1.HypothesisTest

```
x<-rnorm(100) t.test(x,mu=5)
x1<-rnorm(100) y1<-rnorm(100) t.test(x1,y1)
x2<-rnorm(100) t.test(x2,mu=2,alternative='greater')
x3<-rnorm(100) wilcox.test(x3,exact=FALSE)
cor.test(mtcars$mpg,mtcars$hp)
```

…….

2.KMeans Clustering

```
library(cluster) library(ggplot2)
set.seed(20)
irisCluster<-kmeans(iris[,3:4],3,nstart=20) irisCluster
irisCluster$cluster<-as.factor(irisCluster$cluster)
ggplot(iris,aes(Petal.Length,Petal.Width,color=irisCluster$cluster))+geom_p
oint()
d<-dist(as.matrix(mtcars)) hc<-hclust(d)
plot(hc)
x<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5),rnorm(15,5,0.5)))
clusplot(pam(x,2))
x4<-cbind(x,rnorm(25),rnorm(25)) clusplot(pam(x4,2))
```

……

3.Implement Linearand Logistic Regression

```
dataset=read.csv("/content/data-marketing-budget-12mo.csv",header=T,
colClasses=c("numeric","numeric","numeric"))
head(dataset)
simple.fit = lm(Sales~Spend,data=dataset) summary(simple.fit)
multi.fit=lm(Sales~Spend+Month,data=dataset) summary(multi.fit)
input<-mtcars[,c("am","cyl","hp","wt")] print(head(input))
input<-mtcars[,c("am","cyl","hp","wt")]
am.data=glm(formula=am~cyl+hp+wt,data=input,family=binomial)
print(summary(am.data))
```

…….

4.Time Series Analysis.

```
kings <- scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)
kings
kingstimeseries<-ts(kings)
kingstimeseries
births<-scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")birth
stimeseries<-ts(births,frequency=12,start=c(1946,1))
 birthstimeseries
souvenir<-
scan("http://robjhyndman.com/tsdldata/data/fancy.dat")
souvenirtimeseries<-
ts(souvenir,frequency=12,start=c(1987,1))
 Souvenirtimeseries
```

……
5.Data Visualizationtoexplore Various

CodeForHistogram:
library(RColorBrewer) data(VADeaths) par(mfrow=c(2,3))
hist(VADeaths,breaks=10, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
hist(VADeaths,breaks=3 ,col=brewer.pal(3,"Set2"),main="Set2 3 colors")
hist(VADeaths,breaks=7, col=brewer.pal(3,"Set1"),main="Set1 3 colors")
hist(VADeaths,,breaks= 2, col=brewer.pal(8,"Set3"),main="Set3 8 colors")
hist(VADeaths,col=brewer.pal(8,"Greys"),main="Greys8colors")
hist(VADeaths,col=brewer.pal(8,"Greens"),main="Greens8colors")
CodeForLineChart:
data(AirPassengers)
plot(AirPassengers,type="l")
CodeforBarChart:
data("iris")
barplot(iris$Petal.Length) #Creating simple Bar Graph barplot(iris$Sepal.Length,col =
brewer.pal(3,"Set1")) barplot(table(iris$Species,iris$Sepal.Length),col
=brewer.pal(3,"Set1"))
CodeforBoxplot:
data(iris) par(mfrow=c(2,2))
boxplot(iris$Sepal.Length,col="red") boxplot(iris$Sepal.Length~iris$Species,col="red")
boxplot(iris$Sepal.Length~iris$Species,col=heat.colors(3))boxplot(iris$Sepal.Length~iris$Sp
ecies,col=topo.colors(3)) boxplot(iris$Petal.Length~iris$Species)
Code for Scatter Plot: plot(x=iris$Petal.Length) plot(x=iris$Petal.Length,y=iris$Species)
……
6.Install and ConfigureHadoop

SETUP:
SettingupHadooponWindows
PREREQUISITES:
Windows64bitOS
JavaJDK
AdministratorAccess
INSTALLATION
1.DownloadHadoopforwindowsfromtheofficialsite
2.ExtracttheZIPtoyourChosendirectory(Hadoopinstallatio)
3.SetEnvironmentVariables: ‘HADOOP_HOME’ ‘JAVA_HOME’
Edit the “Path” variable and add: ‘%HADOOP_HOME%\bin’ ‘%HADOOP_HOME%\sbin’
‘JAVA_HOME%\bin’
4.ConfigureHadoop
a)Open‘hadoop-env.cmd’,set‘JAVA_HOME’.
Create/edit‘core-site.xml’with:
xml
<property>
<name>fs.defaultFS</name>
<value>hdfs:localhost:9000</value>
</property>

b)Create/edit‘hdfs-site.xml’with: xml
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
d)create/edit‘yarn-site.xml’with: xml
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.env-whitelist</name>
<value>HADOOP_HOME,YARN_HOME</value>
</property>F
ormatting HDFS:
5.Openacommandpromptandrun: hdfs namenode format
StartingHadoopServices:
6.InCommandpromptrun
cd%HADOOP_HOME%
start-dfs.cmd start-yarn.cmd
Testing Hadoop:
7.Openabrowserandvisit‘http://localhost:9870’ toseeHadoopnamenode Web interface.
RunningaMapReduceExample:
8.Totest,runaMapReduceexample(replacejarfile)
Hadoopjar
share/hadoop/mapreduce/hadoop*-mapreduce-examples.jarPI 16 1000
…….
7.MapReduce

PREPARE:
DownloadMapReduceClient.jar
DownloadInput_file.txt Place both files in “C://”
HADOOPOPERATIONS:
1.Opencmdinadministrativemodeandmoveto “C:/Hadoop-2.8.0/sbin” and start the cluster.
2.Start-all.cmd
3.CreateaninputdirectoryinHDFS
hadoopfs-mkdir/input_dir
4.Copytheinputtextfilenamedinput_file.txtintheinputdirectoryof HDFS
hadoopfs-putC:input_file.txt/input_dir
Verifythefileinput_file.txtisavailableinHDFSinputdirectory.
Hadoopfs-ls/input_dir/
5.RunMapReduceClient.jarandalsoprovideinputandoutput directories.
hadoopjarC:/MapReduceClient.jar wordcount/input_dir/output_dir
6.Verifythecontentforgeneratedoutput file
hadoophdfs-cat/output_dir/
Someother usefulcommands:
7.ToleaveSafemode:
hadoophdfsadmin-safemodeleave

8.TodeletefilefromHDFSdirectory
hadoopfs-rm-r/input_dir/input_file.txt
9.To deletedirectoryfromHDFSdirectory
hadoopfs-rm-r/input_dir
……..
8.implement anapplication that stores big data
HBase/MongoDB/PigusingHadoop/R/Cassandra

SetuptheEnvironment:
Youshouldhavehadoop,HBase,MongoDB,Pig,R,andCassandra installed and Configured.
Sampledataset:
Letassumeyouhaveasimpledatasetnamed“sample_data.csv”likethis: Name, Age, City
John,30,New York Alice,25,LosAngles Bob, 35, Chicago
HadoopMapReduce(Python)
PythonmapreducescripttoprocessthedatasetandstoreitinHBase: #mapper
importsys
forlineinsys.stdin:
line = line.strip() fields=line.split(“,”)
Name,age,city = fields print(f‘{name}\t{age}\t{city}’)
#reducer
importhappybase
connection=happybase.connect(host=localhost,port=9090) table =
connection.table('my_table')
forlineinsys.stdin:
name, age, city = line.strip().split("\t') table.put(name,('info:age':age,'info:city':city)
PigScript:
APigscripttoperformsometransformationsandstorethedatain MongoDB:
data=LOAD'sampledata.csv'USINGPigStorage('')AS (name:chararray, age:int,
city:chararray);
filtereddata=FILTERdataBYage>=30;
STORE filtered_data INTO 'mongodb://localhost:27017/mydb.mycollection' USING
org.apache.pig.backend.hadoop.executionengine. mapReducelayer. PigMongoSt orage();
RScript:
AnRscripttoanalyzethedata: library(rhbase)
hbase<-HBaseSnew(host="localhost",port=9090) data <- hbasesget("my_table")
print(data)
Cassandra(CQL):
YouwouldcreateaCassandrakeyspace,defineatableschema,and insert data using CQL
commands. Below is a simplified example.
cql
CREATEKEYSPACEmykeyspaceWITHreplication=('class': 'SimpleStrategy',
'replication_factor': 1};
USEmykeyspace;
CREATETABLEmytable(nameTEXTPRIMARY KEY,age INT, city TEXT);
INSERTINTOmytable(name,age,city)VALUES(John',30, 'New York');
RunningtheCode:
RunthePythonMapReducescriptusingHadoop,executethePig script, run the R script, and
execute the CQLcommands inCassandra.

……..
9.UseapachesparkfordataAnalytics

CreatingSparkSession:
from pyspark import SparkContext frompyspark.sqlimportSparkSession from pyspark.sql.types import *
frompyspark.sql.functionsimport* from pyspark.sql.types import Row from datetime import datetime
InitializingSparkSession:
Sc = sparkContext() spark=SparkSession.builder.appName("PythonSparkSQLbasic example").config("spark.some.config.option","some-value").getOrCreate()
CreationofSparkRDD:
srecord=sc.parallelize([Row(roll_no=1,name="john",passed=True,marks={'Math':89,'Physics':87,'Chemistry':96},sports=['chess','football'],DoB=datetime(2012,5,1,12,1,5)),
row(roll_no=2,name="Vignesh",passed=False,marks={'Math':95,'Physics':66,'Chemistry':77},sports=['carrom','tennis'],DoB=datetime(2012,5,12,14,2,5)),
Row(roll_no=3,name="Sidharth",passed=True,marks={'Math':95,'Physics':100,'Chemistry':95},sports=['football','kabadi'],DoB=datetime(2012,5,14,12,2
,5))])
CreatingaDataFrame:
srdf=srecord.toDF() srdf.show()
CreateTemporaryView:
srdf.createOrReplaceTempView('records') spark.sql("SELECT * FROM records").show()re=spark.sql("SELECT*FROMrecords") type(re)
AccessingElementsofaListorDictionarywithinDataFrame:
spark.sql('SELECTroll_no,marks["Physics"],sports[1]FROM records').show()
UsageofWhereClause:
spark.sql("SELECT * FROM records where passed= True").show()
spark.sql('SELECT*FROMrecordswheremarks["Chemistry"]<40').show()
CreatingGlobalView:
srdf.createGlobalTempView('globalrecord')
spark.sql("SELECT *FROMglobal_temp.globalrecord").show()
DroppingColumnsFromDataFrame:
srdf.columns srdf=srdf.drop('passed')
Fewmorequeries:
spark.sql("SELECTround((marks.Physics+marks.Chemistry+marks.Math)/3)avg_marks FROM records").show()
srdf=spark.sql("SELECT *,round((marks.Physics+marks.Chemistry+marks.Math)/3)avg_marksFROM records")
srdf.show()
……
10.FilterandsortthesalesdatausingPiglatininPig

1.Preparethesalesdatasetintheformatsales.csv
Transaction_id,customer,amount
1, Alice, 500
2, Bob, 1500
3, Charlie, 700

4, Diana, 2000
5, Eve, 1200
2.UploadthisfiletoHDFS:
UploadthiscsvfiletothehdfsbyusingtheCommands:
hdfsdfs-mkdir/input
hdfsdfs-putsales.csv/input/
3.Writethescriptnamedfilter_and_sortandsavethisfileintheextension.pig
Afteruploadingdatasettohdfs,writepigscriptinsaveit intheextension.pig
PigScript:
LoadthesalesdatafromtheHDFS:
sales_data=LOAD'/input/sales.csv'USINGPigStorage(',')AS (transaction_id:int,
customer:chararray, amount:float);
Filter transactions with amounts greater than1000:
filtered_data=FILTERsales_dataBYamount>1000;
Sort the filtered data by amount in descending order:
sorted_data=ORDERfiltered_dataBYamountDES;
StoretheresultinHDFS:
STOREsorted_dataINTO'/output/sorted_sales'USINGPigStorage(',');
4.RunthePigScript:
Runtheabovepigscriptbyusingthecommand intheCMD: Verify Output directory:
hdfsdfs-ls/output/sorted_sales
Viewtheoutput: