Institute of Distance & Open Learning
M.Sc.I.T

UNIVERSITY OF MUMBAI

Certificate

This is to certify that **Rajpurohit Harsh Hargopalsingh** Seat no **600257** has successfully completed all the practical of paper titled **"Big Data Analytics"** for M.Sc. (Information Technology) Part 1 Sem 2 in the year 2022-2023

Signature
Faculty In-Charge

Head of the Department

Examiner

INDEX

Sr.No	Title	Date	Teachers Sign
1.	Install, configure and run Hadoop and HDFS ad explore HDFS.		
2.	Wordcount in hadoop		
3.	Configure the Hive and implement the application in Hive		
4.	Implement an application that stores big data in Hbase / MongoDB & manipulate it using R / Python		
5.	Decision tree classification techniques		
6.	SVM classification techniques		
7.	Linear regression		
8.	Clustering Model		

Practical No 1

Install, configure and run Hadoop and HDFS ad explore HDFS.

Step 1: Download and install VirtualBox

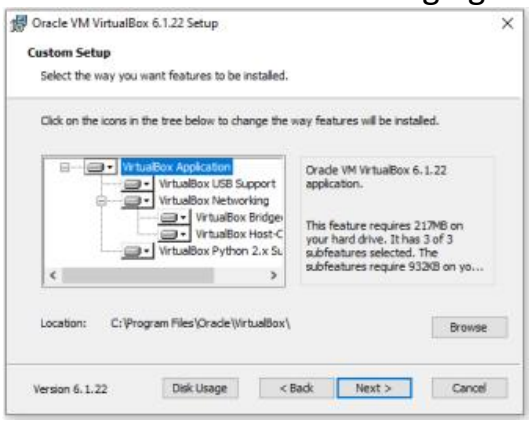
Go to the website of Oracle VirtualBox and get the latest stable version from the following site https://www.virtualbox.org/

click on 'Download''

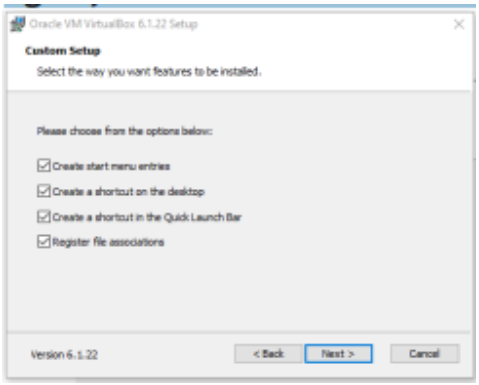
You will get VirtualBox-6.1.22-144080-Win.exe file downloaded.

Double click and run it. Click on next.

Click on 'next' without changing the default folder as shown below:

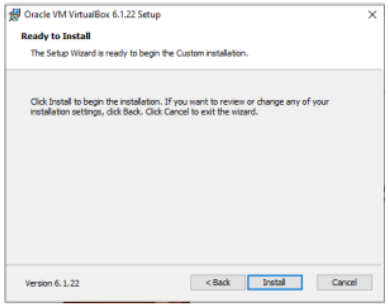


Again, click on next as shown below:

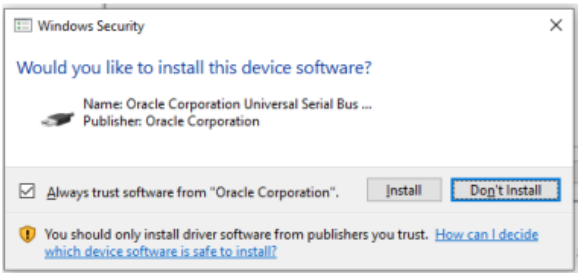


Finally, click on ‘Yes’

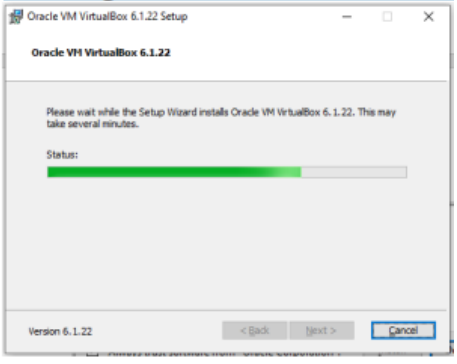
Click on ’Install’



It may ask you for the permission to install, click ’yes’ to allow. Select ‘Install’ as shown below:



You will get the screen as shown below:

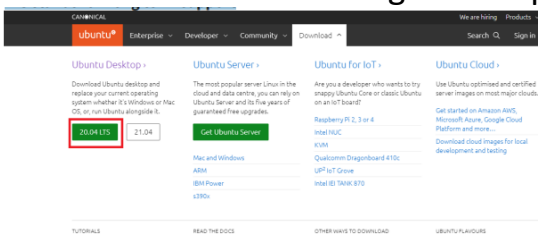


Click on ‘Finish’ to finish Installation of virtual box

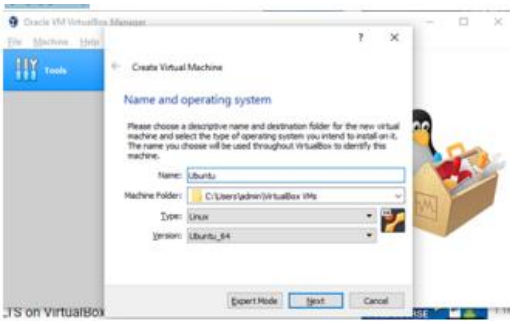
You will get the following screen:

Step 2: download Ubuntu

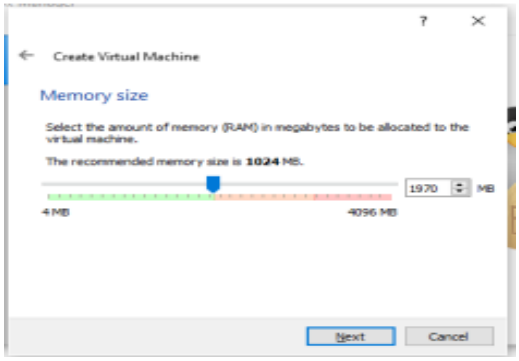
Download iso file ubuntu-20.04.2.0-desktop-amd64; which is required to install Ubuntu.Browse ubuntu.com Click on download and 20.04 LTS as shown below: LTS stands for Long term support



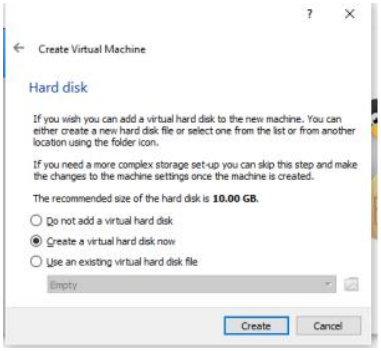
You will get file, which may take few minutes to download. Now, click on ‘New’ to virtual box and write Name as ‘Ubuntu’ as shown below:



Click on ‘Next’

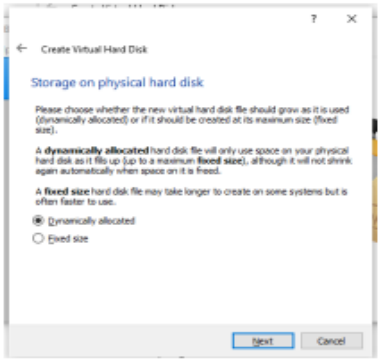


Here, you allow memory size up to green indicator (1970 MB).
Click on ‘Next’.

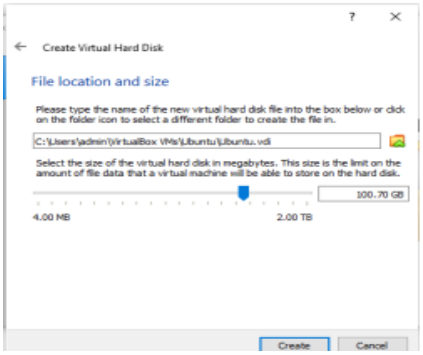


Don’t change anything in this screen and click on ‘Create’

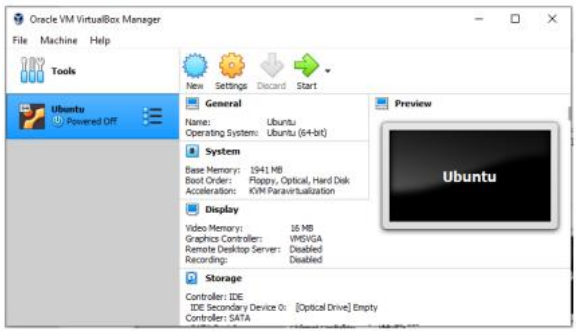
Click on ‘Next’, keeping the selection as it is (on VDI).’



Keep this screen also as it is and click on ‘Next’



Keep the file location as it is but preferably keep size 100 GB and click on ‘Create’.
You may see the following screen having Ubuntu on Virtual Machine.

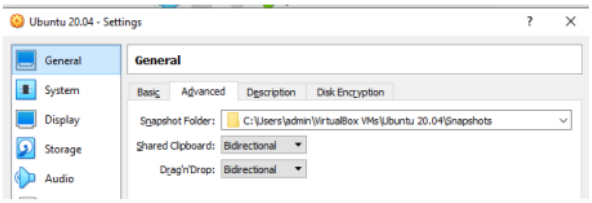


Select ‘settings’

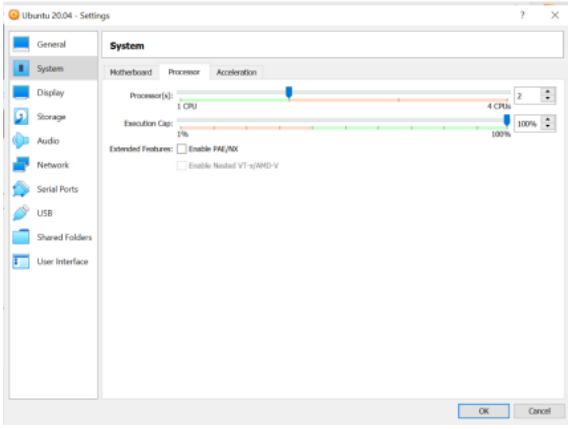
Select ‘General’ -> ’ Basic’ as shown below:

You may change the name from Ubuntu to Ubuntu 20.04

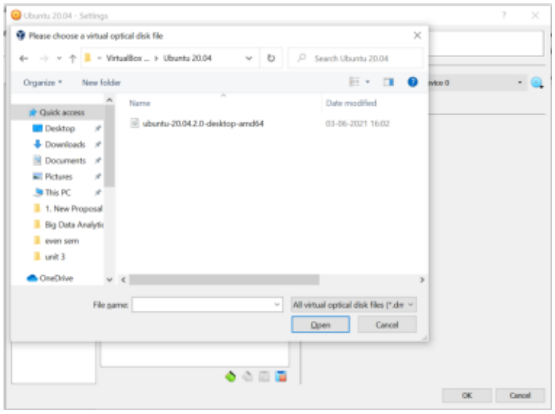
Select bidirectional in ‘General’ -> ’ Advanced’ as shown below:



Go to ‘System’ option and change the processor up to green bar, usually 4.(if it allows)Cut and paste your ubuntu .iso file from current folder to C:\Users\ADMIN\ VirtualBox VMs\Ubuntu 20.04 folder.Click on ‘Storage’ and click on ‘Empty’ followed by ‘Choose a disk file’ as shown below:

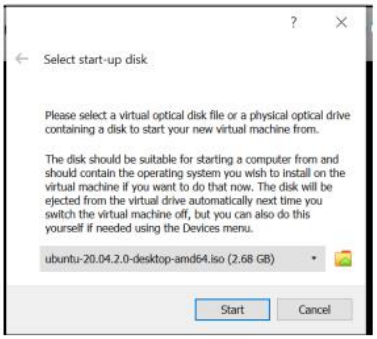


Browse the folder where you have selected ubuntu iso file

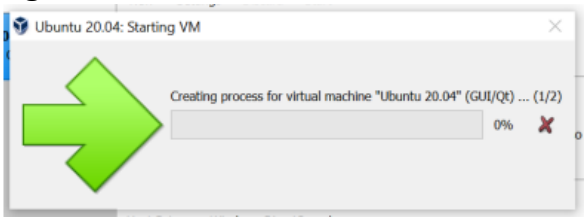


Click on Ubuntu….iso file and click on open and then click on ok.

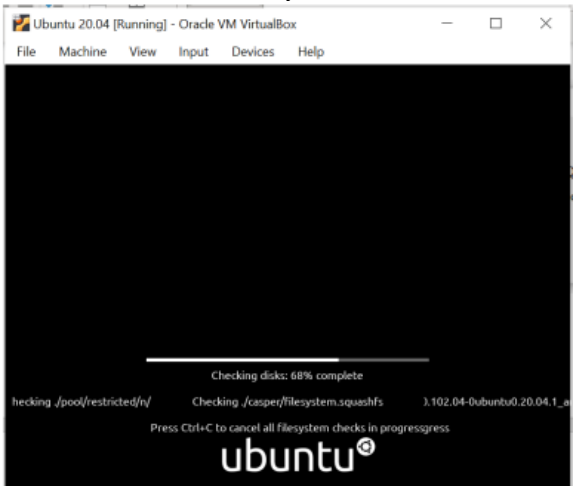
Click on Ubuntu -> start button.



Again, click on ‘Start’ button. It will show you the following screen.

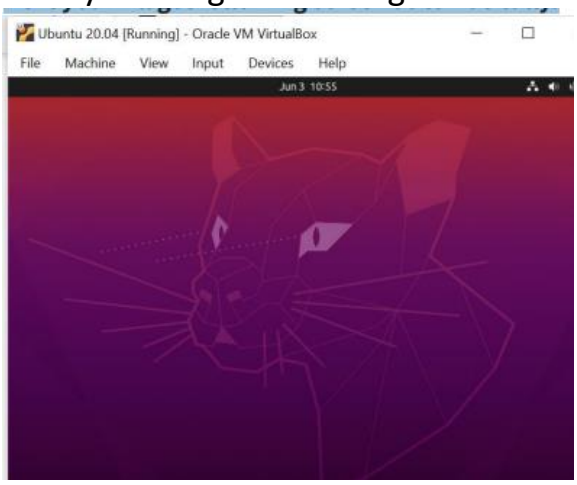


And simultaneously one more screen as follows:



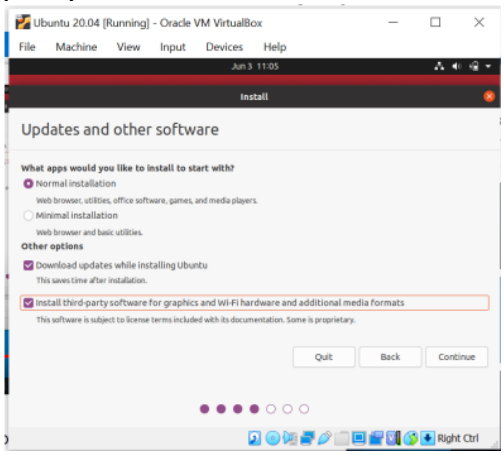
Keep on closing all warnings.

Next you will get following screen automatically.

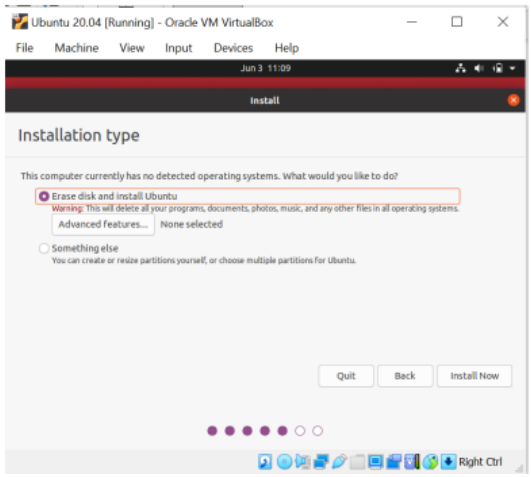


Select language -> English and click on ‘Install Ubuntu’.in ‘Keyboard Layout’

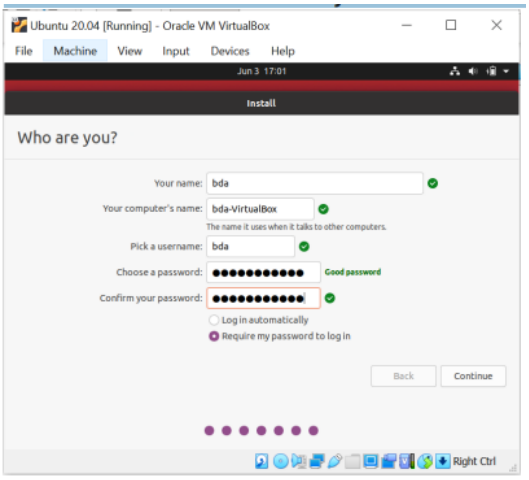
screen, select ‘English UK’. Click on ‘Continue’. Select the checkbox for third party software as shown below:



Click on ‘continue’.



Select Erase disk and Install Ubuntu and click on ‘Install Now’. Click on ‘Continue’ on the next screen. Select “Kolkata” for “where are you?” and click on ‘Continue’.



Click on continue after entering name, company name, username, password and confirm your password.

Installation of Ubuntu started. Click on finish once installation done.
Click on restart and press Enter key.
Step 3 Install Hadoop
Login to ubuntu
Some keys may change like you try to type @ and it types “.
** please refer to note - Some Keys for Ubuntu under UK keyboard layout – at the end.Search for Ubuntu terminal on search bar, after login done. Apply following commands from ubuntu terminal
$ sudo apt update
$ sudo apt install default-jdk
$ ava -version'
$ wget https://hadoop.apache.org/release/3.2.2.html/hadoop-3.2.2.tar.gz
$ tar xzvf hadoop-3.2.2.tar.gz
$ sudo mv hadoop-3.2.2 /usr/local/hadoop
$ readlink -f /usr/bin/java | sed "s:bin/java::"
# : Configuring Hadoop’s Java Home; To begin, open hadoop-env.sh
$ sudo nano /usr/local/hadoop/etc/hadoop/hadoop-env.sh
File will be opened. Add the following line at the end of .sh file
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
to save the changes in the file, press ctrl and x together.
then press Y
then press Enter key
then apply following commands:
$ /usr/lib/jvm/java-11-openjdk-amd64/
Step 4: Running Hadoop
$ /usr/local/hadoop/bin/hadoop
$ mkdir ~/input
$ cp /usr/local/hadoop/etc/hadoop/*.xml ~/input
We can use the following command to run the MapReduce hadoop mapreduce-examples

program, a Java archive with several options. We'll invoke its grep program, one of the many examples included in hadoop-mapreduce-examples, followed by the input directory, input and the output directory grep_example. The MapReduce grep program will count the matches
of a literal word or regular expression. Finally, we'll supply the regular expression allowed[.]* to find occurrences of the word allowed within or at the end of a declarative sentence. The expression is case-sensitive, so we wouldn't find the word if it were capitalizedat the beginning of a sentence:

```
$ /usr/local/hadoop/bin/hadoop jar
/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.2.2.jar grep ~/input ~/grep_example 'allowed[.]*'
$ cat ~/grep_example/*
```

Practical 2
Wordcount in hadoop

Login to bda user of Ubuntu
create a folder wordcount under home folder of Ubuntu
create file WordCount.java and store in that folder:
code:

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class WordCount {
 public static class TokenizerMapper extends Mapper<Object, Text, Text,
IntWritable>{
 private final static IntWritable one = new IntWritable(1);
 private Text word = new Text();
 public void map(Object key, Text value, Context context
 ) throws IOException, InterruptedException {
 StringTokenizer itr = new StringTokenizer(value.toString());
 while (itr.hasMoreTokens()) {
 word.set(itr.nextToken());
 context.write(word, one);
 }
 }
 }
 public static class IntSumReducer extends
Reducer<Text,IntWritable,Text,IntWritable> {
 private IntWritable result = new IntWritable();
 public void reduce(Text key, Iterable<IntWritable> values, Context context)
throws IOException,
InterruptedException
 {
 int sum = 0;
 for (IntWritable val : values) {
```

```
 sum += val.get();
 }
 result.set(sum);
 context.write(key, result);
 }
 }
public static void main(String[] args) throws Exception {
 Configuration conf = new Configuration();
 Job job = Job.getInstance(conf, "word count");
 job.setJarByClass(WordCount.class);
 job.setMapperClass(TokenizerMapper.class);
 job.setCombinerClass(IntSumReducer.class);
 job.setReducerClass(IntSumReducer.class);
 job.setOutputKeyClass(Text.class);
 job.setOutputValueClass(IntWritable.class);
 FileInputFormat.addInputPath(job, new Path(args[0]));
 FileOutputFormat.setOutputPath(job, new Path(args[1]));
 System.exit(job.waitForCompletion(true)?0:1);
 }
}
```

create a file input.txt with text editor and store it in wordcount folder.

input.txt should have list of names in it, with each name in one line.

create a folder wordcount_classes in wordcount folder.

create env variable using hdoop terminal as follows and apply subsequent commands:

bda@bda-VirtualBox:~$ su hdoop

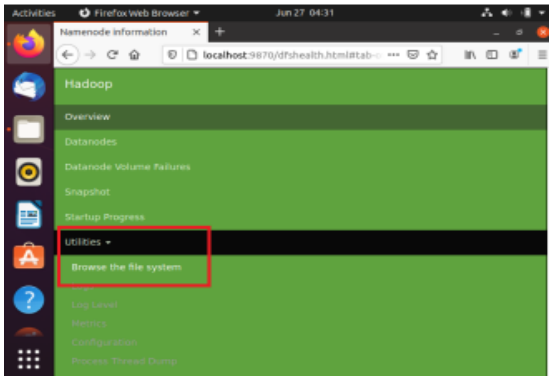
Password:

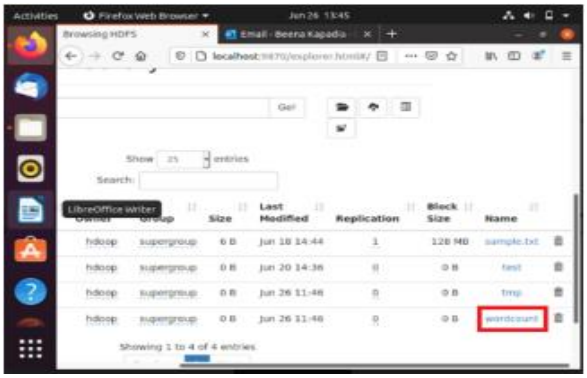
hdoop@bda-VirtualBox:/home/bda$ export HADOOP_CLASSPATH=$(hadoop classpath)

hdoop@bda-VirtualBox:/home/bda$ echo $HADOOP_CLASSPATH

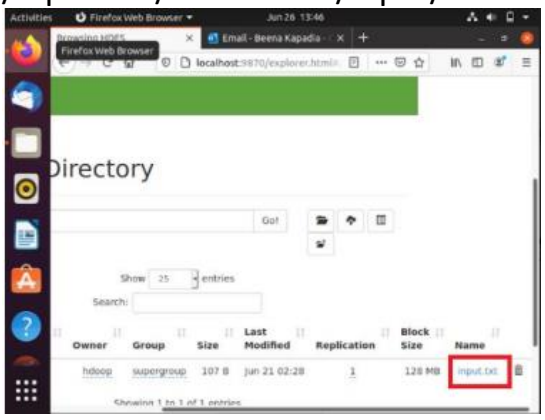
/home/hdoop/hadoop-3.3.1/etc/hadoop:/home/hdoop/Hadoop-3.3.1/share/hadoop/common/lib/*:/home/hdoop/Hadoop-3.3.1/share /hadoop/common/*:/home/hdoop/Hadoop-3.3.1/share /hadoop/hdfs:/home/hdoop/Hadoop-3.3.1/share/hadoop/hdfs/lib/*:/home/hdoop/hadoop-3.3.1/share /hadoop/hdfs/*:/home/hdoop/Hadoop 3.3.1/share/hadoop/mapreduce/*:/home/hdoop/hadoop-3.3.1/share /hadoop/yarn:/home/hdoop/Hadoop 3.3.1/share/hadoop/yarn/lib/*:/home/hdoop/hadoop-3.3.1/share /hadoop/yarn/*

hdoop@bda-VirtualBox:/home/bda$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bda-VirtualBox]
2021-06-26 13:13:58,694 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
hdoop@bda-VirtualBox:/home/bda$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hdoop@bda-VirtualBox:/home/bda$ jps
3248 ResourceManager
3779 Jps
3382 NodeManager
2631 NameNode
2760 DataNode
2953 SecondaryNameNode
hdoop@bda-VirtualBox:/home/bda$ hdfs dfs -mkdir /wordcount/
(Browse localhost:9870 on any browser and click on utility and select browse the file systemyou can see your folder there)





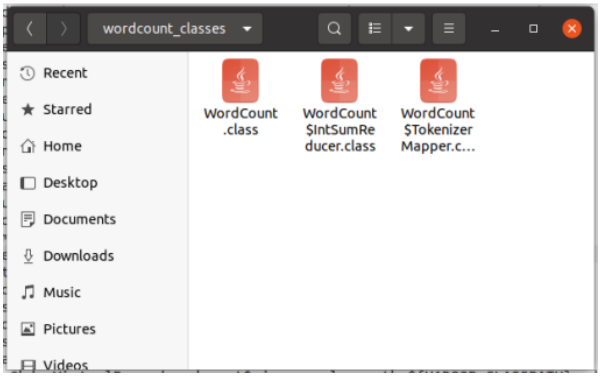
hdoop@bda-VirtualBox:/home/bda$ hdfs dfs -mkdir /wordcount/input/
(now, move local file to hdoop folder)
hdoop@bda-VirtualBox:/home/bda$ hdfs dfs –put '/home/bda/wordcount /input.txt'/wordcount/input/



hdoop@bda-VirtualBox:/home/bda$ hdfs dfs -cat /wordcount/input/*
2021-06-26 13:16:53,156 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

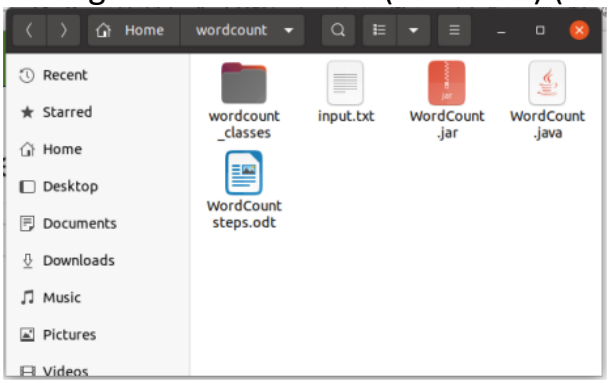
```
Welcome
To
To
The
The
The
World
Of
Information
Technology
Information
Technology
```

hdoop@bda-VirtualBox:/home/bda$ su bda
password
bda@bda-VirtualBox:~$ cd wordcount
bda@bda-VirtualBox:~/wordcount$ javac -classpath ${HADOOP_CLASSPATH} -d '/home/bda/wordcount/wordcount_classes''/home/bda/wordcount/WordCount.java'
check your wordcount_classes folder, which now has three classes in it: WordCount.class, WordCount$IntSumReducer.class and WordCount$TokenizerMapper.class
javac -classpath ${HADOOP_CLASSPATH} -d '/home/bda/wordcount/classes' '/home/bda/wordcount/WordCount.java'
check your wordcount_classes folder, which now has three classes in it: WordCount.class, WordCount$IntSumReducer.class and WordCount$TokenizerMapper.class

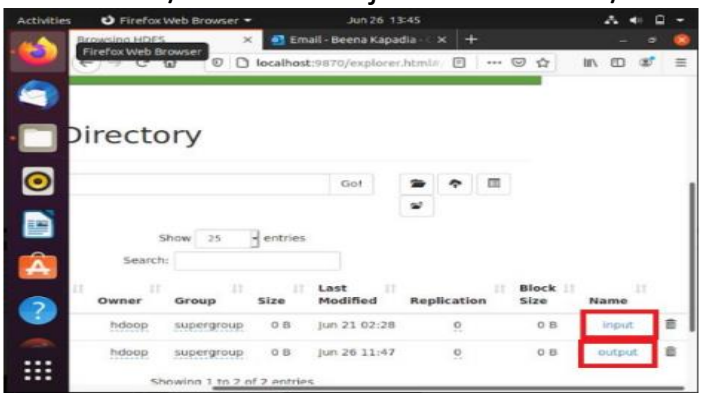


bda@bda-VirtualBox:~/wordcount$ jar -cvf WordCount.jar –C '/home/bda/wordcount/classes'/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1755) (out= 750)(deflated 57%)

adding: WordCount$TokenizerMapper.class(in = 1752) (out= 761)(deflated 56%)
adding: WordCount.class(in = 1511) (out= 832)(deflated 44%)



bda@bda-VirtualBox:~/wordcount$ jar -cvf WordCount.jar -C '/home/bda/wordcount/wordcount_classes'/ .
added manifest
adding: WordCount$IntSumReducer.class(in = 1755) (out= 750)(deflated 57%)
adding: WordCount$TokenizerMapper.class(in = 1752) (out= 761)(deflated 56%)
adding: WordCount.class(in = 1511) (out= 832)(deflated 44%)
hdoop@bda-VirtualBox:/home/bda/wordcount$ hadoop jar '/home/bda/wordcount/WordCount.jar' WordCount /wordcount/input//wordcount/output



Get the output:
hdoop@bda-VirtualBox:/home/bda/wordcount$ hdfs dfs –cat /wordcount /output/*
2021-06-26 13:33:57,781 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

```
Welcome   1
To        2
The       3
World     1
Of        1
Information 2
Technology  2
```

hdoop@bda-VirtualBox:/home/bda/wordcount$

Practical 3

Configure the Hive and implement the application in Hive

Steps for hive installation

1. Download and Unzip Hive

```
tar -xvf apache-hive-1.2.2-bin.tar.gz
```

2. Edit .bashrc file

```
$ sudo nano ~/.bashrc
```

3. Edit hive-config.sh file

```
user@ubuntu:~$ cd  /usr/lib/hive/apache-hive-0.13.0-bin/bin
user@ubuntu:~$ sudo gedit hive-config.sh
```

4. Create Hive directories in HDFS

```
user@ubuntu:~$   hadoop fs -mkdir /usr/hive/warehouse
```

5. Initiate Derby database

```
hive> create database mydb;
```

6. Configure hive-site.xml file

```
<property>
    <name>hive.metastore.local</name>
    <value>TRUE</value>
    <description>controls whether to connect to remove metastore server or open a new metastore server
</property>

<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://usr/lib/hive/apache-hive-0.13.0-bin/metastore_db? createDatabaseIfNotExist=true
    <description>JDBC connect string for a JDBC metastore</description>
</property>

<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>Driver class name for a JDBC metastore</description>
</property>

<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/usr/hive/warehouse</value>
    <description>location of default database for the warehouse</description>
 </property>
```

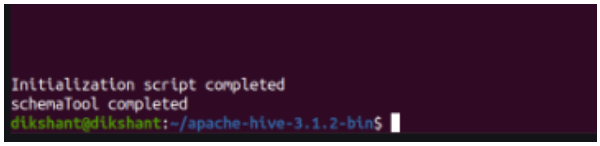
Fixing guava problem – Additional step
rm $HIVE_HOME/lib/guava-19.0.jar cp $HADOOP_HOME/share/Hadoop /hdfs/lib/guava-27.0-jre.jar $HIVE_HOME/lib/

```
ls $HADOOP_HOME/share/hadoop/hdfs/lib
```


```
cp $HADOOP_HOME/share/hadoop/hdfs/lib/guava-27.0-jre.jar $HIVE_HOME/lib/
```

Initialize Derby and hive

```
bin/schematool -dbType derby -initSchema
```



optional Step – Edit hive-site.xml

```
<property>
  <name>hive.metastore.warehouse.dir</name>
  <value>/Users/user/hive/hive</value>
  <description>location of default database for the warehouse</description>
</property>
```

Practical 4

Implement an application that stores big data in Hbase / MongoDB & manipulate it using R / Python

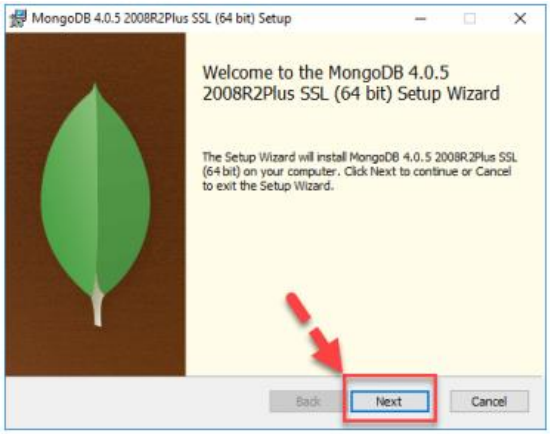
Requirement

a. PyMongo
b. Mongo Database

Step A: Install Mongo database

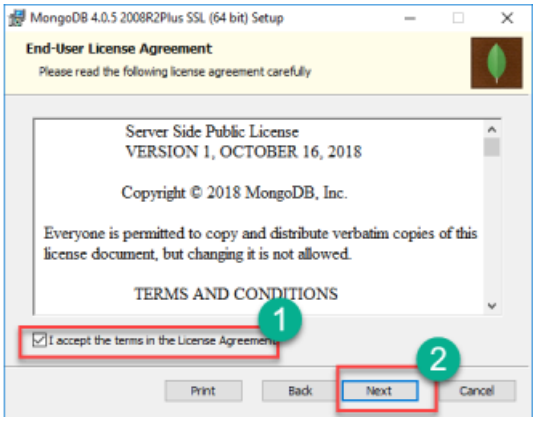
Step 1-Go to (https://www.mongodb.com/download-center/community) and DownloadMongoDB Community Server. We will install the 64-bit version for Windows.

Step 2- Once download is complete open the msi file. Click Next in the start up screen

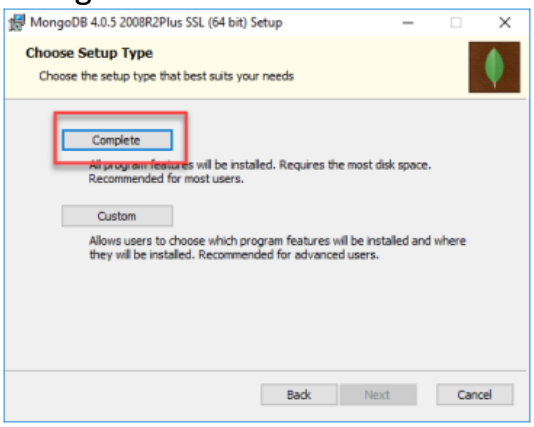


Step 3-

1. Accept the End-User License Agreement
2. Click Next

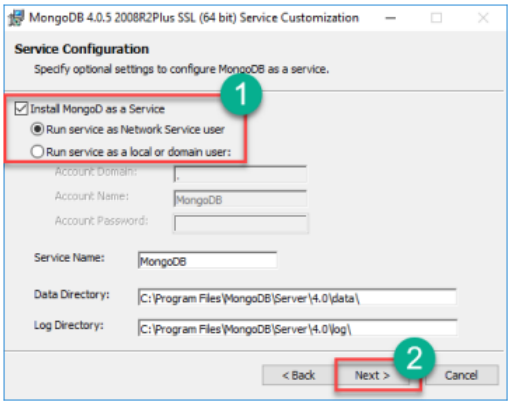


Step 4- Click on the "complete" button to install all of the components. The custom option can be used to install selective components or if you want to change the location of the installation.

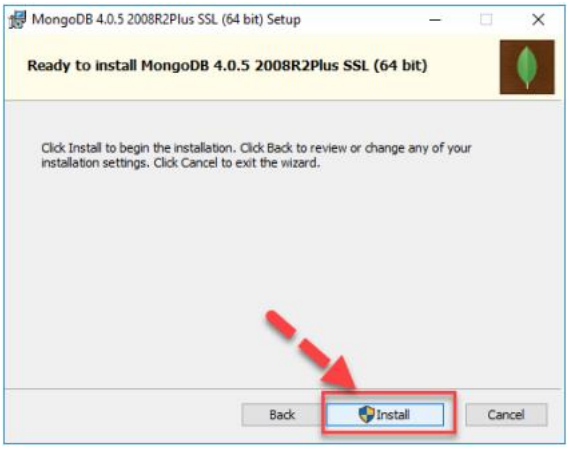


Step 5-

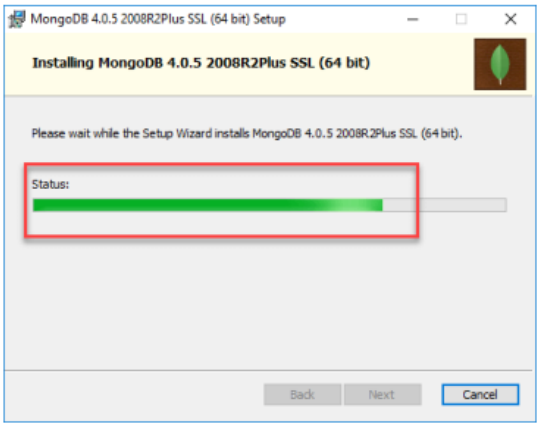
1. Select “Run service as Network Service user”. make a note of the data directory, we’ll need this later.
2. Click Next



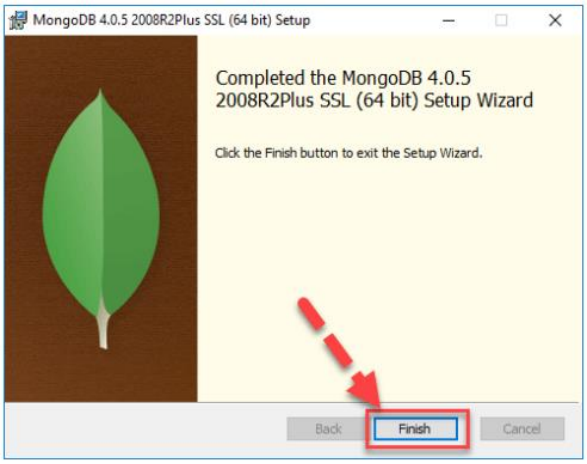
Step 6- Click on the Install button to start the installation.



Step 7-Installation begins. Click Next once completed



Step 8-Click on the Finish button to complete the installation



Test Mongodb

Step 1) Go to " C:\Program Files\MongoDB\Server\4.0\bin" and double click on mongo.exe. Alternatively, you can also click on the MongoDB desktop item

- ☆ Create the directory where MongoDB will store it's files. From the command prompt run md \data\db . This is the default location. However, other locations can be specified using the --dbpath parameter. See the Mongo docs for more information.
  - C:\>md data
  - C:\md data\db
  - C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe –dbpath "C:\data"
- ☆ Start the mongodb daemon by running C:\mongodb\bin\mongod.exe in the Command Prompt. Or by running, C:\path\to\mongodb\bin \mon god.exe
- ☆ Connect to MongoDB using the Mongo shell While the MongoDB daemon is running, from a different Command prompt window run C:\mongodb\bin\mongo.exe
- ☆ C:\Program Files\MongoDB\Server\4.05\bin>mongod.exe --dbpath "C:\data"
- ☆ C:\Program Files\MongoDB\Server\4.05\bin>mongo.exe

Step B: Install PyMongo

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>python –m pip install pymongo

Now you have downloaded and installed a mongoDB driver.

Test PyMongo

demo_mongodb_test.py:

```
import pymongo
```

Program 1: Creating a Database

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
print(myclient.list_database_names())
```

Progam 2: Creating a Collection

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb[“student’]
print(mydb.list_collection_names())
```

Progam 3: Insert into Collection

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb[“student’]
mydict={“name”:”Kaushal”, “address”:”Mumbai”}
x=mycol.insert_one(mydict) # insert_one(containing the name(s) and value(s) of each field
```

Program 4: Insert Multiple data into Collection

```
import pymongo
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mybigdata"]
mycol=mydb[“student’]
mylist=[{“name”:”Kaushal”, “address”:”Mumbai”}, {“name”:”A”, “address”:”Mumbai”}, {“name”:”B”, “address”:”Pune”}, {“name”:”C”, “address”:”Pune”},]
x=mycol.insert_many(mylist)
```

Test in Mongodb to check database and data inserted in collection

a. If you want to check your database list, use the command show dbs in mongo command prompt
b. If you want to use a database with name mybigdata, then use database statement would be as follow: use mybigdata

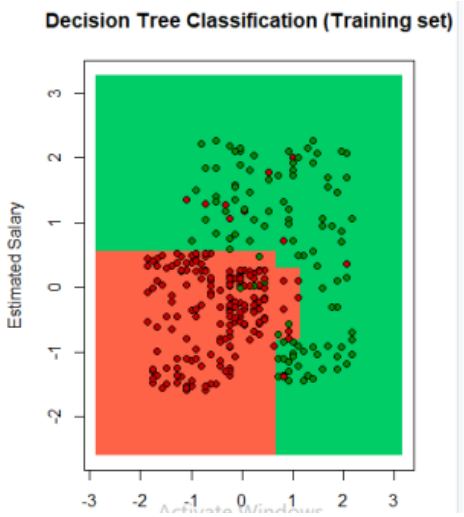
c. If you want to check collection in mongodb use the command show collections
d. If you want to display all the data from collection: db.collection_name.find() ordb.collection_name.find().pretty()

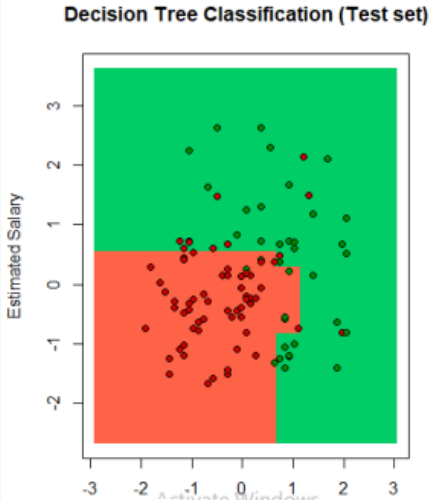
Practical 5
Decision tree classification techniques

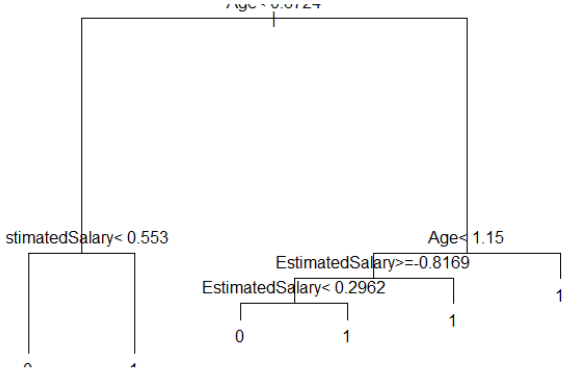
A. Decision Tree

```
install.packages("C:/Users/User-18/Downloads/ElemStatLearn_2015.6.26.
2.tar.gz", repos = NULL, type = "source")
* DONE (ElemStatLearn)
> dataset = read.csv('C:/Users/User-18/Desktop/MSc Sem 2 Practicals/Social-
Network-ads-Boost-master/Social_Network_Ads.csv')
> dataset = dataset[3:5]
> dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
> install.packages('caTools')
The downloaded binary packages are in
> library(caTools)
> set.seed(123)
> split = sample.split(dataset$Purchased, SplitRatio = 0.75)
> training_set = subset(dataset, split == TRUE)
> test_set = subset(dataset, split == FALSE)
> training_set[-3] = scale(training_set[-3])
> test_set[-3] = scale(test_set[-3])
> install.packages('rpart')
The downloaded binary packages are in
> library(rpart)
> classifier = rpart(formula = Purchased ~ .,+ data = training_set)
> y_pred = predict(classifier, newdata = test_set[-3], type = 'class')
> cm = table(test_set[, 3], y_pred)
> install.packages("ElemStatLearn")
> library(ElemStatLearn)
> set = training_set
> X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
> X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
> grid_set = expand.grid(X1, X2)
> colnames(grid_set) = c('Age', 'EstimatedSalary')
> y_grid = predict(classifier, newdata = grid_set, type = 'class')
> plot(set[, -3],
     + main = 'Decision Tree Classification (Training set)',
     + xlab = 'Age', ylab = 'Estimated Salary',
     + xlim = range(X1), ylim = range(X2))
> contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
```

```
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
>
 > library(ElemStatLearn)
> set = test_set
> X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
> X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
> grid_set = expand.grid(X1, X2)
> colnames(grid_set) = c('Age', 'EstimatedSalary')
> y_grid = predict(classifier, newdata = grid_set, type = 'class')
> plot(set[, -3], main = 'Decision Tree Classification (Test set)',
      + xlab = 'Age', ylab = 'Estimated Salary',
      + xlim = range(X1), ylim = range(X2))
> contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
> plot(classifier)
> text(classifier)
```







B. Naïve Bayes Classification

```
dataset = read.csv('C:/Users/User-18/Desktop/MSc Sem 2 Practicals/Social-
Network-ads-Boost-master/Social_Network_Ads.csv')
> dataset = dataset[3:5]
> dataset$Purchased = factor(dataset$Purchased, levels = c(0, 1))
> install.packages("caTools")
Installing package into ‘C:/Users/User-18/AppData/Local/R/win-library/4.2’
(as ‘lib’ is unspecified)
trying URL
'https://cran.rstudio.com/bin/windows/contrib/4.2/caTools_1.18.2.zip'
Content type 'application/zip' length 246244 bytes (240 KB)
downloaded 240 KB
package ‘caTools’ successfully unpacked and MD5 sums checked
The downloaded binary packages are in
C:\Users\User-18\AppData\Local\Temp\RtmpqixA2I\downloaded_packages
> library(caTools)
> set.seed(123)
> split = sample.split(dataset$Purchased, SplitRatio = 0.75)
> training_set = subset(dataset, split == TRUE)
> test_set = subset(dataset, split == FALSE)
> training_set[-3] = scale(training_set[-3])
> test_set[-3] = scale(test_set[-3])
> install.packages('e1071')
The downloaded binary packages are in
C:\Users\User-18\AppData\Local\Temp\RtmpqixA2I\downloaded_packages
> library(e1071)
> classifier = naiveBayes(x = training_set[-3],
                          + y = training_set$Purchased)
> y_pred = predict(classifier, newdata = test_set[-3])
> cm = table(test_set[, 3], y_pred)
> print(cm)
y_pred
0  1
0 57  7
1  7 29
> library(ElemStatLearn)
> set = training_set
> print(set)
Age EstimatedSalary Purchased
1   -1.76554750    -1.47334137         0
```

3	-1.09629664	-0.78837605	0
6	-1.00068938	-0.36027273	0
7	-1.00068938	0.38177303	0
8	-0.52265305	2.26542765	1
10	-0.23583125	-0.16049118	0
11	-1.09629664	0.26761214	0
13	-1.66994024	0.43885347	0
14	-0.52265305	-1.50188159	0
15	-1.86115477	0.32469259	0
16	-0.80947485	0.26761214	0
17	0.91145593	-1.30210004	1
21	0.72024140	-1.38772071	1
23	1.00706320	-0.84545650	1
24	0.72024140	-1.38772071	1
25	0.81584866	-1.35918049	1
26	0.91145593	-1.44480115	1
27	1.10267046	-1.21647938	1
28	0.91145593	-1.15939893	1
30	-0.61826032	-1.50188159	0
31	-0.61826032	0.09637081	0
33	-1.57433297	-1.55896204	0
36	-0.23583125	-1.24501960	0
37	-0.42704579	-1.21647938	0
39	-1.09629664	0.03929037	0
40	-1.00068938	-1.13085871	0
41	-1.00068938	-1.53042182	0
42	-0.42704579	-0.56005428	0
43	-0.23583125	1.06673835	0
44	-0.71386758	-1.58750226	0
47	-1.19190391	0.23907192	0
49	-0.71386758	1.83732433	1
50	-0.61826032	0.52447414	0
51	-1.28751117	-1.10231849	0
53	-0.80947485	0.35323281	0
54	-0.23583125	-1.35918049	0
55	-1.00068938	-0.36027273	0
56	-1.28751117	-0.44589340	0
57	-1.38311844	-0.64567495	0
58	-0.90508211	0.23907192	0
59	-1.47872570	-1.50188159	0

60	-0.52265305	1.32360034	0
61	-1.00068938	-1.44480115	0
62	-1.19190391	0.46739370	0
63	-1.38311844	-0.13195096	0
64	-0.52265305	1.40922101	1
65	2.05874311	0.35323281	0
67	-1.28751117	-1.47334137	0
68	-1.38311844	0.32469259	0
70	-0.61826032	-0.07487051	0
71	-1.19190391	0.26761214	0
72	-1.28751117	-1.24501960	0
73	-1.66994024	-1.35918049	0
76	-0.33143852	1.18089923	1
77	-1.86115477	-0.53151406	0
78	-1.47872570	-1.24501960	0
79	-0.90508211	0.46739370	0
80	-1.09629664	-1.53042182	0
81	-0.71386758	0.26761214	0
83	-1.66994024	-0.61713472	0
88	-0.90508211	0.41031325	0
90	-0.23583125	-0.58859450	0
91	-1.47872570	0.29615237	0
92	-0.71386758	1.29506012	0
93	-1.09629664	-1.58750226	0
94	-0.80947485	-1.21647938	0
95	-0.80947485	0.35323281	0
96	-0.23583125	-0.75983583	0
97	-0.23583125	-1.30210004	0
98	-0.90508211	1.49484167	1
99	-0.23583125	0.06783059	0
100	-0.90508211	-0.95961738	0
101	-1.00068938	0.49593392	0
102	-0.90508211	-0.33173251	0
105	-1.76554750	-1.41626093	0
106	-1.57433297	0.03929037	0
110	0.05099054	0.26761214	0
111	0.14659781	0.01075015	0
112	-0.04461672	0.01075015	0
113	0.05099054	-0.27465207	0
114	-0.04461672	-0.44589340	0

115	0.43341960	0.26761214	0
116	0.24220507	-0.38881295	0
118	-0.14022399	-0.53151406	0
119	0.24220507	-0.33173251	0
120	0.33781234	-0.33173251	0
121	-0.14022399	0.12491104	0
122	-0.04461672	0.03929037	0
123	0.24220507	0.12491104	0
125	0.33781234	-0.56005428	0
128	-1.09629664	-1.10231849	0
129	-0.71386758	-1.53042182	0
130	-1.09629664	0.38177303	0
132	-0.42704579	-1.13085871	0
133	-0.71386758	0.46739370	0
135	-0.90508211	-0.44589340	0
136	-1.38311844	-0.21757162	0
137	-1.66994024	0.32469259	0
138	-0.71386758	1.03819813	1
140	-1.76554750	-1.30210004	0
141	-1.76554750	0.41031325	0
142	-1.86115477	-0.07487051	0
143	-0.23583125	-0.33173251	0
144	-0.71386758	0.52447414	0
145	-0.33143852	-1.30210004	0
146	-1.28751117	0.52447414	0
147	-1.00068938	0.72425569	1
149	-0.80947485	-0.27465207	0
150	-1.66994024	0.09637081	0
151	-1.09629664	-1.58750226	0
152	0.33781234	-0.73129561	0
153	-0.61826032	0.15345126	0
155	0.24220507	-0.67421517	0
157	0.81584866	-0.33173251	0
158	-0.80947485	0.12491104	0
160	-0.52265305	1.83732433	1
161	-0.52265305	0.83841658	1
164	-0.23583125	-0.93107716	0
165	-0.42704579	-0.04633029	0
166	-1.86115477	0.43885347	0
167	-1.47872570	-0.44589340	0

168	-0.23583125	0.01075015	0
169	-0.80947485	2.20834721	1
171	-1.57433297	0.49593392	0
172	-0.33143852	1.26651990	0
173	-1.09629664	1.35214056	0
174	-0.33143852	-0.78837605	0
177	-0.23583125	-0.67421517	0
178	-1.19190391	-1.38772071	0
179	-1.28751117	-1.35918049	0
180	-0.61826032	-1.04523805	0
181	-1.09629664	-1.55896204	0
182	-0.61826032	0.01075015	0
183	-0.52265305	1.32360034	1
184	-0.42704579	-0.78837605	0
185	-0.42704579	-0.30319229	0
186	-0.61826032	-0.13195096	0
187	-1.66994024	0.32469259	0
188	-0.42704579	-0.84545650	0
189	-0.23583125	0.03929037	0
190	-0.90508211	-1.10231849	0
191	-1.28751117	0.38177303	0
192	-1.76554750	-1.27355982	0
194	-1.76554750	-0.01779007	0
195	-0.90508211	0.52447414	0
196	-0.33143852	-0.78837605	0
197	-0.71386758	0.23907192	0
198	-1.66994024	-0.98815761	0
201	-0.23583125	-0.90253694	0
202	1.10267046	0.09637081	0
203	0.14659781	1.80878411	1
204	0.33781234	0.01075015	0
205	1.96313585	0.86695680	1
206	0.91145593	-0.67421517	0
207	1.67631405	1.69462322	1
209	0.24220507	2.03710588	1
210	0.81584866	-1.38772071	0
211	1.00706320	0.72425569	1
212	1.38949226	2.26542765	1
214	-0.23583125	-0.36027273	0
215	0.91145593	-0.78837605	0

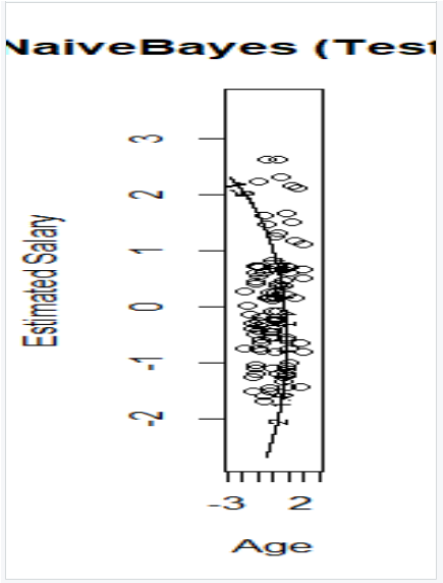
216	2.15435038	1.06673835	1
217	1.10267046	-0.16049118	0
218	0.24220507	0.21053170	0
219	0.81584866	0.72425569	0
220	2.05874311	2.06564610	1
221	0.33781234	0.26761214	0
222	-0.23583125	0.58155458	1
223	-0.04461672	2.09418633	1
225	-0.23583125	-0.30319229	0
227	-0.14022399	1.58046234	1
231	-0.23583125	2.17980699	1
232	0.14659781	-0.81691628	0
233	0.24220507	1.03819813	1
235	0.05099054	1.18089923	0
238	-0.04461672	0.26761214	0
240	1.48509952	2.06564610	1
242	0.05099054	-0.33173251	0
243	1.19827773	0.49593392	1
244	1.77192132	0.95257746	1
245	0.33781234	0.03929037	0
246	1.29388499	2.15126677	1
247	-0.23583125	-0.58859450	0
248	1.86752858	1.46630145	1
249	0.33781234	-0.53151406	0
250	-0.23583125	0.75279591	1
251	0.62463413	-0.90253694	0
252	-0.04461672	-0.53151406	0
253	1.00706320	1.80878411	1
254	-0.04461672	2.15126677	1
256	1.38949226	0.55301436	1
257	0.33781234	0.03929037	0
258	0.24220507	-0.38881295	0
259	1.96313585	0.69571547	1
260	0.72024140	1.72316344	1
261	-0.23583125	0.18199148	0
262	-0.14022399	2.09418633	1
263	1.67631405	1.55192212	1
267	0.24220507	0.12491104	0
268	-0.04461672	0.09637081	0
269	0.91145593	2.09418633	1

270	0.24220507	-0.27465207	0
271	0.52902687	1.78024389	0
272	2.05874311	0.15345126	1
275	1.86752858	-1.27355982	1
276	1.86752858	0.09637081	1
277	0.05099054	0.01075015	0
278	1.10267046	0.49593392	1
279	1.38949226	-0.93107716	1
280	1.19827773	-0.98815761	1
282	-0.23583125	-0.27465207	0
283	-0.04461672	-0.01779007	1
284	1.38949226	-1.41626093	1
285	1.00706320	2.00856566	0
287	-0.04461672	-0.24611184	0
288	1.00706320	1.92294500	1
289	0.33781234	0.23907192	0
290	-0.04461672	0.21053170	1
291	0.14659781	1.80878411	1
293	1.67631405	-0.90253694	1
294	-0.04461672	0.18199148	0
295	-0.23583125	-0.38881295	0
296	-0.14022399	-0.21757162	0
297	0.43341960	0.06783059	1
298	0.52902687	1.18089923	1
300	0.81584866	1.32360034	1
301	1.96313585	-0.93107716	1
303	-0.04461672	1.89440477	1
304	-0.04461672	0.23907192	1
306	0.43341960	-0.47443362	0
308	0.91145593	1.20943946	1
309	-0.14022399	1.55192212	1
311	0.43341960	-0.01779007	0
312	0.14659781	0.72425569	1
313	0.05099054	-0.58859450	0
314	1.10267046	2.00856566	1
315	0.14659781	0.23907192	0
317	1.58070679	0.95257746	1
318	-0.23583125	-0.44589340	0
319	0.72024140	-1.10231849	1
320	-0.14022399	-0.30319229	0

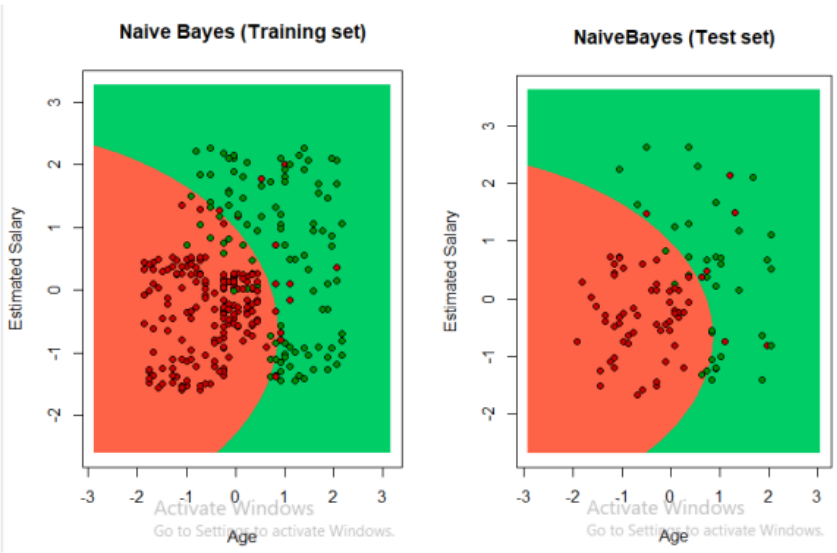
321	1.38949226	1.92294500	1
322	1.48509952	0.32469259	1
323	0.33781234	-0.53151406	0
325	1.00706320	1.72316344	1
327	0.33781234	0.03929037	0
328	0.43341960	0.12491104	0
329	-0.14022399	1.35214056	1
330	0.91145593	1.03819813	1
331	0.05099054	-0.56005428	0
333	0.43341960	-0.16049118	0
334	0.24220507	-0.16049118	0
335	1.86752858	-0.30319229	1
336	-0.14022399	-0.47443362	0
337	1.96313585	2.09418633	1
338	-0.23583125	0.23907192	0
340	0.14659781	1.46630145	1
342	-0.23583125	0.12491104	0
344	0.91145593	-0.56005428	1
345	0.91145593	0.98111768	1
346	0.33781234	-0.21757162	0
348	1.58070679	1.06673835	1
349	0.14659781	0.18199148	0
350	0.05099054	-0.27465207	0
351	0.05099054	1.20943946	1
352	-0.04461672	0.12491104	0
354	-0.04461672	-0.38881295	0
355	-0.14022399	0.80987635	1
356	2.15435038	-1.04523805	1
357	1.58070679	-0.01779007	1
358	0.33781234	0.03929037	0
359	0.24220507	0.01075015	1
360	0.43341960	-0.47443362	0
361	0.52902687	1.66608300	1
362	1.48509952	-1.04523805	1
365	0.43341960	0.95257746	1
366	2.05874311	-1.18793916	1
370	1.58070679	-1.27355982	1
371	2.15435038	-0.70275539	1
374	2.05874311	1.69462322	1
375	-0.04461672	0.26761214	0

```
376  0.81584866    -1.10231849        1
377  0.81584866     0.09637081        0
378  0.43341960    -0.50297384        0
379  0.33781234     0.46739370        1
381  0.43341960    -0.18903140        0
382  1.00706320    -1.07377827        1
384  1.10267046    -1.21647938        1
385  1.86752858    -1.07377827        1
386  1.77192132    -0.30319229        1
387  1.10267046    -0.90253694        1
388  0.14659781     0.01075015        0
390  1.00706320    -1.01669783        1
391  1.00706320    -1.07377827        1
393  0.72024140    -0.73129561        1
394  2.15435038    -0.81691628        1
396  0.81584866    -0.84545650        1
397  1.29388499    -1.35918049        1
398  1.19827773    -1.44480115        1
399 -0.14022399    -1.07377827        0
> X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
> X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
> grid_set = expand.grid(X1, X2)
> colnames(grid_set) = c('Age', 'EstimatedSalary')
> y_grid = predict(classifier, newdata = grid_set)
> plot(set[, -3],
     + main = 'Naive Bayes (Training set)',
     + xlim = range(X1), ylim = range(X2))
> contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add =
TRUE)
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))
> library(ElemStatLearn)
> set = test_set
> X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
> X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
> grid_set = expand.grid(X1, X2)
> colnames(grid_set) = c('Age', 'EstimatedSalary')
> y_grid = predict(classifier, newdata = grid_set)
> plot(set[, -3], main = 'NaiveBayes (Test set)',
     + xlab = 'Age', ylab = 'Estimated Salary',
```

+ xlim = range(X1), ylim = range(X2))
> contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
>
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))




```
> classifier = naiveBayes(x = training_set[-3],
+                         y = training_set$Purchased)
> # Predicting the Test set results
> y_pred = predict(classifier, newdata = test_set[-3])
> # Making the Confusion Matrix
> cm = table(test_set[, 3], y_pred)
> # Making the Confusion Matrix
> cm = table(test_set[, 3], y_pred)
> print(cm)
   y_pred
     0  1
  0 57  7
  1  7 29
```



Practical 6
SVM classification techniques

Apriori algorithm

Code:

```
> install.packages("arules")
package ‘generics’ successfully unpacked and MD5 sums checked
package ‘arules’ successfully unpacked and MD5 sums checked
> install.packages("arulesViz")
package ‘fs’ successfully unpacked and MD5 sums checked
package ‘rappdirs’ successfully unpacked and MD5 sums checked
package ‘sass’ successfully unpacked and MD5 sums checked
package ‘cachem’ successfully unpacked and MD5 sums checked
package ‘memoise’ successfully unpacked and MD5 sums checked
package ‘sys’ successfully unpacked and MD5 sums checked
package ‘iterators’ successfully unpacked and MD5 sums checked
package ‘evaluate’ successfully unpacked and MD5 sums checked
package ‘highr’ successfully unpacked and MD5 sums checked
package ‘xfun’ successfully unpacked and MD5 sums checked
package ‘bslib’ successfully unpacked and MD5 sums checked
package ‘tinytex’ successfully unpacked and MD5 sums checked
package ‘askpass’ successfully unpacked and MD5 sums checked
package ‘foreach’ successfully unpacked and MD5 sums checked
package ‘tweenr’ successfully unpacked and MD5 sums checked
package ‘polyclip’ successfully unpacked and MD5 sums checked
package ‘systemfonts’ successfully unpacked and MD5 sums checked
package ‘RcppEigen’ successfully unpacked and MD5 sums checked
package ‘RcppArmadillo’ successfully unpacked and MD5 sums checked
package ‘stringi’ successfully unpacked and MD5 sums checked
package ‘fastmap’ successfully unpacked and MD5 sums checked
package ‘ellipsis’ successfully unpacked and MD5 sums checked
package ‘yaml’ successfully unpacked and MD5 sums checked
package ‘knitr’ successfully unpacked and MD5 sums checked
package ‘rmarkdown’ successfully unpacked and MD5 sums checked
package ‘later’ successfully unpacked and MD5 sums checked
package ‘curl’ successfully unpacked and MD5 sums checked
package ‘mime’ successfully unpacked and MD5 sums checked
package ‘openssl’ successfully unpacked and MD5 sums checked
package ‘TSP’ successfully unpacked and MD5 sums checked
package ‘qap’ successfully unpacked and MD5 sums checked
package ‘gclus’ successfully unpacked and MD5 sums checked
```

```
package ‘ca’ successfully unpacked and MD5 sums checked
package ‘registry’ successfully unpacked and MD5 sums checked
package ‘lmtest’ successfully unpacked and MD5 sums checked
package ‘ggforce’ successfully unpacked and MD5 sums checked
package ‘digest’ successfully unpacked and MD5 sums checked
package ‘ggrepel’ successfully unpacked and MD5 sums checked
package ‘viridis’ successfully unpacked and MD5 sums checked
package ‘tidygraph’ successfully unpacked and MD5 sums checked
package ‘graphlayouts’ successfully unpacked and MD5 sums checked
package ‘purrr’ successfully unpacked and MD5 sums checked
package ‘stringr’ successfully unpacked and MD5 sums checked
package ‘tidyselect’ successfully unpacked and MD5 sums checked
package ‘cpp11’ successfully unpacked and MD5 sums checked
package ‘htmltools’ successfully unpacked and MD5 sums checked
package ‘htmlwidgets’ successfully unpacked and MD5 sums checked
package ‘jsonlite’ successfully unpacked and MD5 sums checked
package ‘crosstalk’ successfully unpacked and MD5 sums checked
package ‘jquerylib’ successfully unpacked and MD5 sums checked
package ‘promises’ successfully unpacked and MD5 sums checked
package ‘httr’ successfully unpacked and MD5 sums checked
package ‘base64enc’ successfully unpacked and MD5 sums checked
package ‘lazyeval’ successfully unpacked and MD5 sums checked
package ‘data.table’ successfully unpacked and MD5 sums checked
package ‘seriation’ successfully unpacked and MD5 sums checked
package ‘vcd’ successfully unpacked and MD5 sums checked
package ‘igraph’ successfully unpacked and MD5 sums checked
package ‘scatterplot3d’ successfully unpacked and MD5 sums checked
package ‘ggraph’ successfully unpacked and MD5 sums checked
package ‘tidyr’ successfully unpacked and MD5 sums checked
package ‘dplyr’ successfully unpacked and MD5 sums checked
package ‘DT’ successfully unpacked and MD5 sums checked
package ‘plotly’ successfully unpacked and MD5 sums checked
package ‘visNetwork’ successfully unpacked and MD5 sums checked
package ‘arulesViz’ successfully unpacked and MD5 sums checked
> install.packages("RColorBrewer")
package ‘RColorBrewer’ successfully unpacked and MD5 sums checked
> library(arules)
> library(arulesViz)
> library(RColorBrewer)
> data(Groceries)
```

```
> Groceries
transactions in sparse format with
9835 transactions (rows) and
169 items (columns)
> summary(Groceries)
transactions as itemMatrix in sparse format with
9835 rows (elements/itemsets/transactions) and
169 columns (items) and a density of 0.02609146
most frequent items:
  whole milk other vegetables       rolls/buns             soda
2513            1903            1809            1715
yogurt         (Other)
1372           34055

element (itemset/transaction) length distribution:
  sizes
1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
17   18   19   20   21   22   23   24   26   27   28   29   32
29   14   14    9   11    4    6    1    1    1    1    3    1
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   2.000   3.000   4.409   6.000  32.000
includes extended item information - examples:
  labels  level2          level1
1 frankfurter sausage meat and sausage
2     sausage sausage meat and sausage
3  liver loaf sausage meat and sausage
> class(Groceries)
[1] "transactions"
attr(,"package")
[1] "arules"
> rules = apriori(Groceries, parameter = list(supp = 0.02, conf = 0.2))
Apriori Parameter specification:
  confidence minval smax arem  aval originalSupport maxtime support minlen
0.2    0.1    1 none FALSE            TRUE       5    0.02      1
maxlen target  ext
10  rules TRUE
Algorithmic control:
  filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

```
Absolute minimum support count: 196
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [73 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
> summary (rules)
set of 73 rules
rule length distribution (lhs + rhs):sizes
1  2  3
1 66  6
Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
1.000   2.000   2.000   2.068   2.000   3.000
summary of quality measures:
  support         confidence       coverage         lift
Min.   :0.02003   Min.   :0.2006   Min.   :0.04342   Min.   :0.8991
1st Qu.:0.02257   1st Qu.:0.2369   1st Qu.:0.07168   1st Qu.:1.3112
Median :0.02664   Median :0.3079   Median :0.09395   Median :1.5570
Mean   :0.03424   Mean   :0.3187   Mean   :0.11739   Mean   :1.6061
3rd Qu.:0.03589   3rd Qu.:0.3868   3rd Qu.:0.11052   3rd Qu.:1.8502
Max.   :0.25552   Max.   :0.5129   Max.   :1.00000   Max.   :2.8421
count
Min.   : 197.0
1st Qu.: 222.0
Median : 262.0
Mean   : 336.8
3rd Qu.: 353.0
Max.   :2513.0
mining info:
  data ntransactions support confidence
Groceries        9835    0.02       0.2
call
apriori(data = Groceries, parameter = list(supp = 0.02, conf = 0.2))
> inspect(rules[1:10])
lhs                 rhs              support    confidence coverage
[1]  {}                => {whole milk}       0.25551601 0.2555160  1.00000000
[2]  {frozen vegetables} => {whole milk}       0.02043721 0.4249471  0.04809354
[3]  {beef}            => {whole milk}       0.02125064 0.4050388  0.05246568
```

```
[4]  {curd}           => {whole milk}      0.02613116 0.4904580  0.05327911
[5]  {pork}           => {other vegetables} 0.02165735 0.3756614  0.05765125
[6]  {pork}           => {whole milk}      0.02216573 0.3844797  0.05765125
[7]  {frankfurter}    => {whole milk}      0.02053889 0.3482759  0.05897306
[8]  {bottled beer}   => {whole milk}      0.02043721 0.2537879  0.08052872
[9]  {brown bread}    => {whole milk}      0.02521607 0.3887147  0.06487036
[10] {margarine}      => {whole milk}      0.02419929 0.4131944  0.05856634
lift     count
[1]  1.0000000 2513
[2]  1.6630940  201
[3]  1.5851795  209
[4]  1.9194805  257
[5]  1.9414764  213
[6]  1.5047187  218
[7]  1.3630295  202
[8]  0.9932367  201
[9]  1.5212930  248
[10] 1.6170980  238
> arules::itemFrequencyPlot(Groceries, topN = 20,
                    + col = brewer.pal(8, 'Pastel2'),
                    + main = 'Relative Item Frequency Plot',
                    + type = "relative",
                    + ylab = "Item Frequency (Relative)")
> itemsets = apriori(Groceries, parameter = list(minlen=2,
maxlen=2,support=0.02, target="frequent
+ itemsets"))
Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
NA    0.1    1 none FALSE          TRUE       5    0.02      2
maxlen          target  ext
2 frequent itemsets TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE    2    TRUE
Absolute minimum support count: 196
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
```

```
checking subsets of size 1 2 done [0.00s].
sorting transactions ... done [0.00s].
writing ... [61 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
 most frequent items:
  whole milk other vegetables          yogurt      rolls/buns
                         25              17               9               9
                         soda         (Other)
                         9              53

                         element (itemset/transaction) length distribution:sizes
                         2
                         61

                         Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
                         2       2       2       2       2       2

                         summary of quality measures:
                           support           count
                         Min.   :0.02003   Min.   :197.0
                         1st Qu.:0.02227   1st Qu.:219.0
                         Median :0.02613   Median :257.0
                         Mean   :0.02951   Mean   :290.3
                         3rd Qu.:0.03223   3rd Qu.:317.0
                         Max.   :0.07483   Max.   :736.0

                         includes transaction ID lists: FALSE

                         mining info:
                           data ntransactions support confidence
                         Groceries         9835    0.02         1
                         call
                         apriori(data = Groceries, parameter = list(minlen = 2,
maxlen = 2, support = 0.02, target = "frequent itemsets"))
                         > inspect(itemsets[1:10])
                         items                       support    count
                         [1]  {whole milk, frozen vegetables} 0.02043721 201
                         [2]  {beef, whole milk}              0.02125064 209
                         [3]  {whole milk, curd}              0.02613116 257
                         [4]  {pork, other vegetables}        0.02165735 213
```

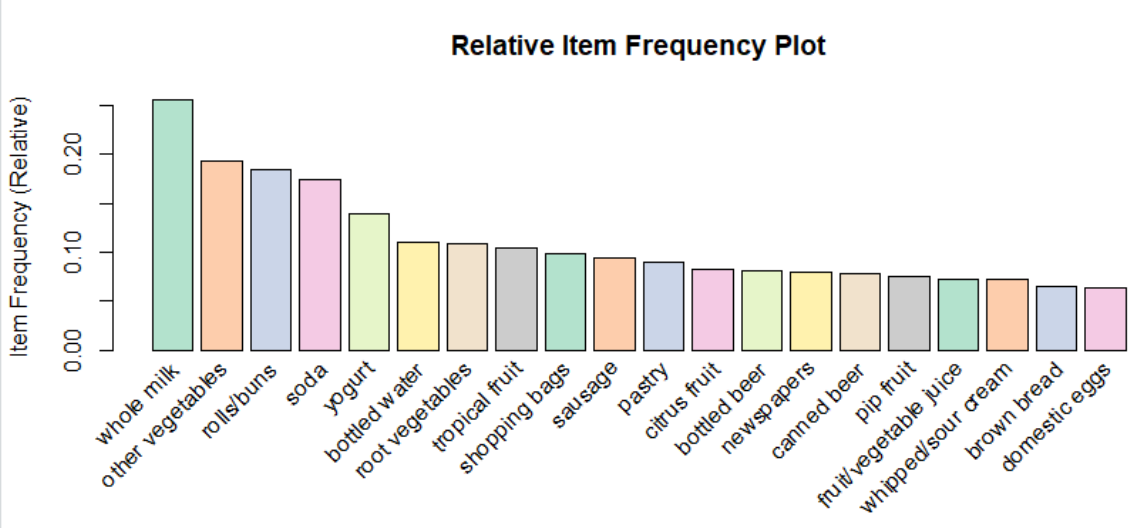
```
[5]  {pork, whole milk}            0.02216573 218
[6]  {frankfurter, whole milk}     0.02053889 202
[7]  {whole milk, bottled beer}    0.02043721 201
[8]  {whole milk, brown bread}     0.02521607 248
[9]  {whole milk, margarine}       0.02419929 238
[10] {other vegetables, butter}    0.02003050 197
> itemsets_3 = apriori(Groceries, parameter =
list(minlen=3, maxlen=3,support=0.02, target="frequent itemsets"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport
maxtime support minlen
NA    0.1    1 none FALSE          TRUE       5    0.02      3
maxlen          target  ext
3 frequent itemsets TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 196

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)]
done [0.00s].
sorting and recoding items ... [59 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
sorting transactions ... done [0.00s].
writing ... [2 set(s)] done [0.00s].
creating S4 object  ... done [0.00s].
Warning message:
 In apriori(Groceries, parameter = list(minlen = 3,
maxlen = 3, support = 0.02,  :
                                    Mining stopped (maxlen
reached). Only patterns up to a length of 3 returned!
                                    > summary(itemsets_3)
                                   set of 2 itemsets
```

```
                                           most frequent items:
                                             other vegetables      whole milk
root vegetables          yogurt
                                           2              2              1              1
                                           frankfurter         (Other)
                                           0              0

                                           element (itemset/transaction)
length distribution:sizes
                                           3
                                           2

                                           Min. 1st Qu.  Median    Mean 3rd
Qu.    Max.
                                           3      3      3      3      3      3

                                           summary of quality measures:
                                             support           count
                                           Min.   :0.02227   Min.   :219.0
                                           1st Qu.:0.02250   1st Qu.:221.2
                                           Median :0.02272   Median :223.5
                                           Mean   :0.02272   Mean   :223.5
                                           3rd Qu.:0.02295   3rd Qu.:225.8
                                           Max.   :0.02318   Max.   :228.0

                                           includes transaction ID lists:
FALSE

                                           mining info:
                                             data ntransactions support
confidence
                                           Groceries         9835    0.02         1
                                           call
                                           apriori(data = Groceries,
parameter = list(minlen = 3, maxlen = 3, support = 0.02, target = "frequent
itemsets"))
                                           > inspect(itemsets_3)
                                           items
support    count
```

[1] {root vegetables, other vegetables, whole milk} 0.02318251 228

[2] {other vegetables, whole milk, yogurt} 0.02226741 219

> lhs rhs support confidence coverage lift count

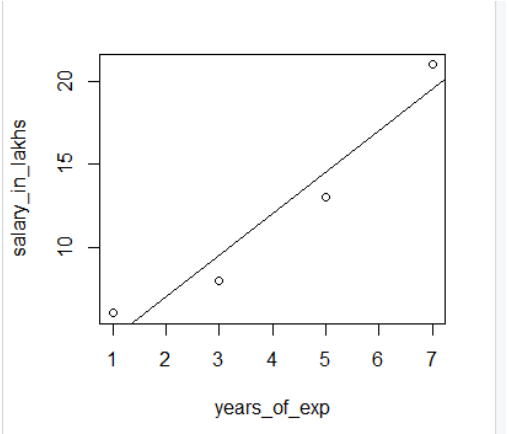


Practical 7
Linear regression

A. Simple Linear regression

Code:

```
> years_of_exp = c(7,5,1,3)
> salary_in_lakhs = c(21,13,6,8)
> employee.data = data.frame(years_of_exp, salary_in_lakhs)
> employee.data
years_of_exp salary_in_lakhs
1            7             21
2            5             13
3            1              6
4            3              8
> model <- lm(salary_in_lakhs ~ years_of_exp, data = employee.data)
> summary(model)
Call:
  lm(formula = salary_in_lakhs ~ years_of_exp, data = employee.data)
Residuals:
  1    2    3    4
1.5 -1.5  1.5 -1.5
Coefficients:
  Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.0000     2.1737    0.92   0.4547
years_of_exp   2.5000     0.4743    5.27   0.0342 *
  ---
  Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 2.121 on 2 degrees of freedom
Multiple R-squared:  0.9328,      Adjusted R-squared:  0.8993
F-statistic: 27.78 on 1 and 2 DF,  p-value: 0.03417
> plot(salary_in_lakhs ~ years_of_exp, data = employee.data)
> abline(model)
```



B. Logistic regression

```
install.packages("ISLR")
The downloaded binary packages are in
> library(ISLR)
> data <- ISLR::Default
> print (head(ISLR::Default))
default student   balance    income
1      No      No  729.5265 44361.625
2      No     Yes  817.1804 12106.135
3      No      No 1073.5492 31767.139
4      No      No  529.2506 35704.494
5      No      No  785.6559 38463.496
6      No     Yes  919.5885  7491.559
> summary(data)
default    student       balance           income
No :9667   No :7056   Min.   :   0.0   Min.   :  772
Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340
                      Median : 823.6   Median :34553
                      Mean   : 835.4   Mean   :33517
                      3rd Qu.:1166.3   3rd Qu.:43808
                      Max.   :2654.3   Max.   :73554
> nrow(data)
[1] 10000
> set.seed(1)
> sample <- sample(c(TRUE, FALSE), nrow(data), replace=TRUE,
prob=c(0.7,0.3))
> print (sample)
  [1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
 [11]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
 [21] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [31]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE
 [41] FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
 [51]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [61] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
 [71]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE
 [81]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
 [91]  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE
[101]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[111] FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[121] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

[131] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE
[141] TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE TRUE FALSE
[151] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[161] TRUE FALSE TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE
[171] TRUE FALSE FALSE TRUE TRUE FALSE TRUE FALSE TRUE FALSE
[181] TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE
[191] FALSE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
[201] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[211] FALSE TRUE FALSE FALSE FALSE TRUE TRUE FALSE FALSE TRUE
[221] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE
[231] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[241] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[251] FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[261] TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE TRUE TRUE
[271] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[281] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[291] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[301] TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE
[311] TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE FALSE TRUE
[321] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE
[331] FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE
[341] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[351] TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE
[361] TRUE TRUE FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE
[371] TRUE TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
[381] TRUE TRUE TRUE FALSE TRUE TRUE FALSE FALSE TRUE FALSE
[391] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[401] TRUE TRUE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE
[411] FALSE TRUE FALSE FALSE FALSE TRUE TRUE TRUE TRUE FALSE
[421] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
[431] FALSE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE FALSE
[441] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
[451] FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
[461] TRUE TRUE TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE
[471] TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE
[481] FALSE TRUE FALSE FALSE TRUE TRUE TRUE TRUE FALSE TRUE
[491] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
[501] TRUE TRUE TRUE TRUE TRUE FALSE TRUE FALSE FALSE TRUE
[511] FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE
[521] FALSE FALSE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE

[531] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[541] TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE FALSE
[551] FALSE TRUE TRUE FALSE FALSE TRUE TRUE TRUE FALSE FALSE
[561] TRUE TRUE TRUE FALSE FALSE TRUE TRUE TRUE TRUE FALSE
[571] TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE FALSE
[581] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[591] TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE
[601] FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE FALSE TRUE
[611] FALSE TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE
[621] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[631] FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE
[641] FALSE TRUE FALSE TRUE FALSE TRUE TRUE FALSE TRUE FALSE
[651] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[661] TRUE FALSE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE
[671] FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[681] FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE
[691] TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE TRUE
[701] FALSE FALSE FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[711] FALSE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE
[721] TRUE TRUE FALSE FALSE TRUE FALSE TRUE TRUE TRUE TRUE
[731] FALSE TRUE FALSE TRUE FALSE TRUE TRUE FALSE TRUE TRUE
[741] TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE
[751] TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
[761] FALSE TRUE FALSE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
[771] TRUE FALSE TRUE FALSE TRUE TRUE TRUE FALSE TRUE FALSE
[781] FALSE FALSE FALSE TRUE FALSE TRUE TRUE TRUE TRUE TRUE
[791] TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
[801] FALSE TRUE FALSE FALSE TRUE TRUE TRUE FALSE TRUE TRUE
[811] TRUE TRUE TRUE FALSE TRUE TRUE FALSE FALSE TRUE FALSE
[821] TRUE TRUE FALSE FALSE FALSE TRUE TRUE TRUE TRUE FALSE
[831] TRUE FALSE FALSE FALSE TRUE TRUE TRUE FALSE FALSE TRUE
[841] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
[851] FALSE FALSE TRUE FALSE TRUE TRUE TRUE FALSE TRUE TRUE
[861] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE FALSE TRUE
[871] TRUE FALSE FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE
[881] TRUE FALSE FALSE FALSE TRUE FALSE FALSE TRUE TRUE TRUE
[891] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[901] FALSE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[911] TRUE FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE
[921] TRUE FALSE FALSE TRUE FALSE FALSE FALSE TRUE TRUE TRUE

```
[931]  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE
[941]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
[951]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
[961]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE
[971] FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[981]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[991]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
[ reached getOption("max.print") -- omitted 9000 entries ]
> train <- data[sample, ]
> test <- data[!sample, ]
> nrow(train)
[1] 6964
> nrow(test)
[1] 3036
> model <- glm(default~student+balance+income, family="binomial",
data=train)
> summary(model)
Call:
  glm(formula = default ~ student + balance + income, family = "binomial",
data = train)
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.5586  -0.1353  -0.0519  -0.0177   3.7973
Coefficients:
   Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.148e+01  6.234e-01 -18.412   <2e-16 ***
  studentYes  -4.933e-01  2.857e-01  -1.726   0.0843 .
balance      5.988e-03  2.938e-04  20.384   <2e-16 ***
  income       7.857e-06  9.965e-06   0.788   0.4304
  Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Dispersion parameter for binomial family taken to be 1)
Null deviance: 2021.1  on 6963  degrees of freedom
Residual deviance: 1065.4  on 6960  degrees of freedom
AIC: 1073.4
Number of Fisher Scoring iterations: 8
> install.packages("C:/Users/User-
18/Downloads/InformationValue_1.2.3.tar.gz", repos = NULL, type = "source")
* DONE (InformationValue)
> library(InformationValue)
> predicted <- predict(model, test, type="response")
```

> confusionMatrix(test$default, predicted)
No Yes
0 2912 64
1 21 39

```
> summary(model)

Call:
glm(formula = default ~ student + balance + income, family = "binomial",
    data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5586  -0.1353  -0.0519  -0.0177   3.7973

Coefficients:
                Estimate    Std. Error z value             Pr(>|z|)
(Intercept) -11.478101194   0.623409555 -18.412 <0.0000000000000002 ***
studentYes   -0.493292438   0.285735949  -1.726               0.0843 .
balance       0.005988059   0.000293765  20.384 <0.0000000000000002 ***
income        0.000007857   0.000009965   0.788               0.4304
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2021.1  on 6963  degrees of freedom
Residual deviance: 1065.4  on 6960  degrees of freedom
AIC: 1073.4

Number of Fisher Scoring iterations: 8
> confusionMatrix(test$default, predicted)
     0  1
0 2912 64
1   21 39
```

Practical 8
Clustering Model

K means clustering.

```
> install.packages("plyr")
package ‘Rcpp’ successfully unpacked and MD5 sums checked
package ‘plyr’ successfully unpacked and MD5 sums checked
> install.packages("ggplot2")
package ‘colorspace’ successfully unpacked and MD5 sums checked
package ‘utf8’ successfully unpacked and MD5 sums checked
package ‘farver’ successfully unpacked and MD5 sums checked
package ‘labeling’ successfully unpacked and MD5 sums checked
package ‘munsell’ successfully unpacked and MD5 sums checked
package ‘R6’ successfully unpacked and MD5 sums checked
package ‘RColorBrewer’ successfully unpacked and MD5 sums checked
package ‘viridisLite’ successfully unpacked and MD5 sums checked
package ‘fansi’ successfully unpacked and MD5 sums checked
package ‘magrittr’ successfully unpacked and MD5 sums checked
package ‘pillar’ successfully unpacked and MD5 sums checked
package ‘pkgconfig’ successfully unpacked and MD5 sums checked
package ‘cli’ successfully unpacked and MD5 sums checked
package ‘glue’ successfully unpacked and MD5 sums checked
package ‘gtable’ successfully unpacked and MD5 sums checked
package ‘isoband’ successfully unpacked and MD5 sums checked
package ‘lifecycle’ successfully unpacked and MD5 sums checked
package ‘rlang’ successfully unpacked and MD5 sums checked
package ‘scales’ successfully unpacked and MD5 sums checked
package ‘tibble’ successfully unpacked and MD5 sums checked
package ‘vctrs’ successfully unpacked and MD5 sums checked
package ‘withr’ successfully unpacked and MD5 sums checked
package ‘ggplot2’ successfully unpacked and MD5 sums checked
> install.packages("cluster")
package ‘cluster’ successfully unpacked and MD5 sums checked
> install.packages("lattice")
package ‘lattice’ successfully unpacked and MD5 sums checked
> install.packages("grid")
  package ‘grid’ is a base package, and should not be updated
> install.packages("gridExtra")
package ‘gridExtra’ successfully unpacked and MD5 sums checked
> library(plyr)
> library(ggplot2)
```

```
> library(cluster)
> library(lattice)
> library(grid)
> library(gridExtra)
> grade_input=as.data.frame(read.csv("C:/grades_km_input.csv"))
> kmdata_orig=as.matrix(grade_input[, c
("Student","English","Math","Science")])
> kmdata=kmdata_orig[,2:4]
> kmdata[1:10,]
English Math Science
[1,]     99   96      97
[2,]     99   96      97
[3,]     98   97      97
[4,]     95  100      95
[5,]     95   96      96
[6,]     96   97      96
[7,]    100   96      97
[8,]     95   98      98
[9,]     98   96      96
[10,]     99   99      95
> wss=numeric(15)
> for(k in 1:15)wss[k]=sum(kmeans(kmdata,centers=k,nstart=25)$withinss)
> plot(1:15,wss,type="b",xlab="Number of Clusters",ylab="Within sum of
square")
> km = kmeans(kmdata,3,nstart=25)
> km
K-means clustering with 3 clusters of sizes 244, 158, 218
Cluster means:
  English     Math  Science
1 85.84426 79.68033 81.50820
2 97.21519 93.37342 94.86076
3 73.22018 64.62844 65.84862
Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[29] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[57] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[85] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[113] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[141] 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
[169] 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 1 2 1 1 1 1 1 2 1 1 1
```

```
[197] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[225] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[253] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[281] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[309] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[337] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[365] 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 1 3 1 3 1 1 1 3 3
[393] 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[421] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3
[449] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3
[477] 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[505] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[533] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[561] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[589] 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 1 1 1 1 2 2 1 1 1 3 3 1
[617] 3 1 1 1
Within cluster sum of squares by cluster:
  [1] 22984.131  6692.589 34806.339
(between_SS / total_SS =  76.5 %)
Available components:
  [1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> c( wss[3] , sum(km$withinss))
[1] 64483.06 64483.06
> df=as.data.frame(kmdata_orig[,2:4])
> df$cluster=factor(km$cluster)
> centers=as.data.frame(km$centers)
> g1=ggplot(data=df, aes(x=English, y=Math, color=cluster )) +
  +    geom_point() + theme(legend.position="right") +
  +    geom_point(data=centers,aes(x=English,y=Math,
color=as.factor(c(1,2,3))),size=10, alpha=.3,
                +              show.legend =FALSE)
> g2=ggplot(data=df, aes(x=English, y=Science, color=cluster )) +
  +    geom_point () +geom_point(data=centers,aes(x=English,y=Science,
+                                    color=as.factor(c(1,2,3))),size=10, alpha=.3,
show.legend=FALSE)
> g3 = ggplot(data=df, aes(x=Math, y=Science, color=cluster )) +
+    geom_point () + geom_point(data=centers,aes(x=Math,y=Science,
```

```
+                                    color=as.factor(c(1,2,3))),size=10, alpha=.3,
show.legend=FALSE)
> tmp=ggplot_gtable(ggplot_build(g1))
> grid.arrange(arrangeGrob(g1 + theme(legend.position="none"),g2 +
+                          theme(legend.position="none"),g3 +
theme(legend.position="none"),top ="High School Student
+ Cluster Analysis" ,ncol=1))
>
```