DATA SCIENCE

LAB RECORD FILE

(CBCPE01)

NETAJI SUBHASH UNIVERSITY OF TECHNOLOGY

EAST CAMPUS

SUBMITTED TO:-

Dr. Amita Jain

Assistant Professor

Department of Computer Science

And Engineering

SUBMITTED BY:-

Jyoti Sinha

2021UCB6001

B.Tech – CSDA

5TH Semester

INDEX

EXPERIMENT - 1

Install,configure,and run Hadoop and HDFS

Steps of installing Apache Hadoop on WSL Ubuntu 22.04 (Jammy, terminal-based). These steps remain the same for the GUI version of Ubuntu.

Installing Java

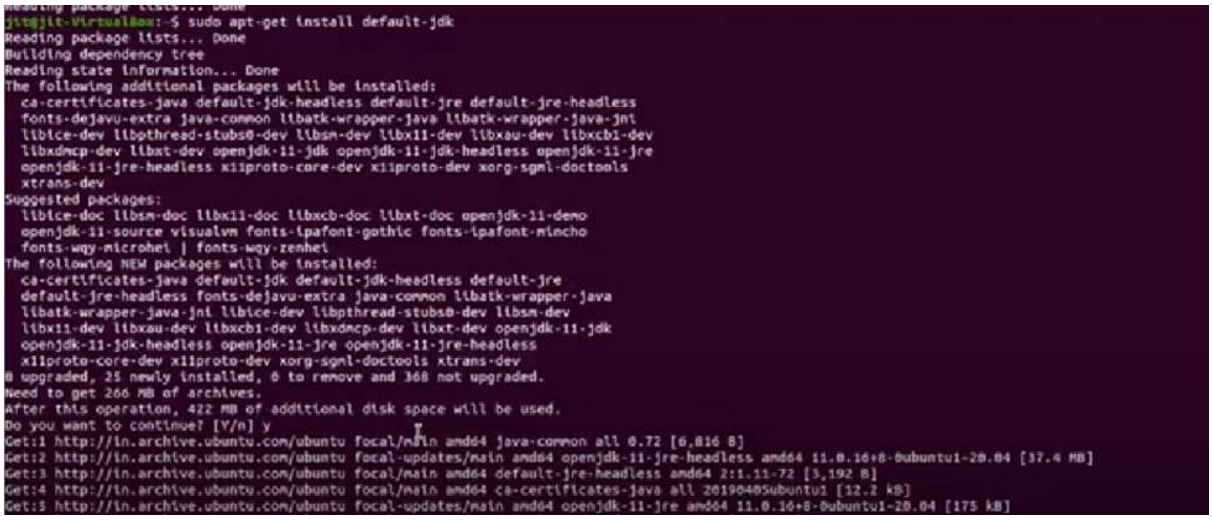
If you are on a GUI version of Ubuntu, the following commands are to be executed inside the shell window.

1. Install OpenJDK 11 (Java Development Kit and Java Runtime Environment:

```
sudo apt install openjdk-11-jdk openjdk-11-jre
```

2. Check if java and javac are installed:

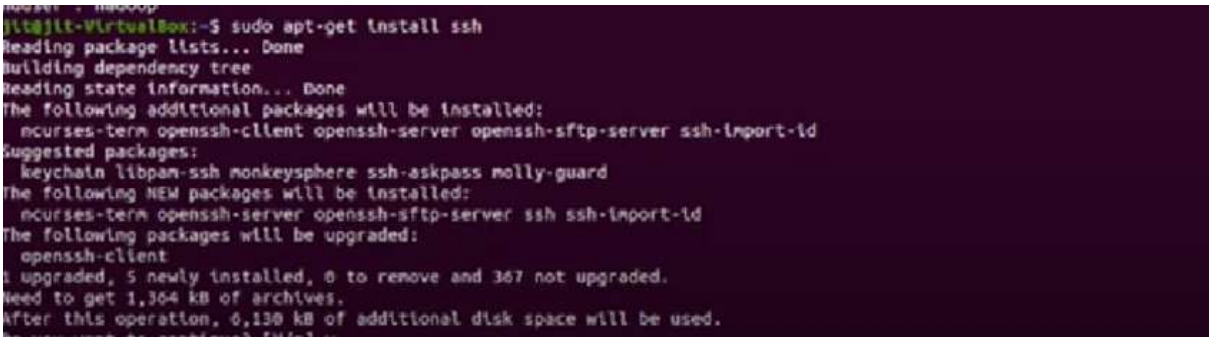
```
java -version && javac -version
```



Installing OpenSSH

1. Install OpenSSH Server and Client:

```
sudo apt install openssh-server openssh-client pdsh
```



Setting up a New User

1. Create a new user for running Hadoop-related tools:

```
sudo useradd -m -s /bin/bash Hadoop
```

2. Create a password for the new user:

```
sudo passwd Hadoop
```

You will be prompted to enter a new password. Verify and retype the password to set it to the Hadoop user account.

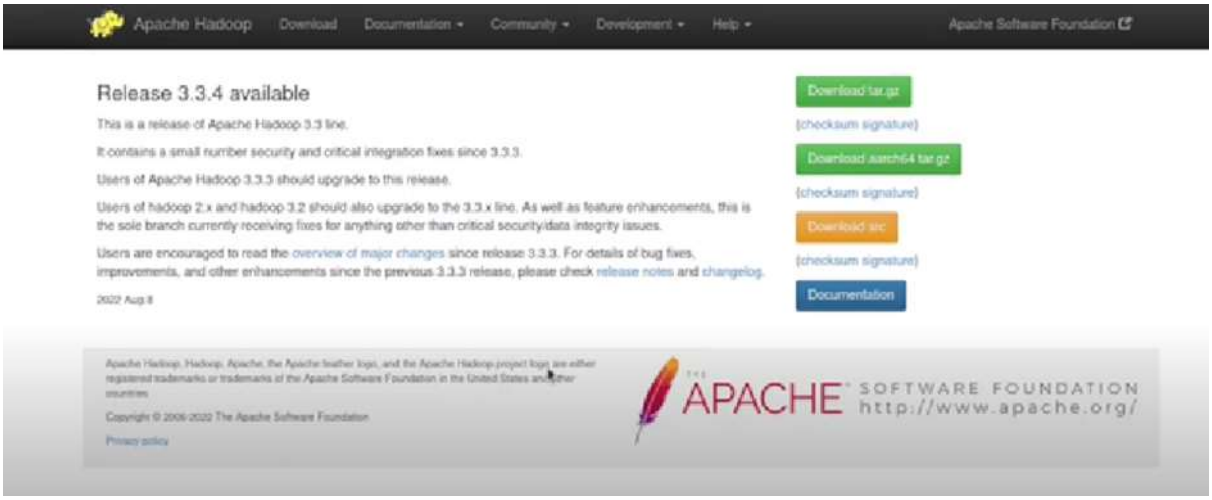
3. Add Hadoop to the list of super users:

```
sudo usermod -aG sudo hadoop
```

4. Log in to the hadoop user account:

```
su – Hadoop
```

You will be prompted to enter the password. Enter the one you set before to login.



Setting up Tools

1. Generate RSA key for SSH instance:

```
ssh-keygen -t rsa
```

2. Append the contents of the id_pub.rsa file to authorized_keys:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

3. Change the permissions of the authorized_keys file:

```
chmod 600 ~/.ssh/authorized_keys
```

Downloading Hadoop

1. Connect to localhost using SSH:

```
ssh localhost
```

2. Download Apache Hadoop from the official CDN:

```
wget
https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3
.6-src.tar.gz
```

3. Extract the Hadoop archive:

```
tar -xvzf hadoop-3.3.6-src.tar.gz
```

4. Move the extracted folder to /usr/local/hadoop:

```
sudo mv hadoop-3.3.6 /usr/local/hadoop
```

5. Change the ownership permissions:

```
sudo chown -R hadoop:hadoop /usr/local/Hadoop
```

Setting up Hadoop

1. Go back to the local user directory and open .bashrc:

```
cd ~; vi .bashrc
```

2. Add these lines to the end of the file:

```
# Hadoop environment variables
 export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
 export HADOOP_HOME=/usr/local/hadoop
 export HADOOP_INSTALL=$HADOOP_HOME
 export HADOOP_MAPRED_HOME=$HADOOP_HOME
 export HADOOP_COMMON_HOME=$HADOOP_HOME
 export HADOOP_HDFS_HOME=$HADOOP_HOME
 export HADOOP_YARN_HOME=$HADOOP_HOME
 export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
 export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
 export
HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

3. Refresh the shell:

```
source .bashrc
```

4. Check if the newly added variables work:

```
echo $JAVA_HOME && \
 echo $HADOOP_HOME && \
 echo $HADOOP_OPTS && \
 nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

5. Check if Hadoop is properly installed:

```
hadoop version
```

Configuring Hadoop

Configure the following files as shown in this section.

1. Add the following lines to the $HADOOP_HOME/etc/hadoop/core-site.xml file:

```
<configuration>
     <property>
     <name>fs.default</name>
     <value>hdfs://localhost:9000</value>
     </property>
 </configuration>
```

2. Create directories namenode and datanode, and change the ownership:

```
sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}; \
 sudo chown -R hadoop:hadoop /home/hadoop/hdfs
```

3. Add the following lines to the $HADOOP_HOME/etc/hadoop/hdfs-site.xml:

```
<configuration>
     <property>
     <name>dfs.replication</name>
     <value>1</value>
     </property>
     <property>
     <name>dfs.name.dir</name>
     <value>file://home/hadoop/hdfs/namenode</value>
     </property>
     <property>
     <name>dfs.data.dir</name>
     <value>file://home/hadoop/hdfs/datanode</value>
```

```
     </property>
 </configuration>
```

4. Format namenode and start dfs.sh:

```
hdfs namenode -format
 start-dfs.sh
```

5. Add the following lines to the $HADOOP_HOME/etc/hadoop/mapred-site.xml:

```
<configuration>
     <property>
     <name>mapreduce.framework.name</name>
     <value>yarn</value>
     </property>
     <property>
     <name>mapreduce.application.classpath</name>

<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MA
PRED_HOME/share/hadoop/mapreduce/lib/*</value>
     </property>
 </configuration>
```

6. Add the following lines to the $HADOOP_HOME/etc/hadoop/yarn-site.xml:

```
<configuration>
     <property>
     <name>yarn.nodemanager.aux-services</name>
     <value>mapreduce_shuffle</value>
     </property>
     <property>
     <name>yarn.nodemanager.env-whitelist</name>

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CO
NF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOM
E,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
     </property>
 </configuration>
```

7. Start yarn.sh:

```
Start-yarn.sh
```

```
<!-- Put site-specific property overrides in this file. -->

<configuration>
<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.
used to
The uri's authority is
determine the host, port, etc. for a filesystem.</description>
</property>
</configuration>
```

```
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/css/site.css
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/breadcrumbs.jpg
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/apache-maven-project-2.png
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/maven-logo-2.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/collapsed.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/logo_maven.jpg
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/icon_warning_sml.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/logos/
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/logos/build-by-maven-black.pn
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/logos/maven-feather.png
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/logos/build-by-maven-white.pn
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/banner.jpg
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/h5.jpg
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/icon_error_sml.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/icon_success_sml.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/expanded.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/external.png
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/icon_info_sml.gif
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/logo_apache.jpg
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/bg.jpg
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/newwindow.png
hadoop-3.3.4/share/doc/hadoop/hadoop-hdfs-nfs/images/h3.jpg
hduser@jit-VirtualBox:~$ ls
hadoop-3.3.4  hadoop-3.3.4.tar.gz
hduser@jit-VirtualBox:~$ sudo mkdir -p /usr/local/hadoop
[sudo] password for hduser:
hduser@jit-VirtualBox:~$ cd hadoop-3.3.4/
hduser@jit-VirtualBox:~/hadoop-3.3.4$ sudo mv * /usr/local/hadoop
hduser@jit-VirtualBox:~/hadoop-3.3.4$ ls
hduser@jit-VirtualBox:~/hadoop-3.3.4$ cd
hduser@jit-VirtualBox:~$ sudo chown -R hduser:hadoop /usr/local/hadoop
```

Output

```
hadoop@LAPTOP-R736CCLJ:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [LAPTOP-R736CCLJ]
```

```
hadoop@LAPTOP-R736CCLJ:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

```
hadoop@LAPTOP-R736CCLJ:~$ jps
4389 ResourceManager
5032 Jps
3821 NameNode
4174 SecondaryNameNode
4526 NodeManager
```

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Overview 'localhost:9000' (✔active)

Started:	Sun Nov 05 23:57:21 +0530 2023
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 13:52:00 +0530 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-14ba33b3-9c7c-42cc-959d-bcd1635b52f7
Block Pool ID:	BP-2116377916-127.0.1.1-1697379470851

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 85.95 MB of 151 MB Heap Memory. Max Heap Memory is 1.86 GB.

Non Heap Memory used 51.58 MB of 54.69 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	0 B
Configured Remote Capacity:	0 B
DFS Used:	0 B (100%)
Non DFS Used:	0 B
DFS Remaining:	0 B (0%)
Block Pool Used:	0 B (100%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion (including replicas)	0
Block Deletion Start Time	Sun Nov 05 23:57:21 +0530 2023
Last Checkpoint Time	Sun Nov 05 23:57:21 +0530 2023
Enabled Erasure Coding Policies	RS-6-3-1024k

NameNode Journal Status

Current transaction ID: 26

Journal Manager	State
FileJournalManager(root=/home/hadoop/hadoopdata/hdfs/namenode)	EditLogFileOutputStream(/home/hadoop/hadoopdata/hdfs/namenode/current/edits_inprogress_0000000000000000026)

NameNode Storage

Storage Directory	Type	State
/home/hadoop/hadoopdata/hdfs/namenode	IMAGE_AND_EDITS	Active

DFS Storage Types

Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service

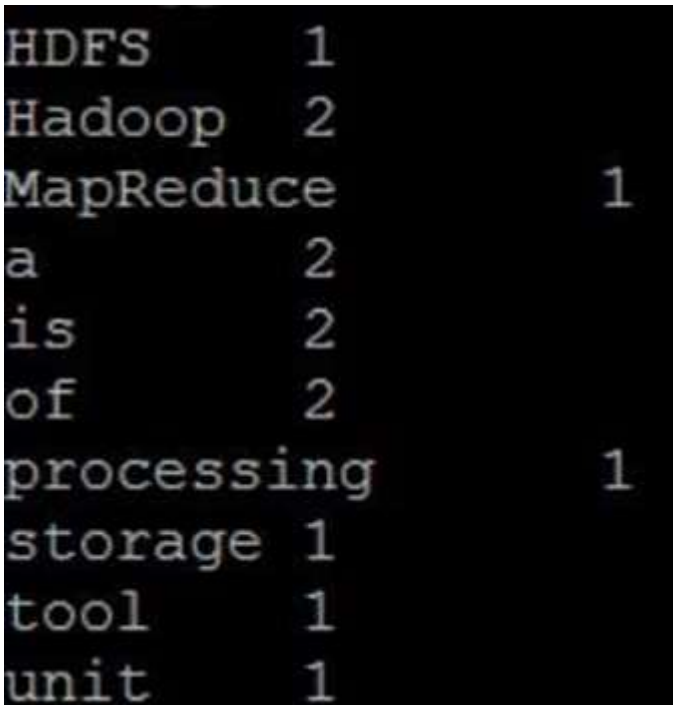
EXPERIMENT - 2

Implement word count/frequency programs using MapReduce

```
import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable; import
org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.MapReduceBase; import
org.apache.hadoop.mapred.Mapper;

import org.apache.hadoop.mapred.OutputCollector; import
org.apache.hadoop.mapred.Reporter;

public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable> {

private final static IntWritable one = new IntWritable(1);private
Text word = new Text();

public void map(LongWritable key, Text value,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws
IOException {

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line); while
(tokenizer.hasMoreTokens()) {

word.set(tokenizer.nextToken()); output.collect(word, one);

           }

     }

}

import java.io.IOException; import java.util.Iterator;

import org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapred.MapReduceBase; import
org.apache.hadoop.mapred.OutputCollector; import
org.apache.hadoop.mapred.Reducer;

import org.apache.hadoop.mapred.Reporter;

public class WC_Reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {

public voidreduce(Text key, Iterator<IntWritable> values,
OutputCollector<Text,IntWritable> output, Reporter reporter) throws
IOException {

int sum=0;

while (values.hasNext()) { sum+=values.next().get()

output.collect(key,new IntWritable(sum));}}

import java.io.IOException; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable; import
org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.FileInputFormat; import
org.apache.hadoop.mapred.FileOutputFormat; import
org.apache.hadoop.mapred.JobClient;

import org.apache.hadoop.mapred.JobConf;

import org.apache.hadoop.mapred.TextInputFormat; import
org.apache.hadoop.mapred.TextOutputFormat;

public class WC_Runner {

public static void main(String[] args) throws IOException { JobConf
conf = new JobConf(WC_Runner.class);

conf.setJobName("WordCount"); conf.setOutputKeyClass(Text.class);
conf.setOutputValueClass(IntWritable.class);
conf.setMapperClass(WC_Mapper.class);
conf.setCombinerClass(WC_Reducer.class);
conf.setReducerClass(WC_Reducer.class);
conf.setInputFormat(TextInputFormat.class);
conf.setOutputFormat(TextOutputFormat.class);
FileInputFormat.setInputPaths(conf,new Path(args[0]));

FileOutputFormat.setOutputPath(conf,newPath(args[1]));
JobClient.runJob(conf);}}
```

Output


HDFS 1
Hadoop 2
MapReduce 1
a 2
is 2
of 2
processing 1
storage 1
tool 1
unit 1

EXPERIMENT - 3

Implement a MR program that processes a weather dataset

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.conf.Configuration;
//Mapper
public class MyMaxMin {
public static class MaxTemperatureMapper extends
Mapper<LongWritable, Text, Text, Text> {
@Override
public void map(LongWritable arg0, Text Value, Context context)
throws IOException, InterruptedException {
String line = Value.toString();
// Check for the empty line
```

```
if (!(line.length() == 0)) {
float temp_Max = Float.parseFloat(line.substring(39, 45).trim());
float temp_Min = Float.parseFloat(line.substring(47, 53).trim());
if (temp_Max > 30.0) {
// Hot day
context.write(new Text("The Day is Hot Day :" + date),
new Text(String.valueOf(temp_Max)));
}
// if the minimum temperature is
// less than 15, it is a cold day
if (temp_Min < 15) {
// Cold day
context.write(new Text("The Day is Cold Day :" + date),
new Text(String.valueOf(temp_Min)));
                }
           }
     }
}
// Reducer
public static class MaxTemperatureReducer extends
Reducer<Text, Text, Text, Text> {
public void reduce(Text Key, Iterator<Text> Values, Context context)
throws IOException, InterruptedException {
String temperature = Values.next().toString();
context.write(Key, new Text(temperature));
     }
```

```
}
public static void main(String[] args) throws Exception {
Configuration conf = new Configuration();
Job job = new Job(conf, "weather example");
job.setJarByClass(MyMaxMin.class);
job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setMapperClass(MaxTemperatureMapper.class);
job.setReducerClass(MaxTemperatureReducer.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path OutputPath = new Path(args[1]);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
OutputPath.getFileSystem(conf).delete(OutputPath);
System.exit(job.waitForCompletion(true) ? 0 : 1);
     }
}
```

Output

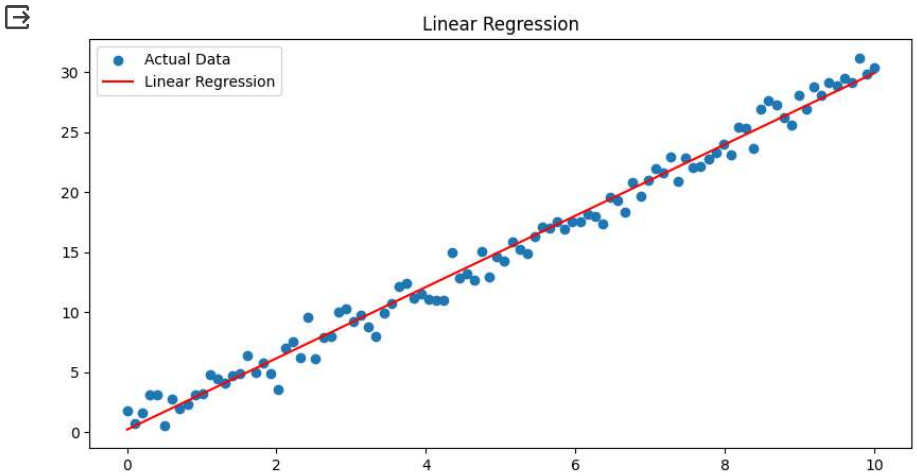
```
1 The Day is Cold Day :20200101   -21.8
2 The Day is Cold Day :20200102   -23.4
3 The Day is Cold Day :20200103   -25.4
4 The Day is Cold Day :20200104   -26.8
5 The Day is Cold Day :20200105   -28.8
6 The Day is Cold Day :20200106   -30.0
7 The Day is Cold Day :20200107   -31.4
8 The Day is Cold Day :20200108   -33.6
9 The Day is Cold Day :20200109   -26.6
10 The Day is Cold Day :20200110   -24.3
```

EXPERIMENT - 4

Implement Linear and logistic Regression.

LINEAR REGRESSION

```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Generate random dataset
np.random.seed(0)
X = np.linspace(0, 10, 100)
y = 3*X + np.random.normal(0, 1, 100)  # Linear relationship with some noise

# Reshape X for sklearn compatibility
X = X.reshape(-1, 1)

# Linear Regression
lr = LinearRegression()
lr.fit(X, y)

# Generate predictions
y_pred = lr.predict(X)

# Plot the results
plt.figure(figsize=(10, 5))
plt.scatter(X, y, label='Actual Data')
plt.plot(X, y_pred, color='red', label='Linear Regression')
plt.title('Linear Regression')
plt.legend()
plt.show()
```



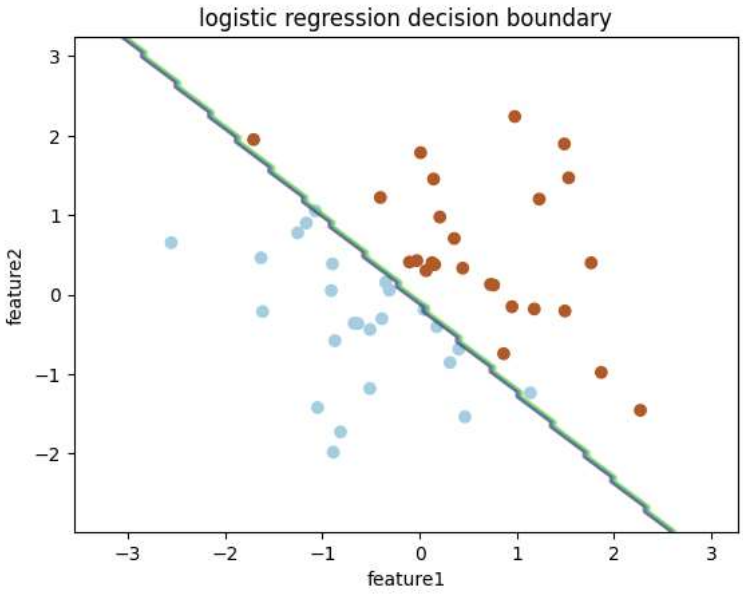
LOGISTIC REGRESSION

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Generate some sample data
np.random.seed(0)
X = np.random.randn(50,2)
y = (X[:, 0] + X[:, 1] > 0).astype(int)  # Binary classification task
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model
model = LogisticRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print(f"Confusion Matrix:\n{confusion}")
print(f"Classification Report:\n{classification_rep}")
```

```
Accuracy: 1.0
Confusion Matrix:
[[8 0]
 [0 2]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         8
           1       1.00      1.00      1.00         2

    accuracy                           1.00        10
   macro avg       1.00      1.00      1.00        10
weighted avg       1.00      1.00      1.00        10
```

```
xx , yy = np.meshgrid(np.linspace(X[:,0].min() - 1 , X[:,0].max() + 1 , 100) ,
                      np.linspace(X[:,1].min() - 1 , X[:,1].max() + 1 , 100)
                      )

Z = model.predict(np.c_[xx.ravel(),yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contour(xx,yy,Z,alpha = 0.4)
plt.scatter(X[:,0] , X[:,1], c = y , cmap = plt.cm.Paired)
plt.xlabel("feature1")
plt.ylabel("feature2")
plt.title("logistic regression decision boundary")
plt.show()
```



EXPERIMENT - 5

Implement SVM/Decision tree classification techniques.

SUPPORT VECTOR MACHINE

```
import numpy as np
import pandas as pd
from sklearn.svm import SVC


data = pd.read_csv('/content/cell_samples (1).csv')
data
```

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	N
0	1000025	5	1	1	1	2	1	3	
1	1002945	5	4	4	5	7	10	3	
2	1015425	3	1	1	1	2	2	3	
3	1016277	6	8	8	1	3	4	3	
4	1017023	4	1	1	3	2	1	3	
...	...	...	...	...	...	...	...	...	
694	776715	3	1	1	1	3	2	1	
695	841769	2	1	1	1	2	1	1	
696	888820	5	10	10	3	7	3	8	
697	897471	4	8	6	4	3	4	10	
698	897471	4	8	8	5	4	5	10	

699 rows × 11 columns

```
data = data.apply(pd.to_numeric, errors='coerce')
data = data.fillna(data.mean())


data
```

	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	I
0	1000025	5	1	1	1	2	1.0	3	
1	1002945	5	4	4	5	7	10.0	3	
2	1015425	3	1	1	1	2	2.0	3	
3	1016277	6	8	8	1	3	4.0	3	
4	1017023	4	1	1	3	2	1.0	3	
...	...	...	...	...	...	...	...	...	
694	776715	3	1	1	1	3	2.0	1	
695	841769	2	1	1	1	2	1.0	1	
696	888820	5	10	10	3	7	3.0	8	

```
from sklearn.model_selection import train_test_split
tr, te = train_test_split(data, test_size = 0.3, random_state = 42)
```

```
# from sklearn.svm import NuSVC
svc = SVC(kernel = 'linear', gamma = 'auto')
svc.fit(tr.iloc[:,1:-1], tr.iloc[:,-1])
```

```
SVC
SVC(gamma='auto', kernel='linear')
```

```
data.iloc[:,-1]
```

```
0      2
1      2
2      2
3      2
4      2
      ..
694    2
695    2
696    4
697    4
698    4
Name: Class, Length: 699, dtype: int64
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(te.iloc[:,-1], svc.predict(te.iloc[:,1:-1]))
```

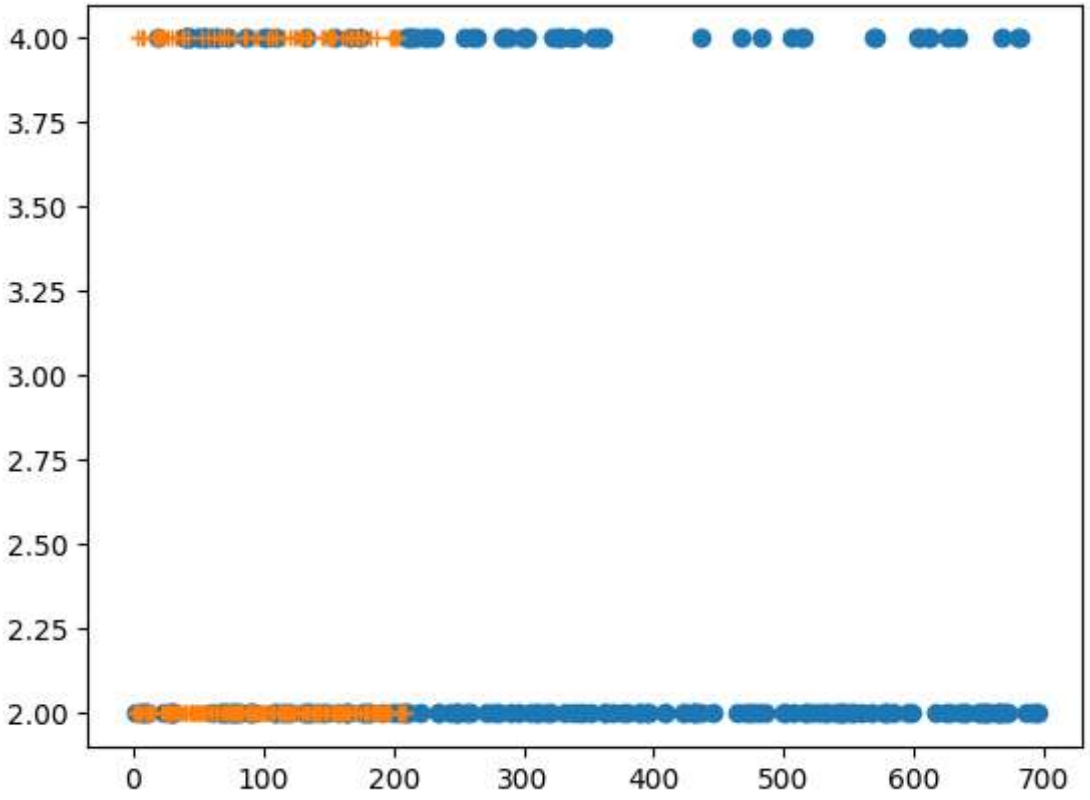
```
0.9666666666666667
```

```
import matplotlib.pyplot as plt
for col in data.iloc[:,1:-1].columns:
  # data[[col, 'Class']].plot(kind = 'scatter')
```

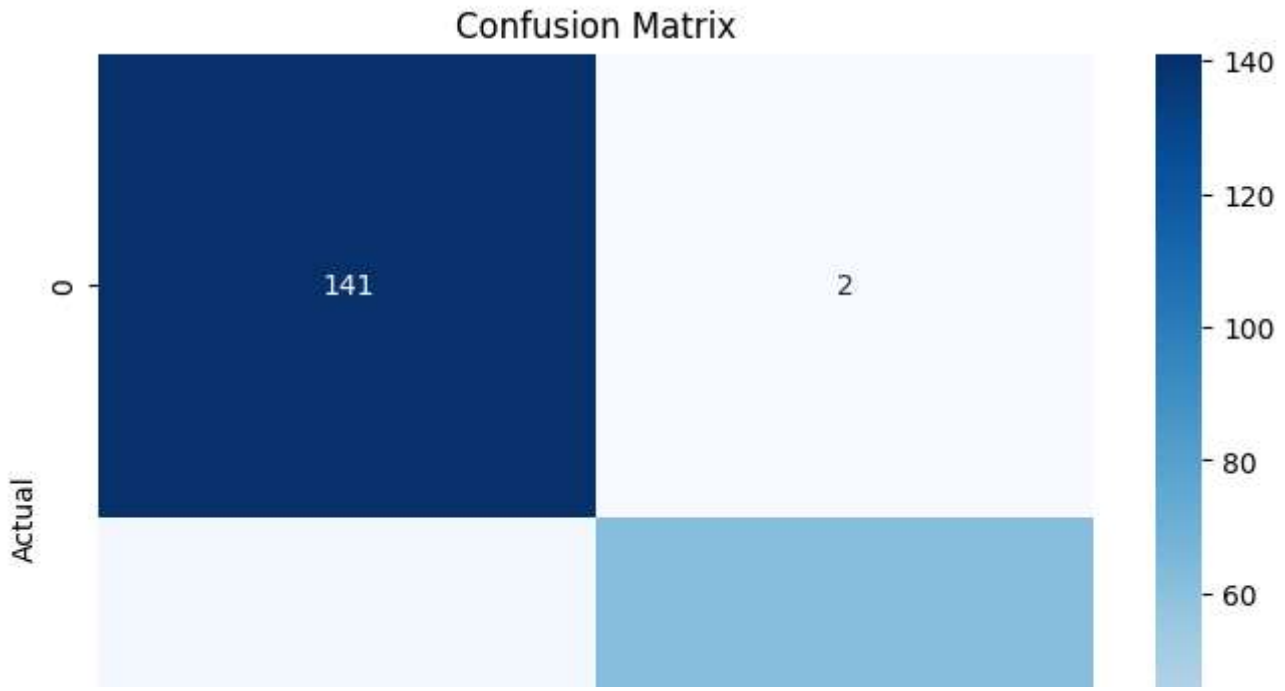
```
  plt.figure()
  plt.scatter(data['Class'], data[col])
  plt.show()
```

```
plt.plot(te.iloc[:,-1], 'o')
plt.plot(svc.predict(te.iloc[:,1:-1]), '+')
```

```
[<matplotlib.lines.Line2D at 0x7c64628684f0>]
```




```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(te.iloc[:,-1], svc.predict(te.iloc[:,1:-1]))
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

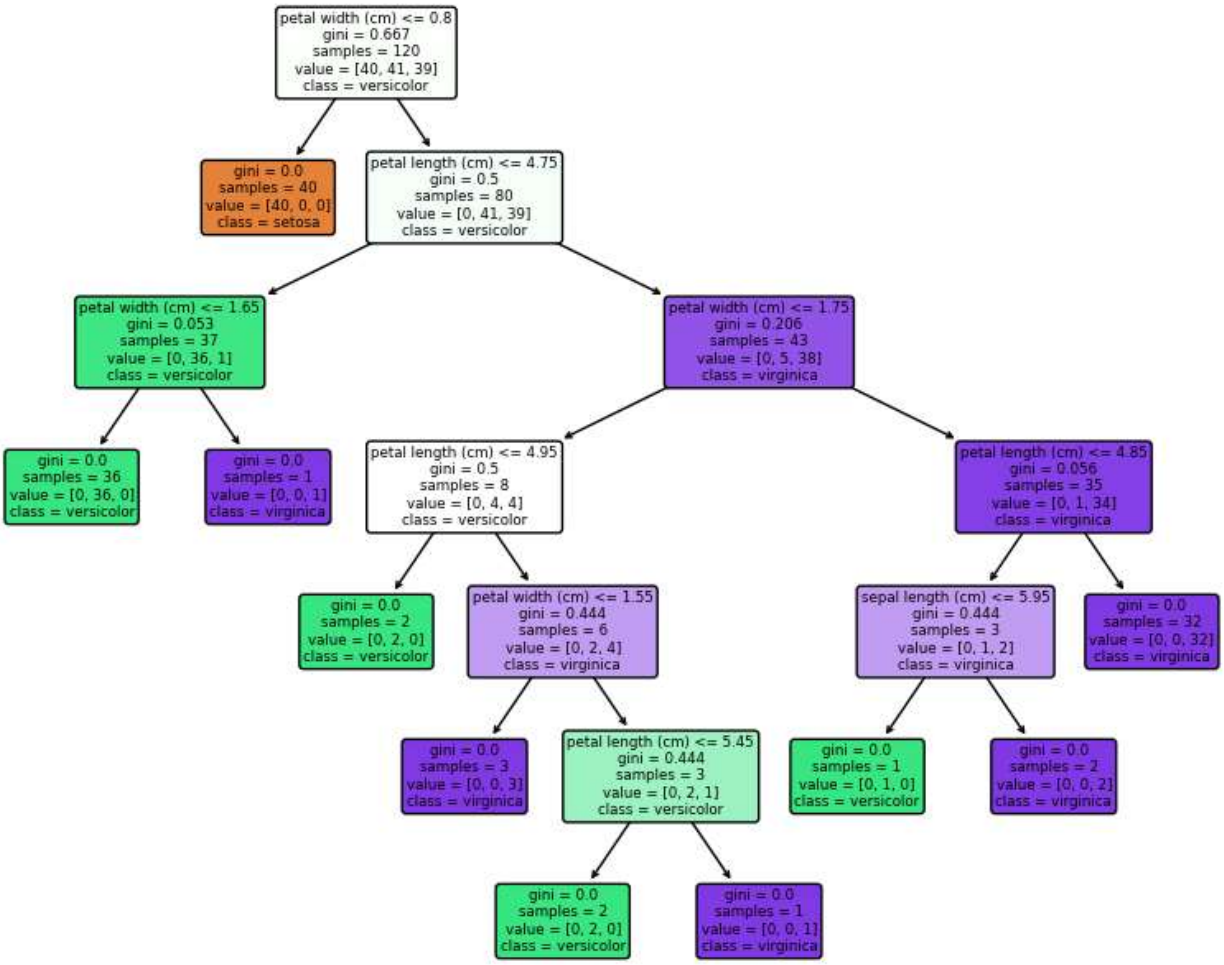


DECISION TREE

```
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np


# Load the Iris dataset (you can replace this with your own dataset)
data = load_iris()
X, y = data.data, data.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Decision Tree classifier
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Predict the labels for the test set
y_pred = clf.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}")
```

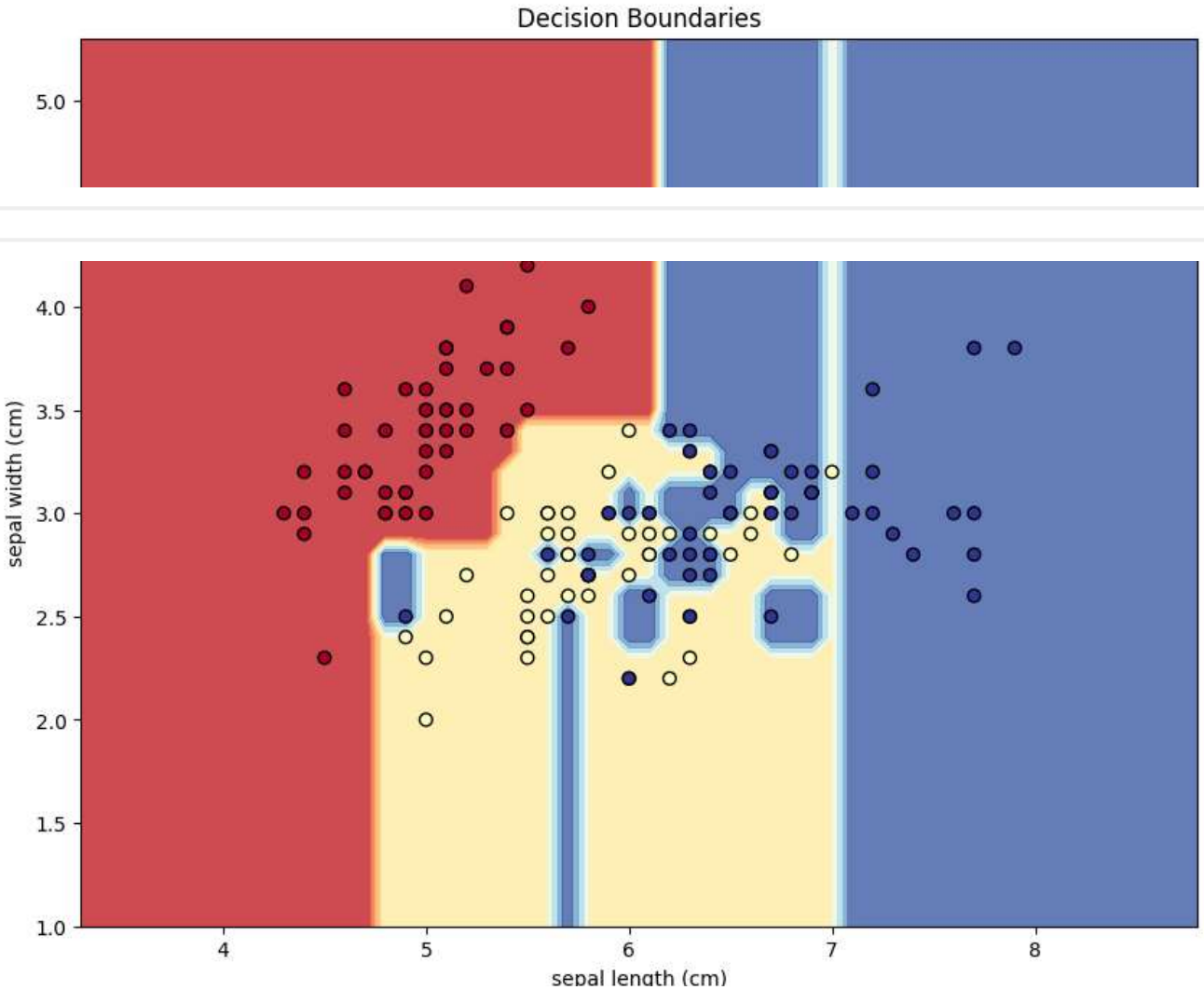
```
Accuracy: 1.0
```

```
plt.figure(figsize=(10, 8))
plot_tree(clf, filled=True, feature_names=data.feature_names, class_names=data.target_name
plt.show()
```




```
# Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
import numpy as np

# Load the Iris dataset
data = load_iris()
X, y = data.data[:, :2], data.target  # Using only the first two features for visualizatio

# Create a meshgrid for plotting decision boundaries
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
```

```
                     np.arange(y_min, y_max, 0.1))

# Create a Decision Tree classifier
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Predict the class for each point in the meshgrid
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundaries
plt.figure(figsize=(10, 8))
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.RdYlBu)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o', s=40, linewidth=1, cmap=plt

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
plt.title("Decision Boundaries")
plt.show()
```


Decision Boundaries
5.0
4.0
3.5
3.0
2.5
2.0
1.5
1.0
sepal width (cm)
4
5
6
7
8
sepal length (cm)

EXPERIMENT - 6

Implement Random Forest classification using any dataset.

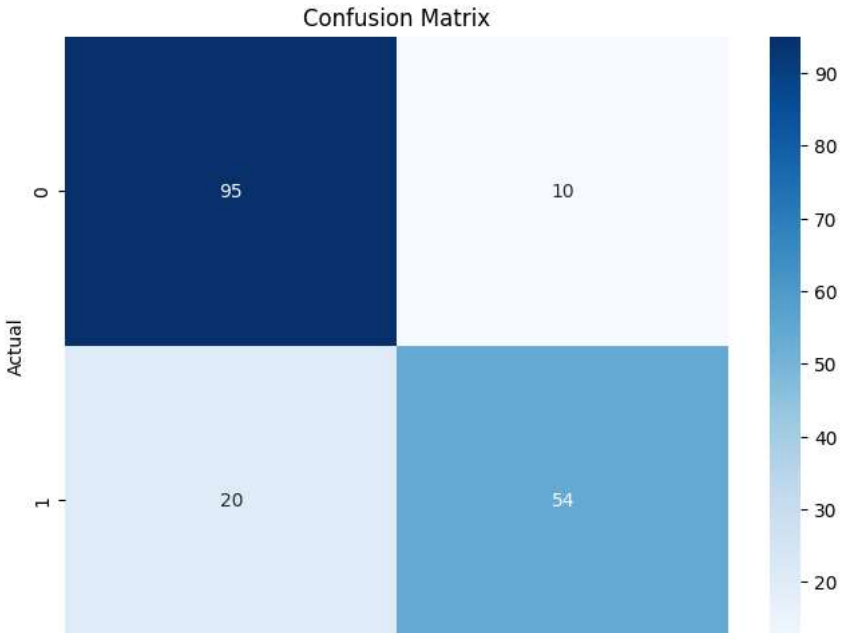
```
#importing libraries

from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
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
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv('train.csv')
data['Age'] = data['Age'].fillna(data['Age'].mean())
data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode())

label_encoder = preprocessing.LabelEncoder()
data['Sex'] = label_encoder.fit_transform(data['Sex'])
data['Embarked'] = label_encoder.fit_transform(data['Embarked'])

X = data.iloc[:, [2, 4, 5, 6, 7, 11]]
Y = data.iloc[:, 1]

# Performing feature scaling
sc = StandardScaler()
sc.fit(X)
X = sc.transform(X)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.2, random_state=42)
```

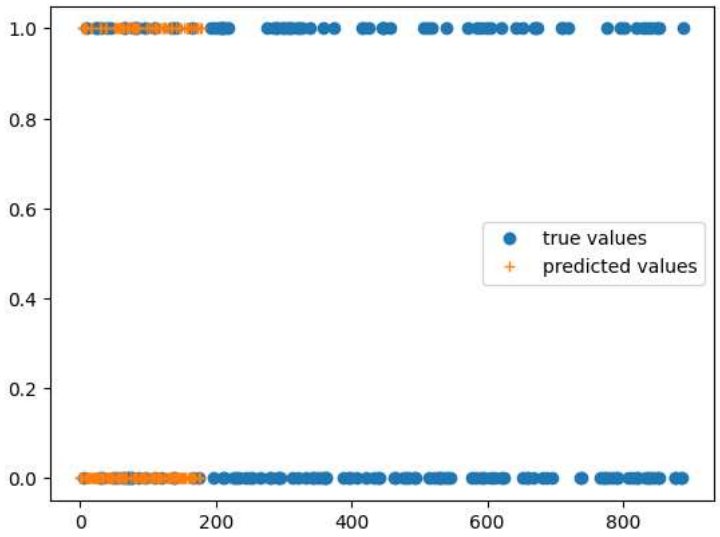
```
# Random Forest classifier
rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy*100)
```

```
Accuracy: 83.24022346368714
```

```
# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```




```
# Plotting the results
plt.plot(y_test, 'o', label="true values")
plt.plot(y_pred, '+', label="predicted values")
plt.legend()
plt.show()
```



EXPERIMENT - 7

Implement Naïve Bayes theorem to classify the English text.

```
import numpy as np
import pandas as pd


data = {
    "corpus": [
        "I love this movie, it's so entertaining!",
        "The weather is terrible today.",
        "This book is amazing, highly recommended.",
        "I don't like the taste of this dish.",
        "The performance was outstanding, I was truly impressed.",
        "I hate waiting in long lines.",
        "The food at the restaurant was delicious.",
        "The service was very slow and disappointing.",
        "The concert was fantastic, I had a great time.",
        "I'm not a fan of the new design.",
        "The team played poorly, it was a disappointing match.",
        "I adore the artwork in this museum.",
        "The customer support was helpful and friendly.",
        "I can't stand the noise in this neighborhood.",
        "The movie was boring, I fell asleep halfway through.",
        "The software is user-friendly and efficient.",
        "The traffic was unbearable, I was stuck for hours.",
        "This product is a waste of money.",
        "The atmosphere in the cafe was cozy and inviting.",
        "I'm impressed with the quality of this product.",
        "The hotel room was dirty and smelled bad.",
        "I enjoy spending time with my friends.",
        "The play was thought-provoking and well-executed.",
        "I couldn't stop laughing, the comedy show was hilarious.",
        "The experience was underwhelming, I expected more.",
    ],
    "label": [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0]
}
df = pd.DataFrame(data)



positive = df[df['label'] == 1]
negative = df[df['label'] == 0]

print("No. of Positive: ", len(positive))
print("No. of Negative: ", len(negative))
```

```
No. of Positive:  12
No. of Negative:  13
```

```
total = len(df)
prior_positive = len(positive) / total
prior_negative = len(negative) / total

print("Prior probability of Positive samples: ", prior_positive)
print("Prior probability of Negative samples: ", prior_negative)
```

```
Prior probability of Positive samples:  0.48
Prior probability of Negative samples:  0.52
```

```
positive_text = ' '.join(positive['corpus']).split()
negative_text = ' '.join(negative['corpus']).split()

vocabulary = list(set(negative_text + positive_text))
print('Vocabulary: \n', vocabulary)
```

```
Vocabulary:
 ['book', 'impressed', 'in', 'time', 'outstanding,', 'was', 'artwork', 'food', 'highly', 'impressed.', 'comedy', 'friendly.', 'amaz
```

```
positive_word_freq = { word: positive_text.count(word) for word in vocabulary }
negative_word_freq = { word: negative_text.count(word) for word in vocabulary }

word_data = {
    'Positive Word Frequency': positive_word_freq,
```

```
    'Negative Word Frequency': negative_word_freq,
}
word_df = pd.DataFrame(word_data)
word_df.head(10)
```

	Positive Word Frequency	Negative Word Frequency
book	1	0
impressed	1	0
in	2	2
time	1	0
outstanding,	1	0
was	7	8
artwork	1	0
food	1	0
highly	1	0
impressed.	1	0

```
test_data = {
    "corpus": [
        "This restaurant serves delicious food and has excellent service.",
        "I'm really disappointed with the customer service I received.",
        "The weather is perfect for outdoor activities today.",
        "I can't believe how fast and efficient the delivery service was.",
        "The movie was a complete waste of time and money, I regret watching it."
    ],
    "label": [1, 0, 1, 1, 0]
}
test_df = pd.DataFrame(test_data)
```

```
def predict(string, positive_word_freq, negative_word_freq, prior_positive, prior_negative, smoothing_factor = 1): # Using Laplace Smooth

    new_text_words = string.split()
    likelihood_positive = 1
    for word in new_text_words:
        likelihood_positive *= (positive_word_freq.get(word, 0) + smoothing_factor) / (len(positive_text) + smoothing_factor * len(vocabu

    likelihood_negative = 1
    for word in new_text_words:
        likelihood_negative *= (negative_word_freq.get(word, 0) + smoothing_factor) / (len(negative_text) + smoothing_factor * len(vocabu

    # Apply Naive Bayes formula
    posterior_positive = prior_positive * likelihood_positive
    posterior_negative = prior_negative * likelihood_negative

    # Classify based on the higher posterior probability
    predicted_class = 1 if posterior_positive > posterior_negative else 0
    return predicted_class
```

```
test_df['Predicted Label'] = [
    predict(
        string, positive_word_freq, negative_word_freq, prior_positive, prior_negative
    ) for string in test_df['corpus']
]
```

```
test_df.head()
```

	corpus	label	Predicted Label
0	This restaurant serves delicious food and has ...	1	1
1	I'm really disappointed with the customer serv...	0	1
2	The weather is perfect for outdoor activities ...	1	0
3	I can't believe how fast and efficient the del...	1	0
4	The movie was a complete waste of time and mon...	0	0

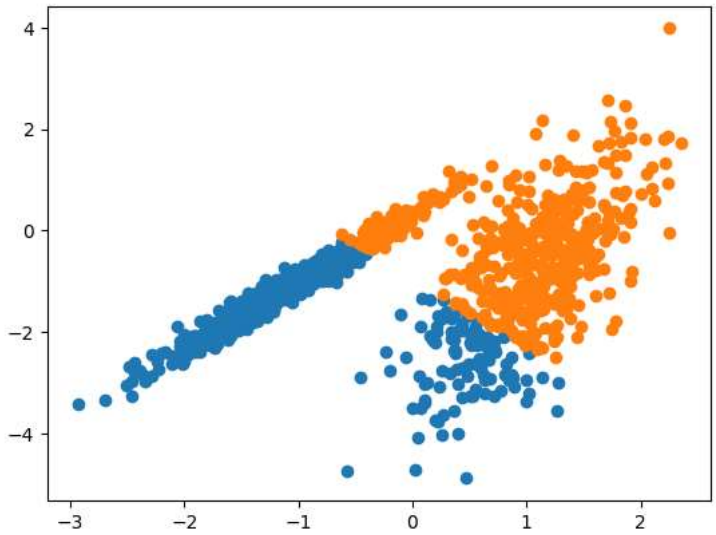
EXPERIMENT - 8

Implement clustering techniques (KMean, KMedoids).

K-MEANS

```
# k-means clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans
from matplotlib import pyplot
# define dataset
X, _ = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
# define the model
model = KMeans(n_clusters=2)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
  # get row indexes for samples with this cluster
  row_ix = where(yhat == cluster)
  # create scatter of these samples
  pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

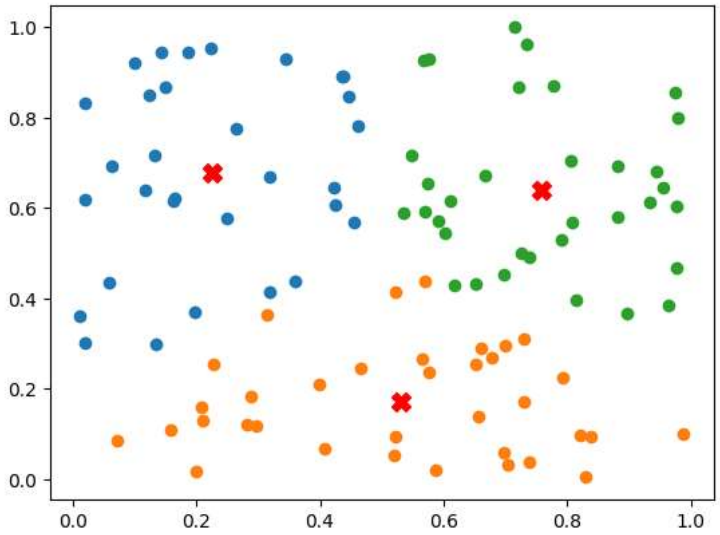
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
```




```
import numpy as np
from sklearn.cluster import KMeans
from matplotlib import pyplot

# Generate random data
np.random.seed(0)
X = np.random.rand(100, 2)

# Define the number of clusters (k)
k = 3

# Define the model
model = KMeans(n_clusters=k)

# Fit the model
model.fit(X)

# Assign a cluster to each example
```

```
yhat = model.predict(X)

# Retrieve unique clusters
clusters = np.unique(yhat)

# Create scatter plot for samples from each cluster
for cluster in clusters:
    # Get row indexes for samples with this cluster
    row_ix = np.where(yhat == cluster)
    # Create scatter of these samples
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])

# Get cluster centers
centers = model.cluster_centers_

# Mark cluster centers with 'X' symbols
pyplot.scatter(centers[:, 0], centers[:, 1], c='red', marker='X', s=100)

# Show the plot
pyplot.show()
```

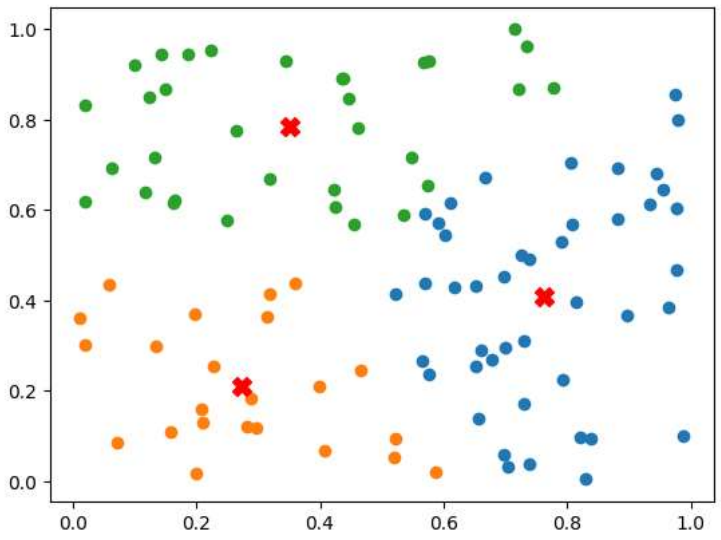
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
  warnings.warn(
```



K-MEDOIDS

```
import numpy as np
from scipy.spatial.distance import cdist
from scipy.spatial import distance_matrix
from scipy.cluster.vq import kmeans, vq
from matplotlib import pyplot

def kmedoids(X, k, max_iters=100):
    # Initialize medoids
    medoids = X[np.random.choice(X.shape[0], k, replace=False)]
    for _ in range(max_iters):
        # Calculate distance matrix
        D = distance_matrix(X, medoids)
        # Assign each point to the closest medoid
        labels = np.argmin(D, axis=1)
        # Update medoids
        new_medoids = np.array([X[labels == i].mean(axis=0) for i in range(k)])
        if np.all(new_medoids == medoids):
            break
        medoids = new_medoids
    return labels, medoids

# Generate random data
np.random.seed(0)
X = np.random.rand(100, 2)

# Perform K-medoids clustering
k = 3
labels, medoids = kmedoids(X, k)

# Create scatter plot for samples from each cluster
for i in range(k):
```

```
    row_ix = np.where(labels == i)
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])

# Show the plot
pyplot.scatter(medoids[:, 0], medoids[:, 1], c='red', marker='X', s=100)  # Mark medoids with red 'X'
pyplot.show()
```



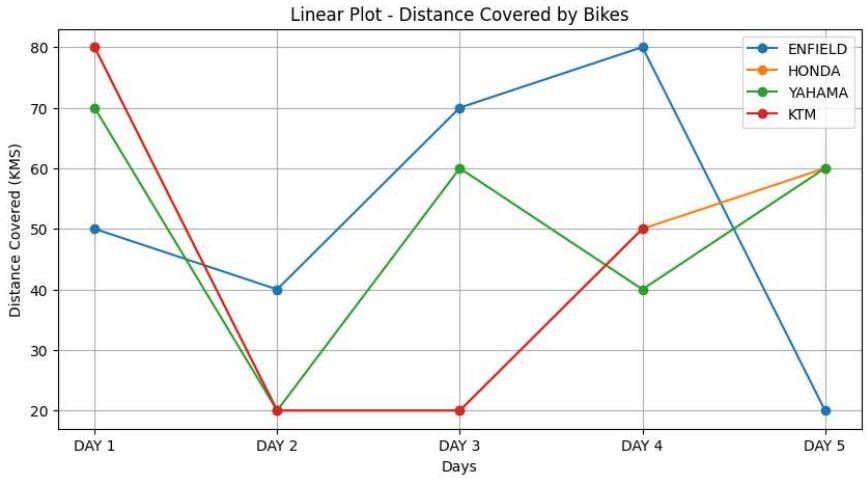
EXPERIMENT - 9

Visualize data using any plotting framework.

```
import pandas as pd
import matplotlib.pyplot as plt

# Create a DataFrame with the given data
data = {
    "DAYS": ["DAY 1", "DAY 2", "DAY 3", "DAY 4", "DAY 5"],
    "ENFIELD": [50, 40, 70, 80, 20],
    "HONDA": [80, 20, 20, 50, 60],
    "YAHAMA": [70, 20, 60, 40, 60],
    "KTM": [80, 20, 20, 50, None],  # Note: Missing value for DAY 5
}

df = pd.DataFrame(data)
```

LINEAR PLOT

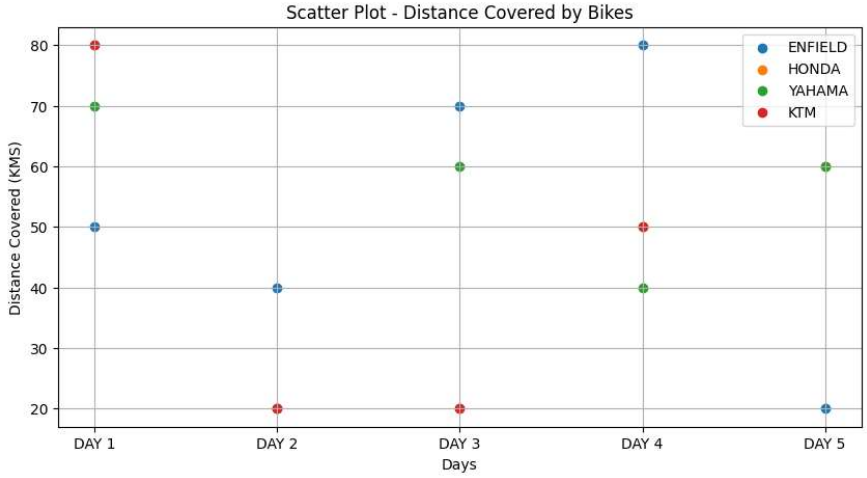
```
# Linear Plot
plt.figure(figsize=(10, 5))
plt.plot(df['DAYS'], df['ENFIELD'], label='ENFIELD', marker='o')
plt.plot(df['DAYS'], df['HONDA'], label='HONDA', marker='o')
plt.plot(df['DAYS'], df['YAHAMA'], label='YAHAMA', marker='o')
plt.plot(df['DAYS'], df['KTM'], label='KTM', marker='o')
plt.xlabel('Days')
plt.ylabel('Distance Covered (KMS)')
plt.title('Linear Plot - Distance Covered by Bikes')
plt.legend()
plt.grid(True)
plt.show()
```



SCATTER PLOT

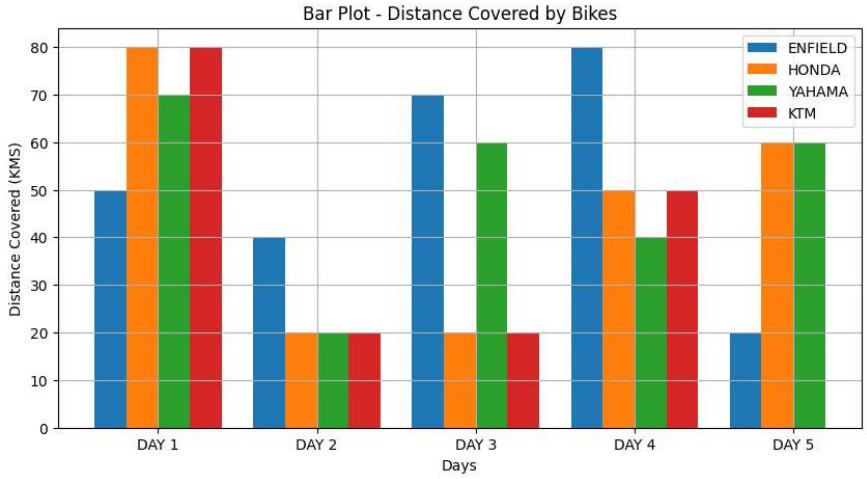
```
# Scatter Plot
plt.figure(figsize=(10, 5))
plt.scatter(df['DAYS'], df['ENFIELD'], label='ENFIELD', marker='o')
plt.scatter(df['DAYS'], df['HONDA'], label='HONDA', marker='o')
plt.scatter(df['DAYS'], df['YAHAMA'], label='YAHAMA', marker='o')
plt.scatter(df['DAYS'], df['KTM'], label='KTM', marker='o')
plt.xlabel('Days')
```

```
plt.ylabel('Distance Covered (KMS)')
plt.title('Scatter Plot - Distance Covered by Bikes')
plt.legend()
plt.grid(True)
plt.show()
```



BAR PLOT

```
# Bar Plot
plt.figure(figsize=(10, 5))
width = 0.2
x = range(len(df['DAYS']))
plt.bar(x, df['ENFIELD'], width=width, label='ENFIELD')
plt.bar([i + width for i in x], df['HONDA'], width=width, label='HONDA')
plt.bar([i + 2 * width for i in x], df['YAHAMA'], width=width, label='YAHAMA')
plt.bar([i + 3 * width for i in x], df['KTM'], width=width, label='KTM')
plt.xlabel('Days')
plt.ylabel('Distance Covered (KMS)')
plt.title('Bar Plot - Distance Covered by Bikes')
plt.xticks([i + 1.5 * width for i in x], df['DAYS'])
plt.legend()
plt.grid(True)
plt.show()
```



EXPERIMENT - 10

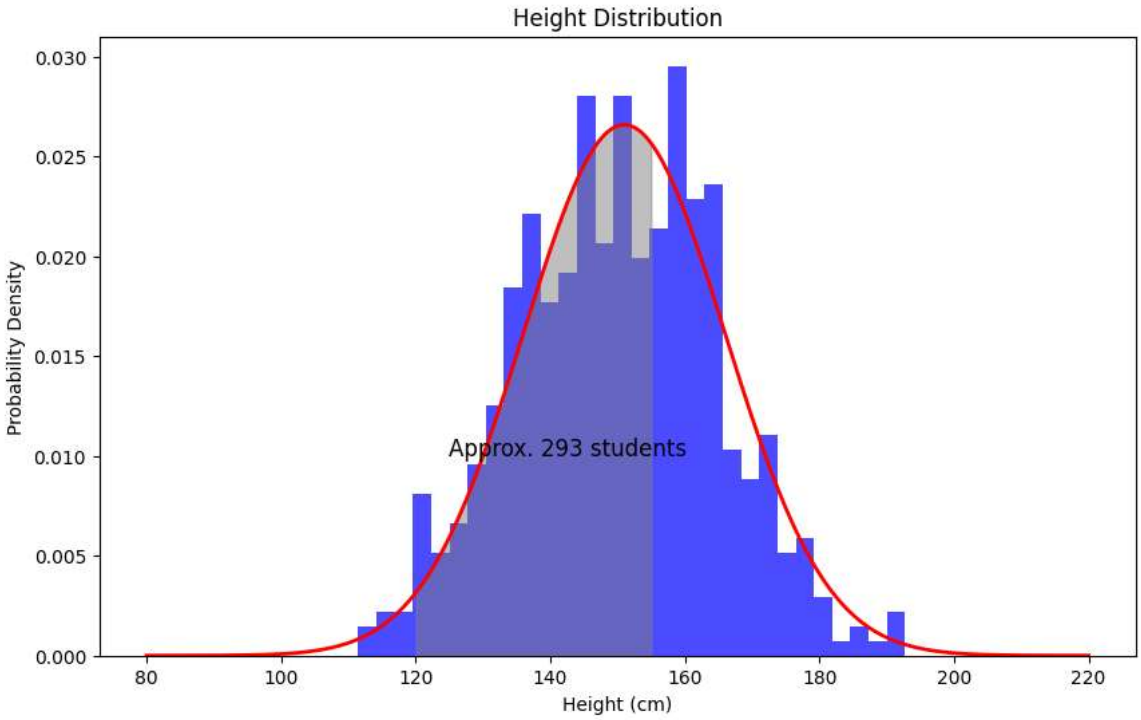
Solve a numerical problem on Normal Distribution using python.

```
# import required libraries
from scipy.stats import norm
import numpy as np


# Given information
mean = 78
std_dev = 25
total_students = 100
score = 70


# Calculate z-score for 70
z_score = (score - mean) / std_dev


# Calculate the probability of getting a more than 70
prob = norm.cdf(z_score)
prob1 = norm.cdf(abs(z_score))


# Calculate the percentage of students who got more than 70 marks
percent = (1-prob) * 100


# Print the result
print(percent)
```

```
62.551583472332005
```

-> The mean height of 500 students is 151 cm and SD is 15 cm. Assuming that the heights are normally distributed , find how many students height lie between 120 cm and 150 cm .

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats

# Given data
mean_height = 151
std_deviation = 15
total_students = 500

# Generate a range of heights
x = np.linspace(80, 220, 1000)

# Calculate the probability density function (PDF)
pdf = stats.norm.pdf(x, mean_height, std_deviation)

# Calculate the z-scores
z_120 = (120 - mean_height) / std_deviation
z_155 = (155 - mean_height) / std_deviation

# Find the probabilities using the cumulative distribution function (CDF)
probability_120 = stats.norm.cdf(z_120)
probability_155 = stats.norm.cdf(z_155)

# Calculate the final probability
final_probability = probability_155 - probability_120

# Calculate the number of students
num_students_in_range = round(final_probability * total_students)



# Create the histogram
plt.figure(figsize=(10, 6))
plt.hist(np.random.normal(mean_height, std_deviation, total_students), bins=30, density=True, alpha=0.7, color='blue')

# Plot the probability density function
plt.plot(x, pdf, color='red', lw=2)

# Highlight the range of interest
plt.fill_between(x, 0, pdf, where=(x >= 120) & (x <= 155), color='gray', alpha=0.5)

# Add labels and title
plt.xlabel('Height (cm)')
plt.ylabel('Probability Density')
plt.title('Height Distribution')

# Add a text annotation
plt.text(125, 0.01, f'Approx. {num_students_in_range} students', fontsize=12, color='black')

# Show the plot
plt.show()
```


Height Distribution
Probability Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Approx. 293 students
80
100
120
140
160
180
200
220
Height (cm)