <- aov(mpg~hp+am,data = input)
> print(summary(result))
            Df Sum Sq Mean Sq F value  Pr(>F)
hp           1 678.4  678.4  80.15 7.63e-10 ***
am           1 202.2  202.2  23.89 3.46e-05 ***
```

```
Residuals  29 245.4   8.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

result1 <- aov(mpg~hp*am,data = input)
> result2 <- aov(mpg~hp+am,data = input)
> # Compare the two models.
> print(anova(result1,result2))
Analysis of Variance Table

Model 1: mpg ~ hp * am
Model 2: mpg ~ hp + am
  Res.Df   RSS Df  Sum of Sq    F Pr(>F)
1     28 245.43
2     29 245.44 -1 -0.0052515 6e-04 0.9806


>
```

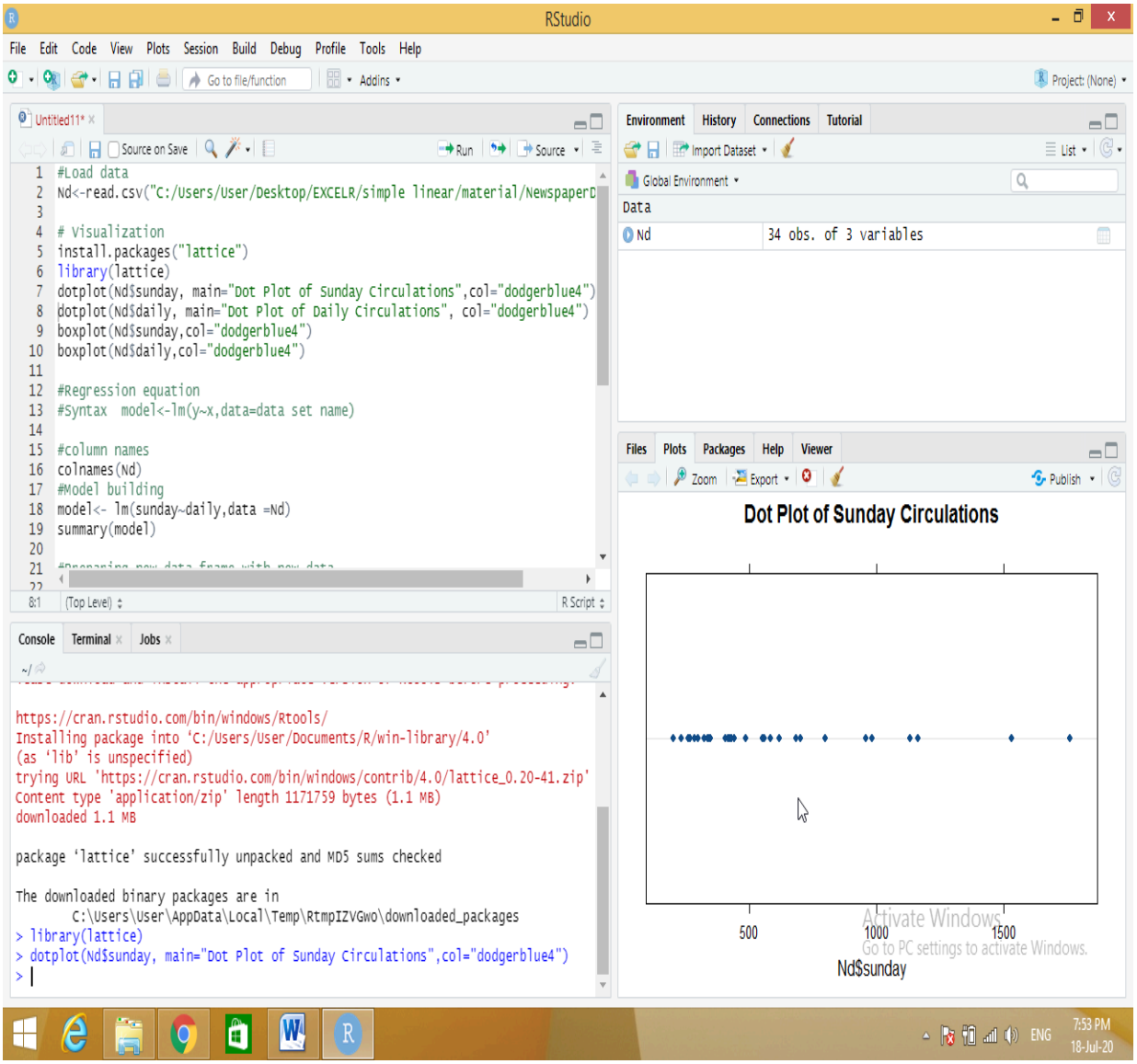
2. Write R program to find the following Regressions with the sample data and visualize the regressions graphically.
Source code:

a) Linear Regression

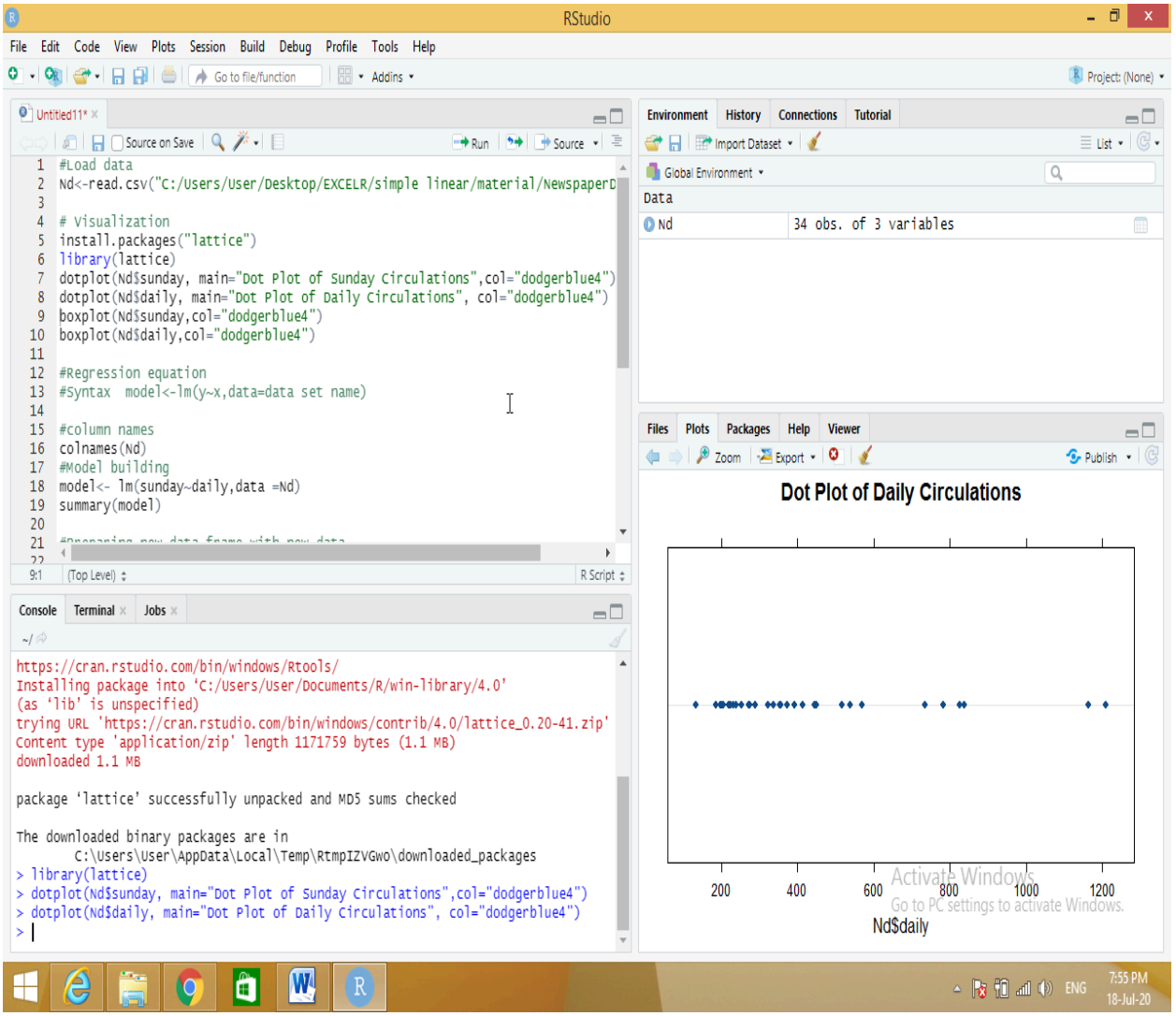
```
#Load data
Nd<-read.csv("C:/Users/User/Desktop/EXCELR/simple
linear/material/NewspaperData.csv")

# Visualization
install.packages("lattice")
library(lattice)
dotplot(Nd$sunday, main="Dot Plot of Sunday Circulations",col="dodgerblue4")
dotplot(Nd$daily, main="Dot Plot of Daily Circulations", col="dodgerblue4")
boxplot(Nd$sunday,col="dodgerblue4")
boxplot(Nd$daily,col="dodgerblue4")

#Regression equation
#Syntax  model<-lm(y~x,data=data set name)

#column names
colnames(Nd)
#Model building
model<- lm(sunday~daily,data =Nd)
summary(model)

#Preparing new data frame with new data
new_daily=data.frame(daily=c(300,200))

#Predict for the new data
sun1=predict(model,new_daily)

#Print predicted value
sun1

#Predict for daily variable from the historical data
pred<-predict(model)
#Print predicted values
pred
#Prepare a new data frame with pred and error
finaldata<-data.frame(Nd,pred,"Error"= Nd$sunday-pred)
```

Output:

dotplot(Nd$sunday, main="Dot Plot of Sunday Circulations",col="dodgerblue4")



dotplot(Nd$daily, main="Dot Plot of Daily Circulations", col="dodgerblue4")



b) Multiple Regression

Source code:

mileage<-read.csv("C:/Users/User/Desktop/EXCELR/multi linear/Cars.csv")

#mileage<-Cars
#Scatter Plot Matrix:

```
pairs(mileage)

#Correlation Matrix:###
cor(mileage)
#Regression Model and Summary
model.car<-lm(MPG~HP+VOL+SP+WT,data = mileage)
summary(model.car)


#########Experiment#####################
reg_vol<-lm(MPG~VOL,data = mileage)
summary(reg_vol)
reg_wt<-lm(MPG~WT,data = mileage)
summary(reg_wt)
reg_wt_vol<-lm(MPG~WT+VOL,data = mileage)
summary(reg_wt_vol)
###################

####stepAIC to find out variable that removes multicollinearity
#Regression Model and Summary
model.car<-lm(MPG~.,data = mileage)
summary(model.car)
#Multi-colinearity
install.packages("car")
library(car)
#Variance Inflation Factor - Multi collinearity values
car::vif(model.car)
##Subset selection
install.packages("MASS")
library(MASS)
stepAIC(model.car)
############################


#################################
#Full Model Building process

#mileage<-Cars
#Scatter Plot Matrix:
pairs(mileage)

#Correlation Matrix:###
```

```
cor(mileage)

#Regression Model and Summary
model.car<-lm(MPG~.,data = mileage)

#Multi-colinearity
install.packages("car")
library(car)
#Variance Inflation Factor
car::vif(model.car)
#Diagnostic Plots:
  #Residual Plots, QQ-Plos, Std. Residuals vs Fitted
plot(model.car)
#Residuals vs Regressors

residualPlots(model.car)
#Added Variable Plots
avPlots(model.car)
#QQ plots of studentized residuals
qqPlot(model.car)
#Deletion Diagnostics
influenceIndexPlot(model.car) # Index Plots of the influence measures

####Iteration 1
#Remove 77th observation
mileage["HP2"] <-mileage$HP*mileage$HP
mileage["SP2"] <-mileage$SP*mileage$SP
mileage1<-mileage[-77,]


model1<-lm(MPG~.,data = mileage1)
summary(model1)

plot(model1)
residualPlots(model1)
qqPlot(model1)
influenceIndexPlot(model1)

#iteration2
mileage2<-mileage[-c(77,66,81,79),]

model2<-lm(MPG~.,data = mileage2[,-c(5,4)])
```

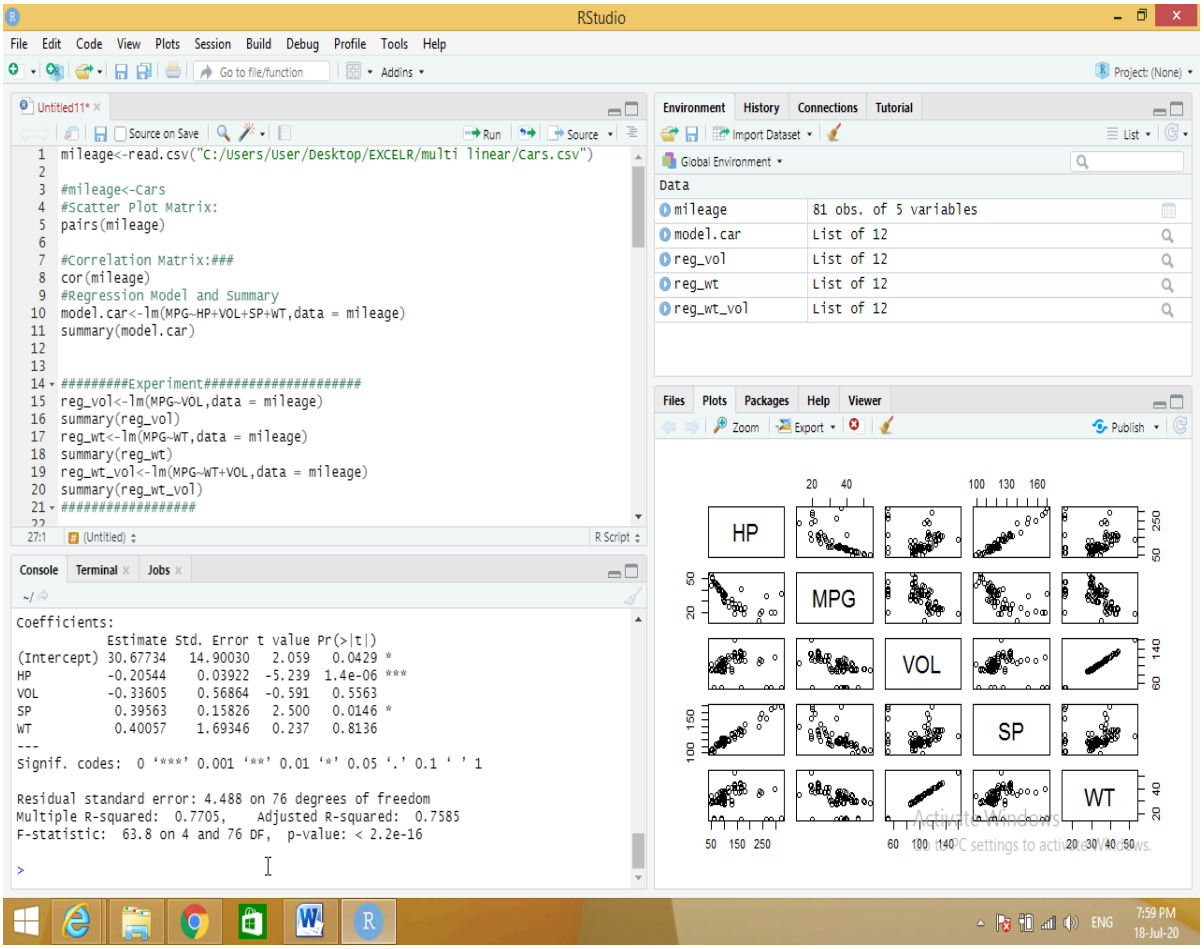
summary(model2)
influenceIndexPlot(model2)

output:

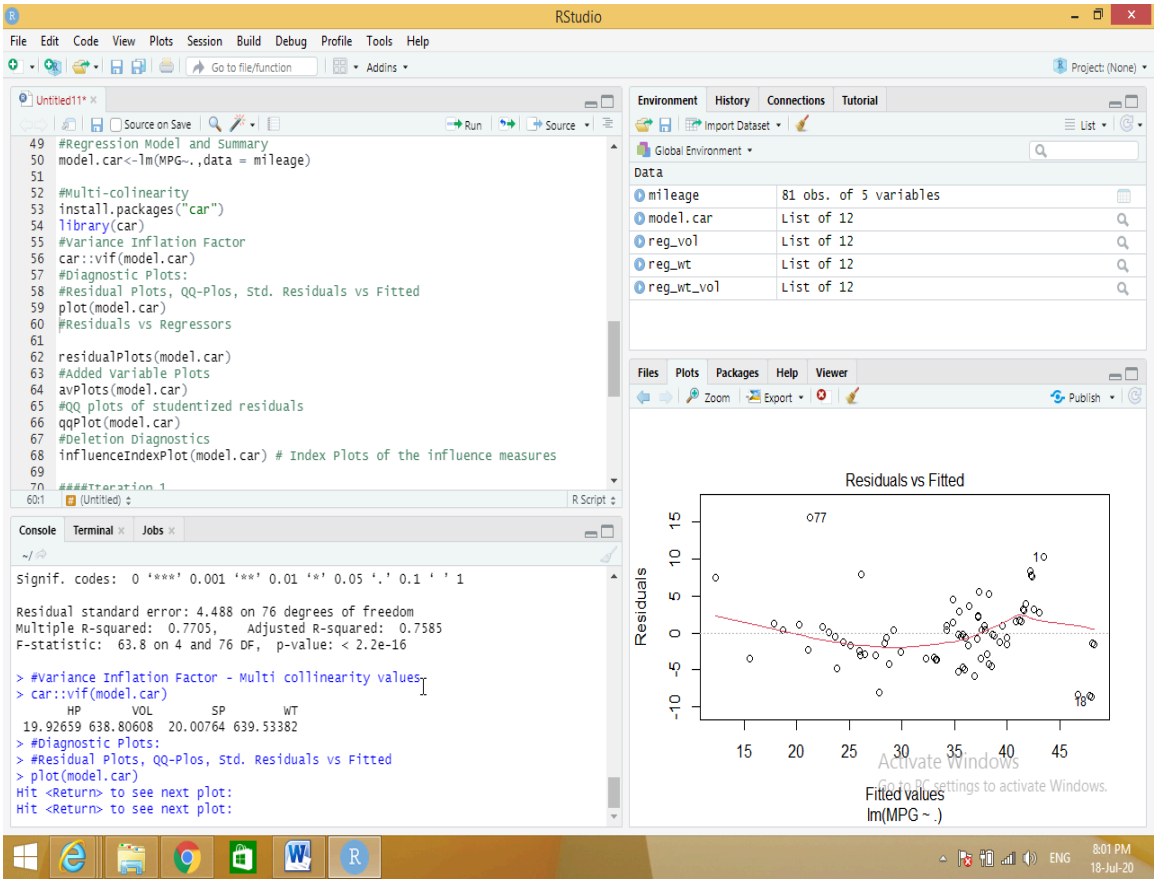
```
cor(mileage)
       HP    MPG     VOL     SP      WT
HP  1.00000000 -0.7250383 0.07745947 0.9738481
0.07651307
MPG -0.72503835 1.0000000 -0.52905658 -0.6871246
-0.52675909
VOL 0.07745947 -0.5290566 1.00000000 0.1021700
0.99920308
SP  0.97384807 -0.6871246 0.10217001 1.0000000
0.10243919
WT  0.07651307 -0.5267591 0.99920308 0.1024392
1.00000000
> #Regression Model and Summary
> model.car<-lm(MPG~HP+VOL+SP+WT,data = mileage)
> summary(model.car)

Call:
lm(formula = MPG ~ HP + VOL + SP + WT, data = mileage)

Residuals:
  Min   1Q Median   3Q   Max
-8.6320 -2.9944 -0.3705 2.2149 15.6179

Coefficients:
      Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.67734  14.90030  2.059  0.0429 *
HP     -0.20544  0.03922 -5.239 1.4e-06 ***
VOL    -0.33605  0.56864 -0.591  0.5563
SP      0.39563  0.15826  2.500  0.0146 *
WT      0.40057  1.69346  0.237  0.8136
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.488 on 76 degrees of freedom
```

```
Multiple R-squared: 0.7705,    Adjusted R-squared: 0.7585
F-statistic: 63.8 on 4 and 76 DF,  p-value: < 2.2e-16

> ##########Experiment#######################
> reg_vol<-lm(MPG~VOL,data = mileage)
> summary(reg_vol)

Call:
lm(formula = MPG ~ VOL, data = mileage)

Residuals:
    Min      1Q  Median      3Q     Max
-25.3074  -5.2026   0.1902   5.4536  17.1632

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 55.81709    3.95696  14.106  < 2e-16 ***
VOL         -0.21662    0.03909  -5.541 3.82e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.798 on 79 degrees of freedom
Multiple R-squared: 0.2799,    Adjusted R-squared: 0.2708
F-statistic: 30.71 on 1 and 79 DF,  p-value: 3.823e-07

> reg_wt<-lm(MPG~WT,data = mileage)
> summary(reg_wt)

Call:
lm(formula = MPG ~ WT, data = mileage)

Residuals:
    Min      1Q  Median      3Q     Max
-25.3933  -5.4377   0.2738   5.2951  16.9351

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 55.2296    3.8761  14.249  < 2e-16 ***
WT          -0.6420    0.1165  -5.508 4.38e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.811 on 79 degrees of freedom
```

```
Multiple R-squared: 0.2775,    Adjusted R-squared: 0.2683
F-statistic: 30.34 on 1 and 79 DF,  p-value: 4.383e-07

> reg_wt_vol<-lm(MPG~WT+VOL,data = mileage)
> ###stepAIC to find out variable that removes
multicollinearity
> #Regression Model and Summary
> model.car<-lm(MPG~.,data = mileage)
> summary(model.car)

Call:
lm(formula = MPG ~ ., data = mileage)

Residuals:
    Min      1Q  Median      3Q     Max
-8.6320 -2.9944 -0.3705  2.2149 15.6179

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.67734   14.90030   2.059   0.0429 *
HP          -0.20544    0.03922  -5.239 1.4e-06 ***
VOL         -0.33605    0.56864  -0.591   0.5563
SP           0.39563    0.15826   2.500   0.0146 *
WT           0.40057    1.69346   0.237   0.8136
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.488 on 76 degrees of freedom
Multiple R-squared:  0.7705,    Adjusted R-squared:  0.7585
F-statistic:  63.8 on 4 and 76 DF,  p-value: < 2.2e-16
```


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled11*
1 mileage<-read.csv("C:/Users/User/Desktop/EXCELR/multi linear/Cars.csv")
2
3 #mileage<-Cars
4 #Scatter Plot Matrix:
5 pairs(mileage)
6
7 #Correlation Matrix:###
8 cor(mileage)
9 #Regression Model and Summary
10 model.car<-lm(MPG~HP+VOL+SP+WT,data = mileage)
11 summary(model.car)
12
13
14 ########Experiment####################
15 reg_vol<-lm(MPG~VOL,data = mileage)
16 summary(reg_vol)
17 reg_wt<-lm(MPG~WT,data = mileage)
18 summary(reg_wt)
19 reg_wt_vol<-lm(MPG~WT+VOL,data = mileage)
20 summary(reg_wt_vol)
21 ##################
Console Terminal Jobs
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.67734 14.90030 2.059 0.0429 *
HP -0.20544 0.03922 -5.239 1.4e-06 ***
VOL -0.33605 0.56864 -0.591 0.5563
SP 0.39563 0.15826 2.500 0.0146 *
WT 0.40057 1.69346 0.237 0.8136

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 4.488 on 76 degrees of freedom
Multiple R-squared: 0.7705, Adjusted R-squared: 0.7585
F-statistic: 63.8 on 4 and 76 DF, p-value: < 2.2e-16
Environment History Connections Tutorial
Import Dataset
Global Environment
Data
mileage 81 obs. of 5 variables
model.car List of 12
reg_vol List of 12
reg_wt List of 12
reg_wt_vol List of 12
Files Plots Packages Help Viewer
Zoom Export Publish
HP
MPG
VOL
SP
WT
ENG 7:59 PM 18-Jul-20



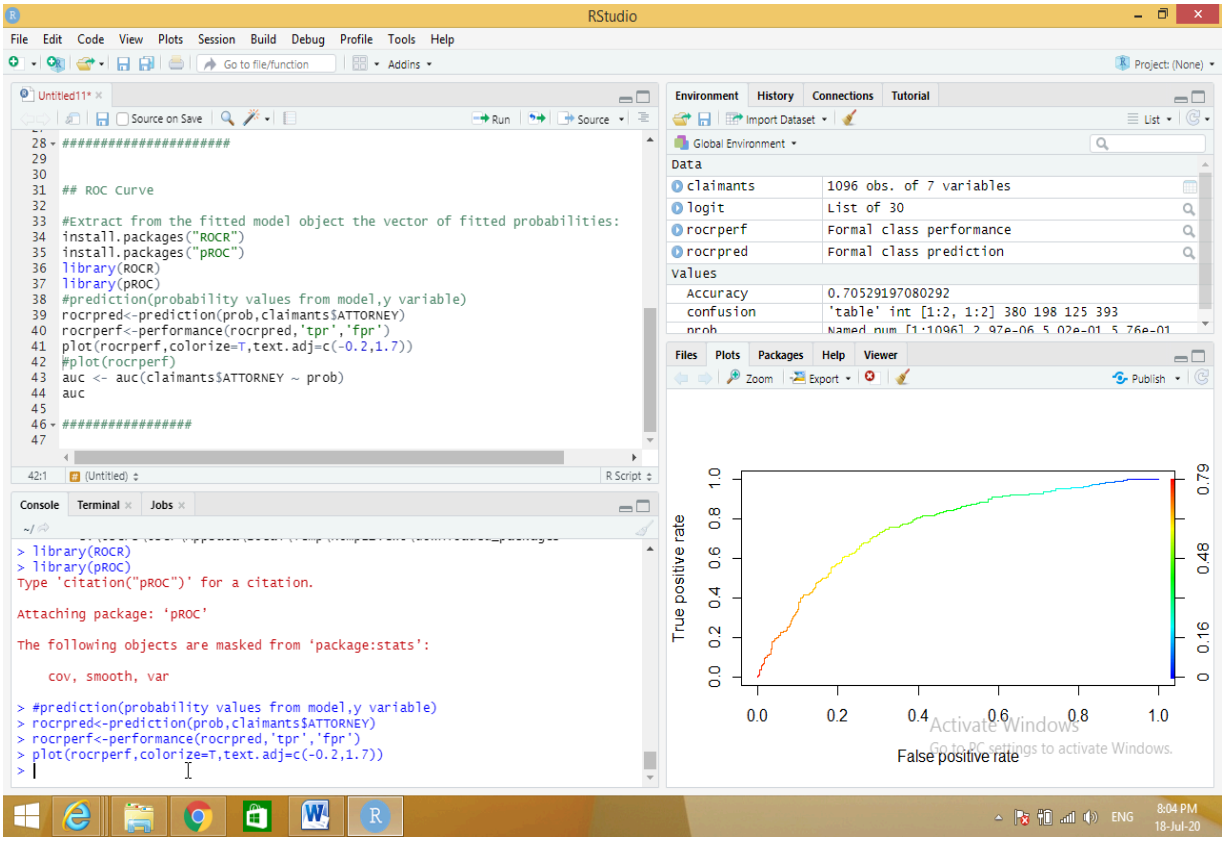
c) Logistic Regression

Source code:

```
claimants<-read.csv("C:/Users/User/Desktop/EXCELR/logistic regression/claimants.csv")
#Finding null values
sum(is.na(claimants))
#Removing null values- na.omit(dataset)
claimants <- na.omit(claimants)
# Logistic Regression
#glm(y~x,family="bin....)
logit<-glm(ATTORNEY ~ factor(CLMSEX) + factor(CLMINSUR) + factor(SEATBELT)
        + CLMAGE + LOSS,family= "binomial",data=claimants)
summary(logit)

# Confusion Matrix Table
#predict(modelobject,testdataset)
prob=predict(logit,type=c("response"),claimants)
prob
```

```
#table(dataframe1,dataframe2) ..to create 2X2 matrix
confusion<-table(prob>0.5,claimants$ATTORNEY)
confusion

# Model Accuracy
#adding diagonal elements in the confusion matrix
Accuracy<-sum(diag(confusion))/sum(confusion)
Accuracy

###############

#######################


## ROC Curve

#Extract from the fitted model object the vector of fitted probabilities:
install.packages("ROCR")
install.packages("pROC")
library(ROCR)
library(pROC)
#prediction(probability values from model,y variable)
rocrpred<-prediction(prob,claimants$ATTORNEY)
rocrperf<-performance(rocrpred,'tpr','fpr')
plot(rocrperf,colorize=T,text.adj=c(-0.2,1.7))
#plot(rocrperf)
auc <- auc(claimants$ATTORNEY ~ prob)
auc

#################
```

Output:



d) Poisson Regression.

Source code:

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

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

Output:

```
input <- warpbreaks
> print(head(input))
  breaks wool tension
1     26    A       L
2     30    A       L
```

```
3  54  A  L
4  25  A  L
5  70  A  L
6  52  A  L
> output <-glm(formula = breaks ~ wool+tension, data =
warpbreaks,
+         family = poisson)
> print(summary(output))

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

Deviance Residuals:
   Min     1Q  Median     3Q    Max
-3.6871 -1.6503 -0.4269  1.1902  4.2616

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 3.69196    0.04541 81.302  < 2e-16 ***
woolB       -0.20599    0.05157 -3.994 6.49e-05 ***
tensionM    -0.32132    0.06027 -5.332 9.73e-08 ***
tensionH    -0.51849    0.06396 -8.107 5.21e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 297.37  on 53  degrees of freedom
Residual deviance: 210.39  on 50  degrees of freedom
AIC: 493.06

Number of Fisher Scoring iterations: 4
```

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

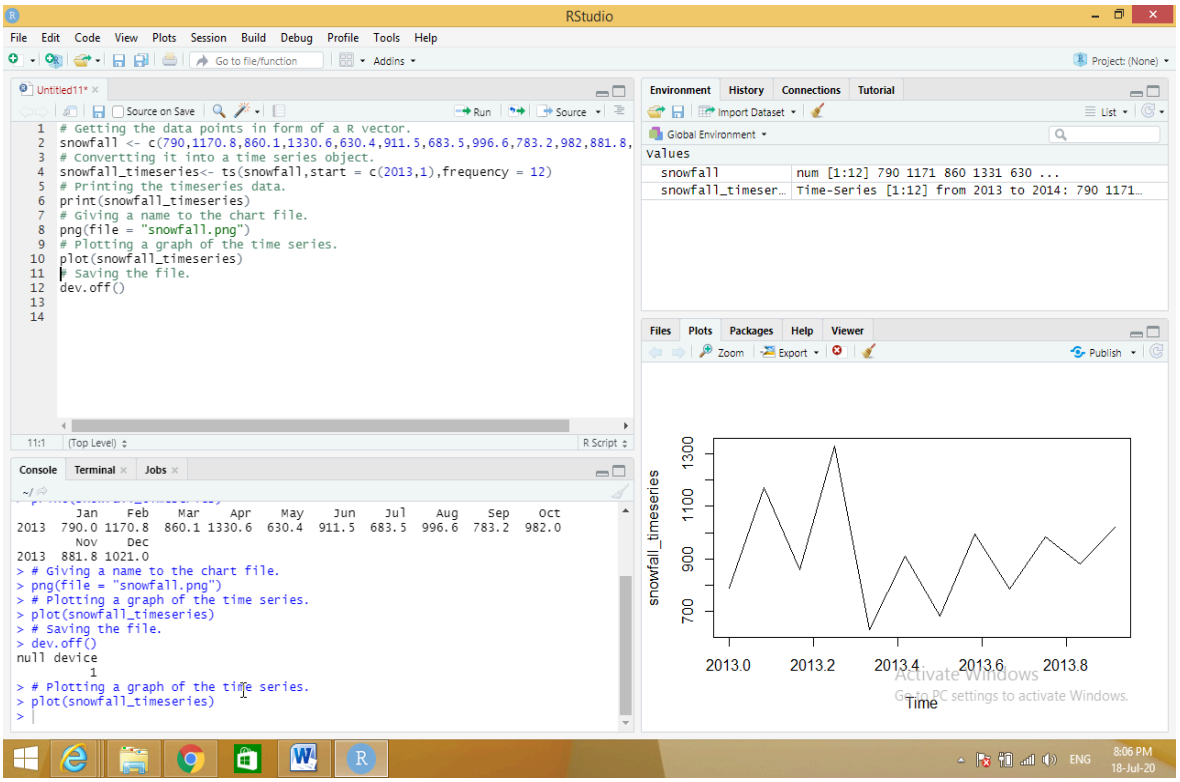
Source code:

```
# Getting the data points in form of a R vector.
snowfall <- c(790,1170.8,860.1,1330.6,630.4,911.5,683.5,996.6,783.2,982,881.8,1021)
# Convertting it into a time series object.
snowfall_timeseries<- ts(snowfall,start = c(2013,1),frequency = 12)
# Printing the timeseries data.
print(snowfall_timeseries)
# Giving a name to the chart file.
png(file = "snowfall.png")
# Plotting a graph of the time series.
plot(snowfall_timeseries)
# Saving the file.
dev.off()
```

Output:

```
snowfall <-
c(790,1170.8,860.1,1330.6,630.4,911.5,683.5,996.6,783.2,982
,881.8,1021)
> # Convertting it into a time series object.
> snowfall_timeseries<- ts(snowfall,start =
c(2013,1),frequency = 12)
> # Printing the timeseries data.
```

```
> print(snowfall_timeseries)
     Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct
2013 790.0 1170.8 860.1 1330.6 630.4 911.5 683.5 996.6 783.2
982.0
     Nov  Dec
2013 881.8 1021.0
> # Giving a name to the chart file.
> png(file = "snowfall.png")
> # Plotting a graph of the time series.
> plot(snowfall_timeseries)
> # Saving the file.
> dev.off()
null device
          1
> # Plotting a graph of the time series.
> plot(snowfall_timeseries)
```

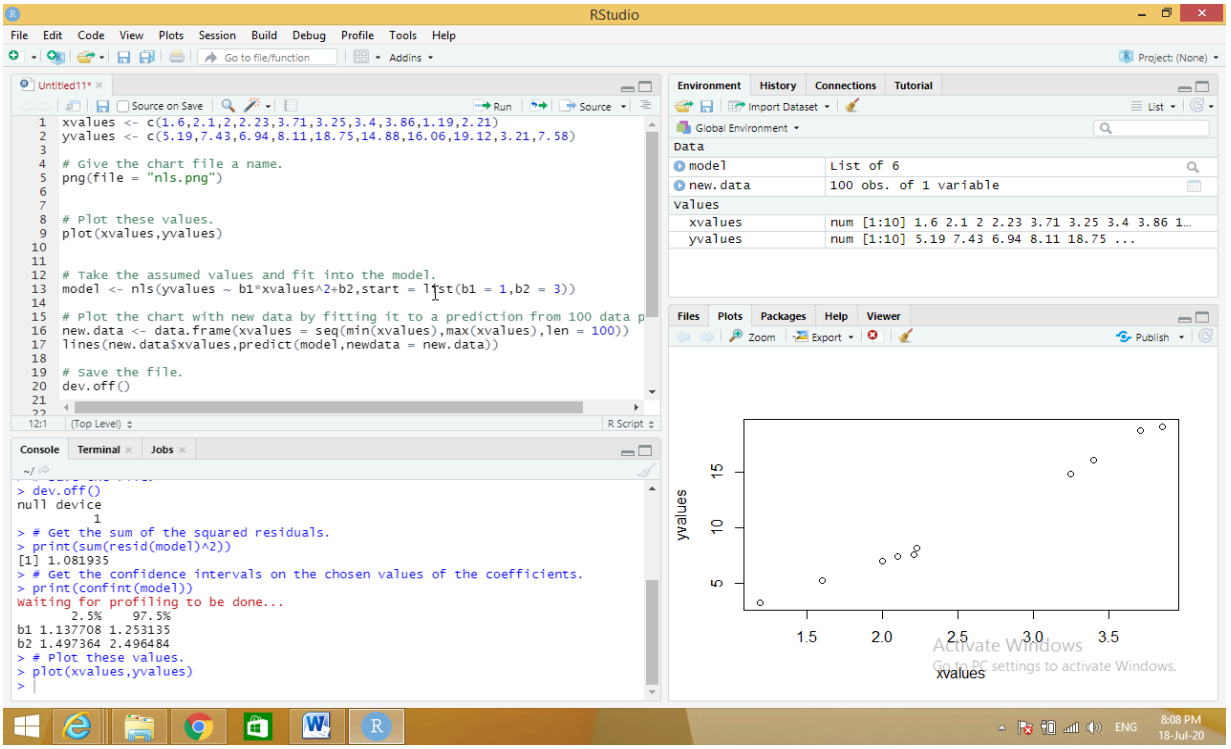


b) Write R program to find Non Linear Least Square with the sample data and visualize the regression graphically.

Source code:

```
xvalues <- c(1.6,2.1,2,2.23,3.71,3.25,3.4,3.86,1.19,2.21)
yvalues <- c(5.19,7.43,6.94,8.11,18.75,14.88,16.06,19.12,3.21,7.58)

# Give the chart file a name.
png(file = "nls.png")


# Plot these values.
plot(xvalues,yvalues)


# Take the assumed values and fit into the model.
model <- nls(yvalues ~ b1*xvalues^2+b2,start = list(b1 = 1,b2 = 3))

# Plot the chart with new data by fitting it to a prediction from 100 data points.
new.data <- data.frame(xvalues = seq(min(xvalues),max(xvalues),len = 100))
lines(new.data$xvalues,predict(model,newdata = new.data))

# Save the file.
dev.off()

# Get the sum of the squared residuals.
print(sum(resid(model)^2))

# Get the confidence intervals on the chosen values of the coefficients.
print(confint(model))
```

Output:



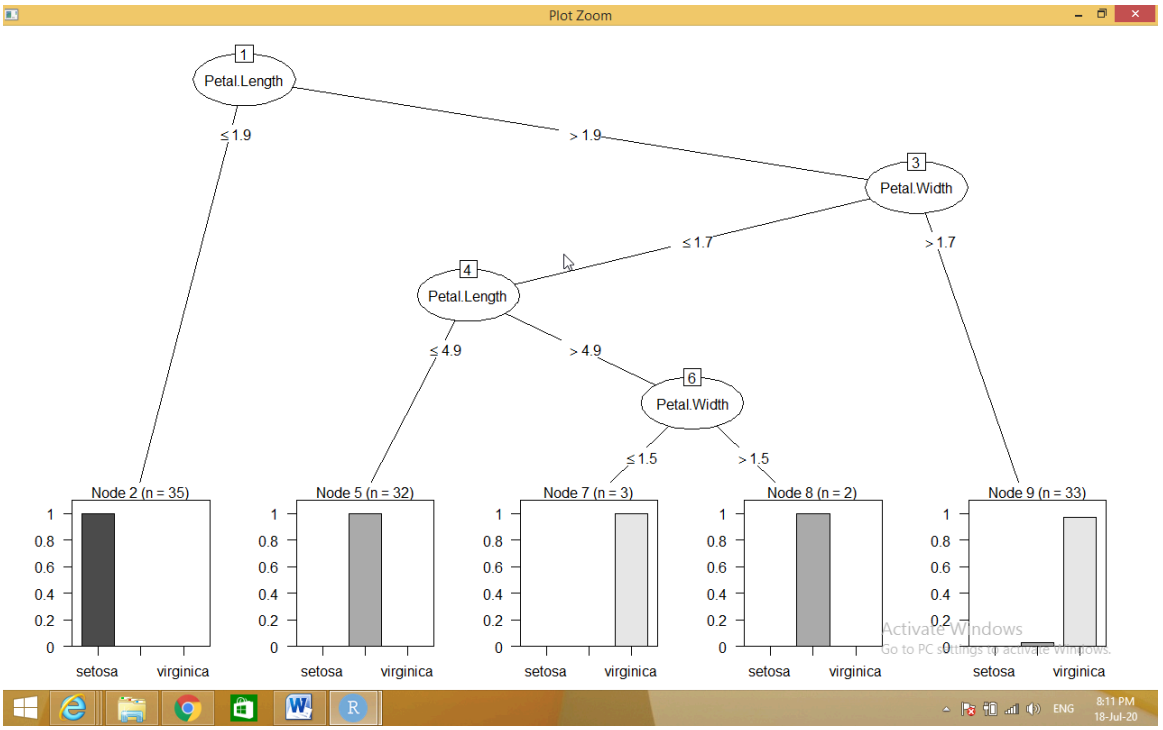
c) Write R program to find Decision Tree with the sample data and visualize the regression graphically.

Source code:

```
#Data Load
data("iris")
#Install the required packages
install.packages("caret")
install.packages("C50")
#Library invoke
library(caret)
library(C50)
#To make the results consistent across the runs

set.seed(7)
#Data Partition
inTraininglocal<-createDataPartition(iris$Species,p=.70,list = F)
training<-iris[inTraininglocal,]
testing<-iris[-inTraininglocal,]

#Model Building
model<-C5.0(Species~.,data = training)
#Generate the model summary
summary(model)
#Predict for test data set
```

```
pred<-predict.C5.0(model,testing[,-5]) #type ="prob"
#Accuracy of the algorithm
a<-table(testing$Species,pred)
sum(diag(a))/sum(a)
#Visualize the decision tree
plot(model)
```

OUTPUT:



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

Source code:

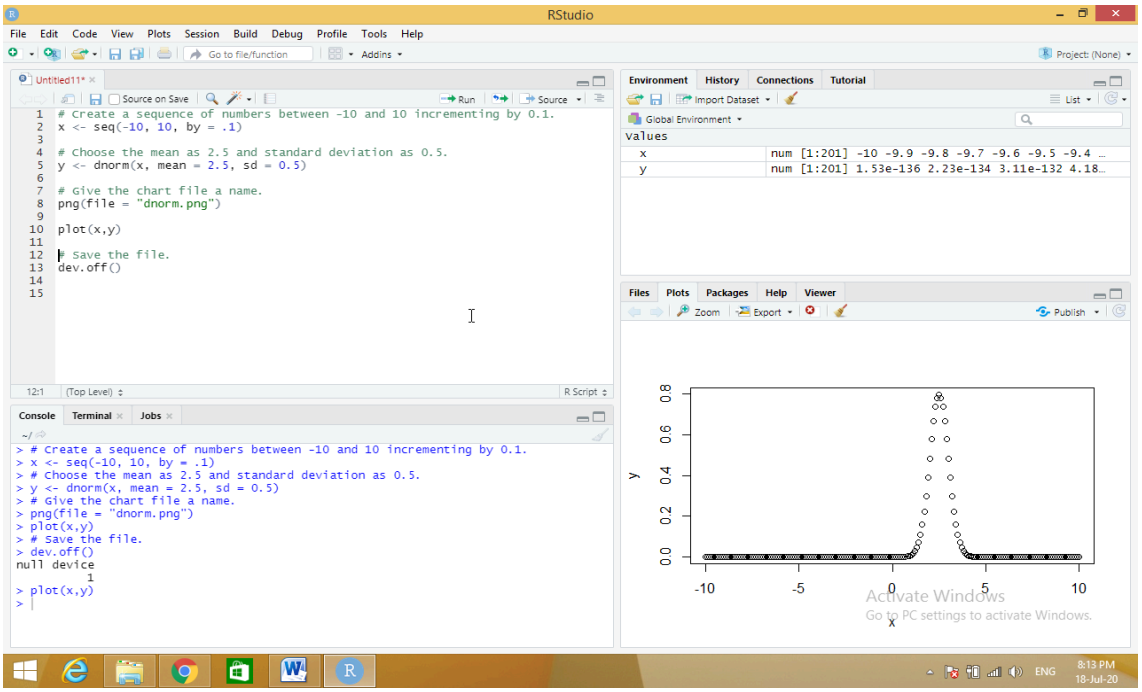
a) Normal Distribution

dnorm

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.1.
x <- seq(-10, 10, by = .1)

# Choose the mean as 2.5 and standard deviation as 0.5.
y <- dnorm(x, mean = 2.5, sd = 0.5)
```

```
# Give the chart file a name.
png(file = "dnorm.png")

plot(x,y)

# Save the file.
dev.off()
```

Output:

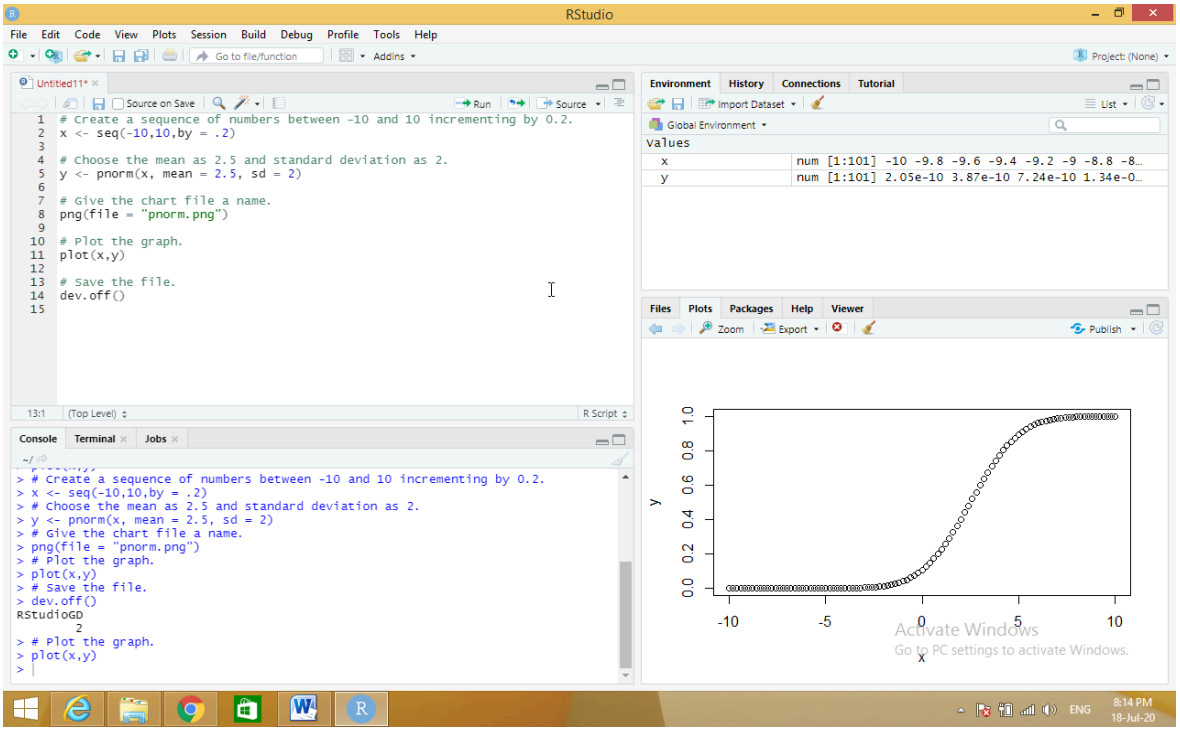


Pnorm

```
# Create a sequence of numbers between -10 and 10 incrementing by 0.2.
x <- seq(-10,10,by = .2)

# Choose the mean as 2.5 and standard deviation as 2.
y <- pnorm(x, mean = 2.5, sd = 2)

# Give the chart file a name.
png(file = "pnorm.png")
```

```
# Plot the graph.
plot(x,y)

# Save the file.
dev.off()
```

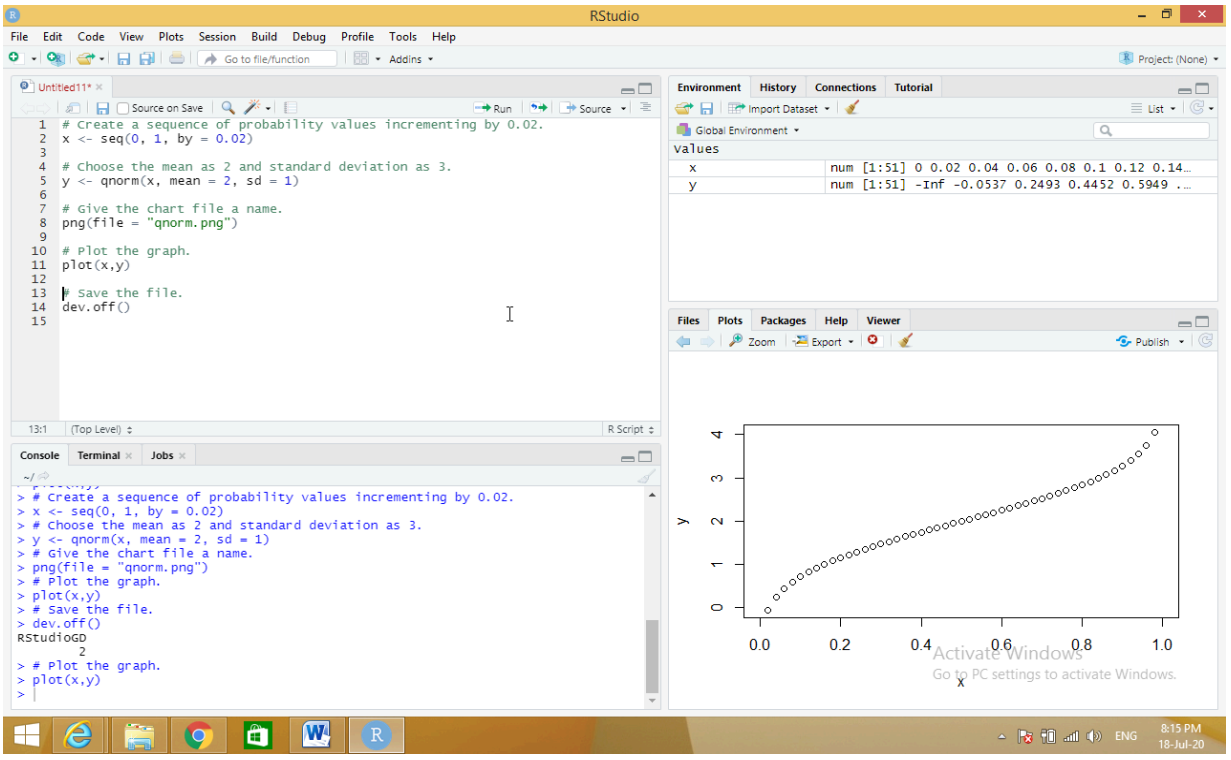
Output:



qnorm

```
# Create a sequence of probability values incrementing by 0.02.
x <- seq(0, 1, by = 0.02)

# Choose the mean as 2 and standard deviation as 3.
y <- qnorm(x, mean = 2, sd = 1)

# Give the chart file a name.
png(file = "qnorm.png")
```

```
# Plot the graph.
plot(x,y)

# Save the file.
dev.off()
```

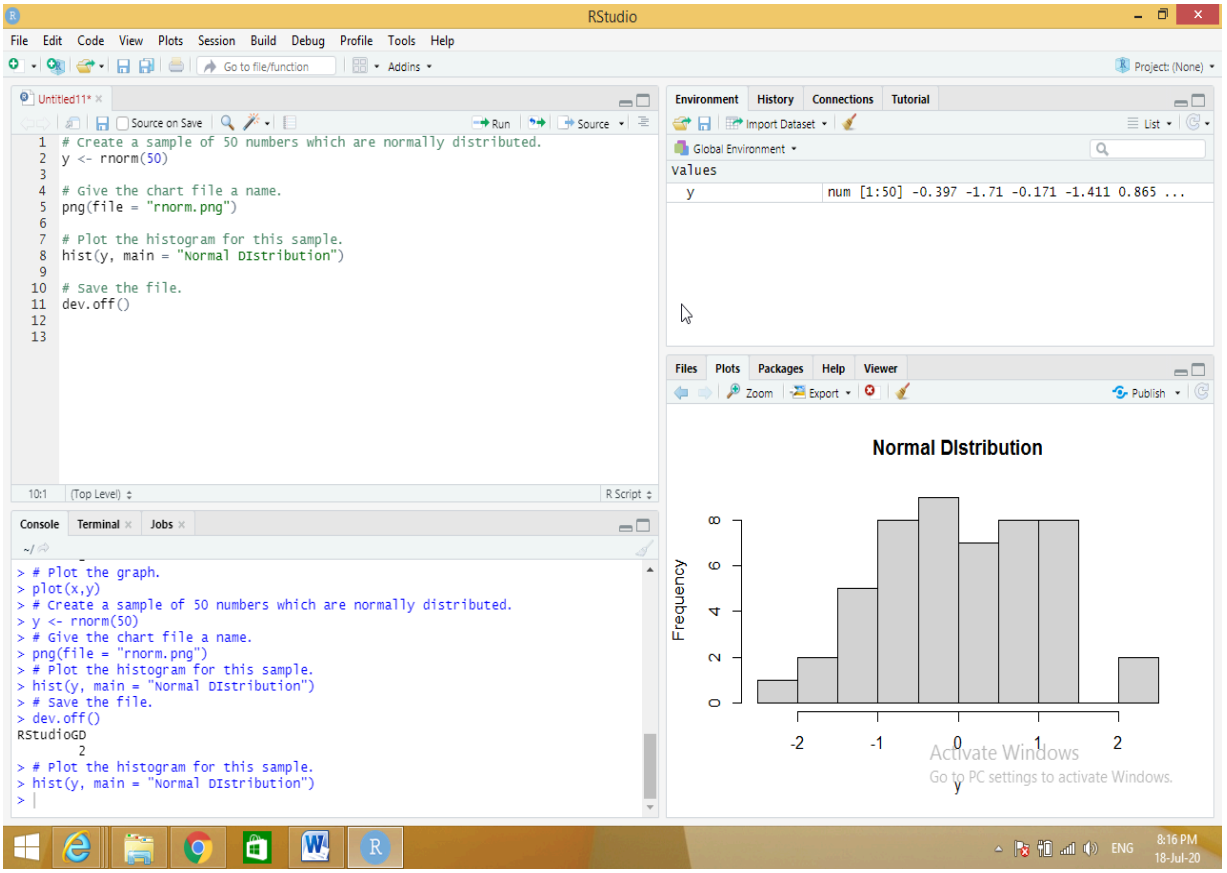


rnorm

```
# Create a sample of 50 numbers which are normally distributed.
y <- rnorm(50)

# Give the chart file a name.
png(file = "rnorm.png")

# Plot the histogram for this sample.
hist(y, main = "Normal DIstribution")
```

```
# Save the file.
dev.off()
```

Output:



b) Binomial Distribution

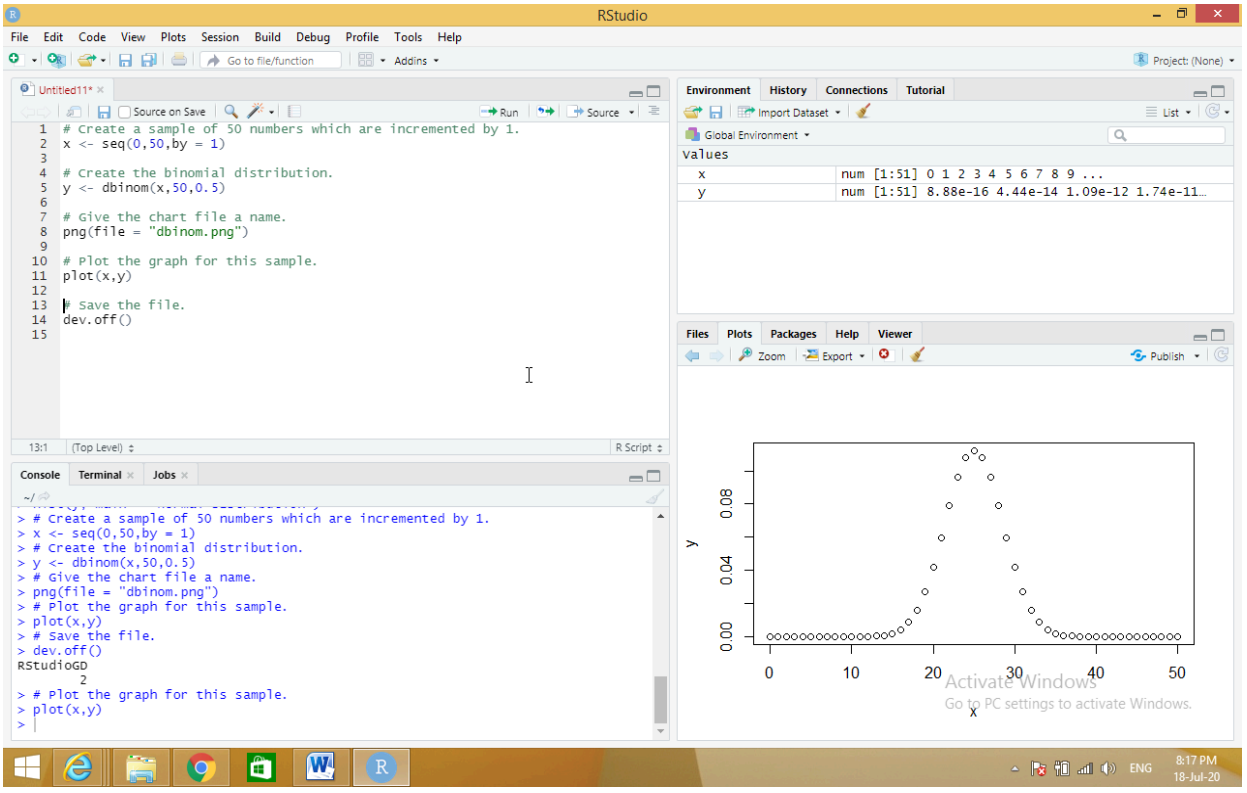
dbinom

Source code:

```
# Create a sample of 50 numbers which are incremented by 1.
x <- seq(0,50,by = 1)

# Create the binomial distribution.
```

```
y <- dbinom(x,50,0.5)

# Give the chart file a name.
png(file = "dbinom.png")

# Plot the graph for this sample.
plot(x,y)

# Save the file.
dev.off()
```

Outputs:



Pbinom

```
# Probability of getting 26 or less heads from a 51 tosses of a coin.
x <- pbinom(26,51,0.5)

print(x)
```

Outputs:

```
print(x)
```

[1] 0.610116

qbinom

# How many heads will have a probability of 0.25 will come out when a coin
# is tossed 51 times.
x <- qbinom(0.25,51,1/2)

print(x)
Output:

print(x)
[1] 23

rbinom

# Find 8 random values from a sample of 150 with probability of 0.4.
x <- rbinom(8,150,.4)

print(x)
Output:

print(x)
[1] 68 59 55 49 51 59 53 55

5. Write R program to do the following tests with the sample data and visualize the results graphically.
Source code:

a) χ_2-test

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

```
# Loading the Mass library.

# Creating a data frame from the main data set.
car_data<- data.frame(Cars93$AirBags, Cars93$Type)
# Creating a table with the needed variables.
car_data = table(Cars93$AirBags, Cars93$Type)
print(car_data)
# Performing the Chi-Square test.
print(chisq.test(car_data))
```

Output:

```
print(str(Cars93))
'data.frame':        93 obs. of  27 variables:
 $ Manufacturer   : Factor w/ 32 levels "Acura","Audi",..: 1 1
2 2 3 4 4 4 4 5 ...
 $ Model          : Factor w/ 93 levels "100","190E","240",..: 49 56
9 1 6 24 54 74 73 35 ...
 $ Type           : Factor w/ 6 levels "Compact","Large",..: 4 3 1 3
3 3 2 2 3 2 ...
 $ Min.Price      : num  12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6
26.3 33 ...
 $ Price          : num  15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7
...
 $ Max.Price      : num  18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9
26.3 36.3 ...
 $ MPG.city       : int  25 18 20 19 22 22 19 16 19 16 ...
 $ MPG.highway    : int  31 25 26 26 30 31 28 25 27 25 ...
 $ AirBags        : Factor w/ 3 levels "Driver & Passenger",..: 3
1 2 1 2 2 2 2 2 2 ...
 $ DriveTrain     : Factor w/ 3 levels "4WD","Front",..: 2 2 2 2 3
2 2 3 2 2 ...
 $ Cylinders      : Factor w/ 6 levels "3","4","5","6",..: 2 4 4 4 2 2
4 4 4 5 ...
 $ EngineSize     : num  1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...
 $ Horsepower     : int  140 200 172 172 208 110 170 180 170 200
...
 $ RPM            : int  6300 5500 5500 5500 5700 5200 4800 4000
4800 4100 ...
 $ Rev.per.mile   : int  2890 2335 2280 2535 2545 2565 1570
1320 1690 1510 ...
 $ Man.trans.avail : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 1
1 1 1 ...
```

```
$ Fuel.tank.capacity: num 13.2 18 16.9 21.1 21.1 16.4 18 23 18.8
18 ...
$ Passengers    : int 5 5 5 6 4 6 6 6 5 6 ...
$ Length        : int 177 195 180 193 186 189 200 216 198 206 ...
$ Wheelbase     : int 102 115 102 106 109 105 111 116 108 114 ...
$ Width         : int 68 71 67 70 69 69 74 78 73 73 ...
$ Turn.circle   : int 37 38 37 37 39 41 42 45 41 43 ...
$ Rear.seat.room  : num 26.5 30 28 31 27 28 30.5 30.5 26.5 35
...
$ Luggage.room  : int 11 15 14 17 13 16 17 21 14 18 ...
$ Weight        : int 2705 3560 3375 3405 3640 2880 3470 4105
3495 3620 ...
$ Origin        : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1
1 1 ...
$ Make          : Factor w/ 93 levels "Acura Integra",..: 1 2 4 3 5
6 7 9 8 10 ...
NULL
> # Creating a data frame from the main data set.
> car_data<- data.frame(Cars93$AirBags, Cars93$Type)
> # Creating a table with the needed variables.
> car_data = table(Cars93$AirBags, Cars93$Type)
> 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       4    16      3   6
> # Performing the Chi-Square test.
> print(chisq.test(car_data))

        Pearson's Chi-squared test

data:  car_data
X-squared = 33.001, df = 10, p-value = 0.0002723
```

b) t-test

```
x  <- c(0.593, 0.142, 0.329, 0.691, 0.231, 0.793, 0.519, 0.392, 0.418)
t.test(x, alternative="greater", mu=0.3)
```

output:

```
t.test(x, alternative="greater", mu=0.3)
```

One Sample t-test

data: x
t = 2.2051, df = 8, p-value = 0.02927
alternative hypothesis: true mean is greater than 0.3
95 percent confidence interval:
0.3245133 Inf
sample estimates:
mean of x
0.4564444

c) F-test

```
install.packages("randomForest")
# Load the party package. It will automatically load other
# required packages.
library(party)

# Print some records from data set readingSkills.
print(head(readingSkills))


# Load the party package. It will automatically load other
# required packages.
library(party)
library(randomForest)

# Create the forest.
output.forest <- randomForest(nativeSpeaker ~ age + shoeSize + score,
                              data = readingSkills)

# View the forest results.
print(output.forest)
```

Output:

print(output.forest)

Call:

```
randomForest(formula = nativeSpeaker ~ age + shoeSize
+ score,  data = readingSkills)
        Type of random forest: classification
              Number of trees: 500
No. of variables tried at each split: 1

    OOB estimate of  error rate: 1.5%
Confusion matrix:
  no yes class.error
no 99  1        0.01
yes 2 98        0.02
```

Viva Questions:

1) What is the Main Goal of Data Mining?

Data mining is an interdisciplinary subfield of computer science and statistics with an overall goal to extract information (with intelligent methods) from a data set and transform the information into a comprehensible structure for further use.

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

The data mining tasks can be classified generally into two types based on what a specific task tries to achieve. Those two categories are descriptive tasks and predictive tasks.

3) What are the Data Mining Functionalities?

- Class/Concept Description
- Mining of Frequent Patterns
- Mining of Associations

- Mining of Correlations
- Mining of Clusters
- Classification
- Prediction
- Outlier Analysis
- Evolution Analysis

4) What are the Features of WEKA?

Weka features include machine learning, data mining, preprocessing, classification, regression, clustering, association rules, attribute selection, experiments, workflow and visualization.

5) Navigate the options available in the WEKA.

- Preprocess
- Classify
- Cluster
- Associate
- Select Attributes
- Visualize

6) What is ARFF file format?

An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes.

7) Define Support and Confidence.

Support represents the popularity of that product of all the product transactions. Confidence can be interpreted as the likelihood of purchasing both the products A and B.

8) What are the frequent patterns?

Frequent patterns are itemsets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set, is a frequent itemset.

9) Where we are using Apriori Algorithm in Real time scenario?

Usually, you operate this algorithm on a database containing a large number of transactions. One such example is the items customers buy at a supermarket. It helps the customers buy their items with ease, and enhances the sales performance of the departmental store.

10) Explain Association rule with a suitable example.

A classic example of association rule mining refers to a relationship between diapers and beers. The example, which seems to be fictional, claims that men who go to a store to buy diapers are also likely to buy beer.

11) What is Apriori Property?

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database.

12) How can we further improve the efficiency of Apriori-based mining?

Based on the inherent defects of Apriori algorithm, some related improvements are carried out: 1) using new database mapping way to avoid scanning the database repeatedly; 2) further pruning frequent itemsets and candidate itemsets in order to improve joining efficiency;

13) Define Classification.

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.

14) Define Prediction.

The prediction, as it name implied, is one of a data mining techniques that discovers relationship between independent variables and relationship between dependent variables.

15) What are the classification techniques in data mining?

- Logistic Regression
- Naïve Bayes
- Stochastic Gradient Descent
- K-Nearest Neighbours
- Decision Tree
- Random Forest
- Support Vector Machine

16) What are the advantages of different classification algorithms

Mining Based Methods are cost effective and efficient. Helps in identifying criminal suspects. Helps in predicting risk of diseases.

17) What is the Kappa Statistic

"The Kappa statistic (or value) is a metric that compares an Observed Accuracy with an Expected Accuracy (random chance). The kappa statistic is used not only to evaluate a single classifier, but also to evaluate classifiers amongst themselves.

18) Which classification algorithm is best for prediction and analysis?

- Time Series Model. The time series model comprises a sequence of data points captured, using time as the input parameter. ...
- Random Forest. Random Forest is perhaps the most popular classification algorithm, capable of both classification and regression.

19) What are the classification algorithms in data mining?

Six classification algorithms—Naive Bayes, Bayesian networks, J48, random forest, multilayer perceptron, and logistic regression

20) Which data mining algorithm provides best accuracy for classification?

The performances of the algorithms were compared according to accuracy, root mean squared error, ROC area, F-measure, precision, and recall criteria, and the logistic regression classification algorithm was found to be the best algorithm.

21) What are the best classification algorithms?

Top 5 Data Mining Algorithms for Classification

- Decision Trees.
- Logistic Regression.
- Naive Bayes Classification.
- k-nearest neighbors.
- Support Vector Machine.

22) Which are the best classifier algorithms for disease predictions?

Comparative analysis of classification techniques has shown that decision tree classifiers are simple and accurate [9]. Naïve Bayes was found to be the best algorithm, followed by neural networks and decision trees

23) Explain Decision Tree.

A decision tree is a diagram or chart that people use to determine a course of action or show a statistical probability. Each branch of the decision tree represents a possible decision, outcome, or reaction.

24) What is the Cross-Validation?

Cross validation is a technique for assessing how the statistical analysis generalizes to an independent data set. It is a technique for evaluating machine learning models by training

several models on subsets of the available input data and evaluating them on the complementary subset of the data.

25) What is the Naïve-Bayes Classification?

It is a **classification** technique based on **Bayes'** Theorem with an assumption of independence among predictors. In simple terms, a **Naive Bayes classifier** assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

26) What are the methods in classification methods in data mining?

There are many **techniques** for solving **classification** problems: **classification** trees, logistic regression, discriminant analysis, neural networks, boosted trees, random forests, deep learning **methods**, nearest neighbors, support vector machines.

27) Briefly explain the K-nearest neighbor algorithm

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (**K**) **closest** to the query, then votes for the most frequent label (in the case of classification) or averages the labels

28) How do you choose an algorithm for a classification problem?

- Size of the training data. It is usually recommended to gather a good amount of data to get reliable predictions.
- Accuracy and/or Interpretability of the output.
- Speed or Training time.
- Linearity.
- Number of features.

29) What are the most useful algorithms used for data mining

K Nearest Neighbors Algorithm, Naïve Bayes Algorithm, SVM Algorithm, ANN Algorithm, 48 Decision Trees, Support Vector Machines, and SenseClusters.

30) What is the difference between classification and prediction?

Classification is the process of identifying the category or class label of the new observation to which it belongs. Predication is the process of identifying the missing or unavailable numerical data for a new observation.

31) What is clustering with example?

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

32) What are the examples of clustering?

- Identifying Fake News. Fake news is not a new phenomenon, but it is one that is becoming prolific.
- Spam filter
- Marketing and Sales
- Classifying network traffic
- Identifying fraudulent or criminal activity
- Document analysis
- Fantasy Football and Sports

33) Why Clustering is used in data mining

Clustering in Data Mining helps in the classification of animals and plants are done using similar functions or genes in the field of biology. It helps in gaining insight into the structure of the species. Areas are identified using the clustering in data mining.

34) Why we use K means clustering

The K-means clustering algorithm is used to find groups which have not been explicitly labeled in the data. This can be used to confirm business assumptions about what types of groups exist or to identify unknown groups in complex data sets.

35) What are the advantages and disadvantages of K means clustering

Advantages

Relatively simple to implement.
Scales to large data sets.
Guarantees convergence.

Disadvantages

Clustering data of varying sizes and density.
Clustering outliers.

36) Which is the best clustering algorithm?

- K-means Clustering Algorithm. ...
- Mean-Shift Clustering Algorithm. ...
- DBSCAN – Density-Based Spatial Clustering of Applications with Noise.

37) Explain the Hierarchical Clustering

HCA is an unsupervised clustering algorithm which involves creating clusters that have predominant ordering from top to bottom.

38) State the other Clustering Techniques.

- Connectivity-based Clustering (Hierarchical clustering)
- Centroids-based Clustering (Partitioning methods)
- Distribution-based Clustering.
- Density-based Clustering (Model-based methods)
- Fuzzy Clustering.
- Constraint-based (Supervised Clustering)

39) What are the two types of hierarchical clustering?

- Agglomerative clustering: It's also known as AGNES (Agglomerative Nesting). It works in a bottom-up manner. ...
- Divisive hierarchical clustering: It's also known as DIANA (Divise Analysis) and it works in a top-down manner.

40) What are the disadvantages of agglomerative hierarchical clustering?

One drawback is that groups with close pairs can merge sooner than is optimal, even if those groups have overall dissimilarity. Complete Linkage: calculates similarity of the farthest away pair.

41) What is the use of hierarchical clustering?

Hierarchical clustering is the most popular and widely used method to analyze social network data. In this method, nodes are compared with one another based on their similarity. Larger groups are built by joining groups of nodes based on their similarity.

42) What is the difference between K means and hierarchical clustering?

Hierarchical clustering can't handle big data well but K Means clustering can. This is because the time complexity of K Means is linear i.e. O(n) while that of hierarchical clustering is quadratic i.e. $O(n^2)$.

43) What is the mean, median and mode

The mean (average) of a data set is found by adding all numbers in the data set and then dividing by the number of values in the set. The median is the middle value when a data set is ordered from least to greatest. The mode is the number that occurs most often in a data set.

44) How do you find the mode in r?

To find the mode, or modal value, it is best to put the numbers in order. Then count how many of each number. A number that appears most often is the mode.

45) Define analysis of covariance

Analysis of Covariance (ANCOVA) is the inclusion of a continuous variable in addition to the variables of interest (i.e., the dependent and independent variable) as means for control.

46) What is analysis of covariance used for?

Analysis of covariance is used to test the main and interaction effects of categorical variables on a continuous dependent variable, controlling for the effects of selected other continuous variables, which co-vary with the dependent. The control variables are called the "covariates."

47) How do you analyze covariance?

The Analysis of covariance (ANCOVA) is done by using linear regression. This means that Analysis of covariance (ANCOVA) assumes that the relationship between the independent variable and the dependent variable must be linear in nature.

48) Why should we use R?

R is a programming language for statistical computing and graphics that you can use to clean, analyze, and graph your data. It is widely used by researchers from diverse disciplines to estimate and display results and by teachers of statistics and research methods.

49) Define regression and its types

Regression is a technique used to model and analyze the relationships between variables and often times how they contribute and are related to producing a particular outcome together.
The two basic types of regression are simple linear regression and multiple linear regression.

50) What is regression in statistics with example?

Linear regression quantifies the relationship between one or more predictor variable(s) and one outcome variable. ... For example, it can be used to quantify the relative impacts of age, gender, and diet (the predictor variables) on height (the outcome variable).

51) What is meant by linear regression?

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.

52) What is multiple linear regression explain with example

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. Multiple regression is an extension of linear (OLS) regression that uses just one explanatory variable.

53) What is the difference between linear regression and multiple regression?

Linear regression is one of the most common techniques of regression analysis. Multiple regression is a broader class of regressions that encompasses linear and nonlinear regressions with multiple explanatory variables.

54) What is difference between linear and logistic regression

The essential difference between these two is that
Logistic regression is used when the dependent variable is binary in nature.
Incontrast, linear regression is used when the dependent variable is continuous and nature of t he regression line is linear.

55) What is time series analysis with example?

a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of discrete-time data. Examples of time series are heights of ocean tides, counts of sunspots, and the daily closing value of the Dow Jones Industrial Average.

56) How do you analyze time series data in R?

The first thing that you will want to do to analyse your time series data will be to read it into R, and to plot the time series. You can read data into R using the scan() function, which assumes that your data for successive time points is in a simple text file with one column.

57) What is the purpose of time series analysis?

Time series analysis can be useful to see how a given asset, security, or economic variable changes over time. It can also be used to examine how the changes associated with the chosen data point compare to shifts in other variables over the same time period.

58) What is difference between linear and nonlinear?

While a linear equation has one basic form, nonlinear equations can take many different forms. Thetas represent the parameters and X represents the predictor in the nonlinear functions. Unlike linear regression, these functions can have more than one parameter per predictor variable.

59) What is decision tree and example?

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. ... An example of a decision tree can be explained using above binary tree.

60) How do you create a decision tree in R?

Step 1: Import the data.

Step 2: Clean the dataset.

Step 3: Create train/test set.

Step 4: Build the model.

Step 5: Make prediction.

Step 6: Measure performance.

Step 7: Tune the hyper-parameters.

61) What is the Rnorm function in R?

rnorm is the R function that simulates random variates having a specified normal distribution. As with pnorm , qnorm , and dnorm , optional arguments specify the mean and standard deviation of the distribution.

62) What is the Pnorm and Qnorm in R

pnorm: cumulative density function of the normal distribution. qnorm: quantile function of the normal distribution.

63) What is the normal distribution with example?

For example, heights, blood pressure, measurement error, and IQ scores follow the normal distribution. It is also known as the Gaussian distribution and the bell curve.

64) Where is normal distribution used?

Normal distribution, also called Gaussian distribution, the most common distribution function for independent, randomly generated variables. Its familiar bell-shaped curve is ubiquitous in statistical reports, from survey analysis and quality control to resource allocation.

65) What is Dbinom

The function dbinom returns this probability. There are three required arguments: the value(s) for which to compute the probability (j), the number of trials (n), and the success probability for each trial (p).

66) What is the difference between Pbinom and Dbinom

dbinom is a probability mass function of binomial distribution, while pbinom is a cumulative distribution function of this distribution.

67) Define chi square test and its application

The Chi Square test is a statistical hypothesis test in which the sampling distribution of the test statistic is a chi-square distribution when the null hypothesis is true. The Chi square test is used to compare a group with a value, or to compare two or more groups, always using categorical data.

68) When should we use chi square test

The Chi Square statistic is commonly used for testing relationships between categorical variables. The null hypothesis of the Chi-Square test is that no relationship exists on the categorical variables in the population; they are independent.

69) What are the limitations of chi square?

Chi-square, like any analysis has its limitations. One of the limitations is that all participants measured must be independent, meaning that an individual cannot fit in more than one category. If a participant can fit into two categories a chi-square analysis is not appropriate.

70) What is the difference between t test and F TEST?

T-test is used to test if two samples have the same mean. The assumptions are that they are samples from normal distribution. F-test is used to test if two samples have the same variance.

71) What is the difference between chi square and t test?

A t-test tests a null hypothesis about two means; most often, it tests the hypothesis that two means are equal, or that the difference between them is zero. A chi-square test tests a null hypothesis about the relationship between two variables.

72) Where do we use chi square test

The Chi Square statistic is commonly used for testing relationships between categorical variables. The null hypothesis of the Chi-Square test is that no relationship exists on the categorical variables in the population; they are independent.