KJ'S Educational Institutes

TRINITY ACADEMY OF ENGINEERING, PUNE

(Accredited with 'A' grade by NAAC)

Department of MCA

LABORATORY MANUAL

Big Data Analytics Laboratory

(Subject Code: 410904A)

For the Academic Year 2024 - 2025

SYMCA- Semester I

Teaching Scheme:
PR: 04Hours/Week

Credit 02

Examination Scheme:
TW:25Marks
PR: 50 Marks

PROGRAM OUTCOMES	
PO No.	**Program Outcome Description**
PO 1	Apply knowledge of mathematics, computer science, computing specializations appropriate for real world applications.
PO 2	Identify, formulate, analyze and solve *complex* computing problems using relevant domain disciplines.
PO 3	Design and evaluate solutions for *complex* computing problems that meet specified needs with appropriate considerations for real world problems.
PO 4	Find solutions of complex computing problems using design of experiments, analysis and interpretation of data.
PO 5	Apply appropriate techniques and modern computing tools for development of complex computing activities.
PO 6	Apply professional ethics, cyber regulations and norms of professional computing practices.
PO 7	Recognize the need to have ability to engage in independent and life-long learning in the broadest context of technological change.
PO 8	Demonstrate knowledge and understanding of the computing and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO 9	Communicate effectively with the computing community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO 10	Assess societal, environmental, health, safety, legal and cultural issues within local and global contexts, and the consequent responsibilities relevant to the professional computing practices.
PO 11	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary environments.
PO 12	Identify a timely opportunity and use innovation, to pursue opportunity, as a successful Entrepreneur /professional.

OBJECTIVE:

- Understand big data analytics concepts
- Solve big data problems using Hadoop
- Apply different Supervised learning and Unsupervised Learning algorithms
- Understand different data visualization techniques.
- Understand Hadoop Architecture
- Solve Complex real world problems in various applications like recommender systems, social media applications, etc.

Course Outcomes	
CO 1	Understand big data analytics concepts
CO 2	Solve big data problems using Hadoop
CO 3	Apply different Supervised learning and Unsupervised Learning algorithms
CO 4	Understand different data visualization techniques.
CO 5	Understand Hadoop Architecture
CO 6	Solve Complex real world problems in various applications like recommender systems,

Guidelines for Student Journal:

- The laboratory assignments are to be submitted by student in the form of journal.
- Journal consists of prologue, Certificate, table of contents, and **handwritten write-up** of each assignment (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory-Concept in brief conclusion/analysis.
- Program codes with sample output of all Performed assignments are to be submitted as softcopy. As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal may be avoided. Use of DVD containing students' programs maintained by lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at the Laboratory.

Sample Programs:

Practical No:01

Aim : Installation of Hadoop Framework, it‘s components and study the HADOOP

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Hadoop Architecture:

The Apache Hadoop framework includes following four modules:

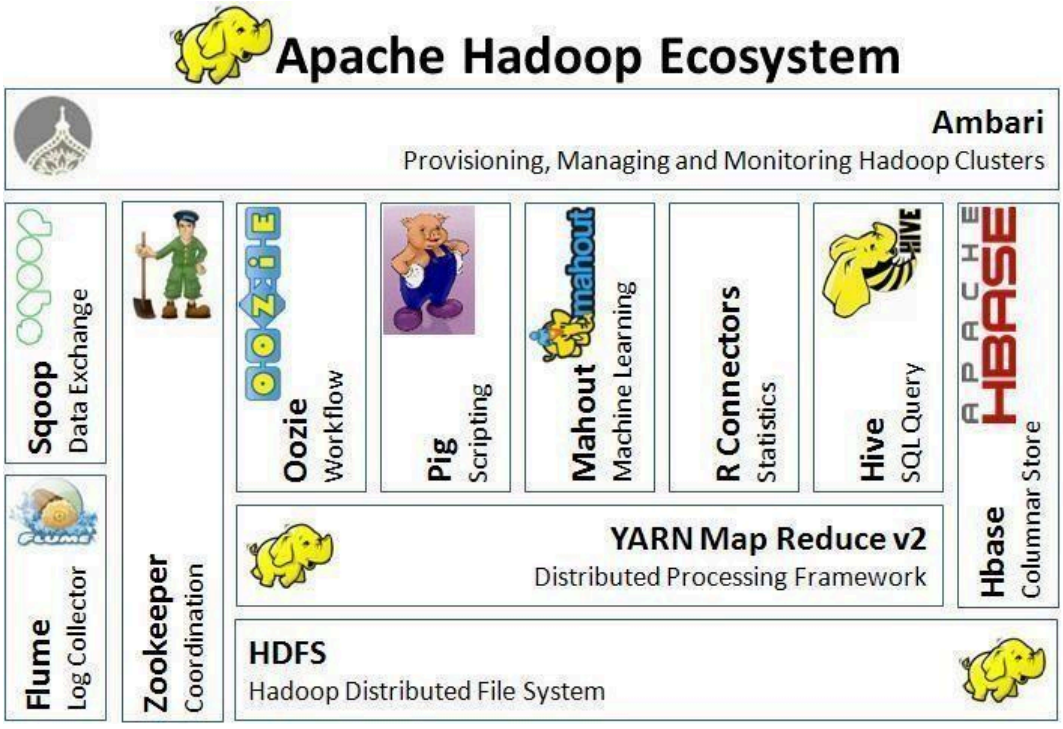
Hadoop Common: Contains Java libraries and utilities needed by other Hadoop modules. These libraries give file system and OS level abstraction and comprise of the essential Java files and scripts that are required to start Hadoop.

Hadoop Distributed File System (HDFS): A distributed file-system that provides high-throughput access to application data on the community machines thus providing very high aggregate bandwidth across the cluster.

Hadoop YARN: A resource-management framework responsible for job scheduling and cluster resource management.

Hadoop MapReduce: This is a YARN- based programming model for parallel processing of large data sets.

Hadoop Ecosystem:



Hadoop has gained its popularity due to its ability of storing, analyzing and accessing large amount of data, quickly and cost effectively through clusters of commodity hardware. It won ‘t be wrong if we say that Apache Hadoop is actually a collection of several components and not just a single product.

With Hadoop Ecosystem there are several commercial along with an open source product which are broadly used to make Hadoop laymen accessible and more usable.

MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process big amounts of data in-parallel on large clusters of commodity hardware in a reliable, fault- tolerant manner. In terms of programming, there are two functions which are most common in MapReduce.

• The Map Task: Master computer or node takes input and convert it into divide it into smaller parts and distribute it on other worker nodes. All worker nodes solve their own small problem and give answer to the master node.

• The Reduce Task: Master node combines all answers coming from worker node and forms it in some form of output which is answer of our big distributed problem.

Generally both the input and the output are reserved in a file-system. The framework is responsible for scheduling tasks, monitoring them and even re-executes the failed tasks.

Hadoop Distributed File System (HDFS)

HDFS is a distributed file-system that provides high throughput access to data. When data is pushed to HDFS, it automatically splits up into multiple blocks and stores/replicates the data thus ensuring high availability and fault tolerance.

Note: A file consists of many blocks (large blocks of 64MB and above).

Here are the main components of HDFS:

• Name Node: It acts as the master of the system. It maintains the name system i.e.,directories and files and manages the blocks which are present on the Data Nodes.

• Data Nodes: They are the slaves which are deployed on each machine and provide theactual storage. They are responsible for serving read and write requests for the clients.

• Secondary Name Node: It is responsible for performing periodic checkpoints. In theevent of Name Node failure, you can restart the Name Node using the checkpoint.

Hive

Hive is part of the Hadoop ecosystem and provides an SQL like interface to Hadoop. It is a data warehouse system for Hadoop that facilitates easy data summarization, ad-hoc queries, and the analysis of large datasets stored in Hadoop compatible file systems.

HBase (Hadoop DataBase)

HBase is a distributed, column oriented database and uses HDFS for the underlying storage. As said earlier, HDFS works on write once and read many times pattern, but this isn‘t a case always. We may require real time read/write random access for huge dataset; this is where HBase comes into the picture. HBase is built on top of HDFS and distributed on column- oriented database.

Installation Steps –

Following are steps to Install Apache Hadoop on Ubuntu 14.04

Step1. Install Java (OpenJDK) - Since hadoop is based on java, make sure you have java jdk installed on the system. Please check the version of java (It should be 1.7 or above it)

```
$ java –version
```

If it returns "The program java can be found in the following packages", If Java isn't been installed yet, so execute the following command:

```
$sudo apt-get install default-jdk
```

Step2: Configure Apache Hadoop

1. Open bashrc in gedit mode

```
$sudo gedit ~/.bashrc
```

2. Set java environment variable

```
export JAVA_HOME=/usr/jdk1.7.0_45/
```

3. Set Hadoop environment variable

```
export HADOOP_HOME=/usr/Hadoop 2.6/
```

4. Apply environment variables

```
$source ~/.bashrc
```

Step3: Install eclipse

Step4: Copy Hadoop plug-ins such as

- hadoop-eclipse-kepler-plugin-2.2.0.jar
- hadoop-eclipse-kepler-plugin-2.4.1.jar
- hadoop-eclipse-plugin-2.6.0.jar from release folder of hadoop2x-eclipse-plugin-master to eclipse plugins

Step5: In eclipse, start new MapReduce project

```
File->new->other->MapReduce project
```

Step 6: Copy Hadoop packages such as commons-io-2.4.jar commons-lang3-3.4.jar in src file of MapReduce project

Step 7: Create Mapper, Reducer, and driver

Inside a project->src->File->new->other->Mapper/Reducer/Driver

Step 8: Copy Log file log4j.properties from src file of hadoop in src file of MapReduce project

Hadoop is powerful because it is extensible and it is easy to integrate with any component. Its popularity is due in part to its ability to store, analyze and access large amounts of data, quickly and cost effectively across clusters of commodity hardware. Apache Hadoop is not actually a single product but instead a collection of several components. When all these components are merged, it makes the Hadoop very user friendly.

Practical No:02

Aim : On the iris dataset, perform KNN algorithm and discuss result

Code :

```
import pandas as pd

from sklearn.datasets import load_iris iris=load_iris()
iris.feature_names iris.target_names
df= pd.DataFrame(iris.data,columns=iris.feature_names) df.head()
df['target']=iris.target df.head() df[df.target==1].head() df.shape df['flower_name']
=df.target.apply(lambda x: iris.target_names[x] ) df.head() df0 =df[:50]
df1=df[50:100]
df2=df[100:]

from sklearn.model_selection import train_test_split x=df.drop(['target','flower_name'],axis='columns') y=df.target
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.2,random_state=1) len(x_train) from
sklearn.neighbors import KNeighborsClassifier # create knn classifier knn
=KNeighborsClassifier(n_neighbors=10) knn.fit(x_train,
y_train) knn.score(x_test,y_test)
from sklearn.metrics import confusion_matrix y_pred=knn.predict(x_test) cm =
confusion_matrix(y_test,y_pred) cm

%matplotlib inline
import matplotlib.pyplot as plt import seaborn as sn plt.figure(figsize=(7,5)) sn.heatmap(cm,
annot=True) plt.xlabel('predicted') plt.ylabel('True')
from sklearn.metrics import classification_report print(classification_report(y_test,y_pred)
```

Output :

feature names :

Target names :

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

len(x_train) : 120

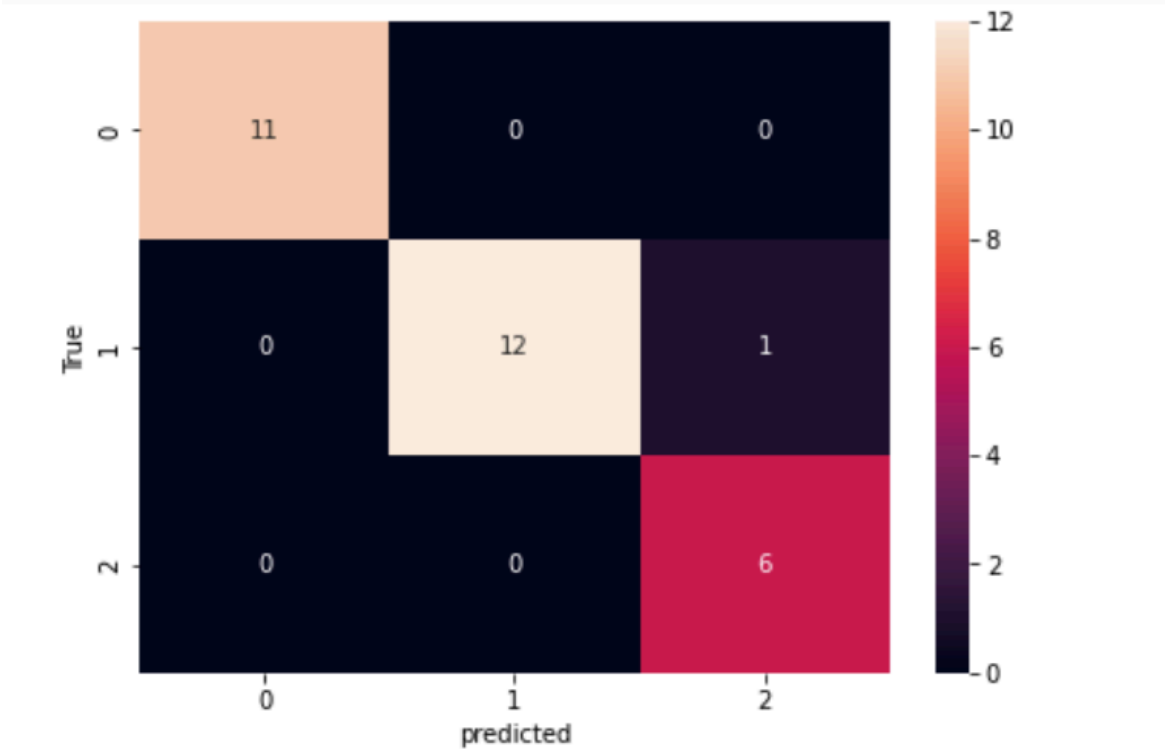
len(x_test) : 30

Knn score : 0.9666666666666667

Confusion matrix:

```
array([[11,  0,  0],
       [ 0, 12,  1],
       [ 0,  0,  6]], dtype=int64)
```

Heatmap:



Classification report :

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	0.92	0.96	13
2	0.86	1.00	0.92	6
accuracy			0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

Practical No:03

Aim : Implement apriori algorithm on Online retail dataset and discuss results**.**

Code :

```
import pandas as pd import numpy as np import seaborn as sns import matplotlib as mlp import
matplotlib.pyplot as plt
from mlxtend.frequent_patterns import apriori

from mlxtend.frequent_patterns import association_rules import os

os.getcwd() os.chdir("C:\\Users\\PramodJ\\Desktop ") transaction_df=pd.read_csv('OnlineRetail.csv')
transaction_df.head()
transaction_df = transaction_df[transaction_df.Country=='France'] transaction_filtered =
transaction_df[['InvoiceNo','Description','Quantity']].copy() transaction_filtered transaction_filtered =
transaction_filtered[transaction_filtered.Quantity > 0 ] transaction_filtered.sort_values(by='Quantity',
ascending=True) transaction_filtered['Quantity']= [1]*len(transaction_filtered)

invoice = list(transaction_filtered.InvoiceNo)

index_no = [invoice[index] for index in np.arange(len(invoice)) if not invoice[index].isnumeric()]
transaction_filtered[transaction_filtered['InvoiceNo'].isin(index_no)]
temp_df = transaction_filtered[transaction_filtered.Description != transaction_filtered.Description] temp_df
for invoice in list(temp_df.InvoiceNo):
if len(transaction_filtered[transaction_filtered.Invoice == invoice]) > 1: print((str)(invoice))

temp = transaction_filtered[transaction_filtered.Invoice == invoice].groupby(['Invoice']).agg({'Description':lambda
x: list(x)}) if len(list(set(temp)))>0 :
print(temp) transaction_filtered.dropna(axis=0, inplace=True) transaction_filtered def
return_one(x): return 1
table = pd.pivot_table(transaction_filtered, values='Quantity', index=['InvoiceNo'],
columns=['Description'], aggfunc=return_one, fill_value=0)
table

frequent_itemsets = apriori(table, min_support=0.01, use_colnames=True) frequent_itemsets rules =
association_rules(frequent_itemsets, metric="lift", min_threshold=1) rules
rules.sort_values(by=['support','confidence'], ascending=False)
```

Output :

Apriori :

	support	itemsets
0	0.022959	(DOLLY GIRL BEAKER)
1	0.012755	(I LOVE LONDON MINI BACKPACK)
2	0.017857	(SET 2 TEA TOWELS I LOVE LONDON)
3	0.040816	(SPACEBOY BABY GIFT SET)
4	0.030612	(10 COLOUR SPACEBOY PEN)
...	...	...
39622	0.010204	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...
39623	0.010204	(ALARM CLOCK BAKELIKE GREEN, JUMBO BAG APPLES,...
39624	0.010204	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...
39625	0.010204	(ALARM CLOCK BAKELIKE RED , JUMBO BAG APPLES, ...
39626	0.010204	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...

39627 rows × 2 columns

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(CHARLOTTE BAG DOLLY GIRL DESIGN)	(DOLLY GIRL BEAKER)	0.066327	0.022959	0.012755	0.192308	8.376068	0.011232	1.209670
1	(DOLLY GIRL BEAKER)	(CHARLOTTE BAG DOLLY GIRL DESIGN)	0.022959	0.066327	0.012755	0.555556	8.376068	0.011232	2.100765
2	(DOLLY GIRL CHILDRENS BOWL)	(DOLLY GIRL BEAKER)	0.045918	0.022959	0.017857	0.388889	16.938272	0.016803	1.598794
3	(DOLLY GIRL BEAKER)	(DOLLY GIRL CHILDRENS BOWL)	0.022959	0.045918	0.017857	0.777778	16.938272	0.016803	4.293367
4	(DOLLY GIRL BEAKER)	(DOLLY GIRL CHILDRENS CUP)	0.022959	0.040816	0.015306	0.666667	16.333333	0.014369	2.877551
...	...	...	...	...	...	...	...	...	...
1349507	(LUNCH BAG APPLE DESIGN)	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...	0.125000	0.010204	0.010204	0.081633	8.000000	0.008929	1.077778
1349508	(LUNCH BOX I LOVE LONDON)	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...	0.068878	0.010204	0.010204	0.148148	14.518519	0.009501	1.161934
1349509	(DOLLY GIRL CHILDRENS CUP)	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...	0.040816	0.010204	0.010204	0.250000	24.500000	0.009788	1.319728
1349510	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...	0.102041	0.010204	0.010204	0.100000	9.800000	0.009163	1.099773
1349511	(ROUND SNACK BOXES SET OF4 WOODLAND)	(ALARM CLOCK BAKELIKE GREEN, ALARM CLOCK BAKEL...	0.158163	0.010204	0.010204	0.064516	6.322581	0.008590	1.058058

Practical No:04

Aim : Implement Naïve Bayes Classifier and K-Nearest Neighbor Classifier on Data set of your choice. Test and Compare for Accuracy and Precision.

Code : KNN

```
import numpy as np import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report from sklearn.model_selection import train_test_split from sklearn.preprocessing import MinMaxScaler from sklearn.linear_model import LogisticRegression from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis from sklearn.svm
import SVC import os
os.chdir("C:\\Users\PramodJ\Desktop\Python") fruits = pd.read_table('fruit.txt') print(fruits.head())
print(fruits.tail())
print (fruits.describe()) feature_names=['mass','width','height','color_score'] X = fruits[feature_names] Y
= fruits['fruit_label']
X_train, X_test,y_train,y_test=train_test_split(X,Y,random_state=0) scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train) X_test= scaler.transform(X_test)
knn = KNeighborsClassifier() knn.fit(X_train, y_train) print('Accuracy of K-NN classifier on training
set: {:.2f}'.format(knn.score(X_train,y_train))) print('Accuracy of K-NN classifier on test
set: {:.2f}'.format(knn.score(X_test,y_test)))

import numpy as np
import matplotlib.pyplot as plt import pandas as pd
os.chdir("C:\\Users\\PramodJ\\Desktop\\Python") dataset = pd.read_csv('iris.csv') X
= dataset.iloc[:,:4].values y = dataset['variety'].values dataset.head(5)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2) from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train) X_test = sc.transform(X_test)
from sklearn.naive_bayes import GaussianNB classifier = GaussianNB()
classifier.fit(X_train, y_train) y_pred = classifier.predict(X_test) y_pred
from sklearn.metrics import confusion_matrix cm = confusion_matrix(y_test, y_pred)
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred)) cm
```

Output : KNN

	fruit_label	fruit_name	fruit_subtype	mass	width	height	color_score
0	1	apple	granny_smith	192	8.4	7.3	0.55
1	1	apple	granny_smith	180	8.0	6.8	0.59
2	1	apple	granny_smith	176	7.4	7.2	0.60
3	2	mandarin	mandarin	86	6.2	4.7	0.80
4	2	mandarin	mandarin	84	6.0	4.6	0.79
5	2	mandarin	mandarin	80	5.8	4.3	0.77
6	2	mandarin	mandarin	80	5.9	4.3	0.81
7	2	mandarin	mandarin	76	5.8	4.0	0.81
8	1	apple	braeburn	178	7.1	7.8	0.92
9	1	apple	braeburn	172	7.4	7.0	0.89

	fruit_label	mass	width	height	color_score
count	59.000000	59.000000	59.000000	59.000000	59.000000
mean	2.542373	163.118644	7.105085	7.693220	0.762881
std	1.208048	55.018832	0.816938	1.361017	0.076857
min	1.000000	76.000000	5.800000	4.000000	0.550000
25%	1.000000	140.000000	6.600000	7.200000	0.720000
50%	3.000000	158.000000	7.200000	7.600000	0.750000
75%	4.000000	177.000000	7.500000	8.200000	0.810000
max	4.000000	362.000000	9.600000	10.500000	0.930000

```
Accuracy of K-NN classifier on training set:0.95

Accuracy of K-NN classifier on training set:1.00
```

naïve bayes :

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
Accuracy :  0.9333333333333333
```

Confusion matrix :

```
array([[10,  0,  0],
       [ 0,  8,  1],
       [ 0,  1, 10]], dtype=int64)
```

Practical No:05

Aim : Implement K-Means Clustering on the proper data set of your choice.

Code :

```
from sklearn.cluster import KMeans import pandas as pd
import matplotlib.pyplot as plt from matplotlib import style style.use("ggplot")
%matplotlib inline
data = pd.DataFrame([[1, 2], [5, 8],
[1.5, 1.8],
[8, 8],
[1, 0.6],
[9, 11]], columns=['x','y']) print( data )
kmeans = KMeans(n_clusters=2).fit(data) centroids = kmeans.cluster_centers_ labels = kmeans.labels_
print(centroids) print(labels)
```

Output :

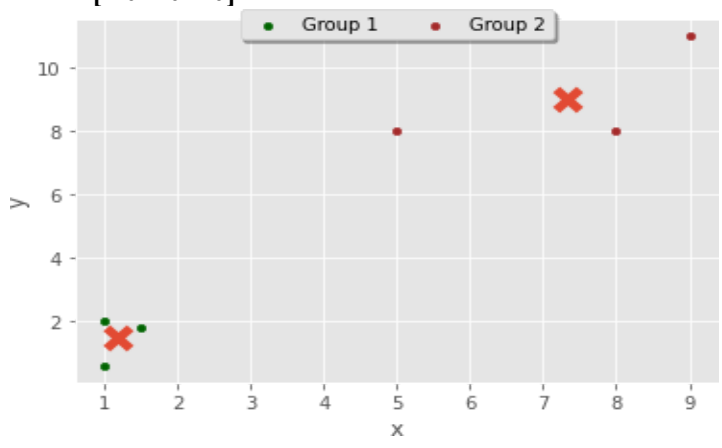
Centroids :

```
[[1.16666667 1.46666667 1.       ]
[7.33333333 9. 0.          ]]
```

	x	y
0	1.0	2.0
1	5.0	8.0
2	1.5	1.8
3	8.0	8.0
4	1.0	0.6
5	9.0	11.0

Labels

[1 0 1 0 1 0]



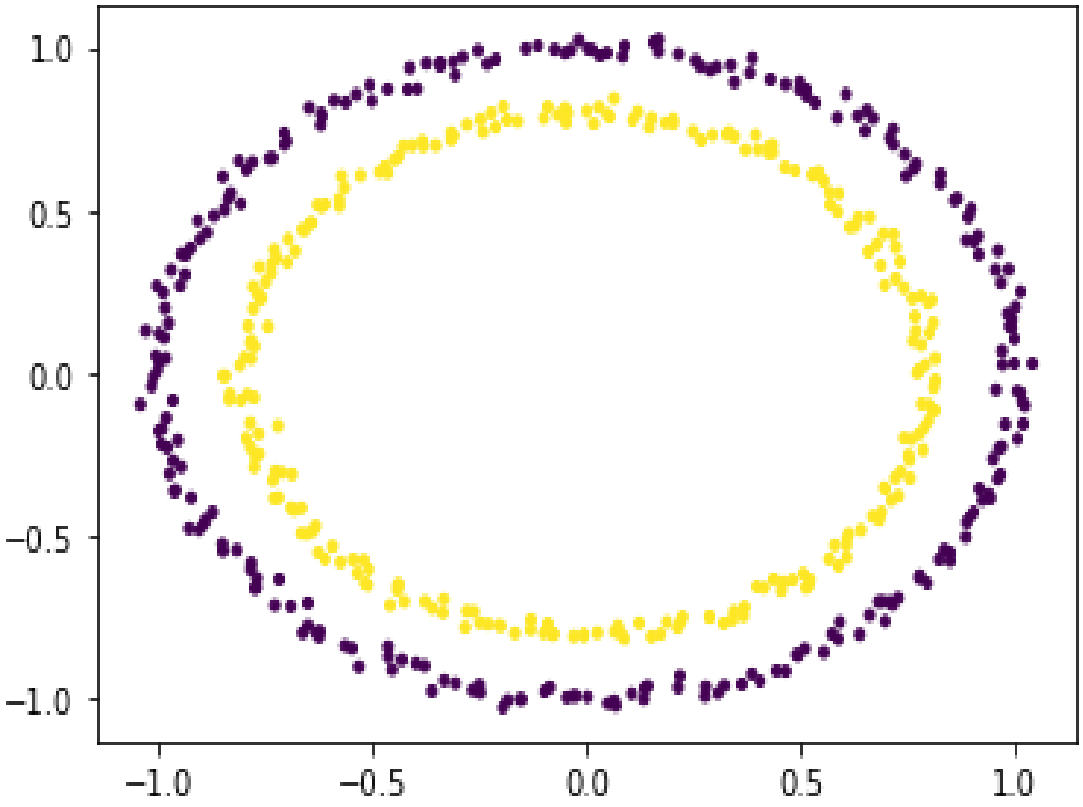
Practical No:06

Aim : Design and implement SVM for classification with the proper data set of your choice. Comment on Design and Implementation for Linearly non separable Dataset.

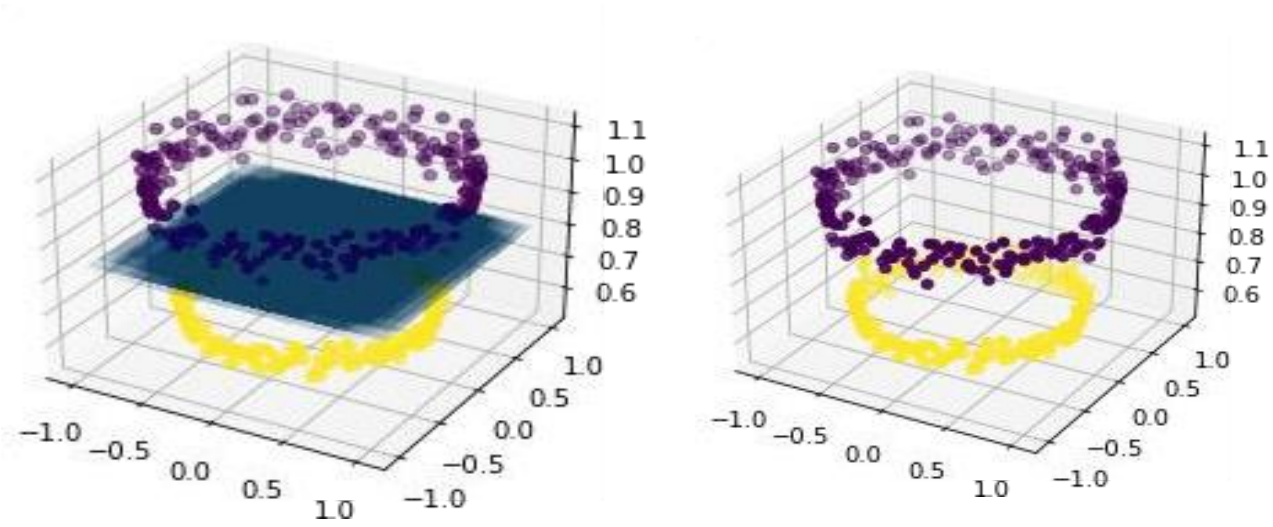
Code :

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles from mpl_toolkits.mplot3d import Axes3D X, Y=

make_circles(n_samples = 500, noise = 0.02)

plt.scatter(X[:, 0], X[:, 1], c = Y, marker = '.') plt.show() X1 =
X[:, 0].reshape((-1, 1))
X2 = X[:, 1].reshape((-1, 1)) X3 = (X1**2 + X2**2)
X = np.hstack((X, X3))

# visualizing data in higher dimension fig = plt.figure()
axes = fig.add_subplot(111, projection = '3d') axes.scatter(X1, X2, X1**2 + X2**2, c = Y, depthshade = True)
plt.show()
from sklearn import svm

svc = svm.SVC(kernel = 'linear') svc.fit(X, Y) w =
svc.coef_
b = svc.intercept_

x1 = X[:, 0].reshape((-1, 1))
x2 = X[:, 1].reshape((-1, 1))
x1, x2 = np.meshgrid(x1, x2)
x3 = -(w[0][0]*x1 + w[0][1]*x2 + b) / w[0][2]

fig = plt.figure()
axes2 = fig.add_subplot(111, projection = '3d') axes2.scatter(X1, X2, X1**2 + X2**2, c = Y, depthshade = True)
axes1 = fig.gca(projection = '3d')
axes1.plot_surface(x1, x2, x3, alpha = 0.01) plt.show()
```

Output:



support vector classifier using a linear kernel



Practical No:07

Aim : Implementing distinct word count problem using Map-Reduce

The function of the mapper is as follows:

- Create a IntWritable variable 'one' with value as 1
- Convert the input line in Text type to a String
- Use a tokenizer to split the line into words
- Iterate through each word and a form key value pairs as
  Assign each work from the tokenizer (of String type) to a Text 'word'
- Form key value pairs for each word as < word,one > and push it to the output collector

The function of Sort and Group:

After this, "aggregation" and "Shuffling and Sorting" done by framework. Then Reducers task these final pair to produce output.

The function of the reducer is as follows

- Initialize a variable 'sum' as 0
- Iterate through all the values with respect to a key and sum up all of them
- Push to the output collector the Key and the obtained sum as value For

Example:
For the given sample input1 data file (input1.txt : Hello World Bye World) mapper emits:
<Hello, 1>
<World, 1>
<Bye, 1>
<World, 1>

The second input2 data file (input2.txt : Hello Hadoop Goodbye Hadoop) mapper emits:
<Hello, 1>
<Hadoop, 1>
<Goodbye, 1>
<Hadoop, 1>

WordCount also specifies a combiner. Hence, the output of each map is passed through the local combiner (which is same as the Reducer as per the job configuration) for local aggregation, after being sorted on the keys.

The output of the first map:
<Bye, 1>

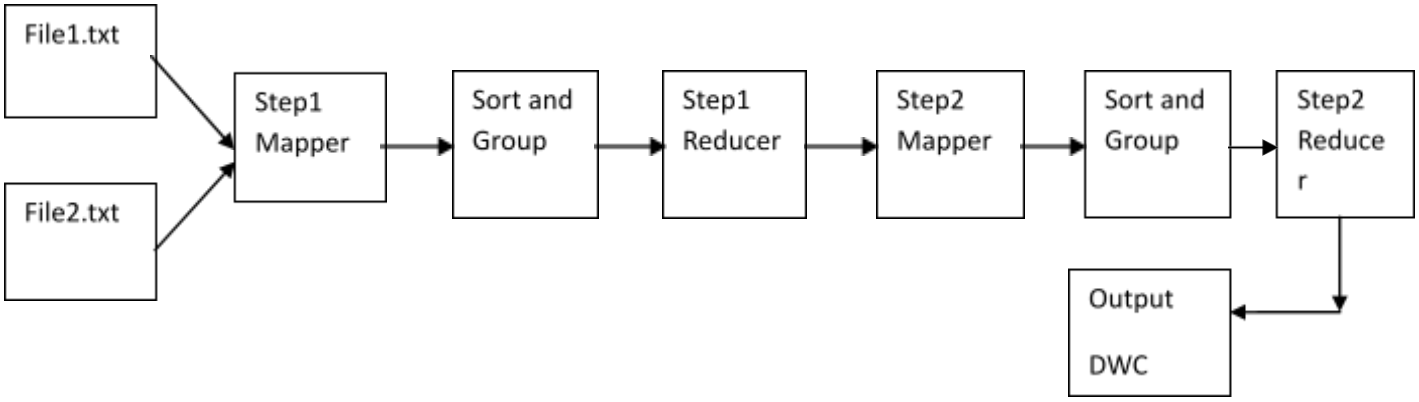
<Hello, 1>
<World, 2>

The output of the second map:
<Goodbye, 1>
<Hadoop, 2>
<Hello, 1>

The Reducer implementation via the reduce method just sums up the values, which are the occurrence counts for each key (i.e. words in this example).

Thus the output of the job is:
<Bye, 1>
<Goodbye, 1>
<Hadoop, 2>
<Hello, 2>
<World, 2>

Data Flow Diagram:



Content of Lab Experiments for Journal

S.No.	Title of Program
1	Installation of Hadoop Framework, its components and study of HADOOP.
2	On the iris dataset, perform a knn algorithm and discuss the result.
3	Implement apriori algorithm on Online retail dataset and discuss results.
4	Implement Naïve Bayes Classifier and K-Nearest Neighbor Classifier on Data set of your choice. Test and Compare for Accuracy and Precision.
5	Implement K-Means Clustering on the proper data set of your choice.
6	Design and implement SVM for classification with the proper data set of your choice. Comment on Design and Implementation for Linearly non separable Dataset.
7	Implementing distinct word count problem using Map-Reduce