Download VirtualBox: <https://download.virtualbox.org/virtualbox/7.0.14/VirtualBox-7.0.14-161095-Win.exe>

To copy inside out:

Go to settings → Advanced → Mark *Shared Clipboard* and *Drag and Drop* to **Bidirectional**

# Practical 1

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

1. sudo apt-get update
2. sudo apt-get install build-essential
3. Install jdk and ssh using the below command.
   1. sudo apt-get install openjdk-8-jdk
   2. sudo apt install openssh-server
4. Add a new group “hadoop” and add the hadoop user in it.
   1. sudo addgroup hadoop
   2. sudo adduser --ingroup hadoop hduser
   3. sudo usermod -aG sudo hduser
   4. su hduser
5. Create a ssh keygen for the user.
   1. ssh-keygen -t rsa -P ""
   2. cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys
6. Execute the below commands to install Hadoop
   1. cd /usr/local
   2. sudo wget <https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0.tar.gz>
   3. sudo tar xvf hadoop-3.4.0.tar.gz
   4. sudo mv hadoop-3.4.0 hadoop
   5. sudo chown -R hduser: hadoop Hadoop
7. Add the below lines to bashrc file.
   1. To open the file use the command: **nano ~/.bashrc**

export HADOOP\_HOME=/usr/local/hadoop

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64

unalias fs &> /dev/null

alias fs="hadoop fs"

unalias hls &> /dev/null

alias hls="fs -ls"

lzohead () {

hadoop fs -cat $1 | lzop -dc | head -1000 | less

}

export PATH=$PATH:$HADOOP\_HOME/bin

1. To apply the bashrc changes run this command: **source ~/.bashrc**
2. Add the following line to hadoop-env.sh

cd /usr/local/hadoop/etc/hadoop

sudo nano hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64

1. Create a tmp folder in /app/hadoop/tmp and change the owner to hduser.
2. Add the below lines in core-site.xml file

<property>

<name>hadoop.tmp.dir</name>

<value>/app/hadoop/tmp</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. The

uri's scheme determines the config property (fs.SCHEME.impl) naming

the FileSystem implementation class. The uri's authority is used to

determine the host, port, etc. for a filesystem.</description>

</property>

1. Add the below lines in mapred-site.xml

<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs

at. If "local", then jobs are run in-process as a single map

and reduce task.

</description>

</property>

1. Add the below lines in hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

1. Format namenode using the command - “hadoop namenode -format”
2. start hadoop

ssh localhost

/usr/local/hadoop/sbin/start-all.sh

1. Access localhost:9870 to get nanmenode status

# Practical 2

*Implement word count/ frequency program using MapReduce*

1. Start hadoop
2. Create a text file with some repeated words

sudo nano bda.txt

1. Move bda.txt file to HDFS

hdfs dfs -put /home/hduser/bda.txt /

1. Running MapReduce for wordcount file bda.txt

hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples3.2.3.jar wordcount /bda.txt /output

1. Check output

hdfs dfs -head /output/part-r-00000

1. To get output in a .txt file in HDFS

hdfs dfs -mv /output/part-r-00000 /output/opt.txt

1. To check the file system in HDFS: hdfs dfs -ls /

# Practical 3

*Implement a MapReduce program that processes a weather dataset*

1. Start hadoop
2. Download the sample-weather dataset and create the jar file from the java file and load the .txt and .jar in

the default file location

Data link: <https://www.ncei.noaa.gov/pub/data/uscrn/products/daily01/>

<https://www.geeksforgeeks.org/mapreduce-program-weather-data-analysis-for-analyzing-hot-and-cold-days/>

1. Load the input dataset into HDFS

hdfs dfs -put CRND0103-2020-AK\_Fairbanks\_11\_NE.txt /Pract3

1. Run mapreduce on the input file

hadoop jar temperature.jar MyMaxMin /user/hduser/practical3/CRND0103-2020- AK\_Fairbanks\_11\_NE.txt /user/hduser/practical3/WeatherOutput

1. Check the Output

hdfs dfs -cat /user/hduser/practical3/WeatherOutput/part-r-00000

# Practical 4

*Implement an application that stores big data in HBase and manipulate it using Python*

1. Change the pwd to the path where you want to install Hbase

Download, Extract, Rename the directory to Hbase and change working directory to hbase/conf.

sudo wget https://dlcdn.apache.org/hbase/2.5.8/hbase-2.5.8-bin.tar.gz

sudo tar xzvf hbase-2.4.11-bin.tar.gz

sudo mv hbase-2.4.11 hbase

cd hbase/conf/

1. Add the following line to /conf/hbase-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk-amd64

1. Add the following lines in hbase-site.xml

<property>

<name>hbase.rootdir</name>

<value>file:///usr/local/hbase</value>

</property>

<property>

<name>hbase.zookeeper.property.dataDir</name>

<value>/usr/local/hbase/zookeeper</value>

</property>

1. Change the rights of hbase
2. Start Hbase

In hbase/bin : ./start-hbase.sh

1. Using hbase through shell
2. Create a table in Hbase

create ‘test’, ‘cf’

1. Insert some data into table using put command.

put 'test', 'row1', 'cf:a', 'value1'

put 'test', 'row2', 'cf:b', 'value2'

put 'test', 'row3', 'cf:c', 'value3'

1. View the inserted data using the scan command

scan ‘test’

1. Access the created data using python Install python first and then install the happybase library.

sudo apt install python3-pip

pip install happybase

1. Start the thrift server then start HBaseStart the thrift server then start HBase

./hbase-daemon.sh start thrift

1. Connect to the HBase database and check the data using the python shell. Exit when done

python3

import happybase as hb

conn = hb.Connection(‘127.0.0.1’, 9000)

conn.table(‘test’).row(‘row2’)

conn.table(‘test’).row(‘row1’)

conn.table(‘test’).row(‘row3’)

exit()

1. Stop HBase then stop thrift server

./stop-hbase.sh

./hbase-daemon.sh stop thrift

# Practical 5

*Implement the above practical using Pig*

1. Start Hadoop
2. Change the pwd to the path where you want to install Pig

Download, extract and rename the pig tar file.

cd /usr/local

sudo wget <https://downloads.apache.org/pig/pig-0.17.0/pig-0.17.0.tar.gz>

sudo tar -xvzf pig-0.17.0.tar.gz

sudo mv pig-0.17.0 pig

1. Add the Pig environment variables in bashrc

export PIG\_HOME=/usr/local/pig

export PATH=$PATH:$PIG\_HOME/bin

export PIG\_CLASSPATH=$HADOOP\_HOME/conf

1. Check pig version
2. Create a database file.

sudo nano products.txt

1. Run the pig in Local mode

pig -x local

product = LOAD 'products.txt' USING PigStorage(',');

dump product;

quit

1. Running in HDFS mode

hdfs dfs -put products.txt /

1. Start pig

pig

product = LOAD 'hdfs://localhost:54310/products.txt' USING PigStorage(',');

dump product;

quit;

# Practical 6

*Configure the Hive and implement the application in Hive*

1. Start Hadoop
2. Change the pwd to the path where you want to install Hive

Download and extract the hive tar file.

sudo wget [hive download link]

sudo tar -xvzf apache-hive…tar.gz

sudo mv apache-hive hive

sudo chmod 777 hive

1. Add the HIVE\_HOME path in the bashrc file

export HIVE\_HOME=/usr/local/hive

export PATH=$PATH: $HIVE\_HOME/bin

1. Add the following lines in hive-config.sh (/usr/local/hive/bin)

export HADOOP\_HOME=/usr/local/hadoop

1. Hive is installed now. Create some directories in HDFS and give it permissions
2. Initialize the derby database

$HIVE\_HOME/bin/schematool -initSchema -dbType derby

1. Start hive using the ‘hive’ command
2. Create a database and a table in a hive.

create database company;

show databases;

create table employees (id int, name string, country string, department string, salary int) row format delimited fields terminated by ' ';

1. Load the data into a table from a file

load data local inpath "/home/hduser/employees.txt" into table employees;

1. Display the loaded data
2. Exit hive using ‘exit’ command

# Practical 7

*Implement Decision tree classification techniques*

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.datasets import load\_iris

data = load\_iris()

X = data.data

y = data.target

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['target'] = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

clf = DecisionTreeClassifier()

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

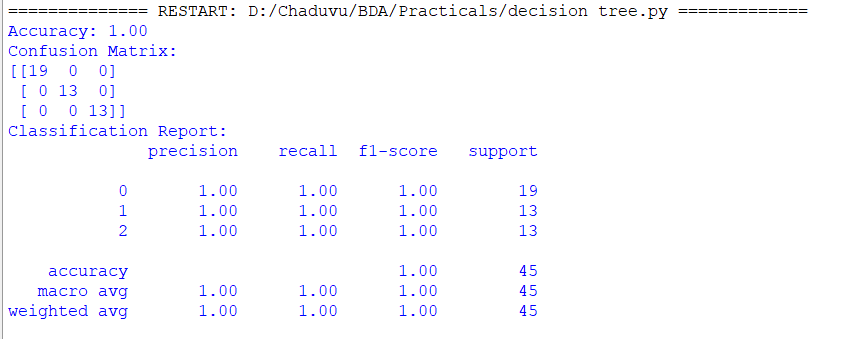
print('Confusion Matrix:')

print(conf\_matrix)

class\_report = classification\_report(y\_test, y\_pred)

print('Classification Report:')

print(class\_report)



# Practical 8

*Implement SVM classification techniques*

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.datasets import load\_iris

data = load\_iris()

X = data.data

y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

def evaluate\_svm(kernel, \*\*kwargs):

clf = SVC(kernel=kernel, \*\*kwargs)

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

class\_report = classification\_report(y\_test, y\_pred)

print(f'\n{kernel.capitalize()} Kernel')

if 'degree' in kwargs:

print(f"Degree: {kwargs['degree']}")

print(f'Accuracy: {accuracy:.2f}')

print('Confusion Matrix:')

print(conf\_matrix)

print('Classification Report:')

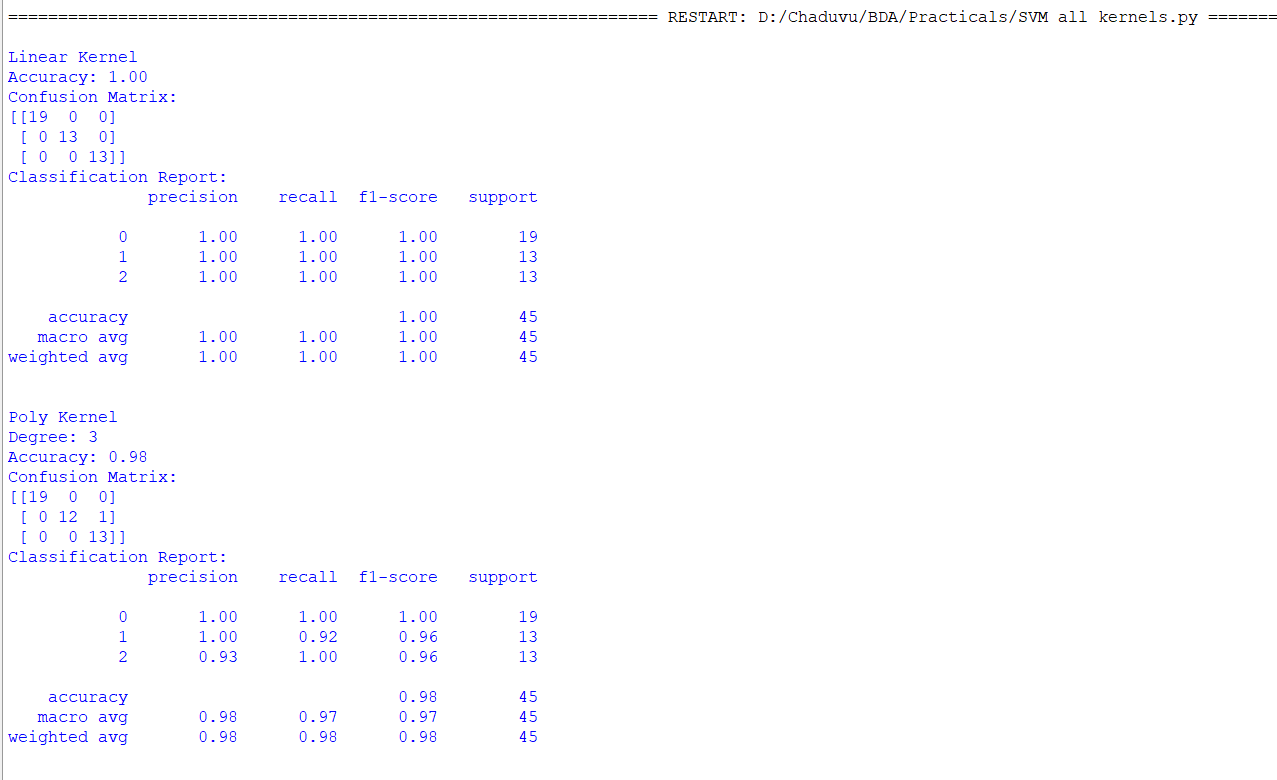
print(class\_report)

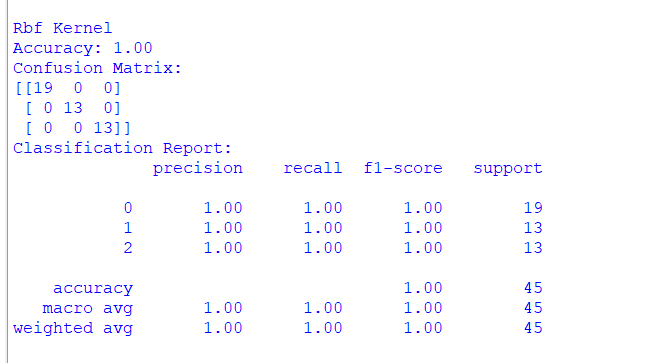
evaluate\_svm('linear')

evaluate\_svm('poly', degree=3) # You can specify the degree of the polynomial

evaluate\_svm('rbf')

evaluate\_svm('sigmoid')





# Practical 9

*REGRESSION MODEL*

*Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS).*

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

import statsmodels.api as sm

data = pd.read\_csv('Admission\_Predict.csv')

print(data.head())

print(data.isnull().sum())

data.dropna(inplace=True)

X = data[['GRE Score', 'CGPA', 'University Rating']]

y = data['Chance of Admit']

y = (y > 0.5).astype(int)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

log\_reg = LogisticRegression()

log\_reg.fit(X\_train, y\_train)

y\_pred = log\_reg.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print('Confusion Matrix:')

print(conf\_matrix)

class\_report = classification\_report(y\_test, y\_pred)

print('Classification Report:')

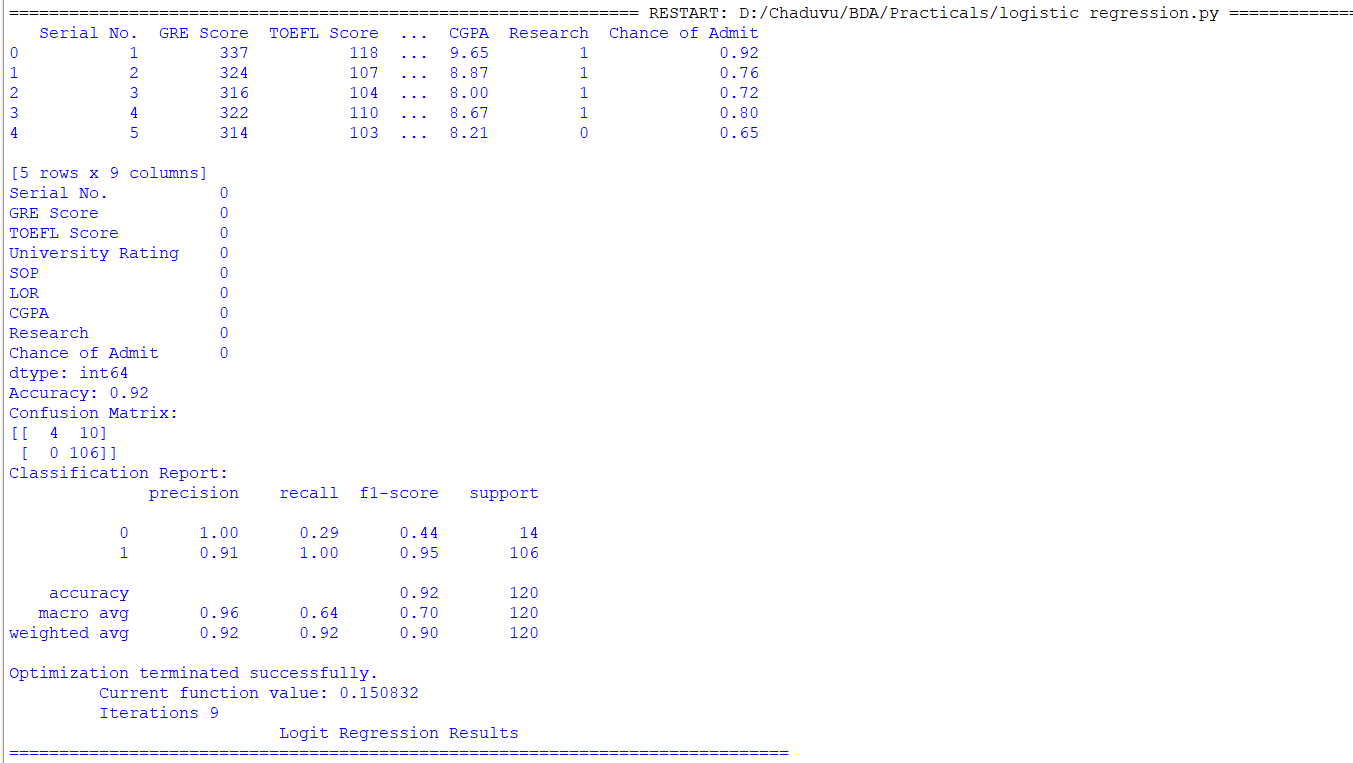
print(class\_report)

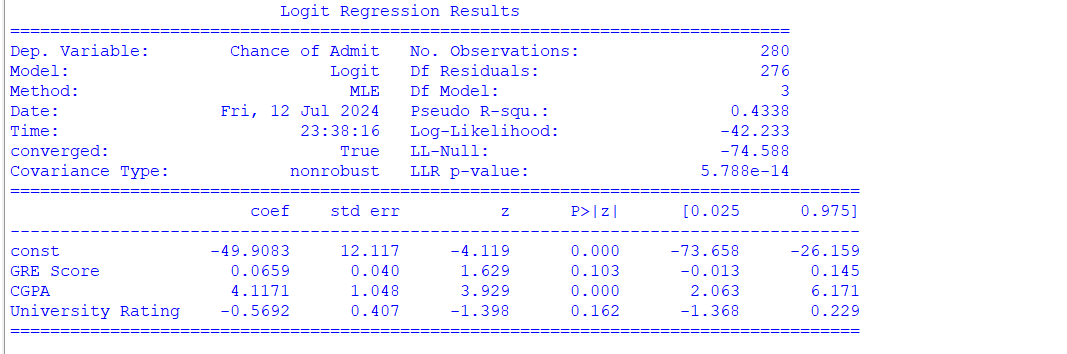
X\_train\_sm = sm.add\_constant(X\_train) # Add a constant term for the intercept

log\_reg\_sm = sm.Logit(y\_train, X\_train\_sm)

result = log\_reg\_sm.fit()

print(result.summary())





# Practical 10

*MULTIPLE REGRESSION MODEL*

*Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.*

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

import statsmodels.api as sm

data = pd.read\_csv('Admission\_Predict.csv')

print(data.head())

print(data.isnull().sum())

data.dropna(inplace=True)

X = data[['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA', 'Research']]

y = data['Chance of Admit']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

lin\_reg = LinearRegression()

lin\_reg.fit(X\_train, y\_train)

y\_pred = lin\_reg.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse:.2f}')

r2 = r2\_score(y\_test, y\_pred)

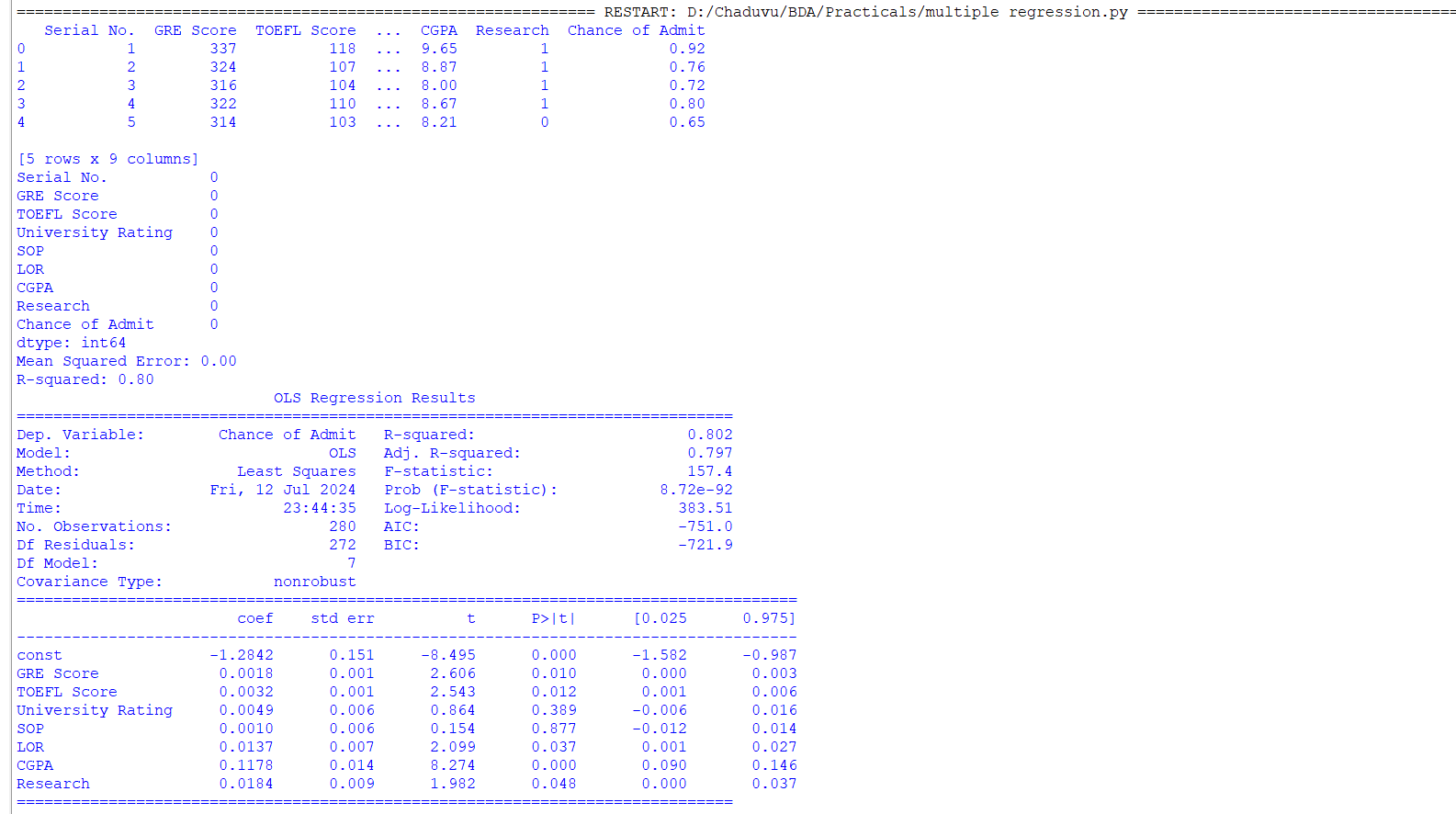
print(f'R-squared: {r2:.2f}')

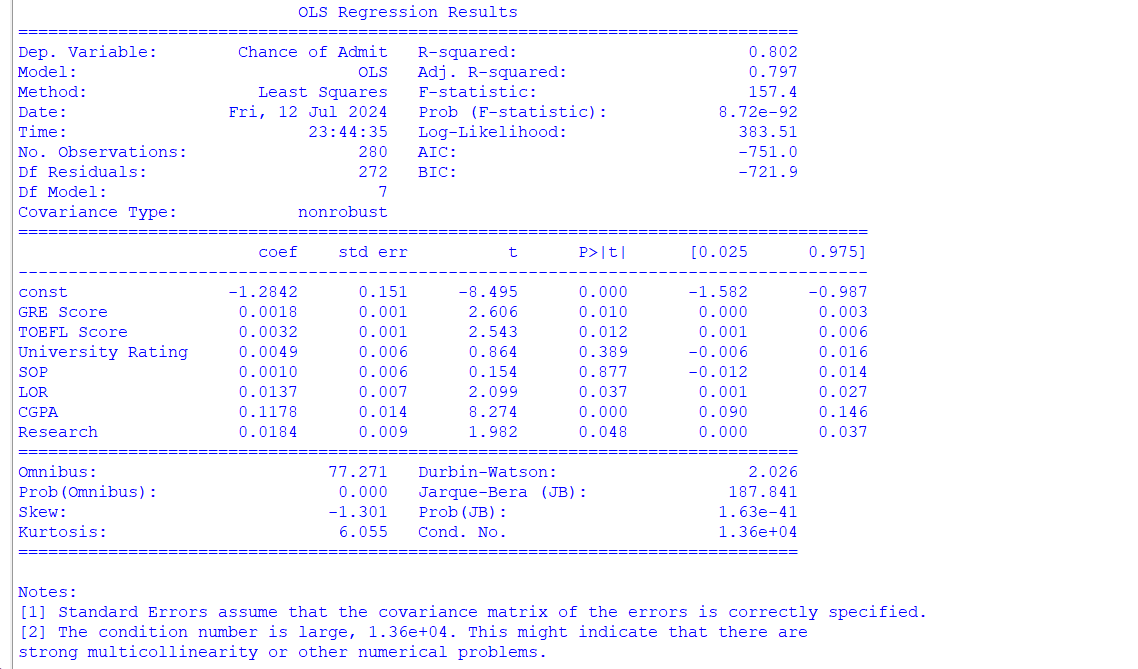
X\_train\_sm = sm.add\_constant(X\_train)

lin\_reg\_sm = sm.OLS(y\_train, X\_train\_sm)

result = lin\_reg\_sm.fit()

print(result.summary())





# Practical 11

*CLASSIFICATION MODEL*

*a. Install relevant package for classification.*

*b. Choose classifier for classification problem.*

*c. Evaluate the performance of classifier.*

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.datasets import load\_breast\_cancer

data = load\_breast\_cancer()

X = data.data

y = data.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

y\_pred = clf.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

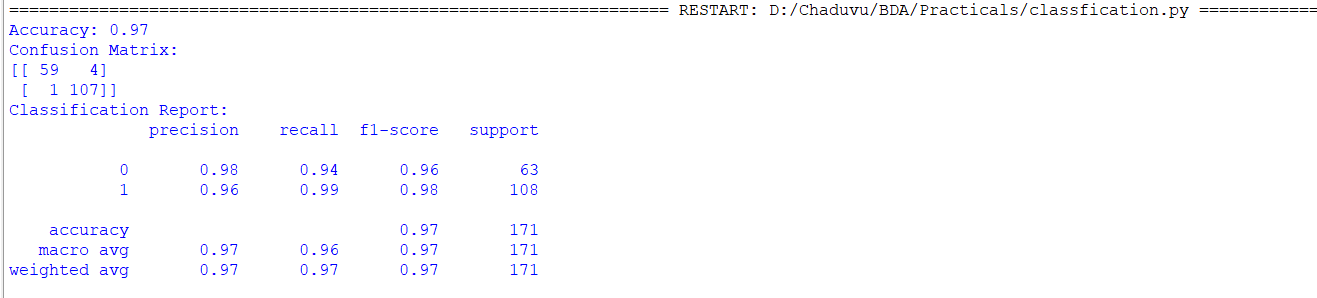
print('Confusion Matrix:')

print(conf\_matrix)

class\_report = classification\_report(y\_test, y\_pred)

print('Classification Report:')

print(class\_report)



# Practical 12

*CLUSTERING MODEL*

*a. Clustering algorithms for unsupervised classification.*

*b. Plot the cluster data using R visualizations*

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

X, \_ = make\_blobs(n\_samples=300, centers=3, cluster\_std=0.60, random\_state=0)

plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], s=50, cmap='viridis')

plt.title('Data Points')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

kmeans = KMeans(n\_clusters=3, random\_state=0)

clusters = kmeans.fit\_predict(X)

plt.figure(figsize=(8, 6))

plt.scatter(X[:, 0], X[:, 1], c=clusters, s=50, cmap='viridis', label='Data Points')

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], marker='x', s=200, color='red', label='Centroids')

plt.title('Clusters and Centroids')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.legend()

plt.show()

