

INDEX

Prac. No.	Practical
1	Install, configure and run Hadoop and HDFS
2	Implement Decision tree classification techniques
3	Classification using SVM
4	Implement an application that stores big data in Hbase / MongoDB and manipulate it using R / Python
5	Write Program Naive baye's theorem's
6	Write a Program showing implementation of Regression model.
7	Write a Program showing clustering.


PRACTICAL NO : 1

Aim: Install, configure and run Hadoop and HDFS

Description:

Hadoop Installation.

Step 1: download java jdk first .the package size 168.67MB

Windows x64	168.67 MB	 jdk-8u291-windows-x64.exe
-------------	-----------	---

 hadoop-2.10.1-src.tar.gz	16-05-2021 17:16	WinRAR archive	43,967 KB
 hqbhib.txt	06-05-2021 08:23	Text Document	1 KB
 jdk-8u291-windows-x64.exe	16-05-2021 17:16	Application	1,72,731 KB
 LogisticRegressionGFG.png	23-05-2021 17:04	PNG File	4 KB




Step 2: download Hadoop binaries from the official website. The binary package size is about 342 MB.

Download

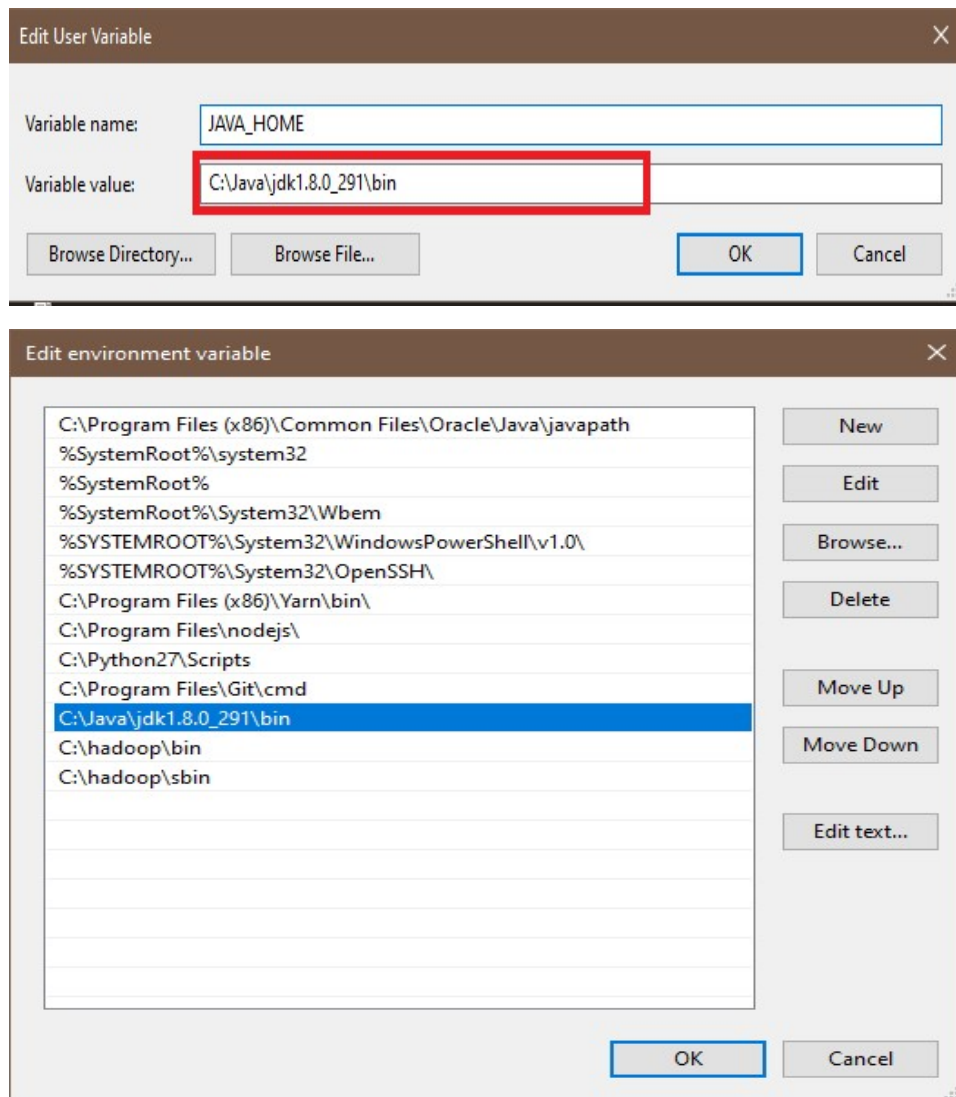
Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed via mirror sites and should be checked for tampering using GPG or SHA-512.

Version	Release date	Source download	Binary download	Release notes
3.2.2	2021 Jan 9	source (checksum signature)	binary (checksum signature)	Announcement
2.10.1	2020 Sep 21	source (checksum signature)	binary (checksum signature)	Announcement
3.1.4	2020 Aug 3	source (checksum signature)	binary (checksum signature)	Announcement
3.3.0	2020 Jul 14	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)	Announcement

Step 3: After finishing the file download, we should unpack the package using 7zip into two steps. First, we should extract the hadoop-3.2.1.tar.gz library, and then, we should unpack the extracted tar file:

Name	Date modified	Type	Size
 hadoop-3.3.0.tar.gz	12-05-2021 08:51	WinRAR archive	4,89,013 KB
 wavelets_0.3-0.2.tar.gz	12-05-2021 08:27	WinRAR archive	114 KB
 govind.data	12-05-2021 08:24	DATA File	283 KB

Step 4: When the “Advanced system settings” dialog appears, go to the “Advanced” tab and click on the “Environment variables” button located on the bottom of the dialog.



Step 5: Check the version of java

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>javac
Usage: javac <options> <source files>
where possible options include:
  -g                        Generate all debugging info
  -g:none                   Generate no debugging info
  -g:{lines,vars,source}   Generate only some debugging info
  -nowarn                   Generate no warnings
  -verbose                  Output messages about what the compiler is doing
  -deprecation              Output source locations where deprecated APIs are used
  -classpath <path>        Specify where to find user class files and annotation process
  -cp <path>                Specify where to find user class files and annotation process
  -sourcepath <path>       Specify where to find input source files
  -bootclasspath <path>    Override location of bootstrap class files
  -extdirs <dirs>          Override location of installed extensions
  -endorseddirs <dirs>     Override location of endorsed standards path
  -proc:{none,only}        Control whether annotation processing and/or compilation is d
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; by

```

```

C:\Users\hp>java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)

```

Step 6: Configuration core-site.xml

container-executor.cfg	07-07-2020 01:03	CFG File
core-site.xml	19-05-2021 17:57	XML File
hadoop-env.cmd	19-05-2021 17:57	Windows Comma...

```

core-site.xml
C: > hadoop > etc > hadoop > core-site.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3
4  <configuration>
5
6  <property>
7    <name>fs.defaultFS</name>
8    <value>hdfs://localhost:9000</value>
9  </property>
10 </configuration>

```

Step 7: Configuration core-site.xml

hdfs-rbf-site.xml	07-07-2020 00:26	XML File
hdfs-site.xml	19-05-2021 17:58	XML File
https-env.sh	07-07-2020 00:25	Shell Script

```
core-site.xml • hdfs-site.xml •
C: > hadoop > etc > hadoop > hdfs-site.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3
4  <configuration>
5  <property>
6      <name>dfs.replication</name>
7      <value>1</value>
8  </property>
9  <property>
10     <name>dfs.namenode.name.dir</name>
11     <value>C:\hadoop\data\namenode</value>
12
13 </property>
14 <property>
15     <name>dfs.namenode.data.dir</name>
16     <value>C:\hadoop\data\datanode</value>
17 </property>
18 </configuration>
```

Step 8: Configuration core-site.xml

mapred-queues.xml.template	07-07-2020 01:04	TEMPLATE File
mapred-site.xml	19-05-2021 17:58	XML File
ssl-client.xml.example	07-07-2020 00:16	EXAMPLE File

```
File Edit Selection View Go Run Terminal Help • mapre
core-site.xml • hdfs-site.xml • mapred-site.xml •
C: > hadoop > etc > hadoop > mapred-site.xml
1  <?xml version="1.0"?>
2  <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
3
4  <configuration>
5  <property>
6      <name>mapreduce.framework.name</name>
7      <value>yarn</value>
8  </configuration>
```

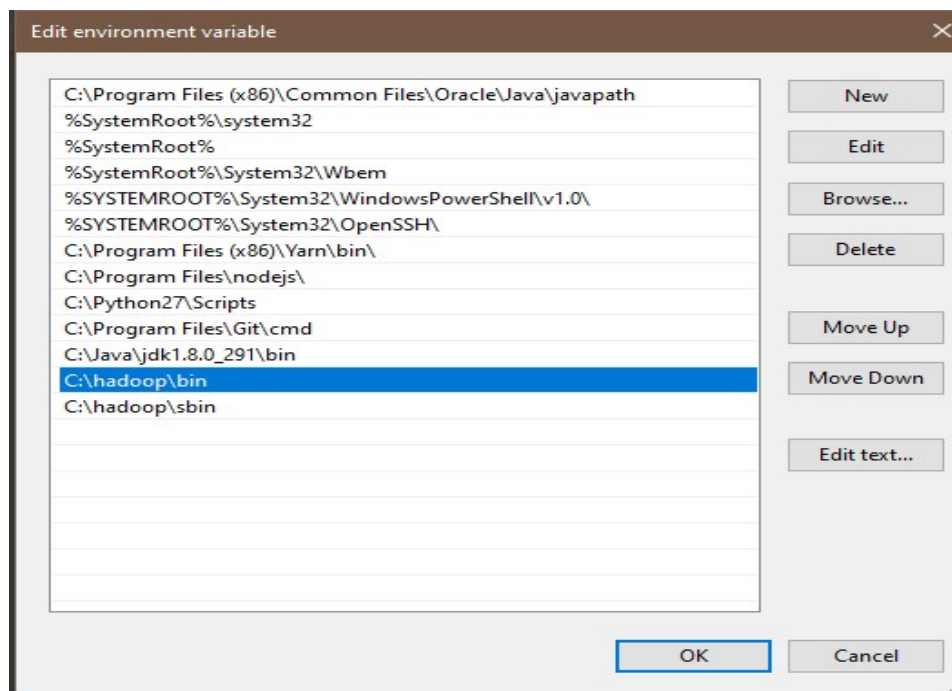
Step 9: Configuration core-site.xml

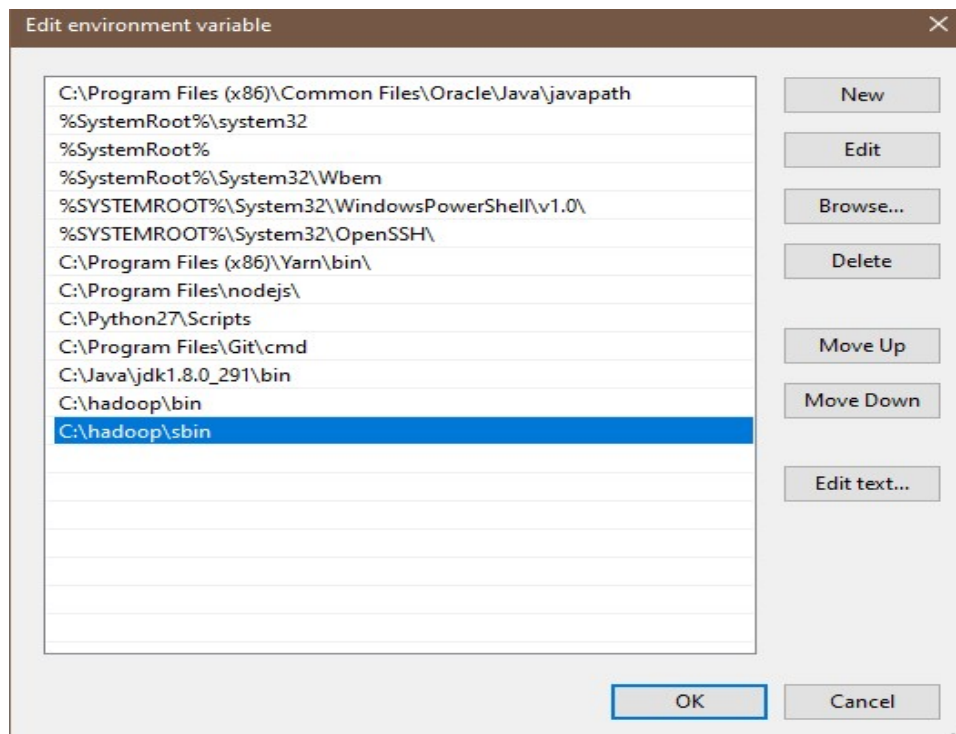
yarnservice-log4j.properties	07-07-2020 01:03	PROPERTIES File
yarn-site.xml	19-05-2021 17:58	XML File

```
C: > hadoop > etc > hadoop > yarn-site.xml
```

```
1  <?xml version="1.0"?>
2  <configuration>
3  <!-- Site specific YARN configuration properties -->
4  <property>
5  |   <name>yarn.nodemanager.aux-services</name>
6  |   <value>mapreduce_shuffle</value>
7  </property>
8  <property>
9  |   <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
10 |   <value>org.apache.hadoop.mapred.shuffleHandles</value>
11 </property>
12 </configuration>
```

Step 10: When the “Advanced system settings” dialog appears, go to the “Advanced” tab and click on the “Environment variables” button located on the bottom of the dialog.





Step 11: let's check Hadoop install Successfully

```

C:\Windows\system32\cmd.exe
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)

C:\Users\hpb>hdfs namenode -format
2021-05-23 17:17:11,111 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = DESKTOP-VUUFK2Q/192.168.0.104
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 3.3.0
STARTUP_MSG:   classpath = C:\hadoop\etc\hadoop;C:\hadoop\share\hadoop\common;C:\h
s-smart-1.2.jar;C:\hadoop\share\hadoop\common\lib\animal-sniffer-annotations-1.17.
asm-5.0.4.jar;C:\hadoop\share\hadoop\common\lib\audience-annotations-0.5.0.jar;C:\
7.7.jar;C:\hadoop\share\hadoop\common\lib\checker-qual-2.5.2.jar;C:\hadoop\share\h
.4.jar;C:\hadoop\share\hadoop\common\lib\commons-cli-1.2.jar;C:\hadoop\share\hadoc
\hadoop\share\hadoop\common\lib\commons-collections-3.2.2.jar;C:\hadoop\share\hadoc
r;C:\hadoop\share\hadoop\common\lib\commons-configuration2-2.1.1.jar;C:\hadoop\sha
0.13.jar;C:\hadoop\share\hadoop\common\lib\commons-io-2.5.jar;C:\hadoop\share\hadc
\hadoop\share\hadoop\common\lib\commons-logging-1.1.3.jar;C:\hadoop\share\hadoop\c
adoop\share\hadoop\common\lib\commons-net-3.6.jar;C:\hadoop\share\hadoop\common\li
\hadoop\common\lib\curator-client-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\cura
e\hadoop\common\lib\curator-recipes-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\dr
\common\lib\failureaccess-1.0.jar;C:\hadoop\share\hadoop\common\lib\gson-2.2.4.jar
va-27.0-jre.jar;C:\hadoop\share\hadoop\common\lib\hadoop-annotations-3.3.0.jar;C:\
auth-3.3.0.jar;C:\hadoop\share\hadoop\common\lib\hadoop-shaded-protobuf_3_7-1.0.0.
htrace-core4-4.1.0-incubating.jar;C:\hadoop\share\hadoop\common\lib\httpclient-4.5
ib\httpcore4-4.4.10.jar;C:\hadoop\share\hadoop\common\lib\j2objc-annotations-1.1.ja

```

```
Apache Hadoop Distribution
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
2021-05-23 17:19:33,116 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = DESKTOP-VUUFK2Q/192.168.0.104
STARTUP_MSG: args = []
STARTUP_MSG: version = 3.3.0
STARTUP_MSG: classpath = C:\hadoop\etc\hadoop;C:\hadoop\share\hadoop\common;C:\hadoop\share\hadoop\common\lib\accessor
s-smart-1.2.jar;C:\hadoop\share\hadoop\common\lib\animal-sniffer-annotations-1.17.jar;C:\hadoop\share\hadoop\common\lib\
asm-5.0.4.jar;C:\hadoop\share\hadoop\common\lib\audience-annotations-0.5.0.jar;C:\hadoop\share\hadoop\common\lib\avro-1.
7.7.jar;C:\hadoop\share\hadoop\common\lib\checker-qual-2.5.2.jar;C:\hadoop\share\hadoop\common\lib\commons-beanutils-1.9
.4.jar;C:\hadoop\share\hadoop\common\lib\commons-cli-1.2.jar;C:\hadoop\share\hadoop\common\lib\commons-codec-1.11.jar;C:
\hadoop\share\hadoop\common\lib\commons-collections-3.2.2.jar;C:\hadoop\share\hadoop\common\lib\commons-compress-1.19.ja
r;C:\hadoop\share\hadoop\common\lib\commons-configuration2-2.1.1.jar;C:\hadoop\share\hadoop\common\lib\commons-daemon-1.
0.13.jar;C:\hadoop\share\hadoop\common\lib\commons-io-2.5.jar;C:\hadoop\share\hadoop\common\lib\commons-lang3-3.7.jar;C:
\hadoop\share\hadoop\common\lib\commons-logging-1.1.3.jar;C:\hadoop\share\hadoop\common\lib\commons-math3-3.1.1.jar;C:\h
adoop\share\hadoop\common\lib\commons-net-3.6.jar;C:\hadoop\share\hadoop\common\lib\commons-text-1.4.jar;C:\hadoop\share
\hadoop\common\lib\curator-client-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\curator-framework-4.2.0.jar;C:\hadoop\shar
e\hadoop\common\lib\curator-recipes-4.2.0.jar;C:\hadoop\share\hadoop\common\lib\dnsjava-2.1.7.jar;C:\hadoop\share\hadoop
```

```
Apache Hadoop Distribution
at com.ctc.wstx.sr.StreamScanner.throwParseError(StreamScanner.java:491)
at com.ctc.wstx.sr.StreamScanner.throwParseError(StreamScanner.java:475)
at com.ctc.wstx.sr.BasicStreamReader.reportWrongEndElem(BasicStreamReader.java:3365)
at com.ctc.wstx.sr.BasicStreamReader.readEndElem(BasicStreamReader.java:3292)
at com.ctc.wstx.sr.BasicStreamReader.nextFromTree(BasicStreamReader.java:2911)
at com.ctc.wstx.sr.BasicStreamReader.next(BasicStreamReader.java:1123)
at org.apache.hadoop.conf.Configuration$Parser.parseNext(Configuration.java:3347)
at org.apache.hadoop.conf.Configuration$Parser.parse(Configuration.java:3141)
at org.apache.hadoop.conf.Configuration.loadResource(Configuration.java:3034)
... 9 more
```

Step 12: Let check bin

```
C:\Windows\system32\cmd.exe
C:\Users\hp>cd C:\hadoop\sbin
C:\hadoop\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
C:\hadoop\sbin>
```


PRACTICAL NO : 2

Aim: Implement Decision tree classification techniques

Description:

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

Step 1: The package "party" has the function `ctree()` which is used to create and analyze decision tree.

```
> install.packages("party")
```

Step 2: Load the party package. It will automatically load other# dependent packages
Print some records from data set `readingskills`.

```
> library("party")
> print(head(readingskills))
  nativespeaker age shoesize    score
1          yes   5  24.83189 32.29385
2          yes   6  25.95238 36.63105
3          no  11  30.42170 49.60593
4          yes   7  28.66450 40.28456
5          yes  11  31.88207 55.46085
6          yes  10  30.07843 52.83124
>
```

Step 3 : Call function `ctree` to build a decision tree. The first parameter is a formula, which defines a target variable and a list of independent variables.

```
> library("party")
> str(iris)
'data.frame':  150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1
```

```
> iris_ctree <- ctree(species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data=iris)
> print(iris_ctree)
```

Conditional inference tree with 4 terminal nodes

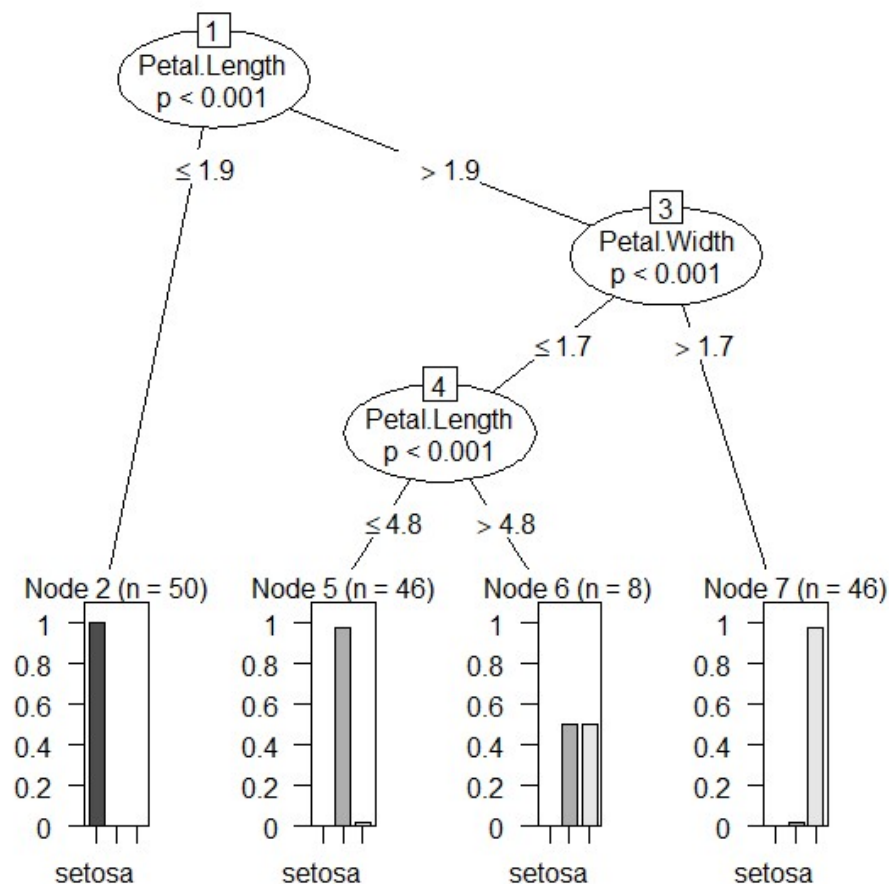
Response: species

Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width

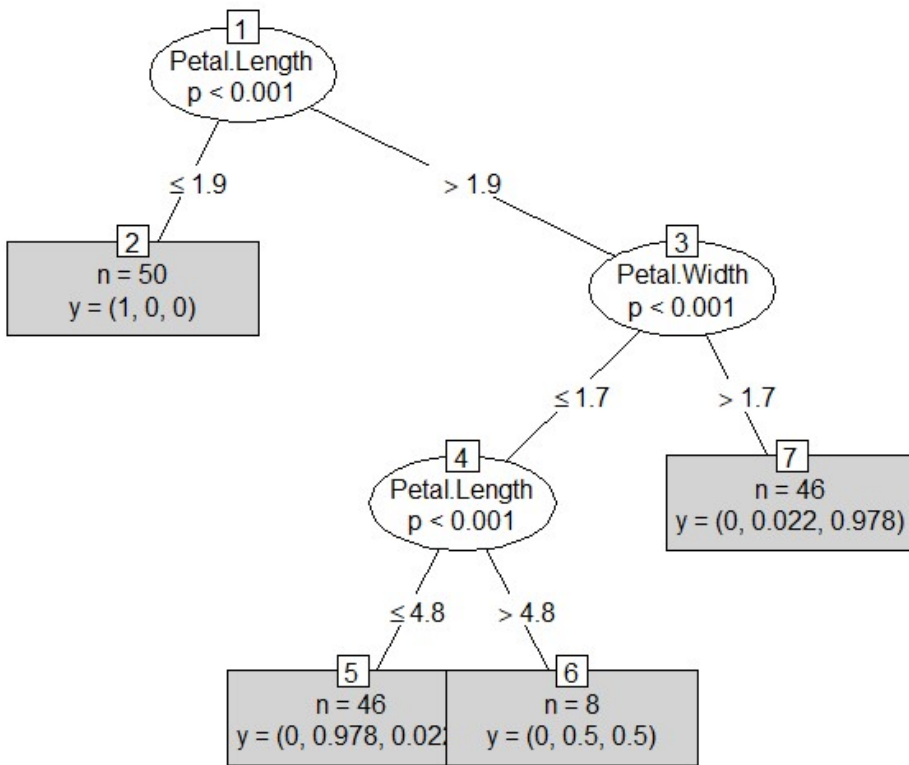
Number of observations: 150

```
1) Petal.Length <= 1.9; criterion = 1, statistic = 140.264
   2)* weights = 50
1) Petal.Length > 1.9
   3) Petal.Width <= 1.7; criterion = 1, statistic = 67.894
     4) Petal.Length <= 4.8; criterion = 0.999, statistic = 13.865
       5)* weights = 46
     4) Petal.Length > 4.8
       6)* weights = 8
   3) Petal.Width > 1.7
     7)* weights = 46
> plot(iris_ctree)
```

Output :



```
> plot(iris_ctree, type="simple")
```



PRACTICAL NO : 3

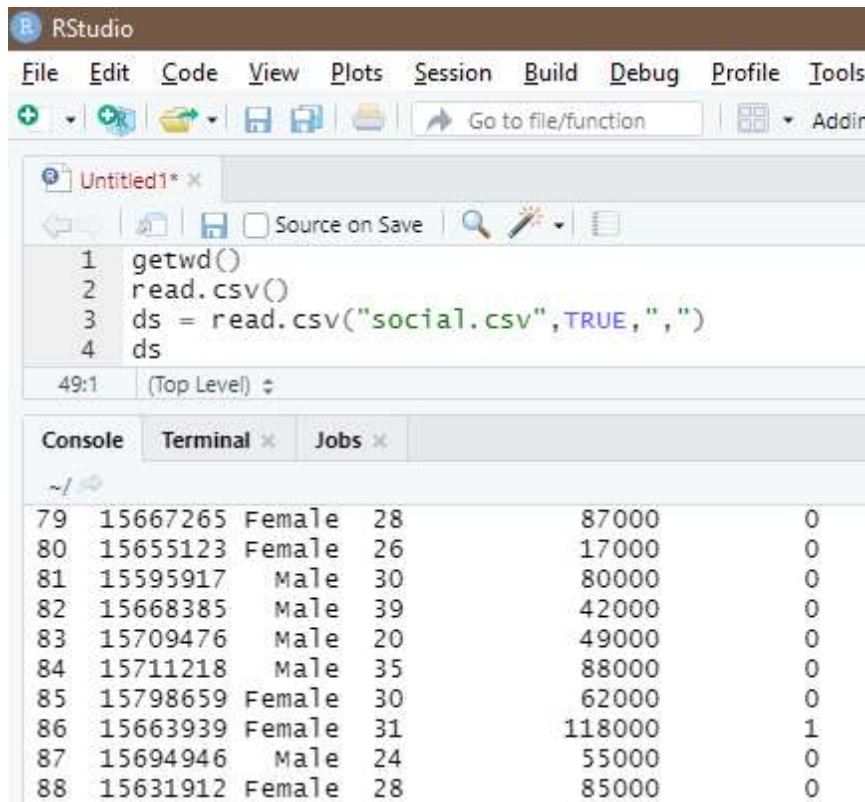
Aim: Classification using SVM

Description:

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text

The implementation is explained in the following steps:

Step 1: Importing the dataset



```
1 getwd()
2 read.csv()
3 ds = read.csv("social.csv", TRUE, ",", "")
4 ds
```

49:1 (Top Level) ↕

	Console	Terminal ×	Jobs ×
~/			
79	15667265	Female	28
80	15655123	Female	26
81	15595917	Male	30
82	15668385	Male	39
83	15709476	Male	20
84	15711218	Male	35
85	15798659	Female	30
86	15663939	Female	31
87	15694946	Male	24
88	15631912	Female	28

Step 2: Selecting columns 3-5


```

> ds = ds[3:5]
> ds[3:5]
Error in `[.data.frame`(ds, 3:5) : undefined
> ds

```

	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
9	25	33000	0
10	35	65000	0
11	26	80000	0
12	26	52000	0

Step 3: install package

```

> install.packages("caTools")

```

Step 4: Splitting the dataset

```

> library(caTools)
> set.seed(123)
> split = sample.split(ds$Purchased, SplitRatio = 0.75)
> training_set = subset(ds, split == TRUE)
> test_set = subset(ds, split == FALSE)
> ds

```

	Age	EstimatedSalary	Purchased
1	19	19000	0
2	35	20000	0
3	26	43000	0
4	27	57000	0
5	19	76000	0
6	27	58000	0
7	27	84000	0
8	32	150000	1
9	25	33000	0
10	35	65000	0
..

Step 5: Feature Scaling

```

332  48      119000      1
333  42      65000      0
[ reached 'max' / getOption("max.print") -- omitted 67 rows ]
> test_set[-3] = scale(test_set[-3])
> training_set[-3] = scale(training_set[-3])
> test_set[-3] = scale(test_set[-3])
> test_set[-3]
      Age EstimatedSalary
2  -0.30419063    -1.51354339
4  -1.05994374    -0.32456026
5  -1.81569686     0.28599864
9  -1.24888202    -1.09579256
12 -1.15441288    -0.48523366
18  0.64050076    -1.32073531
19  0.73496990    -1.25646596
20  0.92390818    -1.22433128
22  0.82943904    -0.58163769
29 -0.87100546    -0.77444577
32 -1.05994374     2.24621408
34 -0.96547460    -0.74231109
35 -1.05994374     0.73588415
38 -0.77653633    -0.58163769
45 -0.96547460     0.54307608
46 -1.43782030    -1.51354339

```

Step 6: Fitting SVM to the training set

```

> install.packages('e1071')

> library(e1071)
> classifier = svm(formula = Purchased ~ .,
+                  data = training_set,
+                  type = 'C-classification',
+                  kernel = 'linear')
> classifier

Call:
svm(formula = Purchased ~ ., data = training_set, type = "C-classification",
    kernel = "linear")

Parameters:
  SVM-Type:  C-classification
 SVM-Kernel: linear
       cost: 1

Number of Support Vectors: 116

```

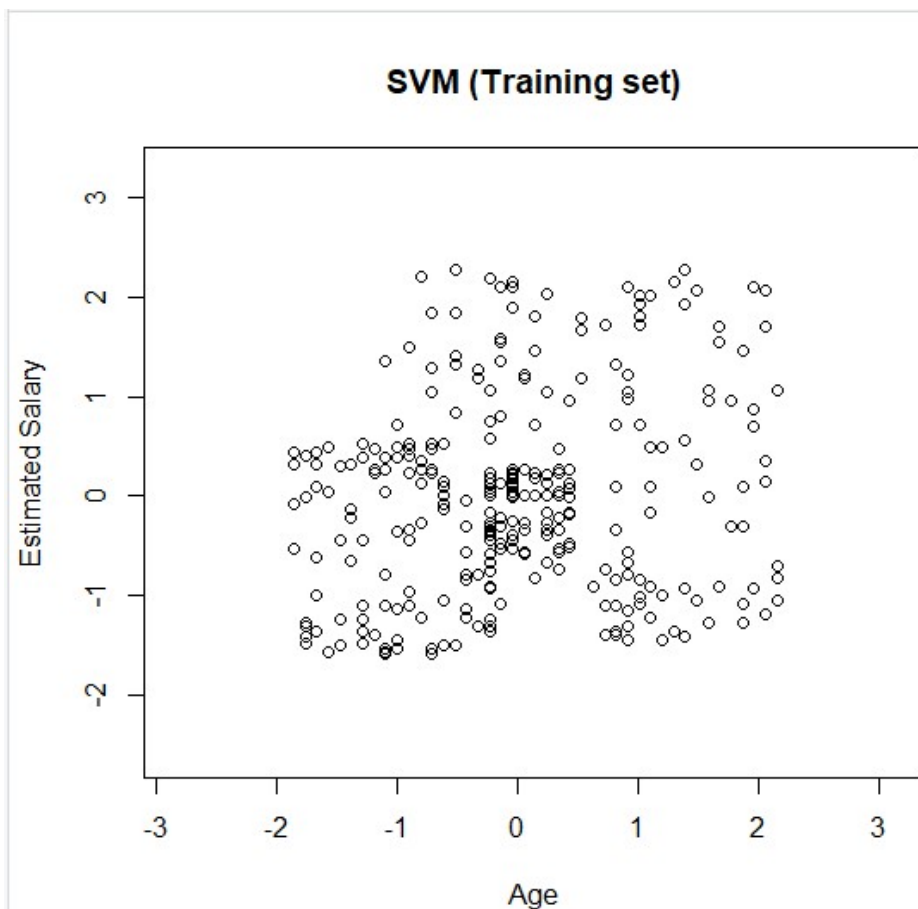
Step 7: Predicting the test set result

```
> y_pred = predict(classifier, newdata = test_set[-3])
> y_pred
 2   4   5   9  12  18  19  20  22  29  32  34  35  38  45  46  48  52  66
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
69  74  75  82  84  85  86  87  89 103 104 107 108 109 117 124 126 127 131
0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
134 139 148 154 156 159 162 163 170 175 176 193 199 200 208 213 224 226 228
0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   1   1   0   1
229 230 234 236 237 239 241 255 264 265 266 273 274 281 286 292 299 302 305
0   1   1   1   0   1   1   1   0   1   1   1   1   1   0   1   1   1   0
307 310 316 324 326 332 339 341 343 347 353 363 364 367 368 369 372 373 380
1   0   0   0   0   1   0   1   0   1   1   0   1   1   1   0   1   0   1
383 389 392 395 400
1   0   0   0   0
Levels: 0 1
```

```
> cm = table(test_set[, 3], y_pred)
> cm
  y_pred
    0    1
0  57    7
1  13   23
```

Step 8: Visualizing the Training set results

```
> 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)
> plot(set[, -3],
+       main = 'SVM (Training set)',
+       xlab = 'Age', ylab = 'Estimated salary',
+       xlim = range(x1), ylim = range(x2))

```



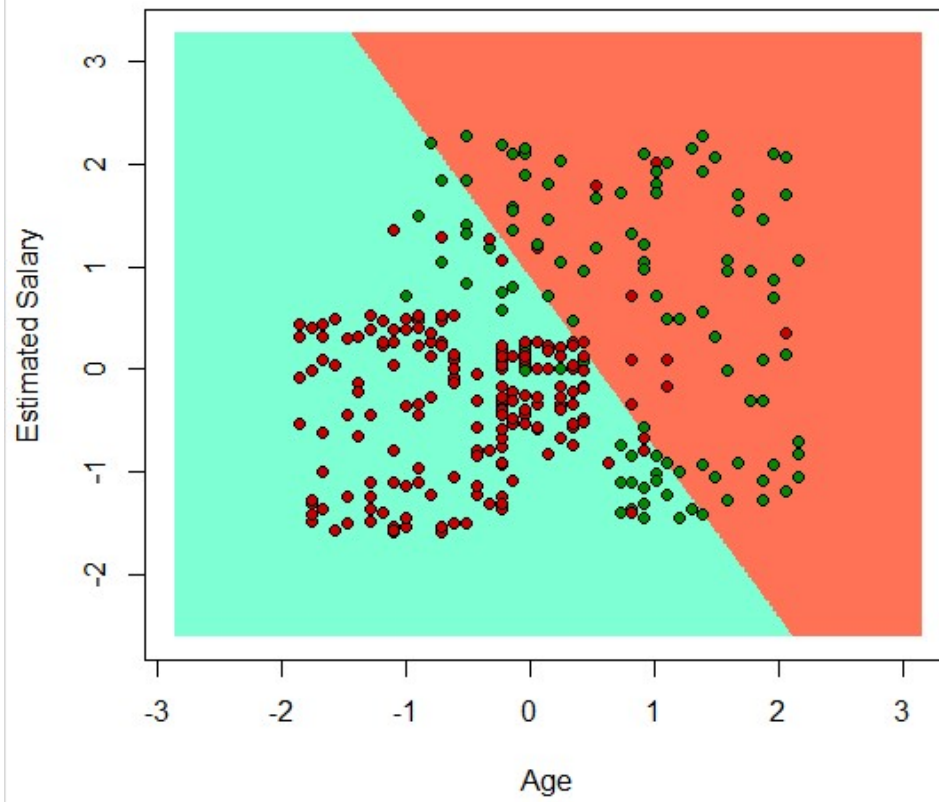
```

> contour(x1, x2, matrix(as.numeric(y_grid), length(x1), length(x2)), add = TRUE)
> points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'coral1', 'aquamarine'))
> points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

```

Output:

SVM (Training set)



PRACTICAL NO : 4

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

Description:

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License

The screenshot shows the MongoDB Atlas sign-up process. The first section, 'Name your organization and project', includes a text input for 'Organization' (containing 'Education') and a text input for 'Project Name' (containing 'govind-prac_4'). The second section, 'What is your preferred language?', displays a grid of language options: JavaScript, C++, C#/.NET, Go, Java, C, Perl, PHP, Python (highlighted with a green border), Ruby, Scala, and Other. At the bottom right, there are 'Skip' and 'Continue' buttons.

Name your organization and project

Organization
Your organization can be a business, team, or an individual

Education

Project Name
Use projects to isolate different environments (development/testing/production)

govind-prac_4

What is your preferred language?
We'll use this to customize code samples and content we share with you. You can always change this later.

JS JavaScript	C++ C++	C#/.NET C#/.NET	Go Go
Java Java	C C	Perl Perl	PHP PHP
Python Python	Ruby Ruby	Scala Scala	Other

Skip Continue

Step 1 : Sign up and create a cluster.

Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our [documentation](#).

Cloud Provider & Region

AWS, Mumbai (ap-south-1) ▼



★ Recommended region ⓘ

NORTH AMERICA	ASIA	EUROPE
🇺🇸 N. Virginia (us-east-1) ★	🇸🇬 Singapore (ap-southeast-1) ★	🇩🇪 Frankfurt (eu-central-1) ★
🇺🇸 Oregon (us-west-2) ★	🇮🇳 Mumbai (ap-south-1)	🇮🇪 Ireland (eu-west-1) ★
		AUSTRALIA
		🇦🇺 Sydney (ap-southeast-2) ★

This is the home page of mongoDB Atlas.

The screenshot shows the MongoDB Atlas interface. The 'Connect to Atlas' modal is open, displaying a checklist for getting started. The first item, 'Build your first cluster', is checked. The other items are 'Create your first database user', 'Add IP Address to your Access List', 'Load Sample Data (Optional)', and 'Connect to your cluster'. The background shows the 'Clusters' page with 'Cluster0' selected under the 'SANDBOX' tier. The cluster details indicate it is a 'Shared Tier Cluster' with a logical size of 0.0 B and 0 connections. A 'Get Started' button is located at the bottom left of the modal.

Step 2 : Click on collections to create and view existing databases.

The 'Explore Your Data' section includes a green icon of a document with a magnifying glass. Below the icon, the text 'Explore Your Data' is displayed. A list of actions is provided: '- Find: run queries and interact with documents', '- Indexes: build and manage indexes', '- Aggregation: test aggregation pipelines', and '- Search: build search indexes'. At the bottom, there are two buttons: 'Load a Sample Dataset' (green) and 'Add My Own Data' (grey). A link 'Learn more in Docs and Tutorials' with an external link icon is also present.

Step 3 : Click on 'Add My Own Data' to create a database.

×

Create Database

DATABASE NAME ?

govind_db

COLLECTION NAME ?

govind

☐ Capped Collection

Before MongoDB can save your new database, a collection name must be specified at the time of creation.

Cancel

Create

Step 4 : Click on insert document to add records.

DATABASES: 1 COLLECTIONS: 2

+ Create Database

Q NAMESPACES

govind_db

7_govind

govind

govind_db.govind

COLLECTION SIZE: 144B TOTAL DOCUMENTS: 2 INDEXES T

Find

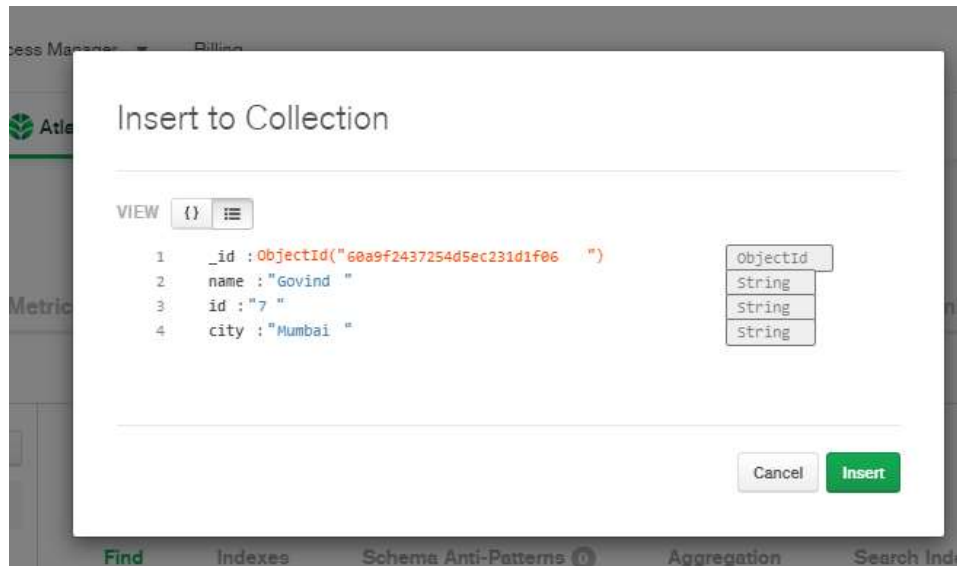
Indexes

Schema Anti-Patterns 0

FILTER

{"filter": "example"}

Since MongoDB is a No-SQL database, so you can add 'n' number of columns for any row/record.



Perform updating data



Performing deleting data



Performing Insert data

QUERY RESULTS 1-2 OF 2

```
_id: ObjectId("60a9ff027254d5ec231d1f0b")
name: "Govind Saini"
id: " 7"
city: "Mumbai"
```

```
_id: ObjectId("60a9ff3a7254d5ec231d1f0c")
name: "Sohrab Sir"
id: " 5"
city: "Mumbai"
```

Step 5 : To start with the connection click on Overview, and then click on Connect.

The screenshot shows the MongoDB Atlas interface. At the top, there's a navigation bar with 'Education' selected, and links for 'Access Manager' and 'Billing'. Below this, the 'govind-prac_4' project is selected, and the 'Atlas' tab is active. The left sidebar shows 'DATA STORAGE' with 'Clusters' selected, and 'SECURITY' with 'Database Access', 'Network Access', and 'Advanced' options. The main content area is titled 'Clusters' and shows a search bar. Below the search bar, the 'SANDBOX' tab is selected, showing 'Cluster0' (Version 4.4.6). The cluster details include: 'CONNECT', 'METRICS', 'COLLECTIONS', and a menu icon; 'CLUSTER TIER: M0 Sandbox (General)'; 'REGION: AWS / Mumbai (ap-south-1)'; 'TYPE: Replica Set - 3 nodes'; and 'LINKED REALM APP: None Linked'. On the right, a message states 'This is a Shared Tier Cluster' with an 'Upgrade' button. Below this, a 'Logical Size' bar chart shows '2.4 KB' out of a '512.0 MB max' capacity, with a '0.0 B' usage at the bottom. The chart is labeled 'Last 6 Hours'.

Step 6 : Select on add your current IP and create a MongoDB user.

×

Connect to Cluster0

Setup connection security

Choose a connection method

Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

You're ready to connect. Choose how you want to connect in the next step.

1 Add a connection IP address

✓ An IP address has been added to the IP Access List. [Add another address in the IP Access List tab.](#)

2 Create a Database User

✓ A MongoDB user has been added to this project. *Not yours?* Create one in the [MongoDB Users](#) tab.
You'll need your MongoDB user's credentials in the next step.

Close

Choose a connection method

Step 7 : Click on 'Connect your application'.

×

Connect to Cluster0

✓ Setup connection security


Choose a connection method

Connect

Choose a connection method


[View documentation](#)

Get your pre-formatted connection string by selecting your tool below.




Connect with the mongo shell
Interact with your cluster using MongoDB's interactive Javascript interface

>



Connect your application
Connect your application to your cluster using MongoDB's native drivers

>



Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI

>

Go Back

Close

Step 8 : Select the driver as 'Python' and version as '3.6 or later'. (Select the version as 3.6 or later only if your Python's version is 3.6 or later.)

Connect to Cluster0

✓ Setup connection security

✓ Choose a connection method

Connect

1 Select your driver and version

DRIVER

Python

VERSION

3.6 or later

2 Add your connection string into your application code

☐ Include full driver code example

```
mongodb+srv://dbGovind:<password>@cluster0.m6shh.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **dbGovind** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Step 9 : Write the code given below in a Python file.

```
prac4.py - C:/Python27/prac4.py (2.7.17)
File Edit Format Run Options Window Help

import pymongo
from pymongo import MongoClient
client = pymongo.MongoClient("mongodb+srv://dbGovind:GmongoDB123@cluster0.m6shh.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
db = client.get_database('govind_db')
records = db.govind
db = client.test
print(records.count_documents({}))
print(list(records.find({})))
```

Output :

```
===== RESTART: C:/Python27/prac.py =====
2[{"_id":{"$_oid":"60a9ff027254d5ec231dlf0b"},"name":"Govind Saini","id":" 7","city":"Mumbai"}{"_id":{"$_oid":"60a9ff3a7254d5ec231dlf0c"},"name":"Sohrab Sir","id":" 5","city":"Mumbai"}]
>>> |
```

PRACTICAL NO : 5

Aim: write program in R of Naive baye's theorem

Description:

Naive Bayes is a Supervised Non-linear classification algorithm in R Programming. Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Baye's theorem with strong(Naive) independence assumptions between the features or variables

Loading data

```
> data(iris)
> str(iris)
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1
```

Installing Packages

```
> install.packages("e1071")
> install.packages("caTools")
> install.packages("caret")
```

Loading package

```
> library(e1071)
> library(caTools)
> library(caret)
Loading required package: lattice
Loading required package: ggplot2
```

Splitting data into train and test data

```

> split <- sample.split(iris, splitRatio = 0.7)
> train_cl <- subset(iris, split == "TRUE")
> test_cl <- subset(iris, split == "FALSE")
>
> train_scale <- scale(train_cl[, 1:4])
> test_scale <- scale(test_cl[, 1:4])
>
> set.seed(120) # Setting Seed
> classifier_cl <- naiveBayes(species ~ ., data = train_cl)
> classifier_cl

```

Naive Bayes Classifier for Discrete Predictors

Call:

```
naiveBayes.default(x = x, y = Y, laplace = laplace)
```

A-priori probabilities:

```

Y
      setosa versicolor virginica
0.3333333 0.3333333 0.3333333

```

Conditional probabilities:

```

      Sepal.Length
Y      [,1]      [,2]
setosa  5.046667 0.3848272
versicolor 5.963333 0.5268536
virginica 6.553333 0.6693967

```

```

      Sepal.width
Y      [,1]      [,2]
setosa  3.413333 0.4256705
versicolor 2.823333 0.3470897
virginica 2.956667 0.3136914

```

```

      Petal.Length
Y      [,1]      [,2]
setosa  1.466667 0.1561019
versicolor 4.320000 0.4759020
virginica 5.496667 0.5738457

```

```

      Petal.width
Y      [,1]      [,2]
setosa  0.2766667 0.1135124
versicolor 1.3533333 0.1960530
virginica 2.0433333 0.2568823

```

Predicting on test data'

```

> y_pred <- predict(classifier_cl, newdata = test_cl)
> cm <- table(test_cl$species, y_pred)
> cm
      y_pred
      setosa versicolor virginica
setosa      20          0          0
versicolor   0         19          1
virginica    0          2         18
>

```

Model Evaluation

```
> confusionMatrix(cm)
```

Confusion Matrix and Statistics

	y_pred		
	setosa	versicolor	virginica
setosa	20	0	0
versicolor	0	19	1
virginica	0	2	18

Overall Statistics

Accuracy : 0.95
95% CI : (0.8608, 0.9896)
No Information Rate : 0.35
P-value [Acc > NIR] : < 2.2e-16

Kappa : 0.925

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: setosa	Class: versicolor	Class: virginica
sensitivity	1.0000	0.9048	0.9474
specificity	1.0000	0.9744	0.9512
Pos Pred Value	1.0000	0.9500	0.9000
Neg Pred Value	1.0000	0.9500	0.9750
Prevalence	0.3333	0.3500	0.3167
Detection Rate	0.3333	0.3167	0.3000
Detection Prevalence	0.3333	0.3333	0.3333
Balanced Accuracy	1.0000	0.9396	0.9493

PRACTICAL NO : 6

Aim: Write a Program showing implementation of Regression model.

Description:

Regression is a method to mathematically formulate relationship between variables that in due course can be used to estimate, interpolate and extrapolate. Suppose we want to estimate the weight of individuals, which is influenced by height, diet, workout, etc.

Here, *Weight* is the **predicted** variable

Lets implementation of Regression Model some Example:

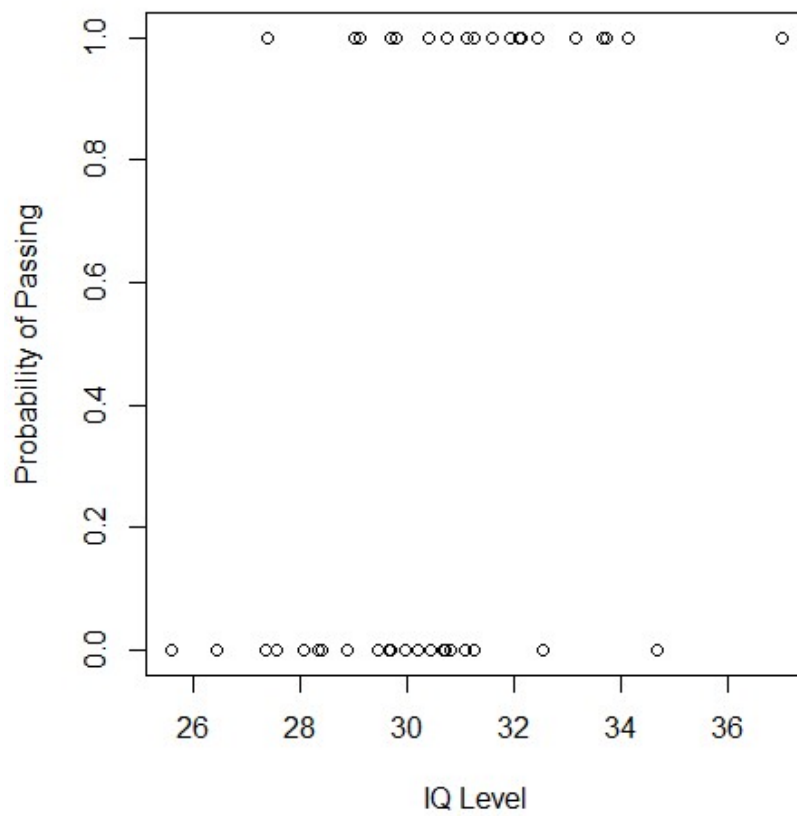
```
> IQ <- rnorm(40, 30, 2)
|> IQ <- sort(IQ)
> result <- c(0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
+ 1, 0, 0, 0, 1, 1, 0, 0, 1, 0,
+ 0, 0, 1, 0, 0, 1, 1, 0, 1, 1,
+ 1, 1, 1, 0, 1, 1, 1, 1, 0, 1)
```

```
> df <- as.data.frame(cbind(IQ, result))
> print(df)
```

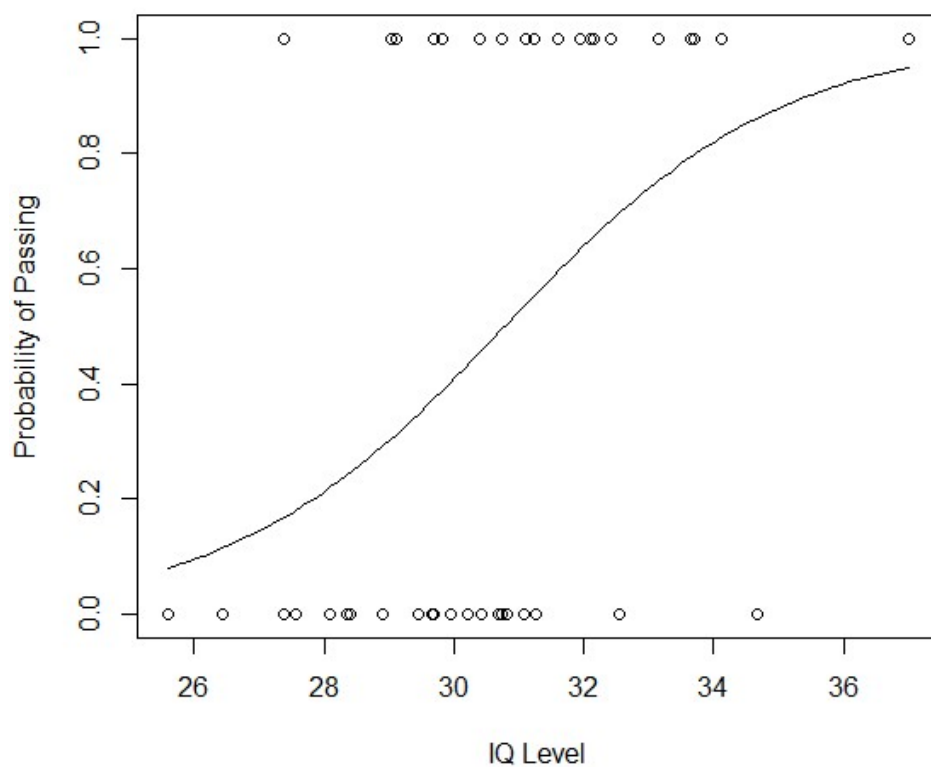
	IQ	result
1	25.58824	0
2	26.43200	0
3	27.37083	0
4	27.37898	1
5	27.56671	0
6	28.08275	0
7	28.35637	0
8	28.41538	0
9	28.89752	0
10	29.03158	1
11	29.12386	1
12	29.46181	0
13	29.66945	0
14	29.68934	0
15	29.69886	1
16	29.80735	1
17	29.95326	0
18	30.21428	0
19	30.39298	1
20	30.43421	0
21	30.67802	0
22	30.72653	0
23	30.74974	1
24	30.82265	0
25	31.07116	0
26	31.11633	1
27	31.24740	1
28	31.25662	0
29	31.60194	1
30	31.93038	1

```
> png(file="LogisticRegressionGFG.png")
```

```
> plot(IQ, result, xlab = "IQ Level",
+ ylab = "Probability of Passing")
> g = glm(result~IQ, family=binomial, df)
```

```
> curve(predict(g, data.frame(IQ=x), type="resp"), add=TRUE)
> points(IQ, fitted(g), pch=30)
```



```
> summary(g)
```

```
call:
```

```
glm(formula = result ~ IQ, family = binomial, data = df)
```

```
Deviance Residuals:
```

	Min	1Q	Median	3Q	Max
	-1.9877	-0.9804	-0.4502	0.9731	1.8898

```
Coefficients:
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-14.4934	5.8835	-2.463	0.0138 *
IQ	0.4708	0.1922	2.450	0.0143 *

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)
```

```
Null deviance: 55.352  on 39  degrees of freedom
Residual deviance: 47.090  on 38  degrees of freedom
AIC: 51.09
```

```
Number of Fisher Scoring iterations: 4
```

```
> dev.off()
```

```
null device
1
```

PRACTICAL NO : 7

Aim: Write a Program showing clustering.

Description:

In this Program we understand about K-Mean Clustering

What Does K-Means Clustering Mean?

- K-means clustering is a simple unsupervised learning algorithm that is used to solve clustering problems.
- It follows a simple procedure of classifying a given data set into a number of clusters, defined by the letter "k," which is fixed beforehand.
- The clusters are then positioned as points and all observations or data points are associated with the nearest cluster, computed, adjusted and then the process starts over using the new adjustments until a desired result is reached.

We Understand in different Steps :

Step 1: Apply `kmeans` to *newiris*, and store the clustering result in *kc*. The cluster number is set to 3.

```

> newiris <- iris
> newiris$Species <- NULL
> (kc <- kmeans(newiris, 3))
K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:
  Sepal.Length Sepal.width Petal.Length Petal.width
1    6.850000    3.073684    5.742105    2.071053
2    5.901613    2.748387    4.393548    1.433871
3    5.006000    3.428000    1.462000    0.246000

Clustering vector:
 [1] 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 3 3 3 3 3
[35] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2
[69] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2
[103] 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 2 1 1
[137] 1 1 2 1 1 1 2 1 1 1 2 1 1 2

within cluster sum of squares by cluster:
[1] 23.87947 39.82097 15.15100
(between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"

```

Step 2: Compare the Species label with the clustering result

```

> table(iris$Species, kc$cluster)

      1   2   3
setosa  0   0 50
versicolor  2 48  0
virginica 36 14  0

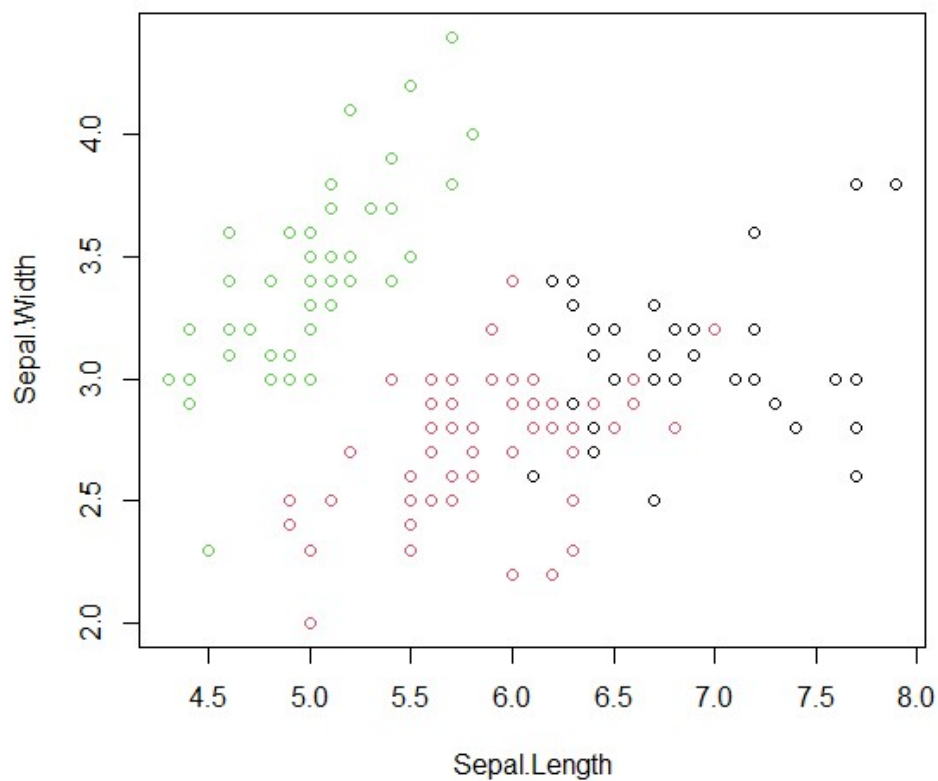
```

Step 3 : Plot the clusters and their centres. Note that there are four dimensions in the data and that only the first two dimensions are used to draw the plot below.

```

> plot(newiris[c("Sepal.Length", "Sepal.width")], col=kc$cluster)

```



Step 4: Some black points close to the green centre (asterisk) are actually closer to the black centre in the four dimensional space.

```
> points(kc$centers[,c("sepal.Length", "sepal.Width")], col=1:3, pch=8, cex=2)
```

