**CHAROTAR UNIVERSITY OF SCIENCE & TECHNOLOGY**

**FACULTY OF TECHNOLOGY & ENGINEERING**

**DEVANG PATEL INSTITUTE OF ADVANCE TECHNOLOGY AND RESEARCH**

**Subject: CE350 Data Warehousing & Data Mining**

**[DWDM (CE350)]**

**Semester: 6th [BTECH CE]**

**Academic year: Jan 2020 - April 2021**

**PRACTICAL - 1**

**AIM: Data Preprocessing using Pandas (Handling Missing Value, Data Wrangling, Dimension Reduction)**

**THEORY:**

**Python**

* Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
* Why Python?
* Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.
* Python is Interpreted
* Python is Interactive
* Python is Object-Oriented
* Python is a Beginner's Language
* Characteristics of Python
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
* Applications of Python
* Easy-to-learn
* Easy-to-read
* Easy-to-maintain
* A broad standard library
* Interactive Mode
* Portable
* Extendable
* Databases
* GUI Programming
* Scalable

**Pandas**

Pandas is the most popular python library that is used for data analysis. It provides highly optimized performance with back-end source code is purely written in C or Python.

* Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
* Size mutability: columns can be inserted and deleted from Data Frame and higher dimensional objects
* Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, Data Frame, etc. automatically align the data for you in computations
* Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
* Make it easy to convert ragged, differently-indexed data in other Python and NumPydata structures into Data Frame objects.
* Intelligent label-based slicing, fancy indexing, and sub setting of large data sets
* Intuitive merging and joining data sets
* Flexible reshaping and pivoting of data sets
* Hierarchical labeling of axes (possible to have multiple labels per tick)
* Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving/loading data from the ultrafast HDF5 format
* Time series-specific functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging.

**NumPy**

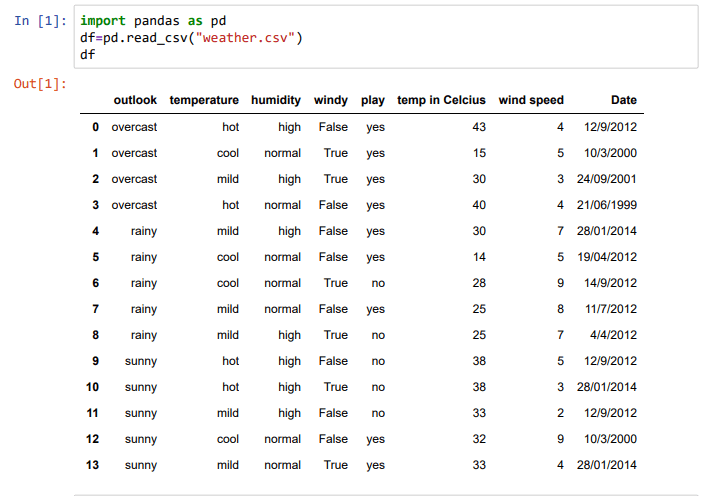
NumPy is an open source library available in Python that aids in mathematical, scientific, engineering, and data science programming. NumPy is an incredible library to perform mathematical and statistical operations. It works perfectly well for multi-dimensional arrays and matrices multiplication

## Why use NumPy?

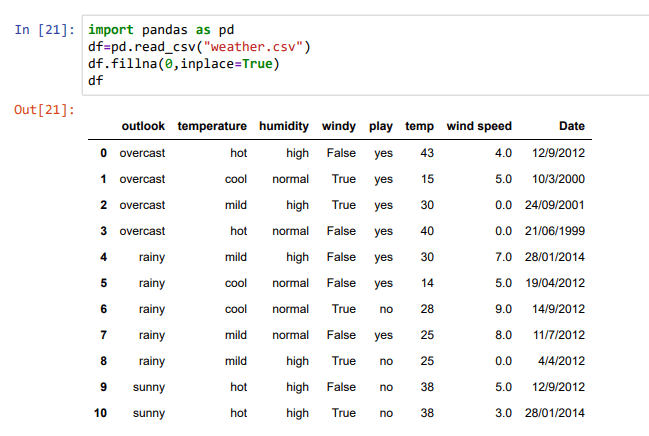
* NumPy is memory efficiency, meaning it can handle the vast amount of data more accessible than any other library. Besides, NumPy is very convenient to work with, especially for matrix multiplication and reshaping. On top of that, NumPy is fast. In fact, Tensor Flow and Scikit learn to use NumPy array to compute the matrix multiplication in the back end.
* Characteristics :
* NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
* The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
* NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
* A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to
* NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python- based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

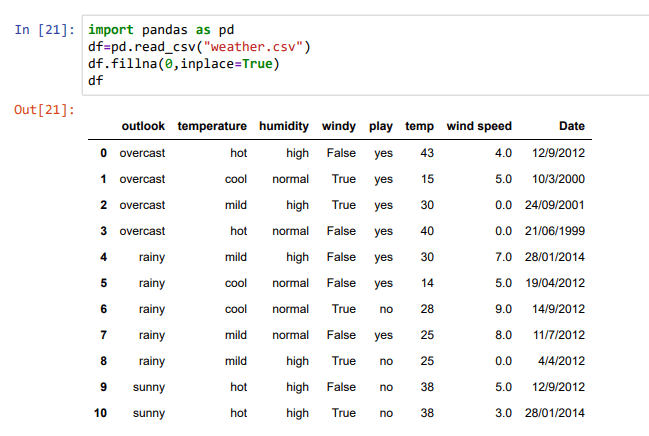
**Working with Pandas**

Data:

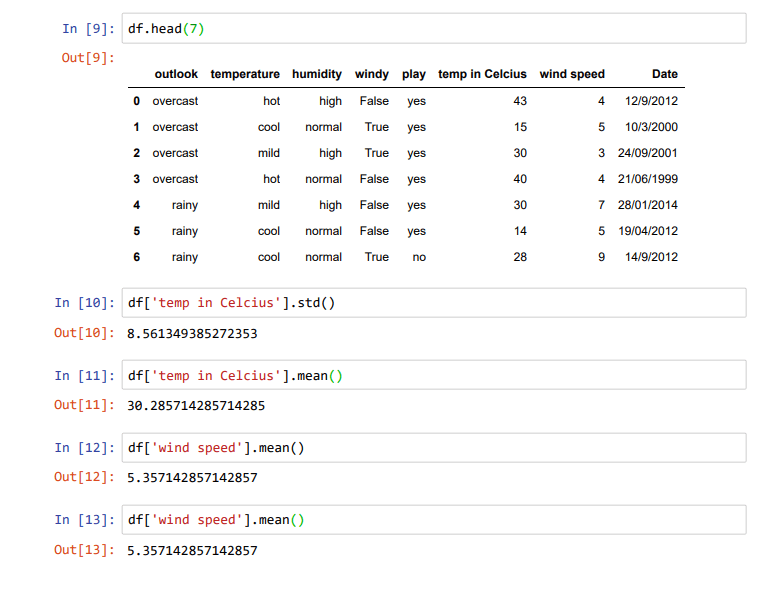
****

**Missing Data:**

****

****

**Operations:**

****

**PRACTICAL - 2**

**Aim: Vehicle data analysis using matplotlib and seaborn**

**Theory:**

**Matplotlib: It** is a [plotting](https://en.wikipedia.org/wiki/Plotter) [library](https://en.wikipedia.org/wiki/Library_(computer_science)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) programming language and its numerical mathematics extension [NumPy](https://en.wikipedia.org/wiki/NumPy). Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.



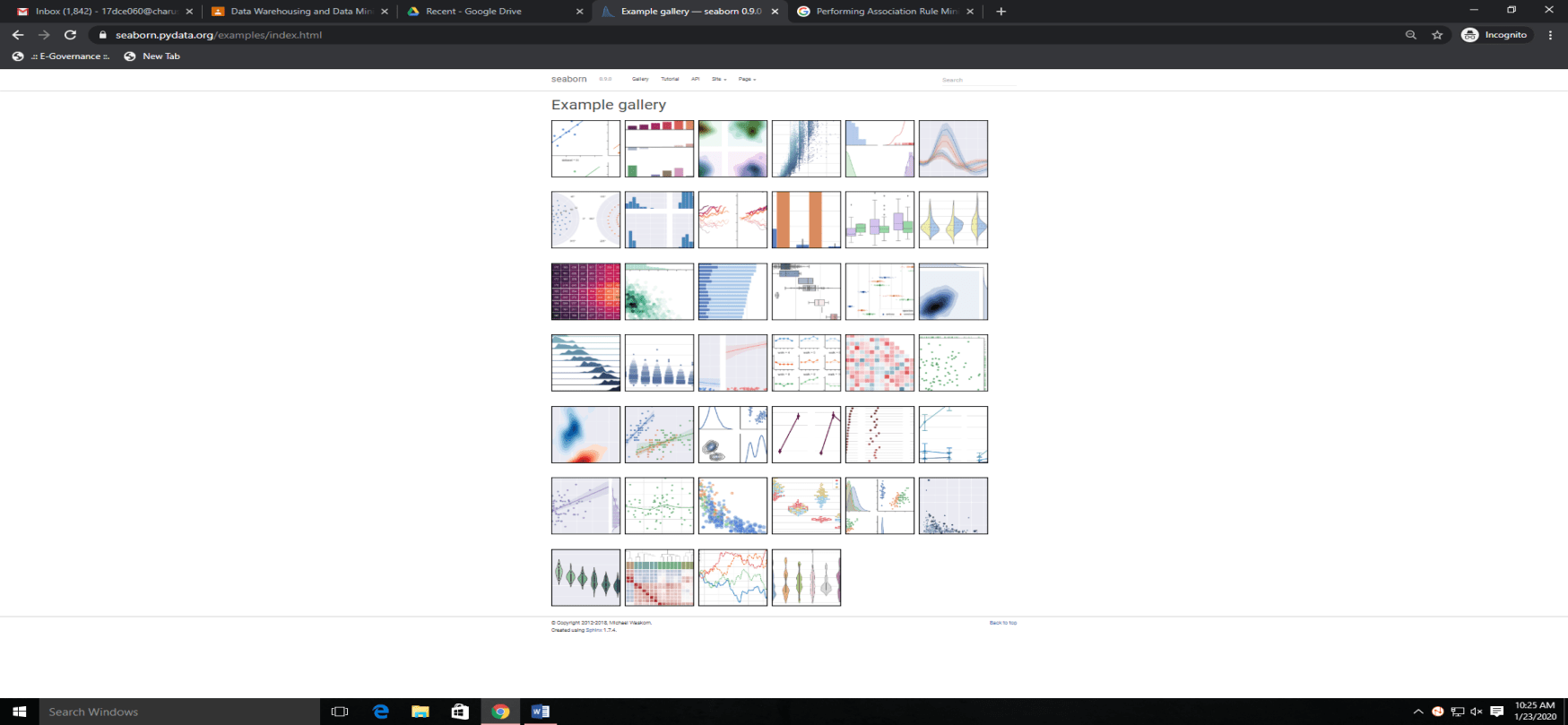
**Seaborn:**

Seaborn is a library for making statistical graphics in Python. It is built on top of [matplotlib](https://matplotlib.org/) and closely integrated with [pandas](https://pandas.pydata.org/) data structures.

Here is some of the functionality that seaborn offers:

* A dataset-oriented API for examining [relationships](https://seaborn.pydata.org/examples/scatter_bubbles.html#scatter-bubbles) between [multiple variables](https://seaborn.pydata.org/examples/faceted_lineplot.html#faceted-lineplot)
* Specialized support for using categorical variables to show [observations](https://seaborn.pydata.org/examples/jitter_stripplot.html#jitter-stripplot) or [aggregate statistics](https://seaborn.pydata.org/examples/pointplot_anova.html#pointplot-anova)
* Options for visualizing [univariate](https://seaborn.pydata.org/examples/distplot_options.html#distplot-options) or [bivariate](https://seaborn.pydata.org/examples/joint_kde.html#joint-kde) distributions and for [comparing](https://seaborn.pydata.org/examples/horizontal_boxplot.html#horizontal-boxplot) them between subsets of data
* Automatic estimation and plotting of [linear regression](https://seaborn.pydata.org/examples/anscombes_quartet.html#anscombes-quartet) models for different kinds [dependent](https://seaborn.pydata.org/examples/logistic_regression.html#logistic-regression) variables
* Convenient views onto the overall [structure](https://seaborn.pydata.org/examples/scatterplot_matrix.html#scatterplot-matrix) of complex datasets
* High-level abstractions for structuring [multi-plot grids](https://seaborn.pydata.org/examples/faceted_histogram.html#faceted-histogram) that let you easily build [complex](https://seaborn.pydata.org/examples/pair_grid_with_kde.html#pair-grid-with-kde) visualizations
* Concise control over matplotlib figure styling with several [built-in themes](https://seaborn.pydata.org/tutorial/aesthetics.html#aesthetics-tutorial)
* Tools for choosing [color palettes](https://seaborn.pydata.org/tutorial/color_palettes.html#palette-tutorial) that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.



**Working:**

**Code:**

import pandas as pd

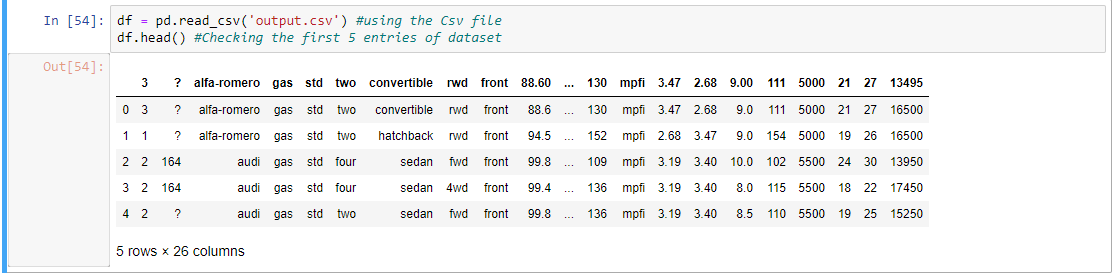
import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import scipy as sp

**Output:**



**Code:**

headers = ["symboling", "normalized-losses", "make",

"fuel-type", "aspiration","num-of-doors",

"body-style","drive-wheels", "engine-location",

"wheel-base","length", "width","height", "curb-weight",

"engine-type","num-of-cylinders", "engine-size",

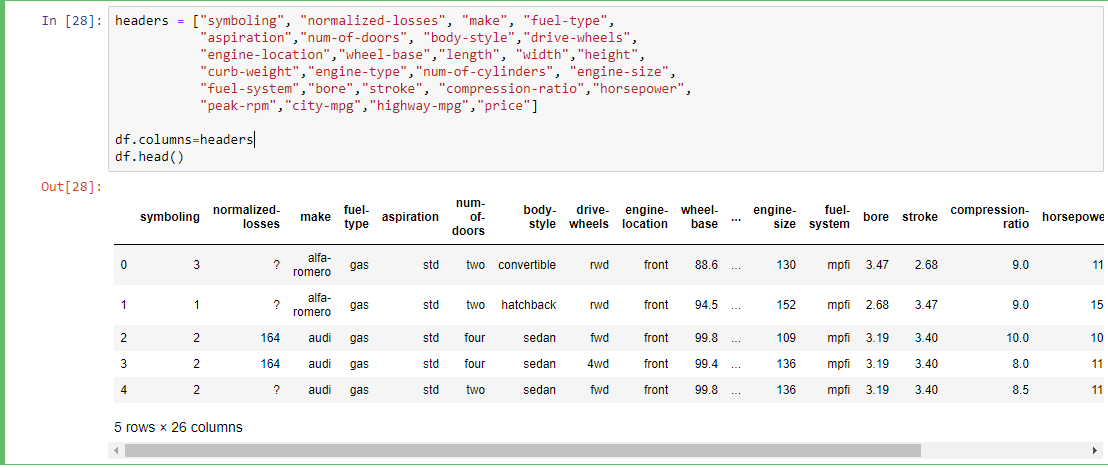
"fuel-system","bore","stroke", "compression-ratio",

"horsepower", "peak-rpm","city-mpg","highway-mpg","price"]

df.columns=headers

df.head()

**Output:**



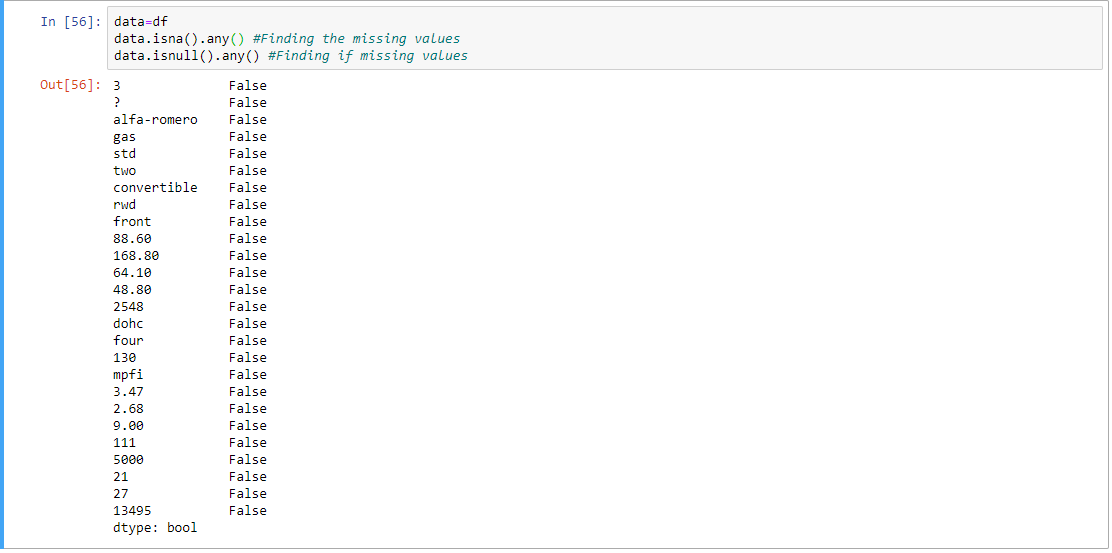
**Code:**

data = df

data.isna().any()

data.isnull().any()

**Output:**



**Code:**

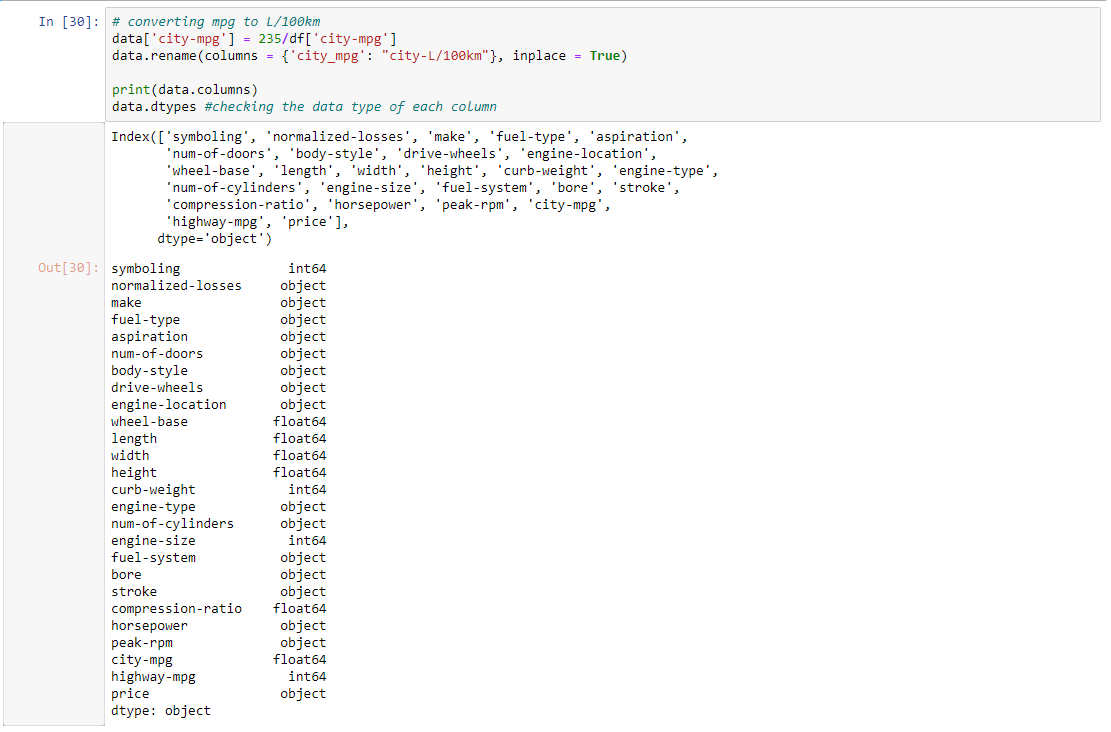
data['city-mpg'] = 235 / df['city-mpg']

data.rename(columns = {'city\_mpg': "city-L / 100km"}, inplace = True)

print(data.columns)

data.dtypes

**Output:**



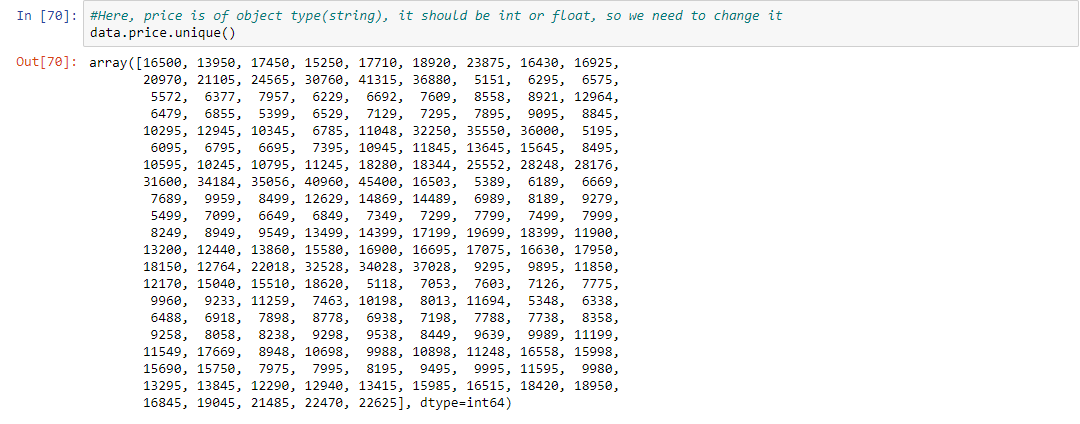
**Code:**

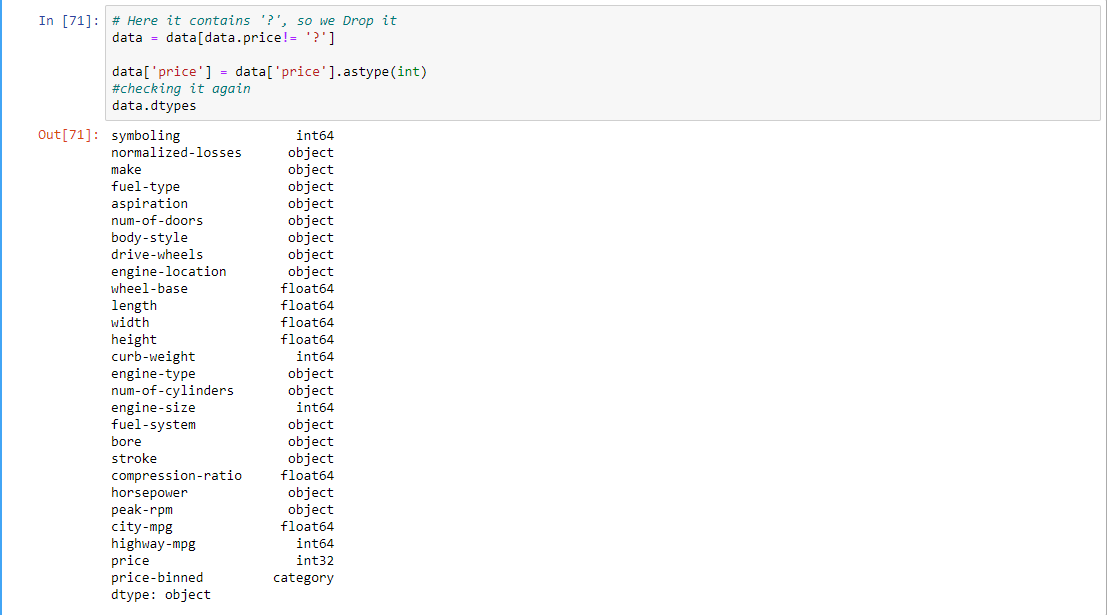
data.price.unique()

data = data[data.price != '?']

data.dtypes

**Output:**





**Code:**

data['length'] = data['length']/data['length'].max()

data['width'] = data['width']/data['width'].max()

data['height'] = data['height']/data['height'].max()

 bins = np.linspace(min(data['price']), max(data['price']), 4)

group\_names = ['Low', 'Medium', 'High']

data['price-binned'] = pd.cut(data['price'], bins,

                              labels = group\_names,

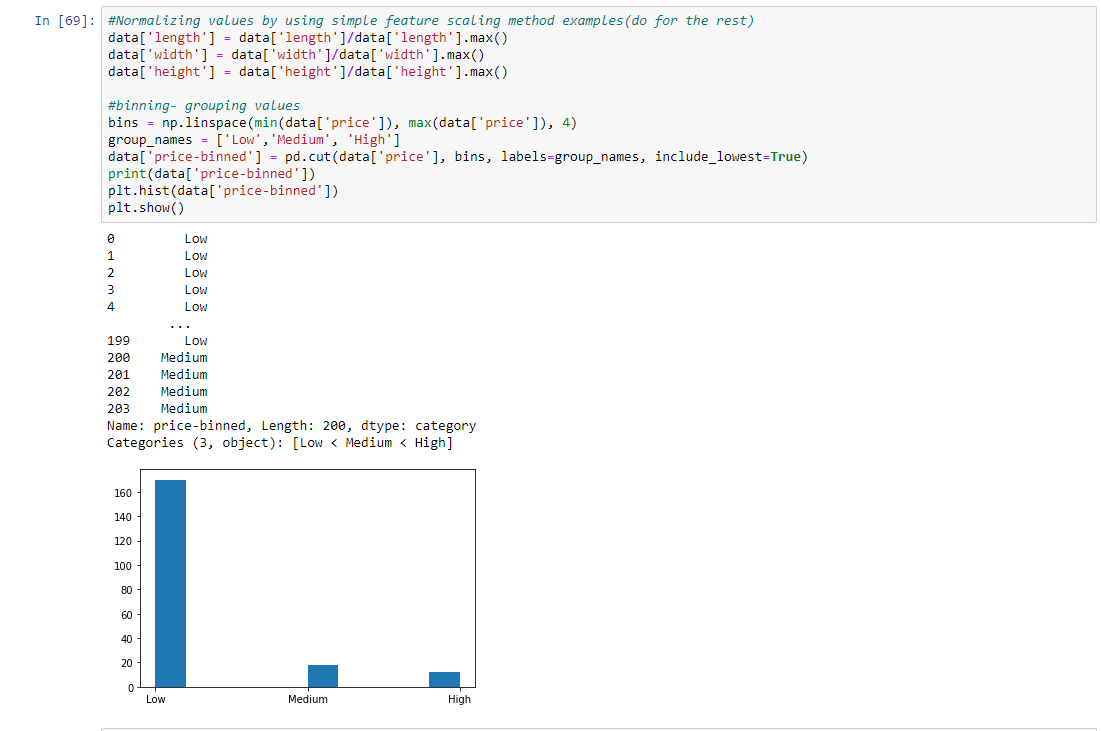
                              include\_lowest = True)

print(data['price-binned'])

plt.hist(data['price-binned'])

plt.show()

**Output:**

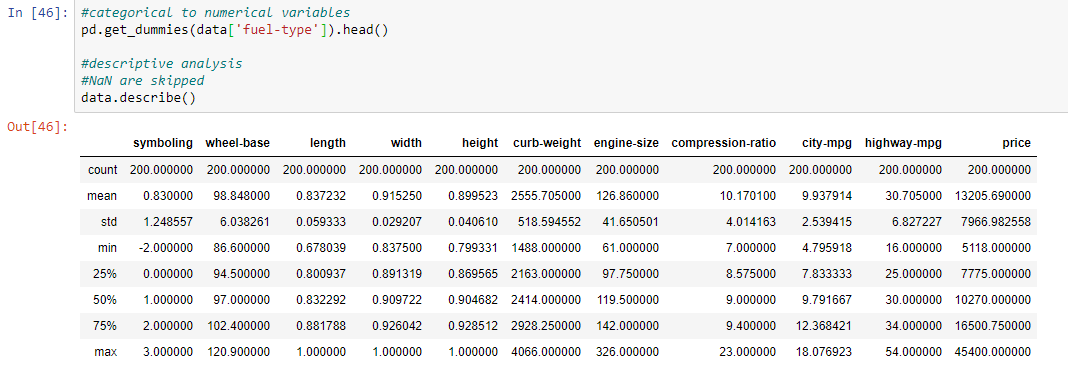


**Code:**

pd.get\_dummies(data['fuel-type']).head()

data.describe()

**Output:**



**Code:**

plt.boxplot(data['price'])

sns.boxplot(x ='drive-wheels', y ='price', data = data)

plt.scatter(data['engine-size'], data['price'])

plt.title('Scatterplot of Enginesize vs Price')

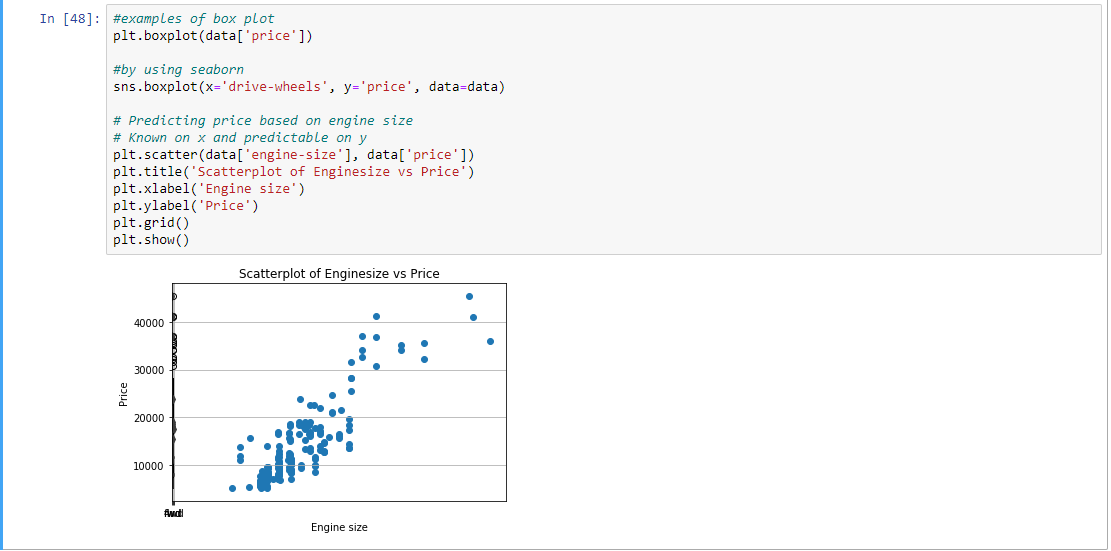
plt.xlabel('Engine size')

plt.ylabel('Price')

plt.grid()

plt.show()

**Output:**



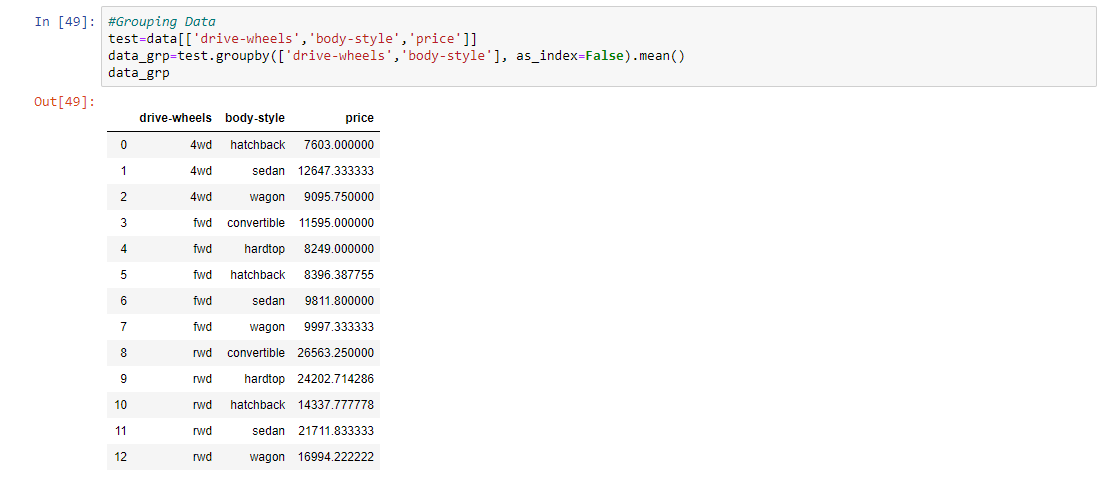
**Code:**

test = data[['drive-wheels', 'body-style', 'price']]

data\_grp = test.groupby(['drive-wheels', 'body-style'], as\_index = False).mean()

data\_grp

**Output:**



**Code:**

data\_pivot = data\_grp.pivot(index = 'drive-wheels', columns = 'body-style')

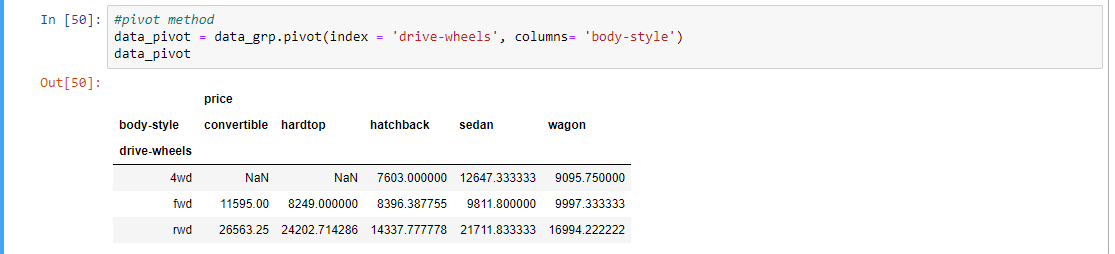
data\_pivot

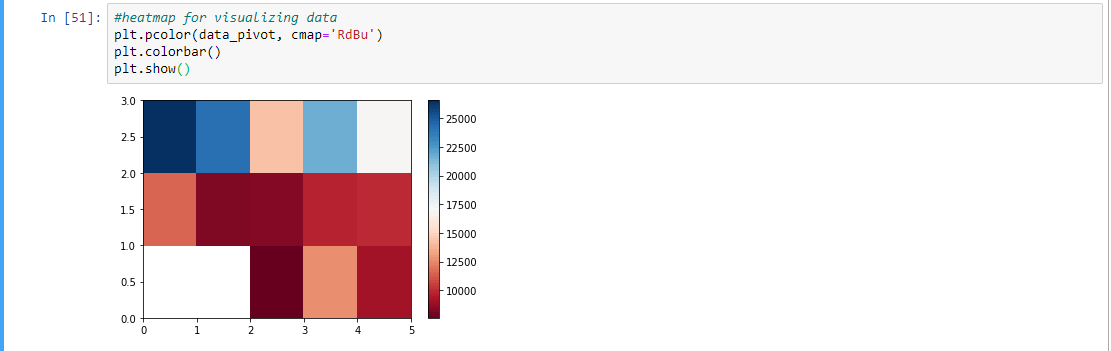
plt.pcolor(data\_pivot, cmap ='RdBu')

plt.colorbar()

plt.show()

**Output:**





**Code:**

data\_annova = data[['make', 'price']]

grouped\_annova = data\_annova.groupby(['make'])

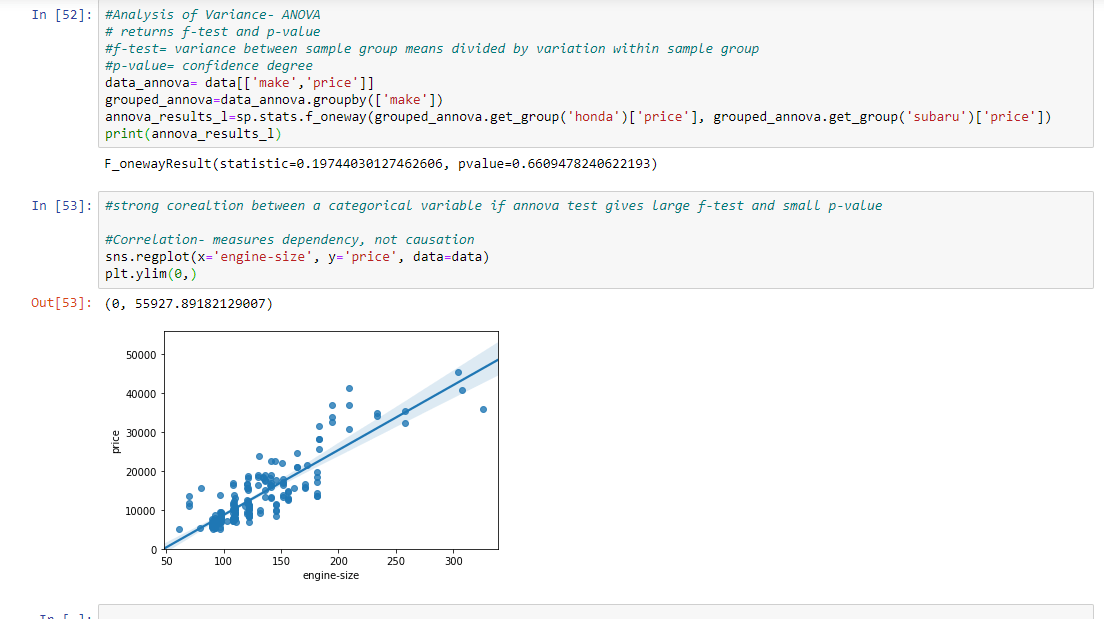
annova\_results\_l = sp.stats.f\_oneway( grouped\_annova.get\_group('honda')['price'], grouped\_annova.get\_group('subaru')['price'] )

print(annova\_results\_l)

sns.regplot(x ='engine-size', y ='price', data = data)

plt.ylim(0, )

**Output:**



**PRACTICAL - 3**

**Aim: Introduction to R programming: R-GUI, RStudio – Basic working & Commands**

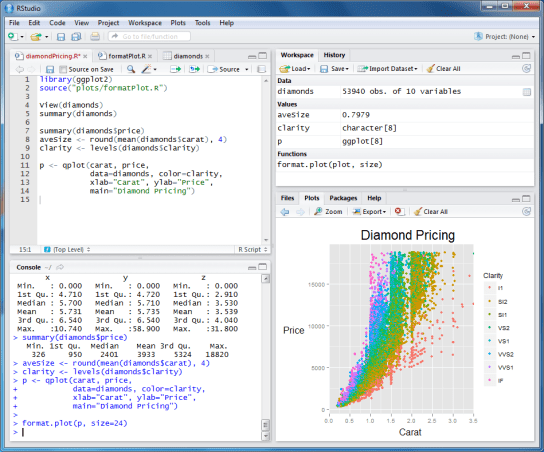
**Theory:**

**R Programming: R** is a programming language and software environment used for statistical analysis, data modeling, and graphical representation and reporting. R is best tool for software programmers, statisticians and data miners who looking forward for to easily manipulate and present data in compelling ways. R is an interpreted language; users typically access it through a command-line interpreter.

**R-GUI:**

As part of the process of downloading and installing R, you get the standard graphical user interface (GUI), called RGui. RGui gives you some tools to manage your R environment — most important, a console window. The console is where you type instructions, or scripts, and generally get R to do useful things for you. **RStudio :**

RStudio is an integrated development environment (IDE) for R, a programming language for statistical computing and graphics. It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser.



**Difference:**

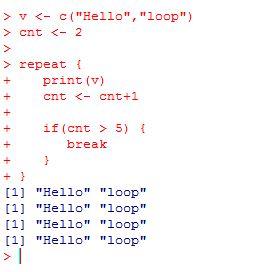
It is important to note the differences between R and RStudio. R is a programming language used for statistical computing while RStudio uses the R language to develop statistical programs. In R, you can write a program and run the code independently of any other computer program. RStudio however, must be used alongside R in order to properly function. Often referred to as an IDE, or integrated development environment, RStudio allows users to develop and edit programs in R by supporting a large number of statistical packages, higher quality graphics, and the ability to manage your workspace.

R and RStudio are not separate versions of the same program, and cannot be substituted for one another. R may be used without RStudio, but RStudio may not be used without R.

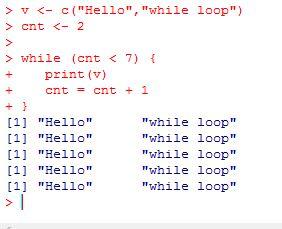
## R Command Prompt

## LOOPS

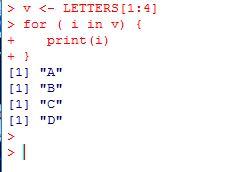
* **repeat loop :**Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.



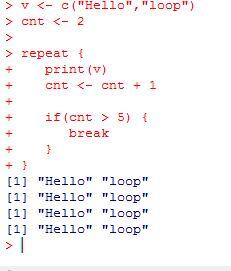
* **While loop: Repeats** a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.



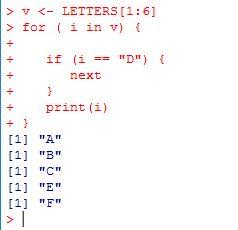
* **For loop: Like** a while statement, except that it tests the condition at the end of the loop body.



* **Break statement: Terminates** the **loop** statement and transfers execution to the statement immediately following the loop.



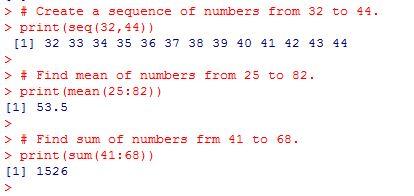
* **Next statement: The** **next** statement simulates the behavior of R switch.



**FUNCTION**

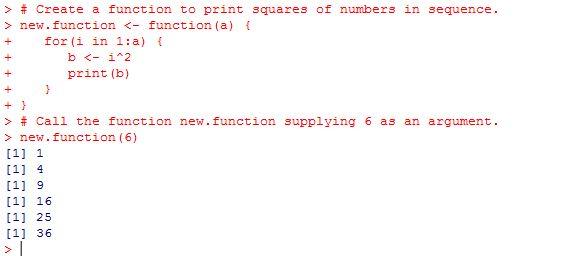
## Built-in Function:

Simple examples of in-built functions are **seq()**, **mean()**, **max()**, **sum(x)** and **paste(...)** etc. They are directly called by user written programs. You can refer most widely used Rfunctions.



## User-defined Function

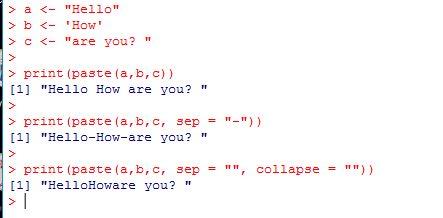
We can create user-defined functions in R. They are specific to what a user wants and once created they can be used like the built-in functions. Below is an example of how a function is created and used.



## STRING

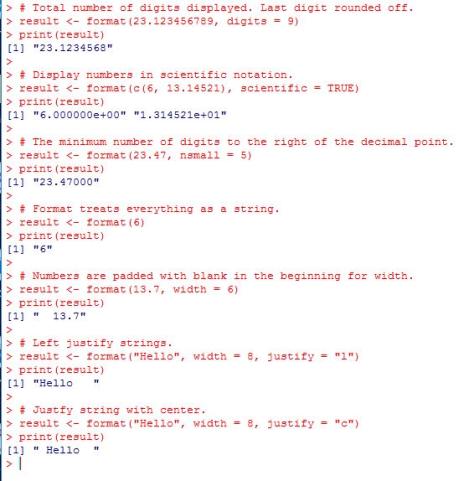
### **Concatenating Strings - paste() function**

Many strings in R are combined using the **paste()** function. It can take any number of arguments to be combined together.



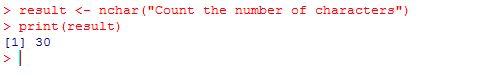
### **Formatting numbers & strings - format() function**

Numbers and strings can be formatted to a specific style using **format()** function.



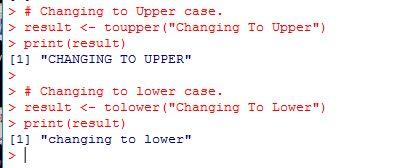
### **Counting number of characters in a string - nchar() function**

This function counts the number of characters including spaces in a string.



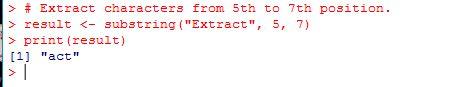
### **Changing the case - toupper() &tolower() functions**

These functions change the case of characters of a string.



### **Extracting parts of a string - substring() function**

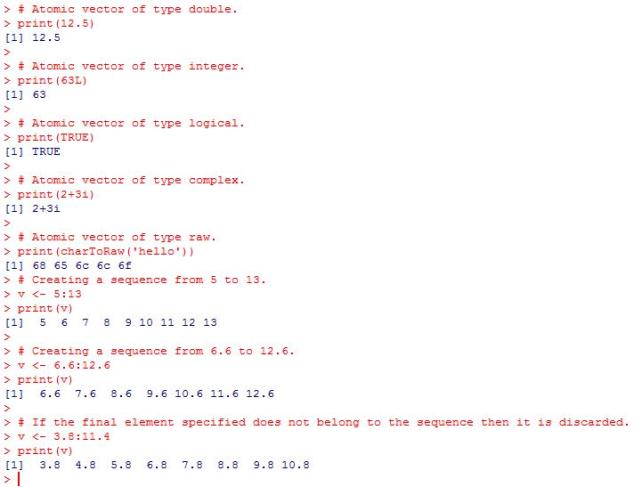
This function extracts parts of a String.



**VECTORS**

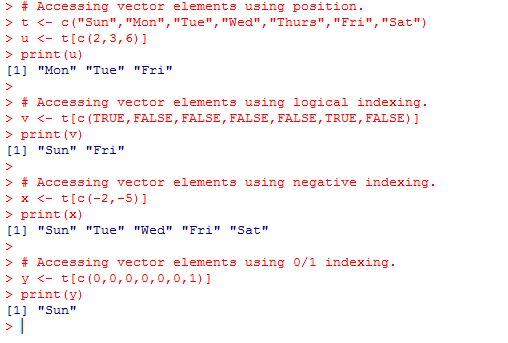
### **Single Element Vector**

Even when you write just one value in R, it becomes a vector of length 1 and belongs to one of the above vector types.



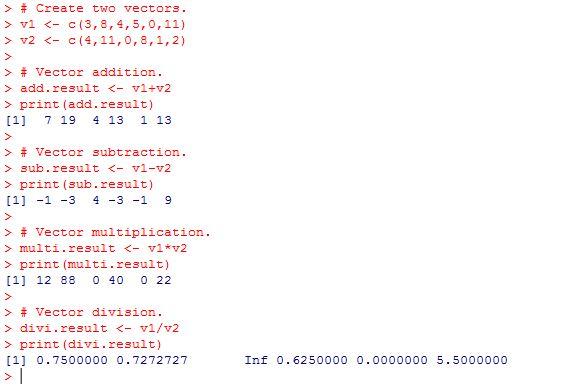
## Accessing Vector Elements

Elements of a Vector are accessed using indexing. The **[ ] brackets** are used for indexing. Indexing starts with position 1. Giving a negative value in the index drops that element from result.**TRUE**,**FALSE** or **0** and **1** can also be used for indexing.



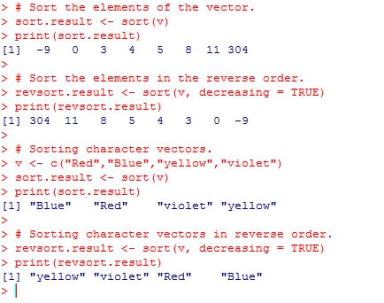
### **Vector arithmetic**

Two vectors of same length can be added, subtracted, multiplied or divided giving the result as a vector output.



### **Vector Element Sorting**

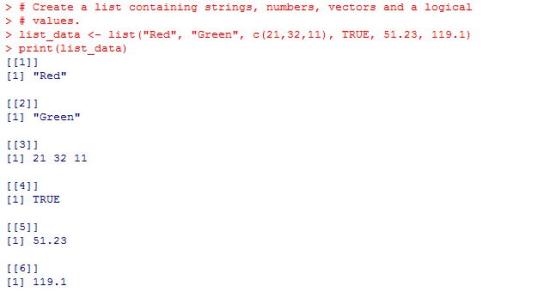
Elements in a vector can be sorted using the **sort()** function.



**LIST**

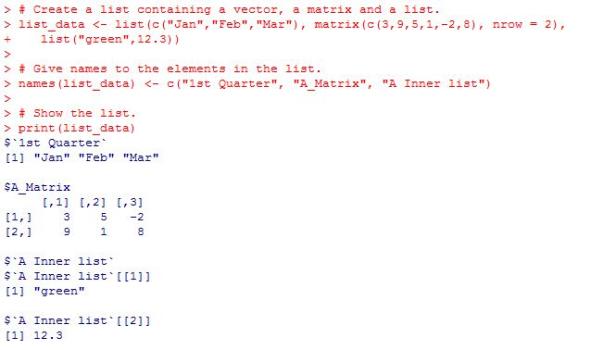
## Creating a List

Following is an example to create a list containing strings, numbers, vectors and a logical values.



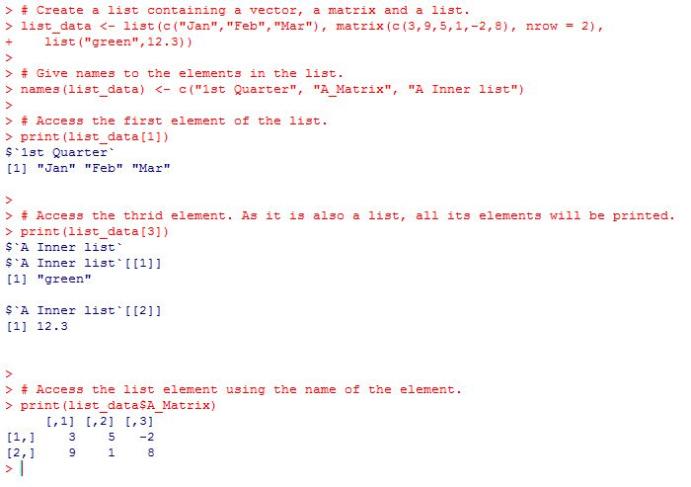
## Naming List Elements

The list elements can be given names and they can be accessed using these names.



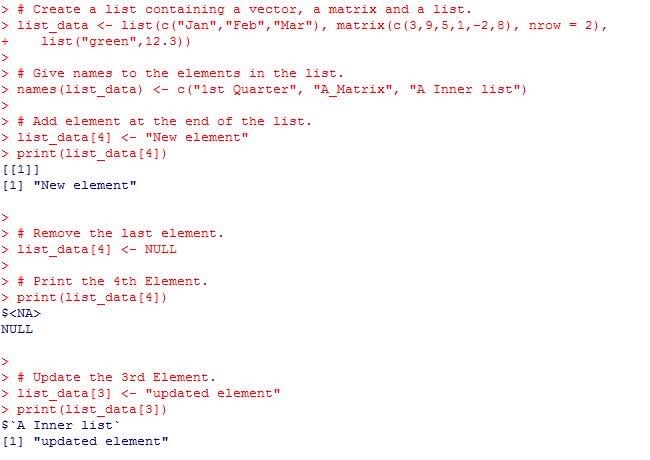
## Accessing List Elements

Elements of the list can be accessed by the index of the element in the list. In case of named lists it can also be accessed using the names.



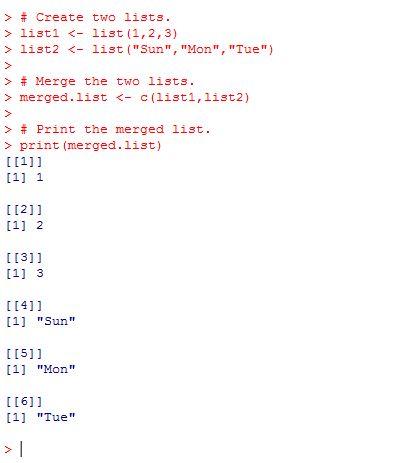
## Manipulating List Elements

We can add, delete and update list elements as shown below. We can add and delete elements only at the end of a list. But we can update any element.



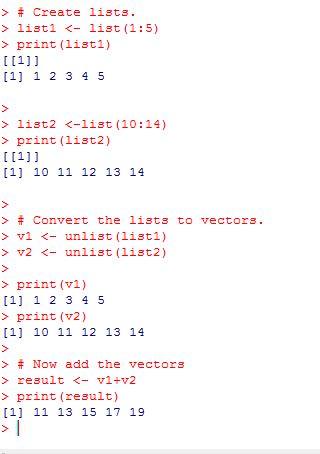
## Merging Lists

You can merge many lists into one list by placing all the lists inside one list() function.



* **Converting List to Vector**

A list can be converted to a vector so that the elements of the vector can be used for further manipulation. All the arithmetic operations on vectors can be applied after the list is converted into vectors. To do this conversion, we use the **unlist()** function. It takes the list as input and produces a vector.



**ARRAY**

## Naming Columns and Rows

We can give names to the rows, columns and matrices in the array by using the **dimnames** parameter.



## Check Available R Packages

* Get library locations containing R packages :libPaths()

F:\DWDM_R command\check available r packages.JPG

## Get the list of all the packages installed :library()

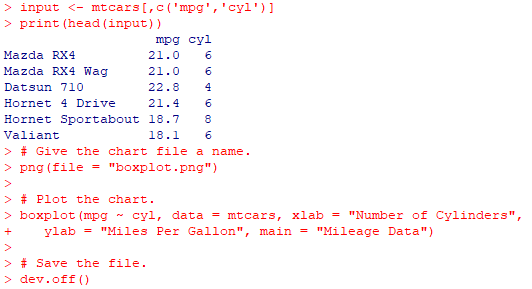
## F:\DWDM_R command\library in packages.JPG

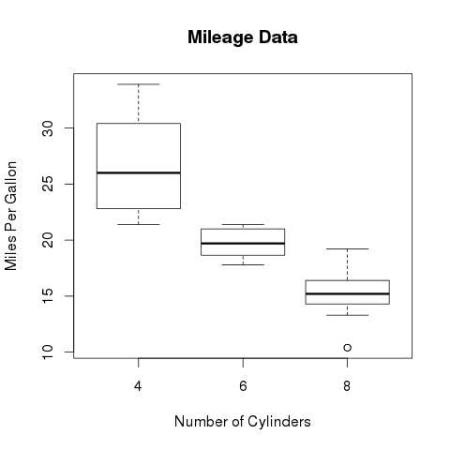
When we execute the above code, it produces the following result. It may vary depending on the local settings of your pc.

**R CHARTS & GRAPHS**

* **Boxplots** are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

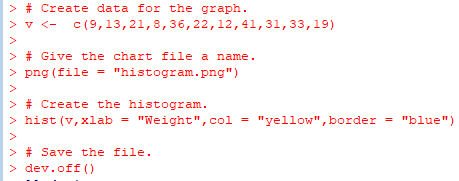
Boxplots are created in R by using the **boxplot()** function.

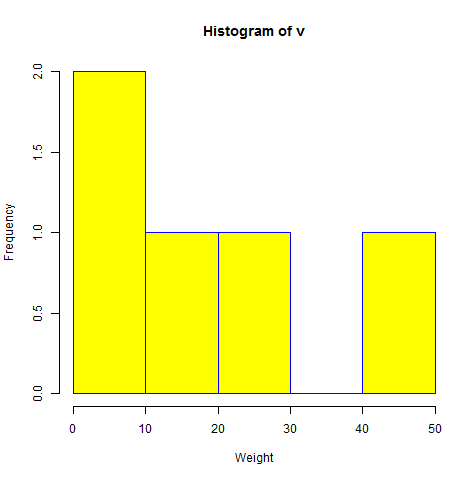




* A **histogram** represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.

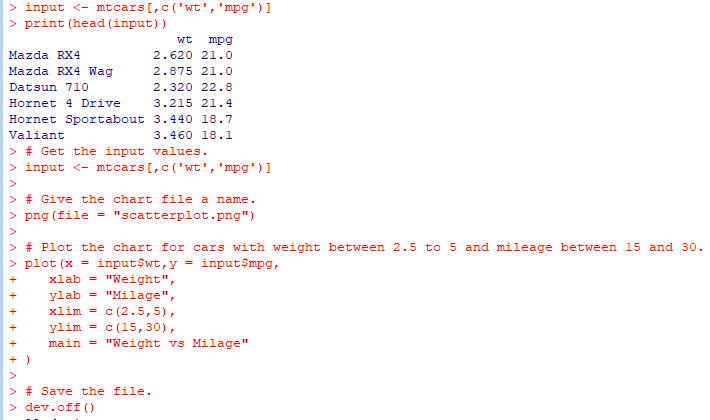
R creates histogram using **hist()** function. This function takes a vector as an input and uses some more parameters to plot histograms.





* **Scatterplots** show many points plotted in the Cartesian plane. Each point represents the values of two variables. One variable is chosen in the horizontal axis and another in the vertical axis.

The simple scatterplot is created using the **plot()** function.





**PRACTICAL - 4**

**Aim: Performing Association Rule Mining using R Programming.**

**Theory:**

**Association Rule Mining** (also called as Association Rule Learning) is a common technique used to find associations between many variables. It is often used by grocery stores, e-commerce websites, and anyone with large transactional databases. A most common example that we encounter in our daily lives — Amazon knows what else you want to buy when you order something on their site. The same idea extends to Spotify too — They know what song you want to listen to next. All of these incorporate, at some level, data mining concepts and association rule mining algorithms.

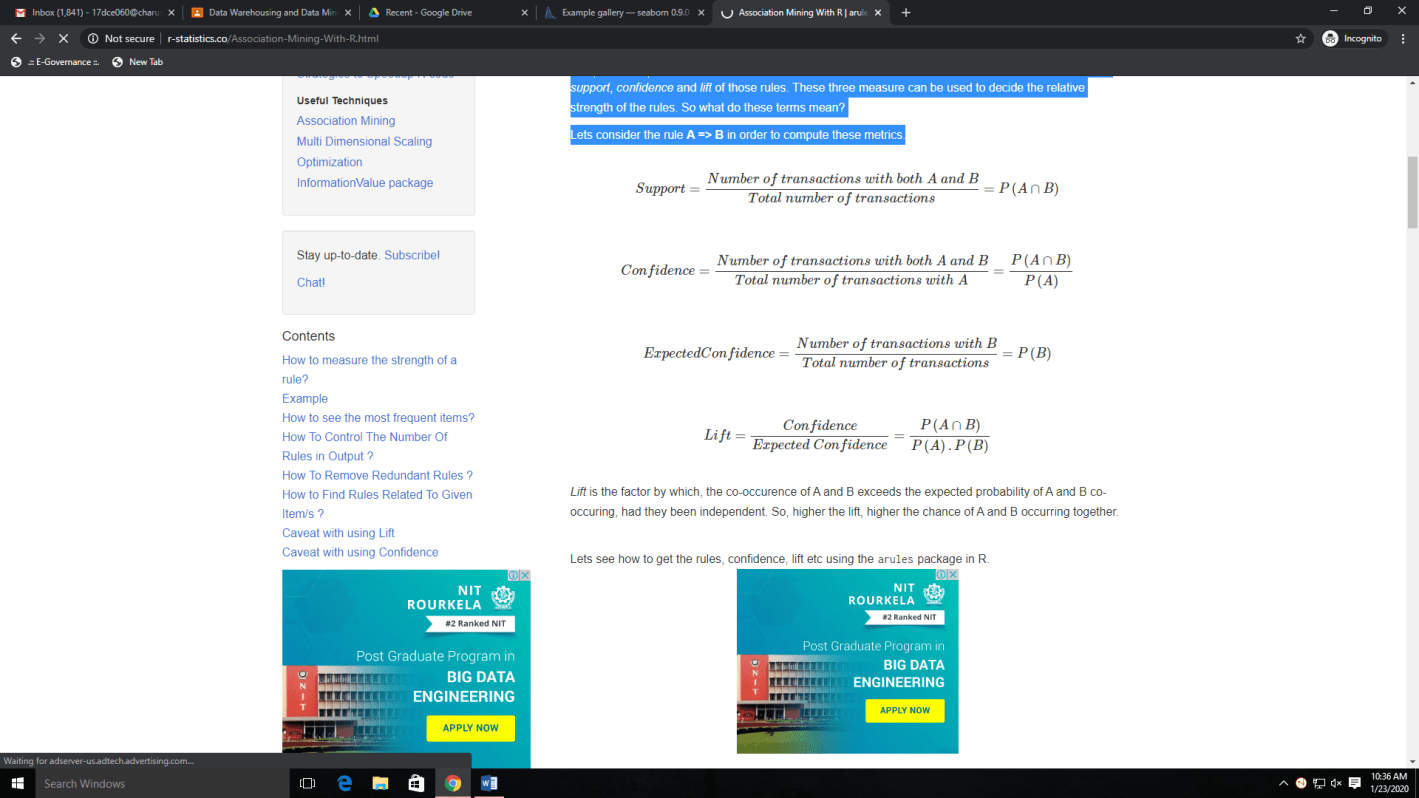
**Market Basket Analysis** is similar to ARM. Market Basket Analysis is a modelling technique based upon the theory that if you buy a certain group of items, you are more (or less) likely to buy another group of items. For example, if you are in an English pub and you buy a pint of beer and don’t buy a bar meal, you are more likely to buy crisps at the same time than somebody who didn’t buy beer.

There are three parameters controlling the number of rules to be generated *viz.* **Support and Confidence**. Another parameter **Lift** is generated using Support and Confidence and is one of the major parameters to filter the generated rules.

* ***Support*** is an indication of how frequently the itemset appears in the dataset. Consider only the two transactions from the above output. The support of the item *citrus fruit* is 1/2 as it appears in only 1 out of the two transactions.
* ***Confidence*** is an indication of how often the rule has been found to be true. We will discuss more about confidence after generating the rules.

The apriori() generates the most relevent set of rules from a given transaction data. It also shows the *support*, *confidence* and *lift* of those rules. These three measure can be used to decide the relative strength of the rules.?

* Lets consider the rule **A => B** in order to compute these metrics.



**Example**

Transactions data

Lets play with the Groceries data that comes with the arules pkg. Unlike dataframe, using head(Groceries) does not display the transaction items in the data. To view the transactions, use the inspect() function instead.

Since association mining deals with transactions, the data has to be converted to one of class transactions, made available in R through the arules pkg. This is a necessary step because the apriori() function accepts transactions data of class transactions only.

**library**(arules)

**class**(Groceries)

*#> [1] "transactions"*

*#>attr(,"package")*

*#> [1] "arules"*

**inspect**(**head**(Groceries, 3))

*#> items*

*#> 1 {citrus fruit,*

*#> semi-finished bread,*

*#> margarine,*

*#> ready soups}*

*#> 2 {tropical fruit,*

*#> yogurt,*

*#> coffee}*

*#> 3 {whole milk}*

tdata<- **read.transactions**("transactions\_data.txt", sep="\t")

tData<- **as** (myDataFrame, "transactions") *# convert to 'transactions' class*

**size**(**head**(Groceries)) *# number of items in each observation*

*#> [1] 4 3 1 4 4 5*

**LIST**(**head**(Groceries, 3)) *# convert 'transactions' to a list, note the LIST in CAPS*

*#> [[1]]*

*#> [1] "citrus fruit" "semi-finished bread" "margarine"*

*#> [4] "ready soups"*

*#>*

*#> [[2]]*

*#> [1] "tropical fruit" "yogurt" "coffee"*

*#>*

*#> [[3]]*

*#> [1] "whole milk"*

frequentItems<- **eclat** (Groceries, parameter = **list**(supp = 0.07, maxlen = 15)) *# calculates support for frequent items*

**inspect**(frequentItems)

*#> items support*

*#>1 {other vegetables,whole milk} 0.07483477*

*#>2 {whole milk} 0.25551601*

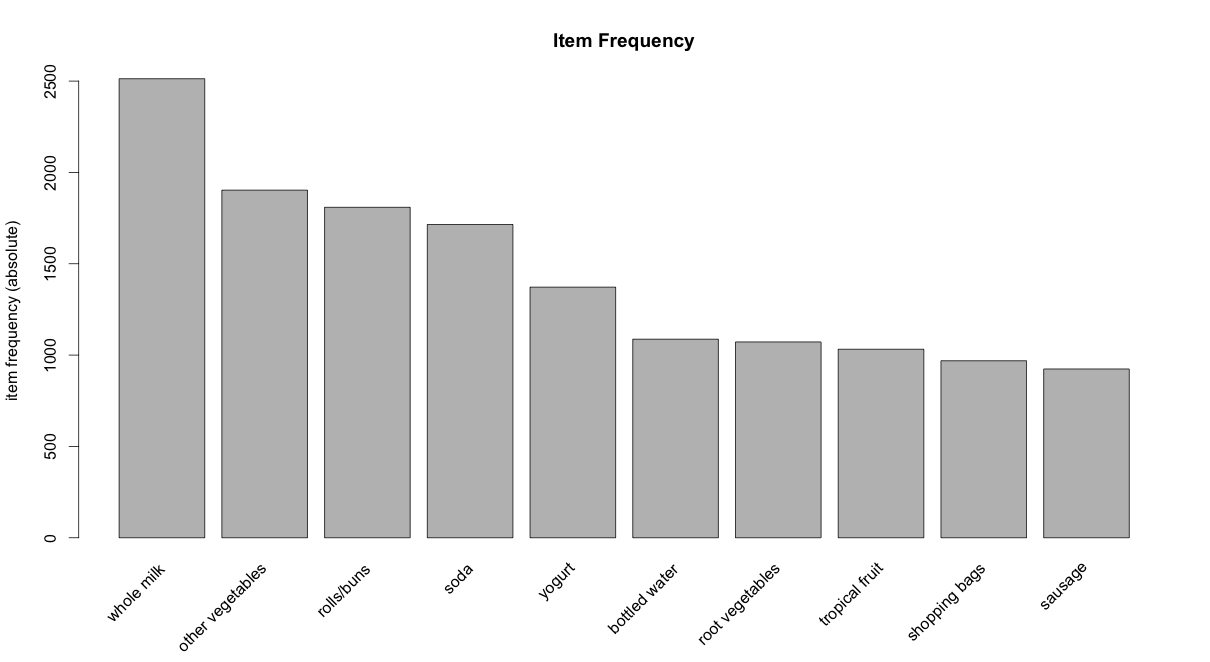
*#>3 {other vegetables} 0.19349263*

*#>4 {rolls/buns} 0.18393493*

*#>5 {yogurt} 0.13950178*

*#>6 {soda} 0.17437722*

**itemFrequencyPlot**(Groceries, topN=10, type="absolute", main="Item Frequency") *# plot frequent items*



rules <- **apriori** (Groceries, parameter = **list**(supp = 0.001, conf = 0.5)) *# Min Support as 0.001, confidence as 0.8.*

rules\_conf<- **sort** (rules, by="confidence", decreasing=TRUE) *# 'high-confidence' rules.*

**inspect**(**head**(rules\_conf)) *# show the support, lift and confidence for all rules*

*#>lhsrhs support confidence lift*

*#>113 {rice,sugar} => {whole milk} 0.001220132 1 3.913649*

*#>258 {canned fish,hygiene articles} => {whole milk} 0.001118454 1 3.913649*

*#> 1487 {root vegetables,butter,rice} => {whole milk} 0.001016777 1 3.913649*

*#> 1646 {root vegetables,whipped/sour cream,flour} => {whole milk} 0.001728521 1 3.913649*

*#> 1670 {butter,softcheese,domestic eggs} => {whole milk} 0.001016777 1 3.913649*

*#> 1699 {citrus fruit,rootvegetables,soft cheese} => {other vegetables} 0.001016777 1 5.168156*

rules\_lift<- **sort** (rules, by="lift", decreasing=TRUE) *# 'high-lift' rules.*

**inspect**(**head**(rules\_lift)) *# show the support, lift and confidence for all rules*

*#>lhsrhssupport confidence lift*

*#> 53 {Instant food products,soda} => {hamburger meat} 0.001220 0.6315789 18.995*

*#> 37 {soda,popcorn} => {salty snack} 0.001220 0.6315789 16.697*

*#>444 {flour,baking powder} => {sugar} 0.001016 0.5555556 16.408*

*#>327 {ham,processed cheese} => {white bread} 0.001931 0.6333333 15.045*

*#> 55 {whole milk,Instant food products} => {hamburger meat} 0.001525 0.5000000 15.038*

*#> 4807 {other vegetables,curd,yogurt,whipped/sour cream} => {cream cheese } 0.001016 0.5882353 14.834*

The rules with confidence of 1 (see rules\_conf above) imply that, whenever the LHS item was purchased, the RHS item was also purchased 100% of the time.

A rule with a lift of 18 (see rules\_lift above) imply that, the items in LHS and RHS are 18 times more likely to be purchased together compared to the purchases when they are assumed to be unrelated.

Adjust the maxlen, supp and conf arguments in the apriori function to control the number of rules generated. You will have to adjust this based on the sparesness of you data.

rules <- **apriori**(Groceries, parameter = **list** (supp = 0.001, conf = 0.5, maxlen=3)) *# maxlen = 3 limits the elements in a rule to 3*

1. To get **‘strong‘** rules, increase the value of *‘conf’* parameter.
2. To get **‘longer‘** rules, increase *‘maxlen’*.

Sometimes it is desirable to remove the rules that are subset of larger rules. To do so, use the below code to filter the redundant rules.

subsetRules<- **which**(**colSums**(**is.subset**(rules, rules)) > 1) *# get subset rules in vector*

**length**(subsetRules) *#> 3913*

rules <- rules[-subsetRules] *# remove subset rules.*

This can be achieved by modifying the appearance parameter in the apriori() function. For example,

To find what factors influenced purchase of product X

To find out what customers had purchased before buying ‘Whole Milk’. This will help you understand the patterns that led to the purchase of ‘whole milk’.

rules <- **apriori** (data=Groceries, parameter=**list** (supp=0.001,conf = 0.08), appearance = **list** (default="lhs",rhs="whole milk"), control = **list** (verbose=F)) *# get rules that lead to buying 'whole milk'*

rules\_conf<- **sort** (rules, by="confidence", decreasing=TRUE) *# 'high-confidence' rules.*

**inspect**(**head**(rules\_conf))

*#>lhsrhs support confidence lift*

*#>196 {rice,sugar} => {whole milk} 0.001220132 1 3.913649*

*#>323 {canned fish,hygiene articles} => {whole milk} 0.001118454 1 3.913649*

*#> 1643 {root vegetables,butter,rice} => {whole milk} 0.001016777 1 3.913649*

*#> 1705 {root vegetables,whipped/sour cream,flour} => {whole milk} 0.001728521 1 3.913649*

*#> 1716 {butter,softcheese,domestic eggs} => {whole milk} 0.001016777 1 3.913649*

*#> 1985 {pip fruit,butter,hygiene articles} => {whole milk} 0.001016777 1 3.913649*

To find out what products were purchased after/along with product X

The is a case to find out the Customers who bought ‘Whole Milk’ also bought . . In the equation, ‘whole milk’ is in LHS (left hand side).

rules <- **apriori** (data=Groceries, parameter=**list** (supp=0.001,conf = 0.15,minlen=2), appearance = **list**(default="rhs",lhs="whole milk"), control = **list** (verbose=F)) *# those who bought 'milk' also bought..*

rules\_conf<- **sort** (rules, by="confidence", decreasing=TRUE) *# 'high-confidence' rules.*

**inspect**(**head**(rules\_conf))

*#>lhsrhs support confidence lift*

*#> 6 {whole milk} => {other vegetables} 0.07483477 0.2928770 1.5136341*

*#> 5 {whole milk} => {rolls/buns} 0.05663447 0.2216474 1.2050318*

*#> 4 {whole milk} => {yogurt} 0.05602440 0.2192598 1.5717351*

*#> 2 {whole milk} => {root vegetables} 0.04890696 0.1914047 1.7560310*

*#> 1 {whole milk} => {tropical fruit} 0.04229792 0.1655392 1.5775950*

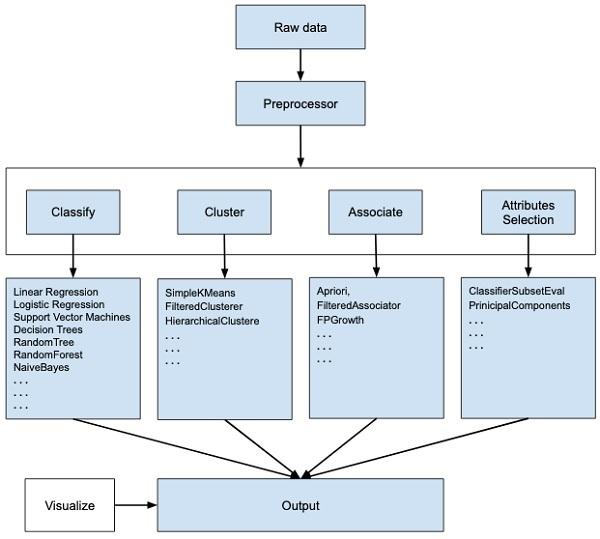
*#> 3 {whole milk} => {soda} 0.04006101 0.1567847 0.8991124*

**PRACTICAL - 5**

**Aim: Perform Different Data Mining Activities using Weka Explorer Tool (Open Source Data Mining Tool) & Experimental Tool (Open Source Data Mining Tool).**

**Theory:**

WEKA - an open source software provides tools for data preprocessing, implementation of several Machine Learning algorithms, and visualization tools so that you can develop machine learning techniques and apply them to real-world data mining problems.

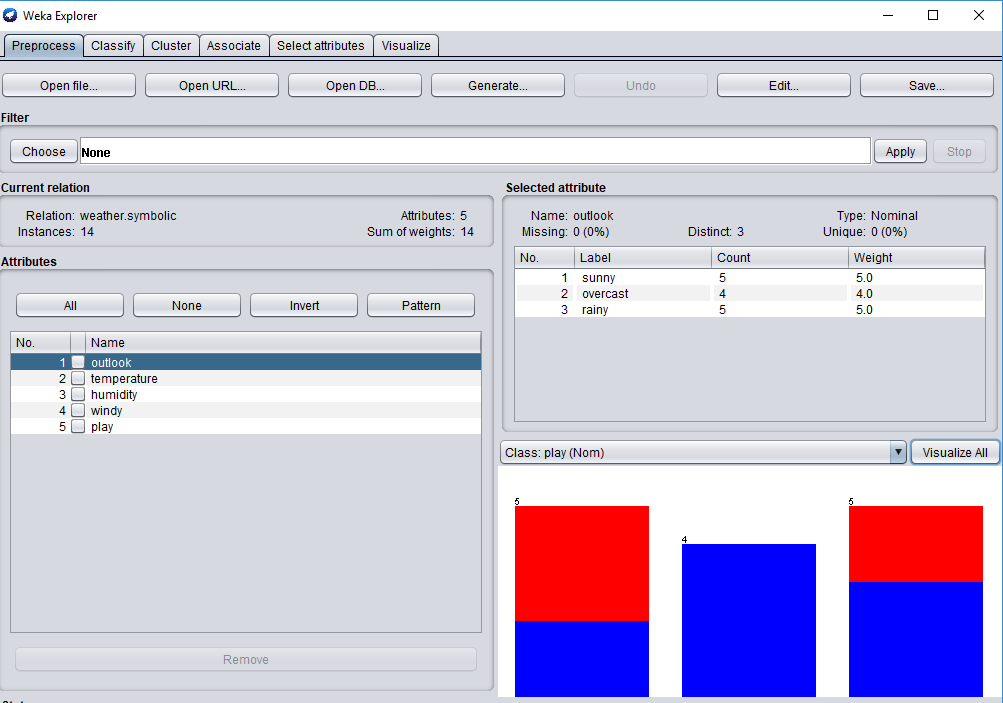


* If you observe the beginning of the flow of the image, you will understand that there are many stages in dealing with Big Data to make it suitable for machine learning −
* First, you will start with the raw data collected from the field. This data may contain several null values and irrelevant fields. You use the data preprocessing tools provided in WEKA to cleanse the data.
* Then, you would save the preprocessed data in your local storage for applying ML algorithms.
* Next, depending on the kind of ML model that you are trying to develop you would select one of the options such as **Classify, Cluster**, or **Associate**. The **Attributes Selection** allows the automatic selection of features to create a reduced dataset.
* Note that under each category, WEKA provides the implementation of several algorithms. You would select an algorithm of your choice, set the desired parameters and run it on the dataset.
* Then, WEKA would give you the statistical output of the model processing. It provides you a visualization tool to inspect the data.
* The various models can be applied on the same dataset. You can then compare the outputs of different models and select the best that meets your purpose.
* Thus, the use of WEKA results in a quicker development of machine learning models on the whole.

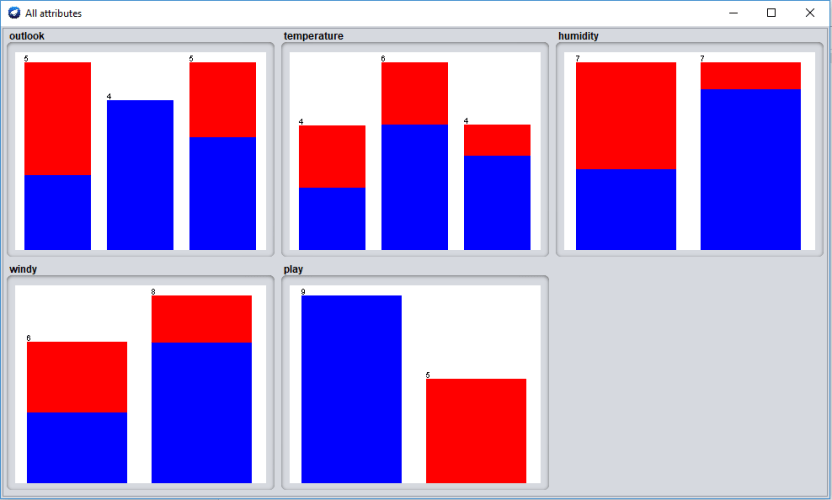
**Output/Working:**

Data: weather\_nominal.arff

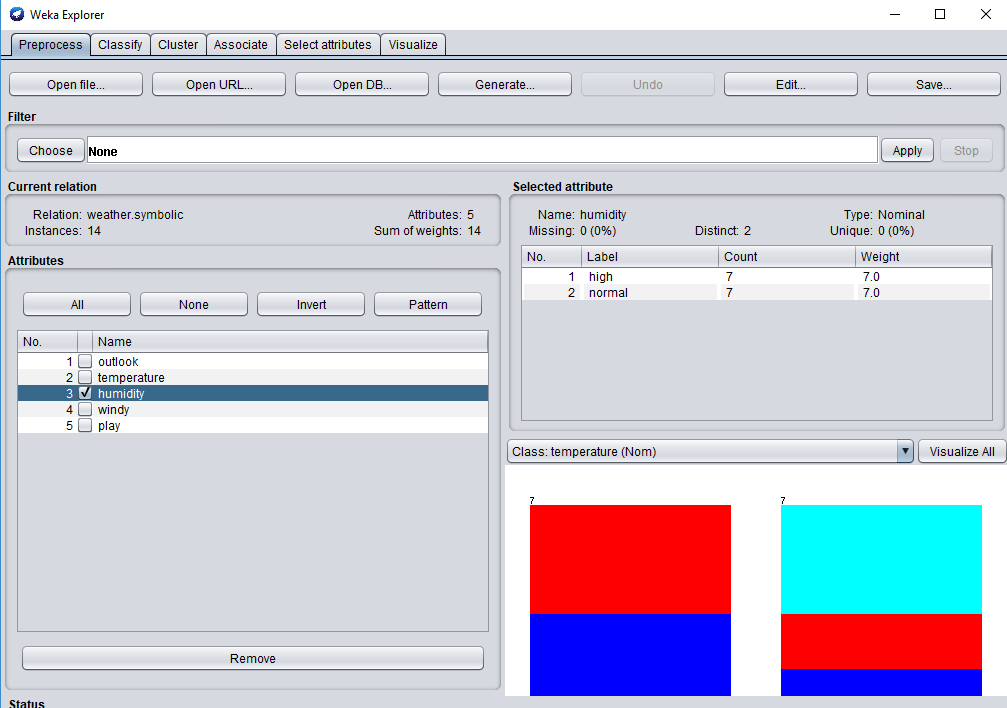
Preprocessing

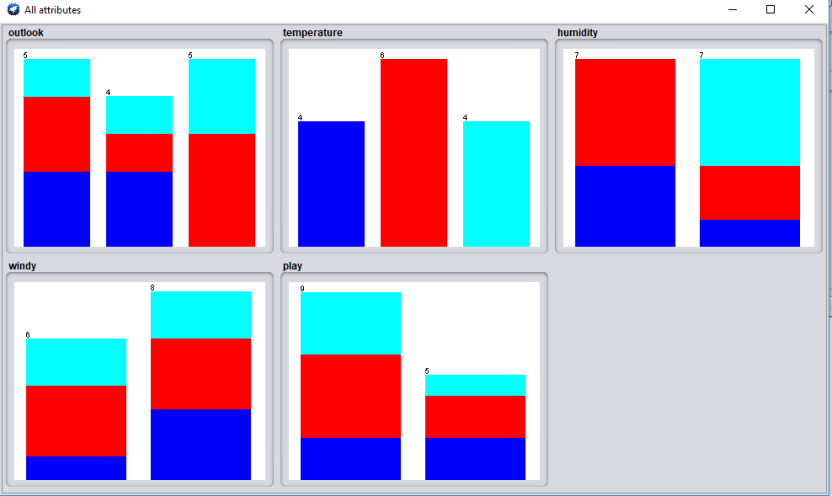


Statistics:



Attribute Selection:





**PRACTICAL - 6**

**Aim: Predicting Sports Winners with Decision Trees.**

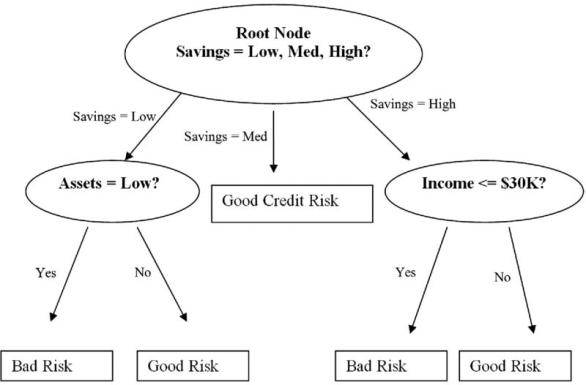
**Theory:**

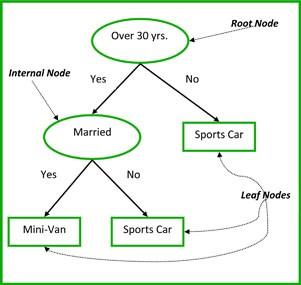
**Introduction to Decision Trees**

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Tree based learning algorithms are considered to be one of the best and mostly used supervised learning methods. Tree based methods empower predictive models with high accuracy, stability and ease of interpretation. Unlike linear models, they map non-linear relationships quite well. They are adaptable at solving any kind of problem at hand (classification or regression). Decision Tree algorithms are referred to as CART (Classification and Regression Trees).

Methods like decision trees, random forest, gradient boosting are being popularly used in all kinds of data science problems.





**Common terms used with Decision trees:**

1. **Root Node:**It represents entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:**It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:**When a sub-node splits into further sub-nodes, then it is called decision node.
4. **Leaf/ Terminal Node:**Nodes do not split is called Leaf or Terminal node.
5. **Pruning**: When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
6. **Branch / Sub-Tree:**A sub section of entire tree is called branch or sub-tree.
7. **Parent and Child Node:**A node, which is divided into sub-nodes is called parent node of sub-nodes whereas sub-nodes are the child of parent node.



As a result, the decision making tree is one of the more popular classification algorithms being used in Data Mining and Machine Learning. Example applications include:

· Evaluation of brand expansion opportunities for a business using historical sales data

· Determination of likely buyers of a product using demographic data to enable targeting of limited advertisement budget

· Prediction of likelihood of default for applicant borrowers using predictive models generated from historical data

· Help with prioritization of emergency room patient treatment using a predictive model based on factors such as age, blood pressure, gender, location and severity of pain, and other measurements

· Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal.

Because of their simplicity, tree diagrams have been used in a broad range of industries and disciplines including civil planning, energy, financial, engineering, healthcare, pharmaceutical, education, law, and business.

**How does Decision Tree works ?**

Decision tree is a type of supervised learning algorithm (having a pre-defined target variable) that is mostly used in classification problems. It works for both categorical and continuous input and output variables. In this technique, we split the population or sample into two or more homogeneous sets (or sub-populations) based on most significant splitter / differentiator in input variables.



**Types of Decision Trees**

1. **Categorical Variable Decision Tree:** Decision Tree which has categorical target variable then it called as categorical variable decision tree. E.g.:- In above scenario of student problem, where the target variable was “Student will play cricket or not” i.e. YES or NO.
2. **Continuous Variable Decision Tree:** Decision Tree has continuous target variable then it is called as Continuous Variable Decision Tree.

The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label.

Place the best attribute of the dataset at the root of the tree.

Split the training set into subsets. Subsets should be made in such a way that each subset contains data with the same value for an attribute.

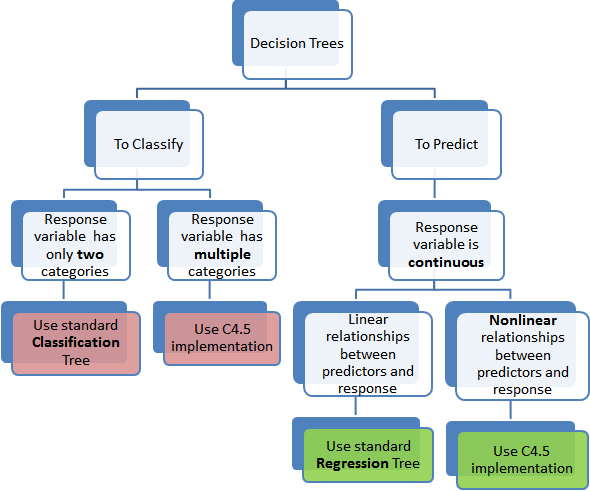
Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

**Advantages:**

1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also does NOT affect the process of building decision tree to any considerable extent.
5. A Decision trees model is very intuitive and easy to explain to technical teams as well as stakeholders.

**Disadvantage:**

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. For a Decision tree sometimes calculation can go far more complex compared to other algorithms.
3. Decision tree often involves higher time to train the model.
4. Decision tree training is relatively expensive as complexity and time taken is more.
5. Decision Tree algorithm is inadequate for applying regression and predicting continuous values.



Here in this practical we will look at predicting the winner of games of the National Basketball Association (NBA) using a different type of classification algorithm—decision trees.

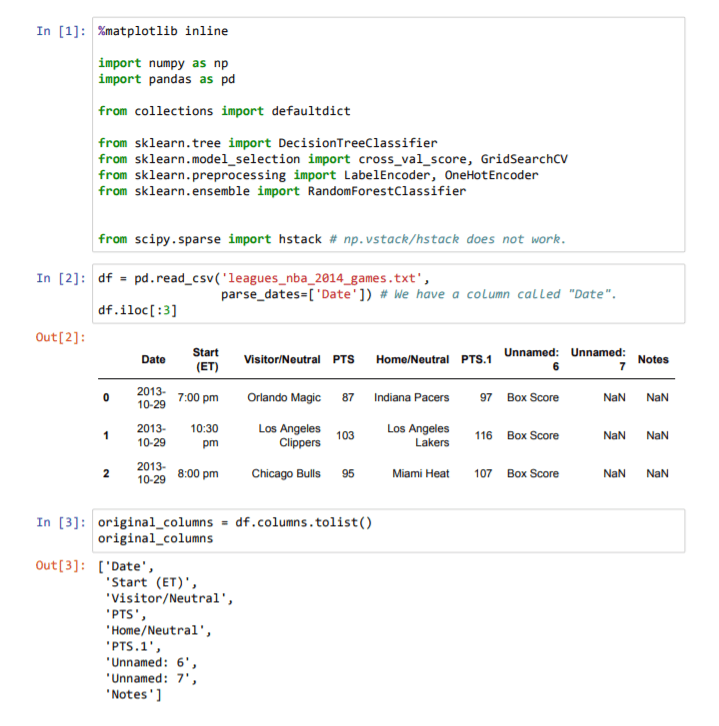
**Collecting the data:**

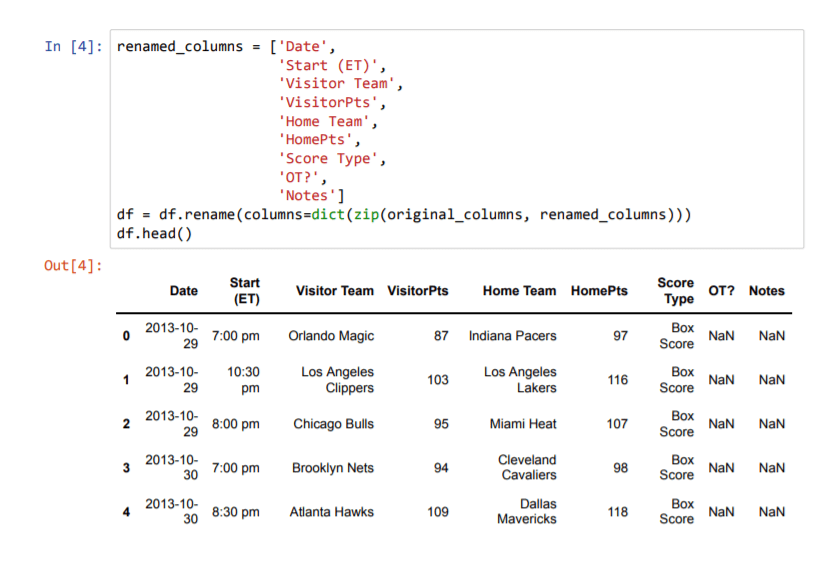
The data we will be using is the match history data for the NBA, for the 2013-2014 season. The [Basketball-Reference.com](http://www.basketball-reference.com/) website contains a significant number of resources and statistics collected from the NBA and other leagues. Perform the following steps to download the dataset:

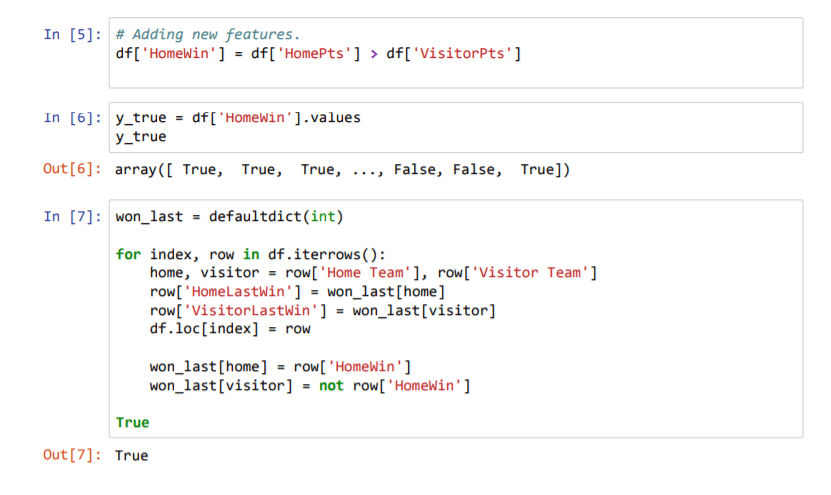
1. Navigate to <http://www.basketball-reference.com/leagues/NBA_2014_games.html> in your web browser.
2. Click on the Export button next to the Regular Season heading.
3. Download the file to your data folder (and make a note of the path).

We will load the file with the pandas library, which is an incredibly useful library for manipulating data. [Python](https://subscribe.packtpub.com/learn-python/) also contains a built-in library called *csv* that supports reading and writing CSV files. We will use pandas instead as it provides more powerful functions to work with datasets.

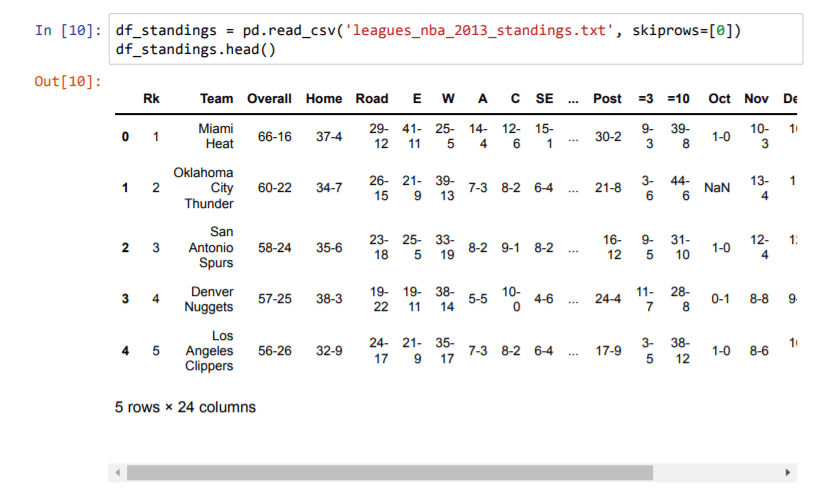
**Working:**



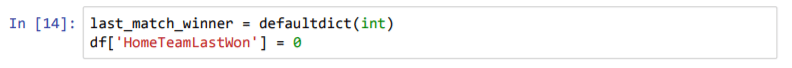




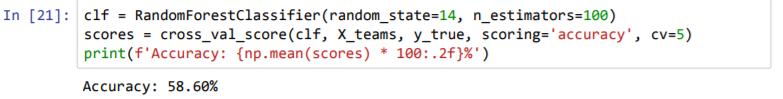








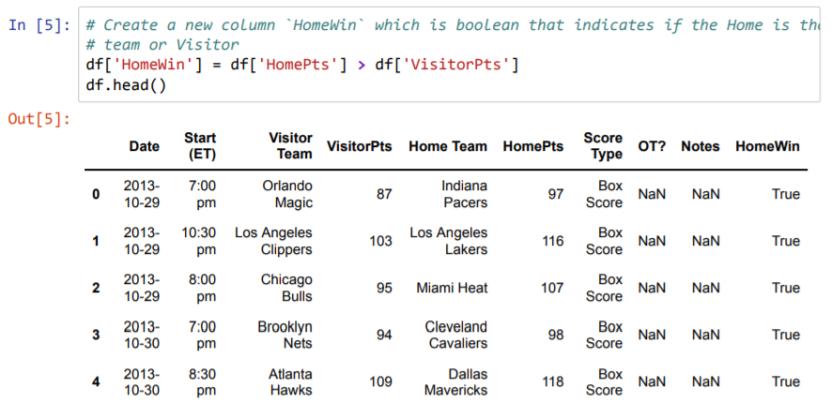


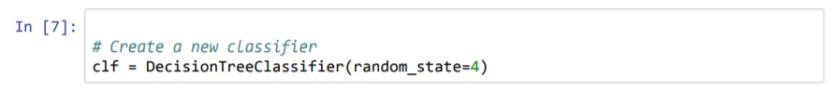


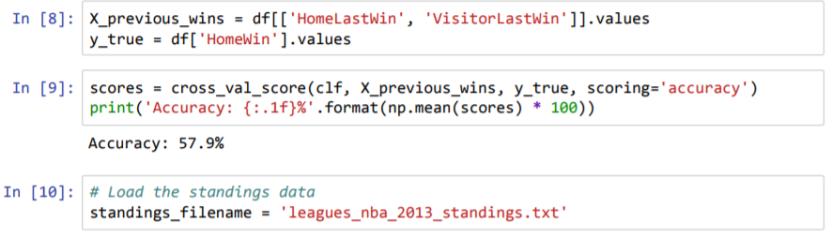


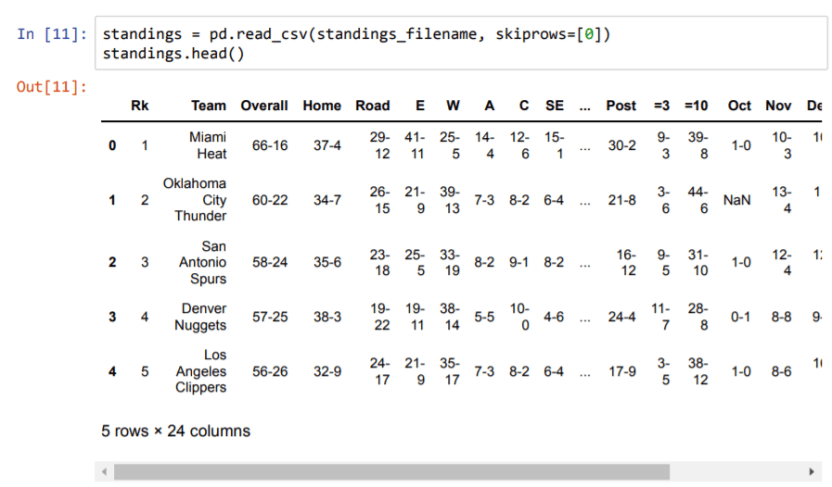




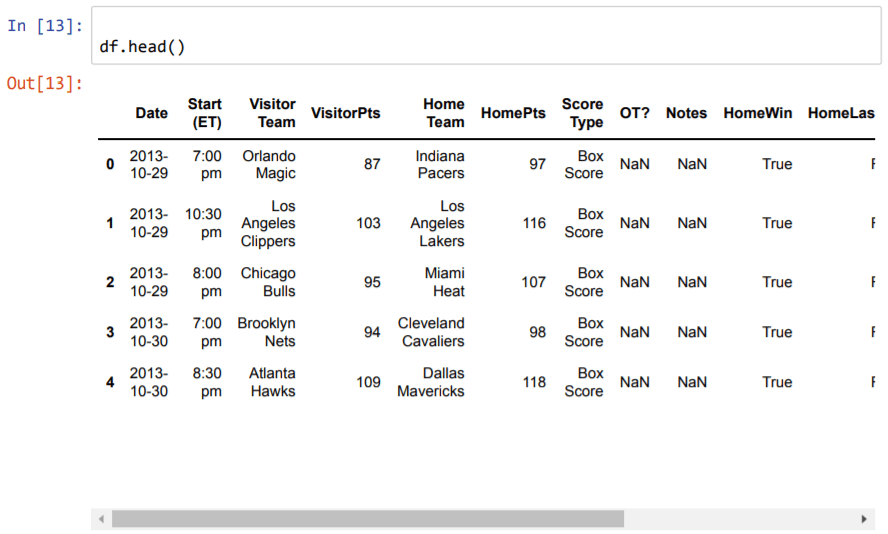




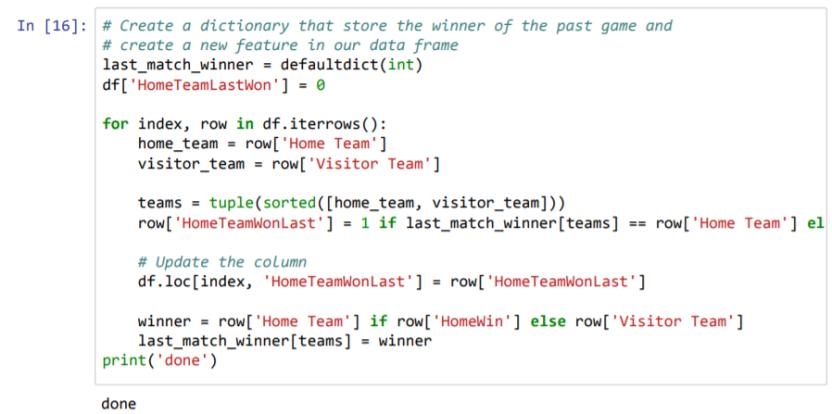


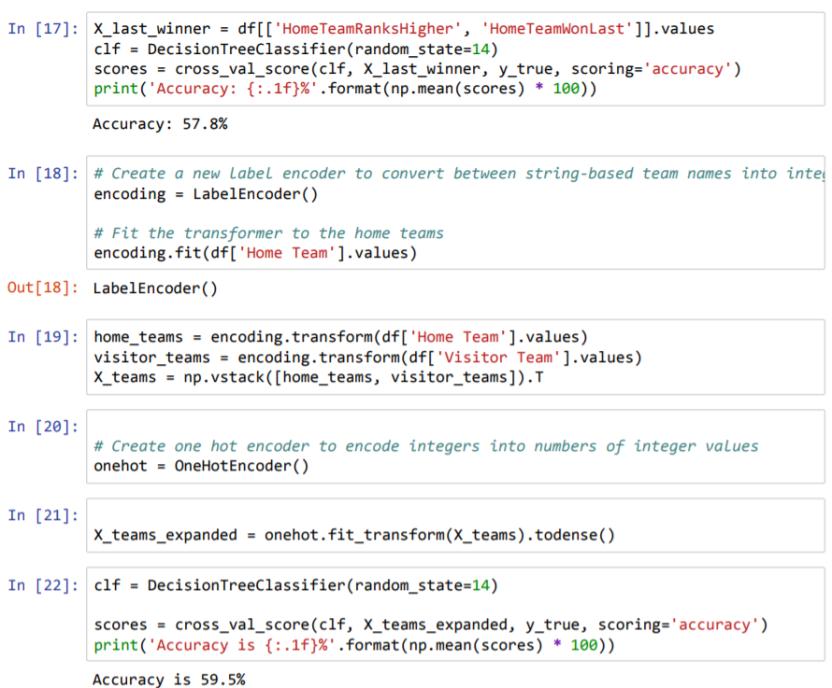


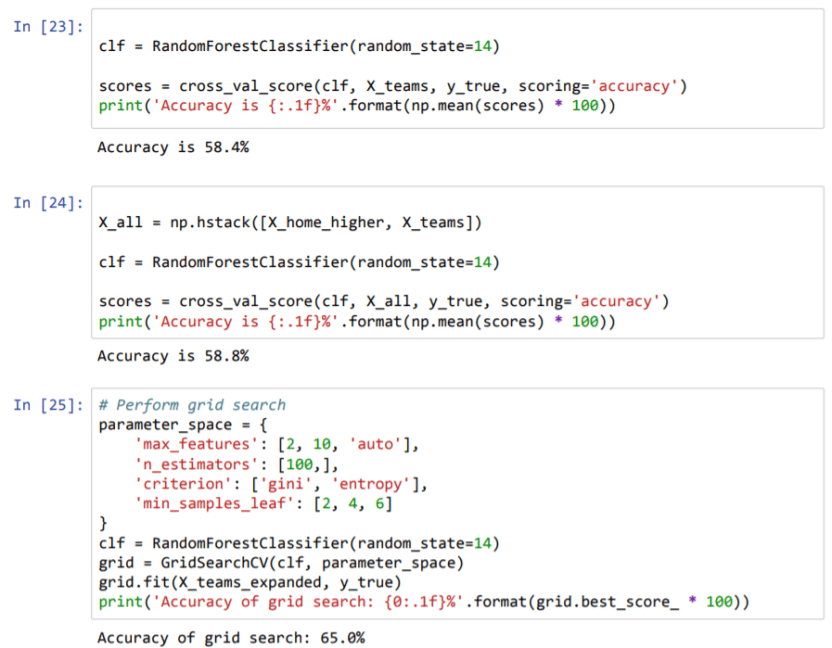


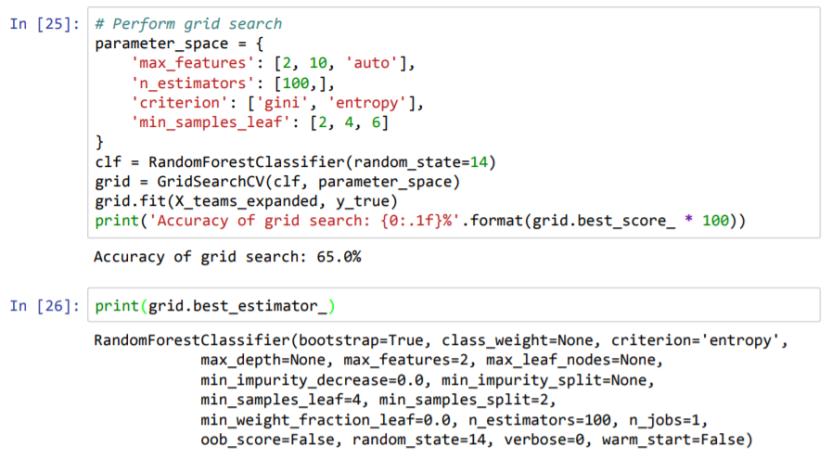












**PRACTICAL - 7**

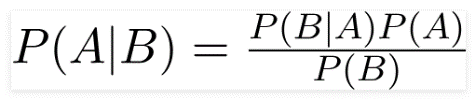
**Aim: Text Classification using Naïve Bayes, SVM and Evaluating Classification Models.**

**Theory:**

# **Principle of Naive Bayes Classifier:**

A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem.

# **Bayes Theorem:**



Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

# **Types of Naive Bayes Classifier:**

## Multinomial Naive Bayes:

This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

## Bernoulli Naive Bayes:

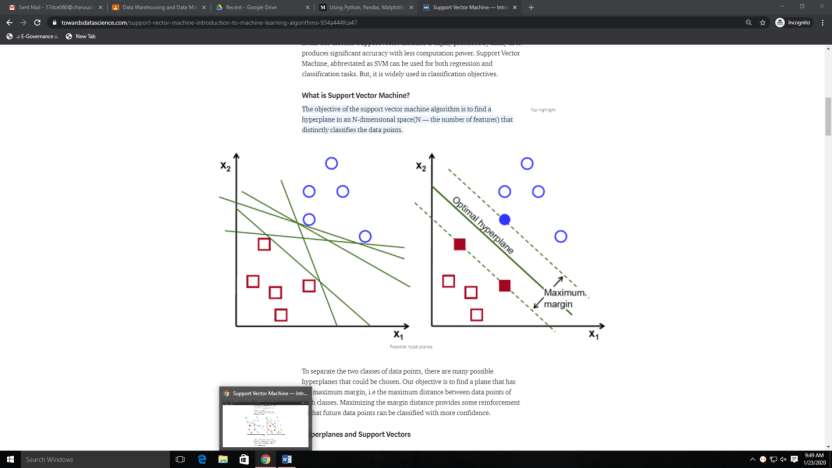
This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

## Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.

## Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.



## SVM Implementation in Python

The dataset we will be using to implement our SVM algorithm is the Iris dataset.

import pandas as pd

df = pd.read\_csv('/Users/rohith/Documents/Datasets/Iris\_dataset/iris.csv')

df = df.drop(['Id'],axis=1)

target = df['Species']

s = set()

for val in target:

s.add(val)

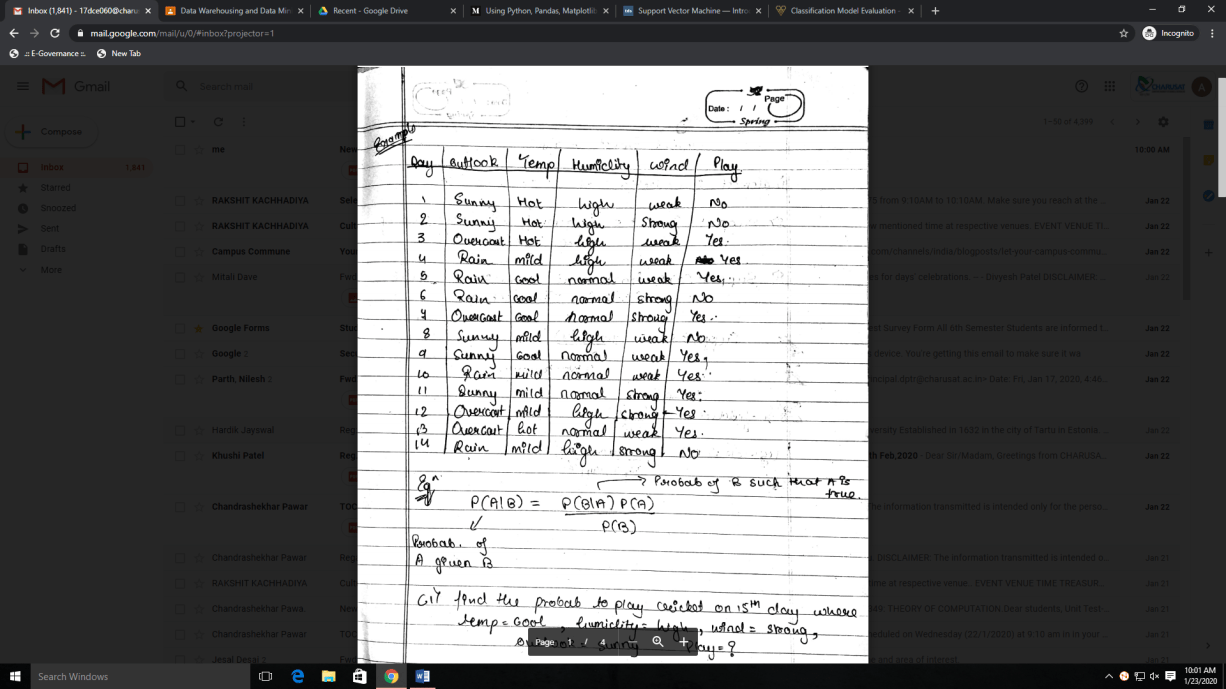
s = list(s)

rows = list(range(100,150))

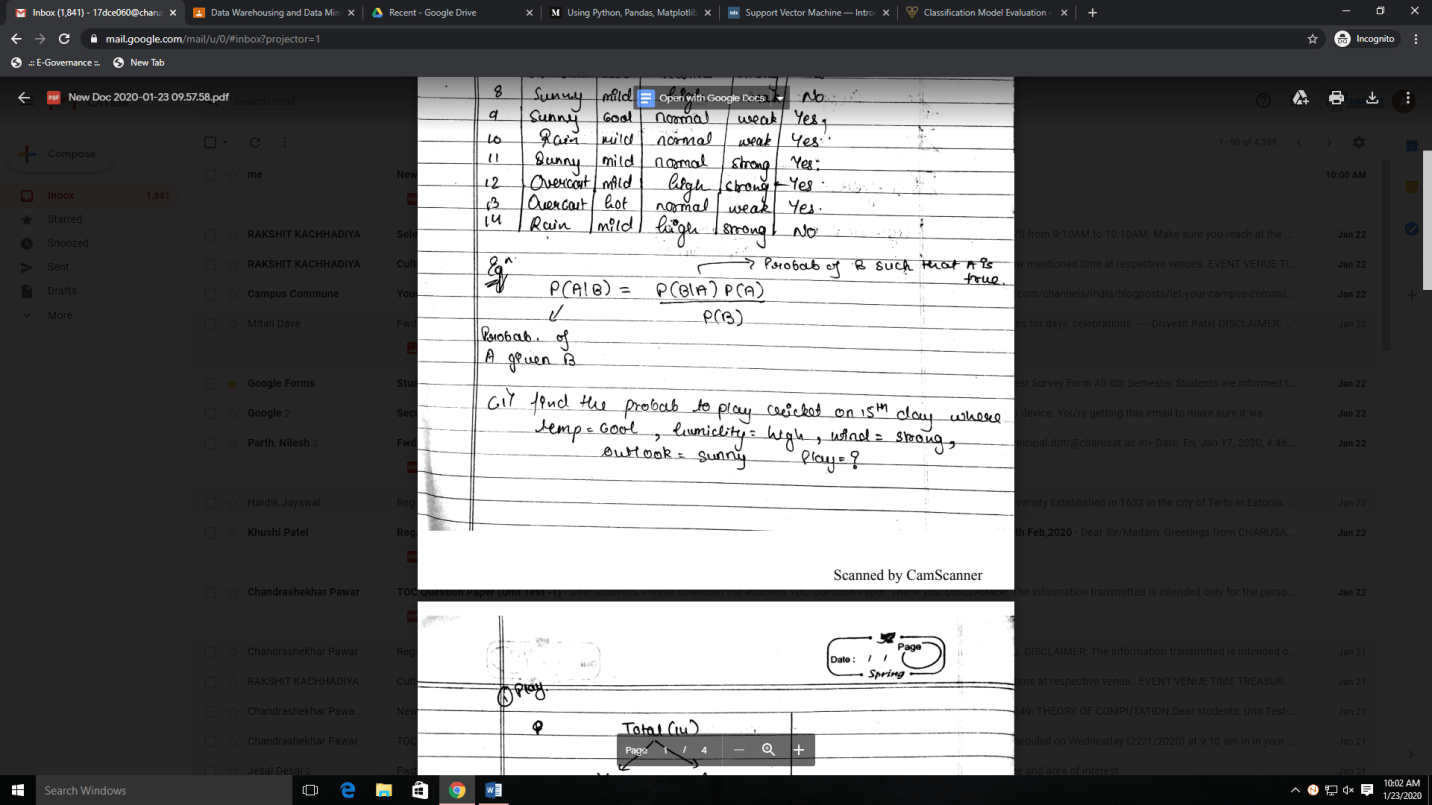
df = df.drop(df.index[rows])

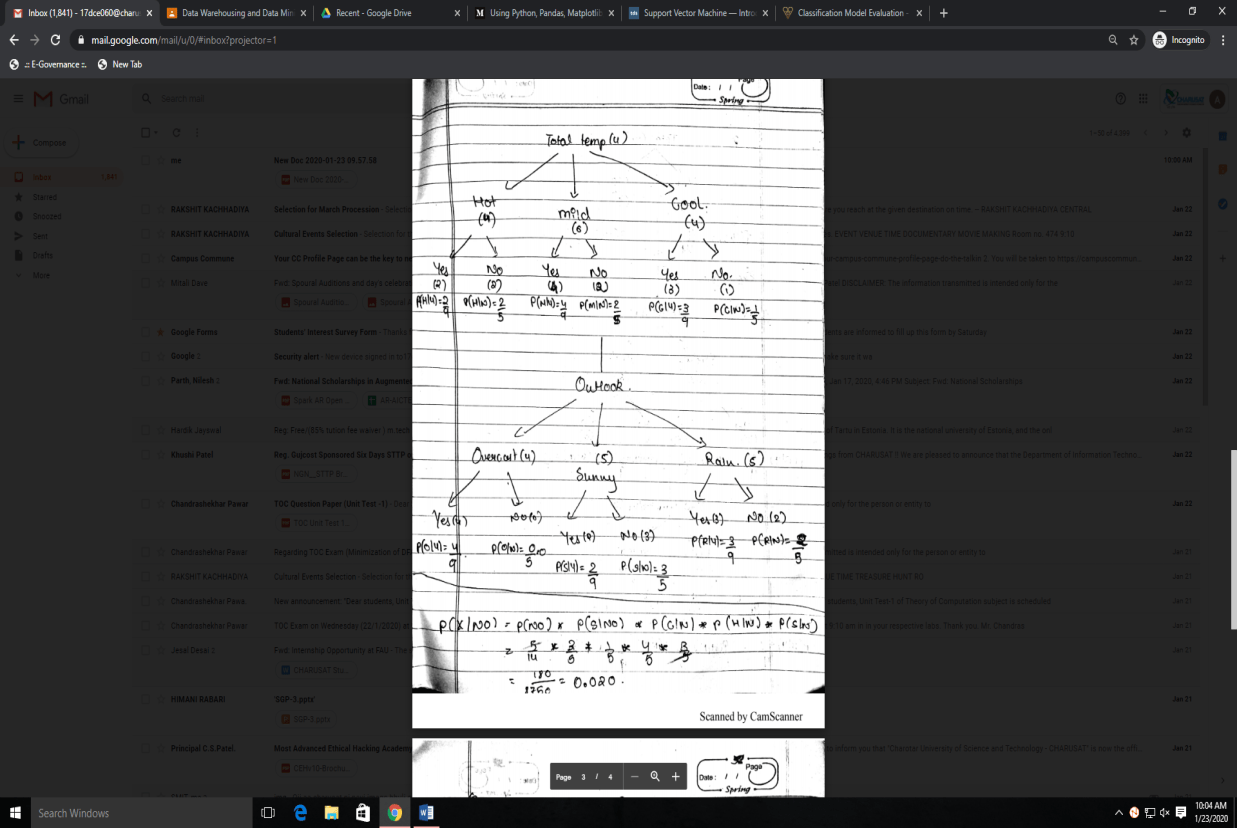
**Program Working:**

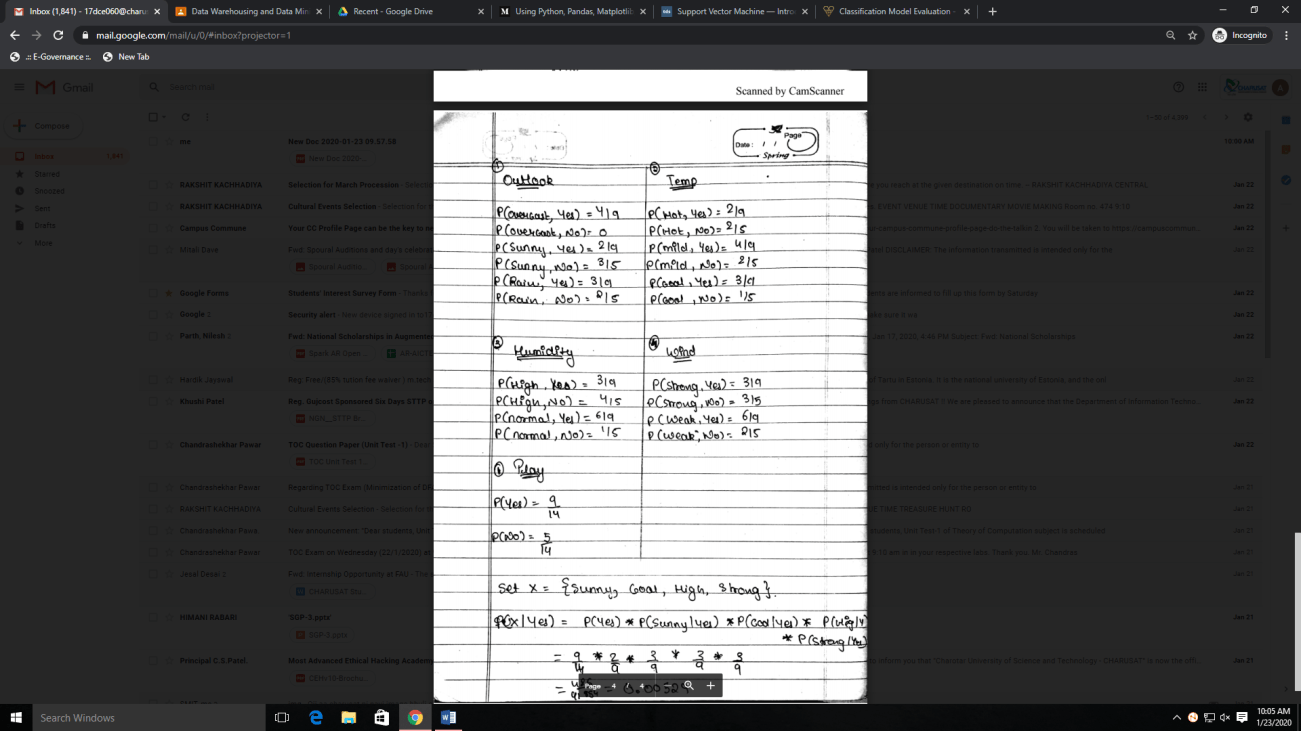
**Dataset :**



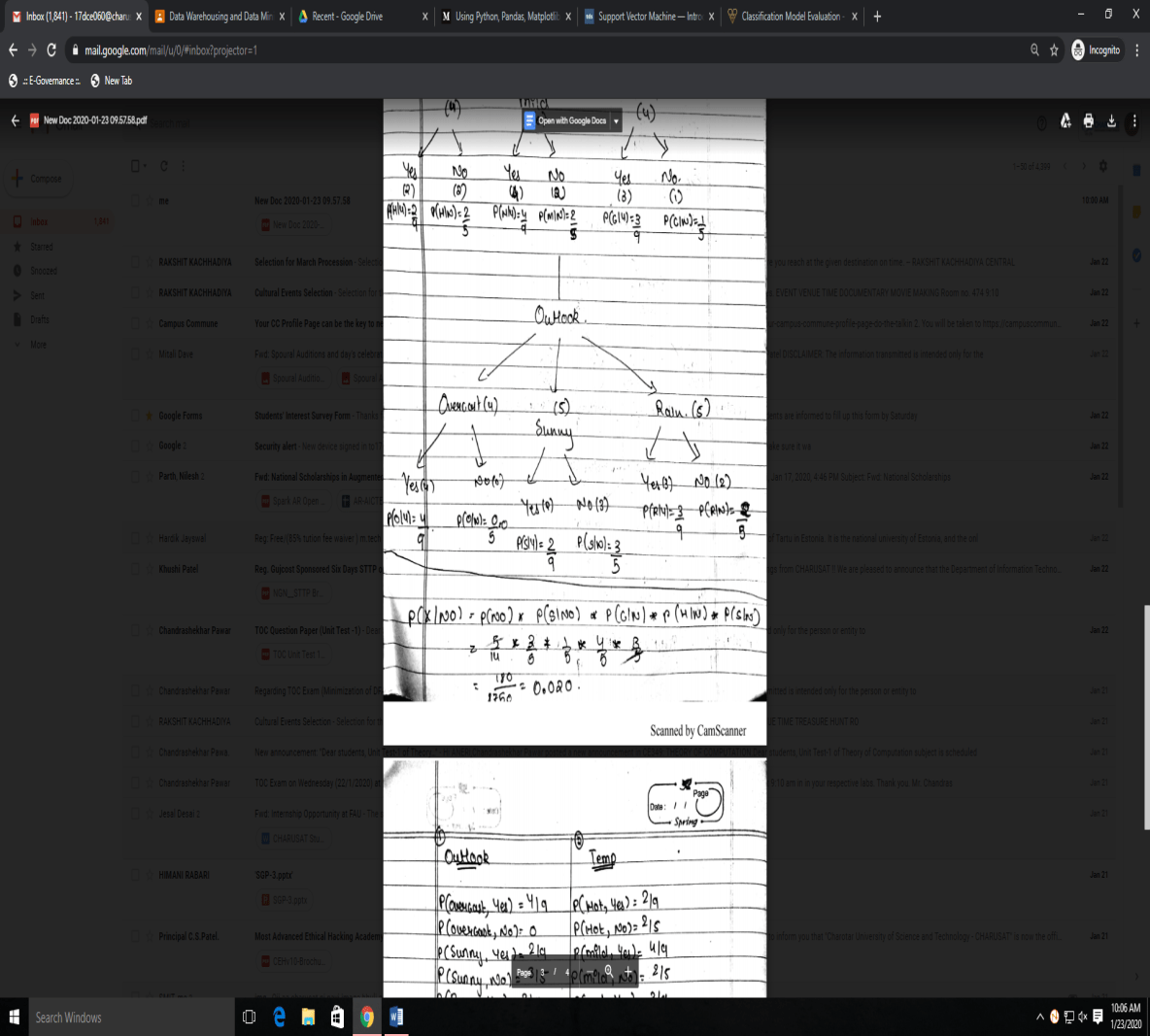
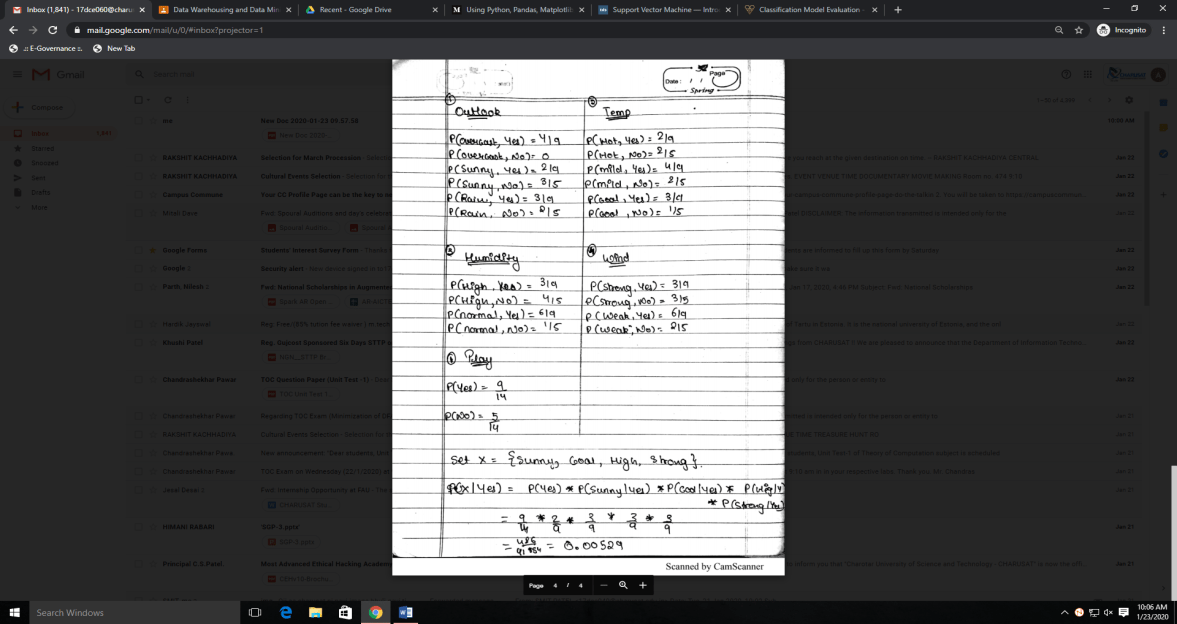
**Question:**

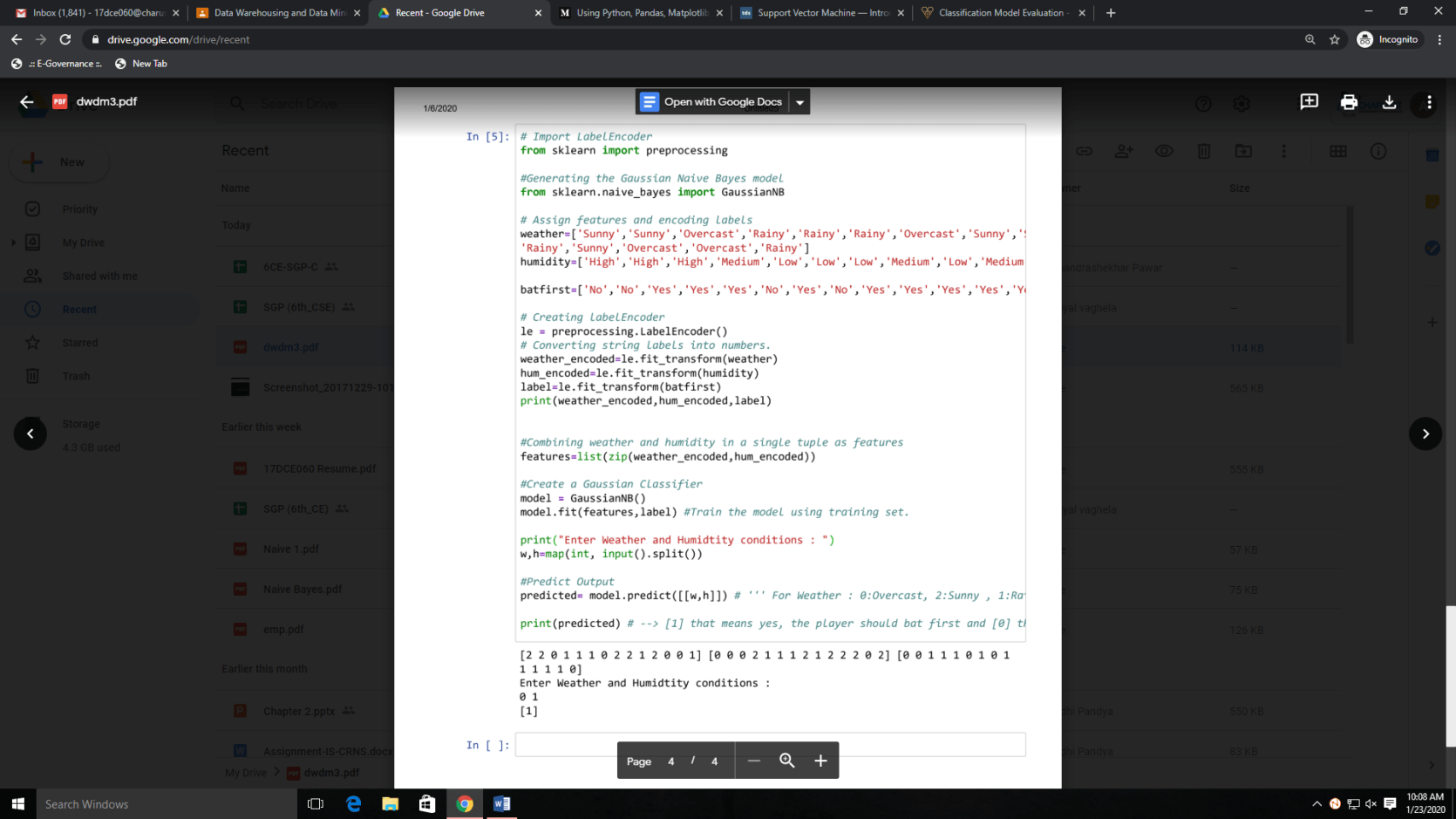


**Solution:**

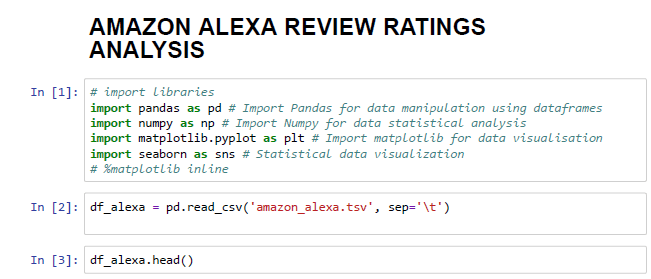


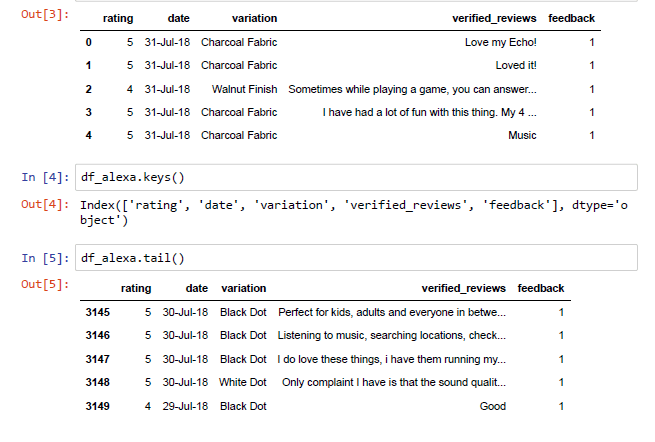
**Calculation:**

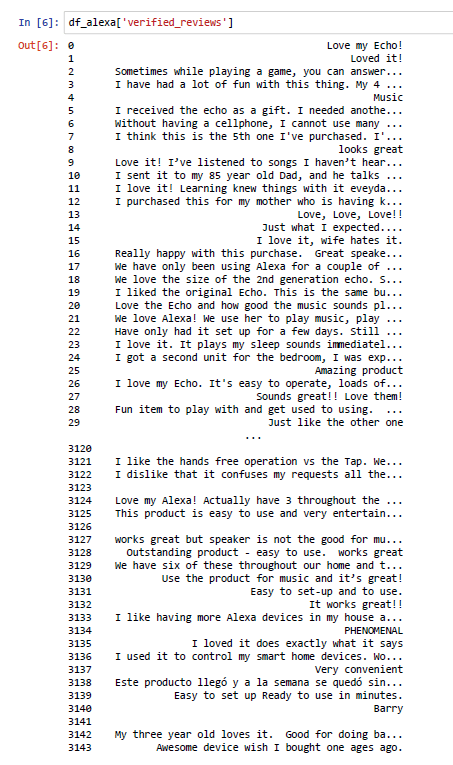


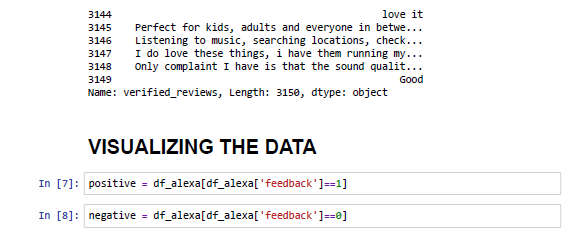
**Using Pandas:**

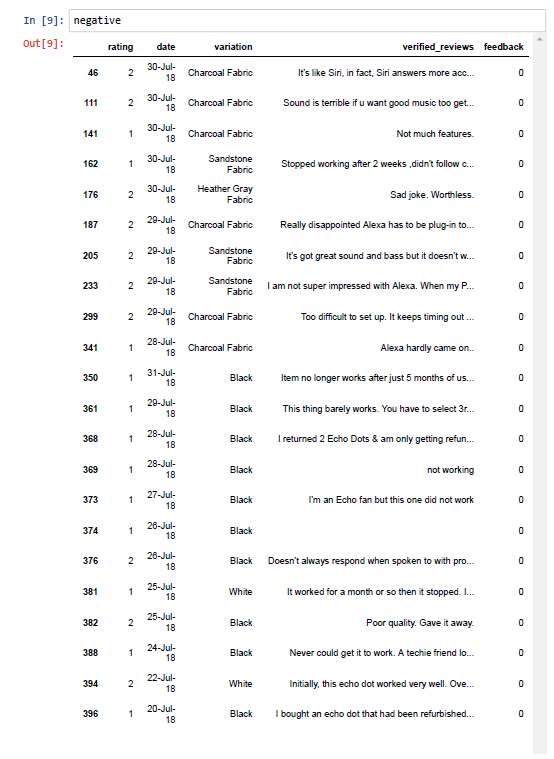
Yes (can play)

**Amazon Data Set**

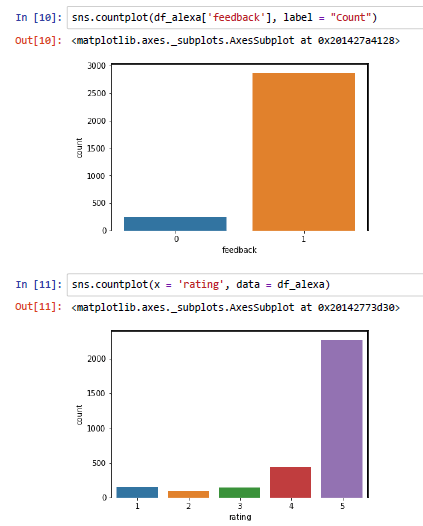


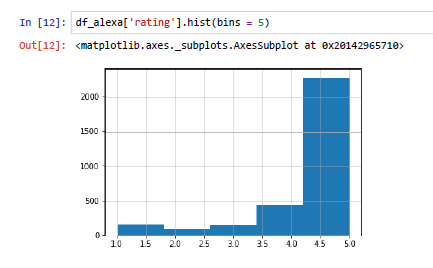


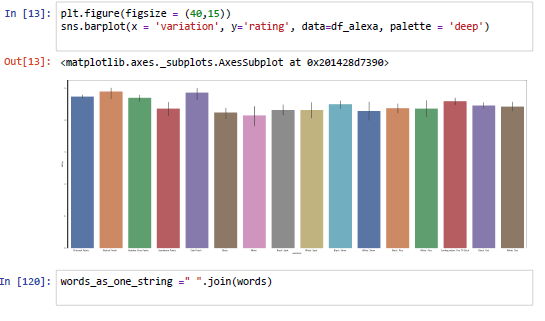


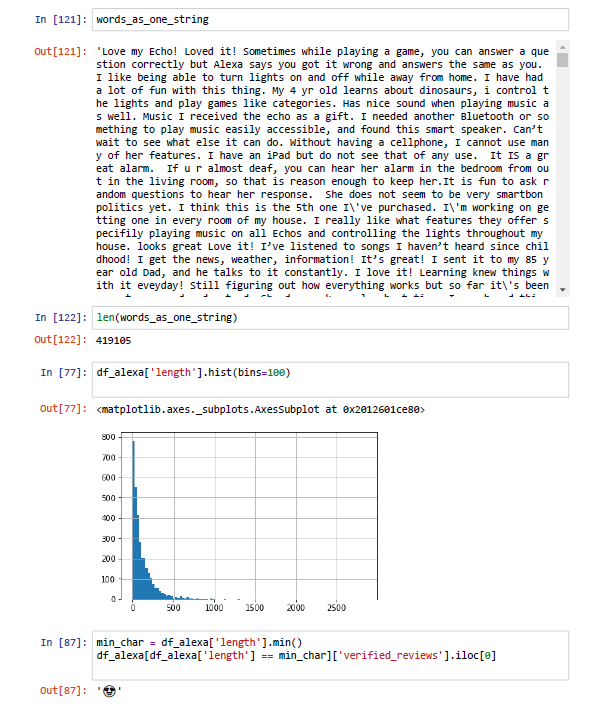


**…. So on**





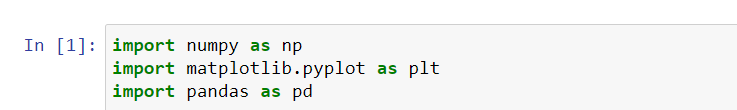


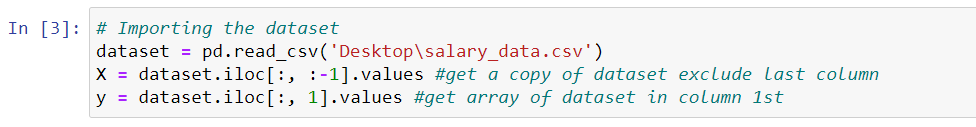


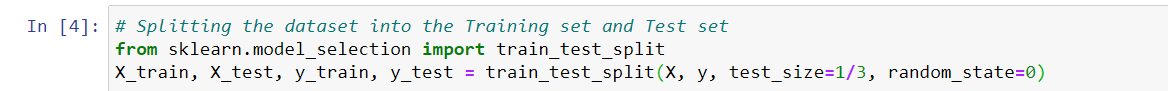
**PRACTICAL - 8**

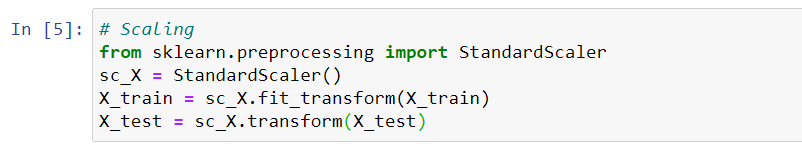
**AIM: Perform the linear regression in python by takingthe data set of employees of any company. Data set should contain Employee ID, Name, Salary, Department, Age, gender etc.**

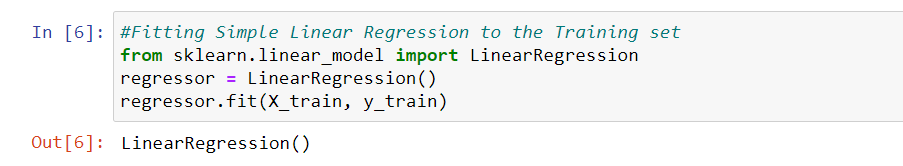
**PROGRAM CODE AND OUTPUT:**

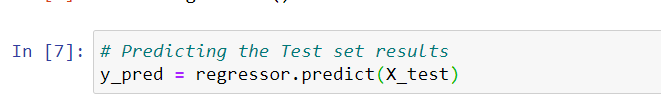
****

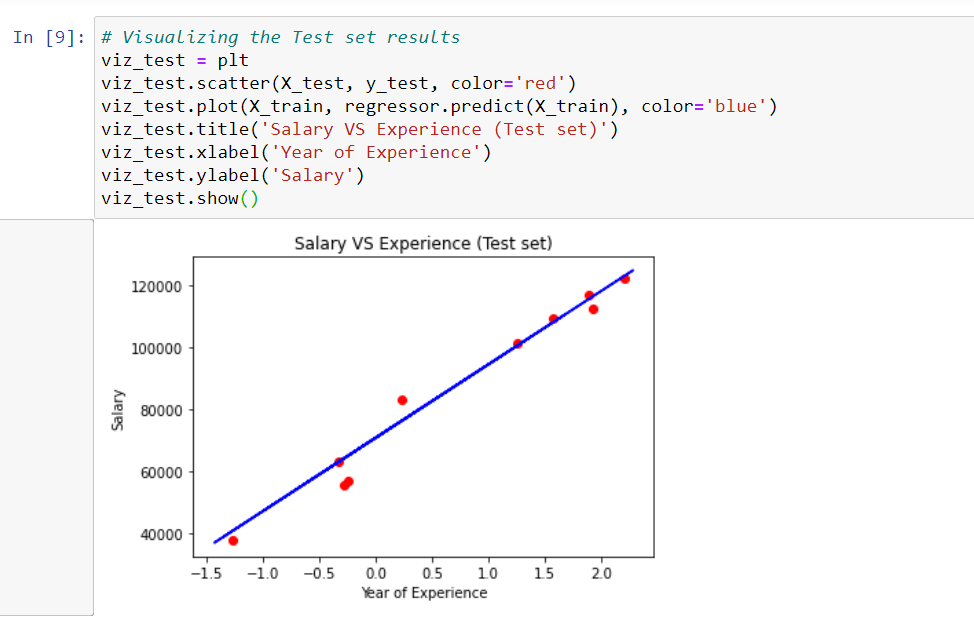
****

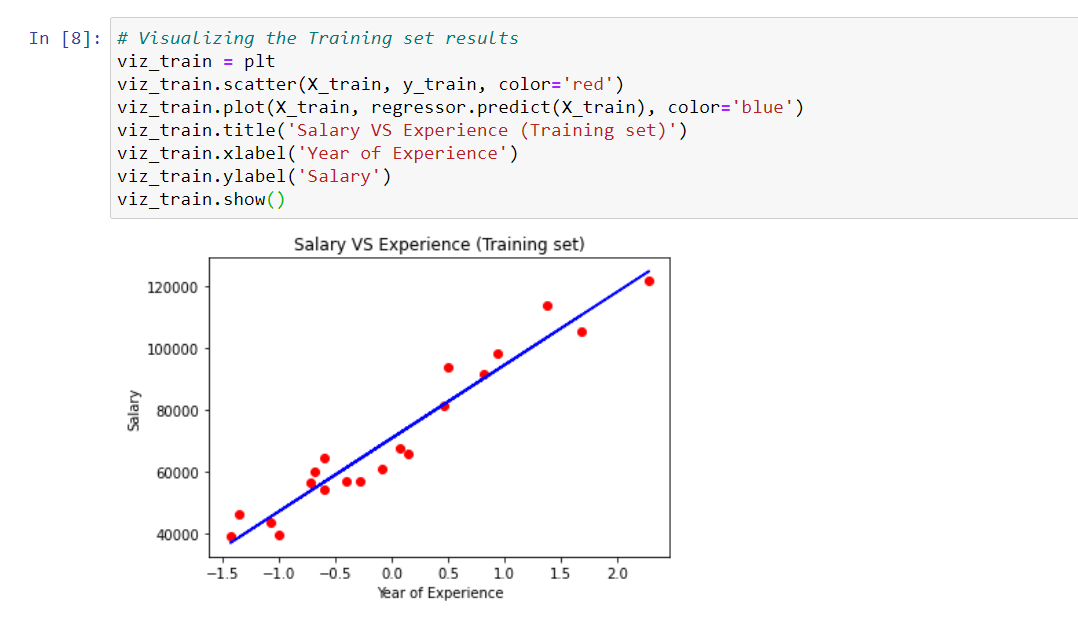
****

****

****

****

****

****

**PRATICAL - 9**

**AIM: Implement the following clustering algorithms using WEKA Tool**

**1. K-means**

**2. Agglomerative**

**3. Divisive.**

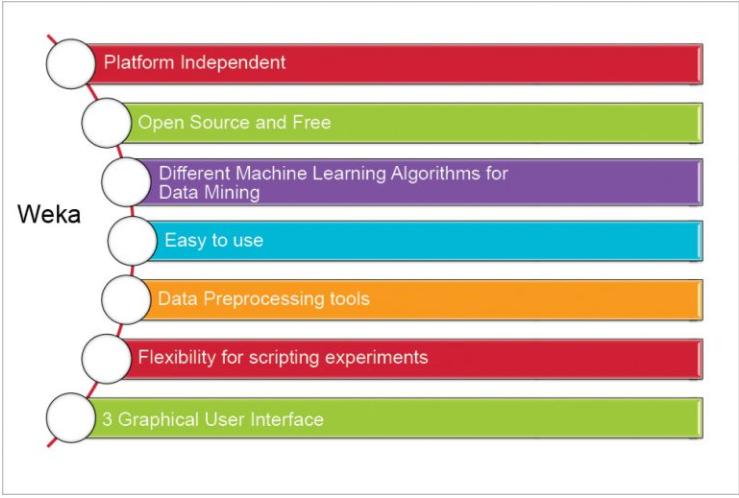
**THEORY:**

**Weka** is a collection of **machine learning** algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. **Weka** contains **tools** for data pre-processing, classification, regression, clustering, association rules, and visualization.

Weka is a collection of tools for:

* Regression
* Clustering
* Association
* Data pre-processing
* Classification
* Visualization

The features of Weka are shown:

****

**Kmeans Algorithm**

**Kmeans** algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

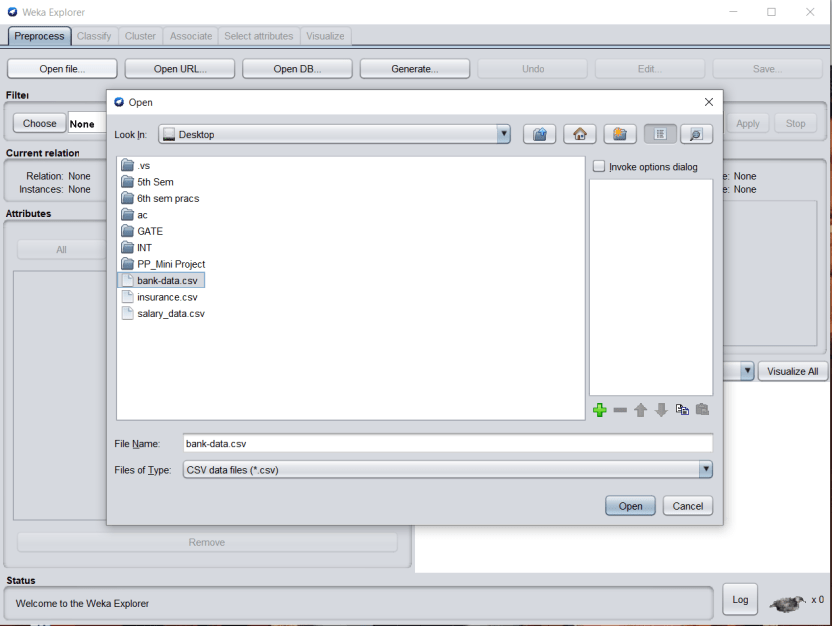
The way kmeans algorithm works is as follows:

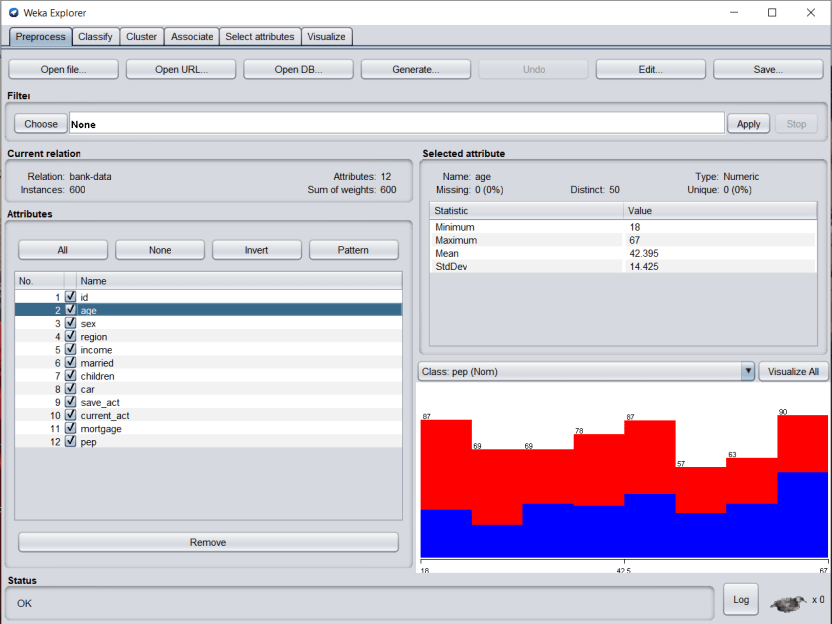
1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn’t changing.

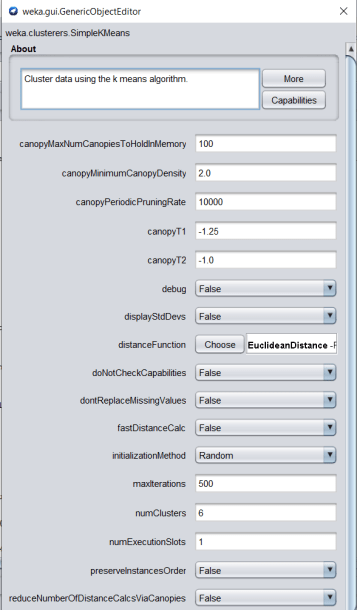
* Compute the sum of the squared distance between data points and all centroids.
* Assign each data point to the closest cluster (centroid).
* Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

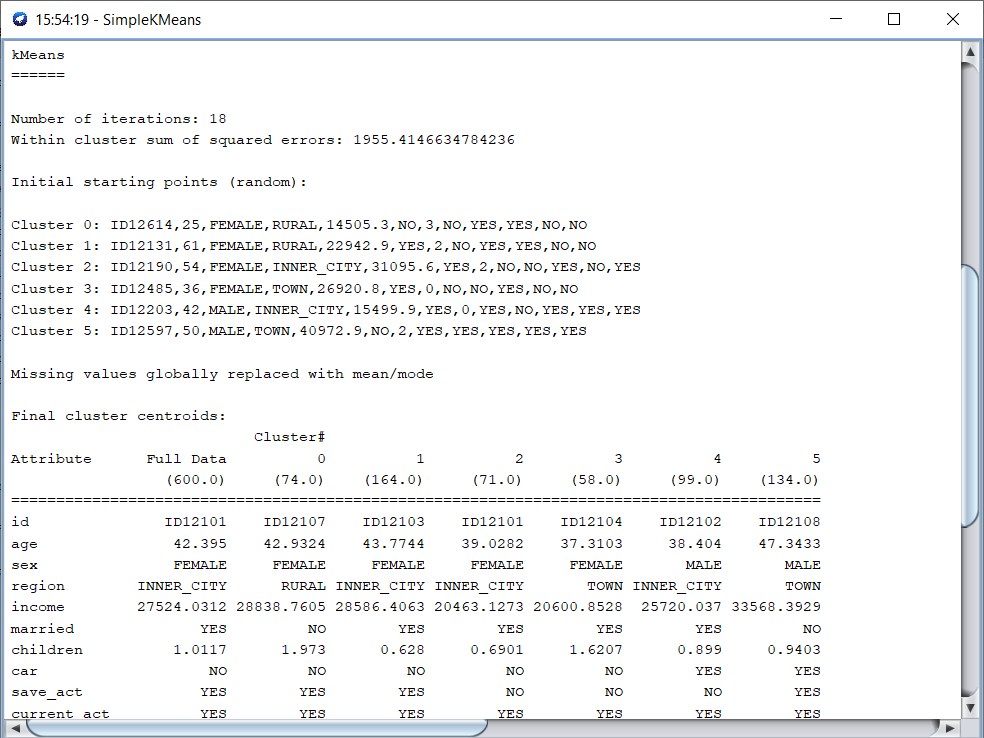
**IMPLEMNTATION:**

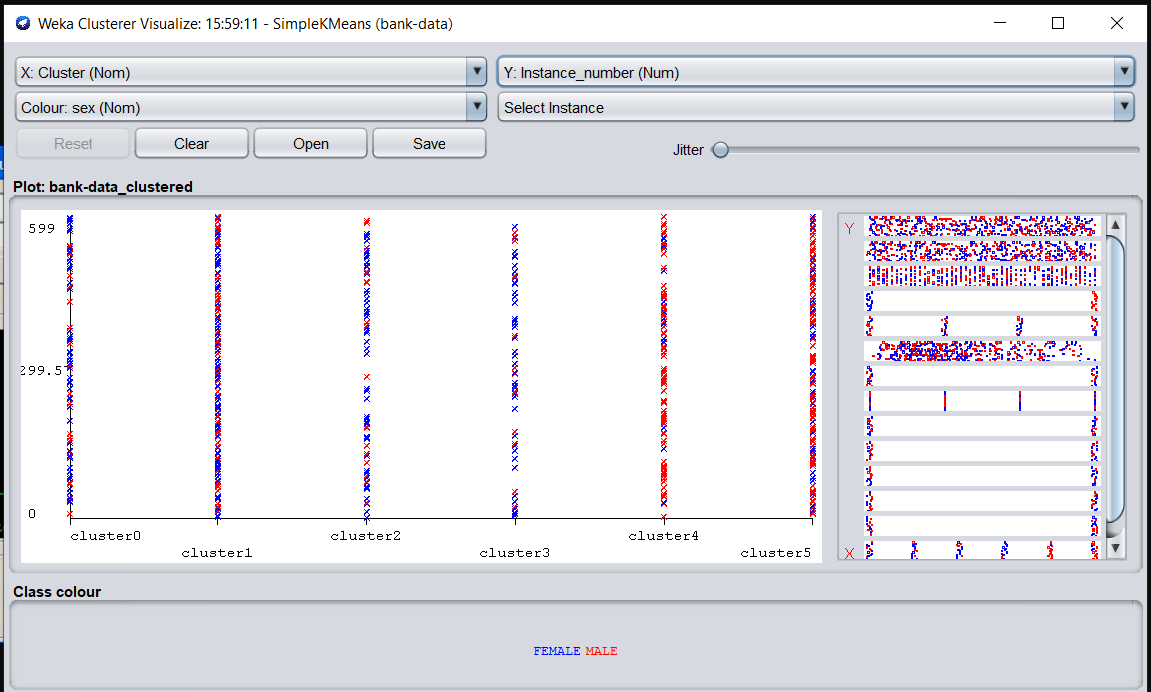
**Importing dataset**

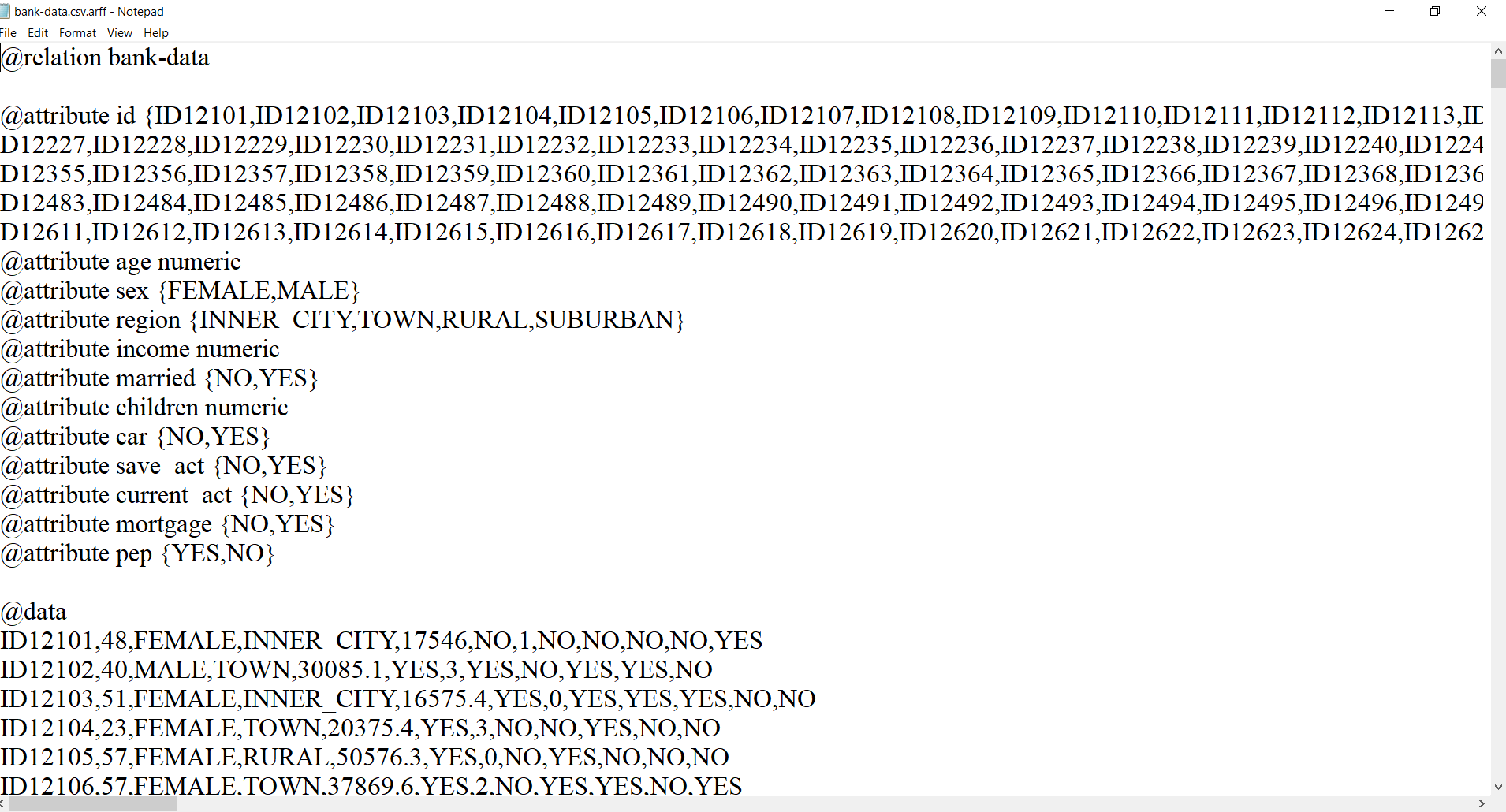
****

****

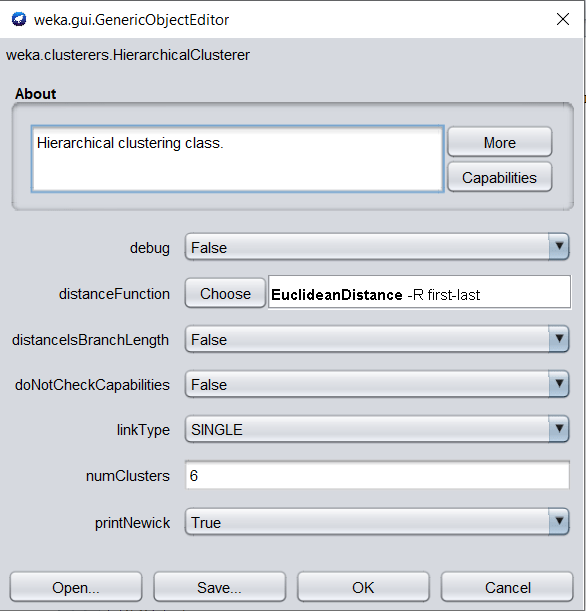
****

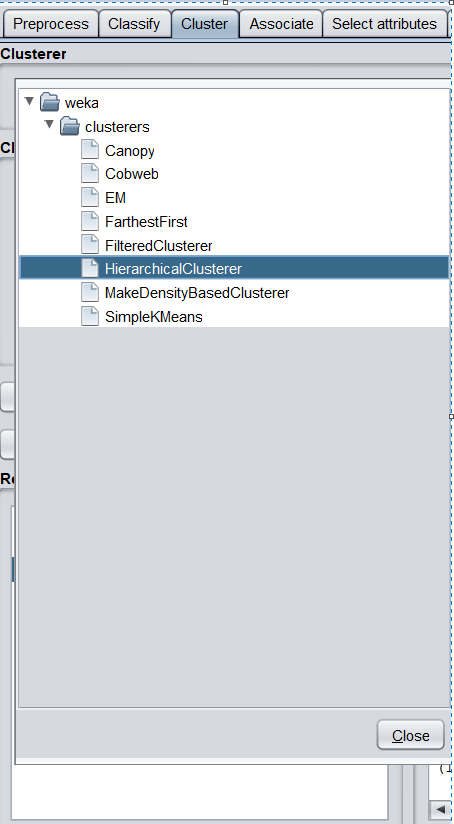
****

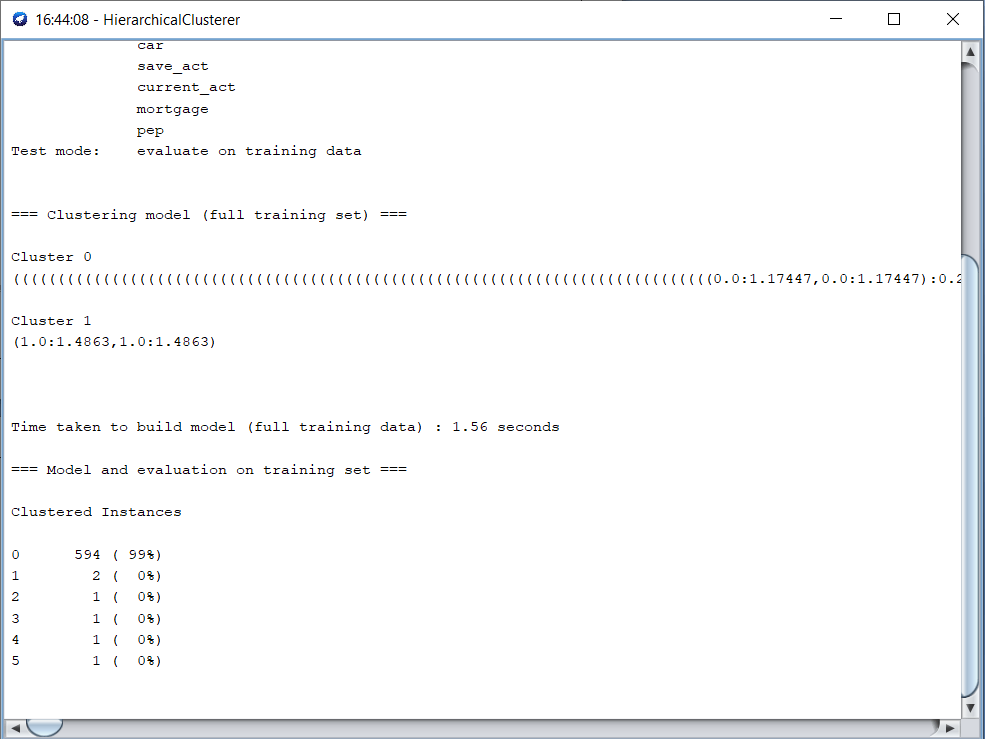
****

****

Hierarchal Clustering: Two types Agglomerative and Divisive





****

**PRACTICAL - 10**

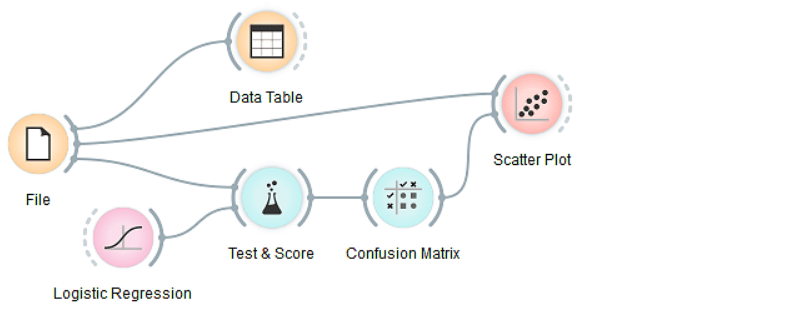
**AIM:** **Perform the Data Preprocessing, Data Visualization, Model, Evaluate, Unsupervised learning and Text mining modules using Orange Tool.**

**THEORY:**

Orange is a great data mining tool for beginners as well as for expert data scientists. Thanks to its user interface users can focus on data analysis instead on laborious coding, making a construction of complex data analytics pipelines simple.

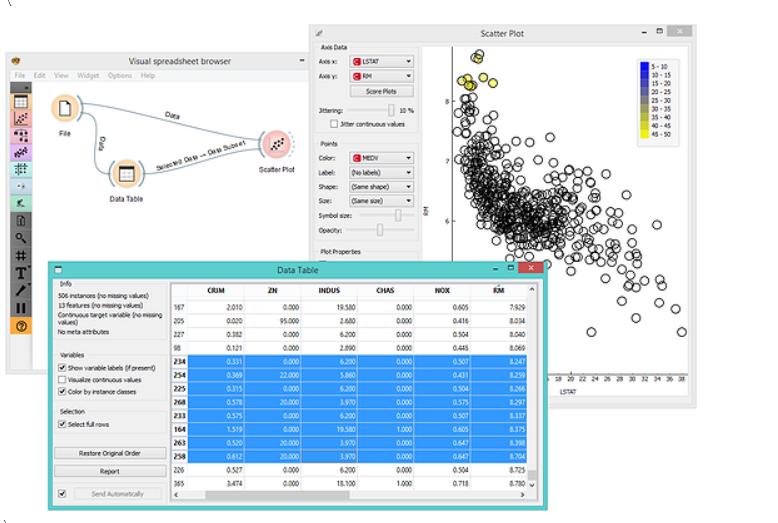
Component-Based Data Mining

In Orange, data analysis is done by stacking components into workflows. Each component, called a widget, embeds some data retrieval, preprocessing, visualization, modeling or evaluation task. Combining different widgets in a workflow enables you to build comprehensive data analysis schemas as you go. With a large library of widgets you won’t be short for choice. Additional widgets are available through add-ons and allow for a more focused and topic-oriented research.



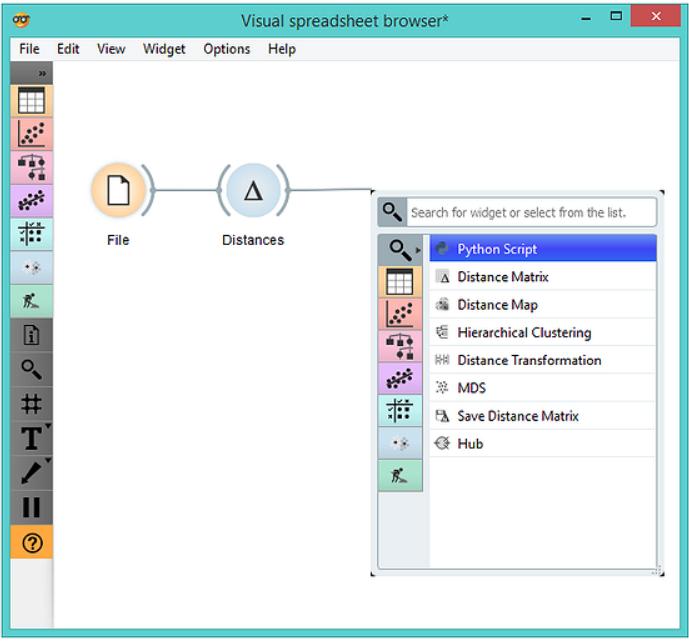
Interactive Data Exploration

Orange widgets communicate with each other. They receive data on the input and send out filtered or processed data, models, or anything the widget does on the output. Say, start with a File widget that reads the data and connect its output to another widget, say, a Data Table, and you have a functioning workflow. Alter any change in one widget, the changes are instantaneously propagated through the downstream workflow. Changing a data file in the File widget will trigger the response in all downstream widgets. This is especially fun if the widgets are open and when you can immediately see the results of any changes in that data, parameters of the methods or selections in interactive visualizations. For example, in a simple workflow below, where selection of the data in the spreadsheet propagates to a scatter plot, which marks the selected data instances.

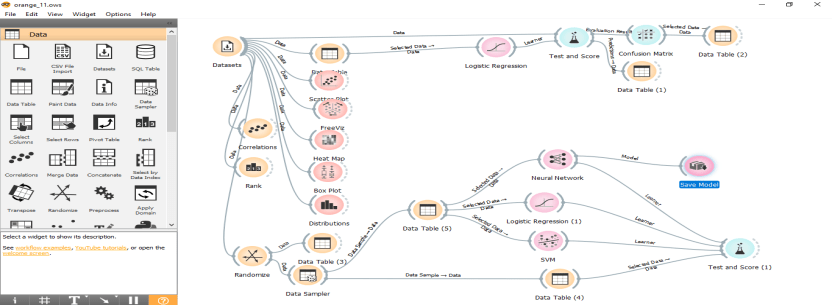


Clever Workflow Design Interface

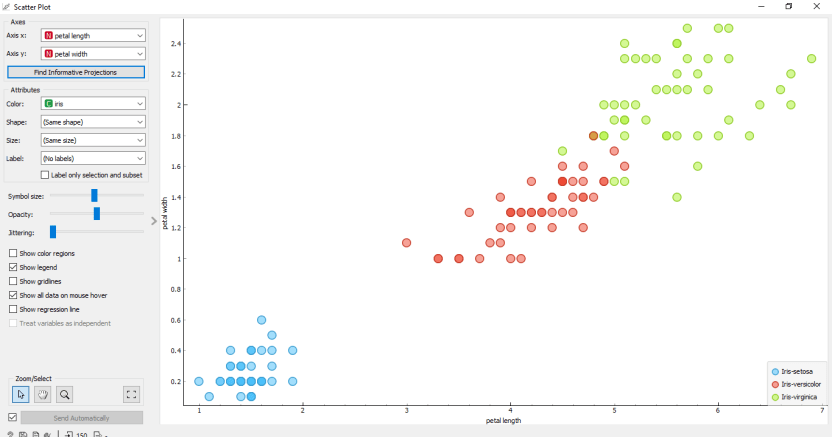
Orange is easy to use even for complete novices. Start with the File widget and Orange will automatically suggest the next widgets that can be connected to it. For example, Orange knows you are likely to want Hierarchical Clustering after you’ve set up your Distances widget. All other defaults in the widgets are also set in a way that enables a simple analysis even without knowing a whole lot about statistics, machine learning, or exploratory data mining in general.

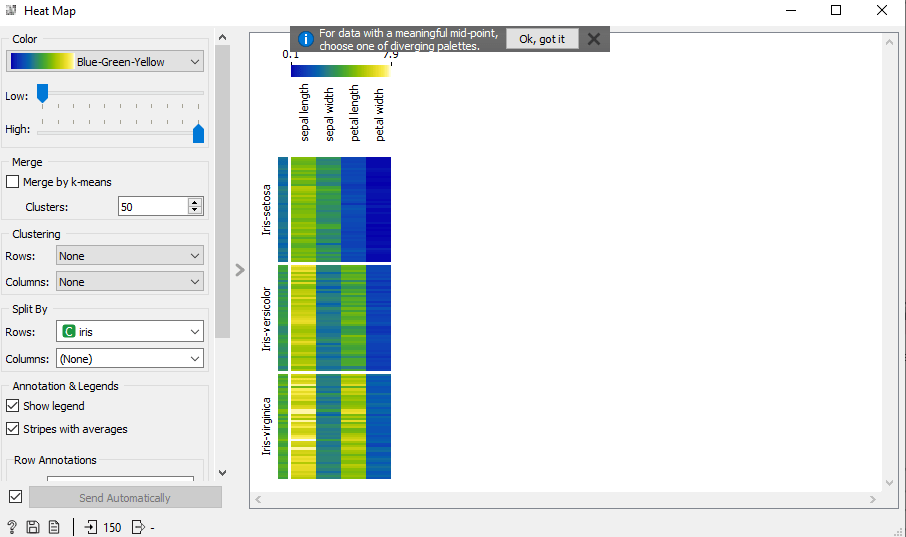
****

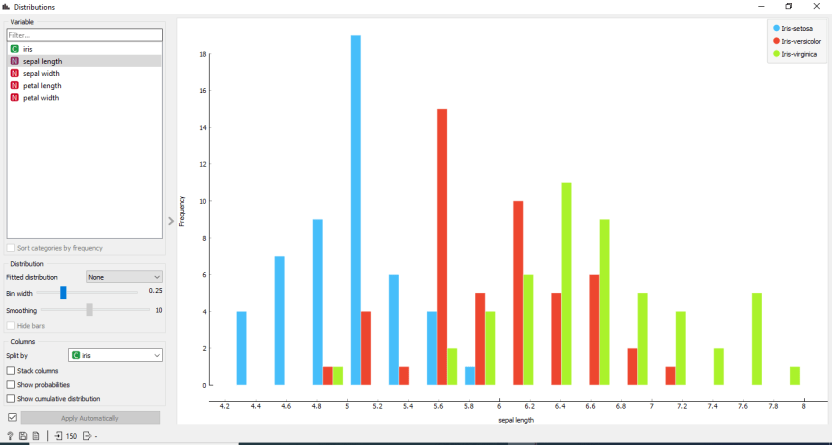
**OUTPUT:**

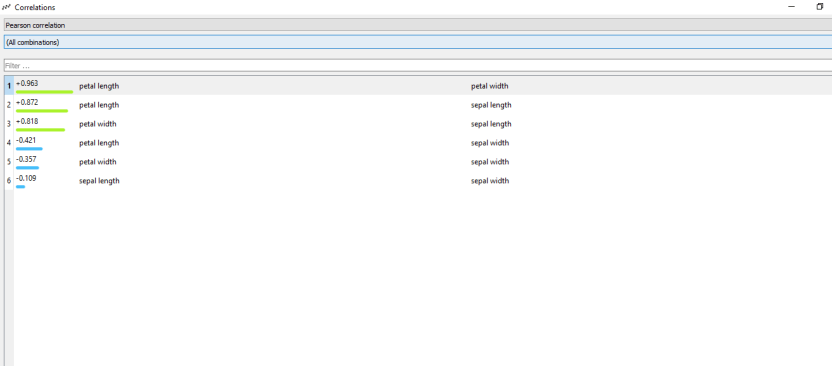


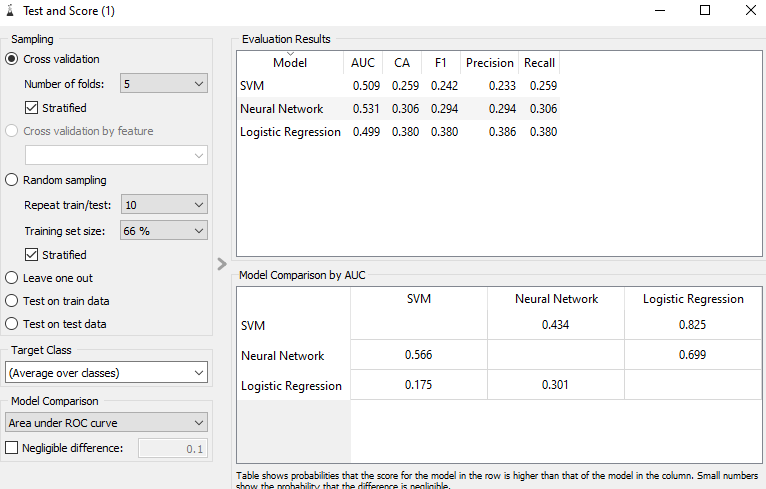












**PRACTICAL - 11**

**AIM:** **Perform Association, Classification and clustering from Data Cube created by OLAP Manager using DBMiner Tool.**

**THEORY AND OUTPUT:**

### Mining Associations from a Data Cube

#### About Association

**Association** mining on a set of data looks for values in different dimensions (attributes) that commonly occur together, suggesting an association between them.

In DBMiner, three kinds of associations could be possibly mined:

**Inter*-*dimensional association**. Associations among or across two or more dimensions.

Customer-Country("Canada") => Product-Sub Category("Coffee")  
i.e. Canadian customers are likely to buy coffee.

**Intra-dimensional association**. Associations present within one dimension grouped by another one or several dimensions. For example, if you want to find out which products customers in Canada are likely to purchase together:

Within Customer-Country("Canada"):  
     Product-ProductName("CarryBags") => Product-ProductName("Tents")  
i.e. Customers in Canada, who buy carry-bags, are also likely to buy tents.

**Hybrid association**. Associations combining elements of both inter- and intra-dimensional association mining. For example,

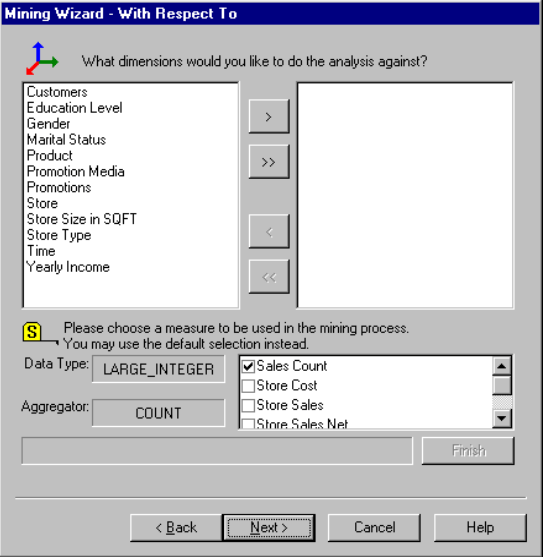
Step (a):

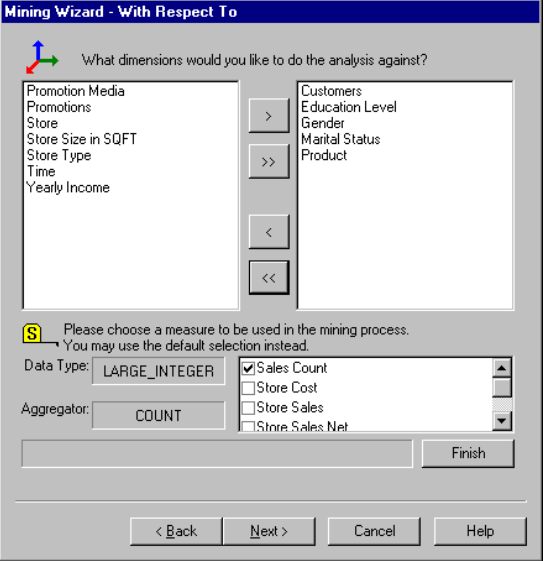


Step (b):

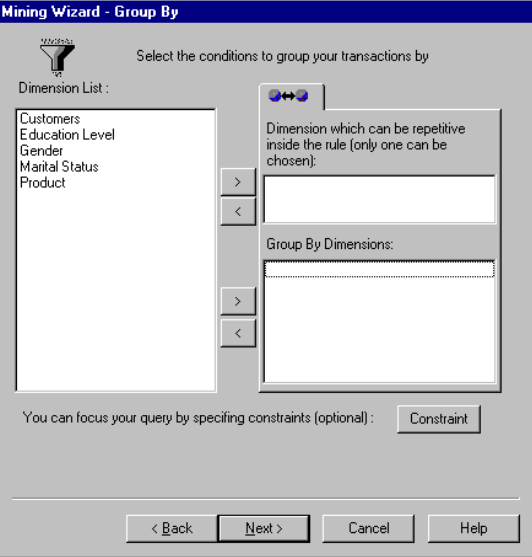


Step (c):

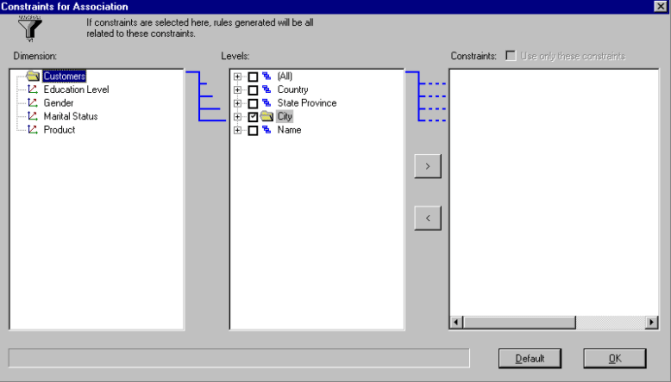




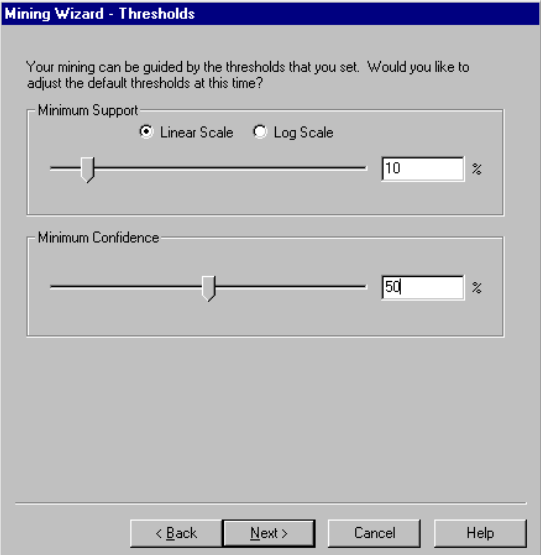
Step (d):



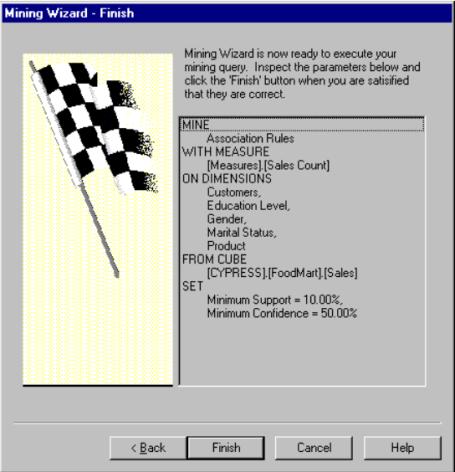
Step (e):



Step (e):



Step (f):



### 2. Mining Classifications from a Data Cube

#### About Classification

**Classification** mining analyzes a set of training data (i.e. a set of objects whose class labels are known) and constructs a model for each class based on the features in the data. A set of classification rules are generated by the classification process, and these can be used to classify future data, as well as develop a better understanding of each class in the database.

In the classification process, attribute relevance analysis is very important. It is performed according to the analysis of an ***uncertainty measurement***, which determines how relevant an attribute is to the chosen classification attribute. Only a few of the most relevant attributes are retained for the classification analysis and the weakly relevant or irrelevant ones are not further considered.

In DBMiner, three thresholds are used to tackle noise and exceptional data and facilitate statistical analysis.

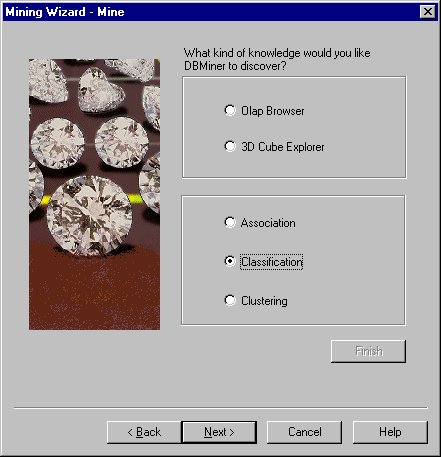
* + - ***Classification threshold:*** Helps justify the classification of a particular subset of the data (found at a single node) when a significant portion of it belong to the same class.
    - ***Noise threshold:*** Helps ignore a node if it contains only a negligible number of examples (i.e. noise).
    - ***Training/testing set size:*** Sets the percentage of the whole data set to be used for training and testing. (i.e. 80% for training, 20% for testing).

The following steps will take you through a classification task.

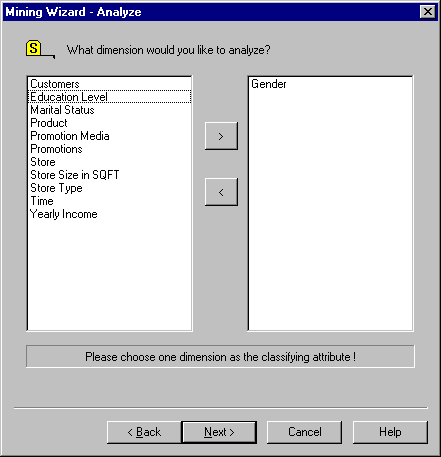
*Step (a):*



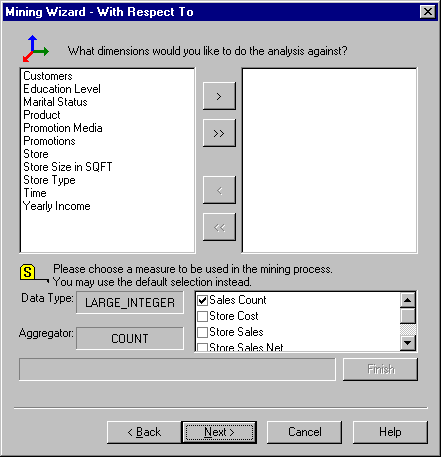
*Step (b)*

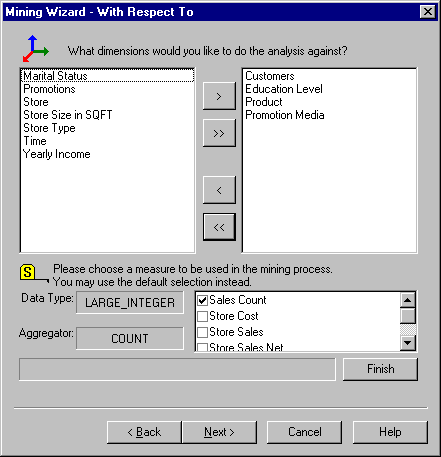


*Step (c)*



*Step (d):*

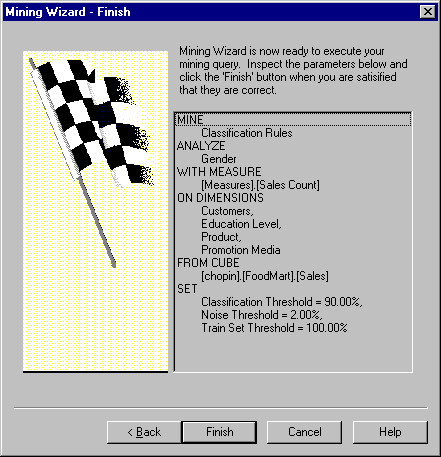




*Step (e):*



*Step (f):*



### 3. Mining Clusters from a Data Cube

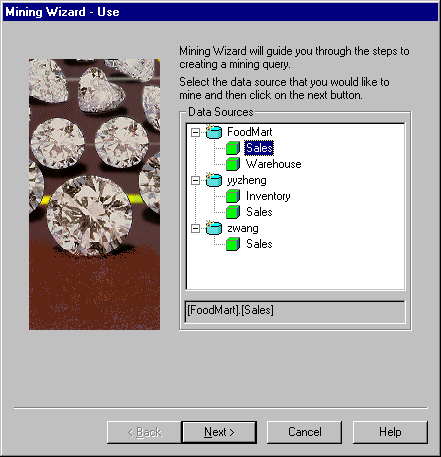
#### About Clustering

**Clustering** groups all the value of one dimension by the values of the second dimension. In DBMiner, only two cube dimensions can be chosen in a mining session since the clustering space is 2-dimensional. The underlying algorithm used in DBMiner is the *k*-means method. For detailed information about the *k*-means method, see the on-line documentation.

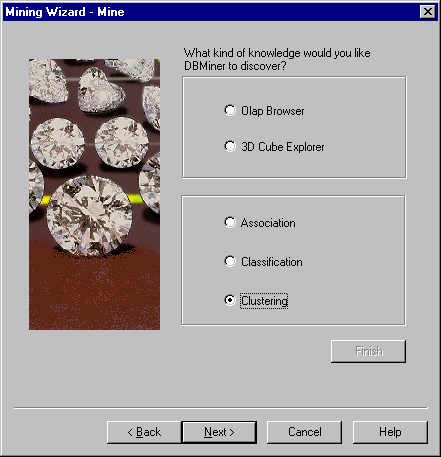
In DBMiner, there are several parameters that can be adjusted to tune the mining process.

* + - ***Number of clusters*** refers to *k* in the *k*-mean algorithm. If the number of clusters requested exceeds the number of points in the plane, an error message will be issued and the clustering process is stopped.
    - ***Dimension weights*** refers to the coefficients for each dimension. The default value is 1.00 but can be decreased if you want a particular dimension to be relatively less influential with respect to the other dimension. Because this scaling reflects the relative different in influence, it is only calibrated between 0.01 - 1.00.
    - ***Max clustering passes*** refers to the number of passes in the *k*-mean algorithm.
    - ***Filter threshold*** ensures that cells in a data cube containing no records are not included in the clusters. This threshold can be raised to exclude more cells and thus reduce the number of points on a 2-dimensional plane.

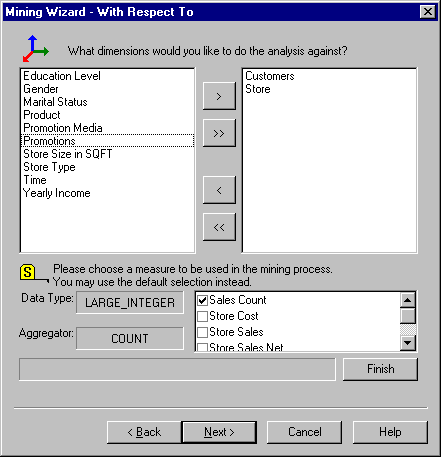
*Step (a):*



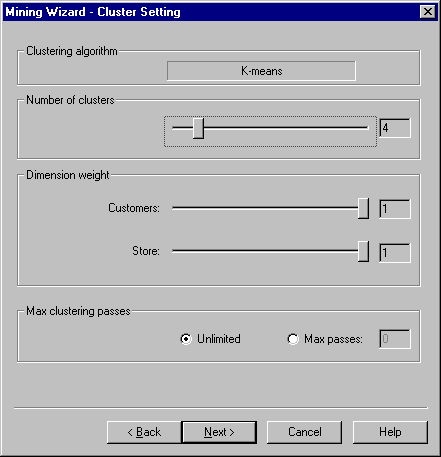
*Step (b):*



*Step (c):*



*Step (d):*



*Step (e):*



**Practical - 12**

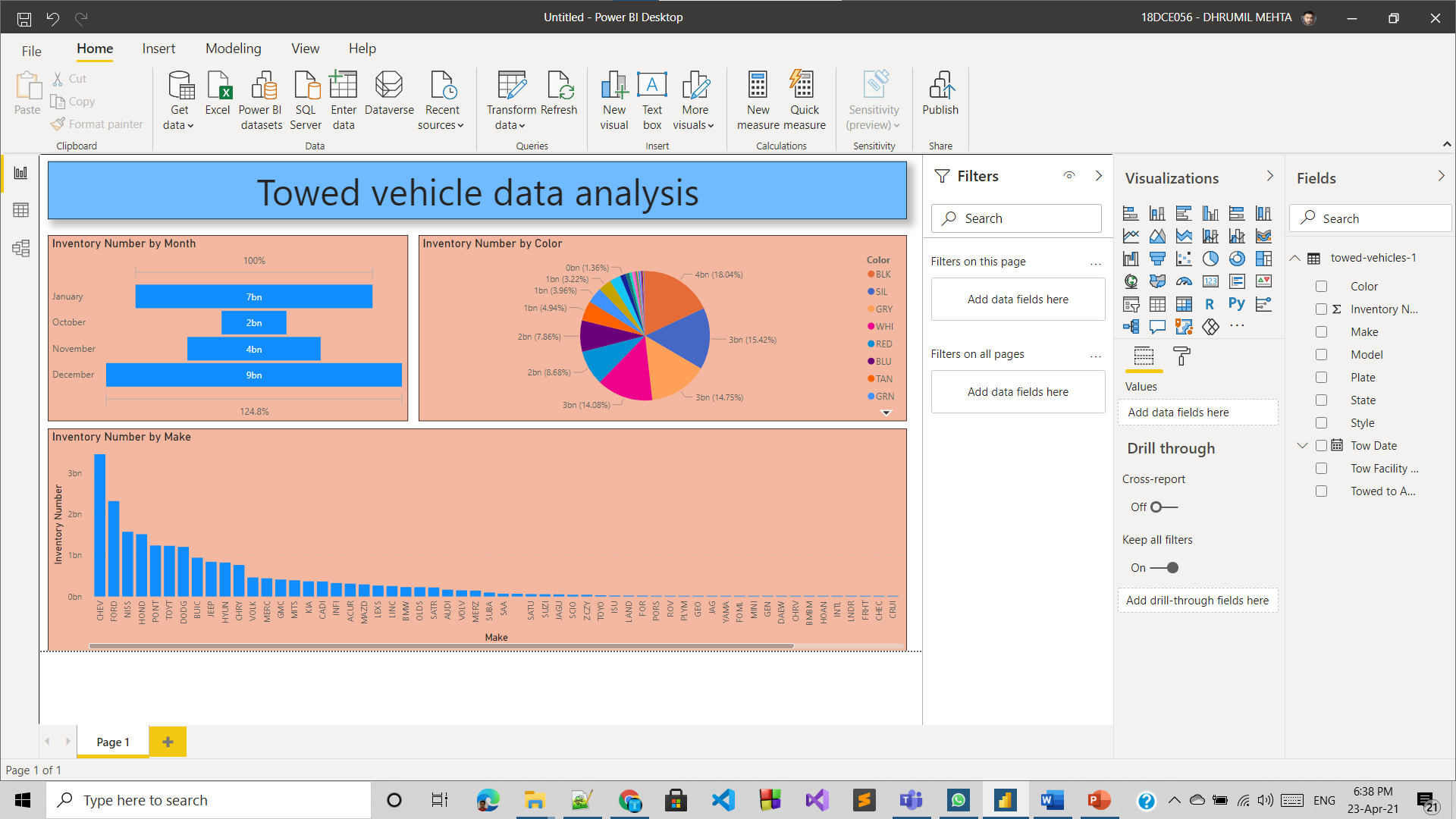
**Aim:** **Implement Data Analysis and Data Visualization using Power BI and Tableau for vehicle dataset.**

**Power BI:**

* Power BI is a Data Visualization and Business Intelligence tool that converts data from different data sources to interactive dashboards and BI reports. Power BI suite provides multiple software, connector, and services - Power BI desktop, Power BI service based on Saas, and mobile Power BI apps available for different platforms. These set of services are used by business users to consume data and build BI reports.
* Power BI desktop app is used to create reports, while Power BI Services (Software as a Service - SaaS) is used to publish the reports, and Power BI mobile app is used to view the reports and dashboards.

**Implementation:**

Vehicle towed dataset used



* **Conclusion:** With the use of PowerBI tool we can visualize any type data easily and it can easily convert different data to interactive dashboards and BI report.