SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM NAGAR, KATTANKULATHUR – 603 203

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

LAB MANUAL

CS3365-DATA SCIENCE LABORATORY

B.E.-CSE – 3rd SEMESTER

REGULATION 2023

ACADEMIC YEAR 2025-26

Prepared by

Mrs. A. Lalitha **Assistant Professor (Sel. G.)**

Mrs. S. Shanthi **Assistant Professor (Sel. G.)**

Mr. T. Rajasekaran **Assistant Professor (Sr. G.)**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

To devise captivating, fascinating and unique practices of teaching that discovers the trained talent and inherent competences of young minds to evolve as humane professional Computer Science Engineers.

Mission

- To provide students with challenging ventures, contributing to the betterment of their selfdom to compete with international talents.
- To act as a motivational hub to exhibit practical knowledge with the latest technological updates and research publications.
- To render ample knowledge to exhibit their ubiquitous talents for the social prosperity and promote industry-institute harmony to upgrade the standards for the international reputation.

Objective

To maintain high standards in the teaching leaning process by moulding students with the potential to acquire more knowledge in the field of Computer Science and Engineering as determined by their evidence both of academic achievement and distinctive careers.

INDEX

Programme Educational Objectives (PEO's) :

PEO1: To mould students to exhibit top performance in the higher education and research and to become the State-of-the-art technocrat.

PEO2: To impart the necessary background in Computer Science and Engineering by providing solid foundation in Mathematical, Science and Engineering fundamentals.

PEO3: To equip the students with the breadth of Computer Science and Engineering innovate novel solutions for the benefit of common man.

PEO4: To groom the students to be multifaceted entrepreneurs with professional ethical attitude in broader social perspective.

PEO5: To provide an ambience learning environment that is conducive for the growth of successful professional career of students.

Programme Outcomes (PO's) :

Engineering Graduates Will Be Able To

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using the first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for the complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including the design of experiments, analysis and interpretation of data, and the synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSO's) :

PSO1: Exhibit proficiency in planning, implementing and evaluating team oriented-software Programming solutions to specific business problems and society needs.
PSO2: Demonstrate professional skills in applying programming skills, competency and decision making capability through hands-on experiences.
PSO3: Apply logical thinking in analyzing complex real world problems, and use professional and ethical behaviors to provide proper solutions to those problems.
PSO4: Demonstrate the ability to work effectively as part of a team in applying technology to business and personal situations.

SYLLABUS

CS3365 DATA SCIENCE LABORATORY

L	T	P	C
0	0	3	1.5

COURSE OBJECTIVES:

- To understand the python libraries for data science
- To understand the basic Statistical and Probability measures for data science.
- To learn descriptive analytics on the benchmark data sets.
- To apply correlation and regression analytics on standard data sets.
- To present and interpret data using visualization packages in Python.

LIST OF EXPERIMENTS:

1. Download, install and explore the features of NumPy, SciPy, Jupyter, Statsmodels and Pandas packages.
2. Working with Numpy arrays
3. Working with Pandas data frames
4. Reading data from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set.
5. Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following:
    a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.
    b. Bivariate analysis: Linear and logistic regression modeling
    c. Multiple Regression analysis
    d. Also compare the results of the above analysis for the two data sets.
6. Apply and explore various plotting functions on UCI data sets.
    a. Normal curves
    b. Density and contour plots
    c. Correlation and scatter plots
    d. Histograms
    e. Three dimensional plotting
7. Visualizing Geographic Data with Basemap

LIST OF EQUIPMENTS:

Tools: Python, Numpy, Scipy, Matplotlib, Pandas, statmodels, seaborn, plotly, bokeh
Note: Example data sets like: UCI, Iris, Pima Indians Diabetes etc.

TOTAL: 45 PERIODS

Ex. No: 1 Date:	**Download, Install and explore the features of numpy, scipy, jupyter, statsmodels and pandas Packages**

AIM:

To Install and explore the features of NumPy, SciPy, Statsmodels, jupyter and Pandas packages in Python.

ALGORITHM:

1. Download Python and Jupyter.

2. Install Python and Jupyter.

3. Install the pack like NumPy, SciPy Satsmodels and Pandas.

4. Verify the proper execution of Python and Jupyter.

Python Installation

1. Open the python official web site. (https://www.python.org/)
2. Downloads ==> Windows ==> Select Recent Release. (Requires Windows 10 or above versions)
3. Install "python-3.10.6-amd64.exe"

Jupyter Installation

1. Open command prompt and enter the following to check whether the pyton was installed properly or not, “python –version”.

2. If installation is proper it returns the version of python

3. Enter the following to check whether the pyton package manager was installed properly or not, “pip –version”

4. If installation is proper it returns the version of python package manager

5. Enter the following command “pip install jupyterlab”.

6. Enter the following command “pip install jupyter notebook”.

7. Copy the above command result from path to upgrade command and paste it and execute for upgrade process.

8. Create a folder and name the folder accordingly.

9. Open command prompt and enter in to that folder. Enter the following code

“jupyter notebook” and then give enter.

 Now new jupyter notebook will be opened for our use.

pip Installation

Installation of NumPy

 pip install numpy

Installation of SciPy

 pip install scipy

Installation of Statsmodels

pip install statsmodels

Installation of Pandas

 pip install pandas

1. **Features of NumPy**

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.

Features:

These are the important features of NumPy

1. Array 2. Random 3. Universal Functions

Program :

```
import numpy as np
def main():
    # 1. Array Creation
    # Creating a 1D array
    arr1 = np.array([1, 2, 3, 4, 5])
    print("1D Array:")
    print(arr1)
       # Creating a 2D array (matrix)
    arr2 = np.array([[1, 2, 3], [4, 5, 6]])
    print("\n2D Array (Matrix):")
    print(arr2)
     # 2. Random Number Generation
    # Generating random integers
    random_ints = np.random.randint(1, 100, size=5)  # 5 random integers between 1 and 100
```

```
    print("\nRandom Integers:")
    print(random_ints)
    # 3. Universal Functions (ufunc)
    # Applying mathematical functions element-wise
    arr = np.array([1, 2, 3, 4, 5])
    # Square root of each element
    sqrt_arr = np.sqrt(arr)
    print("\nSquare Root of Array Elements:")
    print(sqrt_arr)
    # Exponential of each element
    exp_arr = np.exp(arr)
    print("\nExponential of Array Elements:")
    print(exp_arr)
if __name__ == "__main__":
    main()
```

Output

```
1D Array:
[1 2 3 4 5]

2D Array (Matrix):
[[1 2 3]
 [4 5 6]]

Random Integers:
[54 28 83 75  5]

Square Root of Array Elements:
[1.         1.41421356 1.73205081 2.         2.23606798]

Exponential of Array Elements:
[  2.71828183   7.3890561   20.08553692  54.59815003 148.4131591 ]
```

2. SciPy

SciPy stands for Scientific Python, SciPy is a scientific computation library that uses NumPy underneath.

Features

These are the important features of SciPy

1. Constants 2. Sparse Data 3. Graphs

4. Spatial Data 5. Matlab Arrays 6. Interpolation

Program:

```
import numpy as np
from scipy import constants, sparse, spatial, interpolate
import matplotlib.pyplot as plt
def main():
    # 1. Constants
    print("Speed of light in vacuum:", constants.c)
    print("Planck constant (reduced):", constants.hbar)
    print("Boltzmann constant:", constants.k)
    # 2. Sparse Data
    # Create a sparse matrix
    data = np.array([1, 2, 3])
    row = np.array([0, 1, 2])
    col = np.array([0, 1, 2])
    sparse_matrix = sparse.coo_matrix((data, (row, col)), shape=(3, 3))
    print("\nSparse Matrix:")
    print(sparse_matrix.toarray())
    # 3. Graphs (Networks)
    # Create a simple graph
    G = sparse.csgraph.dok_matrix((5, 5), dtype=bool)
    G[0, 1] = G[1, 0] = 1
    G[1, 2] = G[2, 1] = 1
    G[2, 3] = G[3, 2] = 1
    G[3, 4] = G[4, 3] = 1
    print("\nGraph (Adjacency matrix representation):")
    print(G.toarray())
```

```
# 4. Spatial Data
# Delaunay triangulation example
points = np.array([[0, 0], [0, 1.1], [1, 0], [1, 1]])
tri = spatial.Delaunay(points)
plt.figure(figsize=(6, 4))
plt.triplot(points[:, 0], points[:, 1], tri.simplices)
plt.plot(points[:, 0], points[:, 1], 'o')
plt.title('Delaunay Triangulation')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
# 5. Matlab Arrays
mat_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matlab_matrix = np.mat(mat_array)
print("\nMatlab Array:")
print(matlab_matrix)
# 6. Interpolation
x = np.linspace(0, 10, 10)
y = np.sin(x)
f = interpolate.interp1d(x, y)
x_new = np.linspace(0, 10, 100)
y_new = f(x_new)
plt.figure(figsize=(8, 4))
plt.plot(x, y, 'o', label='Original Data')
plt.plot(x_new, y_new, '-', label='Interpolated Data')
plt.title('Interpolation Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.show()
```

```
if __name__ == "__main__":
    main()
```

3. Pandas

- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.

Features

These are the important features of Pandas.

1. Series 2. DataFrames 3. Read CSV

4. Read JSON 5. Viewing the Data 6. Data Cleaning

7. Plotting

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import minimize
# Important features of SciPy
def objective(x):
    return x**2 + 10*np.sin(x)
result = minimize(objective, x0=0)
print("Optimized value using SciPy:", result.x)
# Important features of Pandas
# Creating a DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie'],
    'Age': [25, 30, 35],
    'City': ['New York', 'Los Angeles', 'Chicago']
}
df = pd.DataFrame(data)
```

```
# 1. Series
ages_series = pd.Series([25, 30, 35], name='Age')
# 2. DataFrames
print("\nDataFrame:")
print(df)
# 3. Read CSV
df.to_csv('sample.csv', index=False)
df_csv = pd.read_csv('sample.csv')
print("\nRead CSV:")
print(df_csv)

# 4. Read JSON
df.to_json('sample.json', orient='records')
df_json = pd.read_json('sample.json', orient='records')
print("\nRead JSON:")
print(df_json)
# 5. Viewing the Data
print("\nHead (first few rows):")
print(df.head())
print("\nTail (last few rows):")
print(df.tail())
# 6. Data Cleaning (example: removing rows with missing values)
df.loc[1, 'Age'] = np.nan
df_cleaned = df.dropna()
print("\nData Cleaning:")
print(df_cleaned)
# 7. Plotting (example: bar plot)
df.plot(kind='bar', x='Name', y='Age', rot=45)
plt.title('Ages of Individuals')
plt.xlabel('Name')
```

```
plt.ylabel('Age')
plt.show()
```

4. Statsmodels

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration.

Features

These are the important features of statsmodels

1. Linear regression models
2. Survival analysis

Program

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from lifelines import KaplanMeierFitter
import matplotlib.pyplot as plt
# Example data for linear regression
np.random.seed(42)
X = np.random.rand(100, 2)
beta = np.array([3, 5])
epsilon = np.random.normal(0, 1, 100)
y = np.dot(X, beta) + epsilon
df_lr = pd.DataFrame(X, columns=['X1', 'X2'])
df_lr['y'] = y
# Linear regression model
X_lr = sm.add_constant(df_lr[['X1', 'X2']])
model_lr = sm.OLS(df_lr['y'], X_lr).fit()
print("Linear Regression Model Summary:")
print(model_lr.summary())
# Example data for survival analysis (using lifelines package for Kaplan-Meier estimation)
```

```
from lifelines.datasets import load_dd

# Load and prepare data

data_sf = load_dd()

# Kaplan-Meier estimation

kmf = KaplanMeierFitter()

kmf.fit(data_sf['duration'], event_observed=data_sf['observed'])

# Plotting survival function

plt.figure(figsize=(10, 6))

kmf.plot()

plt.title('Kaplan-Meier Survival Curve')

plt.xlabel('Time (days)')

plt.ylabel('Survival probability')

plt.show()
```

5. Jupyter

It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Features

These are the important features of Jupyter

1. Loading Data and Displaying DataFrame
2. Plotting with Matplotlib and Seaborn
3. Inline Plotting with Jupyter Magic
4. Interactive Visualization with ipywidgets

Program:

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load example dataset

df = sns.load_dataset('iris')

# Display data with pandas DataFrame

print("Head of the dataset:")
```

```
display(df.head())
# Plotting with matplotlib
plt.figure(figsize=(8, 6))
sns.scatterplot(x='sepal_length', y='sepal_width', hue='species', data=df)
plt.title('Scatter Plot of Sepal Length vs Sepal Width')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
# Inline plotting with Jupyter magic command
%matplotlib inline
# More plotting with seaborn
plt.figure(figsize=(8, 6))
sns.boxplot(x='species', y='petal_length', data=df)
plt.title('Box Plot of Petal Length by Species')
plt.xlabel('Species')
plt.ylabel('Petal Length')
plt.show()
# Interactive widgets with ipywidgets
from ipywidgets import interact
def plot_species(species):
    sns.scatterplot(x='sepal_length', y='sepal_width', data=df[df['species'] == species])
    plt.title(f'Scatter Plot for {species}')
    plt.xlabel('Sepal Length')
    plt.ylabel('Sepal Width')
    plt.show()
# Interactive visualization with widgets
print("\nInteractive Visualization:")
interact(plot_species, species=df['species'].unique())
```

RESULT:

The features of packages like NumPy, SciPy, statsmodels, jupyter notebook and Pandas were explored.

Viva Questions

1. What is SciPy library in Python?
2. What is pandas used for?
3. Why NumPy is used in Python?
4. Is Pandas a library or package?
5. What are the applications of SciPy?

Ex. No: 2 Date:	Working with numpy array

1. Create a numpy nd array object by using array() function.

Aim:

To Create a numpy nd array object by using array() function.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as arr
step 4: print array
step 5: stop

Program:

```
import numpy as np
a=np.array([1,2,3,4,5])
print(a)
```

Output:

```
[1 2 3 4 5]
```

Result:

Thus the program to create a numpy nd array object by using array() function was executed and the output verified.

2. Use tuples to create a numpy array.

Aim:

To use tuples to create a numpy array.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as tuple
step 4: print array
step 5: stop

program:

```
import numpy as np
a=np.array((1,2,3,4,5))
print(a)
```

output:

```
[1 2 3 4 5]
```

Result:

Thus the program to use tuples to create a numpy array was executed and the output verified.

3. Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6.

Aim:

To create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an 2-d array of values 1,2,3 & 4,5,6
step 4: print array
step 5: stop

Program:

```
import numpy as np
arr=np.array([[1,2,3],[4,5,6]])
print(arr)
```

Output:

```
 [[1 2 3]
 [4 5 6]]
```

Result:

Thus the program to create a 2-D array was executed and the output verified.

4. create a 3-D array.

Aim:

To create a 3-D array.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an 3-d array
step 4: print array
step 5: stop

Program:

```
import numpy as np
a=np.array([[[1,2,3],[4,5,6]],[[1,2,3],[4,5,6]]])
print(a)
```

Output:

```
[[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]
```

Result: Thus a program to create a 3-D array was executed and the output verified.

5. Displaying dimensions of array from 0 to 3.

Aim:

To display the dimensions of array from 0 to 3.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare arrays of dimensions from 0 to 3
step 4: print dimensions using ndim function
step 5: stop

program:

```
import numpy as np
a=np.array(42)
b=np.array([1,2,3,4,5])
c=np.array([[1,2,3],[4,5,6]])
d=np.array([[[1,2,3],[4,5,6]],[[1,2,3],[4,5,6]]])
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

output:

```
0
1
2
3
```

Result:

Thus a program to display dimensions of array was executed and the output verified.

6. Accessing array elements by indexing and adding it.

Aim:

To access array elements by indexing and adding it.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as arr
step 4: access the 2nd & 3rd element by indexing
step 5: print a[2]+a[3]
step 6: stop

Program:

```
import numpy as np
a=np.array([1,2,3,4])
print(a[2]+a[3])
```

Output:

7

Result:
Thus a program to access array elements by indexing and adding it was executed and the output verified.

7. Access the element on the 2nd row 5th column.

Aim:
To access the element on the 2nd row 5th column.

Algorithm:
step 1: start
step 2: import numpy module
step 3: declare a 2-d array
step 4: print 5th element on 2nd row by accessing the index of the array
step 5: stop

Program:
```
import numpy as np
a=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print("5th element on 2nd row.....",a[1,4])
```

output:

5th element on 2nd row..... 10

Result:
Thus, a program to access the elements on the 2nd row 5th column was executed and the output verified.

8. Slice elements from index 1 to 5.

Aim:
To slice elements from index 1 to 5.

Algorithm:
step 1: start
step 2: import numpy module
step 3: declare an array as arr
step 4: print slicing the array from 1 to 5
step 5: stop

program:

```
import numpy as np
a=np.array([1,2,3,4,5])
print(a[1:5])
```

output:

[2 3 4 5]

Result:

Thus a program to slice elements from index 1 to 5 was executed and the output verified.

9. Slice elements from index 4 to the end of array.

Aim:

To slice elements from index 4 to the end of array.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as arr
step 4: print slicing the array from 4
step 5: stop

program:

```
import numpy as np
a=np.array([1,2,3,4,5])
print(a[4:])
```

output:

[5]

Result:

Thus a program to slice elements from index 4 to the end of array was executed and the output verified.

10. Slice elements from index 3 from the end to index 1 from end.

Aim:

To slice elements from index 3 from the end to index 1 from end.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as arr
step 4: print slicing the array by negative indexing from -5 to -1
step 5: stop

Program:

```
import numpy as np
a=np.array([1,2,3,4,5,6,7,8,9,10])
print(a[-5:-1])
```

Output:

[6 7 8 9]

Result:

Thus a program to slice elements from index 3 from the end to index 1 from end was executed and the output verified.

11. Print the shape of an array.

Aim:

To print the shape of an array.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare a 2-d array
step 4: print the shape of an array using arr.shape
step 5: stop

Program:

```
import numpy as np
a=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print(a.shape)
```

Output:

(2,5)

Result:

Thus a program to print the shape of an array was executed and the output verified.

12. Iterate on the element of 1-D array.

Aim:

To iterate on the element of 1-D array.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as arr
step 4: using for loop print the elements of the arr
step 4: print i
step 5: stop

program:

```
import numpy as np
a=np.array([1,2,3,4,5])
for i in a:
 print (i)
```

Output:

1
2
3
4
5

Result:

Thus a program to iterate on the element of 1-D array was executed and the output verified.

13. Split the array in 3 parts.

Aim:

To split the array in 3 parts.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as a
step 4: create a variable newarr with array_split function to split the array into 3 parts
step 5: print newarr
step 6: stop

program:

```
import numpy as np
a=np.array([1,2,3,4,5,6])
newarr=np.array_split(a,3)
print(newarr)
```

Output:

[array([1, 2]), array([3, 4]), array([5, 6])]

Result:

Thus the program to split the array in 3 parts was executed and the output verified.

14. Find the indexes where the value is 4.

Aim:

To write a program to find the indexes where the value is 4.

Algorithm:

step 1: start
step 2: import numpy module

step 3: declare an array as arr
step 4: create a variable x using the function 'where' find the indexes whcih has the value 4
step 5: print x
step 6: stop

program:

```
import numpy as np
a=np.array([1,2,3,4,5,4,4])
x=np.where(a==4)
print(x)
```

output:

```
(array([3, 5, 6]),)
```

Result:
Thus the program to find the indexes where the value is 4 was executed and the output verified.

15. Find the indexes where the value is even.

Aim:
To write a program to find the indexes where the value is even.

Algorithm:

step 1: start
step 2: import numpy module
step 3: declare an array as a
step 4: create a variable x using the function 'where' find the indexes where the value is even
step 5: print x
step 6: stop

program:

```
import numpy as np
a=np.array([1,2,3,4,5,6,7,8])
x=np.where(a%2==0)
print(x)
```

output:

```
(array([1, 3, 5, 7]),)
```

Result:
Thus a program to find the indexes where the value is even was executed and the output verified.

16. Find the indexes where the value is odd.

Aim:
To write a program to find the indexes where the value is odd.

Algorithm:
step 1: start
step 2: import numpy module
step 3: declare an array as a
step 4: create a variable x using the function 'where' find the indexes where the value is odd
step 5: print x
step 6: stop

program:
```
import numpy as np
a=np.array([1,2,3,4,5,6,7,8])
x=np.where(a%2==1)
print(x)
```

Output:
(array([0, 2, 4, 6]),)

Result:
Thus a program to find the indexes where the value is odd was executed and the output verified.

17. Sort the array.

Aim:
To write a program to sort the given array.

Algorithm:
step 1: start
step 2: import numpy module
step 3: declare an array as a
step 4: print the sorted array using sort function
step 5: stop

program:
```
import numpy as np
a=np.array([3,2,0,1])
print(np.sort(a))
```

Output:

[0 1 2 3]

Result:
Thus, the program to sort the given array was executed and the output verified.

18. Sort the array alphabetically.

Aim:
To write a program to sort the given array alphabetically.

Algorithm:
step 1: start
step 2: import numpy module
step 3: declare an array as a which consists of strings
step 4: print the sorted array using sort function
step 5: stop

program:
```
import numpy as np
a=np.array('banana','cherry','apple')
print(np.sort(a))
```

Output:
['apple' 'banana' 'cherry']

Result:
Thus the program to sort the given array alphabetically was executed and the output verified.

19. Create an array from the elements on index 0 and 2.

Aim:
To create an array from the elements on index 0 and 2.

Algorithm:
step 1: start
step 2: import numpy module
step 3: declare an array as a
step 4: create another array as x which has boolean values(True & False)
step 5: assign the variable newarr to a[x]
step 6: print newarr
step 7: stop

program:
```
import numpy as np
a=np.array([41,42,43,44])
x=[True,False,True,False]
newarr=a[x]
print(newarr)
```

Output:
[41 43]

Result: Thus, a program to create an array from the elements on index 0 and 2 was executed and the output verified.

Viva Questions:

1. What is difference between NumPy and SciPy?
2. What is array in Python?
3. What is the use of NumPy array in Python?
4. What is NumPy full form?
5. What is the benefit of NumPy?

Ex. No: 3 Date:	**Working with pandas data frame**

3(a). Create a data frame using a list of elements

Aim:

To write a python program to create a data frame using a list of elements.

Algorithm:

step 1: start
step 2: import pandas as pd
step 3: declare a list consists of elements a to g.
Step 4: assign df to pd.DataFrame(lst)
step 5: print df
step 6: stop

Program:

```
import pandas as pd
lst=['A','B','C','D','E','F','G']
df=pd.DataFrame(lst)
print(df)
```

Output

```
   0
0  A
1  B
2  C
3  D
4  E
5  F
6  G
```

Result:

Thus the program to create a data frame using a list of elements was executed and the output was verified.

3(b). Create a data frame using the dictionary.

Aim:

To write a python program to create a data frame using the dictionary.

Algorithm:

step 1: start
step 2: import pandas as pd
step 3: assign a variable data to a dictionary which consists of keys: name and age.
step 4: assign df to pd.DataFrame(lst)
step 5: print df
step 6: stop

Program:

```
import pandas as pd
data={'name':['tom','nick','krish','jack'],'age':[20,21,19,18]}
df=pd.DataFrame(data)
print(df)
```

Output:

```
   name  age
0   tom   20
1  nick   21
2 krish   19
3  jack   18
```

Result:

Thus a program to create a data frame using the dictionary was executed and the output was verified.

3(c). select a column from data frame

Aim:

To write a python program to select a column from data frame.

Algorithm:

step 1: start
step 2: import pandas as pd
step 3: assign a variable data to a dictionary which consists of keys: name , age and qualification.
step 4: assign df to pd.DataFrame(data)
step 5: print df
step 6: print df with column name
step 7: stop

Program:

```
import pandas as pd
data = {
   'name': ['jai', 'princy', 'gaurav', 'anuj'],
   'age': [27, 24, 22, 32],
   'address': ['delhi', 'kanpur', 'allahabad', 'kannauj'],
   'qualification': ['MA', 'MCA', 'Phd', 'BE']
}
df = pd.DataFrame(data)
print(df)
print(df[['name', 'qualification']])
```

output:

```
     name  age    address qualification
0     jai   27      delhi            MA
1  princy   24     kanpur           MCA
2  gaurav   22  allahabad           Phd
3    anuj   32    kannauj            BE
     name qualification
0     jai            MA
1  princy           MCA
2  gaurav           Phd
3    anuj            BE
```

Result:

Thus a program to select a column from data frame was executed and the output was verified.

3(d). Checking for missing values using isnull() and notnull()

Aim:

To write a python program to check for missing values using isnull() and notnull().

Algorithm:

Step 1: start
step 2: import pandas as pd
step 3: import numpy as np
step 4: assign a variable dic to a dictionary consists of keys : first, second and third.
Step 5: assign df to pd.DataFrame(dic)
step 6: print(df.isnull())
step 7: stop

program:

```
import pandas as pd
import numpy as np
dic={'first':[100,90,np.nan,95],'second':[30,45,56,np.nan],'third':[np.nan,40,80,98]}
df=pd.DataFrame(dic)
print(df.isnull())
```

output:

```
   first  second  third
0  False   False   True
1  False   False  False
2   True   False  False
3  False    True  False
```

Result:

Thus a program to check for missing values using isnull() and notnull() was executed.

Viva Questions:

1. What is pandas DataFrame in Python?
2. How do I show DataFrame in pandas Python?
3. Where is pandas DataFrame Python?
4. What is a DataFrame?
5. How do you read a DataFrame in Python?

Ex. No: 4 Date:	**Reading Data from Text Files, Excel and The Web and Exploring Various Commands For Doing Descriptive Analytics On The Iris Data Set**

Aim

To read data from the text files, excel and the web and exploring various commands for doing descriptive analytics on the iris data set.

READING DATA FROM EXCEL FILE

Algorithm

Step 1: Start

Step2: Importing pandas as pad

Step 3: using read_csv() function to read the csv file

Step 4: Printing the df with to_string()

Step 5: Stop

Requirement:

open pyxl module to write data to xlsx file user warning pandas requires version 0.98 or neer of " xlsxwriter "(version 0.96 currently installed)

```
!pip install openpyxl     # needs openpyxl module
```

Program:

Read data from a csv file:

```
import pandas as pd
# reading the csv file
df = pd.read_csv('iris_dataset.csv')
# change path correctly
# print data from csv
print(df.to_string())
print(df.describe())
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa
10	11	5.4	3.7	1.5	0.2	Iris-setosa
11	12	4.8	3.4	1.6	0.2	Iris-setosa
12	13	4.8	3.0	1.4	0.1	Iris-setosa
13	14	4.3	3.0	1.1	0.1	Iris-setosa
14	15	5.8	4.0	1.2	0.2	Iris-setosa
15	16	5.7	4.4	1.5	0.4	Iris-setosa
16	17	5.4	3.9	1.3	0.4	Iris-setosa
17	18	5.1	3.5	1.4	0.3	Iris-setosa
18	19	5.7	3.8	1.7	0.3	Iris-setosa
19	20	5.1	3.8	1.5	0.3	Iris-setosa
20	21	5.4	3.4	1.7	0.2	Iris-setosa
21	22	5.1	3.7	1.5	0.4	Iris-setosa
22	23	4.6	3.6	1.0	0.2	Iris-setosa
23	24	5.1	3.3	1.7	0.5	Iris-setosa
24	25	4.8	3.4	1.9	0.2	Iris-setosa
25	26	5.0	3.0	1.6	0.2	Iris-setosa
26	27	5.0	3.4	1.6	0.4	Iris-setosa
27	28	5.2	3.5	1.5	0.2	Iris-setosa
28	29	5.2	3.4	1.4	0.2	Iris-setosa
29	30	4.7	3.2	1.6	0.2	Iris-setosa
30	31	4.8	3.1	1.6	0.2	Iris-setosa
31	32	5.4	3.4	1.5	0.4	Iris-setosa
32	33	5.2	4.1	1.5	0.1	Iris-setosa
33	34	5.5	4.2	1.4	0.2	Iris-setosa

READING DATA FROM TEXT FILE

Algorithm:

Step 1: Start

Step 2: Importing pandas as pd

Step 3: Using read_csv() function to read file

Step 4: Printing the df

Step 5: Stop

Result

Thus, a program to read text file, excel and the web and exploring various commands for doing descriptive analytics on the data set was successfully executed.

Viva Questions:

1. What is data sampling method?
2. How do you create a sample dataset in Python?
3. Why sampling is used in Python?
4. What are the 4 types of samples?
5. How many types of data sampling are there?

Ex. No: 5 Date:	Use the diabetes data set from UCI and PIMA Indian diabetes

5. (a) Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

Aim:
To analyse the Pima Indians diabetes data set for Univariate like Frequency ,Mean, median , etc..

Algorithm:
Step 1: Start
Step 2: importing pandas as pd
Step 3: importing numpy as np
Step 4: importing statistics as st
Step 5: reading csv file using read_csv function
Step 6: printing information , shape ,mean, median , mode , standard deviation, variance skew, kurtosis of data set
Step 7: importing matplotlib as plt
Step 8: representing the data in graphs charts
Step 9: Stop

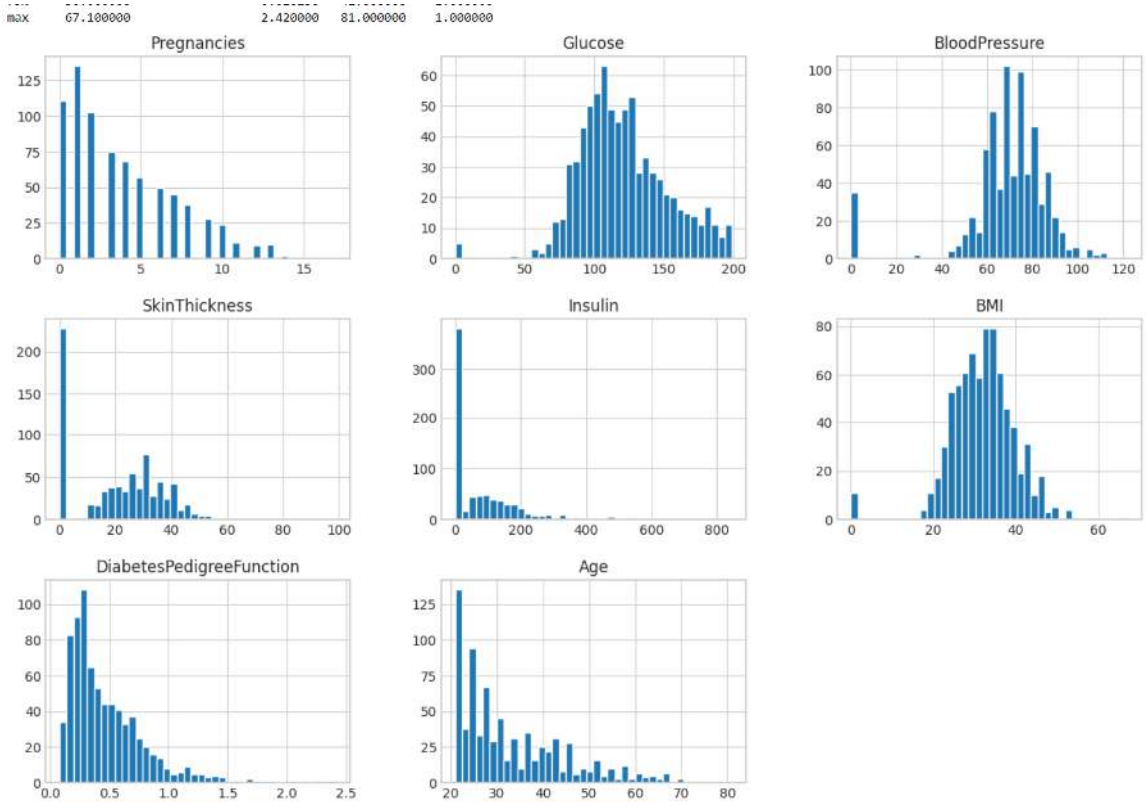
Note: pip install matplotlib

Program:

```
import pandas as pd
import numpy as np
import statistics as st
df=pd.read_csv("pima.csv",engine='python')
print(df.shape)
print(df.info())
print('MEAN:\n',df.mean())
print('MEDIAN:\n:',df.median())
print('MODE:\n:',df.mode())
print('STANDARD DEVIATION:\n:',df.std())
print('VARIANCE:\n:',df.var())
print('SKEWNESS:\n:',df.skew())
print('KURTOSIS:\n:',df.kurtosis())
df.describe()
Data_X=df.copy(deep=True)
print(Data_X.columns)
Data_X=Data_X.drop(['Outcome'],axis=1) # if outcome doesn't exist delete this line
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize']=[40,40]
Data_X.hist(bins=40)

plt.show()
```

Output

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float
 6   DiabetesPedigreeFunction  768 non-null    float
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
None
MEAN:
 Pregnancies                 3.845052
Glucose                    120.894531
BloodPressure               69.105469
SkinThickness               20.536458
Insulin                     79.799479
BMI                         31.992578
DiabetesPedigreeFunction     0.471876
Age                         33.240885
Outcome                      0.348958
dtype: float64
MEDIAN:
 Pregnancies                 3.0000
Glucose                    117.0000
BloodPressure               72.0000
SkinThickness               23.0000
Insulin                     30.5000
BMI                         32.0000
DiabetesPedigreeFunction     0.3725
Age                         29.0000
Outcome                      0.0000
dtype: float64
MODE:
 Pregnancies                 1.000
Glucose                    99.000
BloodPressure              70.000
SkinThickness               0.000
Insulin                     0.000
BMI                        32.000
Insulin                     0.000
BMI                        32.000
DiabetesPedigreeFunction    0.254
Age                        22.000
Outcome                     0.000
Name: 0, dtype: float64
STANDARD DEVIATION:
 Pregnancies                 3.369578
Glucose                    31.972618
BloodPressure              19.355807
SkinThickness              15.952218
Insulin                   115.244002
BMI                         7.884160
DiabetesPedigreeFunction    0.331329
Age                        11.760232
Outcome                     0.476951
dtype: float64
VARIANCE:
 Pregnancies                11.354056
Glucose                  1022.248314
BloodPressure             374.647271
SkinThickness             254.473245
Insulin                 13281.180078
BMI                        62.159984
DiabetesPedigreeFunction    0.109779
Age                       138.303046
Outcome                     0.227483
dtype: float64
SKEWNESS:
 Pregnancies                0.901674
Glucose                    0.173754
BloodPressure             -1.843608
SkinThickness              0.109372
Insulin                    2.272251
BMI                       -0.428982
DiabetesPedigreeFunction   1.919911
Age                        1.129597
Outcome                    0.635017
dtype: float64
KURTOSIS:
 Pregnancies                0.159220
Glucose                    0.640780
BloodPressure              5.180157
SkinThickness             -0.520072
Insulin                    7.214260
BMI                        3.290443
DiabetesPedigreeFunction   5.594954
Age                        0.643159
```



5 (b) Bivariate analysis: Linear regression modeling

Aim:
To analyse the Bivariate operations and Linear regression modeling

Algorithm:
Step 1: Start
Step 2: Import matplotlib as plt
Step 3: Import Numpy as np
Step 4: Import pandas as pd
Step 5: using sklearn module loading the data set
Step 6: calculating the mean squared error
Step 7: then calculating the mse and rmse
Step 8: printing the values of mse and rmse
Step 9: stop

Program:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets,linear_model
from sklearn.metrics import mean_squared_error
diabetes=datasets.load_diabetes()
diabetes.keys()
```

```
df=pd.DataFrame(diabetes['data'],columns=diabetes['feature_names'])
x=df
y=diabetes['target']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=101)
from sklearn import linear_model
model=linear_model.LinearRegression()
model.fit(x_train,y_train)
y_pre=model.predict(x_test)
from sklearn.model_selection import cross_val_score
scores=cross_val_score(model,x,y,scoring="neg_mean_squared_error",cv=10)
rmse_scores=np.sqrt(-scores).mean()
print('Cross validation:',rmse_scores)
from sklearn.metrics import r2_score
print('r^2:',r2_score(y_test,y_pre))
mse=mean_squared_error(y_test,y_pre)
rmse=np.sqrt(mse)
print('RMSE:',rmse)
print("Weights:",model.coef_)
print("\nIntercept",model.intercept_)
```

Output

Cross validation: 54.40461553640237
r^2: 0.45767674177195583
RMSE: 58.009275047551995
Weights: [-8.02566358 -308.83945001 583.63074324 299.9976184 -360.68940198
95.14235214 -93.03306818 118.15005596 662.12887711 26.07401648]
Intercept 153.72029738615726

5. (b) Bivariate analysis: Logistics regression modeling

Aim

To analyse the Bivariate operations on logistics regression modeling

Algorithm

Step 1: Start
Step 2: Import matplotlib as plt
Step 3: Import Numpy as np
Step 4: Import pandas as pd
Step 5: using sklearn module loading the data set
Step 6: calculating the mean squared error
Step 7: then calculating the mse and rmse
Step 8: printing the values of mse and rmse
Step 9: stop

Program

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn import datasets#,linear_model
```

```
from sklearn.metrics import mean_squared_error
diabetes = datasets.load_diabetes()
diabetes.keys()
df=pd.DataFrame(diabetes['data'],columns=diabetes['feature_names'])
x=df
y=diabetes['target']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=101)
#import Model
#from sklearn
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
y_pre= model.predict(x_test)
from sklearn.metrics import r2_score
print('r^2:',r2_score(y_test,y_pre))
mse=mean_squared_error(y_test,y_pre)
rmse=np.sqrt(mse)
print('RMSE:',rmse)
```

Output

r^2: -0.44401265478624397

RMSE: 94.65723681369009

5 (c) Multi regression line analysis

Aim

To Perform the multi regression analysis

Algorithm

step 1: Start
step 2: importing matplotlib as pd
step 3: importing numpy as np
step 4: importing sklearn
step 5: importing pandas as pd
step 6: reading csv file
step 7: fetching data from data set
step 8: calculating the mean squared value
step 9: then calculating the mse and rmse
step 10: printing the mse and rmse
step 11: Stop

Program:

```
import matplotlib.pyplot as pd
import numpy as np
```

```
from sklearn import datasets, linear_model, metrics
import pandas as pd
import numpy as np
df=pd.read_csv('pima-indians-diabetes.csv')
data=df[['Age','Glucose','Bmi','Blood pressure','Pregnancies']]
target = df[['Outcome']]
print(data)
print(target)
X=data
Y=target
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train ,Y_test = train_test_split(X,Y,test_size = 0.3,random_state=101)
reg=linear_model.LinearRegression()
reg.fit(X_train ,Y_train )
Y_predict=reg.predict(X_test)
print('Coeffients',reg.coef_)
print('Variance Scores : {}'.format(reg.score(X_test,Y_test)))
from sklearn.metrics import r2_score
print('r^2 :',r2_score(Y_test,Y_predict))
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(Y_test,Y_predict)
rmse=np.sqrt(mse)
print('RMSE:',rmse)
```

Output

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
   Age  Glucose   BMI  BloodPressure  Pregnancies
0   50      148  33.6             72            6
1   31       85  26.6             66            1
2   32      183  23.3             64            8
3   21       89  28.1             66            1
4   33      137  43.1             40            0
0    1
1    0
2    1
3    0
4    1
Name: Outcome, dtype: int64
Coefficients: [ 0.00362921  0.0057603   0.01359201 -0.0022797   0.01903324]
Intercept: -0.8190744607047454
Variance Score: 0.3119613858813981
r^2: 0.3119613858813981
RMSE: 0.3958061749043919
```

Result:

Thus, a program to check read data from text files, excel and the web and exploring various commands for doing descriptive analytics on the iris data set was executed and the output was verified.

Viva Questions:

1. How to extract data from text file to Excel using Python?
2. How do I read data from an Excel file in Python?
3. How do I read XLS and xlsx files in Python?
4. Can Python read and write Excel files?
5. Can you read a text file in Python?

Ex. No: 6 Date:	Apply And Explore Various Plotting Functions On UCI Data Sets.

6 (a). Normal curves

Aim

To Apply and explore various plotting function on UCI data set for normal curve

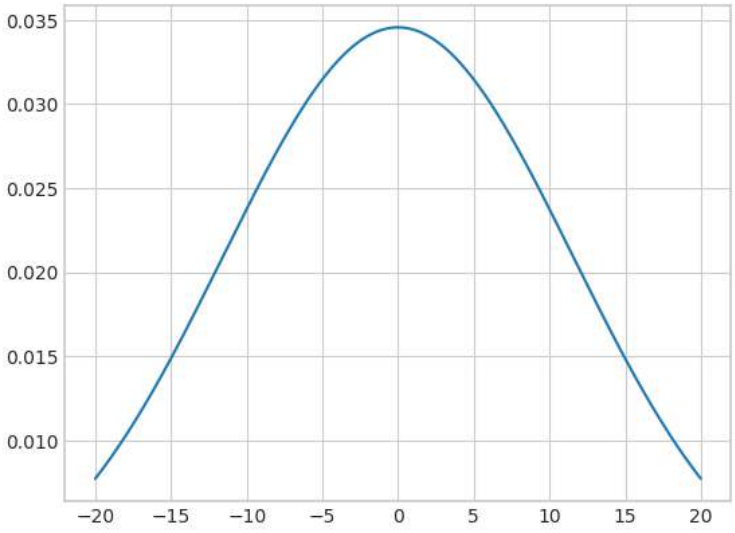
Algorithm

Step 1: start
Step 2: importing numpy as np
Step 3: import matplotlib as plt
Step 4: import scipy.stats
Step 5: import statistics
Step 6: creating np array with x, y dimensions
Step 7: using matplotlib showing the graph
Step 8: stop

Program

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics
x_axis = np.arange(-20, 20, 0.01)
# Calculating mean and standard deviation
mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()
```

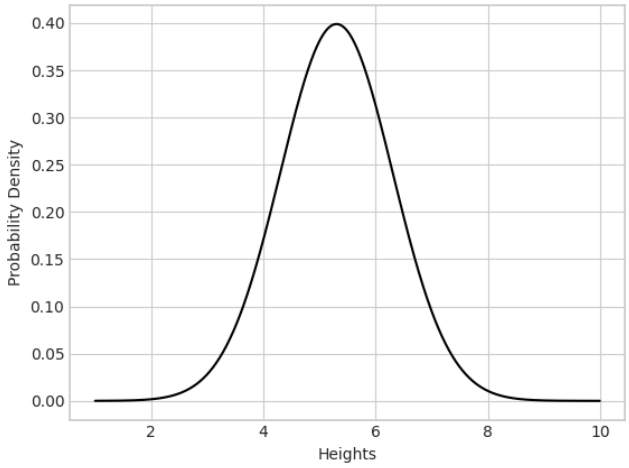
Output



Program

```
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
data = np.arange(1, 10, 0.01)
pdf = norm.pdf(data, loc=5.3, scale=1)
sb.set_style('whitegrid')
sb.lineplot(x=data, y=pdf, color='black')
plt.xlabel('Heights')
plt.ylabel('Probability Density')
plt.show()
```

OUTPUT:



Result

Thus, the program for Apply and explore various plotting function on UCI data set for normal curve was executed and output obtained.

6. (b) Density and contour plots

Aim

To Apply and explore various plotting function on UCI data set for density and contour plots

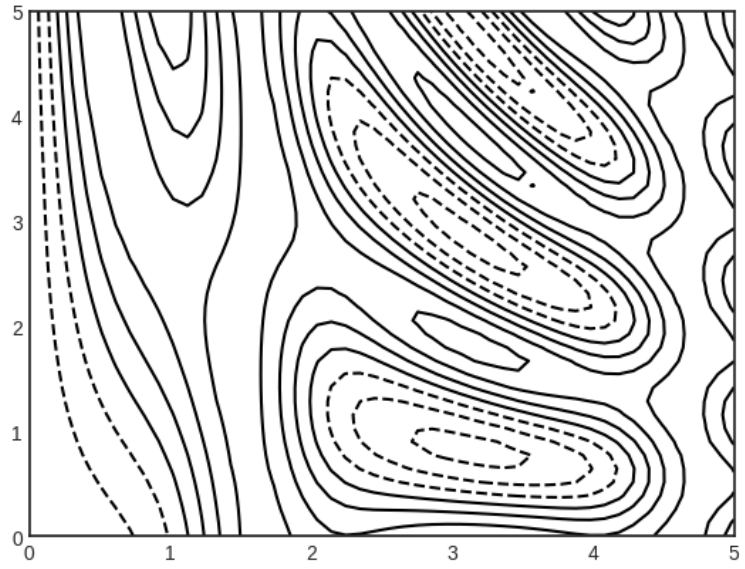
Algorithm

Step 1: start
Step 2: importing numpy as np
Step 3: import matplotlib as plt
Step 4: creating np array with x, y dimensions
Step 5: using matplotlib showing the graph
Step 6: stop

Program

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import numpy as np
def f(x, y):
    return np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
plt.contour(X, Y, Z, colors='black');
```

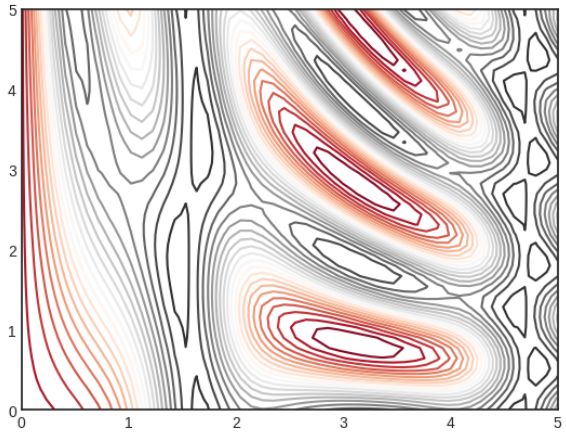
Output



Program

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import numpy as np
def f(x, y):
    return np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
plt.contour(X, Y, Z, 20, cmap='RdGy');
```

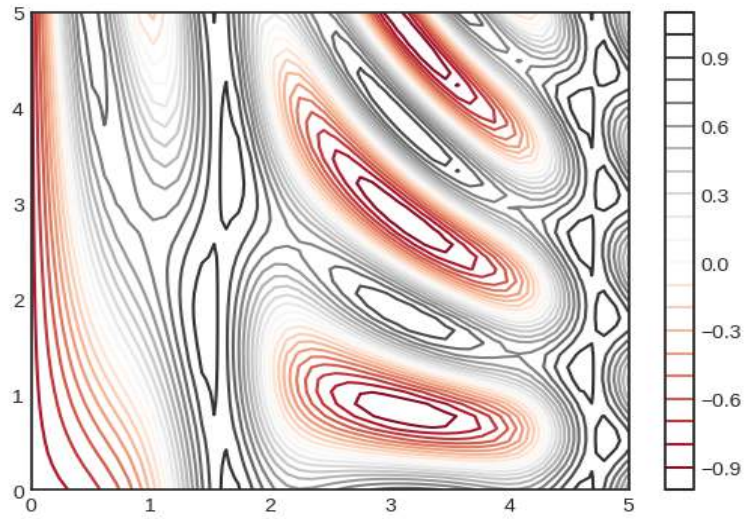
Output



Program

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
import numpy as np
def f(x, y):
    return np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
x = np.linspace(0, 5, 50)
y = np.linspace(0, 5, 40)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
plt.contour(X, Y, Z, 20, cmap='RdGy')
plt.colorbar();
```

Output



Result: Thus, the program for Apply and explore various plotting function on UCI data set for Density and contour plot was executed and output obtained.

6. (c) Correlation and scatter plots

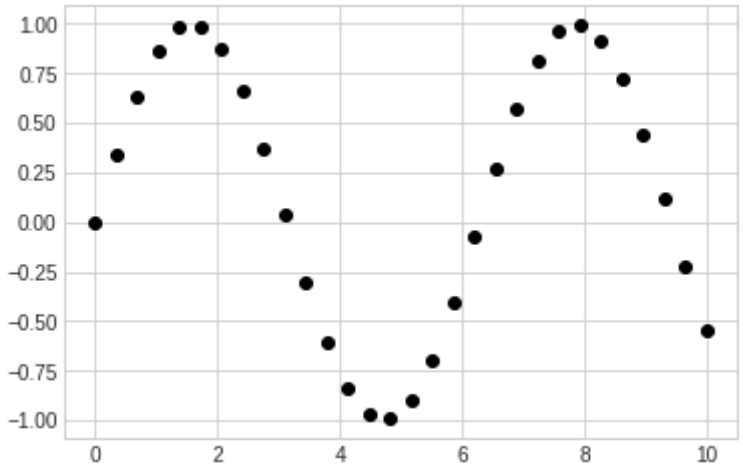
Aim:
To Apply and explore various plotting function on UCI data set from Correlation and scatter plots

Algorithm:
Step 1: start
Step 2: importing numpy as np
Step 3: import matplotlib as plt
Step 4: creating np array with x, y dimensions
Step 5: using matplotlib showing the graph
Step 6: stop

Program:

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
x = np.linspace(0,10,30)
y = np.sin(x)
plt.plot(x, y, 'o', color='black');
```

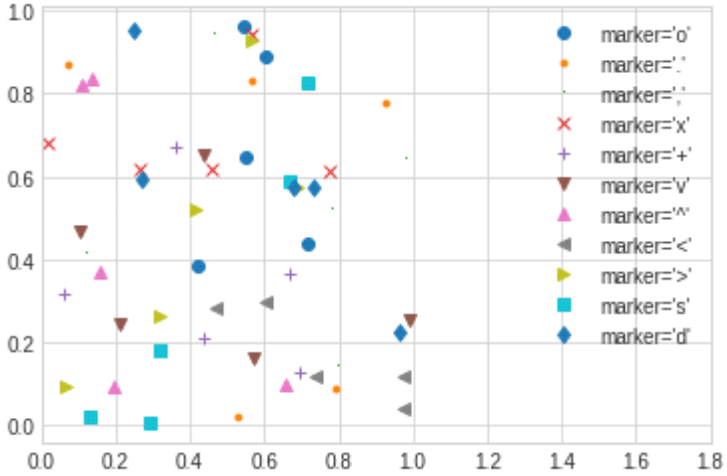
Output



Program

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
rng = np.random.RandomState(0)
for marker in ['o', '.', ',', 'x', '+', 'v', '^', '<', '>', 's', 'd']:
    plt.plot(rng.rand(5), rng.rand(5),marker,   label="marker='{0}'".format(marker))
    plt.legend(numpoints=1)
    plt.xlim(0, 1.8);
```

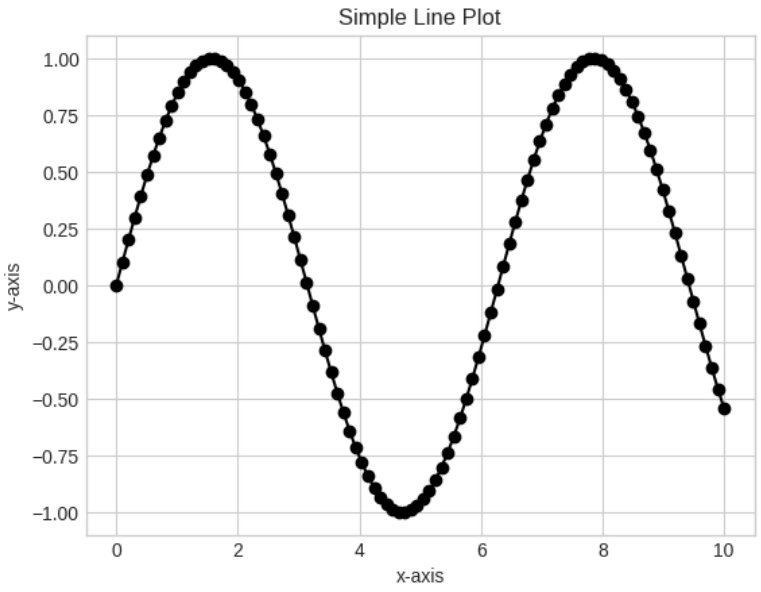
Output



Program

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
# Sample data for x and y
x = np.linspace(0, 10, 100)
y = np.sin(x)
# Plotting the data
plt.plot(x, y, '-ok')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Simple Line Plot')
plt.show()
```

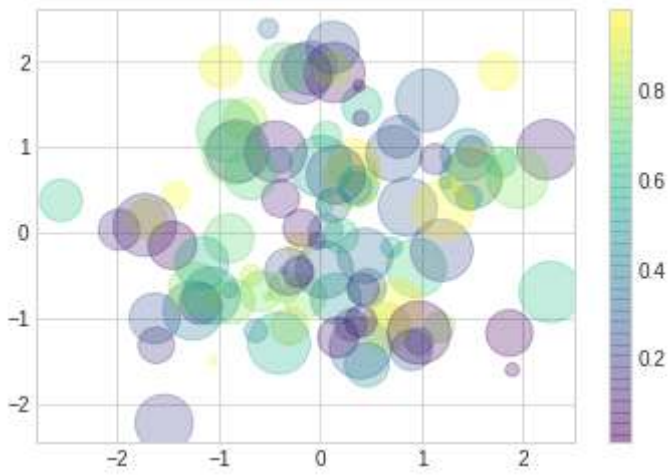
Output



Program

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np
rng=np.random.RandomState(0)
x=rng.randn(100)
y=rng.randn(100)
colors=rng.rand(100)
sizes=1000 * rng.rand(100)
plt.scatter(x,y,c=colors,s=sizes,alpha=0.3,cmap='viridis')
plt.colorbar();#show color scale
```

Output



Result

Thus, the program for Apply and explore various plotting function on UCI data set for correlation and scatterplot was executed and output obtained.

6. (d) Histograms

Aim

To Apply and explore various plotting function on UCI data set for Histogram

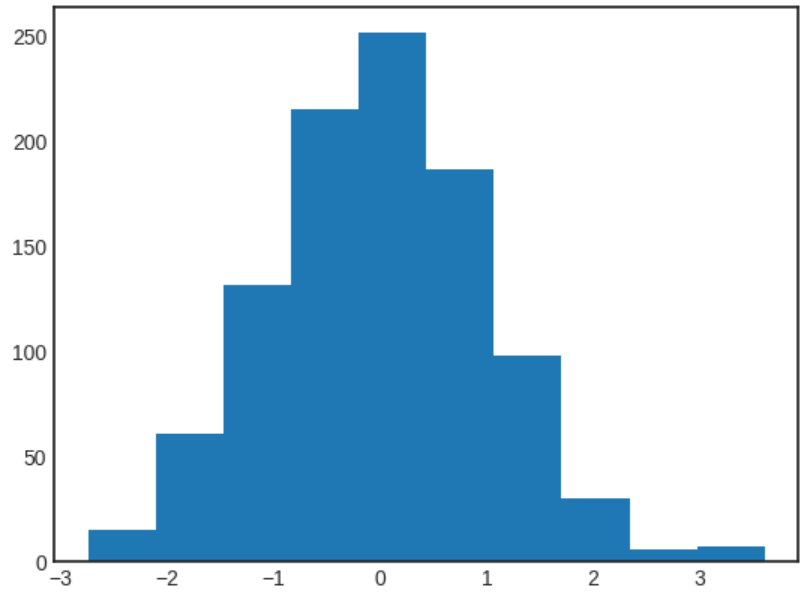
Algorithm

Step 1: start
Step 2: importing numpy as np
Step 3: import matplotlib as plt
Step 4: creating np array with x, y dimensions
Step 5: using matplotlib showing the graph
Step 6: stop

Program

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
data = np.random.randn(1000)
plt.hist(data)
plt.hist(data, bins=30, density=True, alpha=0.5, histtype='stepfilled', color='steelblue',
edgecolor='none')
plt.show()
```

Output



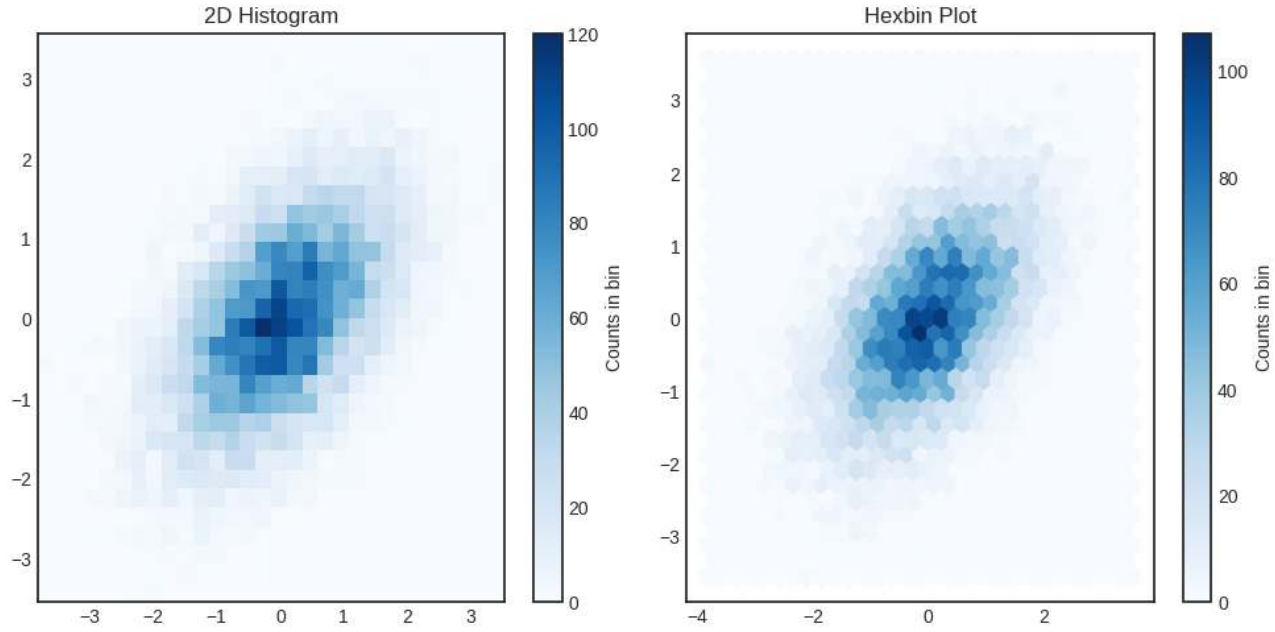
Two-dimensional histogram and binnings

Program

```
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('seaborn-white')
mean = [0, 0]
cov = [[1, 0.5], [0.5, 1]]  # This is a valid covariance matrix

# Generate multivariate normal data
x, y = np.random.multivariate_normal(mean, cov, 10000).T
# Create a 2D histogram plot
plt.figure(figsize=(10, 5))

# 2D Histogram plot
plt.subplot(1, 2, 1)
plt.hist2d(x, y, bins=30, cmap='Blues')
cb = plt.colorbar()
cb.set_label('Counts in bin')
plt.title('2D Histogram')
```

```
# Hexbin plot
plt.subplot(1, 2, 2)
plt.hexbin(x, y, gridsize=30, cmap='Blues')
cb = plt.colorbar(label='Counts in bin')
plt.title('Hexbin Plot')

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

Output



Result

Thus, the program for Apply and explore various plotting function on UCI data set for histogram was executed and output obtained.

6. (d) Three-dimensional plotting

Aim

To Apply and explore three-dimensional plotting function on UCI data set

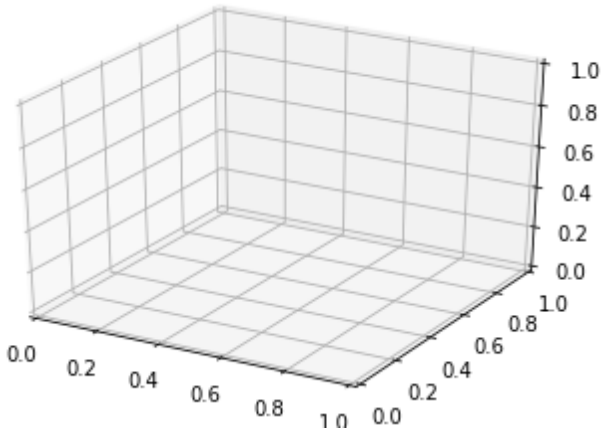
Algorithm

Step 1: start
Step 2: importing numpy as np
Step 3: import matplotlib as plt
Step 4: creating np array with x, y dimensions
Step 5: using matplotlib showing the graph
Step 6: stop

Program

```
from mpl_toolkits import mplot3d
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
ax=plt.axes(projection='3d')
```

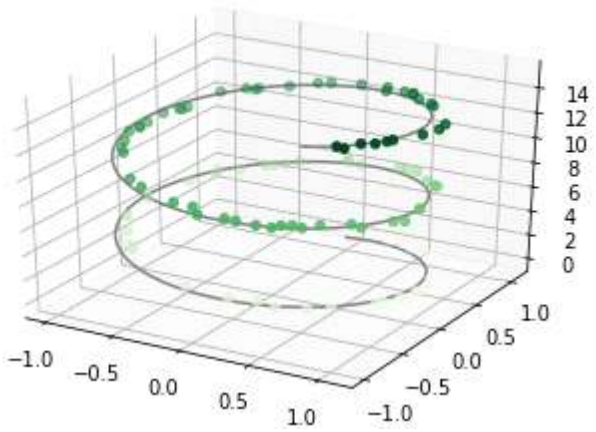
Output



Program

```
from mpl_toolkits import mplot3d
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt

ax=plt.axes(projection='3d')
zline = np.linspace(0,15,1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline,yline,zline,'gray');
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.random(100)
ydata = np.cos(zdata) + 0.1 * np.random.random(100)
ax.scatter3D(xdata,ydata,zdata, c=zdata, cmap='Greens');
```

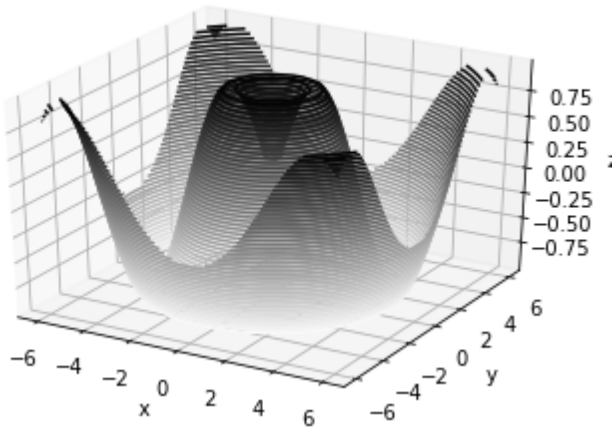
Output



Program

```
from mpl_toolkits import mplot3d
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
def f(x,y):
        return np.sin(np.sqrt(x ** 2 + y ** 2))
x = np.linspace(-6,6,30)
y = np.linspace(-6,6,30)
X, Y = np.meshgrid(x,y)
z = f(X,Y)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.contour3D(x,y,z,50,cmap='binary')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z');
```

Output



Result

Thus, the program for Apply and explore various plotting function on UCI data set for Three-dimensional plotting was executed and output obtained.

Viva Questions:

1. What is the algorithm for clustering data?
2. What is classification in data science?
3. What are the 3 main types of data classification?
4. What is classification and clustering of data?
5. How to cluster data using Python?

Ex. No: 7 Visualizing geographic data with basemap

Aim

To visualizing geographic data with basemap

Requirement

!apt install proj-bin libproj-dev libgeos-dev
!pip install https://github.com/matplotlib/1.0.tar.gz
!pip install Basemap

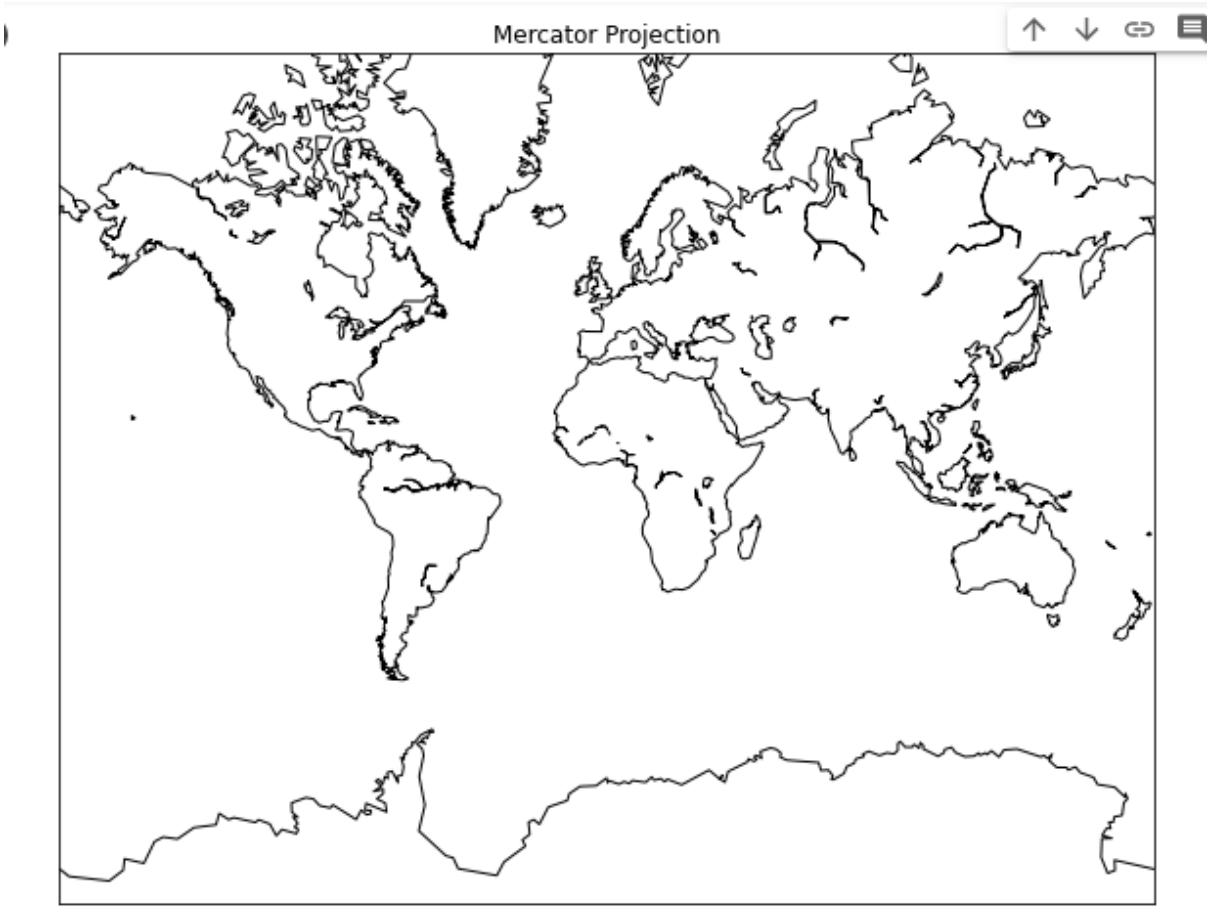
Algorithm

Step 1: start
Step 2: importing numpy as np
Step 3: import matplotlib as plt
Step 4: import Basemap
Step 5: using matplotlib showing the graph
Step 6: stop

Program

```
from mpl_toolkits.basemap import Basemap

import matplotlib.pyplot as plt

import numpy as np

%matplotlib inline

import warnings

import matplotlib.cbook

warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)

Basemap?

fig =plt.figure(num=None,figsize=(12,8))

m =Basemap(projection='merc',llcrnrlat=-80,urcrnrlat=80,llcrnrlon=-
180,urcrnrlon=180,resolution='c')

m.drawcoastlines()

plt.title("Mercator Projection")

plt.show()
```

OUTPUT:

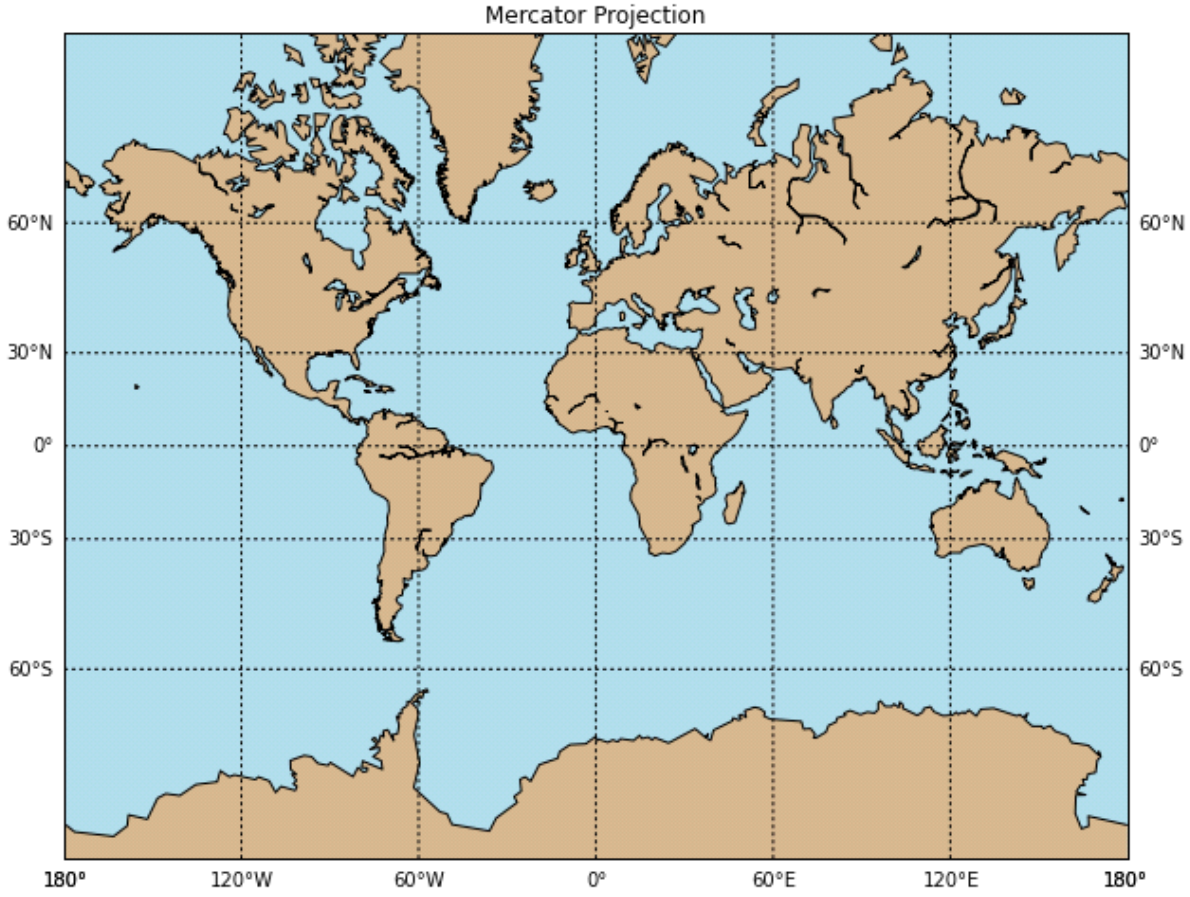


PROGRAM:

```
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import warnings
import matplotlib.cbook
warnings.filterwarnings("ignore",category=matplotlib.cbook.mplDeprecation)
Basemap?
fig =plt.figure(num=None,figsize=(12,8))
m =Basemap(projection='merc',llcrnrlat=-80,urcrnrlat=80,llcrnrlon=-
180,urcrnrlon=180,resolution='c')
```

```
m.drawcoastlines()
m.fillcontinents(color='tan',lake_color='lightblue')
m.drawparallels(np.arange(-90.,91.,30.),labels=[True,True,False,False],dashes=[2,2])
m.drawmeridians(np.arange(-180.,181.,60.),labels=[False,False,False,True],dashes=[2,2])
m.drawmapboundary(fill_color='lightblue')
plt.title("Mercator Projection")
```

Output



Result

Thus, visualizing geographic data with basemap is implemented.

VIVA QUESTIONS

1. What is UCI dataset in Python?
2. Where can I download datasets?
3. Which Python library is used by dataset?
4. What are the types of dataset in Python?
5. What are 2 types of data sets?

Ex. No: 8 Correlation coefficient

Aim:

To write a python program to compute correlation coefficient.
ALGORITHM:

Step 1: Start the Program
Step 2: Import math package
Step 3: Define correlation coefficient function
Step 4: Calculate correlation using formula
Step 5: Print the result
Step 6: Stop the process

Program: # Python Program to find correlation coefficient.

```
Import math
# Function that returns correlation coefficient.
 def correlationCoefficient(X, Y, n) :

        sum_X = 0
        sum_Y = 0
        sum_XY = 0
        squareSum_X = 0
         squareSum_Y = 0

i = 0
while i < n :
        # sum of elements of array X.
         sum_X = sum_X + X[i]

        # sum of elements of array Y.
         sum_Y = sum_Y + Y[i]

        # sum of X[i] * Y[i].
        sum_XY = sum_XY + X[i] * Y[i]

        # sum of square of array elements.
        squareSum_X = squareSum_X + X[i] * X[i]

        squareSum_Y = squareSum_Y + Y[i] * Y[i]
         i = i + 1

# use formula for calculating correlation
# coefficient.
 corr = (float)(n * sum_XY - sum_X * sum_Y)/
                (float)(math.sqrt((n * squareSum_X - sum_X * sum_X)* (n * squareSum_Y -
                sum_Y * sum_Y)))
return corr
```

```
 # Driver function
X = [15, 18, 21, 24, 27
Y = [25, 25, 27, 31, 32]
# Find the size of array.
n = len(X)
# Function call to correlationCoefficient.
print ('{0:.6f}'.format(correlationCoefficient(X, Y, n)))
```

Output : 0.953463

VIVA QUESTIONS:

1. **Define partial correlation and correlation**
2. **Which of the following are types of correlation?**
3. **Which of the following is true for the coefficient of correlation?**
4. **What is the meaning of the testing of the hypothesis?**
5. **What is the value of the correlation coefficient?**