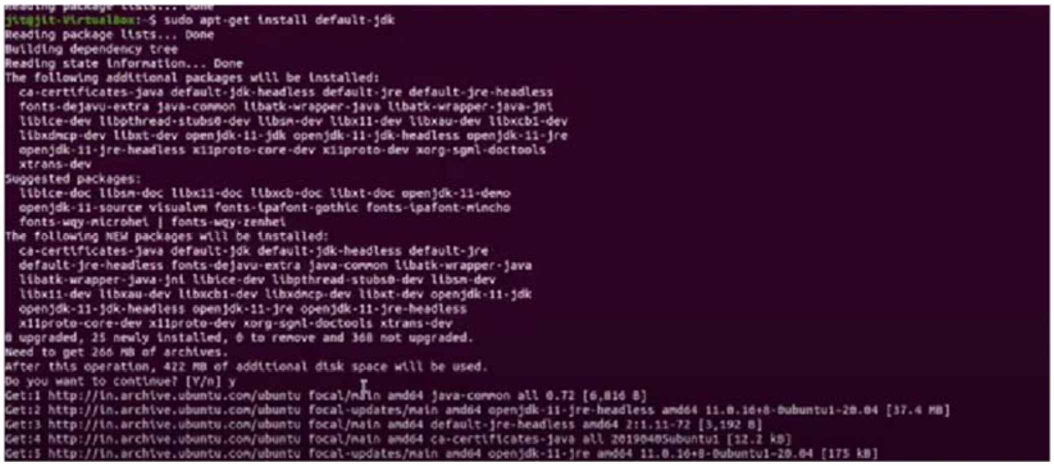
Experiment 1

Aim: Install, configure, and run Hadoop and HDFS.

Steps:

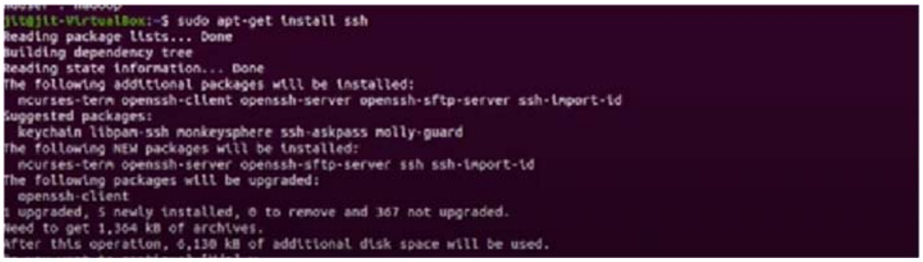
1. Create a user for Hadoop Environment.

2. Installing Java.



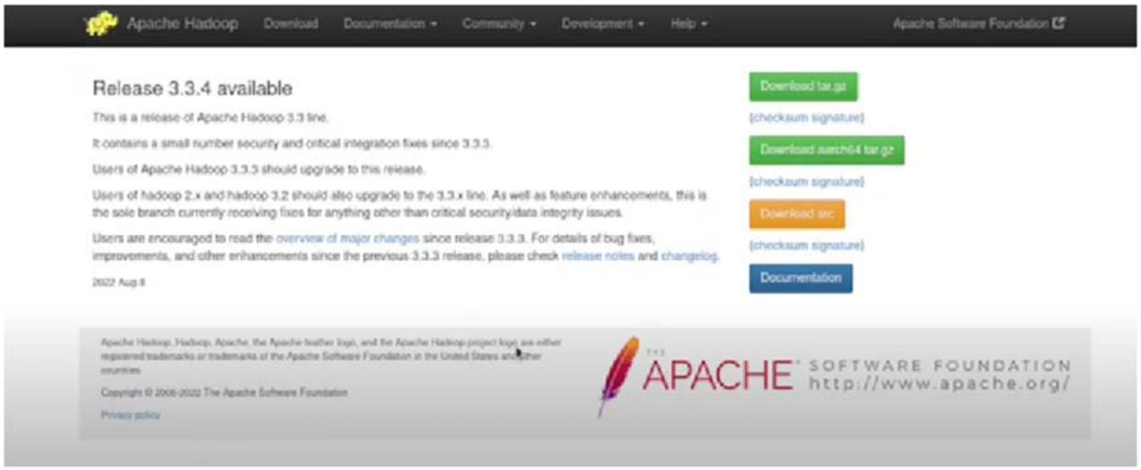
```
Reading package lists... Done
jit@jit-VirtualBox:~$ sudo apt-get install default-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jdk-headless default-jre default-jre-headless
  fonts-dejavu-extra java-common libatk-wrapper-java libatk-wrapper-java-jni
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev
  libxdmcp-dev libxt-dev openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
  openjdk-11-jre-headless x11proto-core-dev x11proto-dev xorg-sgml-doctools
  xtrans-dev
Suggested packages:
  libice-doc libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-11-demo
  openjdk-11-source visualvm fonts-ipafont-gothic fonts-ipafont-mincho
  fonts-wqy-microhei | fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java default-jdk default-jdk-headless default-jre
  default-jre-headless fonts-dejavu-extra java-common libatk-wrapper-java
  libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev
  libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libxt-dev openjdk-11-jdk
  openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless
  x11proto-core-dev x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 25 newly installed, 0 to remove and 368 not upgraded.
Need to get 266 MB of archives.
After this operation, 422 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/main amd64 java-common all 0.72 [6,816 B]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openjdk-11-jre-headless amd64 11.0.16+8-0ubuntu1~20.04 [37.4 MB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal/main amd64 default-jre-headless amd64 2:1.11-72 [3,192 B]
Get:4 http://in.archive.ubuntu.com/ubuntu focal/main amd64 ca-certificates-java all 20190405ubuntu1 [12.2 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 openjdk-11-jre amd64 11.0.16+8-0ubuntu1~20.04 [175 kB]
```

3. Install OpenSSH on Ubuntu.



```
jit@jit-VirtualBox:~$ sudo apt-get install ssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ncurses-term openssh-client openssh-server openssh-sftp-server ssh-import-id
Suggested packages:
  keychain libpam-ssh monkeysphere ssh-askpass molly-guard
The following NEW packages will be installed:
  ncurses-term openssh-server openssh-sftp-server ssh ssh-import-id
The following packages will be upgraded:
  openssh-client
1 upgraded, 5 newly installed, 0 to remove and 367 not upgraded.
Need to get 1,364 kB of archives.
After this operation, 6,130 kB of additional disk space will be used.
```

4. Install Apache Hadoop.



Release 3.3.4 available

This is a release of Apache Hadoop 3.3 line.

It contains a small number security and critical integration fixes since 3.3.3.

Users of Apache Hadoop 3.3.3 should upgrade to this release.

Users of hadoop 2.x and hadoop 3.2 should also upgrade to the 3.3.x line. As well as feature enhancements, this is the sole branch currently receiving fixes for anything other than critical security/data integrity issues.

Users are encouraged to read the overview of major changes since release 3.3.3. For details of bug fixes, improvements, and other enhancements since the previous 3.3.3 release, please check release notes and changelog.

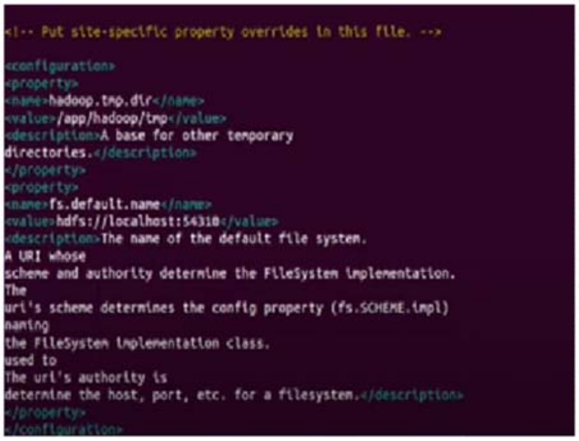
2022 Aug 8

Apache Hadoop, Hadoop, Apache, the Apache feather logo, and the Apache Hadoop project logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and other countries.

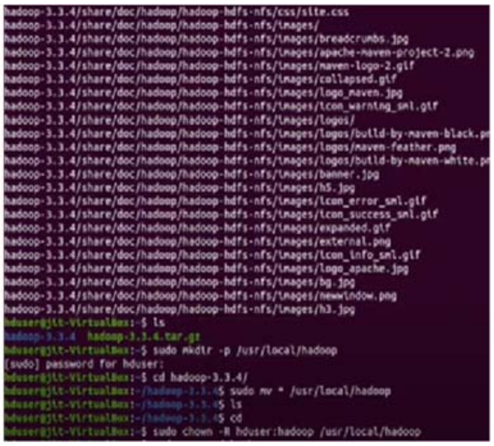
Copyright © 2006-2022 The Apache Software Foundation

Privacy policy

5. Configure Hadoop.



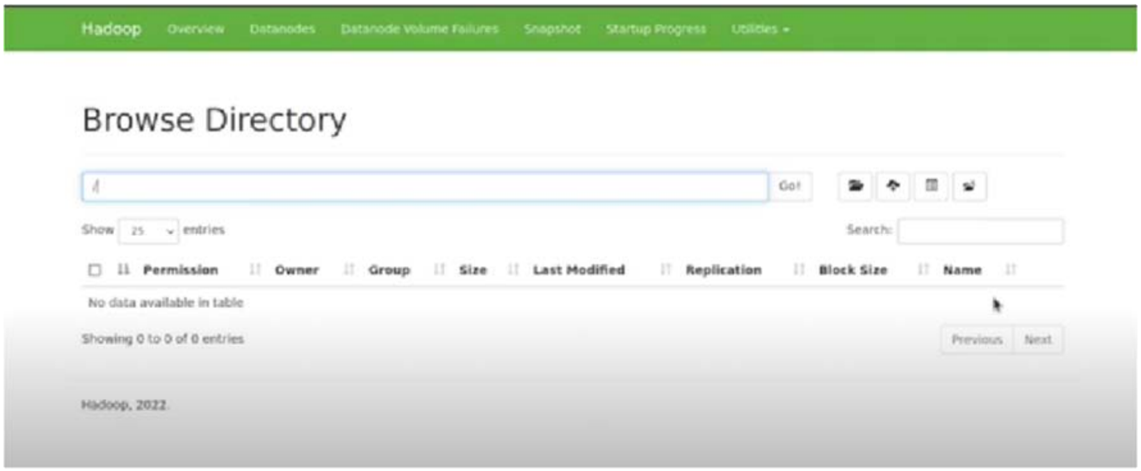
```
<!-- Put site-specific property overrides in this file. -->

<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.p
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.p
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
```

6. Start Hadoop Cluster.

7. Access Hadoop UI from Browser.



Browse Directory

/ Go!

Show 25 entries Search:

□ Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
No data available in table							

Showing 0 to 0 of 0 entries

Hadoop, 2022.

Experiment 2

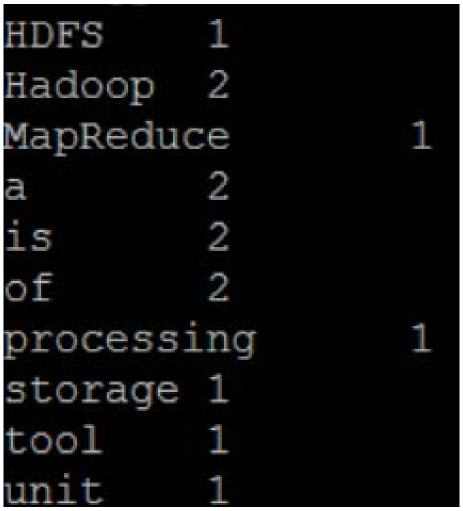
Aim: Implement word count/frequency program in MapReduce.

Code:

```
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 void
reduce(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,new
Path(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
```

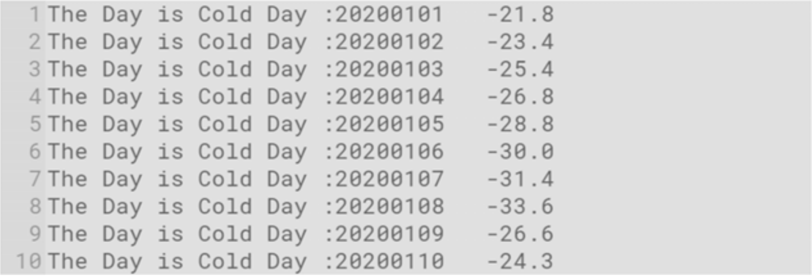
Experiment 3

Aim: Implement a MapReduce program that processes a weather dataset.

Code:

```
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;

public class MyMaxMin {

//Mapper

     public static class MaxTemperatureMapper extends
               Mapper<LongWritable, Text, Text, Text> {
     public static final int MISSING = 9999;

     @Override
          public void map(LongWritable arg0, Text Value,
Context context)
                    throws IOException, InterruptedException {

          String line = Value.toString();

               if (!(line.length() == 0)) {
                    String date = line.substring(6, 14);
                    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) {
                         context.write(new Text("The Day is
Hot Day :" + date),
                                                  new
Text(String.valueOf(temp_Max)));
                    }
                    if (temp_Min < 15) {
                         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));
          }

     }

//Main

     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:



```
The Day is Cold Day :20200101   -21.8
The Day is Cold Day :20200102   -23.4
The Day is Cold Day :20200103   -25.4
The Day is Cold Day :20200104   -26.8
The Day is Cold Day :20200105   -28.8
The Day is Cold Day :20200106   -30.0
The Day is Cold Day :20200107   -31.4
The Day is Cold Day :20200108   -33.6
The Day is Cold Day :20200109   -26.6
The Day is Cold Day :20200110   -24.3
```

Experiment 4

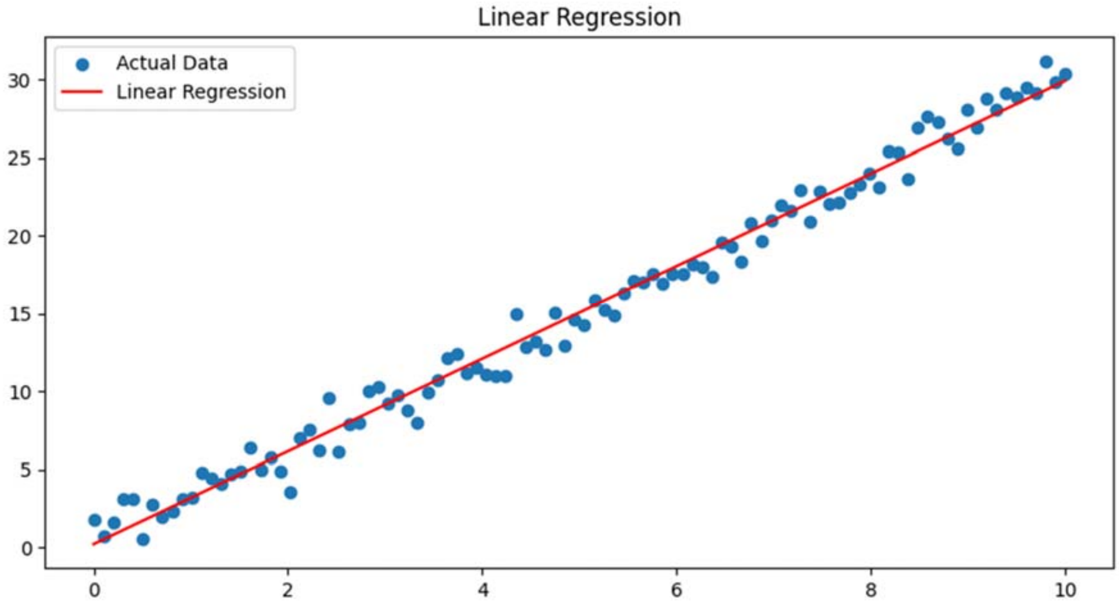
Aim: Implement Linear and Logistic Regression.

Linear Regression

Code:

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

# 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()
```

Output:



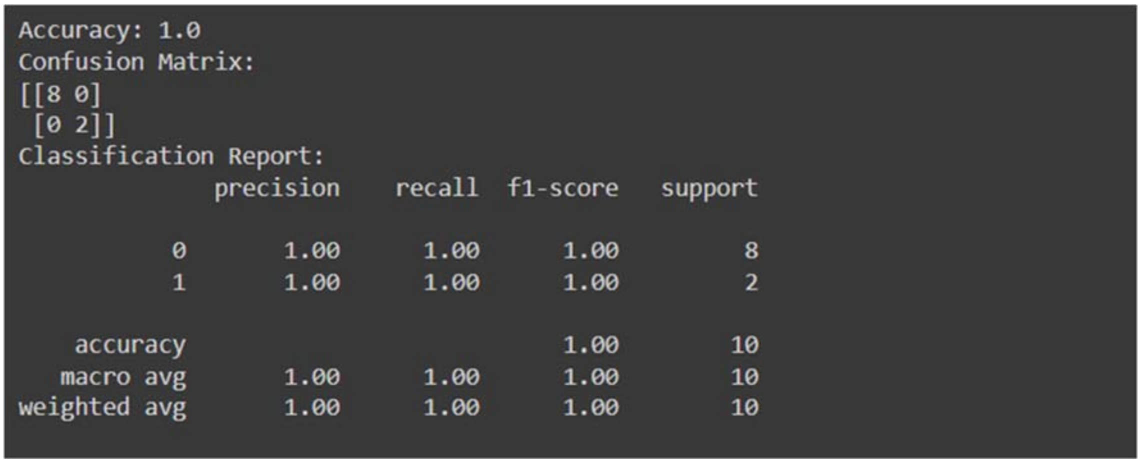
Logistic Regression

Code:

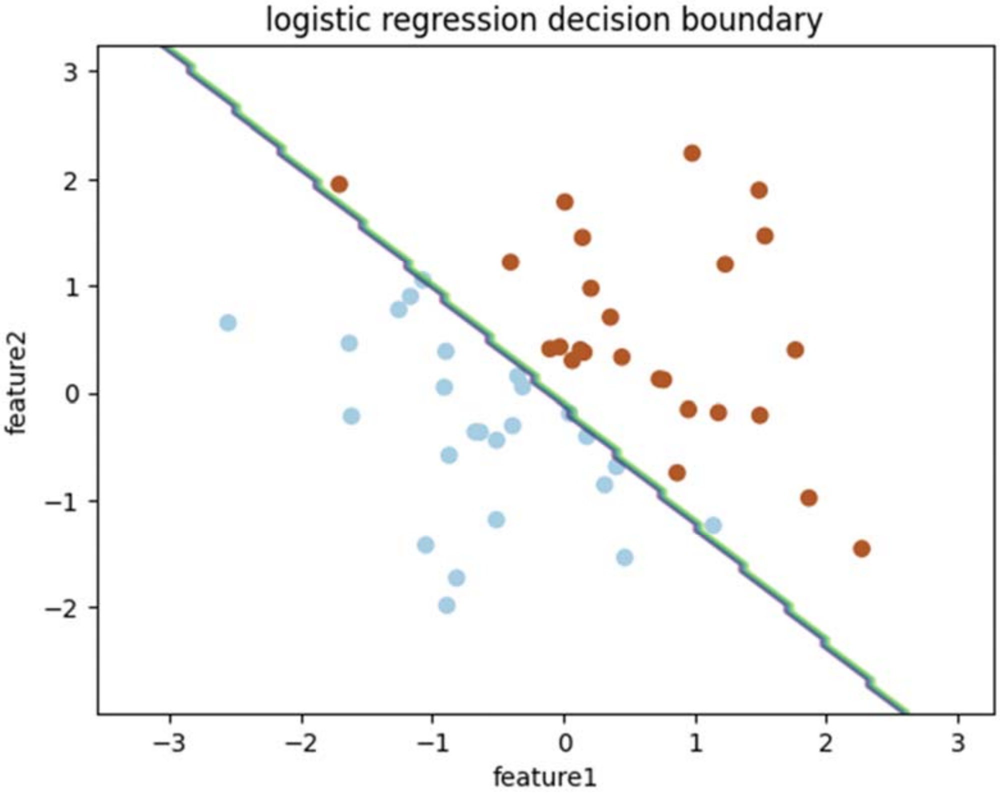
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

# Generate some sample data (replace this with your own
dataset)
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}")

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()
```

Output:



```
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
```


logistic regression decision boundary
feature2
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
feature1

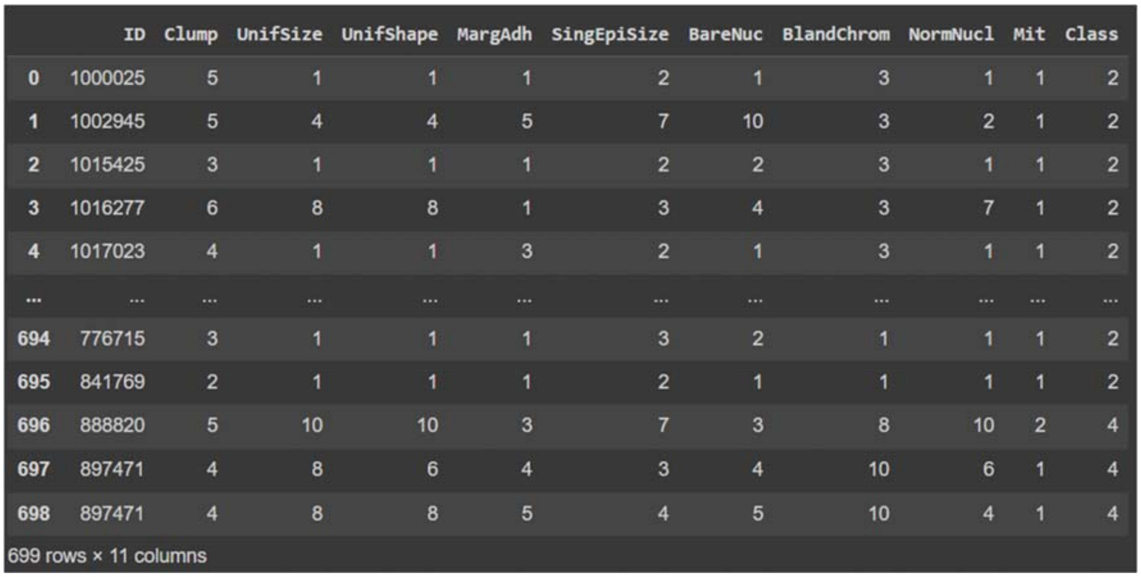
Experiment 5

Aim: Implement SVM/Decision tree classification techniques.

SVM

Code:

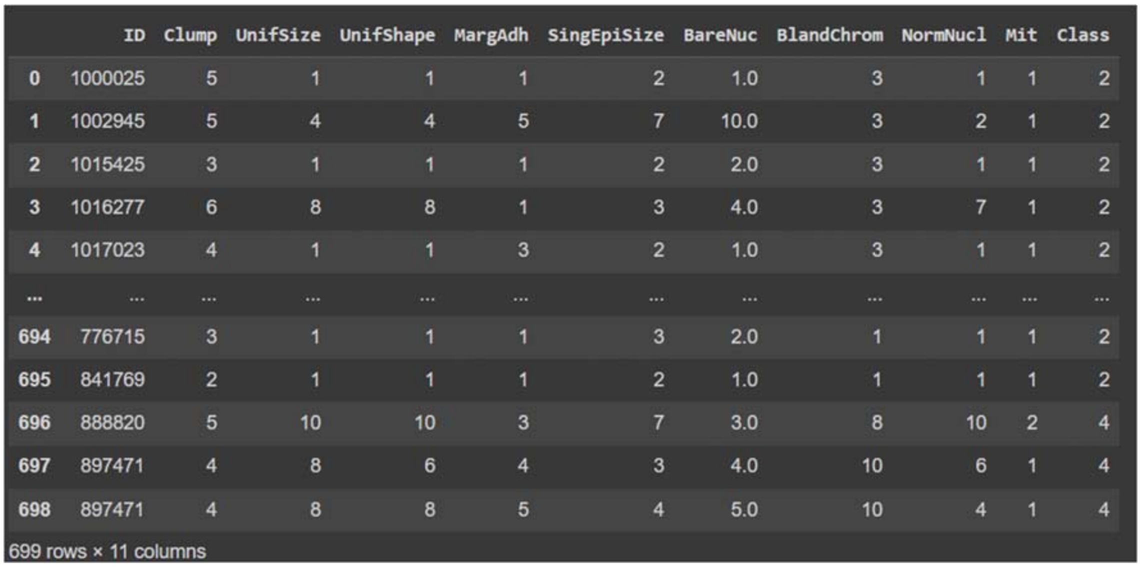
```
import numpy as np
import pandas as pd
from sklearn.svm import SVC
data = pd.read_csv('/content/cell_samples (1).csv')
data
```



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

699 rows × 11 columns

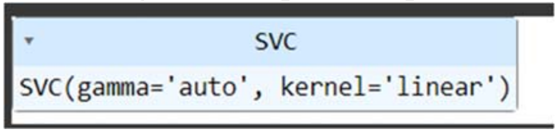
```
data = data.apply(pd.to_numeric, errors='coerce')
data = data.fillna(data.mean())
data
```



	ID	Clump	UnifSize	UnifShape	MargAdh	SingEpiSize	BareNuc	BlandChrom	NormNucl	Mit	Class
0	1000025	5	1	1	1	2	1.0	3	1	1	2
1	1002945	5	4	4	5	7	10.0	3	2	1	2
2	1015425	3	1	1	1	2	2.0	3	1	1	2
3	1016277	6	8	8	1	3	4.0	3	7	1	2
4	1017023	4	1	1	3	2	1.0	3	1	1	2
...	...	...	...	...	...	...	...	...	...	...	...
694	776715	3	1	1	1	3	2.0	1	1	1	2
695	841769	2	1	1	1	2	1.0	1	1	1	2
696	888820	5	10	10	3	7	3.0	8	10	2	4
697	897471	4	8	6	4	3	4.0	10	6	1	4
698	897471	4	8	8	5	4	5.0	10	4	1	4

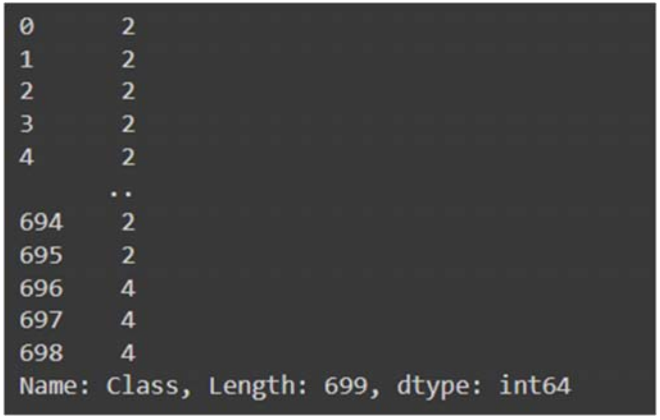
699 rows × 11 columns

```
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])
```



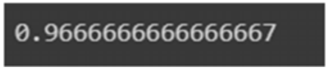
```
SVC
SVC(gamma='auto', kernel='linear')
```

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



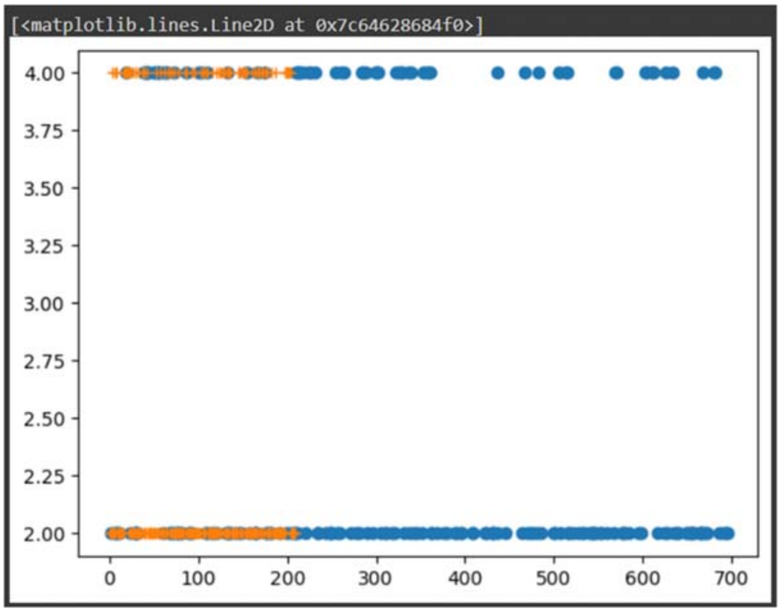
```
0      2
1      2
2      2
3      2
4      2
      ..
694    2
695    2
696    4
697    4
698    4
Name: Class, Length: 699, dtype: int64
```

```
from sklearn.metrics import accuracy_score
accuracy_score(te.iloc[:,-1], svc.predict(te.iloc[:,1:-1]))
```



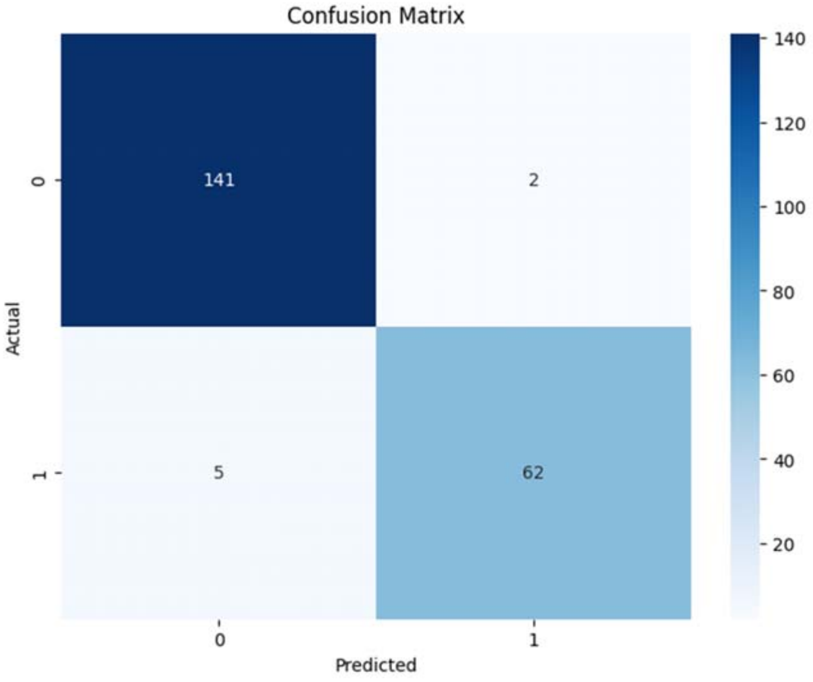
```
0.9666666666666667
```

```
import matplotlib.pyplot as plt
for col in data.iloc[:,1:-1].columns:
  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()
```

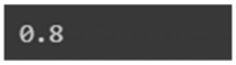


Decision Tree

Code:

```
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Generate a synthetic dataset
X, y = datasets.make_classification(n_samples=100,
n_features=4, n_informative=2, n_redundant=0, random_state=0)

# Split the dataset 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)

# Decision Tree Classifier
tree_clf = DecisionTreeClassifier(max_depth=2,
random_state=42)
tree_clf.fit(X_train, y_train)
tree_predictions = tree_clf.predict(X_test)
tree_accuracy = accuracy_score(y_test, tree_predictions)

tree_accuracy
```

Output:



```
0.8
```

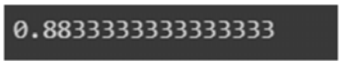
Experiment 6

Aim: Implement Random Forest classification using any dataset.

Code:

```
from sklearn.ensemble import RandomForestClassifier

# Generate a synthetic dataset
X, y = datasets.make_classification(n_samples=1000,
n_features=5, n_informative=3, n_redundant=0,
n_clusters_per_class=1, random_state=42)

# Split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, max_depth=3,
random_state=42)
rf_clf.fit(X_train, y_train)
rf_predictions = rf_clf.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)

rf_accuracy
```

Output:



```
0.8833333333333333
```

Experiment 7

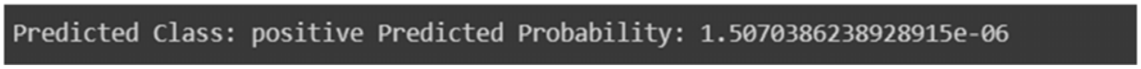
Aim: Implement Naïve Bayes theorem to classify the English text.

Code:

```
import numpy as np
import pandas as pd

# Sample training data
data = {
    'text': [
        "I love this product",
        "This is great",
        "Not good at all",
        "Awful experience",
        "Excellent quality"
    ],
    'label': ['positive', 'positive', 'negative', 'negative',
'positive']
}

# Create a pandas DataFrame
df = pd.DataFrame(data)

# Separate data by class labels
positive_data = df[df['label'] == 'positive']
negative_data = df[df['label'] == 'negative']

# Combine all text data for each class
positive_text = ' '.join(positive_data['text']).split()
negative_text = ' '.join(negative_data['text']).split()

# Calculate prior probabilities
total_documents = len(df)
prior_positive = len(positive_data) / total_documents
prior_negative = len(negative_data) / total_documents

# Create vocabulary
vocabulary = list(set(positive_text + negative_text))

# Calculate word frequencies in each class
positive_word_freq = {word: positive_text.count(word) for word
in vocabulary}
negative_word_freq = {word: negative_text.count(word) for word
in vocabulary}

# Classify new text
new_text = "This is a good product"
new_text_words = new_text.split()
```

```
# Calculate likelihoods using Laplace smoothing
smoothing_factor = 1

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(vocabulary))

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(vocabulary))

# 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 = 'positive' if posterior_positive >
posterior_negative else 'negative'

print("Predicted Class:", predicted_class, "Predicted
Probability:", (posterior_positive)/(posterior_positive +
prior_negative))
```

Output:



```
Predicted Class: positive Predicted Probability: 1.5070386238928915e-06
```

Experiment 8

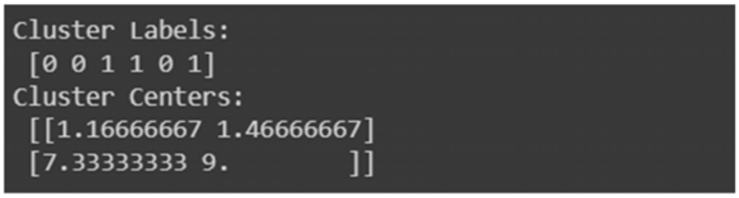
Aim: Implement clustering techniques (KMeans, KMedoids).

KMeans

Code:

```
import numpy as np

def kmeans(data, k, max_iterations=100):
    # Initialize centroids randomly
    centroids = data[np.random.choice(len(data), k,
replace=False)]

    for _ in range(max_iterations):
        # Assign each data point to the nearest centroid
        distances = np.linalg.norm(data[:, np.newaxis] -
centroids, axis=2)
        labels = np.argmin(distances, axis=1)

        # Update centroids
        new_centroids = np.array([data[labels ==
i].mean(axis=0) for i in range(k)])

        # Check for convergence
        if np.all(new_centroids == centroids):
            break

        centroids = new_centroids

    return labels, centroids

# Create a sample dataset (replace this with your data)
data = np.array([[1, 2],
                 [1.5, 1.8],
                 [5, 8],
                 [8, 8],
                 [1, 0.6],
                 [9, 11]])

# Perform K-Means clustering with 2 clusters
k = 2
cluster_labels, cluster_centers = kmeans(data, k)
print("Cluster Labels:\n", cluster_labels)
print("Cluster Centers:\n", cluster_centers)
```

Output:



```
Cluster Labels:
 [0 0 1 1 0 1]
Cluster Centers:
 [[1.16666667 1.46666667]
 [7.33333333 9.        ]]
```

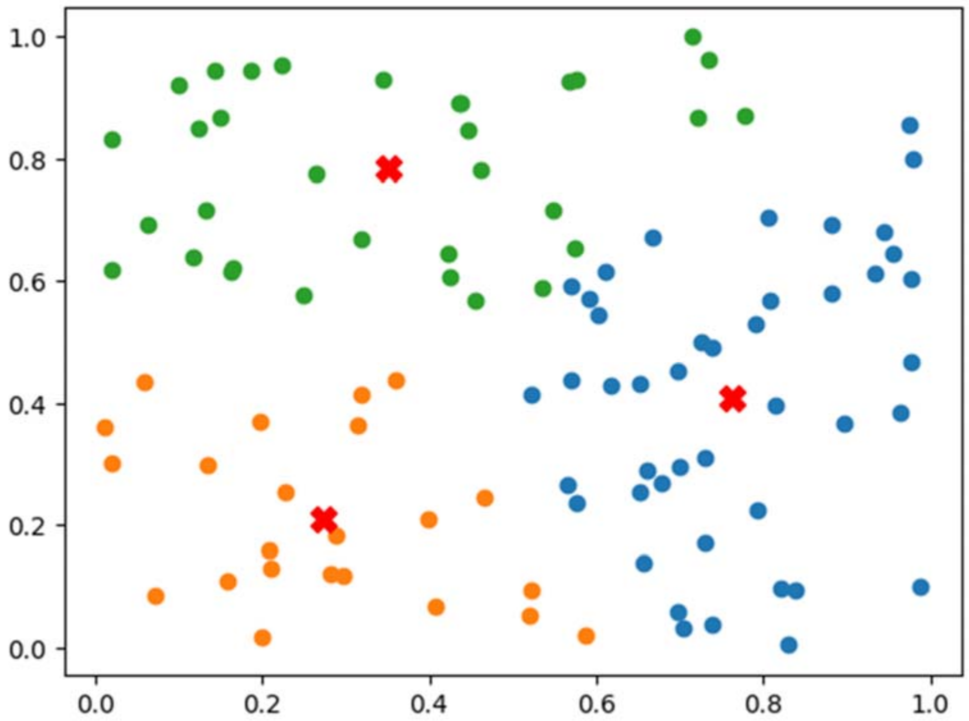
KMedoids

Code:

```
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()
```

Output:



Experiment 9

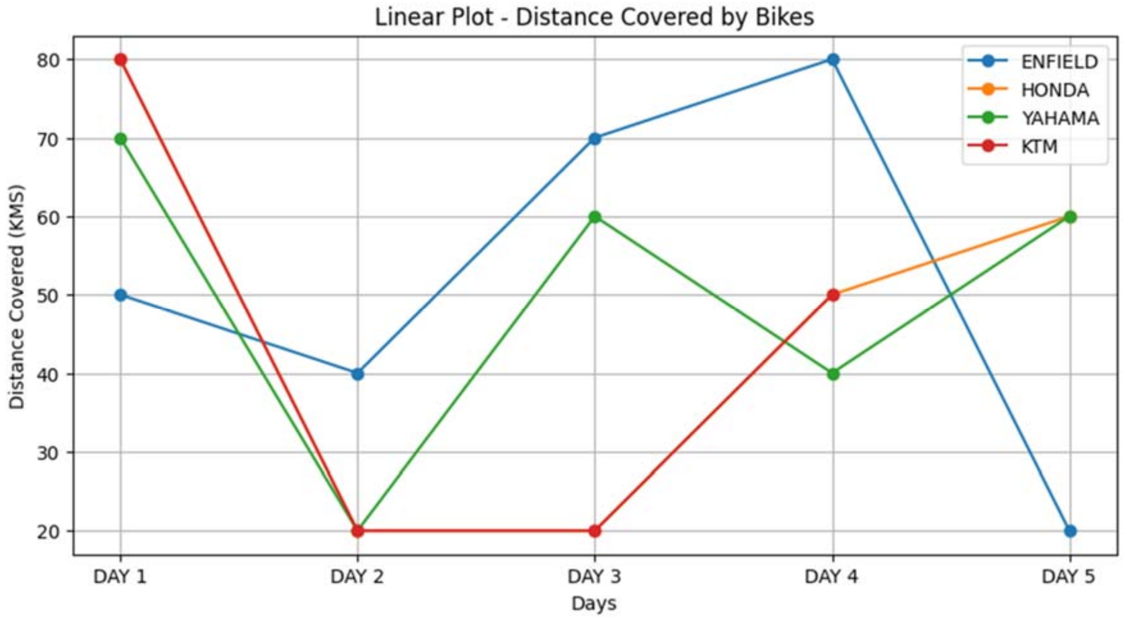
Aim: Visualize data using any plotting framework.

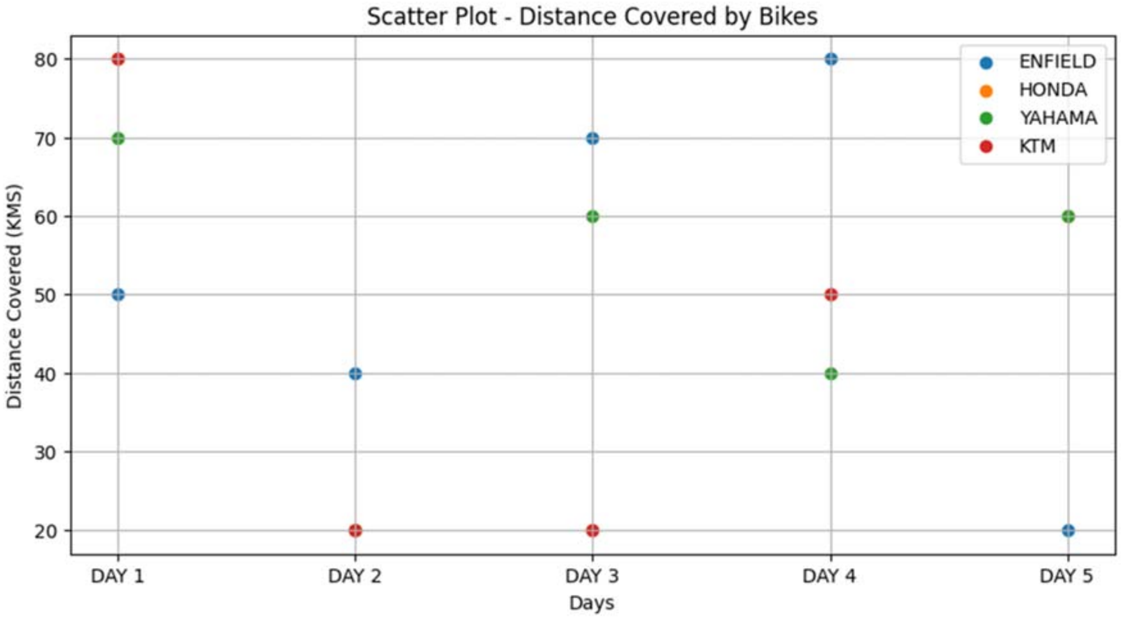
Code:

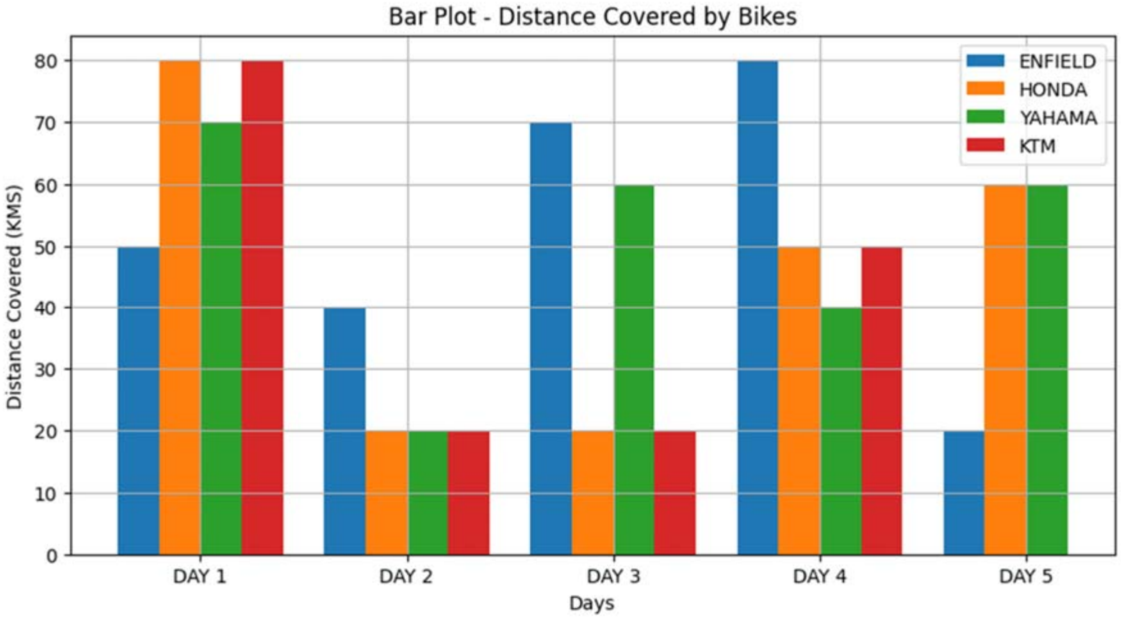
```
import pandas as pd
import matplotlib.pyplot as plt

# 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
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
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
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()
```

Output:




Scatter Plot - Distance Covered by Bikes
Distance Covered (KMS)
80
70
60
50
40
30
20
DAY 1
DAY 2
DAY 3
DAY 4
DAY 5
Days
ENFIELD
HONDA
YAHAMA
KTM


Bar Plot - Distance Covered by Bikes
Distance Covered (KMS)
80
70
60
50
40
30
20
10
0
DAY 1
DAY 2
DAY 3
DAY 4
DAY 5
Days
ENFIELD
HONDA
YAHAMA
KTM

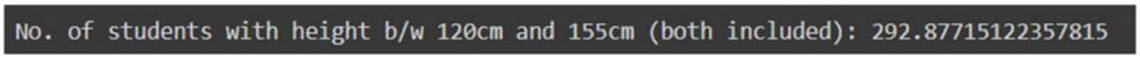
Experiment 10

Aim: Solve a numerical problem on Normal Distribution using python.

Code:

```
# 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 and 155cm.

from scipy.stats import norm

sample_size=500
mean = 151
std_dev = 15

# Calculate the probability using the cumulative distribution
function (CDF)
probability_a = norm.cdf(155, loc=mean, scale=std_dev) - \
    norm.cdf(120, loc=mean, scale=std_dev)
print(f"No. of students with height b/w 120cm and 155cm (both
included): {probability_a*sample_size}")
```

Output:



```
No. of students with height b/w 120cm and 155cm (both included): 292.87715122357815
```