**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM**

**HORMIS NAGAR, MOOKKANNOOR**

#### ANGAMALY-683577

‘**FOCUS ON EXCELLENCE’ DATA SCIENCE**

………………………………………………

#### LABORATORY RECORD

**Name: KRISHNAJA UNNIKRISHNAN**

**Branch: MASTER OF COMPUTER APPLICATION**

**Semester: 3 Batch: B Roll No: 10**



#### FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)TM

**HORMIS NAGAR, MOOKKANNOOR**

#### ANGAMALY-683577

‘**FOCUS ON EXCELLENCE’**

**Name : KRISHNAJA UNNIKRISHNAN**

**Branch : MASTER OF COMPUTER APPLICATION**

**Semester : 3 Roll No: 10**

**University Exam.Reg. No:**

### CERTIFICATE

*This is to certify that this is a Bonafide record of the Practical work done and submitted to Kerala Technological University in partial fulfillment for the award of the Master Of Computer Applications is a record of the original research work done by* ***KRISHNAJA UNNIKRISHNAN i****n the* ***DATA SCIENCE*** *Laboratory of the Federal Institute of Science and Technology during the academic year 2021-2022.*

Signature of Staff in Charge Signature of H.O.D

Name: Name:

Date:

**Date of University practical examination ………………………**

Signature of Signature of

Internal Examiner External Examiner

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CONTENT** | | | | | |
| **SI**  **No** | **Date :** | **Name of Experiment:** | **Page No:** | **Signature of Staff –In –**  **Charge:** |  |
| 1 | 19/11/2021 | Matrix operations (using vectorization) and  transformation using python and SVD using Python | 1 |  |
| 2 | 30/11/2021 | Programs using matplotlib / plotly / bokeh / seaborn for data visualisation. | 4 |  |
| 3 | 26/11/2021 | Programs to handle data using pandas | 16 |  |
| 4 | 07/12/2021 | Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm. | 23 |  |
| 5 | 14/12/2021 | Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm | 37 |  |
| 6 | 04/01/2022 | Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its performance. | 44 |  |
| 7 | 14/01/2022 | Program to implement text classification using Support vector machine. | 49 |  |
|  | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | 21/12/2022 | Program to implement decision trees using any standard dataset available in the public domain and find the  accuracy of the algorithm | 61 |  |
| 9 | 25/01/2022 | Program to implement k-means clustering technique using any standard dataset available in the  public domain | 76 |  |
| 10 | 28/01/2022 | Programs on feedforward network to classify any standard dataset available in the public domain | 88 |  |
| 11 | 01/02/2022 | Programs on convolutional neural network to classify images from any standard dataset in the public domain. | 90 |  |
| 12 | 04/02/2022 | Implement problems on natural language processing - Part of Speech tagging, N-gram & smoothening and Chunking using NLTK | 93 |  |
| 13 | 08/02/2022 | Web Scrap | 99 |  |
| 14 | 11/02/2022 | Implement a simple web crawler (ensure ethical conduct). | 101 |  |
|  |  |  |  |  |

DEPARTMENT OF COMPUTER APPLICATION

Page | 1

**AIM**

**1: Matrix operations(using vectorixation) and transformation using python and SVD.**

#### CODE:

a = np.arange(0,4).reshape((2,2)) b = np.eye(2)

print(np.dot(a,b)) ##Matrix multiplication

#### OUTPUT:

[[0. 1.]

[2. 3.]]

**CODE:**

x = np.arange(1,10).reshape(3,3) print(x)

#### OUTPUT:

[[1 2 3]

[4 5 6]

[7 8 9]]

#### CODE:

#SVD image compresion

import matplotlib.pyplot as plt import matplotlib.image as mpimg import numpy as np

img\_eg = mpimg.imread("rose.jpg") plt.imshow(img\_eg)

print(img\_eg.shape) #Operation results: (800, 1280,3)

#Converting image data into two-dimensional matrix and singu- lar value decomposition

img\_temp = img\_eg.reshape(800, 1280 \* 3) U,Sigma,VT = np.linalg.svd(img\_temp)

# Take the first 10 singular values sval\_nums = 10

DEPARTMENT OF COMPUTER APPLICATION

Page | 2

img\_re-

struct1 = (U[:,0:sval\_nums]).dot(np.diag(Sigma[0:sval\_nums])).dot(VT[0: sval\_nums,:])

img\_restruct1 = img\_restruct1.reshape(800, 1280,3) img\_restruct1.tolist()

# Take the first 50 singular values sval\_nums = 50

img\_re-

struct2 = (U[:,0:sval\_nums]).dot(np.diag(Sigma[0:sval\_nums])).dot(VT[0: sval\_nums,:])

img\_restruct2 = img\_restruct2.reshape(800, 1280,3)

# Take the first 100 singular values sval\_nums = 100

img\_re-

struct3 = (U[:,0:sval\_nums]).dot(np.diag(Sigma[0:sval\_nums])).dot(VT[0: sval\_nums,:])

img\_restruct3 = img\_restruct3.reshape(800, 1280,3)

#Exhibition

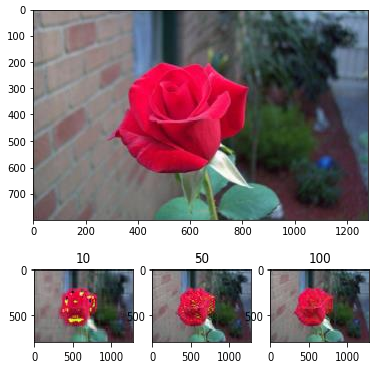
fig, ax = plt.subplots(nrows=1, ncols=3) ax[0].imshow(img\_restruct1.astype(np.uint8)) ax[0].set(title = "10") ax[1].imshow(img\_restruct2.astype(np.uint8)) ax[1].set(title = "50") ax[2].imshow(img\_restruct3.astype(np.uint8)) ax[2].set(title = "100")

plt.show()

DEPARTMENT OF COMPUTER APPLICATION

Page | 3

OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 4

AIM:

1. **Programs using matplotlib / plotly / bokeh / seaborn for data visualisation.**

**Dataset used: iris.csv**

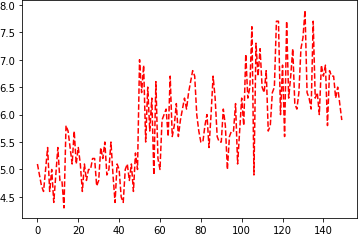
#### CODE:

import pandas as pd

iris = pd.read\_csv(‘iris.csv’)

## Plotting Using Matplotlib import matplotlib.pyplot as plt plt.plot(iris["sepal.length"], "r--") plt.show

#### OUTPUT:



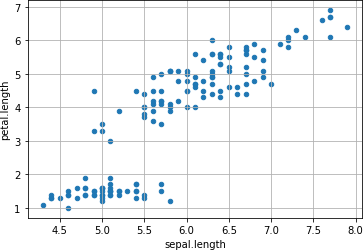
**CODE:**

## Scatter Plot

iris.plot(kind ="scatter", x ='sepal.length', y ='petal.length')

plt.grid()

#### OUTPUT:



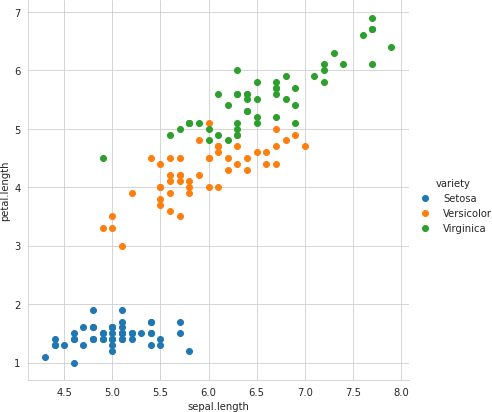
**CODE:**

## Plotting using Seaborn

import seaborn as sns sns.set\_style("whitegrid")

sns.FacetGrid(iris, hue ="variety",height = 6).map(plt.scatter, 'sepal.length', 'petal.length').add\_legend()

#### OUTPUT:

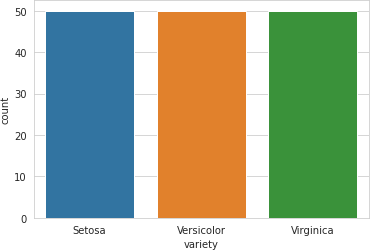


**CODE:**

# Distribution Chart

#Visualizing the target(class label) column sns.countplot(x='variety', data=iris, ) plt.show()

#### OUTPUT:

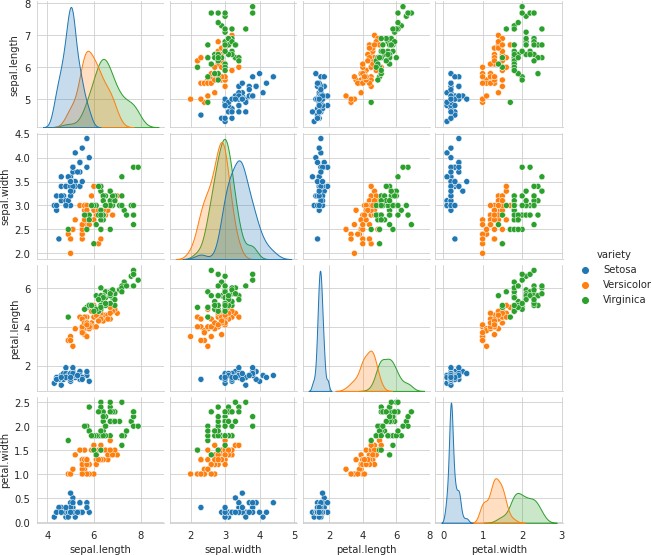


**CODE:**

#plotting all the column’s relationships using a pairplot. It can be used for multivariate analysis.

sns.pairplot(iris,hue='variety', height=2)

#### OUTPUT:



**CODE:**

#Histogram for Sepal Length

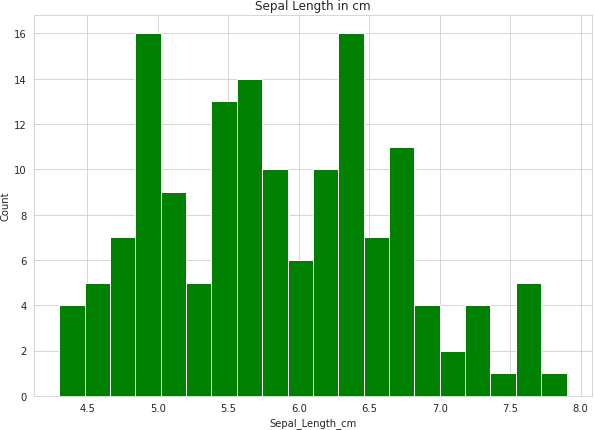
plt.figure(figsize = (10, 7)) x = iris["sepal.length"]

plt.hist(x, bins = 20, color = "green") plt.title("Sepal Length in cm") plt.xlabel("Sepal\_Length\_cm") plt.ylabel("Count")

DEPARTMENT OF COMPUTER APPLICATION

Page | 8

#### OUTPUT:



**CODE:**

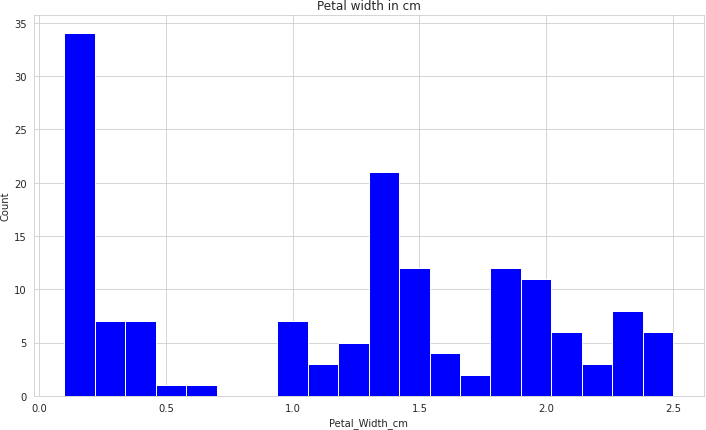
#Histogram for Petal Width plt.figure(figsize = (12, 7)) x = iris["petal.width"]

plt.hist(x, bins =20, color = "blue") plt.title("Petal width in cm") plt.xlabel("Petal\_Width\_cm") plt.ylabel("Count")

DEPARTMENT OF COMPUTER APPLICATION

Page | 9

#### OUTPUT:



**CODE:**

#Histograms allow seeing the distribution of data for various columns. # It can be used for uni as well as bi-variate analysis.

fig, axes = plt.subplots(2, 2, figsize=(10,10))

axes[0,0].set\_title("Sepal Length") axes[0,0].hist(iris['sepal.length'], bins=7)

axes[0,1].set\_title("Sepal Width") axes[0,1].hist(iris['sepal.width'], bins=5);

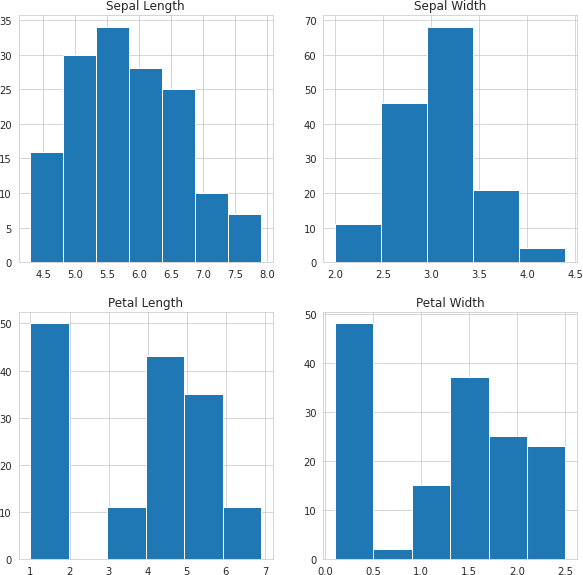
axes[1,0].set\_title("Petal Length") axes[1,0].hist(iris['petal.length'], bins=6);

axes[1,1].set\_title("Petal Width") axes[1,1].hist(iris['petal.width'], bins=6);

DEPARTMENT OF COMPUTER APPLICATION

Page | 10

#### OUTPUT:



**CODE:**

#Histograms with Distplot Plot

plot = sns.FacetGrid(iris, hue="variety") plot.map(sns.distplot, "sepal.length").add\_legend()

plot = sns.FacetGrid(iris, hue="variety") plot.map(sns.distplot, "sepal.width").add\_legend()

plot = sns.FacetGrid(iris, hue="variety") plot.map(sns.distplot, "petal.length").add\_legend()

DEPARTMENT OF COMPUTER APPLICATION

Page | 11

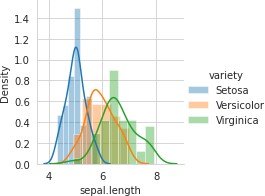
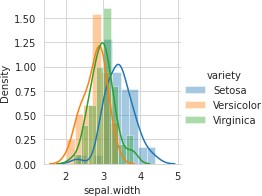
plot = sns.FacetGrid(iris, hue="variety") plot.map(sns.distplot, "petal.width").add\_legend()

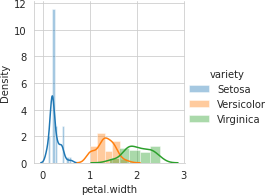
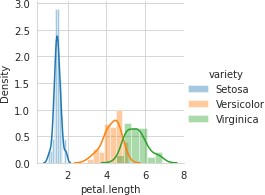
plt.show()

#In the case of Sepal Length, there is a huge amount of overlapping. #In the case of Sepal Width also, there is a huge amount of overlapping.

#In the case of Petal Length, there is a very little amount of overlapping. #In the case of Petal Width also, there is a very little amount of overlapping.

#### OUTPUT:





DEPARTMENT OF COMPUTER APPLICATION

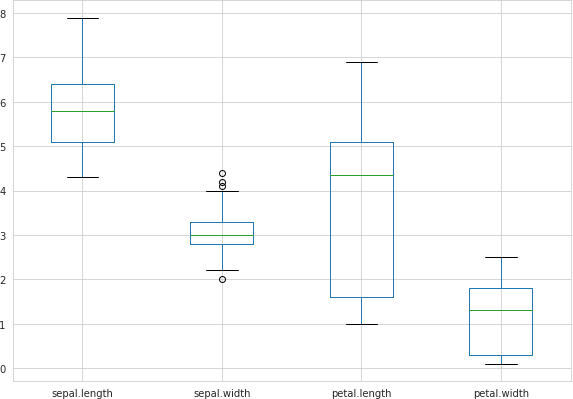
Page | 12

#### CODE:

# Box Plot for Iris Data

plt.figure(figsize = (10, 7)) iris.boxplot()

#### OUTPUT:



**CODE:**

import matplotlib.gridspec as gridspec fig = plt.figure(figsize=(9, 40))

outer = gridspec.GridSpec(4, 1, wspace=0.2, hspace=0.2) for i, col in enumerate(iris.columns[:-1]):

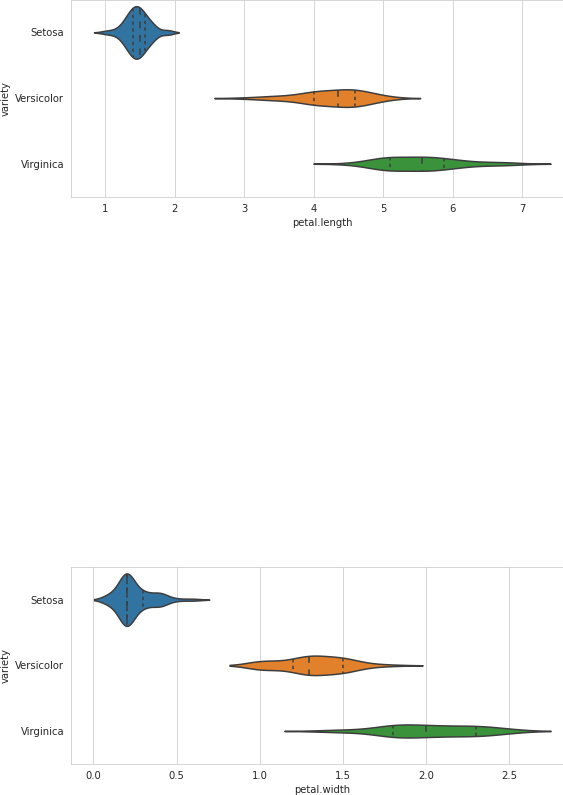
inner = gridspec.GridSpecFromSubplotSpec(2, 1,subplot\_spec=outer[i], wspace=0.2, hspace=0.4)

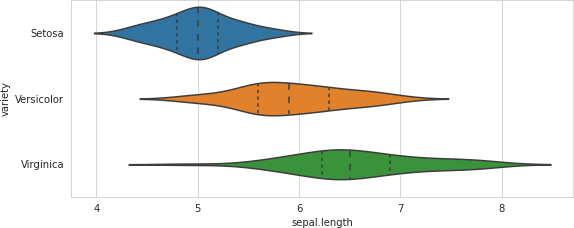
ax = plt.Subplot(fig, inner[1])

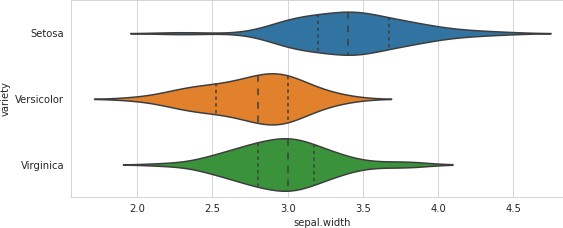
\_ = sns.violinplot(y="variety", x=f"{col}", data=iris, inner='quartile', ax=ax) fig.add\_subplot(ax)

fig.show()

#### OUTPUT:



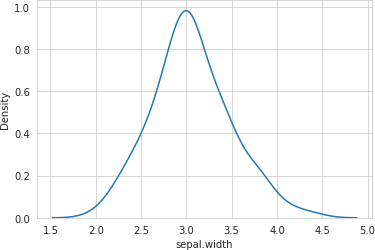




**CODE:**

# Make default density plot sns.kdeplot(iris['sepal.width'])

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 16

#### AIM:

1. **Programs to handle data using pandas.**

#### CODE:

#Pandas is a Python library. #Pandas is used to analyze data. import numpy as np

import pandas as pd

s = pd.Series([1, 3, 5, 6, 8]) print(s)

#### OUTPUT:

|  |  |
| --- | --- |
| 0 | 1 |
| 1 | 3 |
| 2 | 5 |
| 3 | 6 |
| 4 | 8 |

dtype: int64

#### CODE:

dict = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],

"capital": ["Brasilia", "Moscow", "New Dehli", "Beijing", "Pretoria"],

"area": [8.516, 17.10, 3.286, 9.597, 1.221],

"population": [200.4, 143.5, 1252, 1357, 52.98] }

b = pd.DataFrame(dict) print(b)

#### OUTPUT:

country capital area population

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | Brazil | Brasilia | 8.516 | 200.40 |
| 1 | Russia | Moscow | 17.100 | 143.50 |
| 2 | India | New Dehli | 3.286 | 1252.00 |
| 3 | China | Beijing | 9.597 | 1357.00 |
| 4 | South Africa | Pretoria | 1.221 | 52.98 |

#### CODE:

b.index = ["BR", "RU", "IN", "CH", "SA"]

DEPARTMENT OF COMPUTER APPLICATION

Page | 17

print(b)

#### OUTPUT:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | country | capital | area | population |
| BR | Brazil | Brasilia | 8.516 | 200.40 |
| RU | Russia | Moscow | 17.100 | 143.50 |
| IN | India | New Dehli | 3.286 | 1252.00 |
| CH | China | Beijing | 9.597 | 1357.00 |
| SA | South Africa | Pretoria | 1.221 | 52.98 |

**CODE:**

import pandas as pd

cars = pd.read\_csv('cars1.csv') print(cars)

#### OUTPUT:

Car Model Volume Weight CO2

1. Toyoty Aygo 1000 790 99
2. Mitsubishi Space Star 1200 1160 95
3. Skoda Citigo 1000 929 95 3 Fiat 500 900 865 90 4 Mini Cooper 1500 1140 105 5 VW Up! 1000 929 105 6 Skoda Fabia 1400 1109 90

7 Mercedes A-Class 1500 1365 92 8 Ford Fiesta 1500 1112 98 9 Audi A1 1600 1150 99 10 Hyundai I20 1100 980 99

1. Suzuki Swift 1300 990 101
2. Ford Fiesta 1000 1112 99
3. Honda Civic 1600 1252 94 14 Hundai I30 1600 1326 97 15 Opel Astra 1600 1330 97 16 BMW 1 1600 1365 99

17 Mazda 3 2200 1280 104

18 Skoda Rapid 1600 1119 104

19 Ford Focus 2000 1328 105

1. Ford Mondeo 1600 1584 94
2. Opel Insignia 2000 1428 99
3. Mercedes C-Class 2100 1365 99
4. Skoda Octavia 1600 1415 99 24 Volvo S60 2000 1415 99

25 Mercedes CLA 1500 1465 102 26 Audi A4 2000 1490 104

27 Audi A6 2000 1725 114

28 Volvo V70 1600 1523 109

29 BMW 5 2000 1705 114

30 Mercedes E-Class 2100 1605 115 31 Volvo XC70 2000 1746 117 32 Ford B-Max 1600 1235 104 33 BMW 216 1600 1390 108

DEPARTMENT OF COMPUTER APPLICATION

Page | 18

#### CODE:

import pandas as pd

cars = pd.read\_csv('cars1.csv') cars = pd.read\_csv('/cars1.csv') print(cars)

# Print out first 4 observations print(cars[0:4])

# Print out fifth and sixth observation print(cars[4:6])

import pandas as pd

cars = pd.read\_csv('cars1.csv', index\_col = 0) #first column is taen as index column

print(cars.iloc[2])

#### OUTPUT:

Model Citigo

Volume 1000

Weight 929

CO2 95

Name: Skoda, dtype: object

#### CODE:

#Slicing dataframe import pandas as pd

df = pd.DataFrame([['Jay','M',18],['Jennifer','F',17],

['Preity','F',19],['Neil','M',17]],

columns = ['Name','Gender','Age'])

print(df)

df1 = df.iloc[2:,:]

df2 = df.iloc[:2,:] print(df1) print(df2)

#### OUTPUT:

|  |  |  |
| --- | --- | --- |
|  | Name Gender | Age |
| 0 | Jay | M 18 |
| 1 | Jennifer | F 17 |
| 2 | Preity | F 19 |
| 3 | Neil | M 17 |

DEPARTMENT OF COMPUTER APPLICATION

Page | 19

Name Gender Age

1. Preity F 19
2. Neil M 17

Name Gender Age

1. Jay M 18
2. Jennifer F 17

#### CODE:

import pandas as pd import numpy as np

#Create a series with 4 random numbers s = pd.Series(np.random.randn(4)) print(s)

print ("The actual data series is:") print( s.values)

#### OUTPUT:

0 -1.138968

1 -1.097746

2 0.109717

3 1.159537

dtype: float64

The actual data series is:

[-1.13896826 -1.09774589 0.10971687 1.15953676]

CodeText

#### CODE:

print (s.head(2))

#### OUTPUT:

0 -1.138968

1 -1.097746

dtype: float64

#### CODE:

print(s.tail(3))

DEPARTMENT OF COMPUTER APPLICATION

Page | 20

#### OUTPUT:

1 -1.097746

2 0.109717

3 1.159537

dtype: float64

#### CODE:

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']), 'Age':pd.Series([25,26,25,23,30,29,23]), 'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

# Create a DataFrame df = pd.DataFrame(d) print(df)

print ("The transpose of the data series is:") print(df.T)

#### OUTPUT:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Name | Age | Rating | | | | |
| 0 | Tom | 25 | 4.23 | | | | |
| 1 | James | 26 | 3.24 | | | | |
| 2 | Ricky | 25 | 3.98 | | | | |
| 3 | Vin | 23 | 2.56 | | | | |
| 4 | Steve | 30 | 3.20 | | | | |
| 5 | Smith | 29 | 4.60 | | | | |
| 6 | Jack | 23 | 3.80 | | | | |
| The transpose | | | of the | data series | is: |  |  |
| 0 | | | 1 | 2 3 | 4 | 5 | 6 |
| Name Tom | | | James | Ricky Vin | Steve | Smith | Jack |
| Age 25 | | | 26 | 25 23 | 30 | 29 | 23 |
| Rating 4.23 | | | 3.24 | 3.98 2.56 | 3.2 | 4.6 | 3.8 |
| **CODE:** | | |  |  |  |  |  |

import pandas as pd import numpy as np

#Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']), 'Age':pd.Series([25,26,25,23,30,29,23]), 'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

#Create a DataFrame df = pd.DataFrame(d) print(df)

print ("Row axis labels and column axis labels are:")

DEPARTMENT OF COMPUTER APPLICATION

Page | 21

print (df.axes)

#### OUTPUT:

Name Age Rating

|  |  |  |
| --- | --- | --- |
| 0 | Tom 25 | 4.23 |
| 1 | James 26 | 3.24 |
| 2 | Ricky 25 | 3.98 |
| 3 | Vin 23 | 2.56 |
| 4 | Steve 30 | 3.20 |
| 5 | Smith 29 | 4.60 |
| 6 | Jack 23 | 3.80 |

Row axis labels and column axis labels are: [RangeIndex(start=0, stop=7, step=1), Index(['Name', 'Age', 'Rating'], dtype='object')]

#### CODE:

import pandas as pd import numpy as np

#Create a Dictionary of series

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Smith','Jack']), 'Age':pd.Series([25,26,25,23,30,29,23]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])

}

#Create a DataFrame df = pd.DataFrame(d) print ("Our object is:") print (df)

print ("The dimension of the object is:") print (df.ndim)

#### OUTPUT:

Name Age Rating

|  |  |  |
| --- | --- | --- |
| 0 | Tom 25 | 4.23 |
| 1 | James 26 | 3.24 |
| 2 | Ricky 25 | 3.98 |
| 3 | Vin 23 | 2.56 |
| 4 | Steve 30 | 3.20 |
| 5 | Smith 29 | 4.60 |
| 6 | Jack 30 | 3.80 |

Our object is:

The shape of the object is: (7, 3)

#### CODE:

print (df.size)

DEPARTMENT OF COMPUTER APPLICATION

Page | 22

#### OUTPUT:

21

#### CODE:

print (df.values)

#### OUTPUT:

[['Tom' 25 4.23]

['James' 26 3.24]

['Ricky' 25 3.98]

['Vin' 23 2.56]

['Steve' 30 3.2]

['Smith' 29 4.6]

['Jack' 30 3.8]]

#### CODE:

df.isnull().sum() #sum returns the number of missing values

#### OUTPUT:

Name 0

Age 0

Rating 0

dtype: int64

#### CODE:

df = pd.DataFrame(np.arange(12).reshape(3, 4), columns=['A', 'B', 'C', 'D']) print(df)

#### OUTPUT:

|  |  |  |
| --- | --- | --- |
| A B | C | D |
| 0 0 1 | 2 | 3 |
| 1 4 5 | 6 | 7 |
| 2 8 9 | 10 | 11 |

DEPARTMENT OF COMPUTER APPLICATION

Page | 23

#### AIM

**4: Program to implement k-NN classification using any standard dataset available in the public domain and find the accuracy of the algorithm.**

**Dataset used: iris.csv**

#### CODE:

from sklearn.neighbors import KNeighborsClassifier from sklearn.model\_selection import train\_test\_split from sklearn.metrics import classification\_report import pandas as pd

df = pd.read\_csv("iris.csv") print(df)

#### OUTPUT:

sepal.length sepal.width petal.length petal.width variety

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Setosa |
| .. | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Virginica |
| 149  [150 rows x 5 | 5.9  columns] | 3.0 | 5.1 | 1.8 | Virginica |
| **CODE:** |  |  |  |  |  |

df['variety'].value\_counts()

|  |  |
| --- | --- |
| **OUTPUT:** |  |
| Setosa | 50 |
| Versicolor | 50 |
| Virginica | 50 |

Name: variety, dtype: int64

#### CODE:

X = df.drop('variety', axis=1) y = df['variety']

# splitting to trainset and Test set in the ratio 70:30

DEPARTMENT OF COMPUTER APPLICATION

Page | 24

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

print(X\_train) print(" ") print(X\_test)

#### OUTPUT:

sepal.length sepal.width petal.length petal.width

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 46 |  | 5.1 | | 3.8 | 1.6 | 0.2 |
| 95 |  | 5.7 | | 3.0 | 4.2 | 1.2 |
| 67 |  | 5.8 | | 2.7 | 4.1 | 1.0 |
| 45 |  | 4.8 | | 3.0 | 1.4 | 0.3 |
| 143  ..  116 |  | 6.8  ...  6.5 | | 3.2  ...  3.0 | 5.9  ...  5.5 | 2.3  ...  1.8 |
| 41 |  | 4.5 | | 2.3 | 1.3 | 0.3 |
| 62 |  | 6.0 | | 2.2 | 4.0 | 1.0 |
| 91 |  | 6.1 | | 3.0 | 4.6 | 1.4 |
| 123 |  | 6.3 | | 2.7 | 4.9 | 1.8 |
| [105 | rows x 4 | columns] | |  |  |  |
|  | sepal.length | | sepal.width | | petal.length | petal.width |
| 25 | 5.0 | | 3.0 | | 1.6 | 0.2 |
| 141 | 6.9 | | 3.1 | | 5.1 | 2.3 |
| 125 | 7.2 | | 3.2 | | 6.0 | 1.8 |
| 102 | 7.1 | | 3.0 | | 5.9 | 2.1 |
| 128 | 6.4 | | 2.8 | | 5.6 | 2.1 |
| 122 | 7.7 | | 2.8 | | 6.7 | 2.0 |
| 76 | 6.8 | | 2.8 | | 4.8 | 1.4 |
| 103 | 6.3 | | 2.9 | | 5.6 | 1.8 |
| 14 | 5.8 | | 4.0 | | 1.2 | 0.2 |
| 37 | 4.9 | | 3.6 | | 1.4 | 0.1 |
| 100 | 6.3 | | 3.3 | | 6.0 | 2.5 |
| 63 | 6.1 | | 2.9 | | 4.7 | 1.4 |
| 64 | 5.6 | | 2.9 | | 3.6 | 1.3 |
| 61 | 5.9 | | 3.0 | | 4.2 | 1.5 |
| 17 | 5.1 | | 3.5 | | 1.4 | 0.3 |
| 74 | 6.4 | | 2.9 | | 4.3 | 1.3 |
| 111 | 6.4 | | 2.7 | | 5.3 | 1.9 |
| 120 | 6.9 | | 3.2 | | 5.7 | 2.3 |
| 79 | 5.7 | | 2.6 | | 3.5 | 1.0 |
| 85 | 6.0 | | 3.4 | | 4.5 | 1.6 |
| 49 | 5.0 | | 3.3 | | 1.4 | 0.2 |
| 21 | 5.1 | | 3.7 | | 1.5 | 0.4 |
| 110 | 6.5 | | 3.2 | | 5.1 | 2.0 |
| 149 | 5.9 | | 3.0 | | 5.1 | 1.8 |
| 72 | 6.3 | | 2.5 | | 4.9 | 1.5 |
| 11 | 4.8 | | 3.4 | | 1.6 | 0.2 |
| 36 | 5.5 | | 3.5 | | 1.3 | 0.2 |
| 6 | 4.6 | | 3.4 | | 1.4 | 0.3 |
| 68 | 6.2 | | 2.2 | | 4.5 | 1.5 |
| 144 | 6.7 | | 3.3 | | 5.7 | 2.5 |
| 43 | 5.0 | | 3.5 | | 1.6 | 0.6 |
| 80 | 5.5 | | 2.4 | | 3.8 | 1.1 |
| 32 | 5.2 | | 4.1 | | 1.5 | 0.1 |

DEPARTMENT OF COMPUTER APPLICATION

Page | 25

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 5.0 | 3.4 | 1.5 | 0.2 |
| 55 | 5.7 | 2.8 | 4.5 | 1.3 |
| 129 | 7.2 | 3.0 | 5.8 | 1.6 |
| 117 | 7.7 | 3.8 | 6.7 | 2.2 |
| 12 | 4.8 | 3.0 | 1.4 | 0.1 |
| **CODE:** |  |  |  |  |

print("Number transactions X\_train dataset: ", X\_train.shape) print("Number transactions y\_train dataset: ", y\_train.shape) print("Number transactions X\_test dataset: ", X\_test.shape) print("Number transactions y\_test dataset: ", y\_test.shape)

#### OUTPUT:

Number transactions X\_train dataset: (105, 4) Number transactions y\_train dataset: (105,) Number transactions X\_test dataset: (45, 4) Number transactions y\_test dataset: (45,)

#### CODE:

classifier = KNeighborsClassifier(n\_neighbors=5) classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test) print(y\_pred)

print(' ') print(y\_test)

#### OUTPUT:

['Setosa' 'Virginica''Virginica''Virginica''Virginica''Virginica' 'Versicolor''Virginica''Setosa''Setosa''Virginica' 'Versicolor' 'Versicolor''Versicolor''Setosa''Versicolor''Virginica''Virginica

'Versicolor''Versicolor''Setosa''Setosa' 'Virginica''Virginica' 'Virginica''Setosa''Setosa''Setosa''Versicolor''Virginica''Setosa

'Setosa' 'Virginica' 'Versicolor' 'Setosa' 'Setosa' 'Virginica' 'Versicolor''Virginica''Versicolor''Virginica''Setosa''Virginica'

'Virginica' 'Setosa']

1. Versicolor
2. Versicolor

61 Versicolor

17 Setosa

74 Versicolor

111 Virginica

120 Virginica

79 Versicolor

85 Versicolor

49 Setosa

21 Setosa

110 Virginica

149 Virginica

DEPARTMENT OF COMPUTER APPLICATION

Page | 26

|  |  |  |
| --- | --- | --- |
| 72 | Versicolor |  |
| 11 | Setosa |  |
| 36 | Setosa |  |
| 6 | Setosa |  |
| 68 | Versicolor |  |
| 144 | Virginica |  |
| 43 | Setosa |  |
| 47 | Setosa |  |
| 77 | Versicolor |  |
| 80 | Versicolor |  |
| 32 | Setosa |  |
| 7 | Setosa |  |
| 148 | Virginica |  |
| 88 | Versicolor |  |
| 137 | Virginica |  |
| 55 | Versicolor |  |
| 112 | Virginica |  |
| 29 | Setosa |  |
| 129 | Virginica |  |
| 117 | Virginica |  |
| 12  Name: | Setosa  variety, dtype: | object |

#### CODE:

from sklearn.metrics import confusion\_matrix print(confusion\_matrix(y\_test, y\_pred)) print(classification\_report(y\_test, y\_pred))

#### OUTPUT:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [[15 0 | 0] | | | |
| [ 0 11 | 2] | | | |
| [ 0 0 | 17]] | | | |
| precision | | recall | f1-score | support |
| Setosa 1.00 | | 1.00 | 1.00 | 15 |
| Versicolor 1.00 | | 0.85 | 0.92 | 13 |
| Virginica 0.89 | | 1.00 | 0.94 | 17 |
| accuracy | |  | 0.96 | 45 |
| macro avg 0.96 | | 0.95 | 0.95 | 45 |
| weighted avg 0.96 | | 0.96 | 0.95 | 45 |
| **CODE:** | |  |  |  |

weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy',

'Over cast','Sunny','Sunny', 'Rainy','Sunny','Overcast','Over- cast','Rainy']

# Second Feature

DEPARTMENT OF COMPUTER APPLICATION

Page | 27

temp=['Hot','Hot','Hot','Mild','Cool','Cool','Cool','Mild', 'Cool'

,'Mild','Mild','Mild','Hot','Mild'] #

Label or target varible

play=['No','No','Yes','Yes','Yes','No','Yes','No','Yes','Yes', 'Ye s','Yes','Yes','No']

from sklearn import preprocessing #creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers. weather\_encoded=le.fit\_transform(weather) print(weather\_encoded)

**OUTPUT:**

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

#### CODE:

temp\_encoded=le.fit\_transform(temp) print(temp\_encoded) print(" ") label=le.fit\_trans-

form(play) print(label)

#### OUTPUT:

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

[0 0 1 1 1 0 1 0 1 1 1 1 1 0]

DEPARTMENT OF COMPUTER APPLICATION

Page | 28

#### CODE:

features=list(zip(weather\_encoded,temp\_encoded)) print(features)

**OUTPUT:**

[(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2),

(2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]

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

[0 0 1 1 1 0 1 0 1 1 1 1 1 0]

DEPARTMENT OF COMPUTER APPLICATION

Page | 29

#### CODE:

features=list(zip(weather\_encoded,temp\_encoded)) print(features)

#### OUTPUT:

[(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (2, 2),

(2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]

#### CODE:

from sklearn.neighbors import KNeighborsClassifier model = KNeighborsClassifier(n\_neighbors=3)

from sklearn.neighbors import KNeighborsClassifier model = KNeighborsClassifier(n\_neighbors=3)

# Train the model using the training sets model.fit(features,label)

predicted= model.predict([[0,1]]) # 0:Overcast, 1:Hot print(predicted)

#### OUTPUT:

[1]

DEPARTMENT OF COMPUTER APPLICATION

Page | 30

#### CODE:

**Dataset used: Fruit\_classification.csv**

import warnings warnings.filterwarnings('ignore') import numpy as np

import pandas as pd

import matplotlib.pyplot as plt fruits=pd.read\_table('/content/fruit\_data\_with\_colors.txt') fruits.head()

#### OUTPUT:

fruit\_label fruit\_name fruit\_subtype mass width height color\_score

1. 1 apple granny\_smith 192 8.4 7.3

0.55

1. 1 apple granny\_smith 180 8.0 6.8

0.59

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | apple | granny\_smith | 176 | 7.4 | 7.2 |
| 0.60 |  |  |  |  |  |  |
| 3 | 2 | mandarin | mandarin | 86 | 6.2 | 4.7 |
| 0.80 |  |  |  |  |  |  |
| 4 | 2 | mandarin | mandarin | 84 | 6.0 | 4.6 |
| 0.79 |  |  |  |  |  |  |

#### CODE:

fruits.shape

#### OUTPUT:

(59, 7)

#### CODE:

predct = dict(zip(fruits.fruit\_label.unique(), fruits.fruit\_name.unique())) predct

#### OUTPUT:

{1: 'apple', 2: 'mandarin', 3: 'orange', 4: 'lemon'}

#### CODE:

fruits['fruit\_name'].value\_counts()

#### OUTPUT:

apple 19

orange 19

lemon 16

mandarin 5

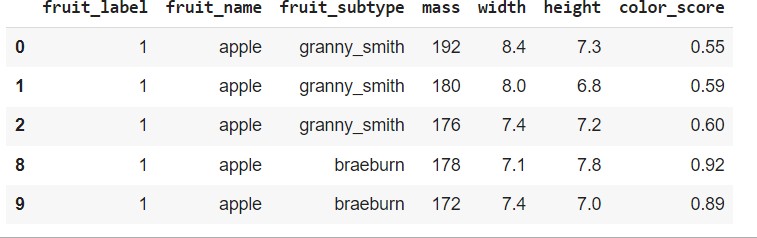
Name: fruit\_name, dtype: int64

#### CODE:

apple\_data=fruits[fruits['fruit\_name']=='apple'] orange\_data=fruits[fruits['fruit\_name']=='orange'] lemon\_data=fruits[fruits['fruit\_name']=='lemon'] mandarin\_data=fruits[fruits['fruit\_name']=='mandarin']

apple\_data.head()

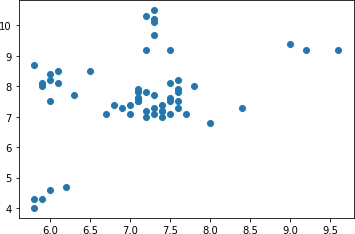
#### OUTPUT:



**CODE:**

plt.scatter(fruits['width'],fruits['height'])

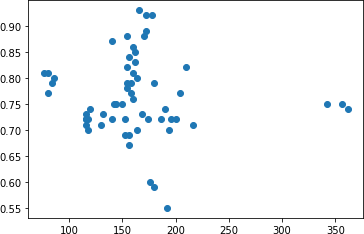
#### OUTPUT:



**CODE:**

plt.scatter(fruits['mass'],fruits['color\_score'])

#### OUTPUT:



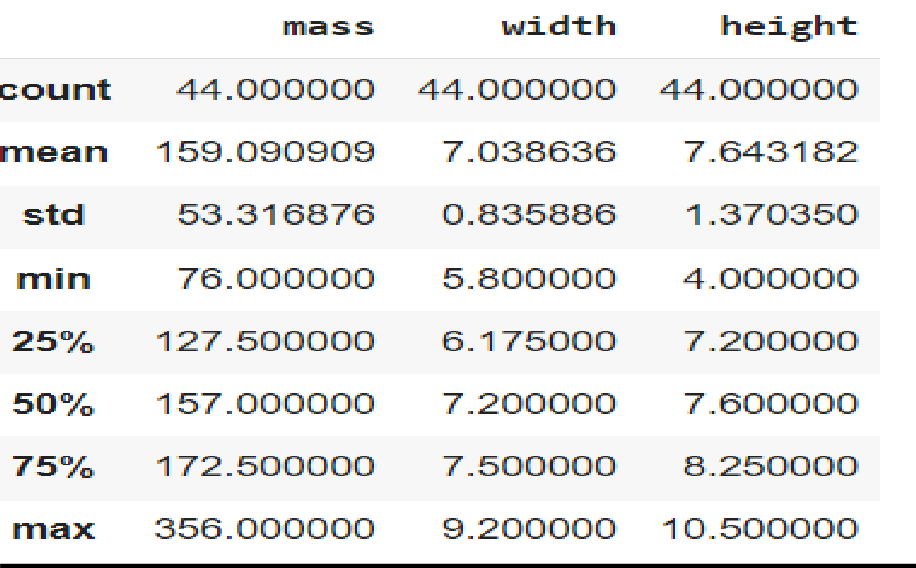
**CODE:**

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

X=fruits[['mass','width','height']] Y=fruits['fruit\_label']

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,Y,random\_state=0) X\_train.describe()

#### OUTPUT:



**CODE:**

X\_test.describe()

#### OUTPUT:



**CODE:**

knn=KNeighborsClassifier() knn.fit(X\_train,y\_train)

#### OUTPUT:

KNeighborsClassifier()

#### CODE:

knn.score(X\_test,y\_test)

#### OUTPUT:

0.5333333333333333

#### CODE:

prediction1=knn.predict([['100','6.3','8']]) predct[prediction1[0]]

#### OUTPUT:

lemon

#### CODE:

prediction2=knn.predict([['300','7','10']]) predct[prediction2[0]]

#### OUTPUT:

orange

DEPARTMENT OF COMPUTER APPLICATION

Page | 37

#### AIM

**5: Program to implement Naïve Bayes Algorithm using any standard dataset available in the public domain and find the accuracy of the algorithm.**

#### CODE:

**Dataset used: Social\_Network\_Ads.csv**

import pandas as pd

dataset = pd.read\_csv("/content/Social\_Network\_Ads.csv") print(dataset.describe())

print(dataset.head())

X = dataset.iloc[:, [1, 2, 3]].values y = dataset.iloc[:, -1].values

from sklearn.preprocessing import LabelEncoder

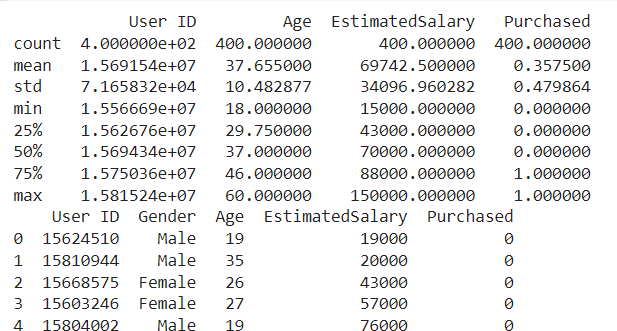
le = LabelEncoder()

X[:,0] = le.fit\_transform(X[:,0])

from sklearn.model\_selection import train\_test\_split X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_si ze = 0.20, random\_state = 0)

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 38

#### CODE:

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

from sklearn.naive\_bayes import GaussianNB classifier = GaussianNB() classi- fier.fit(X\_train, y\_train)

#### OUTPUT:

GaussianNB()

#### CODE:

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

#### OUTPUT:

array([0, 0, 1,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, | 1, | 0, |
| 1, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, |
| , 0, | 0, | 1, | 0, | 1, | 1, | 0, | 0, | 1, | 1, | 0, | 0, | 0, | 1, | 0, | 0, | 1, | 0, | 0, |

0, 0,

0, 1,

0,

1

0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])

#### CODE:

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

DEPARTMENT OF COMPUTER APPLICATION

Page | 39

#### OUTPUT:

array([0, 0, 1,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, |
| 1, | 0, | 1, | 0, | 0, | 0, | 0, | 0, | 1, | 1, | 0, | 0, | 0, | 0, | 0, | 0, | 1, | 0, | 0, |
| , 0, | 0, | 1, | 0, | 1, | 1, | 0, | 0, | 0, | 1, | 1, | 0, | 0, | 1, | 0, | 0, | 1, | 0, | 1, |

0, 0,

0, 1,

0,

1

0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])

DEPARTMENT OF COMPUTER APPLICATION

Page | 40

#### CODE:

from sklearn.metrics import confusion\_matrix,accuracy\_score cm = confusion\_matrix(y\_test, y\_pred)

ac = accuracy\_score(y\_test,y\_pred) print(cm)

print(ac)

**OUTPUT:**

## [[56 2]

[ 4 18]]

## 0.925

DEPARTMENT OF COMPUTER APPLICATION

Page | 41

#### Data set:Naïve\_base.csv CODE

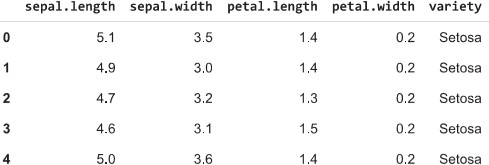
import numpy as np

import matplotlib.pyplot as plt import pandas as pd

df = pd.read\_csv("iris.csv")

X = df.iloc[:,:4].values y = df['variety'].values df.head(5)

#### OUTPUT



**CODE**

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2)

#### CODE

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

#### CODE

from sklearn.naive\_bayes import GaussianNB classifier = GaussianNB() classifier.fit(X\_train, y\_train)

#### OUTPUT

GaussianNB()

DEPARTMENT OF COMPUTER APPLICATION

Page | 42

#### CODE

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

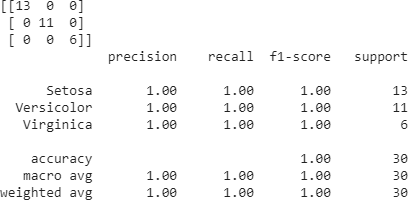
#### OUTPUT

array(['Versicolor', 'Versicolor', 'Versicolor', 'Setosa', 'Setosa', 'Setosa', 'Virginica', 'Versicolor', 'Setosa', 'Setosa', 'Setosa', 'Virginica', 'Virginica', 'Setosa', 'Versicolor', 'Virginica', 'Versicolor', 'Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Versicolor', 'Setosa', 'Setosa', 'Virginica', 'Setosa', 'Setosa', 'Versicolor', 'Virginica', 'Versicolor'], dtype='<U10')

#### CODE

from sklearn.metrics import confusion\_matrix from sklearn.metrics import classification\_report print(confusion\_matrix(y\_test, y\_pred)) print(classification\_report(y\_test, y\_pred))

#### OUTPUT



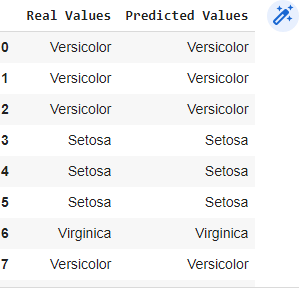
**CODE**

df\_result = pd.DataFrame({'Real Values':y\_test, 'Predicted Values':y\_pred}) df\_result

DEPARTMENT OF COMPUTER APPLICATION

Page | 43

#### OUTPUT



DEPARTMENT OF COMPUTER APPLICATION

Page | 44

#### AIM:

6: Program to implement linear and multiple regression techniques using any standard dataset available in the public domain and evaluate its

performance.

#### CODE:

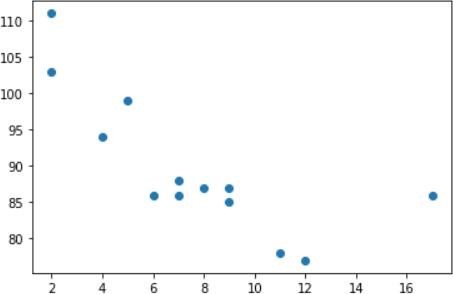
import matplotlib.pyplot as plt

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

plt.scatter(x, y) plt.show()

#### OUTPUT:



**CODE:**

import matplotlib.pyplot as plt from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

+slope, intercept, r, p, std\_err = stats.linregress(x, y) # r corre lation coefficiant

# p probability of hypothesis def myfunc(x):

DEPARTMENT OF COMPUTER APPLICATION

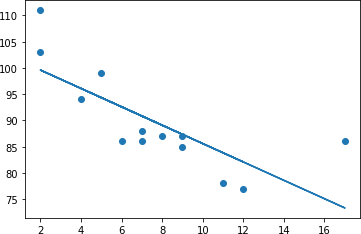
Page | 45

return slope \* x + intercept mymodel = list(map(myfunc, x))

plt.scatter(x, y) plt.plot(x, mymodel) plt.show()

#### OUTPUT:

-0.758591524376155



#### CODE:

import pandas

import warnings warnings.filterwarnings("ignore")

df = pandas.read\_csv("cars1.csv")

X = df[['Weight', 'Volume']] y = df['CO2']

DEPARTMENT OF COMPUTER APPLICATION

Page | 46

from sklearn import linear\_model

regr = linear\_model.LinearRegression() regr.fit(X, y)

#### OUTPUT:

LinearRegression()

#### CODE:

predictedCO2 = regr.predict([[2300, 1000]]) print(predictedCO2)

#### OUTPUT:

[104.86715554]

DEPARTMENT OF COMPUTER APPLICATION

Page | 47

#### Data set:Iris.csv CODE

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

dataset = pd.read\_csv("iris.csv")

X = dataset.iloc[:, [0,1,2, 3]].values y = dataset.iloc[:, 4].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

from sklearn.preprocessing import StandardScaler sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train) X\_test = sc.transform(X\_test)

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0, solver='lbfgs', multi\_class='auto') classifier.fit(X\_train, y\_train)

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

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred) print(cm)

**OUTPUT**

# [[13 0 0]

[ 0 15 1]

# [ 0 0 9]]

#### CODE

import seaborn as sns import pandas as pd ax = plt.axes()

df\_cm = cm

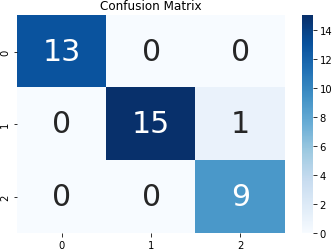
sns.heatmap(df\_cm, annot=True, annot\_kws={"size": 30}, fmt='d',cmap="Blues", ax = ax ) ax.set\_title('Confusion Matrix')

plt.show()

DEPARTMENT OF COMPUTER APPLICATION

Page | 48

#### OUTPUT



DEPARTMENT OF COMPUTER APPLICATION

Page | 49

#### AIM

1. Program to implement text classification using Support vector machine.

#### CODE:

**Dataset used: iris.csv**

import numpy as np

import matplotlib.pyplot as plt from sklearn import svm, datasets

# import some data to play with iris = datasets.load\_iris()

X = iris.data[:, :2]

# we only take the first two features.We could

# avoid this ugly slicing by using a two-dim dataset y = iris.target

# we create an instance of SVM and fit out data. We do not scale our

# data since we want to plot the support vectors C =

1.0 # SVM regularization parameter

svc = svm.SVC(kernel='linear', C=1,gamma='auto').fit(X, y) # create a mesh to plot in

#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

plt.subplot(1, 1, 1)

Z = svc.predict(np.c\_ravel[xx.(), yy.ravel()]) 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')

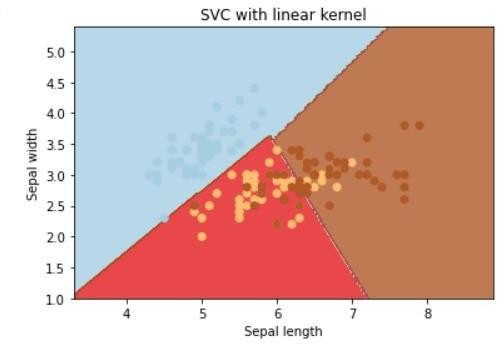
plt.ylabel('Sepal width') plt.xlim(xx.min(), xx.max())

DEPARTMENT OF COMPUTER APPLICATION

Page | 50

plt.title('SVC with linear kernel') plt.show()

#### OUTPUT:



**CODE:**

**Dataset used: True.csv, Fake.csv**

#Importing Libraries im- port pandas as pd import numpy as np

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

from sklearn.feature\_extraction.text import CountVectorizer from sklearn.feature\_extraction.text import TfidfTransformer

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

from sklearn.svm import LinearSVC

import csv

true = pd.read\_csv("True.csv") fake = pd.read\_csv("Fake.csv")

DEPARTMENT OF COMPUTER APPLICATION

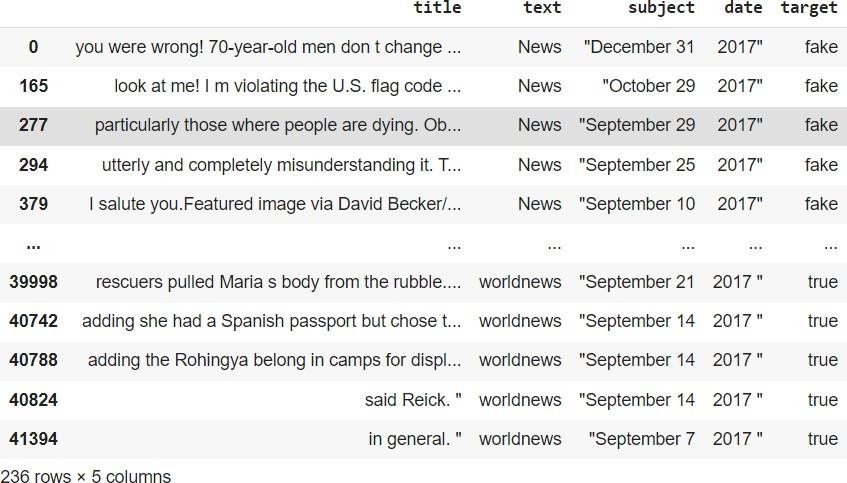
Page | 51

fake['target'] = 'fake' true['target'] = 'true' #News dataset

news = pd.concat([fake, true]).reset\_index(drop = True) news.head()

news.dropna()

#### OUTPUT:



**CODE:**

#Train-test split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(news['text'], new s.target, test\_size=0.2, random\_state=1)

#Term frequency(TF)=count(word)/total(words)6+0ZXCVBNM,./

#TF-IDF: we can even reduce the weightage of more common words like (t he, is, an etc.) which occurs in all document.

#This is called as TF-IDF i.e Term Frequency times inverse document frequency.

#count vectorizer : involves counting the number of occurrences ea ch word appears in a document

DEPARTMENT OF COMPUTER APPLICATION

Page | 52

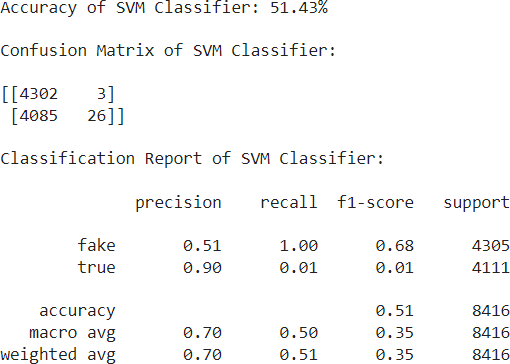
pipe2 = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTran sformer()), ('model', LinearSVC())])

model\_svc = pipe2.fit(x\_train.astype('U'), y\_train.astype('U')) svc\_pred = model\_svc.predict(x\_test.astype('U'))

print("Accuracy of SVM Classifier: {}%".format(round(accuracy\_scor e(y\_test, svc\_pred)\*100,2)))

print("\nConfusion Matrix of SVM Classifier:\n") print(confusion\_matrix(y\_test, svc\_pred)) print("\nClas- sification Report of SVM Classifier:\n") print(classifi- cation\_report(y\_test, svc\_pred))

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 53

**Dataset: apples\_and\_oranges.csv**

#### CODE:

import pandas as pd

data = pd.read\_csv("apples\_and\_oranges.csv")

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

training\_set, test\_set = train\_test\_split(data, test\_size = 0.2, random\_state = 1) X\_train = training\_set.iloc[:,0:2].values

Y\_train = training\_set.iloc[:,2].values X\_test = test\_set.iloc[:,0:2].values Y\_test = test\_set.iloc[:,2].values

#### CODE:

#Use of SVC with kernal='rbf' from sklearn.svm import SVC

classifier = SVC(kernel='rbf', random\_state = 1) classifier.fit(X\_train,Y\_train)

#### OUTPUT:

SVC(random\_state=1)

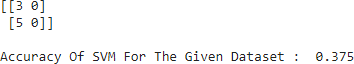
#### CODE:

Y\_pred = classifier.predict(X\_test) test\_set["Predictions"] = Y\_pred

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(Y\_test,Y\_pred) print(cm)

accuracy = float(cm.diagonal().sum())/len(Y\_test) print("\nAccuracy Of SVM For The Given Dataset : ", accuracy)

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 54

#### CODE

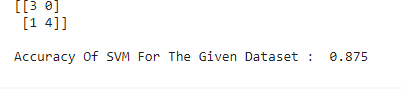
#Use of SVC with kernal='linear'

classifier1 = SVC(kernel='linear', random\_state = 1) classifier1.fit(X\_train,Y\_train)

Y\_pred1 = classifier1.predict(X\_test) cm1 = confusion\_matrix(Y\_test,Y\_pred1) print(cm1)

accuracy1 = float(cm1.diagonal().sum())/len(Y\_test) print("\nAccuracy Of SVM For The Given Dataset : ", accuracy1)

#### OUTPUT:



**CODE**

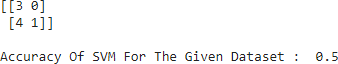
#Use of Linear SVC

from sklearn.svm import LinearSVC classifier2 = LinearSVC(random\_state = 1) classifier2.fit(X\_train,Y\_train)

Y\_pred2 = classifier2.predict(X\_test) cm2 = confusion\_matrix(Y\_test,Y\_pred2) print(cm2)

accuracy2 = float(cm2.diagonal().sum())/len(Y\_test) print("\nAccuracy Of SVM For The Given Dataset : ", accuracy2)

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 55

#### CODE:

from sklearn.preprocessing import LabelEncoder le = LabelEncoder()

Y\_train = le.fit\_transform(Y\_train) from sklearn.svm import SVC

clasifier = SVC(kernel='rbf', random\_state = 1) classifier.fit(X\_train,Y\_train)

#### OUTPUT:

SVC(random\_state=1)

#### CODE:

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap plt.figure(figsize = (7,7))

X\_set, y\_set = X\_train, Y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step=0.01), np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01)) plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('black', 'white'))) plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = ListedColormap(('red', 'orange'))(i), label =j)

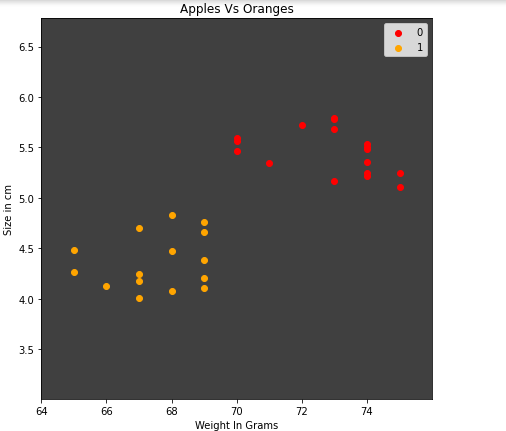
plt.title('Apples Vs Oranges') plt.xlabel('Weight In Grams') plt.ylabel('Size in cm') plt.legend()

plt.show()

DEPARTMENT OF COMPUTER APPLICATION

Page | 56

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 57

**Dataset: Iris.csv**

#### CODE:

# Importing the libraries import numpy as np

import matplotlib.pyplot as plt import pandas as pd

from sklearn.preprocessing import StandardScaler

# Importing the dataset

df = pd.read\_csv("iris.csv") X = df.drop('variety', axis=1) y = df.variety

print ("Number of data points ::", X.shape[0]) print("Number of features ::", X.shape[1])

#### OUTPUT:

Number of data points :: 150

Number of features :: 4

#Using Standard Scaler to transform the data. scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

from sklearn.model\_selection import train\_test\_split X\_train, X\_test, y\_train, y\_test = train\_test\_split( X\_scaled, y, test\_size=0.2, random\_state=42)

#Create the Non Linear SVM model from sklearn.svm import SVC

classifier = SVC(kernel = 'linear', random\_state = 0)

#Fit the model for the data classifier.fit(X\_train, y\_train)

#Make the prediction

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

#### CODE:

print('Accuracy of SVC on training set: {:.2f}'.format(classifier.score(X\_train, y\_train) \* 100)) print('Accuracy of SVC on test set: {:.2f}'.format(classifier.score(X\_test, y\_test) \* 100))

DEPARTMENT OF COMPUTER APPLICATION

Page | 58

#### OUTPUT:

Accuracy of SVC on training set: 98.33

Accuracy of SVC on test set: 96.67

#### CODE:

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred) print(cm)

**OUTPUT:**

[[10 0 0]

[ 0 8 1]

## [ 0 0 11]]

#### CODE:

from sklearn.metrics import accuracy\_score print("Accuracy:",accuracy\_score(y\_test, y\_pred) )

#### OUTPUT:

Accuracy: 0.9666666666666667

#### CODE:

#classification Report on SVC

from sklearn.metrics import classification\_report print("Classification report - \n", classification\_report(y\_test,y\_pred))

#### OUTPUT:

Classification report -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Setosa | 1.00 | 1.00 | 1.00 | 10 |
| Versicolor | 1.00 | 0.89 | 0.94 | 9 |
| Virginica | 0.92 | 1.00 | 0.96 | 11 |
| accuracy |  |  | 0.97 | 30 |
| macro avg | 0.97 | 0.96 | 0.97 | 30 |
| weighted avg | 0.97 | 0.97 | 0.97 | 30 |

DEPARTMENT OF COMPUTER APPLICATION

Page | 59

# Create the SVM model using LinearSVC from sklearn.svm import LinearSVC

clf = LinearSVC(random\_state = 0) #Fit the model for the data clf.fit(X\_train, y\_train)

#Make the prediction y\_pred1 = clf.predict(X\_test)

#### CODE:

print('Accuracy of Linear SVC on training set: {:.2f}'.format(clf.score(X\_train, y\_train) \* 100)) print('Accuracy of Linear SVC on test set: {:.2f}'.format(clf.score(X\_test, y\_test) \* 100))

#### OUTPUT:

Accuracy of Linear SVC on training set: 95.00

Accuracy of Linear SVC on test set: 100.00

#### CODE:

from sklearn.metrics import confusion\_matrix cm = confusion\_matrix(y\_test, y\_pred1) print(cm)

from sklearn.metrics import accuracy\_score print("Accuracy:",accuracy\_score(y\_test, y\_pred1) )

**OUTPUT:**

## [[10 0 0]

[ 0 9 0]

## [ 0 0 11]]

Accuracy: 1.0

#### CODE:

#classification Report on Linear SVC

from sklearn.metrics import classification\_report

print("Classification report - \n", classification\_report(y\_test,y\_pred1))

#### OUTPUT:

Classification report -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | precision | recall | f1-score | support |
| Setosa | 1.00 | 1.00 | 1.00 | 10 |
| Versicolor | 1.00 | 1.00 | 1.00 | 9 |
| Virginica | 1.00 | 1.00 | 1.00 | 11 |
| accuracy |  |  | 1.00 | 30 |
| macro avg | 1.00 | 1.00 | 1.00 | 30 |
| weighted avg | 1.00 | 1.00 | 1.00 | 30 |

#### AIM

1. Program to implement decision trees using any standard dataset available in the public domain and find the accuracy of thealgorithm.

#### CODE:

**Dataset used: iris**

import numpy as np im- port pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

data=load\_iris() X=data.data y=data.target print(X.shape,y.shape)

#### OUTPUT:

(150, 4) (150,)

#### CODE:

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

#for checking testi ng results

from sklearn.metrics import classification\_report, confusion\_matrix #for visualizing tree

from sklearn.tree import plot\_tree

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X , y, test\_si ze

= 25, random\_state = 10)

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

#### OUTPUT:

DecisionTreeClassifier()

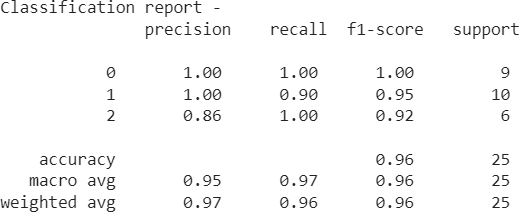
#### CODE:

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

print("Classification report - \n", classification\_report(y\_test,y

\_pred))

#### OUTPUT:



**CODE:**

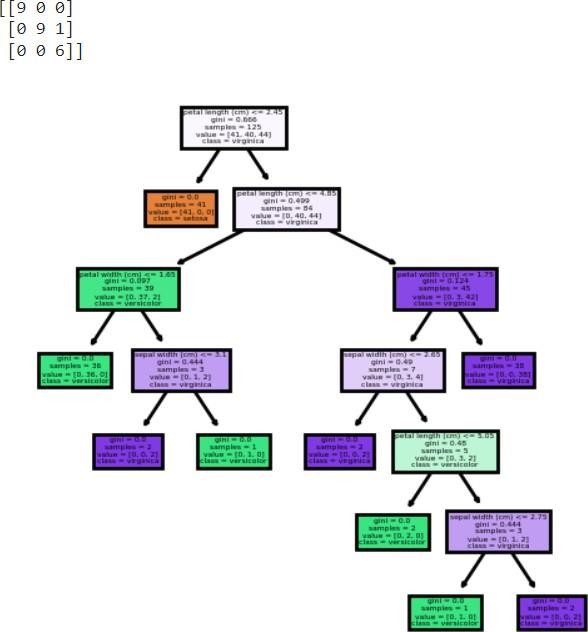
cm = confusion\_matrix(y\_test, y\_pred) print(cm)

from sklearn import tree

fig,axes = plt.subplots(nrows=1,ncols=1,figsize =(3,3),dpi=200) tree.plot\_tree(clf,feature\_names=data.feature\_names,class\_names=da ta.target\_names,filled=True)

plt.show() fig.savefig("/con- tent/iris\_tree.png")

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 63

**Dataset:titanic.csv**

#### CODE:

import pandas as pd

df = pd.read\_csv('titanic.csv', index\_col='PassengerId') print(df.head())

#### OUTPUT:

Survived Pclass \

PassengerId

1 0 3

2 1 1

3 1 3

4 1 1

5 0 3

Name Sex Age \

PassengerId

1. Braund, Mr. Owen Harris male 22.0
2. Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0
3. Heikkinen, Miss. Laina female 26.0
4. Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0
5. Allen, Mr. William Henry male 35.0

SibSp Parch Ticket Fare Cabin Embarked PassengerId

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | A/5 21171 7.2500 NaN | S |  |
| 2 | 1 | 0 | PC 17599 71.2833 C85 | C |
| 3 | 0 | 0 | STON/O2. 3101282 7.9250 | NaN | S |
| 4 | 1 | 0 | 113803 53.1000 C123 | S |  |
| 5 | 0 | 0 | 373450 8.0500 NaN | S |  |
| **CODE:**  df.shape |  |  |  |  |  |

#### OUTPUT:

(891, 11)

DEPARTMENT OF COMPUTER APPLICATION

Page | 64

#### CODE:

#We will be using Pclass, Sex, Age, SibSp (Siblings aboard), Parch (Parents/children aboard), and Fare to predict whether a passenger survived.

df = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Survived']] #We need to convert ‘Sex’ into an integer value of 0 or 1. df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})

#### OUTPUT:

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: [https://pandas.pydata.org/pandas-](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy) [docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy) """Entry point for launching an IPython kernel.

#### CODE:

#We also drop any rows with missing values. df = df.dropna()

#Creating input and output array

X = df.drop('Survived', axis=1) y = df['Survived']

#Generating training and test set

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=1) from sklearn import tree

model = tree.DecisionTreeClassifier() model.fit(X\_train, y\_train)

y\_predict = model.predict(X\_test)

from sklearn.metrics import accuracy\_score print("Accuracy:",accuracy\_score(y\_test, y\_predict) )

DEPARTMENT OF COMPUTER APPLICATION

Page | 65

#### OUTPUT:

Accuracy: 0.8212290502793296

#### CODE:

from sklearn.metrics import confusion\_matrix

pd.DataFrame( confusion\_matrix(y\_test, y\_predict),

columns=['Predicted Not Survival', 'Predicted Survival'], index=['True Not Survival', 'True Survival']

)

|  |  |  |
| --- | --- | --- |
| **OUTPUT**: |  | |
|  | Predicted Not Survival | Predicted Survival |
| True Not Survival | 96 | 16 |
| True Survival | 16 | 51 |
| **CODE:** |  |  |

from sklearn import tree tree.plot\_tree(model,filled=True)

#### OUTPUT:

[Text(0.4976636979427998, 0.9761904761904762, 'X[1] <= 0.5\ngini = 0.486\nsamples =

535\nvalue = [312, 223]'),

Text(0.17671224284997492, 0.9285714285714286, 'X[0] <= 1.5\ngini = 0.331\nsamples =

335\nvalue = [265, 70]'),

Text(0.0863020572002007, 0.8809523809523809, 'X[2] <= 36.5\ngini = 0.481\nsamples =

77\nvalue = [46, 31]'),

Text(0.016056196688409432, 0.8333333333333334, 'X[5] <= 37.812\ngini =

0.475\nsamples = 31\nvalue = [12, 19]'),

Text(0.008028098344204716, 0.7857142857142857, 'gini = 0.0\nsamples = 7\nvalue = [0, 7]'),

Text(0.02408429503261415, 0.7857142857142857, 'X[2] <= 17.5\ngini = 0.5\nsamples =

24\nvalue = [12, 12]'),

Text(0.016056196688409432, 0.7380952380952381, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),

Text(0.032112393376818864, 0.7380952380952381, 'X[2] <= 22.5\ngini = 0.48\nsamples =

20\nvalue = [12, 8]'),

Text(0.02408429503261415, 0.6904761904761905, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),

Text(0.04014049172102358, 0.6904761904761905, 'X[5] <= 51.798\ngini = 0.5\nsamples =

16\nvalue = [8, 8]'),

DEPARTMENT OF COMPUTER APPLICATION

Page | 66

Text(0.032112393376818864, 0.6428571428571429, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),

Text(0.0481685900652283, 0.6428571428571429, 'X[5] <= 64.979\ngini = 0.473\nsamples =

13\nvalue = [5, 8]'),

Text(0.04014049172102358, 0.5952380952380952, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),

Text(0.05619668840943302, 0.5952380952380952, 'X[5] <= 379.925\ngini =

2\nvalue = [1, 1]'),

Text(0.4862017059708981, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'), Text(0.5022579026593076, 0.5, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'), Text(0.4942298043151029, 0.5952380952380952, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'), Text(0.5765178123432012, 0.6428571428571429, 'X[3] <= 0.5\ngini = 0.233\nsamples =

119\nvalue = [103, 16]'),

Text(0.5464124435524336, 0.5952380952380952, 'X[5] <= 41.248\ngini = 0.264\nsamples =

96\nvalue = [81, 15]'),

Text(0.5263421976919217, 0.5476190476190477, 'X[5] <= 20.656\ngini = 0.245\nsamples =

91\nvalue = [78, 13]'),

Text(0.518314099347717, 0.5, 'X[5] <= 17.444\ngini = 0.259\nsamples = 85\nvalue = [72, 13]'),

Text(0.5102860010035123, 0.4523809523809524, 'X[2] <= 26.5\ngini = 0.245\nsamples =

84\nvalue = [72, 12]'),

Text(0.462117410938284, 0.40476190476190477, 'X[5] <= 8.175\ngini = 0.184\nsamples =

39\nvalue = [35, 4]'),

Text(0.43803311590566985, 0.35714285714285715, 'X[2] <= 20.0\ngini = 0.444\nsamples =

9\nvalue = [6, 3]'),

Text(0.43000501756146514, 0.30952380952380953, 'X[2] <= 17.0\ngini = 0.48\nsamples =

5\nvalue = [2, 3]'),

Text(0.42197691921726044, 0.2619047619047619, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),

Text(0.43803311590566985, 0.2619047619047619, 'X[2] <= 18.5\ngini = 0.444\nsamples =

3\nvalue = [1, 2]'),

Text(0.43000501756146514, 0.21428571428571427, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),

Text(0.44606121424987455, 0.21428571428571427, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),

Text(0.44606121424987455, 0.30952380952380953, 'gini = 0.0\nsamples = 4\nvalue = [4, 0]'),

Text(0.4862017059708981, 0.35714285714285715, 'X[0] <= 2.5\ngini = 0.064\nsamples =

30\nvalue = [29, 1]'),

Text(0.4781736076266934, 0.30952380952380953, 'X[5] <= 11.0\ngini = 0.133\nsamples =

14\nvalue = [13, 1]'),

Text(0.4701455092824887, 0.2619047619047619, 'X[2] <= 21.0\ngini = 0.32\nsamples =

5\nvalue = [4, 1]'),

Text(0.462117410938284, 0.21428571428571427, 'X[2] <= 17.5\ngini = 0.444\nsamples =

3\nvalue = [2, 1]'),

Text(0.45408931259407925, 0.16666666666666666, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),

Text(0.4701455092824887, 0.16666666666666666, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),

DEPARTMENT OF COMPUTER APPLICATION

Page | 67

Text(0.4781736076266934, 0.21428571428571427, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),

Text(0.4862017059708981, 0.2619047619047619, 'gini = 0.0\nsamples = 9\nvalue = [9, 0]'),



DEPARTMENT OF COMPUTER APPLICATION

Page | 68

#### CODE:

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

from sklearn.metrics import classification\_report, confusion\_matrix import matplotlib.pyplot as plt

#### CODE:

import warnings warnings.filterwarnings("ignore")

import pandas as pd

df = pd.read\_csv("hepatitis.csv") print(df)

#### OUTPUT:

pstatus age sex steroid antivirals fatigue malaise anorexia \

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2 | 30 | 2 | 1 | 2 | | 2 | 2 | 2 | |
| 1 | 2 | 50 | 1 | 1 | 2 | | 1 | 2 | 2 | |
| 2 | 2 | 78 | 1 | 2 | 2 | | 1 | 2 | 2 | |
| 3 | 2 | 34 | 1 | 2 | 2 | | 2 | 2 | 2 | |
| 4 | 2 | 34 | 1 | 2 | 2 | | 2 | 2 | 2 | |
| .. | ... | ... | ... | ... | ... | | ... | ... | ... | |
| 137 | 1 | 46 | 1 | 2 | 2 | | 1 | 1 | 1 | |
| 138 | 2 | 44 | 1 | 2 | 2 | | 1 | 2 | 2 | |
| 139 | 2 | 61 | 1 | 1 | 2 | | 1 | 1 | 2 | |
| 140 | 2 | 53 | 2 | 1 | 2 | | 1 | 2 | 2 | |
| 141 | 1 | 43 | 1 | 2 | 2 | | 1 | 2 | 2 | |
|  | liver\_big | liver\_firm | | spleen\_palable | | | spiders | ascites | varices | \ |
| 0 | 1 | 2 | | 2 | | | 2 | 2 | 2 |  |
| 1 | 1 | 2 | | 2 | | | 2 | 2 | 2 |  |
| 2 | 2 | 2 | | 2 | | | 2 | 2 | 2 |  |
| 3 | 2 | 2 | | 2 | | | 2 | 2 | 2 |  |
| 4 | 2 | 2 | | 2 | | | 2 | 2 | 2 |  |
| .. | ... | ... | | ... | | | ... | ... | ... |  |
| 137 | 2 | 1 | | 2 | | | 1 | 1 | 1 |  |
| 138 | 2 | 1 | | 2 | | | 2 | 2 | 2 |  |
| 139 | 1 | 2 | | 2 | | | 1 | 2 | 2 |  |
| 140 | 2 | 2 | | 1 | | | 1 | 2 | 1 |  |
| 141 | 2 | 2 | | 1 | | | 1 | 1 | 2 |  |
|  | bilirubin | alk\_phosphate | | | sgot | albumin | protime | histology | | |
| 0 | 1.0 | 85 | | | 18 | 4.0 | 61 | 1 | | |
| 1 | 0.9 | 135 | | | 42 | 3.5 | 61 | 1 | | |
| 2 | 0.7 | 96 | | | 32 | 4.0 | 61 | 1 | | |
| 3 | 1.0 | 105 | | | 200 | 4.0 | 61 | 1 | | |
| 4 | 0.9 | 95 | | | 28 | 4.0 | 75 | 1 | | |
| .. | ... | ... | | | ... | ... | ... | ... | | |
| 137 | 7.6 | 105 | | | 242 | 3.3 | 50 | 2 | | |
| 138 | 0.9 | 126 | | | 142 | 4.3 | 61 | 2 | | |
| 139 | 0.8 | 75 | | | 20 | 4.1 | 61 | 2 | | |
| 140 | 1.5 | 81 | | | 19 | 4.1 | 48 | 2 | | |

DEPARTMENT OF COMPUTER APPLICATION

Page | 69

141 1.2 100 19 3.1 42 2

[142 rows x 20 columns]

#### CODE:

df.shape

#### OUTPUT:

(142, 20)

#### CODE:

df.shape df['pstatus'].value\_counts()

#### OUTPU:

2 116

1 26

Name: pstatus, dtype: int64

#### CODE:

df.pstatus[df.pstatus == 2] = 0 df['pstatus'].value\_counts()

#### OUTPUT:

0 116

1 26

Name: pstatus, dtype: int64

#### CODE:

X = df.drop('pstatus', axis=1) y = df['pstatus']

#### CODE:

# splitting to trainset and Test set in the ratio 70:30

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

#### CODE:

# KNN Classifier

DEPARTMENT OF COMPUTER APPLICATION

Page | 70

from sklearn.neighbors import KNeighborsClassifier classifier1 = KNeighborsClassifier(n\_neighbors=5) classifier1.fit(X\_train, y\_train)

y\_pred1 = classifier1.predict(X\_test) print(confusion\_matrix(y\_test, y\_pred1)) print(classification\_report(y\_test, y\_pred1))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OUTPUT:** |  | | | |
| [[32 1] |
| [10 0]] | precision | recall | f1-score | support |
| 0 | 0.76 | 0.97 | 0.85 | 33 |
| 1 | 0.00 | 0.00 | 0.00 | 10 |
| accuracy |  |  | 0.74 | 43 |
| macro avg | 0.38 | 0.48 | 0.43 | 43 |
| weighted avg | 0.58 | 0.74 | 0.65 | 43 |
| **CODE:** |  |  |  |  |

#AUC for KNN Classifier

from sklearn.metrics import auc, roc\_auc\_score, roc\_curve, recall\_score fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred1)

roc\_auc1 = auc(fpr,tpr)

# Plot ROC

plt.title('Receiver Operating Characteristic') plt.plot(fpr, tpr, 'b',label='AUC = %0.3f'% roc\_auc1) plt.legend(loc='lower right')

plt.plot([0,1],[0,1],'r--')

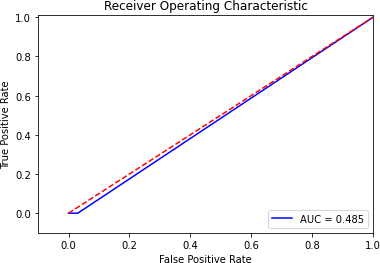
plt.xlim([-0.1,1.0])

plt.ylim([-0.1,1.01]) plt.ylabel('True Positive Rate') plt.xlabel('False Positive Rate') plt.show()

DEPARTMENT OF COMPUTER APPLICATION

Page | 71

#### OUTPUT:



**CODE:**

# Naive Bayes Classifier

from sklearn.naive\_bayes import GaussianNB classifier2 = GaussianNB() classifier2.fit(X\_train, y\_train)

y\_pred2 = classifier2.predict(X\_test) print(confusion\_matrix(y\_test, y\_pred2)) print(classification\_report(y\_test, y\_pred2))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OUTPUT:** |  | | | |
| [[27 6] |
| [ 1 9]] | precision | recall | f1-score | support |
| 0 | 0.96 | 0.82 | 0.89 | 33 |
| 1 | 0.60 | 0.90 | 0.72 | 10 |
| accuracy |  |  | 0.84 | 43 |
| macro avg | 0.78 | 0.86 | 0.80 | 43 |
| weighted avg | 0.88 | 0.84 | 0.85 | 43 |

DEPARTMENT OF COMPUTER APPLICATION

Page | 72

#### CODE:

#AUC for Naive Bayes Classifier

from sklearn.metrics import auc, roc\_auc\_score, roc\_curve, recall\_score fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred2)

roc\_auc2 = auc(fpr,tpr)

# Plot ROC

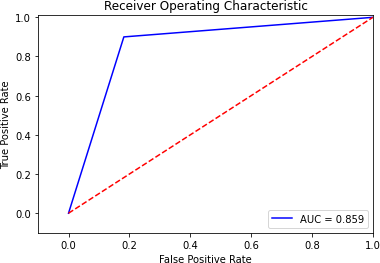
plt.title('Receiver Operating Characteristic') plt.plot(fpr, tpr, 'b',label='AUC = %0.3f'% roc\_auc2) plt.legend(loc='lower right')

plt.plot([0,1],[0,1],'r--')

plt.xlim([-0.1,1.0])

plt.ylim([-0.1,1.01]) plt.ylabel('True Positive Rate') plt.xlabel('False Positive Rate') plt.show()

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 73

#### CODE:

# Decision tree Classifier

from sklearn.tree import DecisionTreeClassifier classifier3=DecisionTreeClassifier() classifier3.fit(X\_train,y\_train)

y\_pred3 = classifier3.predict(X\_test) print(confusion\_matrix(y\_test, y\_pred3)) print(classification\_report(y\_test, y\_pred3))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OUTPUT:** |  | | | |
| [[24 9] |
| [ 4 6]] | precision | recall | f1-score | support |
| 0 | 0.86 | 0.73 | 0.79 | 33 |
| 1 | 0.40 | 0.60 | 0.48 | 10 |
| accuracy |  |  | 0.70 | 43 |
| macro avg | 0.63 | 0.66 | 0.63 | 43 |
| weighted avg | 0.75 | 0.70 | 0.72 | 43 |

#### CODE:

#AUC for Decision tree Classifier

from sklearn.metrics import auc, roc\_auc\_score, roc\_curve, recall\_score fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred3)

roc\_auc3 = auc(fpr,tpr)

# Plot ROC

plt.title('Receiver Operating Characteristic') plt.plot(fpr, tpr, 'b',label='AUC = %0.3f'% roc\_auc3) plt.legend(loc='lower right')

plt.plot([0,1],[0,1],'r--')

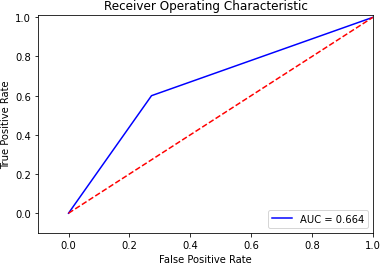
plt.xlim([-0.1,1.0])

plt.ylim([-0.1,1.01]) plt.ylabel('True Positive Rate') plt.xlabel('False Positive Rate') plt.show()

DEPARTMENT OF COMPUTER APPLICATION

Page | 74

#### OUTPUT:

\

#### CODE:

# Logistic Regression

from sklearn.linear\_model import LogisticRegression

classifier4 = LogisticRegression(random\_state = 0, solver='lbfgs', multi\_class='auto') classifier4.fit(X\_train, y\_train)

y\_pred4 = classifier4.predict(X\_test) print(confusion\_matrix(y\_test, y\_pred4)) print(classification\_report(y\_test, y\_pred4))

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OUTPUT:** |  | | | |
| [[30 3] |
| [ 7 3]] | precision | recall | f1-score | support |
| 0 | 0.81 | 0.91 | 0.86 | 33 |
| 1 | 0.50 | 0.30 | 0.37 | 10 |
| accuracy |  |  | 0.77 | 43 |
| macro avg | 0.66 | 0.60 | 0.62 | 43 |
| weighted avg | 0.74 | 0.77 | 0.75 | 43 |

DEPARTMENT OF COMPUTER APPLICATION

Page | 75

#### CODE:

#AUC for Logistic Regression

from sklearn.metrics import auc, roc\_auc\_score, roc\_curve, recall\_score fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred4)

roc\_auc4 = auc(fpr,tpr)

# Plot ROC

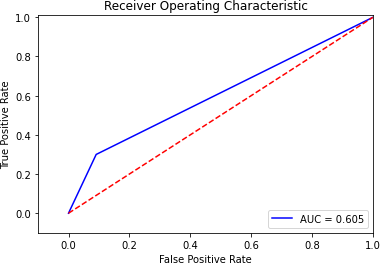
plt.title('Receiver Operating Characteristic') plt.plot(fpr, tpr, 'b',label='AUC = %0.3f'% roc\_auc4) plt.legend(loc='lower right')

plt.plot([0,1],[0,1],'r--')

plt.xlim([-0.1,1.0])

plt.ylim([-0.1,1.01]) plt.ylabel('True Positive Rate') plt.xlabel('False Positive Rate') plt.show()

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 76

#### AIM

1. Program to implement k-means clustering technique using any standard dataset available in the public domain.

#### CODE:

**Dataset used: GENERAL.csv**

# importing the libraries im- port numpy as np

import pandas as pd

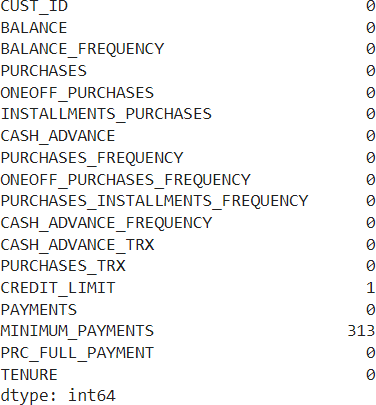
%matplotlib inline

import matplotlib.pyplot as plt dataset= pd.read\_csv('./CC GENERAL.csv')

# checking the presence of null values print(dataset.isnull().sum()) #CREDIT\_LIMIT 1

#MINIMUM\_PAYMENTS 313

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

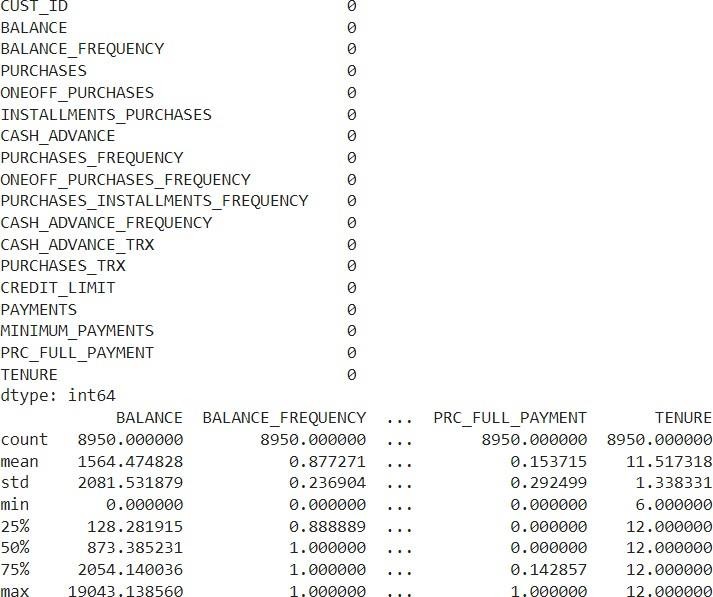
Page | 77

#### CODE:

dataset['CREDIT\_LIMIT'].fillna(dataset.CREDIT\_LIMIT.mean(), inplac e = True) dataset['MINIMUM\_PAYMENTS'].fillna(dataset.MINIMUM\_PAY- MENTS.mean(), inplace = True) # unfilled vaues replaced using mean print(dataset.isnull().sum())

print(dataset.describe())

#### OUTPUT:



**CODE:**

dataset.drop(['CUST\_ID'], axis= 1, inplace = True) #no relevance f or custid

# No Categorical Values found X = dataset.iloc[:,:].values

DEPARTMENT OF COMPUTER APPLICATION

Page | 78

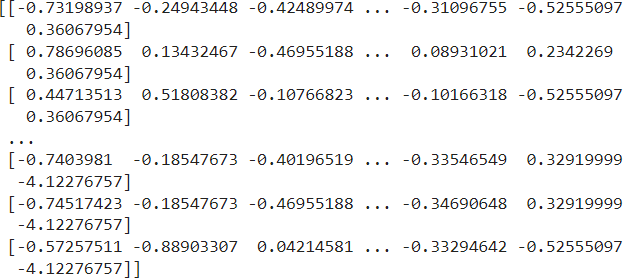
# Using standard scaler

from sklearn.preprocessing import StandardScaler standardscaler= StandardScaler()

X = standardscaler.fit\_transform(X) #scaling the values

print(X)

#### OUTPUT:



**CODE:**

"""K MEANS CLUSTERING """

#Inertia, or the within-

cluster sum of squares criterion, can be recognized as a measure o f how internally coherent clusters are

from sklearn.cluster import KMeans wss= []

for i in range(1, 11):

kmeans= KMeans(n\_clusters = i, init = 'kmeans++', random\_state = 0)

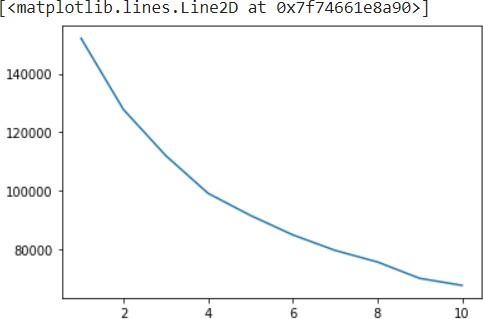
kmeans.fit(X) wss.append(kmeans.in- ertia\_)

plt.plot(range(1,11), wss) # selecting 4

DEPARTMENT OF COMPUTER APPLICATION

Page | 79

#### OUTPUT:



**CODE:**

wss\_mean=np.array(wss).mean() print(wss)

print(wss\_mean)

print([abs(wss\_mean-x) for x in wss]) k=np.argmin([abs(wss\_mean-x) for x in wss])+1

#### OUTPUT:

[152149.99999999983, 127784.92103208725, 111986.41162208859,

99073.93826774803, 91502.98328256077, 84851.13240432573,

79532.40237691796, 75568.97609993909, 69954.91393943134,

67546.56302862825]

95995.22420537268

[56154.775794627145, 31789.69682671457, 15991.187416715911,

3078.714062375351, 4492.240922811907, 11144.091801046947,

16462.82182845472, 20426.248105433595, 26040.31026594134,

28448.661176744426]

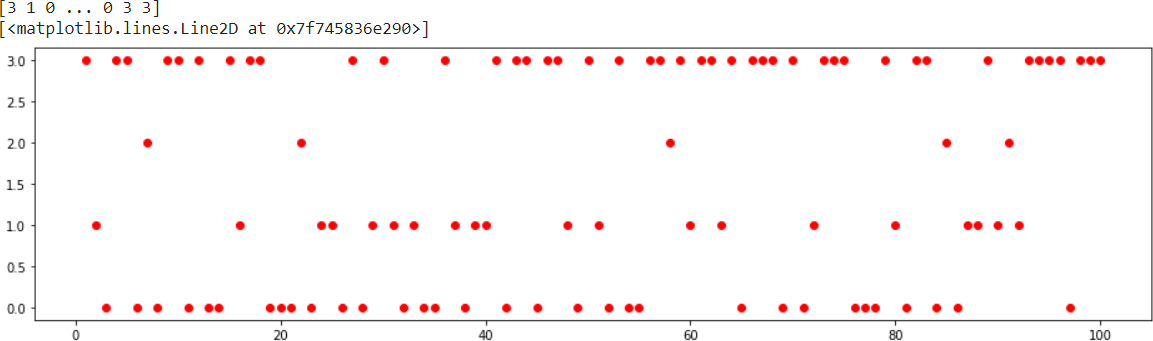
#### CODE:

kmeans = KMeans(n\_clusters = k, init= 'k- means++', random\_state = 0) kmeans.fit(X)

Y\_pred\_K= kmeans.predict(X) print(Y\_pred\_K)

DEPARTMENT OF COMPUTER APPLICATION

Page | 80



#showing the clusters of first 100 persons plt.figure(figsize=(16,4)) plt.plot(range(1,100+1),Y\_pred\_K[:100],'ro')

#### OUTPUT:

**dhggghhhh**

DEPARTMENT OF COMPUTER APPLICATION

Page | 81

#### Dataset:Iris.csv CODE:

import numpy as np

from sklearn.cluster import KMeans from sklearn.datasets import load\_iris

%matplotlib inline

import matplotlib.pyplot as plt iris = load\_iris()

X = iris.data print(X)

#### OUTPUT:

[[5.1 3.5 1.4 0.2] [4.9 3. 1.4 0.2] [4.7 3.2 1.3 0.2] [4.6 3.1 1.5 0.2] [5. 3.6

1.4 0.2] [5.4 3.9 1.7 0.4] [4.6 3.4 1.4 0.3] [5. 3.4 1.5 0.2] [4.4 2.9 1.4 0.2]

[4.9 3.1 1.5 0.1] [5.4 3.7 1.5 0.2] [4.8 3.4 1.6 0.2] [4.8 3. 1.4 0.1] [4.3 3.

1.1 0.1] [5.8 4. 1.2 0.2] [5.7 4.4 1.5 0.4] [5.4 3.9 1.3 0.4] [5.1 3.5 1.4 0.3]

[5.7 3.8 1.7 0.3] [5.1 3.8 1.5 0.3] [5.4 3.4 1.7 0.2] [5.1 3.7 1.5 0.4] [4.6 3.6

1. 0.2] [5.1 3.3 1.7 0.5] [4.8 3.4 1.9 0.2] [5. 3. 1.6 0.2] [5. 3.4 1.6 0.4] [5.2

3.5 1.5 0.2] [5.2 3.4 1.4 0.2] [4.7 3.2 1.6 0.2] [4.8 3.1 1.6 0.2] [5.4 3.4 1.5

0.4] [5.2 4.1 1.5 0.1] [5.5 4.2 1.4 0.2] [4.9 3.1 1.5 0.2] [5. 3.2 1.2 0.2] [5.5

3.5 1.3 0.2] [4.9 3.6 1.4 0.1] [4.4 3. 1.3 0.2] [5.1 3.4 1.5 0.2] [5. 3.5 1.3

0.3] [4.5 2.3 1.3 0.3] [4.4 3.2 1.3 0.2] [5. 3.5 1.6 0.6] [5.1 3.8 1.9 0.4] [4.8

3. 1.4 0.3] [5.1 3.8 1.6 0.2] [4.6 3.2 1.4 0.2] [5.3 3.7 1.5 0.2] [5. 3.3 1.4

0.2] [7. 3.2 4.7 1.4] [6.4 3.2 4.5 1.5] [6.9 3.1 4.9 1.5] [5.5 2.3 4. 1.3] [6.5

2.8 4.6 1.5] [5.7 2.8 4.5 1.3] [6.3 3.3 4.7 1.6] [4.9 2.4 3.3 1. ] [6.6 2.9 4.6

1.3] [5.2 2.7 3.9 1.4] [5. 2. 3.5 1. ] [5.9 3. 4.2 1.5] [6. 2.2 4. 1. ] [6.1 2.9

4.7 1.4] [5.6 2.9 3.6 1.3] [6.7 3.1 4.4 1.4] [5.6 3. 4.5 1.5] [5.8 2.7 4.1 1. ]

[6.2 2.2 4.5 1.5] [5.6 2.5 3.9 1.1] [5.9 3.2 4.8 1.8] [6.1 2.8 4. 1.3] [6.3 2.5

4.9 1.5] [6.1 2.8 4.7 1.2] [6.4 2.9 4.3 1.3] [6.6 3. 4.4 1.4] [6.8 2.8 4.8 1.4]

[6.7 3. 5. 1.7] [6. 2.9 4.5 1.5] [5.7 2.6 3.5 1. ] [5.5 2.4 3.8 1.1] [5.5 2.4 3.7

1. ] [5.8 2.7 3.9 1.2] [6. 2.7 5.1 1.6] [5.4 3. 4.5 1.5] [6. 3.4 4.5 1.6] [6.7

3.1 4.7 1.5] [6.3 2.3 4.4 1.3] [5.6 3. 4.1 1.3] [5.5 2.5 4. 1.3] [5.5 2.6 4.4

1.2] [6.1 3. 4.6 1.4] [5.8 2.6 4. 1.2] [5. 2.3 3.3 1. ] [5.6 2.7 4.2 1.3] [5.7 3.

4.2 1.2] [5.7 2.9 4.2 1.3] [6.2 2.9 4.3 1.3] [5.1 2.5 3. 1.1] [5.7 2.8 4.1 1.3]

[6.3 3.3 6. 2.5] [5.8 2.7 5.1 1.9] [7.1 3. 5.9 2.1] [6.3 2.9 5.6 1.8] [6.5 3. 5.8

2.2]

DEPARTMENT OF COMPUTER APPLICATION

Page | 82

#### CODE:

kmeans= KMeans(n\_clusters = 3, init = 'k-means++', random\_state = 0) kmeans.fit(X)

Y\_pred\_K= kmeans.predict(X) print(Y\_pred\_K)

#### OUTPUT:

[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 0 0 2 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 0 0

0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2 2 2 0 0 2 2 2 2 0 2

0

2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2 2 0]

#### CODE:

inertia = [] ax = []

for i in range(2,10):

ax.append(i)

kmeans= KMeans(n\_clusters = i, init = 'k-means++', random\_state = 0) kmeans.fit(X)

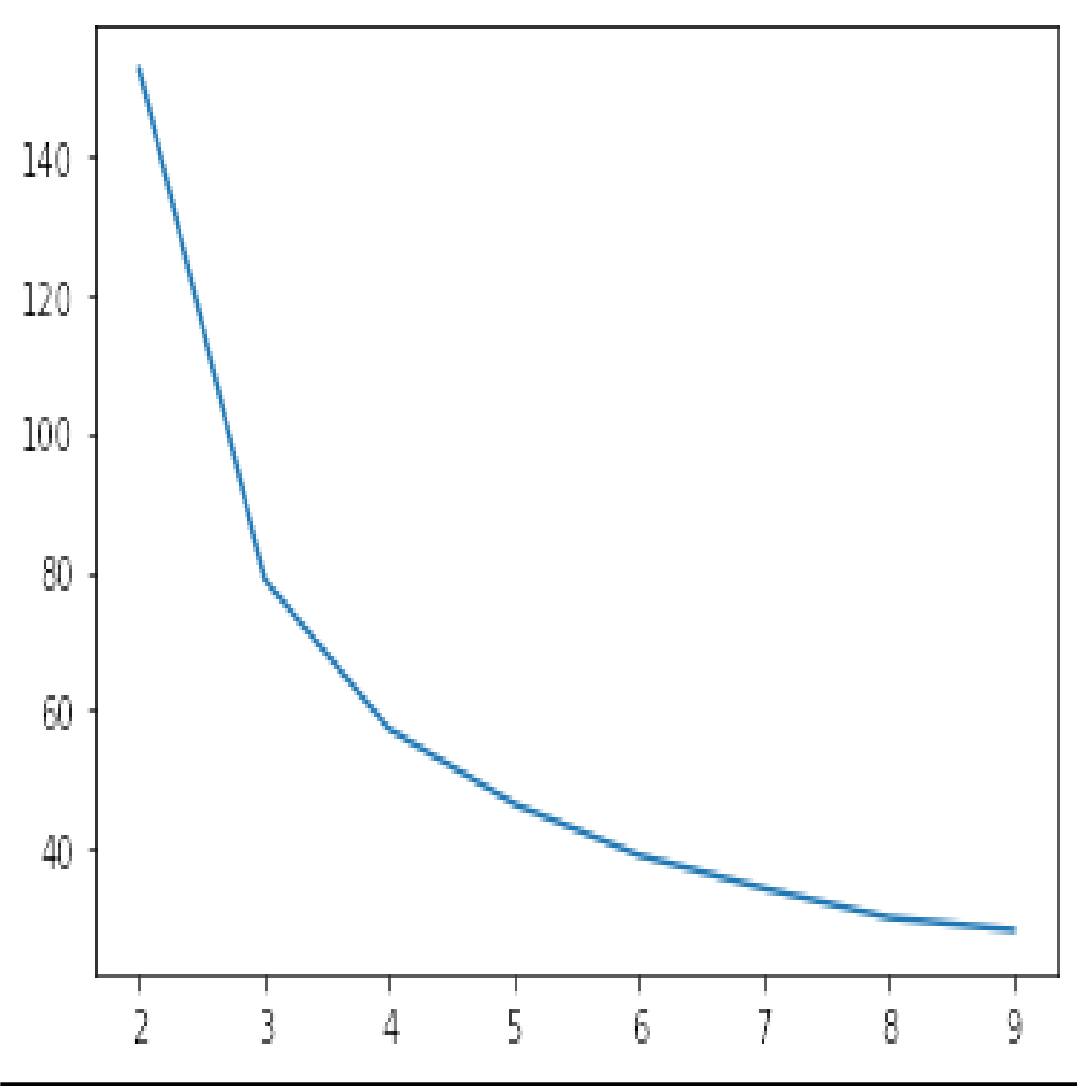
inertia.append(kmeans.inertia\_) plt.plot(ax,inertia)

DEPARTMENT OF COMPUTER APPLICATION

Page | 83

#### OUTPUT:

[<matplotlib.lines.Line2D at 0x7f8639026550>]



#### CODE:

kmeans= KMeans(n\_clusters = 3, init = 'k-means++', random\_state = 0) kmeans.fit(X)

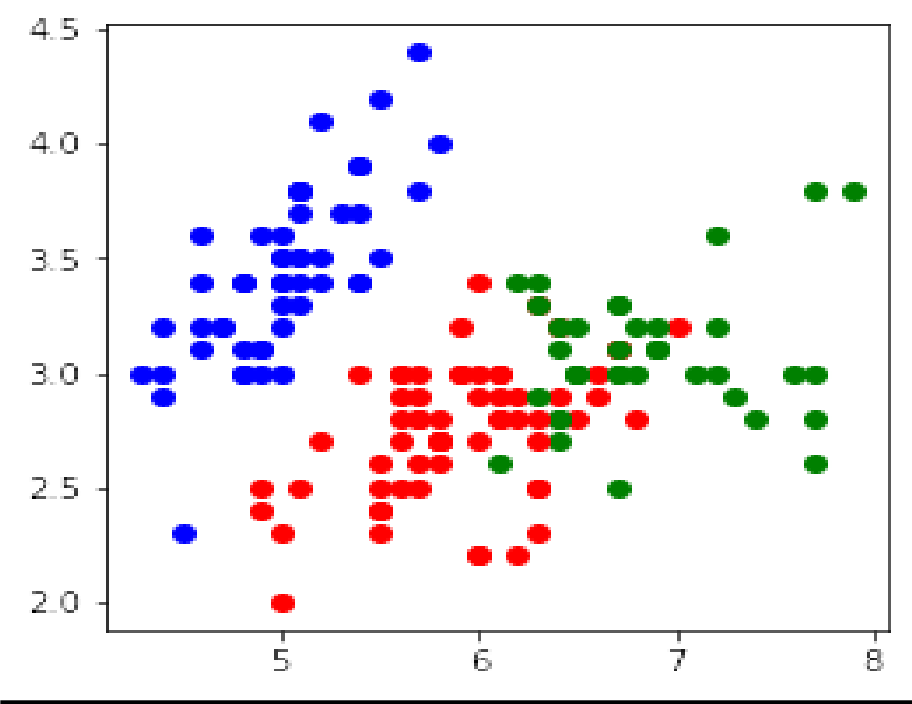
plt.figure(figsize=(4,4)) Y\_pred\_K = kmeans.predict(X)

colors = ['red','blue','green','yellow','cyan'] for x,y in zip(X,Y\_pred\_K): plt.scatter(x[0],x[1],color = colors[y])

DEPARTMENT OF COMPUTER APPLICATION

Page | 84

#### OUTPUT:



DEPARTMENT OF COMPUTER APPLICATION

Page | 85

#### CODE:

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans x1=10\*np.random.rand(100,2) **CODE:**

x1.shape

#### OUTPUT:

(100, 2)

**CODE:**

kmean=KMeans(n\_clusters=3) kmean.fit(x1)

#### OUTPUT :

KMeans(n\_clusters=3)

|  |  |
| --- | --- |
| **CODE:** |  |
| kmean.cluster\_centers\_ |
| **OUTPUT :** |
| array([[1.95688735, | 4.05905136], |
| [7.60153979, | 2.67451186], |
| [7.01154396, | 7.67791651]]) |

DEPARTMENT OF COMPUTER APPLICATION

Page | 86

**CODE:**

kmean.labels\_

#### OUTPUT:

array([2, 0,

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0, | 1, | 0, | 0, | 0, | 2, | 2, | 2, | 2, | 1, | 2, | 2, | 1, | 0, | 1, | 0, | 0, | 1, | 0, | 1, |
| 1, | 1, | 2, | 0, | 2, | 2, | 1, | 0, | 2, | 1, | 0, | 0, | 1, | 2, | 0, | 2, | 1, | 1, | 2, | 0, |
| 0, | 2, | 1, | 1, | 2, | 1, | 2, | 1, | 0, | 0, | 2, | 1, | 0, | 0, | 2, | 2, | 2, | 1, | 0, | 2, |
| 2, | 0, | 2, | 1, | 1, | 0, | 1, | 1, | 0, | 0, | 0, | 0, | 2, | 0, | 2, | 0, | 0, | 0, | 0, | 0, |
| 0, | 1, | 2, | 1, | 0, | 0, | 1, | 2, | 0, | 2, | 0], dtype=int32) | | | | | | | | | |

0,

0,

0,

2,

1,

0,

1,

#### CODE:

wcss = []

for i in range(1,20):

kmeans = KMeans(n\_clusters=i,init= 'k-means++',max\_iter=300,n\_init=10,random\_state=0) kmeans.fit(x1)

wcss.append(kmeans.inertia\_) print('Cluster', i, 'Inertia', kmeans.inertia\_) plt.plot(range(1,20),wcss)

plt.title('The Elbow Curve') plt.xlabel('Number of clusters')

plt.ylabel('WCSS') ##WCSS stands for total within-cluster sum of square plt.show()

#### OUTPUT:

Cluster 1 Inertia 1709.8592837186357

Cluster 2 Inertia 941.6272426718026

Cluster 3 Inertia 612.4712566124308

Cluster 4 Inertia 368.3666143214158

Cluster 5 Inertia 289.2602914923789

Cluster 6 Inertia 229.03053194379697

Cluster 7 Inertia 187.38301059593198

Cluster 8 Inertia 161.92639910808086

Cluster 9 Inertia 142.6648686647746

Cluster 10 Inertia 121.3532493740191

Cluster 11 Inertia 110.4239060692322

Cluster 12 Inertia 98.99605007934787

Cluster 13 Inertia 91.07314617434768

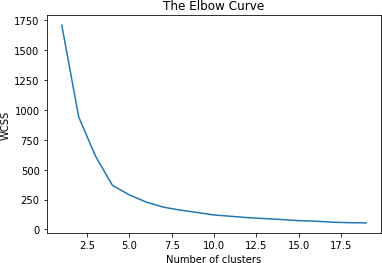
Cluster 14 Inertia 83.05767097627933

Cluster 15 Inertia 74.07981138805766

Cluster 16 Inertia 69.55361615261592

Cluster 17 Inertia 60.80930432109166

Cluster 18 Inertia 57.03871895907935

Cluster 19 Inertia 54.88323560270942

#### AIM

10:Programs on feedforward network to classify any standard dataset available in the public domain.

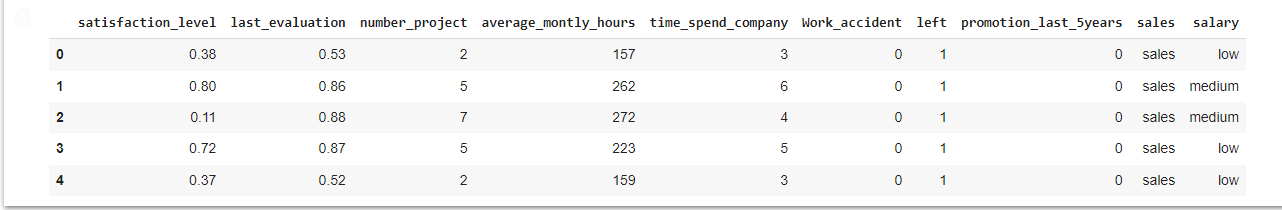
**Dataset used: HR\_comma\_sep.csv**

#### CODE:

import numpy as np import pandas as pd

# Load data data=pd.read\_csv('HR\_comma\_sep.csv') data.head()

#### OUTPUT:



**CODE:**

from sklearn import preprocessing #

Creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers. data['salary']=le.fit\_transform(data['salary']) data['sales']=le.fit\_transform(data['sales'])

X=data[['satisfaction\_level', 'last\_evaluation', 'number\_project', 'average\_montly\_hour s', 'time\_spend\_company', 'Work\_accident', 'promotion\_last\_5years', 'sales', 'salary']]

y=data['left']

# Import train\_test\_split function

from sklearn.model\_selection import train\_test\_split # Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) # 70% training and 30% test

from sklearn.neural\_network import MLPClassifier # Create model object

clf = MLPClassifier(hidden\_layer\_sizes=(6,5), random\_state=5,verbose=False,learning\_rate\_init=. 01)

# Fit data onto the model clf.fit(X\_train,y\_train)

#### OUTPUT:



**CODE:**

ypred=clf.predict(X\_test) #

Import accuracy score

from sklearn.metrics import accuracy\_score # Calcuate accuracy accuracy\_score(y\_test,ypred)

#### OUTPUT:



**AIM:**

**11:Programs on convolutional neural network to classify images from any standard dataset in the public domain.**

#### CODE:

import numpy as np import pandas as pd

# Load data data=pd.read\_csv('HR\_comma\_sep.csv')

data.head()

#### OUTPUT:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **satis- fac- tion\_l evel** | **last\_e valu- ation** | **num- ber\_ pro- ject** | **aver- age\_montly**  **\_hours** | **time\_spen d\_com- pany** | **Work**  **\_acci- dent** | **le ft** | **promo- tion\_last\_ 5years** | **sal es** | **sal- ary** |
| **0** | 0.38 | 0.53 | 2 | 157 | 3 | 0 | 1 | 0 | sal es | lo w |
| **1** | 0.80 | 0.86 | 5 | 262 | 6 | 0 | 1 | 0 | sal es | me diu m |
| **2** | 0.11 | 0.88 | 7 | 272 | 4 | 0 | 1 | 0 | sal es | me diu m |
| **3** | 0.72 | 0.87 | 5 | 223 | 5 | 0 | 1 | 0 | sal es | lo w |
| **4** | 0.37 | 0.52 | 2 | 159 | 3 | 0 | 1 | 0 | sal es | lo w |

**CODE:**

from sklearn import preprocessing

# Creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string labels into numbers. data['salary']=le.fit\_transform(data['salary']) data['sales']=le.fit\_transform(data['sales'])

X=data[['satisfaction\_level', 'last\_evaluation', 'number\_project', 'average\_montly\_hours', 'time\_spend\_company', 'Work\_accident', 'promotion\_last\_5years', 'sales', 'salary']] y=data['left']

# Import train\_test\_split function

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

# Split dataset into training set and test set

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) # 70% training and 30% test

from sklearn.neural\_network import MLPClassifier # Create model object

clf = MLPClassifier(hidden\_layer\_sizes=(6,5),

random\_state=5, verbose=False, learning\_rate\_init=0.01)

# Fit data onto the model clf.fit(X\_train,y\_train)

ypred=clf.predict(X\_test)

#### OUTPUT:

MLPClassifier(hidden\_layer\_sizes=(6, 5), learning\_rate\_init=0.01, random\_state=5)

#### CODE:

# Import accuracy score

from sklearn.metrics import accuracy\_score # Calcuate accuracy

print ("Accuracy:",accuracy\_score(y\_test,ypred))

#### OUTPUT:

Accuracy: 0.9386666666666666

#### CODE:

from sklearn.metrics import classification\_report, confusion\_matrix print(confusion\_matrix(y\_test, ypred)) print(classification\_report(y\_test, ypred))

#### OUTPUT:

|  |  |
| --- | --- |
| [[3248 | 180] |
| [ 96 | 976]] |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | precision  0.97 | recall  0.95 | f1-score  0.96 | support  3428 |
| 1 | 0.84 | 0.91 | 0.88 | 1072 |
| accuracy |  |  | 0.94 | 4500 |
| macro avg | 0.91 | 0.93 | 0.92 | 4500 |
| weighted avg | 0.94 | 0.94 | 0.94 | 4500 |

**Aim:**

**12: Implement problems on natural language processing - Part of Speech tagging, N-gram & smoothening and Chunking using NLTK**

#### CODE:

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize, sent\_tokenize nltk.download('stopwords')

nltk.download('punkt') nltk.download('averaged\_perceptron\_tagger')

stop\_words = set(stopwords.words('english'))

#### TOKENIZATION

#Dummy text

txt = "Hello. MCA S3 is fantastic. We learn many new concepts and implement them in our practical exams. "\

"1st of all the data science is a new paper."

# sent\_tokenize is one of instances of

# PunktSentenceTokenizer from the nltk.tokenize.punkt module

tokenized = sent\_tokenize(txt) for i in tokenized:

# Word tokenizers is used to find the words # and punctuation in a string

wordsList = nltk.word\_tokenize(i)

# removing stop words from wordList

wordsList = [w for w in wordsList if not w in stop\_words]

# Using a Tagger. Which is part-of-speech # tagger or POS-tagger.

tagged = nltk.pos\_tag(wordsList) print(tagged)

#### OUTPUT:

[nltk\_data] Downloading package stopwords to /root/nltk\_data... [nltk\_data] Unzipping corpora/stopwords.zip.

[nltk\_data] Downloading package punkt to /root/nltk\_data... [nltk\_data] Unzipping tokenizers/punkt.zip.

[nltk\_data] Downloading package averaged\_perceptron\_tagger to [nltk\_data] /root/nltk\_data...

[nltk\_data] Unzipping taggers/averaged\_perceptron\_tagger.zip. [('Hello', 'NNP'), ('.', '.')]

[('MCA', 'NNP'), ('S3', 'NNP'), ('fantastic', 'JJ'), ('.', '.')]

[('We', 'PRP'), ('learn', 'VBP'), ('many', 'JJ'), ('new', 'JJ'),

('concepts', 'NNS'), ('implement', 'JJ'), ('practical', 'JJ'),

('exams', 'NN'), ('.', '.')]

[('1st', 'CD'), ('data', 'NNS'), ('science', 'NN'), ('new', 'JJ'), ('paper', 'NN'), ('.', '.')]

#### CODE:

**SENTIMENTAL ANALYSIS**

import numpy as np import pandas as pd

import matplotlib.pyplot as plt plt.style.use(style='seaborn')

#get the data from https:/[/www](http://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-).[kaggle.com/ankurzing/sentiment-analysis-for-financial-](http://www.kaggle.com/ankurzing/sentiment-analysis-for-financial-) news/version/5

colnames=['Sentiment', 'news']

df=pd.read\_csv('all-data.csv',encoding = "ISO-8859-1", names=colnames, header = None) df.head()

#### OUTPUT:

Sentiment news

1. neutral According to Gran , the company has no plans t...
2. neutral Technopolis plans to develop in stages an area...
3. negative The international electronic industry company ...
4. positive With the new production plant the company woul...
5. positive According to the company 's updated strategy f...

#### CODE:

df.info()

#### OUTPUT:

<class 'pandas.core.frame.DataFrame'> RangeIndex: 4846 entries, 0 to 4845 Data columns (total 2 columns):

# Column Non-Null Count Dtype

- - -

* 1. Sentiment 4846 non-null object
  2. news 4846 non-null object dtypes: object(2)

memory usage: 75.8+ KB

#### CODE:

df['Sentiment'].value\_counts()

|  |  |
| --- | --- |
| **OUTPUT:** |  |
| neutral | 2879 |
| positive | 1363 |
| negative | 604 |

Name: Sentiment, dtype: int64

#### CODE:

y=df['Sentiment'].values

#### OUTPUT:

(4846,)

#### CODE:

y.shape x=df['news'].values x.shape

#### OUTPUT:

(4846,)

#### CODE:

from sklearn.model\_selection import train\_test\_split (x\_train,x\_test,y\_train,y\_test)=train\_test\_split(x,y,test\_size=0.4) x\_train.shape

y\_train.shape x\_test.shape y\_test.shape **OUTPUT:**

(1939,)

#### CODE:

df1=pd.DataFrame(x\_train) df1=df1.rename(columns={0:'news'}) df2=pd.DataFrame(y\_train) df2=df2.rename(columns={0:'sentiment'}) df\_train=pd.concat([df1,df2],axis=1) df\_train.head()

#### OUTPUT:

news sentiment

* + 1. Elcoteq 's global service offering covers the neutral
    2. During the past 10 years the factory has produ neutral
    3. This includes a EUR 39.5 mn change in the fair neutral
    4. Loss for the period totalled EUR 15.6 mn compa negative
    5. Residents access to the block is planned to be neutral

#### CODE:

df3=pd.DataFrame(x\_test) df3=df3.rename(columns={0:'news'}) df4=pd.DataFrame(y\_test) df4=df2.rename(columns={0:'sentiment'}) df\_test=pd.concat([df3,df4],axis=1) df\_test.head()

#### OUTPUT:

News sentiment

1. Aldata to Share Space Optimization Vision at A... neutral
2. Biohit already services many current Genesis c... neutral
3. According to Soosalu , particular attention wa... neutral
4. The layoff talks were first announced in August . negative
5. The company has an annual turnover of EUR32 .8 m. neutral

#### CODE:

#removing punctuations

#library that contains punctuation import string

string.punctuation

#### OUTPUT:

!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~

#### CODE:

#defining the function to remove punctuation def remove\_punctuation(text):

if(type(text)==float):

return text ans=""

for i in text:

if i not in string.punctuation:

ans+=i return ans

#storing the puntuation free text in a new column called clean\_msg df\_train['news']= df\_train['news'].apply(lambda x:remove\_punctuation(x)) df\_test['news']= df\_test['news'].apply(lambda x:remove\_punctuation(x)) df\_train.head()

#punctuations are removed from news column in train dataset

#### OUTPUT:

News sentiment

1. Elcoteq s global service offering covers the e... neutral
2. During the past 10 years the factory has produ... neutral
3. This includes a EUR 395 mn change in the fair ... neutral
4. Loss for the period totalled EUR 156 mn compar... negative
5. Residents access to the block is planned to be... neutral

#### CODE:

import nltk

from nltk.corpus import stopwords nltk.download('stopwords')

#### OUTPUT:

[nltk\_data] Downloading package stopwords to /root/nltk\_data... [nltk\_data] Package stopwords is already up-to-date!

True

**CODE:**

**N-gram model**

#method to generate n-grams:

#params:

#text-the text for which we have to generate n-grams

#ngram-number of grams to be generated from the text(1,2,3,4 etc., default value=1) def generate\_N\_grams(text,ngram=1):

words=[word for word in text.split(" ") if word not in set(stopwords.words('english'))] print("Sentence after removing stopwords:",words)

temp=zip(\*[words[i:] for i in range(0,ngram)]) ans=[' '.join(ngram) for ngram in temp]

return ans

#### CODE:

generate\_N\_grams("The sun rises in the east",2)

#### OUTPUT:

Sentence after removing stopwords: ['The', 'sun', 'rises', 'east'] ['The sun', 'sun rises', 'rises east']

#### CODE:

generate\_N\_grams("The sun rises in the east",3)

#### OUTPUT:

Sentence after removing stopwords: ['The', 'sun', 'rises', 'east'] ['The sun rises', 'sun rises east']

#### CODE:

generate\_N\_grams("The sun rises in the east",4)

#### OUTPUT:

Sentence after removing stopwords: ['The', 'sun', 'rises', 'east'] ['The sun rises east']

#### AIM:

**13: Implement a program to scrap the web page of any popular website – suggested python package is scrappy (ensure ethical conduct).**

#### CODE:

class BlogSpider(scrapy.Spider): name = 'blogspider'

start\_urls = ['https://[www.zyte.com/blog/'](http://www.zyte.com/blog/%27)]

def parse(self, response):

for title in response.css('.oxy-post-title'): yield {'title': title.css('::text').get()}

for next\_page in response.css('a.next'):

yield response.follow(next\_page, self.parse)

#### OUTPUT:

[

{&quot;title&quot;: &quot;Zyte named as one of Deloitte Technology Fast 50&quot;},

{&quot;title&quot;: &quot;Zyte named as one of Deloitte Technology Fast 50&quot;},

{&quot;title&quot;: &quot;How to extract data from an HTML table&quot;},

{&quot;title&quot;: &quot;What is a proxy server and how do they work?&quot;},

{&quot;title&quot;: &quot;Extract Summit 2021: Highlights and key takeaways&quot;},

{&quot;title&quot;: &quot;How does a headless browser help with web scraping and data extraction?&quot;},

{&quot;title&quot;: &quot;Proxies versus VPNs: What\u2019s the difference, and which one is right for my

use case?&quot;},

{&quot;title&quot;: &quot;Manage bans and get your data with Zyte Data API Smart Browser&quot;},

{&quot;title&quot;: &quot;How to reduce noise in the data through data parsing&quot;},

{&quot;title&quot;: &quot;What is web data harvesting?&quot;},

{&quot;title&quot;: &quot;In pursuit of perfection: measuring web product data quality&quot;},

{&quot;title&quot;: &quot;Zyte named as one of Deloitte Technology Fast 50&quot;},

{&quot;title&quot;: &quot;Web Data Extraction Summit 2021&quot;},

{&quot;title&quot;: &quot;Residential Proxies: How are they different to data center proxies &amp; how to

manage them&quot;},

{&quot;title&quot;: &quot;Zyte Developers Community newsletter issue #10&quot;},

{&quot;title&quot;: &quot;What is data mining? How is it different from web scraping?&quot;},

{&quot;title&quot;: &quot;Zyte Developers Community newsletter issue #9&quot;},

{&quot;title&quot;: &quot;How Scrapy makes web crawling easy&quot;},

]

#### AIM:

**14:Implement a simple web crawler (ensure ethical conduct).**

#### INSTALLATION CODE:

pip install requests bs4

#### OUTPUT:

Requirement already satisfied: requests in

/usr/local/lib/python3.7/dist-packages (2.23.0)

Requirement already satisfied: bs4 in /usr/local/lib/python3.7/dist- packages (0.0.1)

Requirement already satisfied: chardet<4,>=3.0.2 in

/usr/local/lib/python3.7/dist-packages (from requests) (3.0.4) Requirement already satisfied: certifi>=2017.4.17 in

/usr/local/lib/python3.7/dist-packages (from requests) (2021.10.8) Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests) (1.24.3) Requirement already satisfied: idna<3,>=2.5 in

/usr/local/lib/python3.7/dist-packages (from requests) (2.10) Requirement already satisfied: beautifulsoup4 in

/usr/local/lib/python3.7/dist-packages (from bs4) (4.6.3)

#### CODE:

import logging

from urllib.parse import urljoin import requests

from bs4 import BeautifulSoup

logging.basicConfig(

format='%(asctime)s %(levelname)s:%(message)s', level=logging.INFO)

class Crawler:

def init (self, urls=[]): self.visited\_urls = [] self.urls\_to\_visit = urls

def download\_url(self, url): return requests.get(url).text

def get\_linked\_urls(self, url, html):

soup = BeautifulSoup(html, 'html.parser') for link in soup.find\_all('a'):

path = link.get('href')

if path and path.startswith('/'):

path = urljoin(url, path) yield path

def add\_url\_to\_visit(self, url):

if url not in self.visited\_urls and url not in self.urls\_to\_visit: self.urls\_to\_visit.append(url)

def crawl(self, url):

html = self.download\_url(url)

for url in self.get\_linked\_urls(url, html): self.add\_url\_to\_visit(url)

def run(self):

while self.urls\_to\_visit:

url = self.urls\_to\_visit.pop(0) logging.info(f'Crawling: {url}') try:

self.crawl(url) except Exception:

logging.exception(f'Failed to crawl: {url}') finally:

self.visited\_urls.append(url)

if name == ' main ': Crawler(urls=['https:/[/www](http://www.imdb.com/%27).[imdb.com/'](http://www.imdb.com/%27)]).run()

#### OUTPUT:

2022-03-22 10:42:36,095 INFO:Crawling: <https://www.imdb.com/>

2022-03-22 10:42:36,931 INFO:Crawling:

<https://www.imdb.com/?ref_=nv_home> 2022-03-22 10:42:37,778 INFO:Crawling:

<https://www.imdb.com/calendar/?ref_=nv_mv_cal> 2022-03-22 10:42:38,164 INFO:Crawling:

<https://www.imdb.com/list/ls016522954/?ref_=nv_tvv_dvd> 2022-03-22 10:42:41,281 INFO:Crawling:

<https://www.imdb.com/chart/top/?ref_=nv_mv_250> 2022-03-22 10:42:42,869 INFO:Crawling:

<https://www.imdb.com/chart/moviemeter/?ref_=nv_mv_mpm> 2022-03-22 10:42:44,039 INFO:Crawling:

<https://www.imdb.com/feature/genre/?ref_=nv_ch_gr> 2022-03-22 10:42:44,413 INFO:Crawling:

<https://www.imdb.com/chart/boxoffice/?ref_=nv_ch_cht> 2022-03-22 10:42:44,718 INFO:Crawling:

<https://www.imdb.com/showtimes/?ref_=nv_mv_sh>

2022-03-22 10:42:45,305 INFO:Crawling: [https://www.imdb.com/movies-in-](https://www.imdb.com/movies-in-theaters/?ref_=nv_mv_inth) [theaters/?ref\_=nv\_mv\_inth](https://www.imdb.com/movies-in-theaters/?ref_=nv_mv_inth)

2022-03-22 10:42:45,727 INFO:Crawling: [https://www.imdb.com/coming-](https://www.imdb.com/coming-soon/?ref_=nv_mv_cs) [soon/?ref\_=nv\_mv\_cs](https://www.imdb.com/coming-soon/?ref_=nv_mv_cs)

2022-03-22 10:42:46,672 INFO:Crawling:

<https://www.imdb.com/news/movie/?ref_=nv_nw_mv> 2022-03-22 10:42:47,212 INFO:Crawling:

<https://www.imdb.com/india/toprated/?ref_=nv_mv_in>

2022-03-22 10:42:47,904 INFO:Crawling: [https://www.imdb.com/whats-on-](https://www.imdb.com/whats-on-tv/?ref_=nv_tv_ontv) [tv/?ref\_=nv\_tv\_ontv](https://www.imdb.com/whats-on-tv/?ref_=nv_tv_ontv)

2022-03-22 10:42:48,300 INFO:Crawling:

<https://www.imdb.com/chart/toptv/?ref_=nv_tvv_250> 2022-03-22 10:42:49,114 INFO:Crawling:

<https://www.imdb.com/chart/tvmeter/?ref_=nv_tvv_mptv> 2022-03-22 10:42:49,763 INFO:Crawling:

<https://www.imdb.com/feature/genre/> 2022-03-22 10:42:50,141 INFO:Crawling:

<https://www.imdb.com/news/tv/?ref_=nv_nw_tv> 2022-03-22 10:42:50,478 INFO:Crawling:

<https://www.imdb.com/india/tv?ref_=nv_tv_in>

2022-03-22 10:42:50,898 INFO:Crawling: [https://www.imdb.com/what-to-](https://www.imdb.com/what-to-watch/?ref_=nv_watch) [watch/?ref\_=nv\_watch](https://www.imdb.com/what-to-watch/?ref_=nv_watch)

2022-03-22 10:42:51,572 INFO:Crawling:

<https://www.imdb.com/trailers/?ref_=nv_mv_tr> 2022-03-22 10:42:52,003 INFO:Crawling:

<https://www.imdb.com/originals/?ref_=nv_sf_ori> 2022-03-22 10:42:52,225 INFO:Crawling:

<https://www.imdb.com/imdbpicks/?ref_=nv_pi> 2022-03-22 10:42:52,567 INFO:Crawling:

<https://www.imdb.com/podcasts/?ref_=nv_pod> 2022-03-22 10:42:52,861 INFO:Crawling:

<https://www.imdb.com/oscars/?ref_=nv_ev_acd> 2022-03-22 10:42:53,254 INFO:Crawling:

<https://m.imdb.com/feature/bestpicture/?ref_=nv_ch_osc> 2022-03-22 10:42:53,893 INFO:Crawling:

[https://www.imdb.com/search/title/?count=100&groups=oscar\_best\_picture\_](https://www.imdb.com/search/title/?count=100&groups=oscar_best_picture_winners&sort=year%2Cdesc&ref_=nv_ch_osc) [winners&sort=year%2Cdesc&ref\_=nv\_ch\_osc](https://www.imdb.com/search/title/?count=100&groups=oscar_best_picture_winners&sort=year%2Cdesc&ref_=nv_ch_osc)

2022-03-22 10:42:54,908 INFO:Crawling:

<https://www.imdb.com/emmys/?ref_=nv_ev_rte> 2022-03-22 10:42:55,171 INFO:Crawling:

<https://www.imdb.com/imdbpicks/womenshistorymonth/?ref_=nv_ev_whm> 2022-03-22 10:42:55,686 INFO:Crawling:

<https://www.imdb.com/starmeterawards/?ref_=nv_ev_sma>

2022-03-22 10:42:56,004 INFO:Crawling: [https://www.imdb.com/comic-](https://www.imdb.com/comic-con/?ref_=nv_ev_comic) [con/?ref\_=nv\_ev\_comic](https://www.imdb.com/comic-con/?ref_=nv_ev_comic)

2022-03-22 10:42:56,444 INFO:Crawling:

<https://www.imdb.com/nycc/?ref_=nv_ev_nycc> 2022-03-22 10:42:56,790 INFO:Crawling:

<https://www.imdb.com/sundance/?ref_=nv_ev_sun>

DEPARTMENT OF COMPUTER APPLICATION