C. ABDUL HAKEEM COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai)

(NBA Accredited & ISO 9001:2000 Certified Institution)

Melvisharam – 632 509, Ranipet District, Tamil Nadu.

Name ……………………………………………………………………………………

Year…………………...Semester………………….. Branch……………………………

Subject Code: …….....................Subject Name: ……………………………................

University Register Number:

Certificate

Certified that this is the bonafide record of work done by the above student in the ……………………………………………… Laboratory during the year 2023 – 2024.

Signature of Head of the Department　　　　**Signature of Lab. Incharge**

Submitted for the University Practical Examination held on …………………………

<u>Examiners</u>

Date: …………………………….　　　　**Centre Code:…………………………...**

Internal: ………………………..　　　　**External: ………………………………**

DEPARTMENTAL VISION AND MISSION

VISION

- Providing knowledge of Computer Applications to enable the MCA graduates to meet global challenges.

MISSION

- To inculcate skills that can bridge the gap between industry and academia.
- To develop innovative talents among students.
- To practice ethical values and making them engage in life-long learning.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

The Master of Computer Applications graduate will able to:

PEO	Statements
PEO1	Apply their computing skills to analyse, design and develop innovative software products to meet the industry needs and excel as software professionals.
PEO2	Pursue lifelong learning and do research in the computing field based on solid technical foundations.
PEO3	Communicate and function effectively in teams in multidisciplinary fields within the global, societal and environmental context.
PEO4	Exhibit professional integrity, ethics and an understanding of responsibility to contribute technical solutions for the sustainable development of society.

PROGRAM SPECIFIC OUTCOMES (PSOs)

The Graduates of MCA will be:

PSOs	Statements
PSO1	Able to select suitable data models, appropriate architecture and platform to implement a system with good performance.
PSO2	Able to design and integrate various system based components to provide user interactive solutions for various challenges.
PSO3	Able to develop applications for real time environment using existing and upcoming technologies.

PROGRAM OUTCOMES (POs)

PO1. Computational Knowledge:

Apply knowledge of computing fundamentals, computing specialization, mathematics, and domain knowledge appropriate for the computing specialization to the abstraction and conceptualization of computing models from defined problems and requirements.

PO2. Problem Analysis:

Identify, formulate, research literature, and solve complex computing problems reaching substantiated conclusions using fundamental principles of mathematics, computing sciences, and relevant domain disciplines.

PO3. Design /Development of Solutions:

Design and evaluate solutions for complex computing problems, and design and evaluate systems, components, or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal, and environmental considerations.

PO4. Conduct Investigations of Complex Computing Problems:

Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern Tool Usage:

Create, select, adapt and apply appropriate techniques, resources, and modern computing tools to complex computing activities, with an understanding of the limitations.

PO6. Professional Ethics:

Understand and commit to professional ethics and cyber regulations, responsibilities, and norms of professional computing practice.

PO7. Life-long Learning:

Recognize the need, and have the ability, to engage in independent learning for continual development as a computing professional.

PO8. Project management and finance:

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.

PO9. Communication Efficacy:

Communicate effectively with the computing community, and with society at large, about complex computing activities by being able to comprehend and write effective reports, design documentation, make effective presentations, and give and understand clear instructions.

PO10. Societal and Environmental Concern:

Understand and assess societal, environmental, health, safety, legal, and cultural issues within local and global contexts, and the consequential responsibilities relevant to professional computing practice.

PO11. Individual and Team Work:

Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary environments.

PO12. Innovation and Entrepreneurship

Identify a timely opportunity and using innovation to pursue that opportunity to create value and wealth for the betterment of the individual and society at large.

COURSE OBJECTIVES

The learning objectives of this course are to

S.No	Objectives
1	To understand about data cleaning and data pre-processing.
2	To measure the performance of machine learning models and implement feature selection technique.
3	To implement Bayesian, EM algorithm, parametric and non-parametric machine learning techniques to solve the problems.

COURSE OUTCOMES

Upon completion of the course, the students will be able to

COs	Course Outcomes (CO)
CO.1	Demonstrate Data Cleaning and Data Pre-processing.
CO.2	Measure the performance of machine learning models and implement feature selection technique.
CO.3	Implement various machine learning techniques to solve the problems.

LIST OF EXPERIMENTS

1. Demonstrate how do you structure data in Machine Learning

2. Implement data preprocessing techniques on real time dataset

3. Implement Feature subset selection techniques

4. Demonstrate how will you measure the performance of a machine learning model

5. Write a program to implement the naïve Bayesian classifier for a sample training data set. Compute the accuracy of the classifier, considering few test data sets.

6. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using the standard Heart Disease Data Set.

7. Apply EM algorithm to cluster a set of data stored in a .CSV file.

8. Write a program to implement k-Nearest Neighbor algorithm to classify the data set.

9. Apply the technique of pruning for a noisy data monk2 data, and derive the decision tree from this data. Analyze the results by comparing the structure of pruned and unpruned tree.

10. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets

11. Implement Support Vector Classification for linear kernels.

12. Implement Logistic Regression to classify problems such as spam detection. Diabetes predictions and so on.

Index

Ex No. 1	**Data Structuring Techniques in ML**
Date:	

Aim

To structure data using various data structuring techniques such as Data Binning, Data Sampling, Data Aggregation and Data Dimensionality Reduction, in machine learning.

Definitions

Data Binning

Data binning, also called data discrete binning or data bucketing, is a data pre-processing technique used to reduce the effects of minor observation errors. The original data values which fall into a given small interval, a bin, are replaced by a value representative of that interval, often a central value.

Data Sampling

Sampling is the practice of analyzing a subset of all data in order to uncover the meaningful information in the larger data set.

Data Aggregation

Data aggregation is the process where raw data is gathered and expressed in a summary form for statistical analysis.

Data Dimensionality Reduction

Dimensionality reduction, or dimension reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

Data Binning

Binning.py

```
import numpy as np
import math
from sklearn.datasets import load_iris
from sklearn import datasets, linear_model, metrics

# load iris data set
```

```
dataset = load_iris()
a = dataset.data
b = np.zeros(150)

# take 1st column among 4 column of data set
for i in range(150):
    b[i] = a[i, 1]

b = np.sort(b)  # sort the array

# create bins
bin1 = np.zeros((30, 5))
bin2 = np.zeros((30, 5))
bin3 = np.zeros((30, 5))

# Bin mean
for i in range(0, 150, 5):
    k = int(i / 5)
    mean = (b[i] + b[i + 1] + b[i + 2] + b[i + 3] + b[i + 4]) / 5
    for j in range(5):
        bin1[k, j] = mean
print("Bin Mean: \n", bin1)

# Bin boundaries
for i in range(0, 150, 5):
    k = int(i / 5)
    for j in range(5):
        if (b[i + j] - b[i]) < (b[i + 4] - b[i + j]):
            bin2[k, j] = b[i]
        else:
            bin2[k, j] = b[i + 4]
print("Bin Boundaries: \n", bin2)

# Bin median
for i in range(0, 150, 5):
    k = int(i / 5)
    for j in range(5):
        bin3[k, j] = b[i + 2]
print("Bin Median: \n", bin3)
```

Output

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32

runfile('C:/Users/2mca1/binning.py', wdir='C:/Users/2mca1')

Bin Mean:

 [[2.18 2.18 2.18 2.18 2.18]

 [2.34 2.34 2.34 2.34 2.34]

 [2.48 2.48 2.48 2.48 2.48]

 [2.52 2.52 2.52 2.52 2.52]

 [2.62 2.62 2.62 2.62 2.62]
 [2.7  2.7  2.7  2.7  2.7 ]
 [2.74 2.74 2.74 2.74 2.74]
 [2.8  2.8  2.8  2.8  2.8 ]
 [2.8  2.8  2.8  2.8  2.8 ]
 [2.86 2.86 2.86 2.86 2.86]
 [2.9  2.9  2.9  2.9  2.9 ]
 [2.96 2.96 2.96 2.96 2.96]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.   3.   3.   3.   3.  ]
 [3.04 3.04 3.04 3.04 3.04]
 [3.1  3.1  3.1  3.1  3.1 ]
 [3.12 3.12 3.12 3.12 3.12]
 [3.2  3.2  3.2  3.2  3.2 ]
 [3.2  3.2  3.2  3.2  3.2 ]
 [3.26 3.26 3.26 3.26 3.26]
 [3.34 3.34 3.34 3.34 3.34]
 [3.4  3.4  3.4  3.4  3.4 ]
 [3.4  3.4  3.4  3.4  3.4 ]
 [3.5  3.5  3.5  3.5  3.5 ]
 [3.58 3.58 3.58 3.58 3.58]
 [3.74 3.74 3.74 3.74 3.74]
 [3.82 3.82 3.82 3.82 3.82]
 [4.12 4.12 4.12 4.12 4.12]]
Bin Boundaries:
 [[2.  2.3 2.3 2.3 2.3]
 [2.3 2.3 2.3 2.4 2.4]
 [2.4 2.5 2.5 2.5 2.5]
 [2.5 2.5 2.5 2.5 2.6]
 [2.6 2.6 2.6 2.6 2.7]
 [2.7 2.7 2.7 2.7 2.7]
 [2.7 2.7 2.7 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.8 2.8 2.8]
 [2.8 2.8 2.9 2.9 2.9]
 [2.9 2.9 2.9 2.9 2.9]
 [2.9 2.9 3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.  3. ]
 [3.  3.  3.  3.1 3.1]
 [3.1 3.1 3.1 3.1 3.1]
 [3.1 3.1 3.1 3.1 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.2 3.2 3.2]
 [3.2 3.2 3.3 3.3 3.3]
 [3.3 3.3 3.3 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.4 3.4 3.4 3.4 3.4]
 [3.5 3.5 3.5 3.5 3.5]
 [3.5 3.6 3.6 3.6 3.6]
 [3.7 3.7 3.7 3.8 3.8]
 [3.8 3.8 3.8 3.8 3.9]
 [3.9 3.9 3.9 4.4 4.4]]
Bin Median:
 [[2.2 2.2 2.2 2.2 2.2]
```

```
[2.3 2.3 2.3 2.3 2.3]
[2.5 2.5 2.5 2.5 2.5]
[2.5 2.5 2.5 2.5 2.5]
[2.6 2.6 2.6 2.6 2.6]
[2.7 2.7 2.7 2.7 2.7]
[2.7 2.7 2.7 2.7 2.7]
[2.8 2.8 2.8 2.8 2.8]
[2.8 2.8 2.8 2.8 2.8]
[2.9 2.9 2.9 2.9 2.9]
[2.9 2.9 2.9 2.9 2.9]
[3.  3.  3.  3.  3. ]
[3.  3.  3.  3.  3. ]
[3.  3.  3.  3.  3. ]
[3.  3.  3.  3.  3. ]
[3.  3.  3.  3.  3. ]
[3.  3.  3.  3.  3. ]
[3.1 3.1 3.1 3.1 3.1]
[3.1 3.1 3.1 3.1 3.1]
[3.2 3.2 3.2 3.2 3.2]
[3.2 3.2 3.2 3.2 3.2]
[3.3 3.3 3.3 3.3 3.3]
[3.3 3.3 3.3 3.3 3.3]
[3.4 3.4 3.4 3.4 3.4]
[3.4 3.4 3.4 3.4 3.4]
[3.5 3.5 3.5 3.5 3.5]
[3.6 3.6 3.6 3.6 3.6]
[3.7 3.7 3.7 3.7 3.7]
[3.8 3.8 3.8 3.8 3.8]
[4.1 4.1 4.1 4.1 4.1]]
```

Data Sampling

Random-sampling.py

```
import numpy as np

# generating population data following Normal Distribution
N = 10000
mu = 10
std = 2
population_df = np.random.normal(mu,std,N)

# function that creates random sample
def random_sampling(df, n):
   random_sample = np.random.choice(df,replace = False, size = n)
   return(random_sample)
randomSample = random_sampling(population_df, N)
print(randomSample)
```

Output

```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/randomsampling.py

[10.50911361 10.17222969 10.40282203 ...  8.91980209 12.00965177

 11.12126602]

Process finished with exit code 0
```

Systematic-sampling.py

```
import numpy as np
import pandas as pd
# generating population data following Normal Distribution
N = 10000
mu = 10
std = 2
population_df = np.random.normal(mu,std,N)

# function that creates random sample using Systematic Sampling
def systematic_sampling(df, step):
    id = pd.Series(np.arange(1,len(df),1))
    df = pd.Series(df)
    df_pd = pd.concat([id, df], axis = 1)
    df_pd.columns = ["id", "data"]
    # these indices will increase with the step amount not 1
    selected_index = np.arange(1,len(df),step)
    # using iloc for getting thee data with selected indices
    systematic_sampling = df_pd.iloc[selected_index]
    return(systematic_sampling)

n = 10
step = int(N/n)
sample = systematic_sampling(population_df, step)
print(sample)
```

Output

```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/systematicsampling.py

          id       data

1        2.0  11.376670

1001  1002.0   7.902640

2001  2002.0   9.362893

3001  3002.0  13.432706
```

```
4001  4002.0  12.517116

5001  5002.0   7.566955

6001  6002.0  11.995210

7001  7002.0  11.061732

8001  8002.0   7.779656

9001  9002.0  12.856361

Process finished with exit code 0
```

Data Aggregation

Aggregation.py

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(10, 4),
     index = pd.date_range('1/1/2000', periods=10),
     columns = ['A', 'B', 'C', 'D'])

print(df)

r = df.rolling(window=3,min_periods=1)
print(r)
```

Output

```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/dataaggregation.py


                   A         B         C         D

2000-01-01  1.083986  0.590291 -0.702095 -2.022874

2000-01-02 -1.408233 -1.543042 -0.659563  0.578917

2000-01-03 -1.019651  1.454612 -0.742483  0.310860

2000-01-04 -1.837544 -2.381635  0.275290 -1.383948

2000-01-05 -0.183242  0.285801 -0.156981  0.400867

2000-01-06  1.397718  0.221107  0.910881  0.495881

2000-01-07 -1.013295  0.767176 -2.444918 -1.207323

2000-01-08  0.245632 -0.710832  0.006713  2.212504
```

```
2000-01-09  1.010055 -0.330848 -0.768370 -0.034331

2000-01-10 -1.172167 -1.429019  1.280776  0.227238

Rolling [window=3,min_periods=1,center=False,axis=0,method=single]

Process finished with exit code 0
```

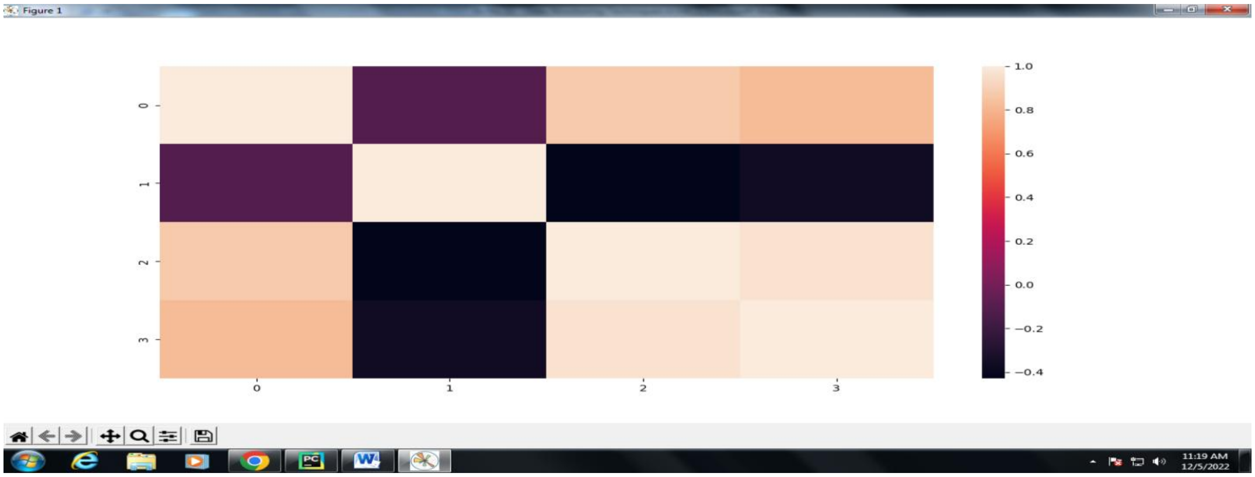
Data Dimensionality Reduction

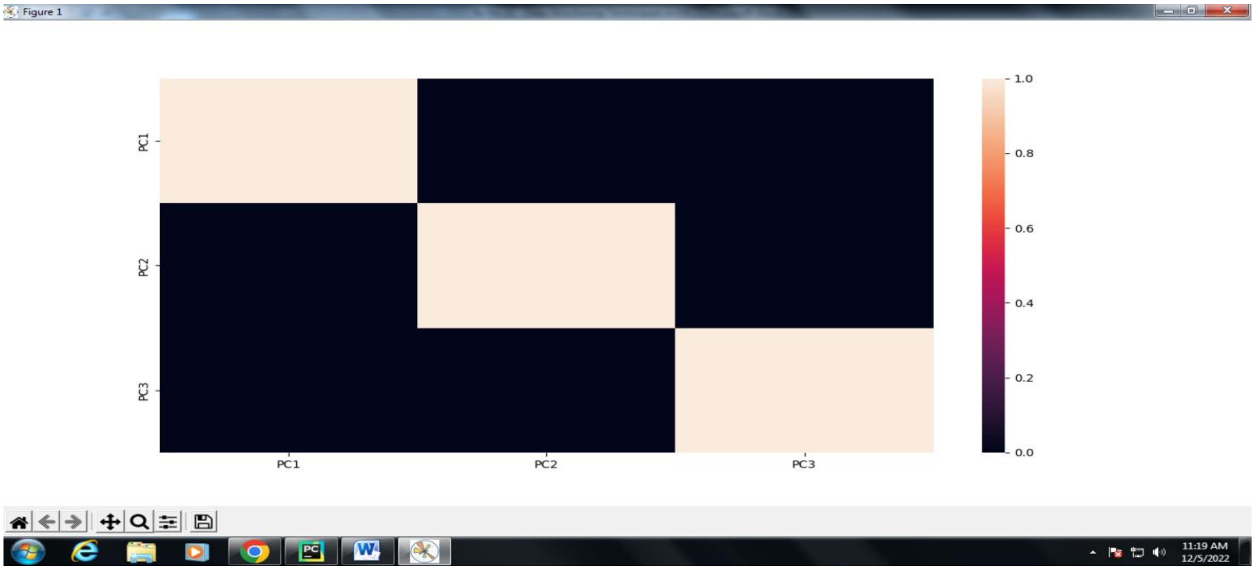
Dimen-reduction.py

```
# Import necessary libraries
from sklearn import datasets # to retrieve the iris Dataset
import pandas as pd # to load the dataframe
from sklearn.preprocessing import StandardScaler # to standardize the features
from sklearn.decomposition import PCA # to apply PCA
import seaborn as sns # to plot the heat maps
import matplotlib.pyplot as plt

#Load the Dataset
iris = datasets.load_iris()
#convert the dataset into a pandas data frame
df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
#display the head (first 5 rows) of the dataset
print(df.head())

#Standardize the features
#Create an object of StandardScaler which is present in sklearn.preprocessing
scalar = StandardScaler()
scaled_data = pd.DataFrame(scalar.fit_transform(df)) #scaling the data
print(scaled_data)


#Check the Co-relation between features without PCA
sns.heatmap(scaled_data.corr())
plt.show()

#Applying PCA
#Taking no. of Principal Components as 3
pca = PCA(n_components = 3)
pca.fit(scaled_data)
data_pca = pca.transform(scaled_data)
data_pca = pd.DataFrame(data_pca,columns=['PC1','PC2','PC3'])
print(data_pca.head())


#Checking Co-relation between features after PCA
sns.heatmap(data_pca.corr())
plt.show()
```

Output

```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/dimensionality-reduction.py

   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)

0             5.1              3.5               1.4               0.2

1             4.9              3.0               1.4               0.2

2             4.7              3.2               1.3               0.2

3             4.6              3.1               1.5               0.2

4             5.0              3.6               1.4               0.2

          0         1         2         3

0   -0.900681  1.019004 -1.340227 -1.315444

1   -1.143017 -0.131979 -1.340227 -1.315444

2   -1.385353  0.328414 -1.397064 -1.315444

3   -1.506521  0.098217 -1.283389 -1.315444

4   -1.021849  1.249201 -1.340227 -1.315444

..        ...       ...       ...       ...

145  1.038005 -0.131979  0.819596  1.448832

146  0.553333 -1.282963  0.705921  0.922303

147  0.795669 -0.131979  0.819596  1.053935

148  0.432165  0.788808  0.933271  1.448832

149  0.068662 -0.131979  0.762758  0.790671

[150 rows x 4 columns]
```





Result

Thus various data structuring techniques have been successfully applied to structure the data in machine learning.

Ex No. 2	**Pre-processing Techniques in Machine Learning**
Date:	

Aim

To implement pre-processing techniques in machine learning.

Definition

Data Pre-processing

Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects.

It involves below steps:

- Getting the dataset
- Importing libraries
- Importing datasets
- Finding Missing Data
- Encoding Categorical Data
- Splitting dataset into training and test set
- Feature scaling

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

1. **Data Rescaling:**

Data Normalization:

```
# Normalize the data attributes for the Iris dataset.

from sklearn.datasets import load_iris

from sklearn import preprocessing

# load the iris dataset

iris = load_iris()

print(iris.data.shape)
```

```
# separate the data from the target attributes

X = iris.data

y = iris.target

# normalize the data attributes

normalized_X = preprocessing.normalize(X)

print(normalized_X)
```

Output:

```
(150, 4)

[[0.80377277 0.55160877 0.22064351
0.0315205 ]

 [0.82813287 0.50702013 0.23660939
0.03380134]

 [0.80533308 0.54831188 0.2227517
0.03426949]

 [0.80003025 0.53915082 0.26087943
0.03478392]

 [0.790965   0.5694948  0.2214702  0.0316386 ]

 [0.78417499 0.5663486  0.2468699
0.05808704]

 [0.78010936 0.57660257 0.23742459
0.0508767 ]

 [0.80218492 0.54548574 0.24065548
0.0320874 ]

 [0.80642366 0.5315065  0.25658935
0.03665562]

 [0.81803119 0.51752994 0.25041771
0.01669451]

 [0.80373519 0.55070744 0.22325977
0.02976797]

 [0.786991   0.55745196 0.26233033
0.03279129]

 [0.82307218 0.51442011 0.24006272
0.01714734]

 [0.8025126  0.55989251 0.20529392
0.01866308]

 [0.81120865 0.55945424 0.16783627
0.02797271]

 [0.77381111 0.59732787 0.2036345
0.05430253]

 [0.79428944 0.57365349 0.19121783
0.05883625]

 [0.80327412 0.55126656 0.22050662
0.04725142]

 [0.8068282  0.53788547 0.24063297
0.04246464]

 [0.77964883 0.58091482 0.22930848
0.0458617 ]

 [0.8173379  0.51462016 0.25731008
0.03027177]

 [0.78591858 0.57017622 0.23115252
0.06164067]

 [0.77577075 0.60712493 0.16864581
0.03372916]

 [0.80597792 0.52151512 0.26865931
0.07901744]

 [0.776114   0.54974742 0.30721179
0.03233808]

 [0.82647451 0.4958847  0.26447184
0.03305898]
```

```
 [0.79778206 0.5424918  0.25529026
0.06382256]

 [0.80641965 0.54278246 0.23262105
0.03101614]

 [0.81609427 0.5336001  0.21971769
0.03138824]

 [0.79524064 0.54144043 0.27072022
0.03384003]

 [0.80846584 0.52213419 0.26948861
0.03368608]

 [0.82225028 0.51771314 0.22840286
0.06090743]

 [0.76578311 0.60379053 0.22089897
0.0147266 ]

 [0.77867447 0.59462414 0.19820805
0.02831544]

 [0.81768942 0.51731371 0.25031309
0.03337508]

 [0.82512295 0.52807869 0.19802951
0.03300492]

 [0.82699754 0.52627116 0.19547215
0.03007264]

 [0.78523221 0.5769053  0.22435206
0.01602515]

 [0.80212413 0.54690282 0.23699122
0.03646019]

 [0.80779568 0.53853046 0.23758697
0.03167826]

 [0.80033301 0.56023311 0.20808658
0.04801998]

 [0.86093857 0.44003527 0.24871559
0.0573959 ]

 [0.78609038 0.57170209 0.23225397
0.03573138]

 [0.78889479 0.55222635 0.25244633
0.09466737]
```

```
 [0.76693897 0.57144472 0.28572236
0.06015208]

 [0.82210585 0.51381615 0.23978087
0.05138162]

 [0.77729093 0.57915795 0.24385598
0.030482  ]

 [0.79594782 0.55370283 0.24224499
0.03460643]

 [0.79837025 0.55735281 0.22595384
0.03012718]

 [0.81228363 0.5361072  0.22743942
0.03249135]

 [0.76701103 0.35063361 0.51499312
0.15340221]

 [0.74549757 0.37274878 0.52417798
0.17472599]

 [0.75519285 0.33928954 0.53629637
0.16417236]

 [0.75384916 0.31524601 0.54825394
0.17818253]

 [0.7581754  0.32659863 0.5365549
0.17496355]

 [0.72232962 0.35482858 0.57026022
0.16474184]

 [0.72634846 0.38046824 0.54187901
0.18446945]

 [0.75916547 0.37183615 0.51127471
0.15493173]

 [0.76301853 0.33526572 0.53180079
0.15029153]

 [0.72460233 0.37623583 0.54345175
0.19508524]

 [0.76923077 0.30769231 0.53846154
0.15384615]

 [0.73923462 0.37588201 0.52623481
0.187941  ]
```

```
 [0.78892752 0.28927343 0.52595168
0.13148792]

 [0.73081412 0.34743622 0.56308629
0.16772783]

 [0.75911707 0.3931142  0.48800383
0.17622361]

 [0.76945444 0.35601624 0.50531337
0.16078153]

 [0.70631892 0.37838513 0.5675777
0.18919257]

 [0.75676497 0.35228714 0.53495455
0.13047672]

 [0.76444238 0.27125375 0.55483721
0.18494574]

 [0.76185188 0.34011245 0.53057542
0.14964948]

 [0.6985796  0.37889063 0.56833595
0.21312598]

 [0.77011854 0.35349703 0.50499576
0.16412362]

 [0.74143307 0.29421947 0.57667016
0.17653168]

 [0.73659895 0.33811099 0.56754345
0.14490471]

 [0.76741698 0.34773582 0.51560829
0.15588157]

 [0.76785726 0.34902603 0.51190484
0.16287881]

 [0.76467269 0.31486523 0.53976896
0.15743261]

 [0.74088576 0.33173989 0.55289982
0.18798594]

 [0.73350949 0.35452959 0.55013212
0.18337737]

 [0.78667474 0.35883409 0.48304589
0.13801311]
```

```
 [0.76521855 0.33391355 0.52869645
0.15304371]

 [0.77242925 0.33706004 0.51963422
0.14044168]

 [0.76434981 0.35581802 0.51395936
0.15814134]

 [0.70779525 0.31850786 0.60162596
0.1887454 ]

 [0.69333409 0.38518561 0.57777841
0.1925928 ]

 [0.71524936 0.40530797 0.53643702
0.19073316]

 [0.75457341 0.34913098 0.52932761
0.16893434]

 [0.77530021 0.28304611 0.54147951
0.15998258]

 [0.72992443 0.39103094 0.53440896
0.16944674]

 [0.74714194 0.33960997 0.54337595
0.17659719]

 [0.72337118 0.34195729 0.57869695
0.15782644]

 [0.73260391 0.36029701 0.55245541
0.1681386 ]

 [0.76262994 0.34186859 0.52595168
0.1577855 ]

 [0.76986879 0.35413965 0.5081134
0.15397376]

 [0.73544284 0.35458851 0.55158213
0.1707278 ]

 [0.73239618 0.38547167 0.53966034
0.15418867]

 [0.73446047 0.37367287 0.5411814
0.16750853]

 [0.75728103 0.3542121  0.52521104
0.15878473]
```

```
 [0.78258054 0.38361791 0.4603415
0.16879188]

 [0.7431482  0.36505526 0.5345452
0.16948994]

 [0.65387747 0.34250725 0.62274045
0.25947519]

 [0.69052512 0.32145135 0.60718588
0.22620651]

 [0.71491405 0.30207636 0.59408351
0.21145345]

 [0.69276796 0.31889319 0.61579374
0.1979337 ]

 [0.68619022 0.31670318 0.61229281
0.232249  ]

 [0.70953708 0.28008043 0.61617694
0.1960563 ]

 [0.67054118 0.34211284 0.61580312
0.23263673]

 [0.71366557 0.28351098 0.61590317
0.17597233]

 [0.71414125 0.26647062 0.61821183
0.19185884]

 [0.69198788 0.34599394 0.58626751
0.24027357]

 [0.71562645 0.3523084  0.56149152
0.22019275]

 [0.71576546 0.30196356 0.59274328
0.21249287]

 [0.71718148 0.31640359 0.58007326
0.22148252]

 [0.6925518  0.30375079 0.60750157
0.24300063]

 [0.67767924 0.32715549 0.59589036
0.28041899]

 [0.69589887 0.34794944 0.57629125
0.25008866]

 [0.70610474 0.3258945  0.59747324
0.1955367 ]

 [0.69299099 0.34199555 0.60299216
0.19799743]

 [0.70600618 0.2383917  0.63265489
0.21088496]

 [0.72712585 0.26661281 0.60593821
0.18178146]

 [0.70558934 0.32722984 0.58287815
0.23519645]

 [0.68307923 0.34153961 0.59769433
0.24395687]

 [0.71486543 0.25995106 0.62202576
0.18567933]

 [0.73122464 0.31338199 0.56873028
0.20892133]

 [0.69595601 0.3427843  0.59208198
0.21813547]

 [0.71529453 0.31790868 0.59607878
0.17882363]

 [0.72785195 0.32870733 0.56349829
0.21131186]

 [0.71171214 0.35002236 0.57170319
0.21001342]

 [0.69594002 0.30447376 0.60894751
0.22835532]

 [0.73089855 0.30454106 0.58877939
0.1624219 ]

 [0.72766159 0.27533141 0.59982915
0.18683203]

 [0.71578999 0.34430405 0.5798805
0.18121266]

 [0.69417747 0.30370264 0.60740528
0.2386235 ]

 [0.72366005 0.32162669 0.58582004
0.17230001]
```

```
 [0.69385414 0.29574111 0.63698085
0.15924521]

 [0.73154399 0.28501714 0.57953485
0.21851314]

 [0.67017484 0.36168166 0.59571097
0.2553047 ]

 [0.69804799 0.338117   0.59988499 0.196326  ]

 [0.71066905 0.35533453 0.56853524
0.21320072]

 [0.72415258 0.32534391 0.56672811
0.22039426]

 [0.69997037 0.32386689 0.58504986
0.25073566]

 [0.73337886 0.32948905 0.54206264
0.24445962]
```

```
 [0.69052512 0.32145135 0.60718588
0.22620651]

 [0.69193502 0.32561648 0.60035539
0.23403685]

 [0.68914871 0.33943145 0.58629069
0.25714504]

 [0.72155725 0.32308533 0.56001458
0.24769876]

 [0.72965359 0.28954508 0.57909015
0.22005426]

 [0.71653899 0.3307103  0.57323119
0.22047353]

 [0.67467072 0.36998072 0.58761643
0.25028107]

 [0.69025916 0.35097923 0.5966647
0.21058754]]
```

Data Standardization:

```
# Standardize the data attributes for the Iris dataset.

from sklearn.datasets import load_iris

from sklearn import preprocessing

# load the Iris dataset

iris = load_iris()

print(iris.data.shape)

# separate the data and target attributes

X = iris.data

y = iris.target

# standardize the data attributes

standardized_X = preprocessing.scale(X)

print(standardized_X)
```

Output:

```
(150, 4)

[[-9.00681170e-01  1.01900435e+00 -1.34022653e+00 -1.31544430e+00]

 [-1.14301691e+00 -1.31979479e-01 -1.34022653e+00 -1.31544430e+00]
```

```
[-1.38535265e+00  3.28414053e-01 -1.39706395e+00 -1.31544430e+00]
[-1.50652052e+00  9.82172869e-02 -1.28338910e+00 -1.31544430e+00]
[-1.02184904e+00  1.24920112e+00 -1.34022653e+00 -1.31544430e+00]
[-5.37177559e-01  1.93979142e+00 -1.16971425e+00 -1.05217993e+00]
[-1.50652052e+00  7.88807586e-01 -1.34022653e+00 -1.18381211e+00]
[-1.02184904e+00  7.88807586e-01 -1.28338910e+00 -1.31544430e+00]
[-1.74885626e+00 -3.62176246e-01 -1.34022653e+00 -1.31544430e+00]
[-1.14301691e+00  9.82172869e-02 -1.28338910e+00 -1.44707648e+00]
[-5.37177559e-01  1.47939788e+00 -1.28338910e+00 -1.31544430e+00]
[-1.26418478e+00  7.88807586e-01 -1.22655167e+00 -1.31544430e+00]
[-1.26418478e+00 -1.31979479e-01 -1.34022653e+00 -1.44707648e+00]
[-1.87002413e+00 -1.31979479e-01 -1.51073881e+00 -1.44707648e+00]
[-5.25060772e-02  2.16998818e+00 -1.45390138e+00 -1.31544430e+00]
[-1.73673948e-01  3.09077525e+00 -1.28338910e+00 -1.05217993e+00]
[-5.37177559e-01  1.93979142e+00 -1.39706395e+00 -1.05217993e+00]
[-9.00681170e-01  1.01900435e+00 -1.34022653e+00 -1.18381211e+00]
[-1.73673948e-01  1.70959465e+00 -1.16971425e+00 -1.18381211e+00]
[-9.00681170e-01  1.70959465e+00 -1.28338910e+00 -1.18381211e+00]
[-5.37177559e-01  7.88807586e-01 -1.16971425e+00 -1.31544430e+00]
[-9.00681170e-01  1.47939788e+00 -1.28338910e+00 -1.05217993e+00]
[-1.50652052e+00  1.24920112e+00 -1.56757623e+00 -1.31544430e+00]
[-9.00681170e-01  5.58610819e-01 -1.16971425e+00 -9.20547742e-01]
[-1.26418478e+00  7.88807586e-01 -1.05603939e+00 -1.31544430e+00]
[-1.02184904e+00 -1.31979479e-01 -1.22655167e+00 -1.31544430e+00]
[-1.02184904e+00  7.88807586e-01 -1.22655167e+00 -1.05217993e+00]
[-7.79513300e-01  1.01900435e+00 -1.28338910e+00 -1.31544430e+00]
[-7.79513300e-01  7.88807586e-01 -1.34022653e+00 -1.31544430e+00]
[-1.38535265e+00  3.28414053e-01 -1.22655167e+00 -1.31544430e+00]
[-1.26418478e+00  9.82172869e-02 -1.22655167e+00 -1.31544430e+00]
```

```
[-5.37177559e-01  7.88807586e-01 -1.28338910e+00 -1.05217993e+00]
[-7.79513300e-01  2.40018495e+00 -1.28338910e+00 -1.44707648e+00]
[-4.16009689e-01  2.63038172e+00 -1.34022653e+00 -1.31544430e+00]
[-1.14301691e+00  9.82172869e-02 -1.28338910e+00 -1.31544430e+00]
[-1.02184904e+00  3.28414053e-01 -1.45390138e+00 -1.31544430e+00]
[-4.16009689e-01  1.01900435e+00 -1.39706395e+00 -1.31544430e+00]
[-1.14301691e+00  1.24920112e+00 -1.34022653e+00 -1.44707648e+00]
[-1.74885626e+00 -1.31979479e-01 -1.39706395e+00 -1.31544430e+00]
[-9.00681170e-01  7.88807586e-01 -1.28338910e+00 -1.31544430e+00]
[-1.02184904e+00  1.01900435e+00 -1.39706395e+00 -1.18381211e+00]
[-1.62768839e+00 -1.74335684e+00 -1.39706395e+00 -1.18381211e+00]
[-1.74885626e+00  3.28414053e-01 -1.39706395e+00 -1.31544430e+00]
[-1.02184904e+00  1.01900435e+00 -1.22655167e+00 -7.88915558e-01]
[-9.00681170e-01  1.70959465e+00 -1.05603939e+00 -1.05217993e+00]
[-1.26418478e+00 -1.31979479e-01 -1.34022653e+00 -1.18381211e+00]
[-9.00681170e-01  1.70959465e+00 -1.22655167e+00 -1.31544430e+00]
[-1.50652052e+00  3.28414053e-01 -1.34022653e+00 -1.31544430e+00]
[-6.58345429e-01  1.47939788e+00 -1.28338910e+00 -1.31544430e+00]
[-1.02184904e+00  5.58610819e-01 -1.34022653e+00 -1.31544430e+00]
[ 1.40150837e+00  3.28414053e-01  5.35408562e-01  2.64141916e-01]
[ 6.74501145e-01  3.28414053e-01  4.21733708e-01  3.95774101e-01]
[ 1.28034050e+00  9.82172869e-02  6.49083415e-01  3.95774101e-01]
[-4.16009689e-01 -1.74335684e+00  1.37546573e-01  1.32509732e-01]
[ 7.95669016e-01 -5.92373012e-01  4.78571135e-01  3.95774101e-01]
[-1.73673948e-01 -5.92373012e-01  4.21733708e-01  1.32509732e-01]
[ 5.53333275e-01  5.58610819e-01  5.35408562e-01  5.27406285e-01]
[-1.14301691e+00 -1.51316008e+00 -2.60315415e-01 -2.62386821e-01]
[ 9.16836886e-01 -3.62176246e-01  4.78571135e-01  1.32509732e-01]
[-7.79513300e-01 -8.22569778e-01  8.07091462e-02  2.64141916e-01]
```

```
[-1.02184904e+00 -2.43394714e+00 -1.46640561e-01 -2.62386821e-01]
[ 6.86617933e-02 -1.31979479e-01  2.51221427e-01  3.95774101e-01]
[ 1.89829664e-01 -1.97355361e+00  1.37546573e-01 -2.62386821e-01]
[ 3.10997534e-01 -3.62176246e-01  5.35408562e-01  2.64141916e-01]
[-2.94841818e-01 -3.62176246e-01 -8.98031345e-02  1.32509732e-01]
[ 1.03800476e+00  9.82172869e-02  3.64896281e-01  2.64141916e-01]
[-2.94841818e-01 -1.31979479e-01  4.21733708e-01  3.95774101e-01]
[-5.25060772e-02 -8.22569778e-01  1.94384000e-01 -2.62386821e-01]
[ 4.32165405e-01 -1.97355361e+00  4.21733708e-01  3.95774101e-01]
[-2.94841818e-01 -1.28296331e+00  8.07091462e-02 -1.30754636e-01]
[ 6.86617933e-02  3.28414053e-01  5.92245988e-01  7.90670654e-01]
[ 3.10997534e-01 -5.92373012e-01  1.37546573e-01  1.32509732e-01]
[ 5.53333275e-01 -1.28296331e+00  6.49083415e-01  3.95774101e-01]
[ 3.10997534e-01 -5.92373012e-01  5.35408562e-01  8.77547895e-04]
[ 6.74501145e-01 -3.62176246e-01  3.08058854e-01  1.32509732e-01]
[ 9.16836886e-01 -1.31979479e-01  3.64896281e-01  2.64141916e-01]
[ 1.15917263e+00 -5.92373012e-01  5.92245988e-01  2.64141916e-01]
[ 1.03800476e+00 -1.31979479e-01  7.05920842e-01  6.59038469e-01]
[ 1.89829664e-01 -3.62176246e-01  4.21733708e-01  3.95774101e-01]
[-1.73673948e-01 -1.05276654e+00 -1.46640561e-01 -2.62386821e-01]
[-4.16009689e-01 -1.51316008e+00  2.38717193e-02 -1.30754636e-01]
[-4.16009689e-01 -1.51316008e+00 -3.29657076e-02 -2.62386821e-01]
[-5.25060772e-02 -8.22569778e-01  8.07091462e-02  8.77547895e-04]
[ 1.89829664e-01 -8.22569778e-01  7.62758269e-01  5.27406285e-01]
[-5.37177559e-01 -1.31979479e-01  4.21733708e-01  3.95774101e-01]
[ 1.89829664e-01  7.88807586e-01  4.21733708e-01  5.27406285e-01]
[ 1.03800476e+00  9.82172869e-02  5.35408562e-01  3.95774101e-01]
[ 5.53333275e-01 -1.74335684e+00  3.64896281e-01  1.32509732e-01]
[-2.94841818e-01 -1.31979479e-01  1.94384000e-01  1.32509732e-01]
```

```
[-4.16009689e-01 -1.28296331e+00  1.37546573e-01  1.32509732e-01]
[-4.16009689e-01 -1.05276654e+00  3.64896281e-01  8.77547895e-04]
[ 3.10997534e-01 -1.31979479e-01  4.78571135e-01  2.64141916e-01]
[-5.25060772e-02 -1.05276654e+00  1.37546573e-01  8.77547895e-04]
[-1.02184904e+00 -1.74335684e+00 -2.60315415e-01 -2.62386821e-01]
[-2.94841818e-01 -8.22569778e-01  2.51221427e-01  1.32509732e-01]
[-1.73673948e-01 -1.31979479e-01  2.51221427e-01  8.77547895e-04]
[-1.73673948e-01 -3.62176246e-01  2.51221427e-01  1.32509732e-01]
[ 4.32165405e-01 -3.62176246e-01  3.08058854e-01  1.32509732e-01]
[-9.00681170e-01 -1.28296331e+00 -4.30827696e-01 -1.30754636e-01]
[-1.73673948e-01 -5.92373012e-01  1.94384000e-01  1.32509732e-01]
[ 5.53333275e-01  5.58610819e-01  1.27429511e+00  1.71209594e+00]
[-5.25060772e-02 -8.22569778e-01  7.62758269e-01  9.22302838e-01]
[ 1.52267624e+00 -1.31979479e-01  1.21745768e+00  1.18556721e+00]
[ 5.53333275e-01 -3.62176246e-01  1.04694540e+00  7.90670654e-01]
[ 7.95669016e-01 -1.31979479e-01  1.16062026e+00  1.31719939e+00]
[ 2.12851559e+00 -1.31979479e-01  1.61531967e+00  1.18556721e+00]
[-1.14301691e+00 -1.28296331e+00  4.21733708e-01  6.59038469e-01]
[ 1.76501198e+00 -3.62176246e-01  1.44480739e+00  7.90670654e-01]
[ 1.03800476e+00 -1.28296331e+00  1.16062026e+00  7.90670654e-01]
[ 1.64384411e+00  1.24920112e+00  1.33113254e+00  1.71209594e+00]
[ 7.95669016e-01  3.28414053e-01  7.62758269e-01  1.05393502e+00]
[ 6.74501145e-01 -8.22569778e-01  8.76433123e-01  9.22302838e-01]
[ 1.15917263e+00 -1.31979479e-01  9.90107977e-01  1.18556721e+00]
[-1.73673948e-01 -1.28296331e+00  7.05920842e-01  1.05393502e+00]
[-5.25060772e-02 -5.92373012e-01  7.62758269e-01  1.58046376e+00]
[ 6.74501145e-01  3.28414053e-01  8.76433123e-01  1.44883158e+00]
[ 7.95669016e-01 -1.31979479e-01  9.90107977e-01  7.90670654e-01]
[ 2.24968346e+00  1.70959465e+00  1.67215710e+00  1.31719939e+00]
[ 2.24968346e+00 -1.05276654e+00  1.78583195e+00  1.44883158e+00]
[ 1.89829664e-01 -1.97355361e+00  7.05920842e-01  3.95774101e-01]
[ 1.28034050e+00  3.28414053e-01  1.10378283e+00  1.44883158e+00]
[-2.94841818e-01 -5.92373012e-01  6.49083415e-01  1.05393502e+00]
[ 2.24968346e+00 -5.92373012e-01  1.67215710e+00  1.05393502e+00]
[ 5.53333275e-01 -8.22569778e-01  6.49083415e-01  7.90670654e-01]
[ 1.03800476e+00  5.58610819e-01  1.10378283e+00  1.18556721e+00]
[ 1.64384411e+00  3.28414053e-01  1.27429511e+00  7.90670654e-01]
[ 4.32165405e-01 -5.92373012e-01  5.92245988e-01  7.90670654e-01]
[ 3.10997534e-01 -1.31979479e-01  6.49083415e-01  7.90670654e-01]
[ 6.74501145e-01 -5.92373012e-01  1.04694540e+00  1.18556721e+00]
[ 1.64384411e+00 -1.31979479e-01  1.16062026e+00  5.27406285e-01]
[ 1.88617985e+00 -5.92373012e-01  1.33113254e+00  9.22302838e-01]
[ 2.49201920e+00  1.70959465e+00  1.50164482e+00  1.05393502e+00]
[ 6.74501145e-01 -5.92373012e-01  1.04694540e+00  1.31719939e+00]
[ 5.53333275e-01 -5.92373012e-01  7.62758269e-01  3.95774101e-01]
[ 3.10997534e-01 -1.05276654e+00  1.04694540e+00  2.64141916e-01]
[ 2.24968346e+00 -1.31979479e-01  1.33113254e+00  1.44883158e+00]
[ 5.53333275e-01  7.88807586e-01  1.04694540e+00  1.58046376e+00]
[ 6.74501145e-01  9.82172869e-02  9.90107977e-01  7.90670654e-01]
[ 1.89829664e-01 -1.31979479e-01  5.92245988e-01  7.90670654e-01]
[ 1.28034050e+00  9.82172869e-02  9.33270550e-01  1.18556721e+00]
[ 1.03800476e+00  9.82172869e-02  1.04694540e+00  1.58046376e+00]
[ 1.28034050e+00  9.82172869e-02  7.62758269e-01  1.44883158e+00]
[-5.25060772e-02 -8.22569778e-01  7.62758269e-01  9.22302838e-01]
[ 1.15917263e+00  3.28414053e-01  1.21745768e+00  1.44883158e+00]
[ 1.03800476e+00  5.58610819e-01  1.10378283e+00  1.71209594e+00]
[ 1.03800476e+00 -1.31979479e-01  8.19595696e-01  1.44883158e+00]
[ 5.53333275e-01 -1.28296331e+00  7.05920842e-01  9.22302838e-01]
```

```
 [ 7.95669016e-01 -1.31979479e-01  8.19595696e-01  1.05393502e+00]
 [ 4.32165405e-01  7.88807586e-01  9.33270550e-01  1.44883158e+00]
 [ 6.86617933e-02 -1.31979479e-01  7.62758269e-01  7.90670654e-01]]
```

2. Standardize IRIS DataSet:

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
std_slc = StandardScaler()
std_slc.fit(X_train)
X_train_std = std_slc.transform(X_train)
X_test_std = std_slc.transform(X_test)
print(X_train[0:5])
print(X_train_std[0:5])
print(X_test[0:5])
print(X_test_std[0:5])
```

Output:

```
[[5.1 3.5 1.4 0.3]
 [5.7 3.  4.2 1.2]
 [6.3 2.9 5.6 1.8]
 [6.1 2.8 4.7 1.2]
 [5.6 2.8 4.9 2. ]]
[[-0.84747571  1.09946882 -1.27248426 -1.12920679]
 [-0.12002875 -0.13036208  0.2977916   0.05114874]
 [ 0.60741821 -0.37632826  1.08292953  0.83805243]
 [ 0.36493589 -0.62229444  0.578198    0.05114874]
 [-0.24126991 -0.62229444  0.69036056  1.10035366]]
[[6.3 3.3 6.  2.5]
 [6.6 3.  4.4 1.4]
 [6.3 2.8 5.1 1.5]
 [4.6 3.2 1.4 0.2]
 [6.7 3.  5.  1.7]]
[[ 0.60741821  0.60753646  1.30725465  1.75610673]
 [ 0.97114169 -0.13036208  0.40995416  0.31344997]
 [ 0.60741821 -0.62229444  0.80252312  0.44460058]
 [-1.45368151  0.36157028 -1.27248426 -1.26035741]
 [ 1.09238285 -0.13036208  0.74644184  0.70690181]]
```

Result

Thus, data pre-processing techniques in machine learning have been successfully implemented.

Ex No. 3	Feature Selection Techniques in Machine Learning
Date:	

Aim

To implement feature subset selection techniques in machine learning.

Definition

Feature selection

Feature Selection is the method of reducing the input variable to your model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

1. Univariate Feature Selection:

univariate_selection.py

```
print(__doc__)

import numpy as np

import matplotlib.pyplot as plt


from sklearn import datasets, svm

from sklearn.feature_selection import SelectPercentile, f_classif

iris = datasets.load_iris()

E = np.random.uniform(0, 0.1, size=(len(iris.data), 20))

X = np.hstack((iris.data, E))

y = iris.target

plt.figure(1)

plt.clf()
```

```
X_indices = np.arange(X.shape[-1])
selector = SelectPercentile(f_classif, percentile=10)
selector.fit(X, y)
scores = -np.log10(selector.pvalues_)
scores /= scores.max()
plt.bar(X_indices - .45, scores, width=.2,
label=r'Univariate score ($-Log(p_{value})$)', color='g')
clf = svm.SVC(kernel='linear')
clf.fit(X, y)
svm_weights = (clf.coef_ ** 2).sum(axis=0)
svm_weights /= svm_weights.max()
plt.bar(X_indices - .25, svm_weights, width=.2, label='SVM weight', color='r')
clf_selected = svm.SVC(kernel='linear')
clf_selected.fit(selector.transform(X), y)
svm_weights_selected = (clf_selected.coef_ ** 2).sum(axis=0)
svm_weights_selected /= svm_weights_selected.max()
plt.bar(X_indices[selector.get_support()] - .05, svm_weights_selected, width=.2, label='SVM weights after selection', color='b')
plt.title("Comparing feature selection")
plt.xlabel('Feature number')
plt.yticks(())
plt.axis('tight')
plt.legend(loc='upper right')
plt.show()
```

Output

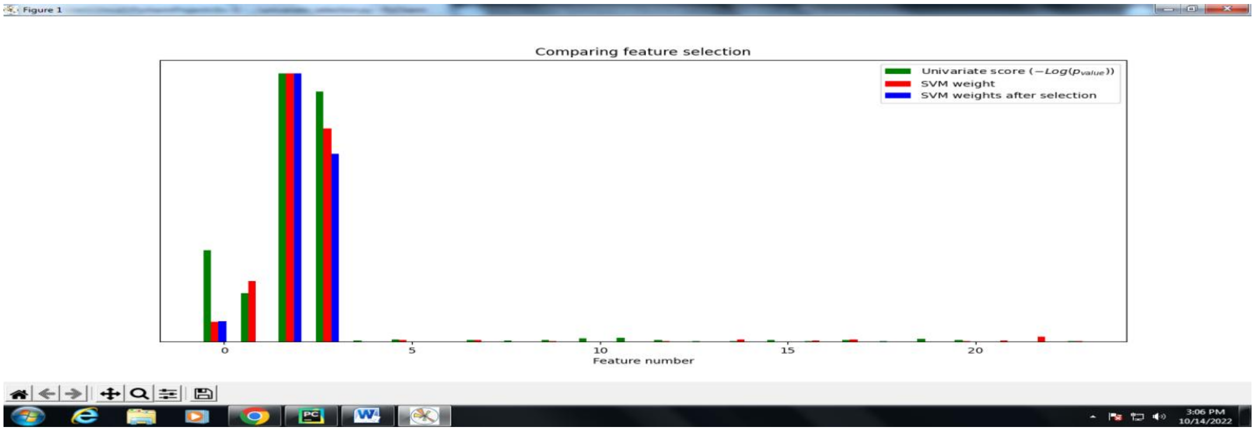


Figure 1
Comparing feature selection
Univariate score ($-Log(p_{value})$)
SVM weight
SVM weights after selection
0
5
10
15
20
Feature number
3:06 PM
10/14/2022

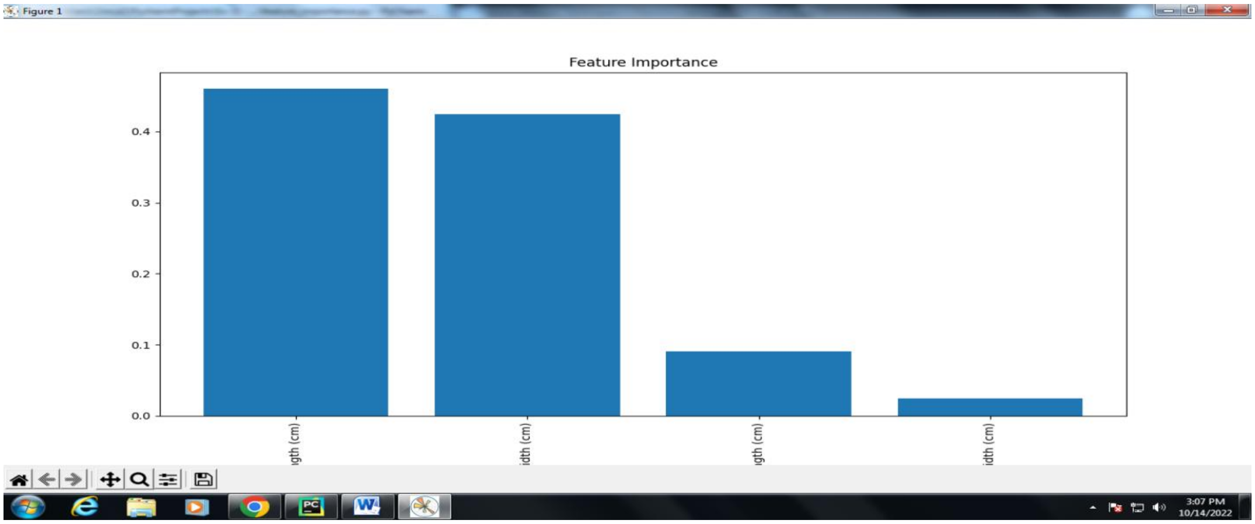
2. Feature Importance:

feature_importance.py

```
# Load libraries

from sklearn.ensemble import RandomForestClassifier

from sklearn import datasets

import numpy as np

import matplotlib.pyplot as plt

# Load data

iris = datasets.load_iris()

X = iris.data

y = iris.target

# Create decision tree classifer object

clf = RandomForestClassifier(random_state=0, n_jobs=-1)

# Train model

model = clf.fit(X, y)

# Calculate feature importances

importances = model.feature_importances_

# Sort feature importances in descending order

indices = np.argsort(importances)[::-1]

# Rearrange feature names so they match the sorted feature importances
```

```
names = [iris.feature_names[i] for i in indices]

# Create plot

plt.figure()

# Create plot title

plt.title("Feature Importance")

# Add bars

plt.bar(range(X.shape[1]), importances[indices])

# Add feature names as x-axis labels

plt.xticks(range(X.shape[1]), names, rotation=90)

# Show plot

plt.show()
```

Output



3. Correlation Matrix with Heatmap:

heatmap.py

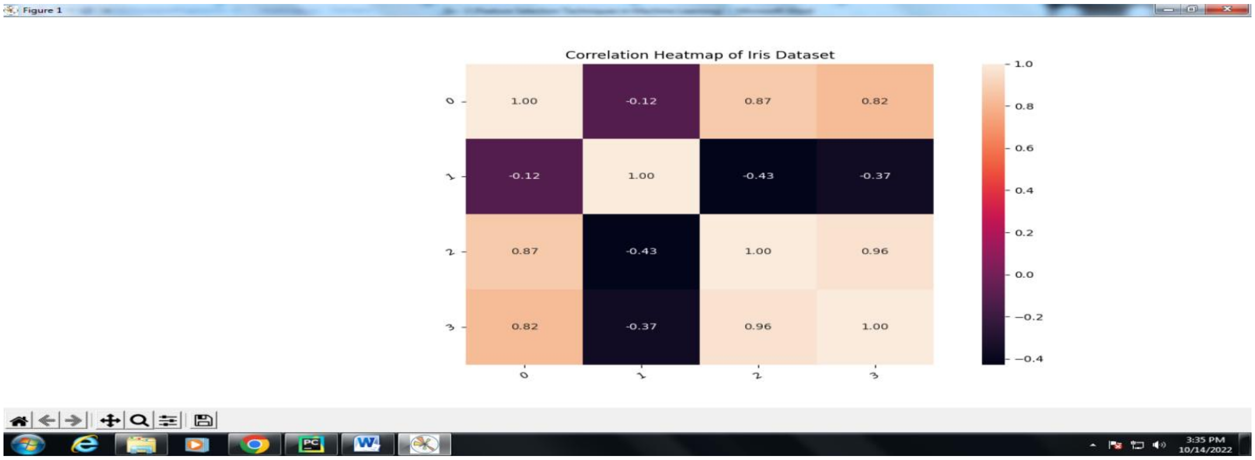
```
# Load iris data
from sklearn.datasets import load_iris
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

iris = load_iris()

# Create features and target
X = iris.data
```

```
y = iris.target
# Convert feature matrix into DataFrame
df = pd.DataFrame(X)

# View the data frame
print(df)
# Create correlation matrix
corr_matrix = df.corr()
print(corr_matrix)
# Create correlation heatmap
plt.figure(figsize=(8,6))
plt.title('Correlation Heatmap of Iris Dataset')
a = sns.heatmap(corr_matrix, square=True, annot=True, fmt='.2f', linecolor='black')
a.set_xticklabels(a.get_xticklabels(), rotation=30)
a.set_yticklabels(a.get_yticklabels(), rotation=30)
plt.show()
 # Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
# Find index of feature columns with correlation greater than 0.9
to_drop = [column for column in upper.columns if any(upper[column] > 0.9)]
print(to_drop)
# Drop Marked Features
df1 = df.drop(df.columns[to_drop], axis=1)
print(df1)
```

Output

```
C:\Users\2mca1\PycharmProjects\Ex-3\venv\Scripts\python.exe C:/Users/2mca1/PycharmProjects/Ex-3/heatmap.py

       0    1    2    3

0    5.1  3.5  1.4  0.2

1    4.9  3.0  1.4  0.2

2    4.7  3.2  1.3  0.2

3    4.6  3.1  1.5  0.2

4    5.0  3.6  1.4  0.2

..   ...  ...  ...  ...

145  6.7  3.0  5.2  2.3

146  6.3  2.5  5.0  1.9

147  6.5  3.0  5.2  2.0

148  6.2  3.4  5.4  2.3

149  5.9  3.0  5.1  1.8
```

```
[150 rows x 4 columns]
          0         1         2         3
0  1.000000 -0.117570  0.871754  0.817941
1 -0.117570  1.000000 -0.428440 -0.366126
2  0.871754 -0.428440  1.000000  0.962865
3  0.817941 -0.366126  0.962865  1.000000
```



Result

Thus, feature subset selection techniques have been implemented successfully.

Ex No. 4	**Measuring the Performance of Machine Learning Model**
Date:	

Aim

To measure the performance of machine learning models in python.

Definition

Accuracy

Accuracy is what its literal meaning says, a measure of how accurate your model is.

Accuracy = Correct Predictions / Total Predictions
By using confusion matrix, Accuracy = (TP + TN)/(TP+TN+FP+FN)
Precision & Recall

Precision: It is the ratio of True Positives (TP) and the total positive predictions. Basically, it tells us how many times your positive prediction was actually positive.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall : It is nothing but TPR (True Positive Rate explained above). It tells us about out of all the positive points how many were predicted positive.

$$\text{Recall} = \frac{tp}{tp + fn}$$

F-Measure: Harmonic mean of precision and recall.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Confusion Matrix

A **confusion matrix** is a correlation between the predictions of a model and the actual class labels of the data points.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

1. **R – Squared**
   Rsquared.py

```
from sklearn.metrics import r2_score

### Assume y is the actual value and f is the predicted values
y =[10, 20, 30]
f =[10, 20, 30]
r2 = r2_score(y, f)
print('r2 score for perfect model is', r2)

### Assume y is the actual value and f is the predicted values
y =[10, 20, 30]
f =[20, 20, 20]
r2 = r2_score(y, f)
print('r2 score for a model which predicts mean value always is', r2)

### Assume y is the actual value and f is the predicted values
y = [10, 20, 30]
f = [30, 10, 20]
r2 = r2_score(y, f)
print('r2 score for a worse model is', r2)
```

Output

```
r2 score for perfect model is 1.0
r2 score for a model which predicts mean value always is 0.0
r2 score for a worse model is -2.0
```

2. **Adjusted R-Squared**
   adjRsquared.py

```
from sklearn.linear_model import LinearRegression
import pandas as pd

#define URL where dataset is located
url = "https://raw.githubusercontent.com/Statology/Python-Guides/main/mtcars.csv"

#read in data
data = pd.read_csv(url)

#fit regression model
model = LinearRegression()
X, y = data[["mpg", "wt", "drat", "qsec"]], data.hp
model.fit(X, y)

#display adjusted R-squared
print(1 - (1-model.score(X, y))*(len(y)-1)/(len(y)-X.shape[1]-1))
```

Output

```
0.7787005290062519
```

3. **Mean Absolute Error**
   MeanAbsError.py

```
# Python program for calculating Mean Absolute
# Error using sklearn

# import the module
from sklearn.metrics import mean_absolute_error as mae

# list of integers of actual and calculated
actual = [2, 3, 5, 5, 9]
calculated = [3, 3, 8, 7, 6]

# calculate MAE
error = mae(actual, calculated)

# display
print("Mean absolute error : " + str(error))
```

Output

```
Mean absolute error : 1.8
```

4. **Mean Squared Error**
   MeanSqaError.py

```
from sklearn.metrics import mean_squared_error

# Given values
Y_true = [1,1,2,2,4] # Y_true = Y (original values)

# calculated values
Y_pred = [0.6,1.29,1.99,2.69,3.4] # Y_pred = Y'

# Calculation of Mean Squared Error (MSE)
MSE = mean_squared_error(Y_true,Y_pred)

print(MSE)
```

Output

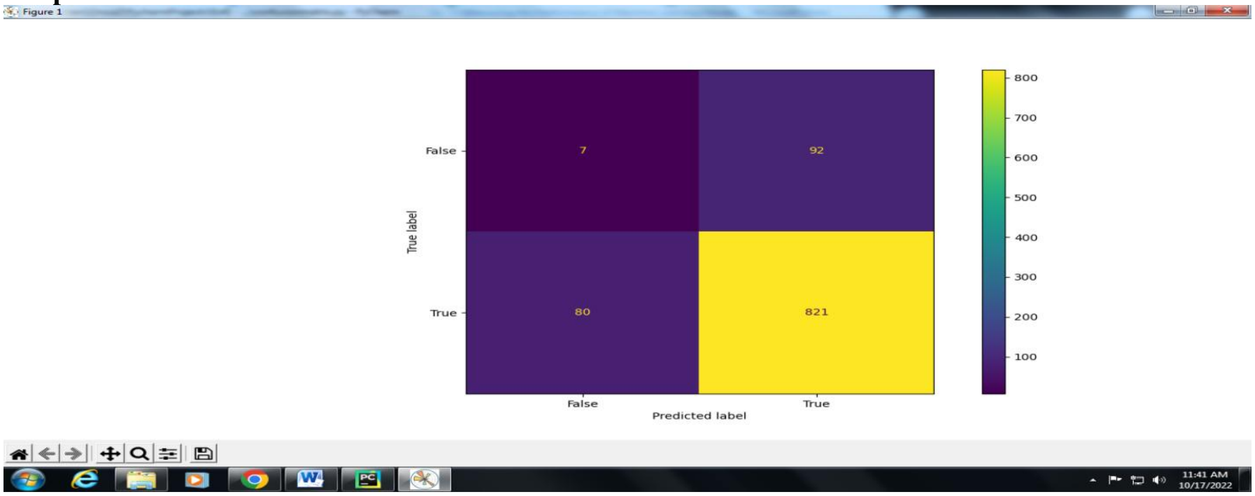
```
0.21606
```

5. **Confusion Matrix and Related Metrics**
   confusionmatrix.py

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

actual = numpy.random.binomial(1,.9,size = 1000)
predicted = numpy.random.binomial(1,.9,size = 1000)

confusion_matrix = metrics.confusion_matrix(actual, predicted)
```

```
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels =
[False, True])

cm_display.plot()
plt.show()
```

Output



relatedmetrics.py

```
import numpy
from sklearn import metrics

actual = numpy.random.binomial(1,.9,size = 1000)
predicted = numpy.random.binomial(1,.9,size = 1000)

Accuracy = metrics.accuracy_score(actual, predicted)
Precision = metrics.precision_score(actual, predicted)
Sensitivity_recall = metrics.recall_score(actual, predicted)
Specificity = metrics.recall_score(actual, predicted, pos_label=0)
F1_score = metrics.f1_score(actual, predicted)

#metrics:
print({"Accuracy":Accuracy,"Precision":Precision,"Sensitivity_recall":Sensitivity_recall,"Specificity":Sp
ecificity,"F1_score":F1_score})
```

Output

```
{'Accuracy': 0.836, 'Precision': 0.9074889867841409, 'Sensitivity_recall': 0.911504424778761,
'Specificity': 0.125, 'F1_score': 0.9094922737306844}
```

6. F1-Score

F1score.py

```
import numpy as np
from sklearn.metrics import f1_score
```

```
#define array of actual classes
actual = np.repeat([1, 0], repeats=[160, 240])

#define array of predicted classes
pred = np.repeat([1, 0, 1, 0], repeats=[120, 40, 70, 170])

#calculate F1 score
print(f1_score(actual, pred))
```

Output

```
0.6857142857142857
```

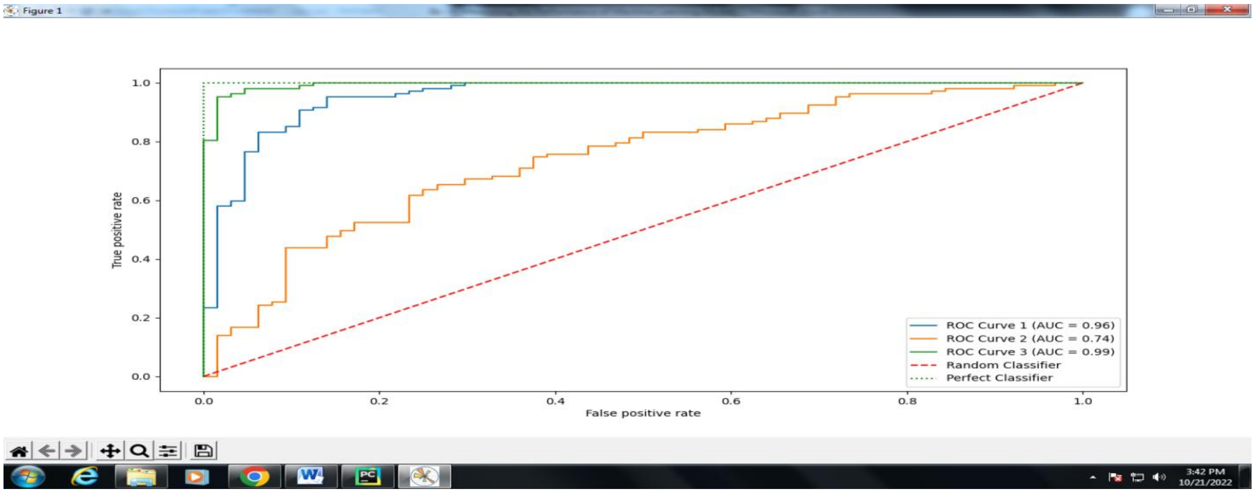
7. AUC-ROC Curve

AUCROC.py

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.pipeline import make_pipeline

#
# Load the breast cancer data set
#
bc = datasets.load_breast_cancer()
X, y = bc.data, bc.target
#
# Create training and test split
#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1, stratify=y)
#
# Create the estimator - pipeline
#
pipeline = make_pipeline(StandardScaler(), LogisticRegression(random_state=1))
#
# Create training test splits using two features
#
pipeline.fit(X_train[:, [2, 13]], y_train)
probs = pipeline.predict_proba(X_test[:, [2, 13]])
fpr1, tpr1, thresholds = roc_curve(y_test, probs[:, 1], pos_label=1)
roc_auc1 = auc(fpr1, tpr1)
#
# Create training test splits using two different features
#
pipeline.fit(X_train[:, [4, 14]], y_train)
probs2 = pipeline.predict_proba(X_test[:, [4, 14]])
fpr2, tpr2, thresholds = roc_curve(y_test, probs2[:, 1], pos_label=1)
roc_auc2 = auc(fpr2, tpr2)
#
```

```
# Create training test splits using all features
#
pipeline.fit(X_train, y_train)
probs3 = pipeline.predict_proba(X_test)
fpr3, tpr3, thresholds = roc_curve(y_test, probs3[:, 1], pos_label=1)
roc_auc3 = auc(fpr3, tpr3)

fig, ax = plt.subplots(figsize=(7.5, 7.5))

plt.plot(fpr1, tpr1, label='ROC Curve 1 (AUC = %0.2f)' % (roc_auc1))
plt.plot(fpr2, tpr2, label='ROC Curve 2 (AUC = %0.2f)' % (roc_auc2))
plt.plot(fpr3, tpr3, label='ROC Curve 3 (AUC = %0.2f)' % (roc_auc3))
plt.plot([0, 1], [0, 1], linestyle='--', color='red', label='Random Classifier')
plt.plot([0, 0, 1], [0, 1, 1], linestyle=':', color='green', label='Perfect Classifier')
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.legend(loc="lower right")
plt.show()
```

Output



Result

Thus, the performance of machine learning models has been measured successfully.

Ex No. 5	**Naïve Bayesian Classifier**
Date:	

Aim

To implement and compute the accuracy of Naïve Bayesian Classifier using training and test datasets.

Definition

Naïve Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press "Create" button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select "Run" to view the result.

Naivebayes.py

```
# load the iris dataset
from sklearn.datasets import load_iris
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split

iris = load_iris()


# store the feature matrix (X) and response vector (y)
X = iris.data
y = iris.target

# splitting X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=1)

# training the model on training set
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# making predictions on the testing set
```

```
y_pred = gnb.predict(X_test)

print(y_pred)

# comparing actual response values (y_test) with predicted response values (y_pred)
print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test,
y_pred)*100)
```

Output

```
C:\Users\TEMP\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/TEMP/PycharmProjects/sumaiya/naive.py

[0 1 1 0 2 2 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2 1 2 1 2 2 0 1

 0 1 2 2 0 1 2 1 2 0 0 0 1 0 0 2 2 2 2 2 1 2 1]

Gaussian Naive Bayes model accuracy(in %): 95.0


Process finished with exit code 0
```

Result

Thus, naïve Bayesian classifier has implemented and its accuracy has been computed successfully.

Ex No. 6	**Construction of Bayesian Belief Network**
Date:	

Aim

To construct a Bayesian belief network on heart disease data set to predict whether a patient is affected by heart disease based on maximum likelihood estimation.

Data set

heart.csv (https://github.com/kb22/Heart-Disease-Prediction/blob/master/dataset.csv)

Definition

Bayesian Belief Network

Bayesian Belief Network is a graphical representation of different probabilistic relationships among random variables in a particular set. It is a classifier with no dependency on attributes i.e it is condition independent. Due to its feature of joint probability, the probability in Bayesian Belief Network is derived, based on a condition — P(attribute/parent) i.e probability of an attribute, true over parent attribute.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press "Create" button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select "Run" to view the result.

BBN.py

```
import numpy as np
import pandas as pd
from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.models import BayesianNetwork

#read Heart Disease data
data = pd.read_csv("C:/Users/lab4/Downloads/heart.csv")
data = data.replace('?',np.nan)

#display the data
print('Few examples from the dataset are given below')
print(data.head())

print('\n Attributes and datatypes')
print(data.dtypes)
```

```
#Model Bayesian Belief Network
model = BayesianNetwork([('age', 'target'), ('sex', 'target'),('cp', 'target'),('trestbps', 'target'),('chol',
'target'),('fbs', 'target'),('restecg', 'target'),('thalach', 'target'),('exang', 'target'),('oldpeak', 'target'),('slope',
'target'),('ca', 'target'),('thal', 'target')])

#Learning CPDs using Maximum Likelihood Estimators
cpd_restecg = MaximumLikelihoodEstimator(model, data).estimate_cpd('restecg')
print(cpd_restecg)

cpd_cp = MaximumLikelihoodEstimator(model, data).estimate_cpd('cp')
print(cpd_cp)

cpd_oldpeak = MaximumLikelihoodEstimator(model, data).estimate_cpd('oldpeak')
print(cpd_oldpeak)

cpd_exang = MaximumLikelihoodEstimator(model, data).estimate_cpd('exang')
print(cpd_exang)

cpd_age = MaximumLikelihoodEstimator(model, data).estimate_cpd('age')
print(cpd_age)

cpd_chol = MaximumLikelihoodEstimator(model, data).estimate_cpd('chol')
print(cpd_chol)
```

Output

```
C:\Users\lab4\PycharmProjects\bnn\venv\Scripts\python.exe C:/Users/lab4/PycharmProjects/bnn/bbn.py

Few examples from the dataset are given below

   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  target

0   63    1   3      145   233    1  ...      0      2.3      0   0     1       1

1   37    1   2      130   250    0  ...      0      3.5      0   0     2       1

2   41    0   1      130   204    0  ...      0      1.4      2   0     2       1

3   56    1   1      120   236    0  ...      0      0.8      2   0     2       1

4   57    0   0      120   354    0  ...      1      0.6      2   0     2       1


[5 rows x 14 columns]


Attributes and datatypes

age          int64

sex          int64

cp           int64

trestbps     int64

chol         int64

fbs          int64

restecg      int64

thalach      int64
```

```
exang        int64
oldpeak     float64
slope        int64
ca           int64
thal         int64
target       int64
dtype: object
+------------+-----------+
| restecg(0) | 0.485149  |
+------------+-----------+
| restecg(1) | 0.50165   |
+------------+-----------+
| restecg(2) | 0.0132013 |
+------------+-----------+
+-------+-----------+
| cp(0) | 0.471947  |
+-------+-----------+
| cp(1) | 0.165017  |
+-------+-----------+
| cp(2) | 0.287129  |
+-------+-----------+
| cp(3) | 0.0759076 |
+-------+-----------+
+--------------+------------+
| oldpeak(0.0) | 0.326733   |
+--------------+------------+
| oldpeak(0.1) | 0.0231023  |
+--------------+------------+
| oldpeak(0.2) | 0.039604   |
+--------------+------------+
| oldpeak(0.3) | 0.00990099 |
+--------------+------------+
| oldpeak(0.4) | 0.029703   |
+--------------+------------+
| oldpeak(0.5) | 0.0165017  |
+--------------+------------+
| oldpeak(0.6) | 0.0462046  |
+--------------+------------+
| oldpeak(0.7) | 0.00330033 |
+--------------+------------+
| oldpeak(0.8) | 0.0429043  |
+--------------+------------+
| oldpeak(0.9) | 0.00990099 |
+--------------+------------+
| oldpeak(1.0) | 0.0462046  |
+--------------+------------+
| oldpeak(1.1) | 0.00660066 |
+--------------+------------+
| oldpeak(1.2) | 0.0561056  |
+--------------+------------+
| oldpeak(1.3) | 0.00330033 |
+--------------+------------+
| oldpeak(1.4) | 0.0429043  |
+--------------+------------+
| oldpeak(1.5) | 0.0165017  |
+--------------+------------+
| oldpeak(1.6) | 0.0363036  |
+--------------+------------+
| oldpeak(1.8) | 0.0330033  |
+--------------+------------+
| oldpeak(1.9) | 0.0165017  |
+--------------+------------+
| oldpeak(2.0) | 0.029703   |
+--------------+------------+
| oldpeak(2.1) | 0.00330033 |
+--------------+------------+
| oldpeak(2.2) | 0.0132013  |
+--------------+------------+
| oldpeak(2.3) | 0.00660066 |
+--------------+------------+
| oldpeak(2.4) | 0.00990099 |
+--------------+------------+
| oldpeak(2.5) | 0.00660066 |
+--------------+------------+
| oldpeak(2.6) | 0.019802   |
+--------------+------------+
| oldpeak(2.8) | 0.019802   |
+--------------+------------+
| oldpeak(2.9) | 0.00330033 |
+--------------+------------+
| oldpeak(3.0) | 0.0165017  |
+--------------+------------+
| oldpeak(3.1) | 0.00330033 |
+--------------+------------+
| oldpeak(3.2) | 0.00660066 |
+--------------+------------+
| oldpeak(3.4) | 0.00990099 |
+--------------+------------+
| oldpeak(3.5) | 0.00330033 |
+--------------+------------+
| oldpeak(3.6) | 0.0132013  |
+--------------+------------+
| oldpeak(3.8) | 0.00330033 |
+--------------+------------+
| oldpeak(4.0) | 0.00990099 |
+--------------+------------+
| oldpeak(4.2) | 0.00660066 |
+--------------+------------+
| oldpeak(4.4) | 0.00330033 |
+--------------+------------+
| oldpeak(5.6) | 0.00330033 |
+--------------+------------+
| oldpeak(6.2) | 0.00330033 |
+--------------+------------+
+----------+----------+
| exang(0) | 0.673267 |
+----------+----------+
| exang(1) | 0.326733 |
+----------+----------+
+---------+------------+
| age(29) | 0.00330033 |
+---------+------------+
| age(34) | 0.00660066 |
+---------+------------+
| age(35) | 0.0132013  |
+---------+------------+
| age(37) | 0.00660066 |
+---------+------------+
| age(38) | 0.00990099 |
+---------+------------+
| age(39) | 0.0132013  |
+---------+------------+
| age(40) | 0.00990099 |
+---------+------------+
| age(41) | 0.0330033  |
+---------+------------+
| age(42) | 0.0264026  |
+---------+------------+
| age(43) | 0.0264026  |
+---------+------------+
| age(44) | 0.0363036  |
+---------+------------+
| age(45) | 0.0264026  |
+---------+------------+
| age(46) | 0.0231023  |
+---------+------------+
| age(47) | 0.0165017  |
+---------+------------+
| age(48) | 0.0231023  |
+---------+------------+
| age(49) | 0.0165017  |
+---------+------------+
| age(50) | 0.0231023  |
+---------+------------+
| age(51) | 0.039604   |
+---------+------------+
| age(52) | 0.0429043  |
+---------+------------+
| age(53) | 0.0264026  |
+---------+------------+
| age(54) | 0.0528053  |
+---------+------------+
| age(55) | 0.0264026  |
+---------+------------+
| age(56) | 0.0363036  |
+---------+------------+
| age(57) | 0.0561056  |
+---------+------------+
| age(58) | 0.0627063  |
+---------+------------+
| age(59) | 0.0462046  |
+---------+------------+
| age(60) | 0.0363036  |
+---------+------------+
| age(61) | 0.0264026  |
+---------+------------+
| age(62) | 0.0363036  |
+---------+------------+
| age(63) | 0.029703   |
+---------+------------+
| age(64) | 0.0330033  |
+---------+------------+
| age(65) | 0.0264026  |
+---------+------------+
| age(66) | 0.0231023  |
+---------+------------+
| age(67) | 0.029703   |
+---------+------------+
| age(68) | 0.0132013  |
+---------+------------+
| age(69) | 0.00990099 |
+---------+------------+
| age(70) | 0.0132013  |
+---------+------------+
| age(71) | 0.00990099 |
+---------+------------+
| age(74) | 0.00330033 |
+---------+------------+
| age(76) | 0.00330033 |
+---------+------------+
| age(77) | 0.00330033 |
+---------+------------+


Process finished with exit code 0
```

Result

Thus, a Bayesian belief network has been successfully constructed on heart disease data set to predict the heart disease based on maximum likelihood estimation.

Ex No. 7	**Implementation of Expectation-Maximization (EM) Algorithm**
Date:	

Aim

To implement EM algorithm to cluster the data in IRIS data set.

Data Set

iris.csv (https://archive.ics.uci.edu/ml/machine-learning-databases/iris/)

Definition

EM Algorithm

Expectation-Maximization algorithm can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables) too in order to predict their values with the condition that the general form of probability distribution governing those latent variables is known to us. This algorithm is actually at the base of many unsupervised clustering algorithms in the field of machine learning.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press "Create" button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

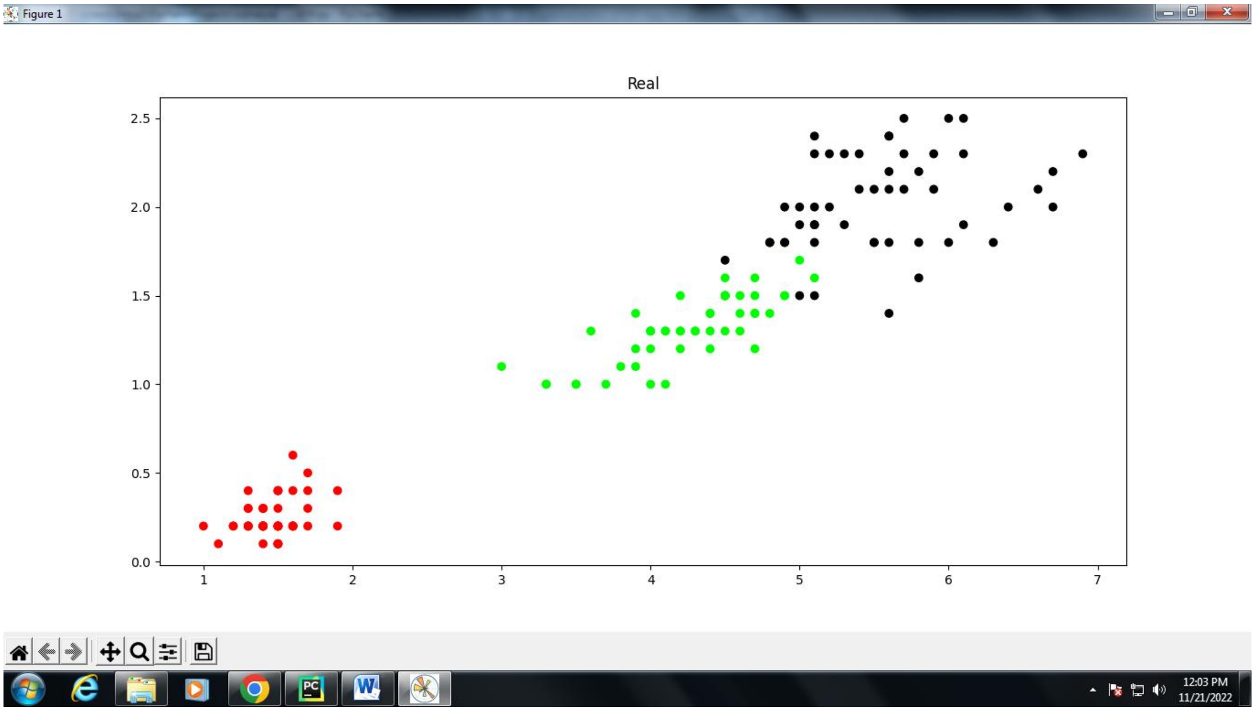
Type the following codes. Right click on file name or coding window → Select "Run" to view the result.

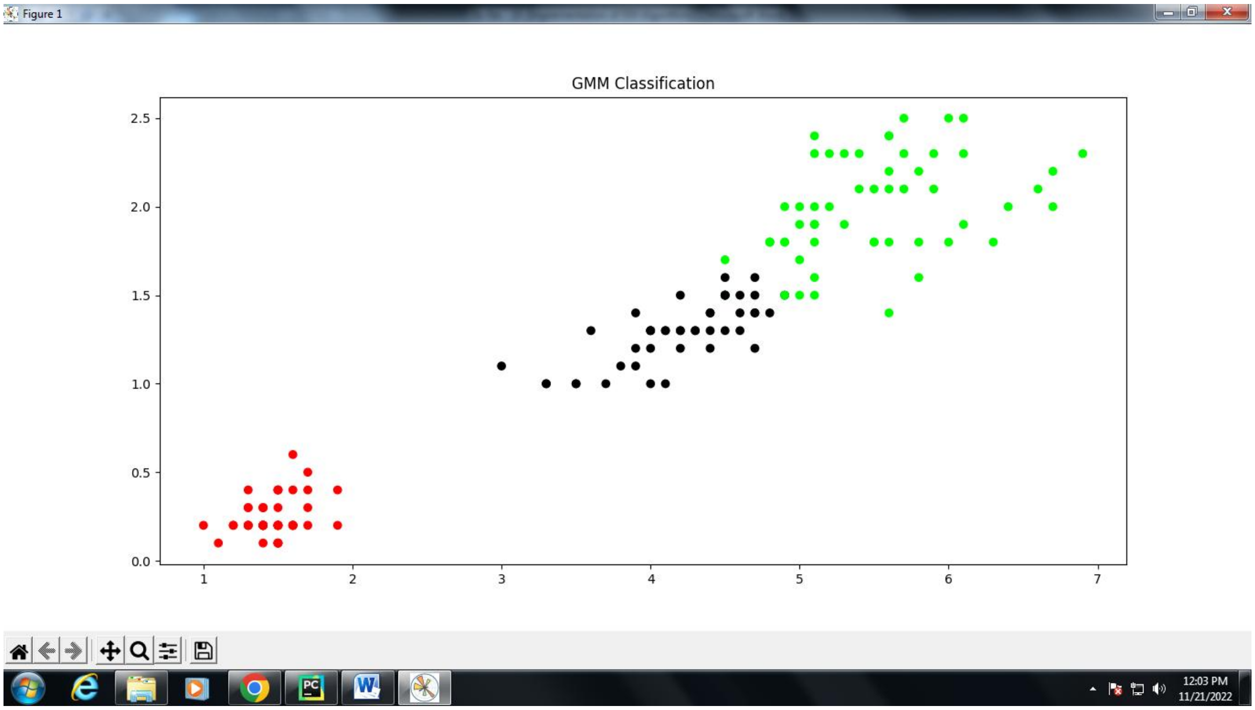
EM.py

```
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import sklearn.metrics as metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt


names = ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width', 'Class']

dataset = pd.read_csv("C:/Users/2mca2/Downloads/iris.csv", names=names)


X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2}

y = [label[c] for c in dataset.iloc[:, -1]]
```

```
plt.figure(figsize=(14,7))
colormap=np.array(['red','lime','black'])

# REAL PLOT

plt.title('Real')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y])
plt.show()
# GMM PLOT
gmm=GaussianMixture(n_components=3, random_state=0).fit(X)
y_cluster_gmm=gmm.predict(X)
print(y_cluster_gmm)
plt.title('GMM Classification')
plt.scatter(X.Petal_Length,X.Petal_Width,c=colormap[y_cluster_gmm])
plt.show()
print('The accuracy score of EM: ', metrics.accuracy_score(y, y_cluster_gmm))
print('The Confusion matrix of EM:\n ', metrics.confusion_matrix(y, y_cluster_gmm))
```

Output





```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/EM.py

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 2 1 2

 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1

 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

 1 1]

The accuracy score of EM:  0.36666666666666664

The Confusion matrix of EM:

  [[50  0  0]

 [ 0  5 45]

 [ 0 50  0]]

Process finished with exit code 0
```

Result

Thus, EM algorithm has been implemented to cluster the data in IRIS dataset.

Ex No. 8	Implementation of K-Nearest Neighbor Algorithm
Date:	

Aim

To write a program to implement k-nearest neighbor algorithm to classify the data set.

Definition

K-nearest neighbor algorithm

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

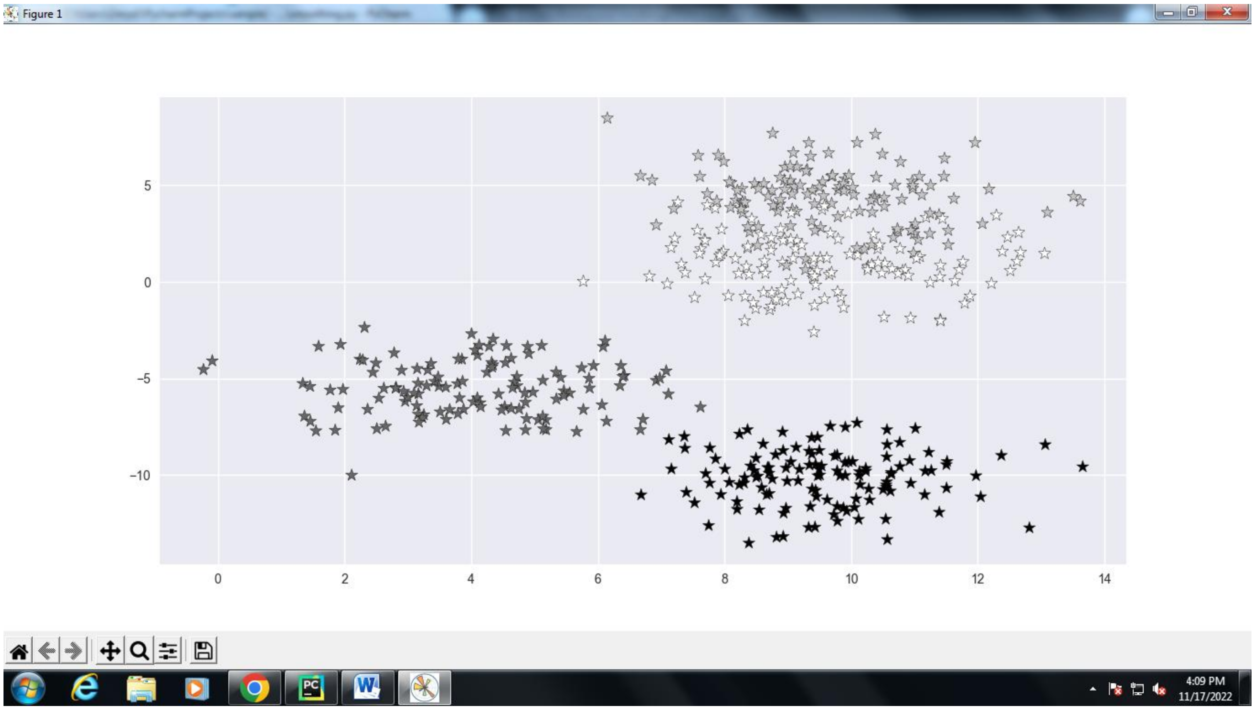
knearest.py

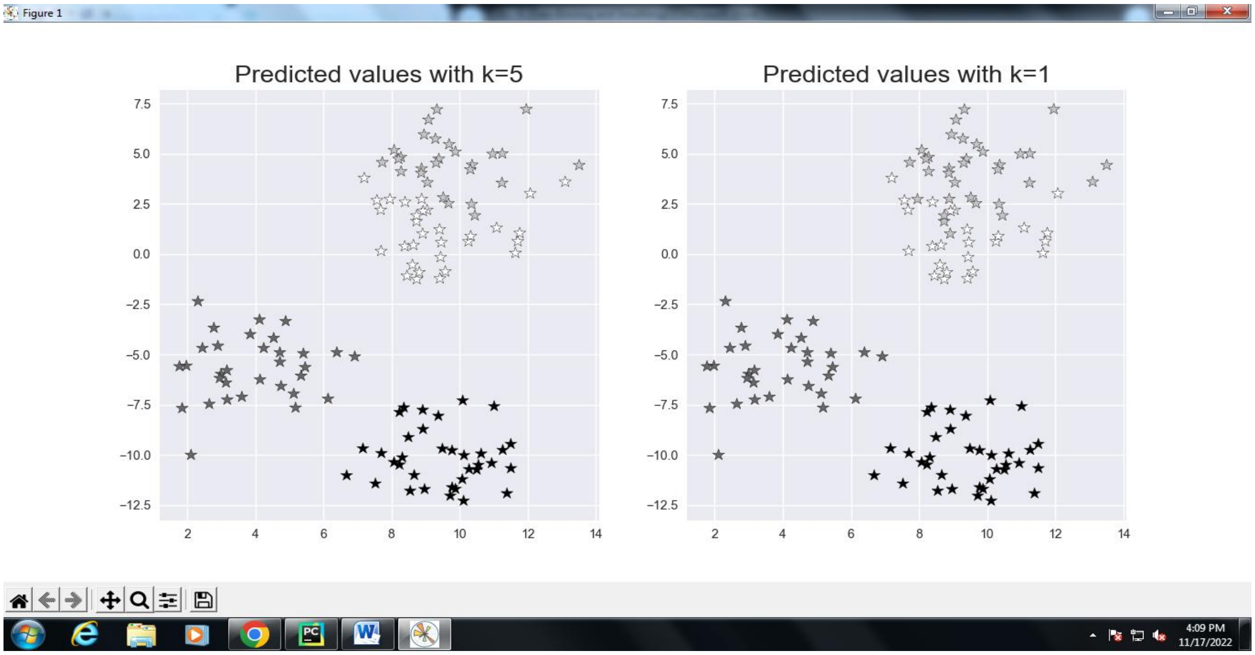
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import make_blobs
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

X, y = make_blobs(n_samples = 500, n_features = 2, centers = 4,cluster_std = 1.5, random_state = 4)
plt.style.use('seaborn')
plt.figure(figsize = (10,10))
plt.scatter(X[:,0], X[:,1], c=y, marker= '*',s=100,edgecolors='black')
plt.show()

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)

knn5 = KNeighborsClassifier(n_neighbors = 5)
knn1 = KNeighborsClassifier(n_neighbors=1)

knn5.fit(X_train, y_train)
knn1.fit(X_train, y_train)
```

```
y_pred_5 = knn5.predict(X_test)
y_pred_1 = knn1.predict(X_test)

print(y_pred_5)
print(y_pred_1)

from sklearn.metrics import accuracy_score
print("Accuracy with k=5", accuracy_score(y_test, y_pred_5)*100)
print("Accuracy with k=1", accuracy_score(y_test, y_pred_1)*100)

plt.figure(figsize = (15,5))
plt.subplot(1,2,1)
plt.scatter(X_test[:,0], X_test[:,1], c=y_pred_5, marker= '*', s=100,edgecolors='black')
plt.title("Predicted values with k=5", fontsize=20)

plt.subplot(1,2,2)
plt.scatter(X_test[:,0], X_test[:,1], c=y_pred_1, marker= '*', s=100,edgecolors='black')
plt.title("Predicted values with k=1", fontsize=20)
plt.show()
```

Output

```
C:\Users\2mca1\PycharmProjects\sample\venv\Scripts\python.exe
C:/Users/2mca1/PycharmProjects/sample/knearest.py

[3 0 3 2 0 1 1 2 0 3 0 1 0 3 3 3 0 3 2 1 1 1 1 3 2 3 3 3 3 0 2 1 2 0 2 3 1

 2 1 3 3 1 2 0 1 1 1 3 0 0 1 2 1 2 1 3 3 2 0 0 1 0 1 3 3 3 2 3 2 3 0 2 3 0

 1 3 3 3 2 0 3 2 0 1 1 0 0 0 0 2 3 1 0 3 0 2 2 2 3 0 2 3 2 3 0 0 2 1 2 2 2

 2 0 1 0 0 2 2 2 2 1 1 3 0 3]

[3 0 3 2 1 1 1 2 0 3 0 1 0 3 3 3 0 3 2 1 1 1 1 3 2 3 3 3 3 0 2 1 2 0 2 3 1

 2 1 3 3 1 2 0 1 1 1 3 1 0 1 2 1 2 1 3 3 2 1 0 1 0 1 3 3 3 2 3 2 3 0 2 3 0

 1 3 3 3 2 0 3 2 0 1 1 0 0 1 0 2 3 1 1 3 0 2 2 2 3 0 2 3 2 3 0 0 2 1 2 2 2

 2 1 1 0 0 2 2 2 2 1 1 3 0 3]

Accuracy with k=5 93.60000000000001

Accuracy with k=1 90.4

Process finished with exit code 0
```





Result

Thus, k-nearest neighbor algorithm in machine learning has implemented successfully.

Ex No. 9	**Decision Tree Pre-pruning and Post-pruning**
Date:	

Aim

To construct a decision tree and apply pre-pruning and post-pruning operations on it.

Definition

Pruning

Decision trees are a machine learning algorithm that is susceptible to overfitting. One of the techniques you can use to reduce overfitting in decision trees is pruning.

Pre-pruning

The pre-pruning technique of Decision Trees is tuning the hyperparameters prior to the training pipeline. It involves the heuristic known as ‘early stopping’ which stops the growth of the decision tree - preventing it from reaching its full depth.

Post-pruning

Post-pruning does the opposite of pre-pruning and allows the Decision Tree model to grow to its full depth. Once the model grows to its full depth, tree branches are removed to prevent the model from overfitting.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

Decisiontree.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Decision Tree Construction and Visualization
X,y=load_breast_cancer(return_X_y=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0)
clf=DecisionTreeClassifier(random_state=0)
clf.fit(X_train,y_train)
```

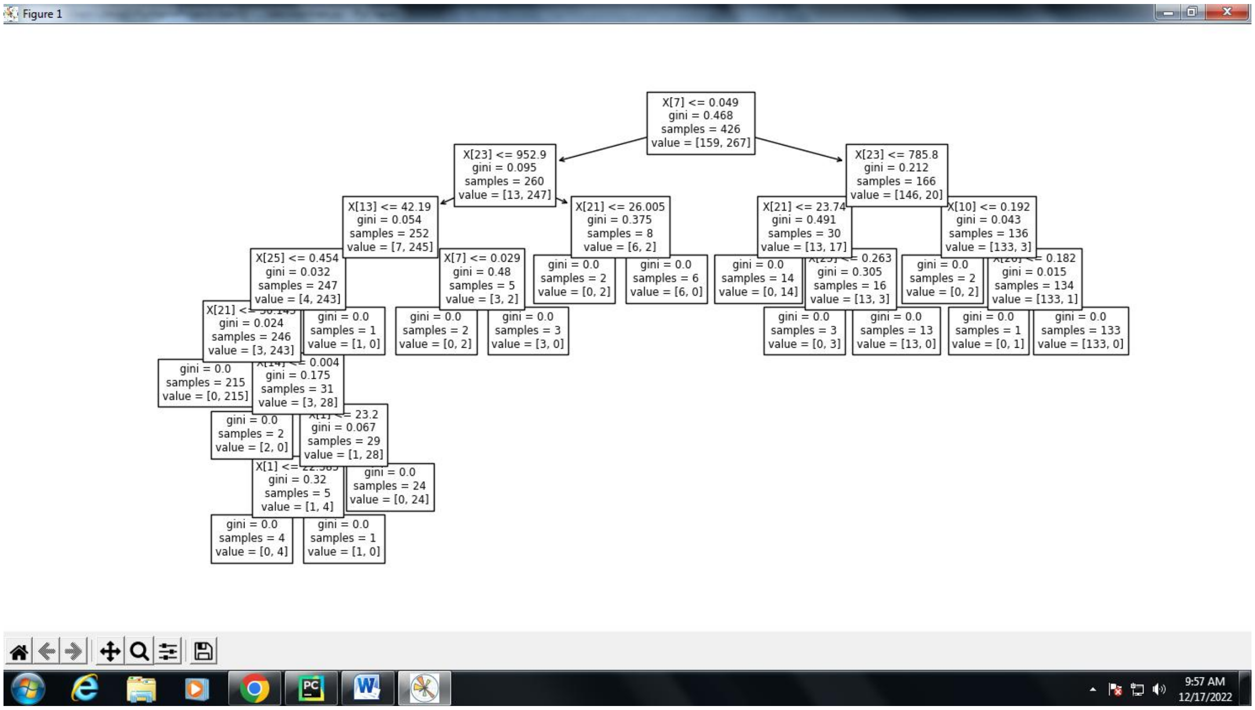
```
y_train_predicted=clf.predict(X_train)
y_test_predicted=clf.predict(X_test)
train_acc=accuracy_score(y_train,y_train_predicted)
test_acc=accuracy_score(y_test,y_test_predicted)
print(y_train_predicted)
print(y_test_predicted)
print("accuracy of training dataset:",train_acc)
print("accuracy of test dataset:",test_acc)

plt.figure(figsize=(16,8))
tree.plot_tree(clf)
plt.show()

# Post-Pruning
path=clf.cost_complexity_pruning_path(X_train,y_train)
#path variable gives two things ccp_alphas and impurities
ccp_alphas,impurities=path.ccp_alphas,path.impurities
print("ccp alpha wil give list of values :",ccp_alphas)
print("***********************************************************")
print("Impurities in Decision Tree :",impurities)

clfs=[]   #will store all the models here
for ccp_alpha in ccp_alphas:
   clf=DecisionTreeClassifier(random_state=0,ccp_alpha=ccp_alpha)
   clf.fit(X_train,y_train)
   clfs.append(clf)
print("Last node in Decision tree is {} and ccp_alpha for last node is {}".format(clfs[-
1].tree_.node_count,ccp_alphas[-1]))

train_scores = [clf.score(X_train, y_train) for clf in clfs]
test_scores = [clf.score(X_test, y_test) for clf in clfs]
fig, ax = plt.subplots()
ax.set_xlabel("alpha")
ax.set_ylabel("accuracy")
ax.set_title("Accuracy vs alpha for training and testing sets")
ax.plot(ccp_alphas, train_scores, marker='o', label="train",drawstyle="steps-post")
ax.plot(ccp_alphas, test_scores, marker='o', label="test",drawstyle="steps-post")
ax.legend()
plt.show()

clf=DecisionTreeClassifier(random_state=0,ccp_alpha=0.02)
clf.fit(X_train,y_train)
plt.figure(figsize=(12,8))
tree.plot_tree(clf,rounded=True,filled=True)
plt.show()

acc=accuracy_score(y_test,clf.predict(X_test))
print("accuracy of post-pruning operation:",acc)
```

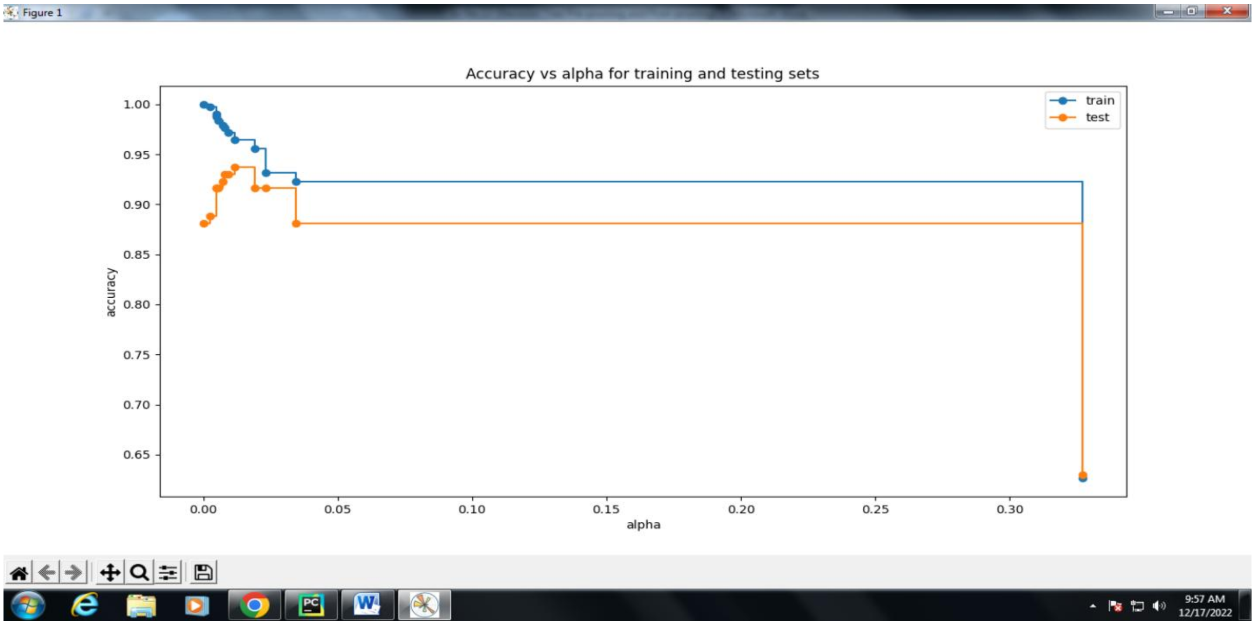
```
# Pre-Pruning

clf=DecisionTreeClassifier(criterion= 'gini',max_depth= 17,min_samples_leaf= 3,min_samples_split=
12,splitter= 'random')
clf.fit(X_train,y_train)
plt.figure(figsize=(20,12))
tree.plot_tree(clf,rounded=True,filled=True)
plt.show()

y_predicted=clf.predict(X_test)
accuracy=accuracy_score(y_test,y_predicted)
print(y_predicted)
print("accuracy of pre-pruning operation:",accuracy)
```

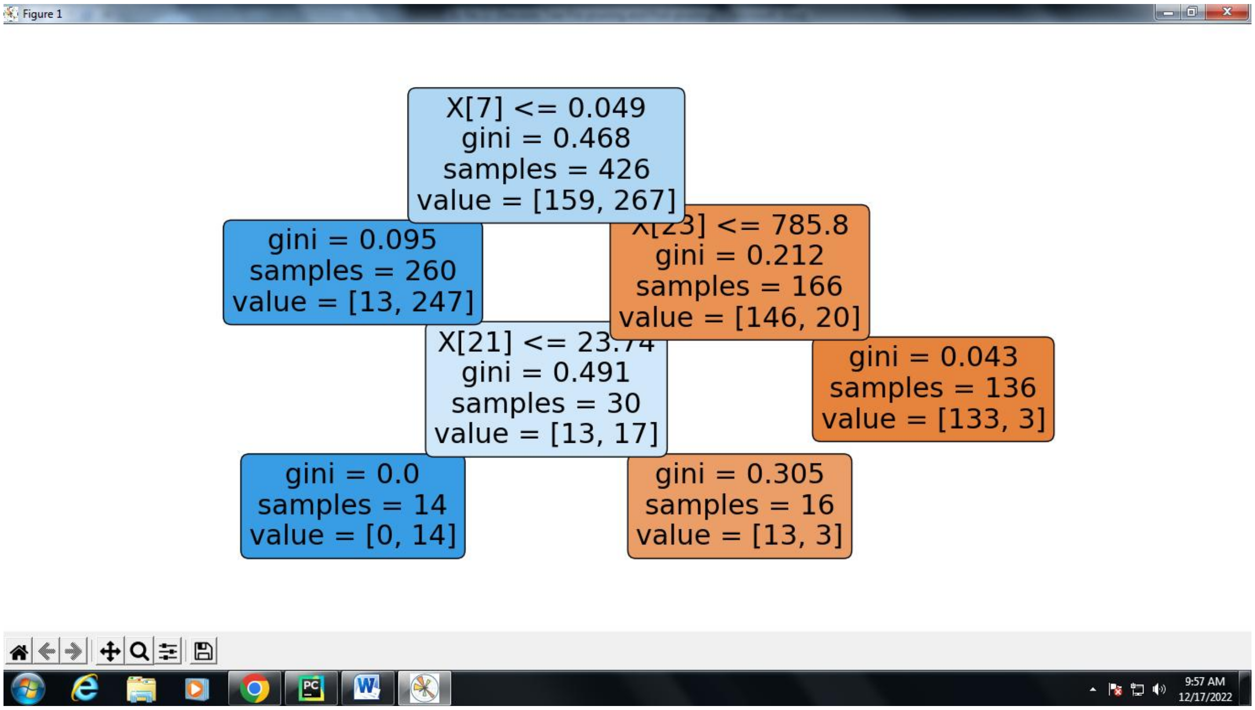
Output

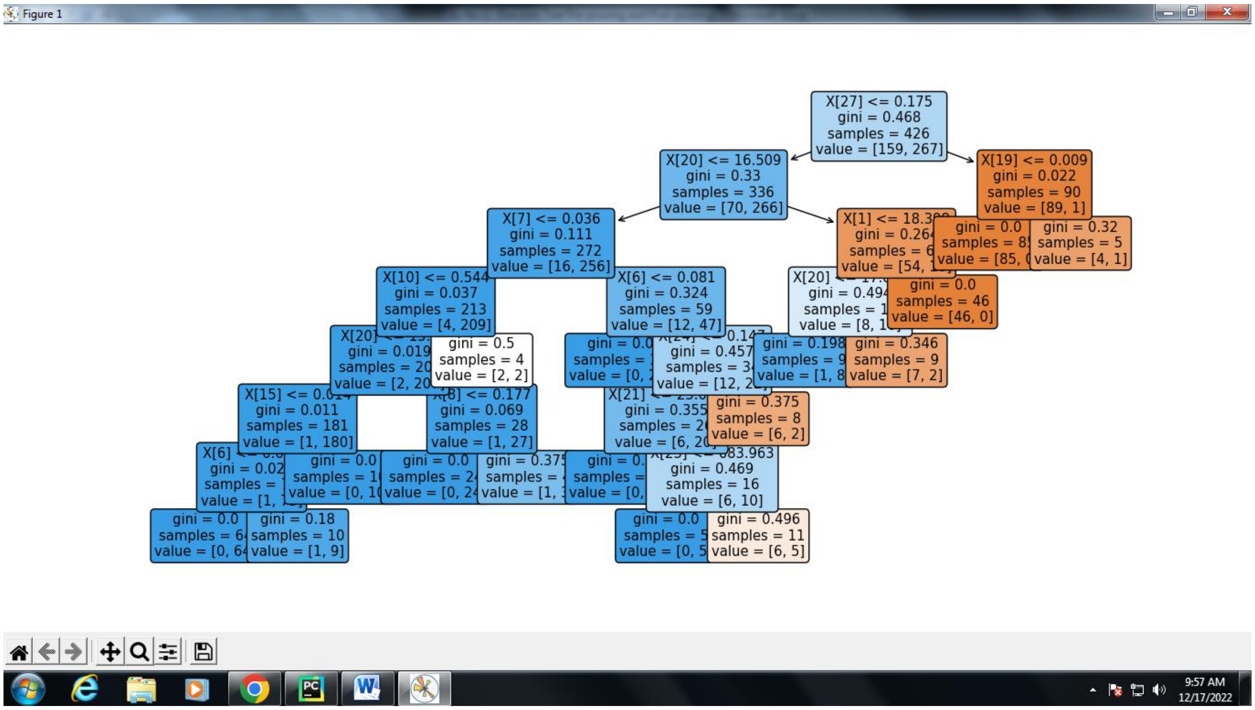
```
C:\Users\2mca2\PycharmProjects\tam1\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/tam1/decisiontree.py
[1 1 0 1 0 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 0
 0 1 1 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0 1 0
 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 1 0 1
 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1
 0 1 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 0 1 0 1 1 1 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 1 1 0
 1 1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1
 1 0 0 1 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 0 1 1 1
 1 1 0 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 1 0
 0 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1
 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 0 1 1 1 0
 0 1 1 1 0 1 1 0 1 1 1 1 1 0 0 0 1 1 1]
[0 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 1 0 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 1
 0 1 1 0 1 1 0 0 0 1 0 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 0 0 0 0 1
 0 0 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0]
accuracy of training dataset: 1.0
accuracy of test dataset: 0.8811188811188811
ccp alpha wil give list of values : [0.         0.00226647 0.00464743 0.0046598  0.0056338  0.00704225
 0.00784194 0.00911402 0.01144366 0.018988   0.02314163 0.03422475
 0.32729844]
**********************************************************
Impurities in Decision Tree : [0.         0.00453294 0.01847522 0.02313502 0.02876883 0.03581108
 0.04365302 0.05276704 0.0642107  0.0831987  0.10634033 0.14056508
 0.46786352]
Last node in Decision tree is 1 and ccp_alpha for last node is 0.3272984419327777
accuracy of post-pruning operation: 0.916083916083916
[0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1
 0 1 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 1
 0 1 1 1 1 1 0 0 0 1 1 1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0]
accuracy of pre-pruning operation: 0.9300699300699301
```

Process finished with exit code 0









Result

Thus, a decision tree has been successfully constructed. Pre-pruning and post-pruning operations have been performed and accuracy of both operations has compared.

Ex No. 10	**Implementation of Backpropagation Algorithm**
Date:	

Aim

To build an artificial neural network by implementing backpropagation algorithm.

Definitions

Neural Network

Artificial neural networks, usually simply called neural networks or neural nets, are computing systems inspired by the biological neural networks that constitute animal brains. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain.

Backpropagation Algorithm

Backpropagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization. Backpropagation in neural network is a short form for "backward propagation of errors." It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press "Create" button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select "Run" to view the result.

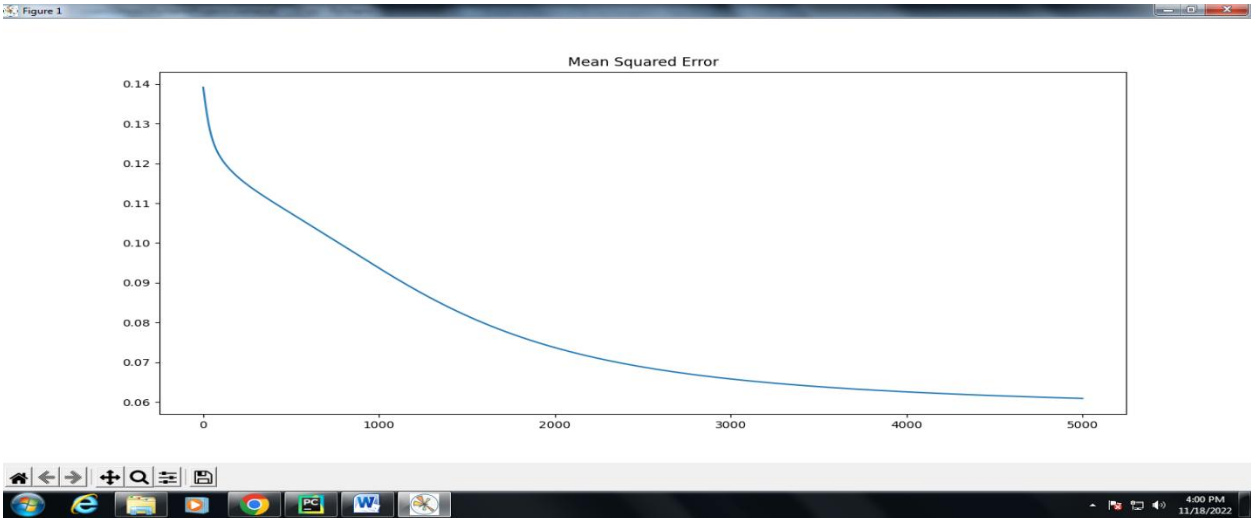
Backprop.py

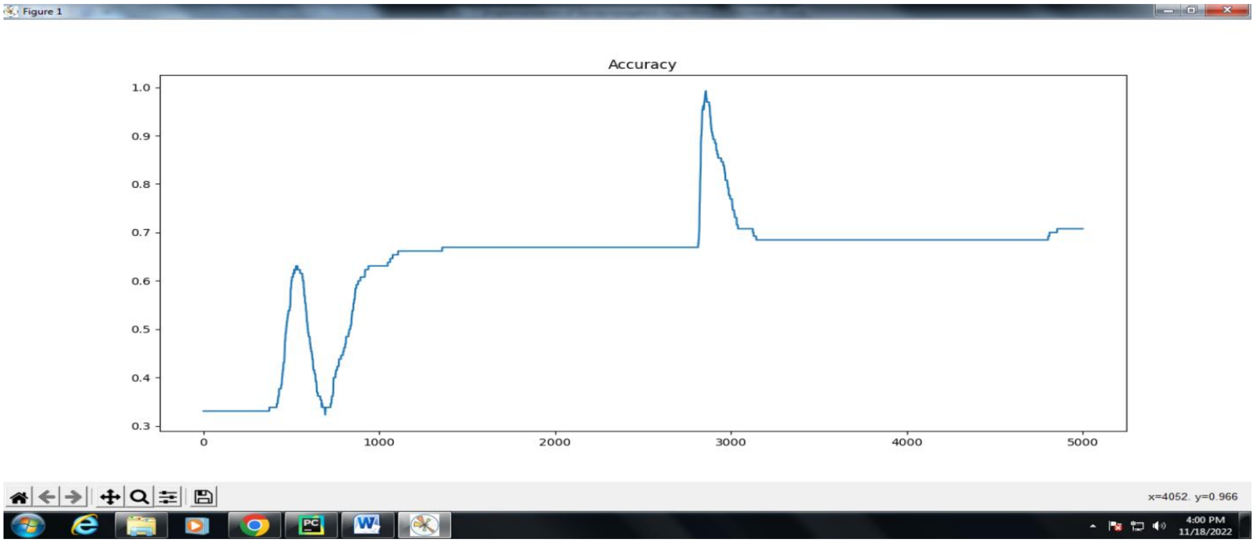
```
# Import Libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Load dataset
data = load_iris()

# Get features and target
X=data.data
y=data.target
```

```
# Get dummy variable
y = pd.get_dummies(y).values

print(y[:3])

#Split data into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=20, random_state=4)

# Initialize variables
learning_rate = 0.1
iterations = 5000
N = y_train.size

# number of input features
input_size = 4

# number of hidden layers neurons
hidden_size = 2

# number of neurons at the output layer
output_size = 3

results = pd.DataFrame(columns=["mse", "accuracy"])

# Initialize weights
np.random.seed(10)

# initializing weight for the hidden layer
W1 = np.random.normal(scale=0.5, size=(input_size, hidden_size))

# initializing weight for the output layer
W2 = np.random.normal(scale=0.5, size=(hidden_size , output_size))


def sigmoid(x):
    return 1 / (1 + np.exp(-x))


def mean_squared_error(y_pred, y_true):
    return ((y_pred - y_true) ** 2).sum() / (2 * y_pred.size)


def accuracy(y_pred, y_true):
    acc = y_pred.argmax(axis=1) == y_true.argmax(axis=1)
    return acc.mean()


for itr in range(iterations):
    # feedforward propagation
    # on hidden layer
    Z1 = np.dot(X_train, W1)
    A1 = sigmoid(Z1)

    # on output layer
    Z2 = np.dot(A1, W2)
    A2 = sigmoid(Z2)

    # Calculating error
    mse = mean_squared_error(A2, y_train)
    acc = accuracy(A2, y_train)
    results = results.append({"mse": mse, "accuracy": acc}, ignore_index=True)

    # backpropagation
    E1 = A2 - y_train
    dW1 = E1 * A2 * (1 - A2)

    E2 = np.dot(dW1, W2.T)
    dW2 = E2 * A1 * (1 - A1)

    # weight updates
    W2_update = np.dot(A1.T, dW1) / N
    W1_update = np.dot(X_train.T, dW2) / N

    W2 = W2 - learning_rate * W2_update
    W1 = W1 - learning_rate * W1_update

print(results.mse.plot(title="Mean Squared Error"))
plt.show()
print(results.accuracy.plot(title="Accuracy"))
plt.show()

# feedforward
Z1 = np.dot(X_test, W1)
A1 = sigmoid(Z1)

Z2 = np.dot(A1, W2)
A2 = sigmoid(Z2)

acc = accuracy(A2, y_test)
print("Accuracy: {}".format(acc))
```

Output

```
C:\Users\2mca1\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca1/PycharmProjects/sumaiya/backprop.py

[[1 0 0]

 [1 0 0]

 [1 0 0]]

AxesSubplot(0.125,0.11;0.775x0.77)
AxesSubplot(0.125,0.11;0.775x0.77)

Accuracy: 0.8

Process finished with exit code 0
```





Result

Thus, an artificial neural network has been constructed by implementing backpropagation algorithm.

Ex No. 11	**Support Vector Classification**
Date:	

Aim

To implement Support Vector Classification algorithm for Linear Kernels.

Definition

Support Vector Machine (SVM)

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

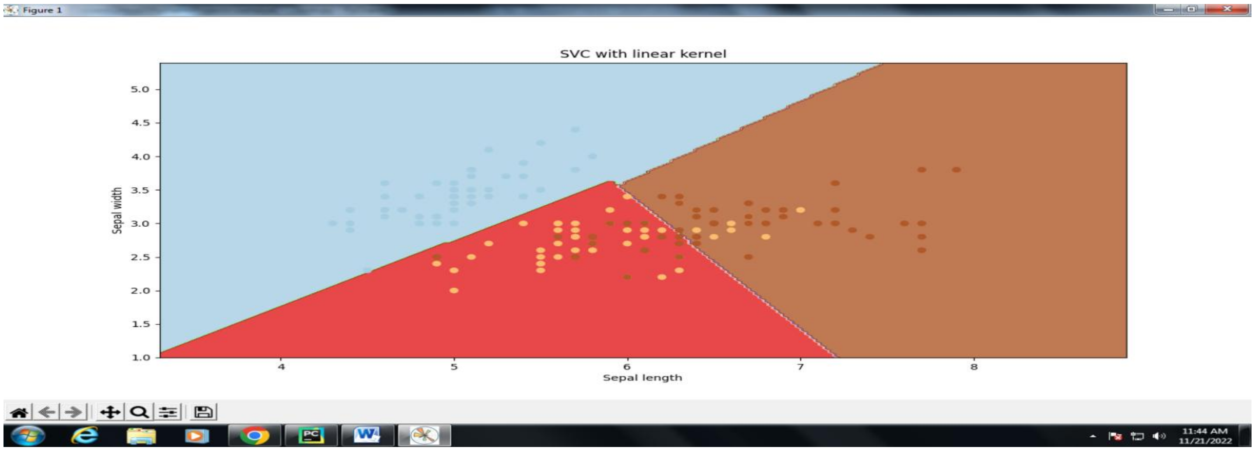
SVM.py

```
# Import the Libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets

# Import some Data from the iris Data Set
iris = datasets.load_iris()

# Take only the first two features of Data.
# To avoid the slicing, Two-Dim Dataset can be used

X = iris.data[:, :2]
y = iris.target

# C is the SVM regularization parameter
C = 1.0

# Create an Instance of SVM and Fit out the data.
# Data is not scaled so as to be able to plot the support vectors
```

```
svc = svm.SVC(kernel ='linear', C = 1).fit(X, y)

# create a mesh to plot
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
h = (x_max / x_min)/100
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

# Plot the data for Proper Visual Representation
plt.subplot(1, 1, 1)

# Predict the result by giving Data to the model
Z = svc.predict(np.c_[xx.ravel(), yy.ravel()])
print(Z)
Z = Z.reshape(xx.shape)
plt.contourf(xx, yy, Z, cmap = plt.cm.Paired, alpha = 0.8)

plt.scatter(X[:, 0], X[:, 1], c = y, cmap = plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.xlim(xx.min(), xx.max())
plt.title('SVC with linear kernel')

# Output the Plot
plt.show()
```

Output

```
C:\Users\lab4\PycharmProjects\sa\venv\Scripts\python.exe C:/Users/lab4/PycharmProjects/sa/svm.py

[1 1 1 ... 2 2 2]

Process finished with exit code 0
```



Result

Thus, Support Vector Classification algorithm has been implemented successfully.

Ex No. 12	**Implementation of Logistic Regression to Classify Problems**
Date:	

Aim

To implement Logistic Regression algorithm to classify problems such as Diabetes prediction and Spam detection.

Data Sets

1. diabetes.csv (https://www.kaggle.com/datasets/saurabh00007/diabetescsv)
2. SMSSpamCollection.csv (https://archive.ics.uci.edu/ml/machine-learning-databases/00228/)

Definition

Logistic Regression

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1.

Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press “Create” button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select “Run” to view the result.

Diabetes Prediction:

Logisticdia.py

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics

data = pd.read_csv("C:/Users/2mca2/Downloads/diabetes.csv")

print(data.head)

print(data.dtypes)

print(data.describe())

X = data.drop("Outcome", axis=1)
Y = data[["Outcome"]]
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=7)

model = LogisticRegression()
model.fit(X_train, Y_train)
Y_predict = model.predict(X_test)
model_score = model.score(X_test, Y_test)
print(model_score)
print(metrics.confusion_matrix(Y_test, Y_predict))
```

Output

```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/logisticdia.py

<bound method NDFrame.head of      Pregnancies  Glucose  ...  Age  Outcome

0            6      148  ...   50        1

1            1       85  ...   31        0

2            8      183  ...   32        1

3            1       89  ...   21        0

4            0      137  ...   33        1

..         ...      ...  ...  ...      ...

763         10      101  ...   63        0

764          2      122  ...   27        0

765          5      121  ...   30        0

766          1      126  ...   47        1

767          1       93  ...   23        0

[768 rows x 9 columns]>

Pregnancies                   int64

Glucose                       int64

BloodPressure                 int64

SkinThickness                 int64

Insulin                       int64

BMI                         float64

DiabetesPedigreeFunction    float64

Age                           int64
```

```
Outcome                     int64
dtype: object
       Pregnancies     Glucose  ...        Age     Outcome
count   768.000000  768.000000  ...  768.000000  768.000000
mean      3.845052  120.894531  ...   33.240885    0.348958
std       3.369578   31.972618  ...   11.760232    0.476951
min       0.000000    0.000000  ...   21.000000    0.000000
25%       1.000000   99.000000  ...   24.000000    0.000000
50%       3.000000  117.000000  ...   29.000000    0.000000
75%       6.000000  140.250000  ...   41.000000    1.000000
max      17.000000  199.000000  ...   81.000000    1.000000
[8 rows x 9 columns]
0.7489177489177489
[[127  20]
 [ 38  46]]
  Process finished with exit code 0
```

Spam Detection:

Logisticspam.py

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score

df = pd.read_csv("C:/Users/2mca2/Downloads/SMSSpamCollection.csv", delimiter='\t', header=None)

print(df.describe)
print(df.dtypes)
print(df.head)
print(df.shape)
X_train_raw, X_test_raw, y_train, y_test = train_test_split(df[1],df[0])

vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train_raw)
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

```
X_test = vectorizer.transform( ['URGENT! Your Mobile No 1234 was awarded a Prize', 'Hey honey,
whats up?'] )
predictions = classifier.predict(X_test)
print("Result:")
print(predictions)
```

Output

```
C:\Users\2mca2\PycharmProjects\sumaiya\venv\Scripts\python.exe
C:/Users/2mca2/PycharmProjects/sumaiya/Logisticspam.py

<bound method NDFrame.describe of         0                                                  1

0      ham  Go until jurong point, crazy.. Available only ...

1      ham                      Ok lar... Joking wif u oni...

2     spam  Free entry in 2 a wkly comp to win FA Cup fina...

3      ham  U dun say so early hor... U c already then say...

4      ham  Nah I don't think he goes to usf, he lives aro...

...    ...                                               ...

5567  spam  This is the 2nd time we have tried 2 contact u...

5568   ham              Will ü b going to esplanade fr home?

5569   ham  Pity, * was in mood for that. So...any other s...

5570   ham  The guy did some bitching but I acted like i'd...

5571   ham                         Rofl. Its true to its name

[5572 rows x 2 columns]>

0    object

1    object

dtype: object

<bound method NDFrame.head of         0                                                  1

0      ham  Go until jurong point, crazy.. Available only ...

1      ham                      Ok lar... Joking wif u oni...

2     spam  Free entry in 2 a wkly comp to win FA Cup fina...

3      ham  U dun say so early hor... U c already then say...

4      ham  Nah I don't think he goes to usf, he lives aro...
```

```
...    ...                                              ...
5567  spam  This is the 2nd time we have tried 2 contact u...
5568   ham               Will ü b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So...any other s...
5570   ham  The guy did some bitching but I acted like i'd...
5571   ham                         Rofl. Its true to its name
[5572 rows x 2 columns]>
(5572, 2)
Result:
['spam' 'ham']
Process finished with exit code 0
```

Result

Thus, Logistic Regression algorithm for Diabetes Prediction and Spam Detection has been implemented successfully.

Additional Experiments

Ex No. 13	Data Structures in Machine Learning
Date:	

Aim

To demonstrate the data structures used in Machine Learning.

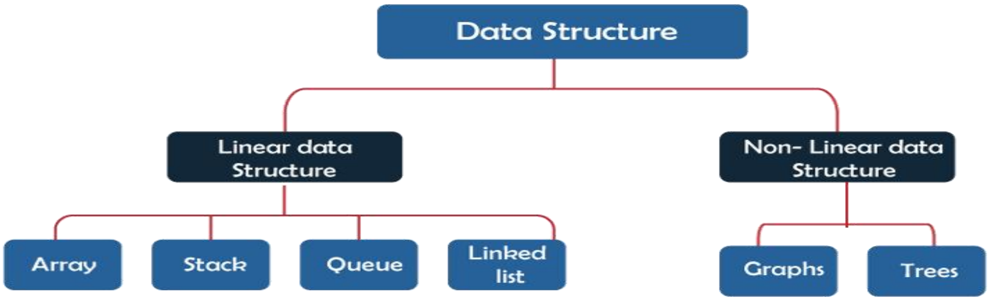
Definition

Data Structure

The data structure is defined as the basic building block of computer programming that helps us to organize, manage and store data for efficient search and retrieval.

The data structure is the ordered sequence of data, and it tells the compiler how a programmer is using the data such as Integer, String, Boolean, etc.

There are two different types of data structures: Linear and Non-linear data structures.



Procedure

Open PyCharm Community Edition.

Go to File menu → New Project → Specify the project name → Press "Create" button.

Right Click on Project name → New → Python File → Specify the file name → Press Enter.

Type the following codes. Right click on file name or coding window → Select "Run" to view the result.

A) Linear Data Structure

1. Array

# Python program to demonstrate

# Creation of Array

```
# importing "array" for array creations
import array as arr
# creating an array with integer type
a = arr.array('i', [1, 2, 3])
# printing original array
print ("The new created array is : ", end =" ")
for i in range (0, 3):
        print (a[i], end =" ")
print()
# creating an array with float type
b = arr.array('d', [2.5, 3.2, 3.3])
# printing original array
print ("The new created array is : ", end =" ")
for i in range (0, 3):
        print (b[i], end =" ")
```

Output

```
The new created array is :  1 2 3
The new created array is :  2.5 3.2 3.3
```

```
# Python program to demonstrate
# Adding Elements to a Array
# importing "array" for array creations
import array as arr
# array with int type
a = arr.array('i', [1, 2, 3])
print ("Array before insertion : ", end =" ")
for i in range (0, 3):
        print (a[i], end =" ")
print()
```

```
# inserting array using
# insert() function
a.insert(1, 4)
print ("Array after insertion : ", end =" ")
for i in (a):
        print (i, end =" ")
print()
# array with float type
b = arr.array('d', [2.5, 3.2, 3.3])
print ("Array before insertion : ", end =" ")
for i in range (0, 3):
        print (b[i], end =" ")
print()
# adding an element using append()
b.append(4.4)
print ("Array after insertion : ", end =" ")
for i in (b):
        print (i, end =" ")
print()
```

Output

```
Array before insertion : 1 2 3
Array after insertion :  1 4 2 3
Array before insertion : 2.5 3.2 3.3
Array after insertion :  2.5 3.2 3.3 4.4
```

```
# Python program to demonstrate
# accessing of element from list
# importing array module
import array as arr
```

```
# array with int type
a = arr.array('i', [1, 2, 3, 4, 5, 6])
# accessing element of array
print("Access element is: ", a[0])
# accessing element of array
print("Access element is: ", a[3])
# array with float type
b = arr.array('d', [2.5, 3.2, 3.3])
# accessing element of array
print("Access element is: ", b[1])
# accessing element of array
print("Access element is: ", b[2])
```

Output

```
Access element is:  1
Access element is:  4
Access element is:  3.2
Access element is:  3.3
```

```
# Python program to demonstrate
# Removal of elements in a Array


# importing "array" for array operations
import array
# initializing array with array values
# initializes array with signed integers
arr = array.array('i', [1, 2, 3, 1, 5])
# printing original array
print ("The new created array is : ", end ="")
for i in range (0, 5):
        print (arr[i], end =" ")
print ("\r")

# using pop() to remove element at 2nd position

print ("The popped element is : ", end ="")

print (arr.pop(2))

# printing array after popping

print ("The array after popping is : ", end ="")

for i in range (0, 4):

    print (arr[i], end =" ")

print("\r")

# using remove() to remove 1st occurrence of 1

arr.remove(1)

# printing array after removing

print ("The array after removing is : ", end ="")

for i in range (0, 3):

    print (arr[i], end =" ")
```

Output

```
The new created array is : 1 2 3 1 5
The popped element is : 3
The array after popping is : 1 2 1 5
The array after removing is : 2 1 5
```

```
# Python code to demonstrate

# searching an element in array

# importing array module

import array

# initializing array with array values

# initializes array with signed integers

arr = array.array('i', [1, 2, 3, 1, 2, 5])

# printing original array

print ("The new created array is : ", end ="")
```

```
for i in range (0, 6):

    print (arr[i], end =" ")

print ("\r")

# using index() to print index of 1st occurrence of 2

print ("The index of 1st occurrence of 2 is : ", end ="")

print (arr.index(2))

# using index() to print index of 1st occurrence of 1

print ("The index of 1st occurrence of 1 is : ", end ="")

print (arr.index(1))
```

Output

```
The new created array is : 1 2 3 1 2 5
The index of 1st occurrence of 2 is : 1
The index of 1st occurrence of 1 is : 0
```

```
# Python code to demonstrate

# how to update an element in array

# importing array module

import array

# initializing array with array values

# initializes array with signed integers

arr = array.array('i', [1, 2, 3, 1, 2, 5])

# printing original array

print ("Array before updation : ", end ="")

for i in range (0, 6):

    print (arr[i], end =" ")

print ("\r")

# updating a element in a array

arr[2] = 6

print("Array after updation : ", end ="")

for i in range (0, 6):
        print (arr[i], end =" ")
print()
# updating a element in a array
arr[4] = 8
print("Array after updation : ", end ="")
for i in range (0, 6):
        print (arr[i], end =" ")
```

Output

```
Array before updation : 1 2 3 1 2 5
Array after updation : 1 2 6 1 2 5
Array after updation : 1 2 6 1 8 5
```

2. Stack

```
# Python program to
# demonstrate stack implementation
# using list
stack = []
# append() function to push
# element in the stack
stack.append('a')
stack.append('b')
stack.append('c')
print('Initial stack')
print(stack)
# pop() function to pop
# element from stack in
# LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')

print(stack)

# uncommenting print(stack.pop())

# will cause an IndexError

# as the stack is now empty
```

Output

```
Initial stack
['a', 'b', 'c']

Elements popped from stack:
c
b
a

Stack after elements are popped:
[]
```

3. Queue

```
# Python program to

# demonstrate queue implementation

# using list

# Initializing a queue

queue = []

# Adding elements to the queue

queue.append('a')

queue.append('b')

queue.append('c')

print("Initial queue")

print(queue)

# Removing elements from the queue

print("\nElements dequeued from queue")

print(queue.pop(0))

print(queue.pop(0))
print(queue.pop(0))

print("\nQueue after removing elements")

print(queue)

# Uncommenting print(queue.pop(0))

# will raise and IndexError

# as the queue is now empty
```

Output

```
Initial queue
['a', 'b', 'c']

Elements dequeued from queue
a
b
c

Queue after removing elements
[]
```

4. Linked List

```
#Creation
class Node:
   def __init__(self, dataval=None):
      self.dataval = dataval
      self.nextval = None
class SLinkedList:
   def __init__(self):
      self.headval = None

list1 = SLinkedList()
list1.headval = Node("Mon")
e2 = Node("Tue")
e3 = Node("Wed")# Link first Node to second node
list1.headval.nextval = e2
# Link second Node to third node
e2.nextval = e3

**********************************************************************

class Node:
   def __init__(self, dataval=None):
      self.dataval = dataval
      self.nextval = None
class SLinkedList:
   def __init__(self):
      self.headval = None
```

```
    def listprint(self):
        printval = self.headval
        while printval is not None:
            print (printval.dataval)
            printval = printval.nextval

list = SLinkedList()
list.headval = Node("Mon")
e2 = Node("Tue")
e3 = Node("Wed")
# Link first Node to second node
list.headval.nextval = e2
# Link second Node to third node
e2.nextval = e3

list.listprint()
```

Output

```
Mon
Tue
Wed
```

B) Non - Linear Data Structure

1. Graphs

```
# Python program for

# validation of a graph

# import dictionary for graph

from collections import defaultdict

# function for adding edge to graph

graph = defaultdict(list)

def addEdge(graph,u,v):

        graph[u].append(v)

# definition of function

def generate_edges(graph):

        edges = []

        # for each node in graph

        for node in graph:

                # for each neighbour node of a single node

                for neighbour in graph[node]:
```

```
                # if edge exists then append
                        edges.append((node, neighbour))
        return edges
# declaration of graph as dictionary
addEdge(graph,'a','c')
addEdge(graph,'b','c')
addEdge(graph,'b','e')
addEdge(graph,'c','d')
addEdge(graph,'c','e')
addEdge(graph,'c','a')
addEdge(graph,'c','b')
addEdge(graph,'e','b')
addEdge(graph,'d','c')
addEdge(graph,'e','c')
# Driver Function call
# to print generated graph
print(generate_edges(graph))
```

Output

```
[('a', 'c'), ('b', 'c'), ('b', 'e'), ('c', 'd'),
 ('c', 'e'), ('c', 'a'), ('c', 'b'), ('e', 'b'),
 ('e', 'c'), ('d', 'c')]
```

2. Trees

```
# node class
class Node:


  def __init__(self, data):
    # left child
    self.left = None
    # right child
```

```
        self.right = None
        # node's value
        self.data = data
    # print function
    def PrintTree(self):
        print(self.data)
root = Node(27)
root.PrintTree()
```

Output

```
27
```

```
class Node:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data
    def insert(self, data):
# Compare the new value with the parent node
        if self.data:
            if data < self.data:
                if self.left is None:
                    self.left = Node(data)
                else:
                    self.left.insert(data)
            elif data > self.data:
                if self.right is None:
                    self.right = Node(data)
                else:
                    self.right.insert(data)
        else:
            self.data = data
# Print the tree
    def PrintTree(self):
        if self.left:
            self.left.PrintTree()
        print( self.data),
        if self.right:
            self.right.PrintTree()
# Use the insert method to add nodes
root = Node(27)
root.insert(14)
root.insert(35)
root.insert(31)
root.insert(10)
root.insert(19)
root.PrintTree()
```

Output

```
10 14 19 27 31 35
```

```
class Node:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data
# Insert method to create nodes
    def insert(self, data):
        if self.data:
            if data < self.data:
                if self.left is None:
                    self.left = Node(data)
                else:
                    self.left.insert(data)
            elif data > self.data:
                if self.right is None:
                    self.right = Node(data)
                else:
                    self.right.insert(data)
        else:
            self.data = data
# findval method to compare the value with nodes
    def findval(self, lkpval):
        if lkpval < self.data:
            if self.left is None:
                return str(lkpval)+" is not Found"
            return self.left.findval(lkpval)
        elif lkpval > self.data:
            if self.right is None:
                return str(lkpval)+" is not Found"
            return self.right.findval(lkpval)
        else:
            return str(self.data) + " is found"
# Print the tree
    def PrintTree(self):
        if self.left:
            self.left.PrintTree()
        print(self.data),
        if self.right:
            self.right.PrintTree()
root = Node(27)
```

```
root.insert(14)

root.insert(35)

root.insert(31)

root.insert(10)

root.insert(19)

print(root.findval(7))

print(root.findval(14))
```

Output

```
7 is not Found

14 is found
```

Result

Thus, the data structures used in machine learning have been implemented successfully.

Ex No. 14	**R Programming in ML**
Date:	

Aim

To write R programs to work on IRIS dataset.

Definition

R Programming Language

R is a programming language for statistical computing and graphics supported by the R Core Team and the R Foundation for Statistical Computing.

Procedure

Open R studio.

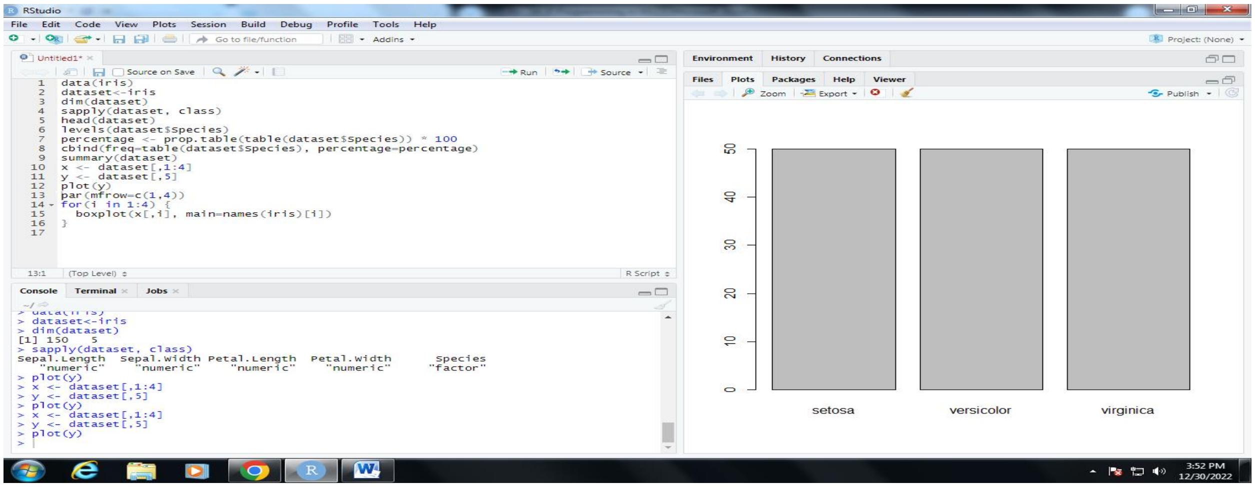
Go to file → new file → R Script.

Type the following R scripts and Click the run button to view the result.

R Scripts

```
> data(iris)

> dataset<-iris

> dim(dataset)

[1] 150   5

> sapply(dataset, class)

Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species

  "numeric"    "numeric"    "numeric"    "numeric"     "factor"

> head(dataset)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species

1         5.1         3.5         1.4         0.2  setosa

2         4.9         3.0         1.4         0.2  setosa

3         4.7         3.2         1.3         0.2  setosa

4         4.6         3.1         1.5         0.2  setosa

5         5.0         3.6         1.4         0.2  setosa

6         5.4         3.9         1.7         0.4  setosa
```

```
> levels(dataset$Species)
[1] "setosa"     "versicolor" "virginica"
> percentage <- prop.table(table(dataset$Species)) * 100
> cbind(freq=table(dataset$Species), percentage=percentage)
           freq percentage
setosa       50   33.33333
versicolor   50   33.33333
virginica    50   33.33333
> summary(dataset)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
> x <- dataset[,1:4]
> y <- dataset[,5]
> plot(y)
```



```
data(iris)
dataset<-iris
dim(dataset)
sapply(dataset, class)
head(dataset)
levels(dataset$Species)
percentage <- prop.table(table(dataset$Species)) * 100
cbind(freq=table(dataset$Species), percentage=percentage)
summary(dataset)
x <- dataset[,1:4]
y <- dataset[,5]
plot(y)
par(mfrow=c(1,4))
for(i in 1:4) {
  boxplot(x[,i], main=names(iris)[i])
}
```

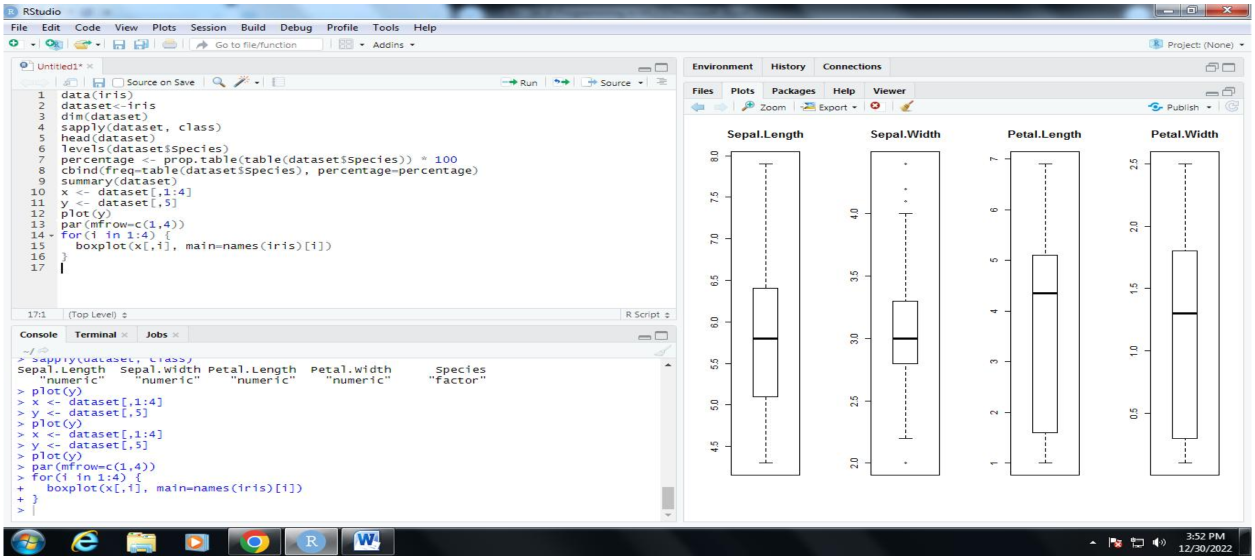
```
> dataset<-iris
> dim(dataset)
[1] 150   5
> sapply(dataset, class)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
   "numeric"    "numeric"    "numeric"    "numeric"     "factor"
> plot(y)
> x <- dataset[,1:4]
> y <- dataset[,5]
> plot(y)
> x <- dataset[,1:4]
> y <- dataset[,5]
> plot(y)
>
```

>par(mfrow=c(1,4))

for(i in 1:4) {

boxplot(x[,i], main=names(iris)[i])

}



```
data(iris)
dataset<-iris
dim(dataset)
sapply(dataset, class)
head(dataset)
levels(dataset$Species)
percentage <- prop.table(table(dataset$Species)) * 100
cbind(freq=table(dataset$Species), percentage=percentage)
summary(dataset)
x <- dataset[,1:4]
y <- dataset[,5]
plot(y)
par(mfrow=c(1,4))
for(i in 1:4) {
  boxplot(x[,i], main=names(iris)[i])
}
```

```
> sapply(dataset, class)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
   "numeric"    "numeric"    "numeric"    "numeric"     "factor"
> plot(y)
> x <- dataset[,1:4]
> y <- dataset[,5]
> plot(y)
> x <- dataset[,1:4]
> y <- dataset[,5]
> plot(y)
> par(mfrow=c(1,4))
> for(i in 1:4) {
+   boxplot(x[,i], main=names(iris)[i])
+ }
> 
```

Result

Thus, R scripts have been written and executed successfully.

Package Installation Procedure

Package Installation in Pycharm IDE:

****Follow the instructions given below to install python packages like sklearn, numpy, pandas, matplotlib etc.,**

--Go to File menu → Select “Settings”.

--Select “Project Interpreter” under the project name at the left part of the opened window.

--Click “+” symbol at the right part of the opened window.

--Specify the package name eg. Sklearn in the search window.

--Click “Install package” to install the specified package.

--Once package is installed, you can see the message “Package Installed Successfully” at the bottom of the opened window.