SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur-603203

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

ACADEMIC YEAR: 2024-2025

EVEN SEMESTER

LAB MANUAL

(REGULATION - 2023)

AD3467 – DATA SCIENCE AND ANALYTICS LABORATORY

FOURTH SEMESTER

B.Tech – Artificial Intelligence and Data Science

Prepared By

R. VAISHNAVI , A.P (O.G) / AI&DS

AD3467 DATA SCIENCE AND ANALYTICS LABORATORY L T P C

0 0 3 1.5

OBJECTIVES:

- To develop data analytic code in python
- To be able to use python libraries for handling data
- To develop analytical applications using python
- To perform data visualization using plots

LIST OF EXPERIMENTS:

Tools: Python, Numpy, Scipy, Matplotlib, Pandas, statmodels, seaborn, plotly, bokeh

1. Working with Numpy arrays
2. Working with Pandas data frames
3. Basic plots using Matplotlib
4. Frequency distributions, Averages, Variability
5. Normal curves, Correlation and scatter plots, Correlation coefficient
6. Regression
7. Z-test
8. T-test
9. ANOVA
10. Building and validating linear models
11. Building and validating logistic models
12. Time series analysis

TOTAL: 45 PERIODS

OUTCOMES:

At the end of the course, the student should be able to:

- Write python programs to handle data using Numpy and Pandas
- Perform descriptive analytics
- Perform data exploration using Matplotlib
- Perform inferential data analytics
- Build models of predictive analytics

SOFTWARE REQUIREMENTS:

Operating Systems: Windows 7 or higher

Software: Python, Numpy, Scipy, Matplotlib, Pandas, statmodels, seaborn, plotly, bokeh

CO - PO - PSO MAPPING:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3	PSO4
AD3467.1	2	-	-	3	-	-	-	-	-	-	-	-	3	2	-	-
AD3467.2	-	2	-	-	2	-	-	-	1	-	3	-	3	2	-	1
AD3467.3	2	-	-	2	-	-	-	-	3	-	-	2	2	3	-	-
AD3467.4	-	3	-	-	2	-	-	-	-	3	-	-	2	1	-	3
AD3467.5	3	-	1	-	2	-	-	-	1	-	-	3	2	2	-	-
AD3467	2.3	2.5	1.0	2.5	2.0	-	-	-	1.7	3.0	3.0	2.5	2.4	2.0	-	2.0

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. To afford the necessary background in the field of Artificial Intelligence and data Science to deal with engineering problems to excel as engineering professionals in industries.
2. To improve the qualities like creativity, leadership, teamwork and skill thus contributing towards the growth and development of society.
3. To develop ability among students towards innovation and entrepreneurship that caters to the needs of Industry and society.
4. To inculcate and attitude for life-long learning process through the use of Artificial Intelligence and Data Science sources.
5. To prepare then to be innovative and ethical leaders, both in their chosen profession and in other activities.

PROGRAMME OUTCOMES (POs)

After going through the four years of study, Bachelor of Technology in Artificial Intelligence and Data Science Graduates will exhibit ability to:

PO#	Graduate Attribute	Programme Outcomes
1	Engineering knowledge	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.

2	Problem analysis	Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3	Design/development of solutions	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4	Conduct investigations of complex problems	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions
5	Modern tool usage	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modelling 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 the 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 (PSOs)

After the completion of Bachelor of Technology in Artificial Intelligence and Data Science programme the student will have following Program specific outcomes

1. Design and develop secured database applications with data analytical approaches of data preprocessing, optimization, visualization techniques and maintenance using state of the art methodologies based on ethical values.
2. Design and develop intelligent systems using computational principles, methods and systems for extracting knowledge from data to solve real time problems using advanced technologies and tools.
3. Design, plan and setting up the network that is helpful for contemporary business environments using latest software and hardware.
4. Planning and defining test activities by preparing test cases that can predict and correct errors ensuring a socially transformed product catering all technological needs.

COURSE OUTCOMES:

Course Name: AD3467 - Data Science and Analytics Laboratory

Year of study: 2024 – 2025 (EVEN SEM)

AD3467.1	Write python programs to handle data using Numpy and Pandas
AD3467.2	Perform descriptive analytics
AD3467.3	Perform data exploration using Matplotlib
AD3467.4	Perform inferential data analytics
AD3467.5	Build models of predictive analytics

CO- PO MATRIX:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
AD3467.1	2	-	-	3	-	-	-	-	-	-	-	-
AD3467.2	-	2	-	-	2	-	-	-	1	-	3	-
AD3467.3	2	-	-	2	-	-	-	-	3	-	-	2
AD3467.4	-	3	-	-	2	-	-	-	-	3	-	-
AD3467.5	3	-	1	-	2	-	-	-	1	-	-	3

JUSTIFICATION:

Course Outcome	Program Outcome	Value	Justification
AD3467.1	**PO1**	**2**	Students develop a solid foundation of engineering knowledge.
	PO4	**3**	Students are prepared with the necessary theoretical and applied knowledge.
AD3467.2	**PO2**	**2**	Students will be enhanced with ability to identify, analyze, and solve complex engineering problems.
	PO5	**2**	Students are encouraged to analyze problems from various perspectives.
	PO9	**1**	Students are expected to collaborate on engineering project.
	PO11	**3**	Students can manage engineering projects efficiently.
AD3467.3	**PO1**	**2**	Students will be able to apply theoretical concepts learned in lectures to practical scenarios in the laboratory.
	PO4	**2**	Improves data analytics skill to solve projects efficiently
	PO9	**3**	Promoting teamwork and interpersonal skills necessary for success in the professional world.
	PO12	**2**	Students are motivated to stay updated with new developments in the field of engineering.
AD3467.4	**PO2**	**3**	Students will be able to apply significant contributions in understanding fundamental concepts and applying them to practical engineering problems
	PO5	**2**	Familiarize students with contemporary tools and techniques used in engineering practices.
	PO10	**3**	Students can express complex ideas in both written and oral forms, crucial for professional success.
AD3467.5	**PO1**	**3**	Students will be able to apply significant contributions to critical areas such as engineering knowledge
	PO3	**1**	Develop the ability to analyze problems.
	PO5	**2**	Suggesting a focus on tools in one part of the course.
	PO9	**1**	Students will be exposed to sustainable engineering practices and environmental considerations.
	PO12	**3**	Students recognize the importance of lifelong learning by fostering a culture of self-improvement and knowledge acquisition.

CO -PO AVERAGE:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
AD3467	2.3	2.5	1.0	2.5	2.0	-	-	-	1.7	3.0	3.0	2.5

CO - PSO MATRIX:

CO	PSO1	PSO2	PSO3	PSO4
AD3467.1	3	2	-	-
AD3467.2	3	2	-	1
AD3467.3	2	3	-	-
AD3467.4	2	1	-	3
AD3467.5	2	2	-	-

JUSTIFICATION:

Course Outcome	PSO	Value	Justification
AD3467.1	**PSO1**	**3**	Provide students with the ability to collect, clean, and preprocess data, which is the foundation of any data science project.
	PSO2	**2**	Emphasizes the ability to apply statistical methods and machine learning algorithms to solve real-world problems.
AD3467.2	**PSO1**	**3**	Students will be well - prepared in preparation of datasets before any analysis can take place.
	PSO2	**2**	Creating predictive models, classification tasks, regression analysis, etc.
	PSO4	**1**	Applying new methodologies, and potentially contributing to new research areas in the field.
AD3467.3	**PSO1**	**2**	students learn how to handle raw data, deal with missing values
	PSO2	**3**	Improve students skill in model selection, evaluation, and optimization.
AD3467.4	**PSO1**	**2**	Students will learn how to Normalize and standardize data, and prepare data for further analysis.
	PSO2	**1**	Improves students critical thinking through various techniques.
	PSO4	**3**	students will have opportunities to engage in research-oriented tasks, possibly through innovative applications of data science and analytics techniques.

AD3467.5	**PSO1**	**2**	Involves the application of innovative techniques in data science, including experimental design,
	PSO2	**2**	Exhibit experimental design and application of advanced techniques

CO - PSO AVERAGE:

CO	**PSO1**	**PSO2**	**PSO3**	**PSO4**
AD3467	2.4	2.0	-	2.0

EVALUATION PROCEDURE FOR EACH EXPERIMENT

S.No	Description	Mark
1.	Aim & Pre-Lab discussion	20
2.	Observation	20
3.	Conduction and Execution	30
4.	Output & Result	10
5.	Viva	20
Total		**100**

INTERNAL ASSESSMENT FOR LABORATORY

S.No	Description	Mark
1.	Conduction & Execution of Experiment	30
2.	Record	10
3.	Model Test	20
Total		**60**

TABLE OF CONTENTS		
EX.NO.	**NAME OF THE EXPERIMENT**	**PAGE NO.**

EX NO :1 **WORKING WITH NUMPY ARRAYS**

AIM:

To Implement NumPy Operations with Arrays using python code.

ALGORITHM:

Step 1: START

Step 2: import the numpy Library.

Step 3: Initialize the numpy arrays to two different variables.

Step 4: Perform the numpy operations on the two arrays.

Step 5: Repeat the Step 4 until the user decides.

Setp 6: STOP

PROGRAM/SOURCE CODE:

 # NUMPY OPERATIONS WITH ARRAYS

```
import numpy as np
a=np.array([[1,2],[4,5]])
b=np.array([[1,2],[4,5]])
while True:
    print("1.Add , 2.Subtract , 3.Multiply , 4.Divide , 5.Dot product , 6.Exponentiation ,7.Logarithm , 8.Natural logarithm , 9.Exit")

    n=int(input("Enter the option number : ")) if not
    n<1 and not n>8:
        if n==1:
            c=np.add(a,b)
            print("Sum\n",c)
            print("\n") elif
        n==2:
            d=np.subtract(a,b)
            print("Difference\n",d)
            print("\n")
        elif n==3: e=np.multiply(a,b)
            print("Product\n",e)
            print("\n")
        elif n==4: f=np.divide(a,b)
            print("Remainder\n",f)
            print("\n")
```

```
    elif            n==5:
        g=np.dot(a,b)
        print("Dot       product\n",g)
        print("\n")
    elif                        n==6:
        h,i=np.exp(a),np.exp(b)
        print("Exponentiation for array a : \n",h)
        print("Exponentiation for array b : \n",i)
        print("\n")
    elif                        n==7:
        l,m=np.log(a),np.log(b)
        print("Logarithm for array a : \n",l)
        print("Logarithm for array b : \n",m)
        print("\n")
    elif n==8: x,y=np.log10(a),np.log10(b)
        print("Natural logarithm for array a : \n",x)
        print("Natural logarithm for array b : \n",y)
        print("\n")
elif n==9:
    break
else:
    print("No such option exist,please enter existing options.\n")
```

OUTPUT:

```
1.Add , 2.Subtract , 3.Multiply , 4.Divide , 5.Dot product , 6.Exponentiation , 7.Logarithm ,
8.Natural logarithm , 9.Exit
Enter the option number : 1
Sum
 [[ 2 4]
 [ 8 10]]

1.Add , 2.Subtract , 3.Multiply , 4.Divide , 5.Dot product , 6.Exponentiation , 7.Logarithm ,
8.Natural logarithm , 9.Exit
Enter the option number : 8
Natural logarithm for array a :
 [[0.          0.30103     ]
 [0.60205999 0.69897         ]]
Natural logarithm for array b :
 [[0.          0.30103     ]
 [0.60205999 0.69897         ]]

1.Add , 2.Subtract , 3.Multiply , 4.Divide , 5.Dot product , 6.Exponentiation , 7.Logarithm ,
8.Natural logarithm , 9.Exit
Enter the option number : 3
Product
 [[ 1  4]
 [16 25]]

1.Add , 2.Subtract , 3.Multiply , 4.Divide , 5.Dot product , 6.Exponentiation , 7.Logarithm ,
8.Natural logarithm , 9.Exit
Enter the option number : 5
Dot product
 [[ 9 12]
 [24 33]]

1.Add , 2.Subtract , 3.Multiply , 4.Divide , 5.Dot product , 6.Exponentiation , 7.Logarithm ,
8.Natural logarithm , 9.Exit
Enter the option number : 9
>>>
```

RESULT:

Thus, the program to Implement NumPy Operations with Arrays using Python code has been executed successfully.

EX NO :2 WORKING WITH PANDAS DATA FRAMES

AIM:

To Implement working with Pandas data frame using python code.

ALGORITHM:

Step 1: Start the program.

Step 3: Import pandas with an aliased name as pd.

Step 4: Create a given list and assign it to variable data.

Step 5: Call the data Frame function (data), and assign it to variable t.

Step 6: Call the Print function to print the Pandas data frame(t).

Step 7: Stop the program.

PROGRAM/SOURCE CODE:

```
import pandas as pd data={"Name":["Ram","Subash","Raghul","Arun","Deepak"],"Age":[24,25,24,
26,25],"CGPA":[9.5,9.3,9.0,8.5,8.8]}
t=pd.DataFrame(data) t.index+=1
print(t)
```

OUTPUT:

	Name	Age	CGPA
1	Ram	24	9.5
2	Subash	25	9.3
3	Raghul	24	9.0
4	Arun	26	8.5
5	Deepak	25	8.8

RESULT:

Thus, the program to Implement working with Pandas data frame using Python code has been executed successfully.

EX NO :3 **BASIC PLOTS USING MATPLOTLIB**

3(a) PLOTTING THE POINTS USING MATPLOTLIB

AIM:

To Implement Plotting the points for line graph using Matplotlib using python code.

ALGORITHM:

Step 1: Store in x1=[1,4,6,8]

Step 2: Store in y1=[2,5,8,9]

Step 3:Using plot() We can set the label as lineA and color of the line as red with x1 and y1 as co-ordinates.

Step 4: Store in x2=[3,6,8,10]

Step 5: Store in y2=[2,4,8,9]

Step 6:Using plot() We can set the label as line B and color of the line as green with x2 and y2 as co-ordinates.

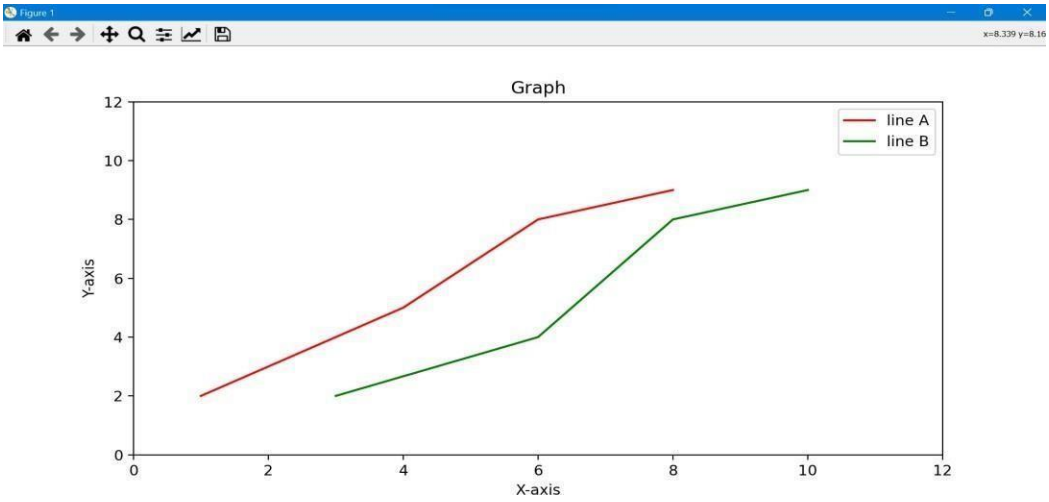
Step 7: Using xlim() and ylim() we can set the points as 0 to 12 on x-axis and y- axis .

Step 8: show the x-axis and y-axis of the plot, show the title as Graph.

PROGRAM/SOURCE CODE:

```
import matplotlib.pyplot as mpl x1=[1,4,6,8]
y1=[2,5,8,9]
mpl.plot(x1,y1,label="line A",color="r") x2=[3,6,8,10]
y2=[2,4,8,9]
mpl.plot(x2,y2,label="line B",color="g") mpl.xlim(0,12)
mpl.ylim(0,12) mpl.xlabel("X-axis")
mpl.ylabel("Y-axis") mpl.title("Graph")
mpl.legend() mpl.show()
```

OUTPUT:



RESULT:

Thus, the program to Implement using Plotting the points using Matplotlib Python code has been executed successfully.

3(b) CREATE A BAR CHART USING MATPLOTLIB

AIM:

To Implement a bar chart using Matplotlib using python code.

ALGORITHM:

Step 1: Store in x=[1,2,3,4,5]

Step 2: Store in y=[50,65,85,87,98]

Step 3: Store in text=["IBM","Amazon","Facebook","Microsoft","Google"] Step4: Store in colors=["red","orange","yellow","blue","green"]

Step 5: Using xlim() and ylim() we can set the points as 0 to 6 on x-axis and 0 to 100 on y- axis respectively.

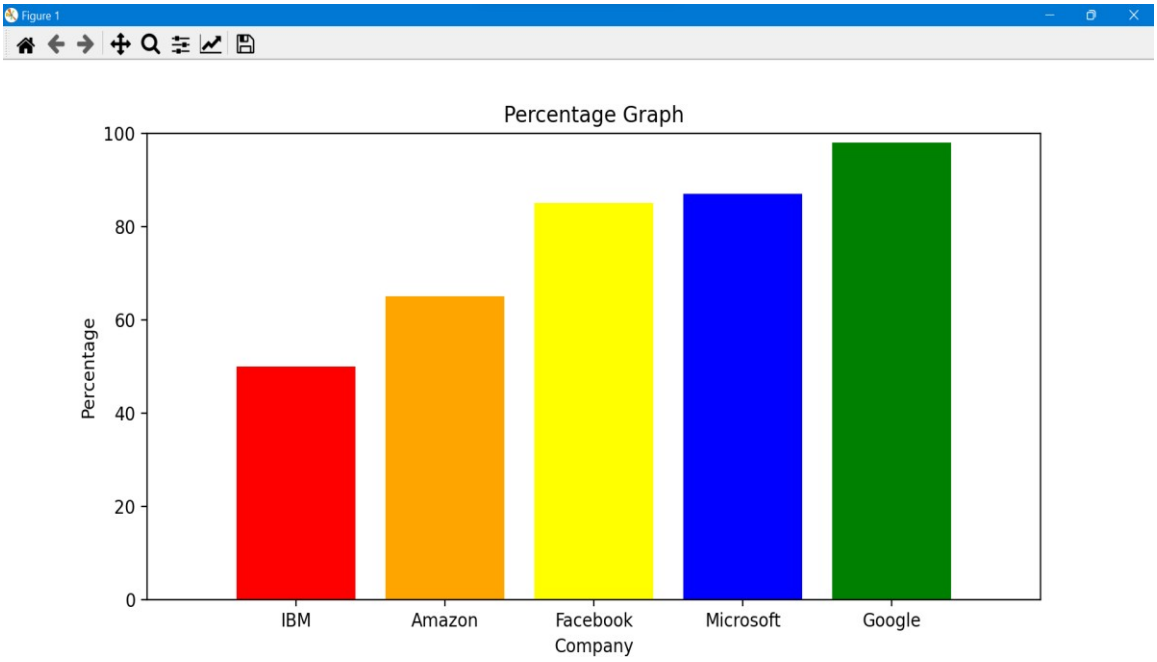
Step 6: Using bar() we can create a bar graph with x,y with label as text and color=colors and line width of the graph as 0.5.

Step 6: show the x-axis and y-axis of the plot as ‘Company' and 'Percentage', show the title as Percentage Graph.

PROGRAM/SOURCE CODE:

```
import matplotlib.pyplot as mpl x=[1,2,3,4,5]
y=[50,65,85,87,98]
text=["IBM","Amazon","Facebook","Microsoft","Google"]
colors=["red","orange","yellow","blue","green"] mpl.xlim(0,6) mpl.ylim(0,100)
mpl.bar(x,y,tick_label=text,color=colors,linewidth=0.5) mpl.xlabel("Company")
mpl.ylabel("Percentage")
mpl.title("Percentage Graph") mpl.show()
```

OUTPUT:



RESULT:

Thus, the program to Implement a bar chart using Matplotlib using Python code has been executed successfully.

EX NO :4 FREQUENCY DISTRIBUTIONS, AVERAGES, VARIABILITY

4a) FREQUENCY DISTRIBUTIONS

AIM:

To Implement Frequency Distributions using python code.

ALGORITHM:

Step 1: Start the program

Step 2: Import numpy with an aliased name as np

Step 3: Import pandas with an aliased name as pd

Step 4: Create a list(a) with elements for which average has to be found

Step 5: Sort the list using a.sort()

Step 6: Create required number of empty list (a1= [], a2= [], a3= []….)

Step 7: After the range is decided, get the elements which fall under specified range using for loop and store it in the empty list then combine all the list into a variable (data=[a1,a2,a3…])

Step 8: Define a function 'interval' which displays range and total when it is called.

Step 9: Define a function 'frequency' which display length of a1,a2,a3 along with total length of a1,a2,a3 which is stored in a variable

Step 10: Display the columns with label as interval and frequency using DataFrame

Step 11: Print the DataFrame

Step 12: Stop the program

PROGRAM/SOURCE CODE:

```
# FREQUENCY DISTRIBUTION
import pandas as pd
import numpy as np
def interval():
    iv=["1-3","4-6","7-9","Total"]
    return iv
def frequency():          # Frequency
    k=len(a1)+len(a2)+len(a3)
    f=[len(a1),len(a2),len(a3),k] return f
# Main Function
a=[2,6,5,3,6,7,9,2,1,4,2]
a.sort() a1=[]
a2=[]
a3=[]
for i in a:
    if i>=1 and i<=3: a1.append(i)
    elif i>=4 and i<=6: a2.append(i)
    elif i>=7 and i<=9: a3.append(i)
data=[a1,a2,a3] z=interval() f=frequency()
s=pd.DataFrame(zip(z,f),columns=["Interval","Frequency"])
s.index+=1
print(s)
```

OUTPUT:

	Interval	Frequency
1	1-3	5
2	4-6	4
3	7-9	2
4	Total	11

RESULT:

Thus, the program to Implement Frequency Distributions using Python code has been executed successfully.

4b) AVERAGES

AIM:

To Implement Averages using python code.

ALGORITHM:

Step 1: Start the program

Step 2: Import numpy with an aliased name as np

Step 3: Import pandas with an aliased name as pd

Step 4: Create a list(a) with elements for which average has to be found

Step 5: Sort the list using a.sort()

Step 6: Create required number of empty list (a1= [], a2= [], a3= []….)

Step 7: After the range is decided, get the elements which fall under specified range using for loop and store it in the empty list then combine all the list into a variable (data=[a1,a2,a3…])

Step 8: Define a function 'interval' which displays range and total when it is called.

Step 9: Define a function 'frequency' which display length of a1,a2,a3 along with total length of a1,a2,a3 which is stored in a variable

Step 10: Display the columns with label as interval and frequency using DataFrame

Step 11: Define a function 'average' which returns avg when it is called.

Step 11.a: avg=sum(a)\len(a)

Step 12: Print the DataFrame

Step 13: Call the average() funtion

Step 14: Stop the program

PROGRAM/SOURCE CODE:

```
# Average
import pandas as pd import numpy
as np def interval():
iv=["1-3","4-6","7-9","Total"]
return iv
def frequency():# Frequency k=len(a1)+len(a2)+len(a3)
f=[len(a1),len(a2),len(a3),k] return f
def average():# Average avg=sum(a)/len(a)
print("Average : ",avg)
# Main Function
a=[2,6,5,3,6,7,9,2,1,4,2]
a.sort() a1=[]
a2=[]
a3=[]
for i in a:
if i>=1 and i<=3: a1.append(i)
elif i>=4 and i<=6:
a2.append(i)
elif i>=7 and i<=9: a3.append(i)
data=[a1,a2,a3]
z=interval() f=frequency()
s=pd.DataFrame(zip(z,f),columns=["Interval","Frequency"]) s.index+=1 print(s)
average()
```

OUTPUT:

	Interval	Frequency
1	1-3	5
2	4-6	4
3	7-9	2
4	Total	11

Average: 4.2727272727272725

RESULT:

Thus, the program to Implement Averages using Python code has been executed successfully.

4(c) VARIABILITY

AIM:

To Implement Variability using python code.

ALGORITHM:

Step 1: Start the program

Step 2: Import numpy with an aliased name as np

Step 3: Import pandas with an aliased name as pd

Step 4: Create a list(a) with elements for which average has to be found

Step 5: Sort the list using a.sort()

Step 6: Create required number of empty list (a1= [], a2= [], a3= []….)

Step 7: After the range is decided, get the elements which fall under specified range using for loop and store it in the empty list then combine all the list into a variable (data=[a1,a2,a3…]) **Step 8:** Define a function 'interval' which displays range and total when it is called.

Step 9: Define a function 'frequency' which display length of a1,a2,a3 along with total length of a1,a2,a3 which is stored in a variable

Step 10: Display the columns with label as interval and frequency using DataFrame

Step 11: Print the DataFrame

Step 12: Call the function variance()

Step 13: print the variance

Step 14: Call the function variability()

Step 15: Stop the program

PROGRAM/SOURCE CODE:

 # Variability

```
import pandas as pd import numpy as np
import math
def interval():
iv=["1-3","4-6","7-9","Total"]
return iv
def frequency():        # Frequency
k=len(a1)+len(a2)+len(a3) f=[len(a1),len(a2),len(a3),k]
return f
def mean(): fr=frequency() mdn=median()
fr=np.array(fr[0:-1],dtype="float") mdn=np.array(mdn[0:-1],dtype="float")
fx=np.multiply(fr,mdn)
fx=list(fx) sm,frs=sum(fx),np.sum(fr)
fxm1=sm/frs
fxm=[fxm1 for i in range(len(data))] fxm.append("-")
return fxm
def median():
m=[]
for h in data:
t=(h[0]+h[-1])/2
m.append(t)

m.append("-")

return m
def variance():
x=median()
xb=mean()
fq=frequency()
x=np.array(x[0:-1],dtype="float") xb=np.array(xb[0:-
1],dtype="float") fq=np.array(fq) x_xb=np.subtract(x,xb)
sig=np.multiply(fq[0:-1],x_xb) sig=np.array(sig,dtype="float")
```

```
sig=np.sum(sig) s_f=fq[-1]
        v=sig/(s_f-1) var="%.7s"%(v)
        return var
        def variability(): vr=variance() vr=float(vr[1:])
        vby=math.sqrt(vr) print("Variability : ",vby)
        # Main
        a=[2,6,5,3,6,7,9,2,1,4,2]
        a.sort()

        a1=[]
        a2=[]
        a3=[]
        for i in a:
        if i>=1 and i<=3: a1.append(i)
        elif i>=4 and i<=6: a2.append(i)
        elif i>=7 and i<=9: a3.append(i)
        data=[a1,a2,a3] z=interval() f=frequency()
        s=pd.DataFrame(zip(z,f),columns=["Interval","Frequency"]) s.index+=1 print(s)
        v=variance()
        print("Variance : ",v) variability()
```

OUTPUT:

	Interval	Frequency
1	1-3	5
2	4-6	4
3	7-9	2
4	Total	11

Variance: 1.77635

Variability: 0.8811072579430952

RESULT:

Thus, the program to Implement Variability using Python code has been executed successfully.

EX. NO.5 NORMAL CURVES, CORRELATION AND SCATTER PLOTS, CORRELATION COEFFICIENT

5(a) NORMAL CURVES

AIM:

To Implement Normal Curves using python code.

ALGORITHM:

Step 1 : START

Step 2 : import the matplotlib package

step 3 : import numpy package to input array of numbers to plot

Step 4 : assign a variable to store n-d array of objects of angles between 0 and 2π by using the arange() function from the NumPy library

Step 5: assign a variable to store the sine values

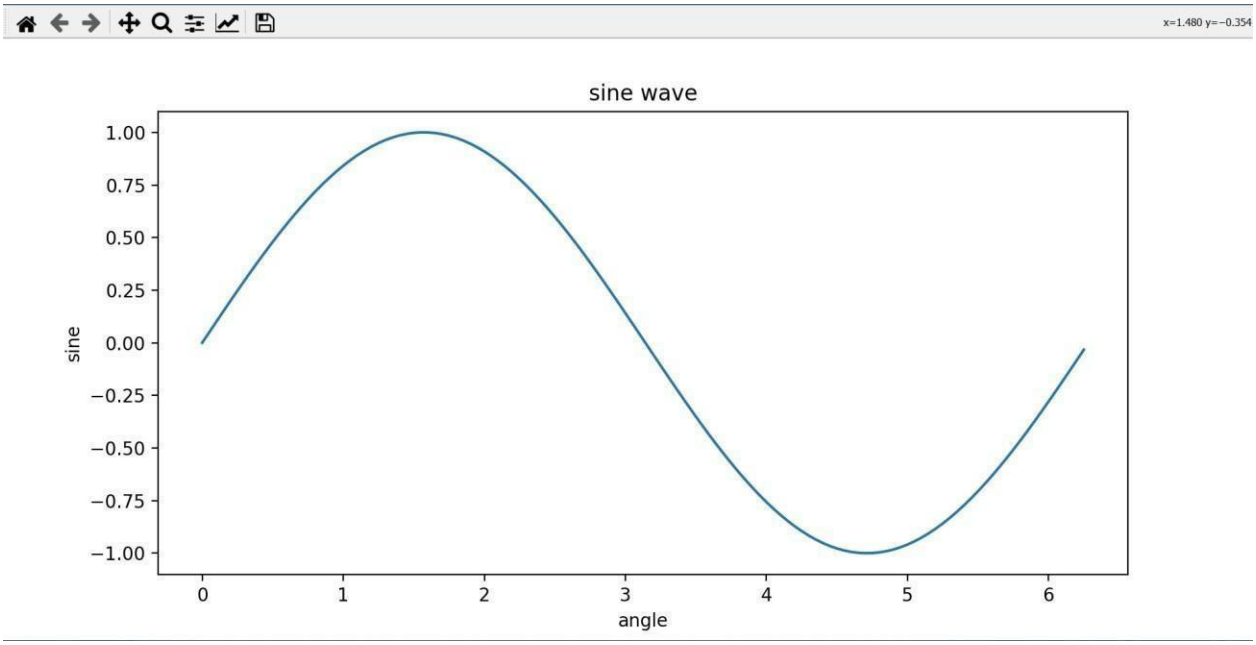
Step 6: Plot the values the values from two array variables Step 7: Print the plotted result

Step 8 : Stop

PROGRAM/SOURCE CODE:

```
#Normal Curve
from matplotlib import pyplot as plt import numpy as np
import math
x = np.arange(0, math.pi*2, 0.05)
y = np.sin(x)
plt.plot(x,y) plt.xlabel("angle") plt.ylabel("sine")
plt.title('sine wave') plt.show()
```

OUTPUT:



RESULT:

Thus, the program to Implement Normal Curves using Python code has been executed successfully.

5(b) CORRELATION AND SCATTER PLOTS

AIM:

To Implement Correlation and Scatter Plots using python code.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Import numpy with an aliased name np.

Step 3 : Import pyplot from matplotlib with an aliased name plt.

Step 4 : Call random.randn with parameter 50 and assign it to variable x.

Step 5 : Multiply 5 with x and add 3 to it and assign it to variable y1.

Step 6 : Multiply -5 with x and assign it to variable y2.

Step 7 : Call random.randn with parameter 50 and assign it to variable y3.

Step 8 : Scatter points x and y1 with color green and label as positive correlation.

Step 9 : Scatter points x and y2 with color red and label as negetive correlation.

Step 10 : Scatter points x and y3 with color blue and label as zero correlation.

Step 11 : Call the rcParams.update function with parameter fig size (10,8) and dpi 100.

Step 12 : Label the x axis as X-axis.

Step 13 : Label the y axis as Y-axis.

Step 14 : Give title as Correlation and Scatter plots.

Step 15 : Call the legend function.

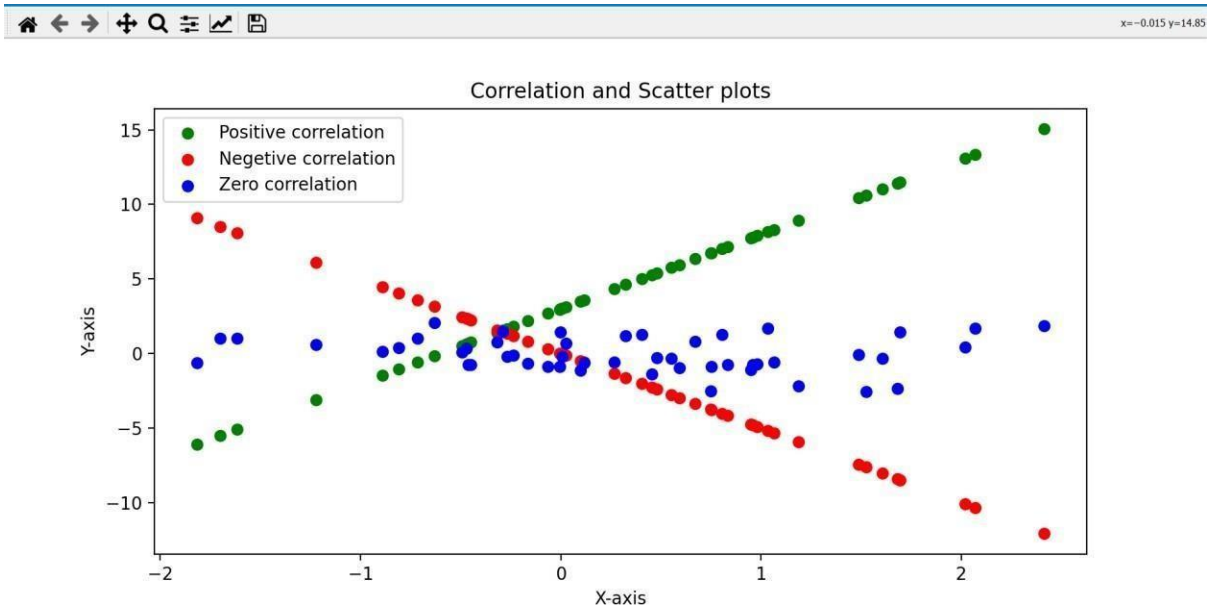
Step 16 : Call the show function.

Step 17 : End the program.

PROGRAM/SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as mpl x=np.random.randn(50)
 y1=x*5+3
y2=-5*x
y3=np.random.randn(50) mpl.scatter(x,y1,color="green",label="Positive correlation")
mpl.scatter(x,y2,color="red",label="Negetive correlation")
mpl.scatter(x,y3,color="blue",label="Zero correlation")
mpl.rcParams.update({'figure.figsize':(10,8),'figure.dpi':100}) mpl.xlabel("X-axis")
mpl.ylabel("Y-axis") mpl.title("Correlation and Scatter plots") mpl.legend()
mpl.show()
```

OUTPUT:



RESULT:

Thus, the program to Implement Correlation and Scatter plots using Python code has been executed successfully.

5(c) CORRELATION COEFFICIENT

AIM:

To Implement Correlation Coefficient using python code.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Import numpy with an aliased name np.

Step 3 : Import pyplot from matplotlib with an aliased name plt.

Step 4 : Call random.randn with parameter 50 and assign it to variable x.

Step 5 : Multiply 5 with x and add 3 to it and assign it to variable y1.

Step 6 : Multiply -5 with x and assign it to variable y2.

Step 7 : Call random.randn with parameter 50 and assign it to variable y3.

Step 8 : Scatter points x and y1 with color green and label as positive correlation and call the corrcoef with parameters (x,y1)[0,1],1.

Step 9 : Scatter points x and y2 with color red and label as negetive correlation and call the corrcoef with parameters (x,y2)[0,1],1.

Step 10 : Scatter points x and y3 with color blue and label as zero correlation and call the corrcoef with parameters (x,y3)[0,1],1.

Step 11 : Call the rcParams.update function with parameter fig size (10,8) and dpi 100.

Step 12 : Label the x axis as X-axis.

Step 13 : Label the y axis as Y-axis.

Step 14 : Give title as Correlation and Scatter plots.

Step 15 : Call the legend function.

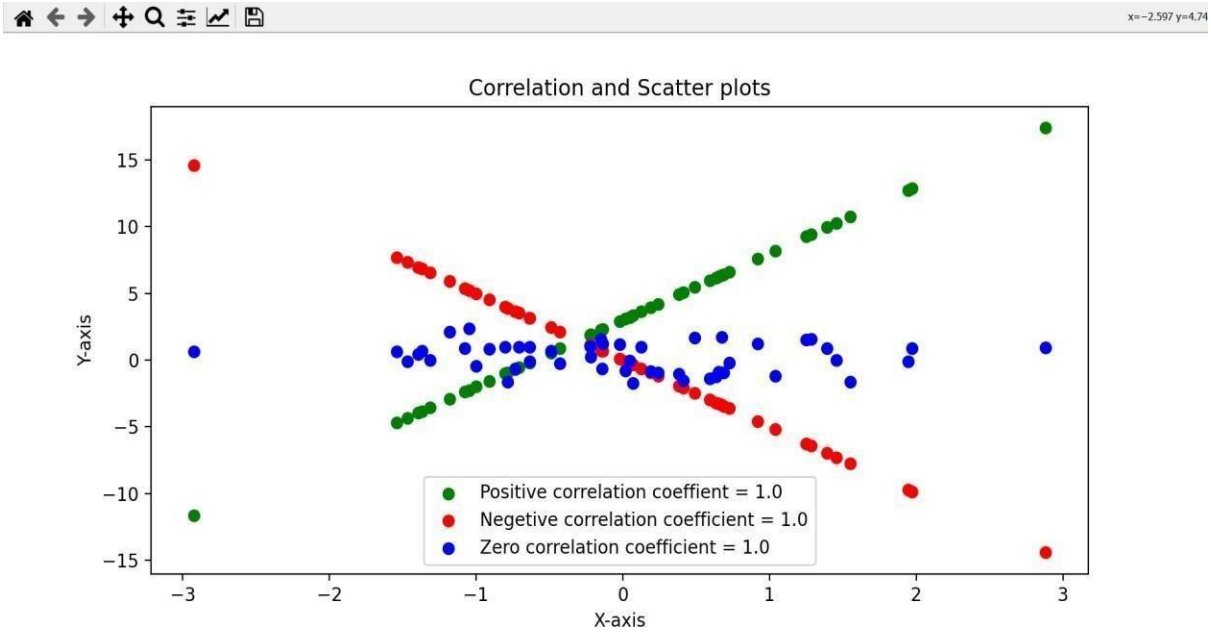
Step 16 : Call the show function.

Step 17 : End the program.

PROGRAM/SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as mpl
x=np.random.randn(50) y1=x*5+3

y2=-5*x
y3=np.random.randn(50)

mpl.scatter(x,y1,color="green",label=f"

Positive correlation coeffient ={np.round(np.corrcoef(x,y1)[0,1],1)}") mpl.scatter(x,y2,color="red",label=f"

Negetive correlation coefficient ={np.round(np.corrcoef(x,y1)[0,1],1)}")
mpl.scatter(x,y3,color="blue",label=f"

Zero correlation coefficient ={np.round(np.corrcoef(x,y1)[0,1],1)}")
mpl.rcParams.update({'figure.figsize':(10,8),'figure.dpi':100})

mpl.xlabel("X-axis")
mpl.ylabel("Y-axis")
mpl.title("Correlation and Scatter plots") mpl.legend()
mpl.show()
```

OUTPUT:



RESULT:

Thus, the program to Implement Correlation Coefficient using Python code has been executed successfully.

EX. NO. 6 REGRESSION

AIM:

To Implement Regression using Python code.

ALGORITHM:

Step 1 : Start the program.

Step 2 : Import numpy with an aliased name np.

Step 3 : Import pyplot from matplotlib with an aliased name plt.

Step 4 : Define a function linreg with parameters x and y.

Step 5 : Call the size function with parameter x and assign it to variable a.

Step 6 : Call the mean function with parameter x and assign it to variable mnx.

Step 7 : Call the mean function with parameter y and assign it to variable mny.

Step 8 : Call the sum function with parameter y*x and subtract a*mny*mnx and assign it to variable cd.

Step 9 : Call the sum function with parameter x*x and subtract a*mnx*mnx and assign it to variable dx.

Step 10 : Divide cd by dx and assign it to r1.

Step 11 : Subtract r1*mnx from mny and assign it to r0.

Step 12 : Print coefficients r0 and r1.

Step 13 : Scatter points x and y with color red and label as observation points.

Step 14 : Add r1*x to r0 and assign it to variable pred.

Step 15 : Plot x and pred with color green and label as regression line.

Step 16 : Label the x axis as X-axis.

Step 17 : Label the y axis as Y-axis.

Step 18 : Give title as Linear Regression.

Step 19 : Call the legend function.

Step 20 : Call the show function.

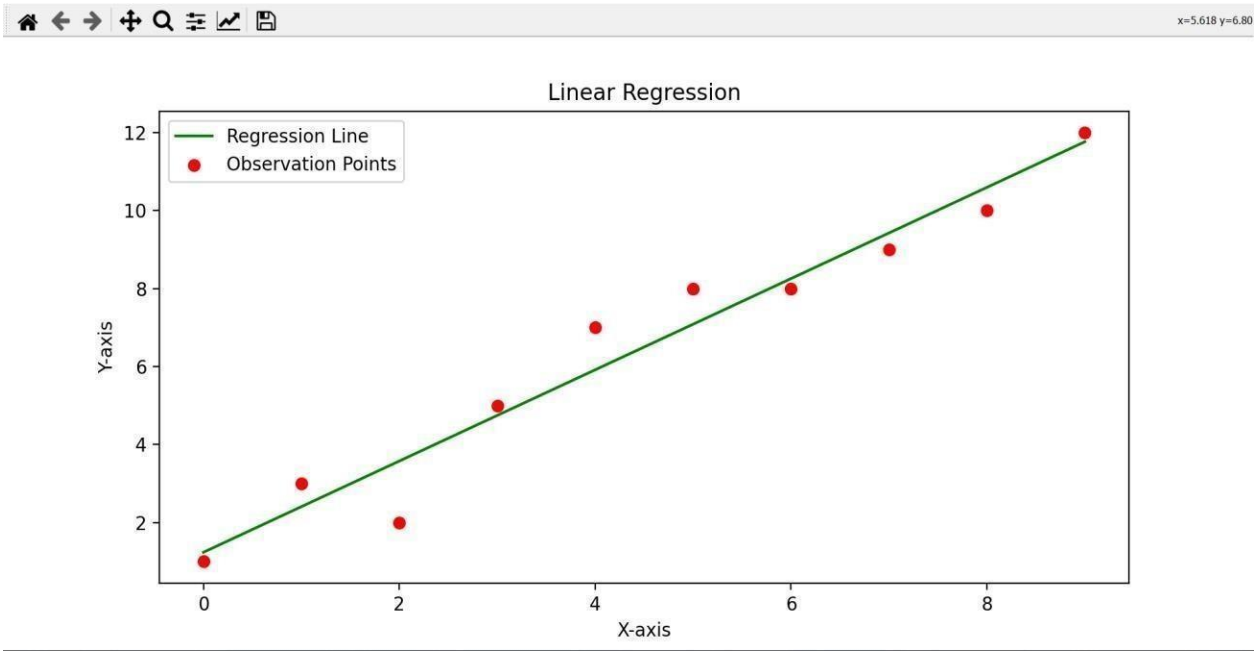
Step 21 : Call the linreg function with parameters x and y.

Step 22 : End the program.

PROGRAM/SOURCE CODE:

```
import numpy as np
import matplotlib.pyplot as mpl def linreg(x, y):
    a=np.size(x)
    mnx=np.mean(x) mny=np.mean(y) cd=np.sum(y*x)-
    a*mny*mnx dx=np.sum(x*x)-a*mnx*mnx r1=cd/dx
    r0=mny-r1*mnx
    print("Coefficients : \nr0 : ",r0,"\nr1 : ",r1)
    mpl.scatter(x,y,color="red",label="Observation Points") pred=r0+r1*x
    mpl.plot(x,pred,color="green",label="Regression Line") mpl.xlabel('X-axis')
    mpl.ylabel('Y-axis') mpl.title("Linear Regression") mpl.legend()
    mpl.show()
x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
linreg(x,y)
```

OUTPUT:



RESULT:

Thus, the program to Implement regression using Python code has been executed successfully.

EX.NO.7. Z-TEST

AIM:

To write a python program to perform z-test.

ALGORITHM:

1. Import the required packages
2. Set the number,alpha number values
3. Create the comparient data value
4. Calculate the perform of 2 test
5. Otherwise reject the NULL Hypothesis
6. End the program

PROGRAM:

```
import numpy as np import scipy.stats as stats

sample_mean = 110

population_mean = 100

population_std = 15

sample_size = 50

alpha = 0.05

z_score = (sample_mean- population _mean)/( population _std/np.sqrt(50))

 print('Z-Score :',z_score)

z_critical = stats.norm.ppf(1-alpha)

print(Critical Z-Score :',z_ critical)if z_score > z_ critical: print("Reject Null Hypothesis")

else:
print("Fail to Reject Null Hypothesis")

p_value = 1-stats.norm.cdf(z_score)

print('p-value :',p_value)

if p_value < alpha:

print("Reject Null Hypothesis")

else:

print("Fail to Reject Null Hypothesis")
```

OUTPUT:

Z-Score : 4.714045207910317

Critical Z-Score : 1.6448536269514722

Reject Null Hypothesis

p-value : 1.2142337364462463e-06

Reject Null Hypothesis

RESULT:

Thus the program was executed & verified successfully.

EX.NO.8 T-TEST

AIM:

To implement the T-test model

ALGORITHM:

1. Import the required packages
2. Create the two variable x & y
3. Assign the values to the 2 variable
4. Find the standardization of it
5. Print the result
6. End the result

PROGRAM:

```
Import numpy as np From scipy
import stats N=10
X=np.random.random(N)+2 Y= np.random.random(N)
Var_X=X.var(df=1) Var_X=Y.var(df=1)
SD=np.sqrt((var_X+var_Y)/2)
Print("Standard deviation=",SD)
Tval=(x.num()-y.num())/(SD*np.sqrt(2/n))
Df=2*n2
tval=1-stats.+.(df(+val.df=df) print("T="+str(+val2))

print("p="+str(pval2))
```

OUTPUT:

Standard deviation=0.764239858

T=4.87688162540 F=4.87688162540

P=0.00012126676

RESULT:

Thus the program was executed & verified successfully.

EX.NO.9 ANOVA

AIM:

To implement the program of Anova using the python program.

ALGORITHM:

1. start the program.
2. Set up null hypothesis
3. calculate test statistics using aov function
4. calculate the F-critical value
5. compare test statistics with F-critical value
6. stop the program

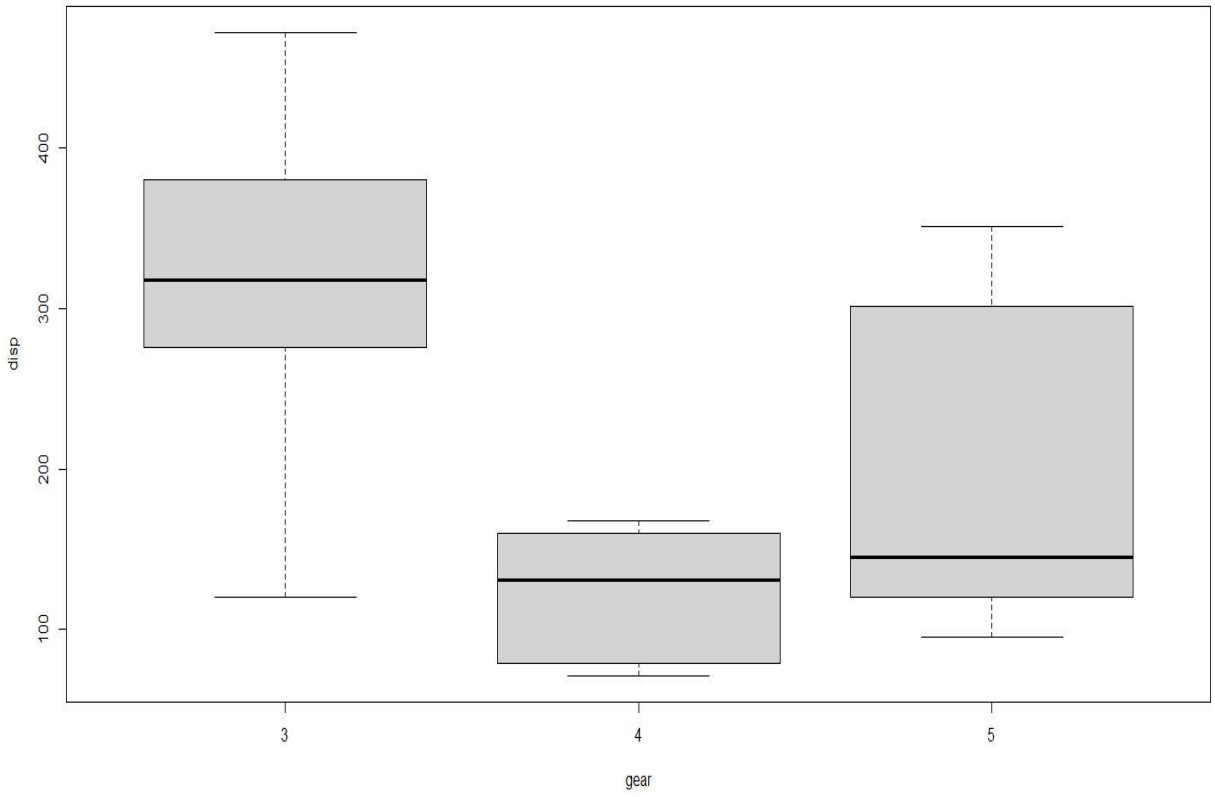
PROGRAM:

Install.packages("dplyr") Library(dplyr)

Boxplot(mtcars$disp~factor(mtcars$gear), Xlab="gear",ylab="disp")

mtcars_aov<-aov(mtcars$disp~factor(mtcars$gear)) summary(mtcars_aov)

OUTPUT:



RESULT:

Thus the program was executed & verified successfully.

EX.NO.10 BUILDING AND VALIDATING LINEAR MODELS

AIM:

To implement a program of building and validating linear model using a python program.

ALGORITHM:

1. Start the program
2. Import pandas as pd
3. Import numpy as np
4. Import seaborn as sns
5. Load the dataset boston
6. Set the style and 昀椀gure
7. Print the botson.key()
8. Stop the program

PROGRAM:

```
import pandas as pd import numpy as np
import matplotlib.pyplot as plt import seaborn as sns
    from sklearn.datasets
    import load_boston sns.set(style="kicks",color_codes=True) plt.rcParams['figure.bigsize'] = (8,5)

    plt.rcParams['figure.dpi'] = 150

    print(boston.keys())
    dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename']) print(boston.DESCR)
```

OUTPUT:

You will find these details in output: Attribute Information (in order):

— CRIM per capita crime rate by town

— ZN proportion of residential land zoned for lots over 25,000 sq.ft.

— INDUS proportion of non-retail business acres per town

— CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

— NOX nitric oxides concentration (parts per 10 million)

— RM average number of rooms per dwelling

— AGE proportion of owner

-occupied units built prior to 1940

— DIS weighted distances to live Boston employment centres

— RAD index of accessibility to radial highways

— TAX full-value property-tax rate per $10,000

— PTRATIO pupil-teacher ratio by town

— B 1000 (Bk — 0.63)² where Bk is the proportion of blacks by town

— LSTAT % lower status of the population

— MEDV Median value of owner

-occupied homes in $1000's :Missing Attribute Values: None

PROGRAM:

```
df=pd.DataFrame(boston.data,columns=botson.features_names)
df.head()Print(df.columns) Print(df.head())
```

OUTPUT:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

PROGRAM:

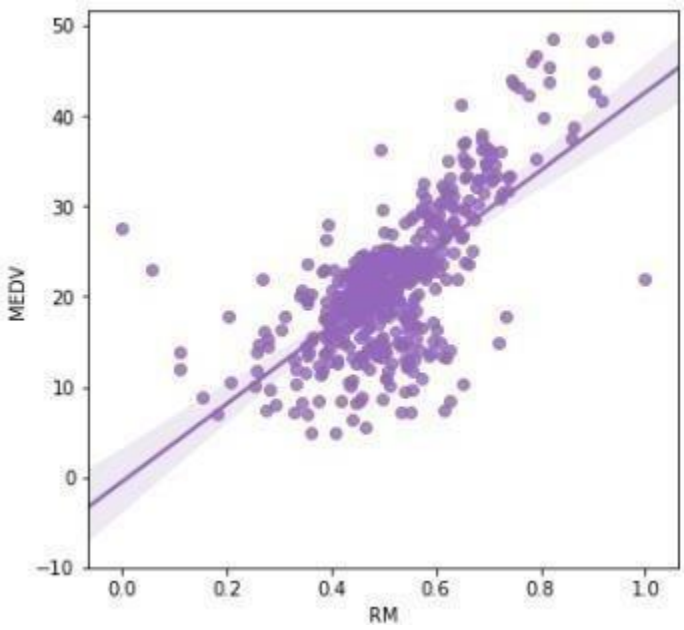
#Plotting heatmap for overall data setsns.heatmap(df.corr(),square=True,cmap='RdY1Gn')

OUTPUT:

PROGRAM:

Sns.Implot(x='RM',Y='MEDV',data=df)

OUTPUT:



RESULT:

Thus the program was executed & verified successfully.

EX.NO.11 BUILDING AND VALIDATING LOGISTICS MODELS

AIM:

To implement a program of building and validating logistics model using an python program.

ALGORITHM:

1. Start the program
2. Import the libraries
3. Load the training dataset
4. Insert the variable of xtrain and ytrain
5. Optimization terminated current value
6. Print the values
7. Stop the program

PROGRAM:

```
import statsmodels.api as sm import

pandas as pd

df = pd.read_csv('logit_train1.csv', index_col = 0)

Xtrain = df[['gmat', 'gpa', 'work_experience']]

ytrain = df[['admitted']]

    log_reg = sm.Logit(ytrain, Xtrain).att()
```

OUTPUT:

```
        Optimization terminated successfully.

Current function value: 0.352707 Iterations 8
```

PROGRAM:

```
print(log_reg.summary())
```

OUTPUT:

Logit Regression Results

==========

Dep. Variable:	admitted	No. Observations:	30
Model:	Logit	Df Residuals:	27
Method:	MLE	Df Model:	2
Date:	Wed, 15 Jul 2020	Pseudo R-squ.:	0.4912
Time:	16:09:17	Log-Likelihood:	-10.581
converged:	True	LL-Null:	-20.794
Covariance Type:	nonrobust	LLR p-value:	3.668e-05

==========

	coef	std err	z	P>|z|	[0.025	0.975]
gmat	-0.0262	0.011	-2.383	0.017	-0.048	-0.005
gpa	3.9422	1.964	2.007	0.045	0.092	7.792
work_experience	1.1983	0.482	2.487	0.013	0.254	2.143

==========

PROGRAM:

```
df = pd.read_csv('logit_test1.csv', index_col = 0)
Xtest = df[['gmat', 'gpa', 'work_experience']]
ytest = df['admitted']
yhat = log_reg.predict(Xtest)
prediction = list(map(round, yhat))
```

```
print('Actual values', list(ytest.values))
print('Predictions :', prediction)
```

OUTPUT:

```
Optimization terminated successfully.
Current function value: 0.352707
Iterations 8
Actual values [0, 0, 0, 0, 0, 1, 1, 0, 1, 1]
Predictions : [0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
```

PROGRAM:

```
from sklearn.metrics import (confusion_matrix,
accuracy_score) cm = confusion_matrix(ytest,
prediction)
print ("Confusion Matrix : \n", cm)
print('Test accuracy = ', accuracy_score(ytest, predic玲椀on))
```

OUTPUT:

```
Confusion Matrix : [[6 0]
[2 2]]
Test accuracy =  0.8
```

RESULT:

Thus the program was executed & verified successfully.

EX.NO.12 TIME SERIES ANALYSIS

AIM:

The aim of performing time series analysis is to model and forecast the behavior of a time series data over a period of time, using statistical methods, in order to identify patterns, trends, and seasonality in the data.

ALGORITHM:

The algorithm for performing time series analysis involves the following steps:

1. Collect data: Collect data on the time series variable over a period of time.
2. Visualize the data: Plot the time series data to identify patterns, trends, and seasonality.
3. Decompose the time series: Decompose the time series into its components, which are trend, seasonality, and residual variation. This can be done using techniques such as moving averages, exponential smoothing, or the Box-Jenkins method.
4. Model the trend: Model the trend component of the time series using techniques such as linear regression, exponential smoothing, or ARIMA models.
5. Model the seasonality: Model the seasonality component of the time series using techniques such as seasonal decomposition, dummy variables, or Fourier series.
6. Model the residual variation: Model the residual variation component of the time series using techniques such as autoregressive models, moving average models, or ARIMA models.
7. Choose the best model: Evaluate the fit of the different models using measures such as AIC, BIC, and RMSE, and choose the model that best fits the data.
8. Forecast future values: Use the chosen model to forecast future values of the time series variable.
9. Validate the model: Validate the model by comparing the forecasted values with actual values from a hold-out sample, or by using cross-validation techniques.
10. Refine the model: Refine the model by making adjustments to the model specification, such as adding or removing variables, transforming variables, or adding interaction terms.
11. Interpret the results: Interpret the results of the time series analysis in terms of the patterns, trends, and seasonality of the data, and use the forecasted values to make predictions and inform decision-making.

PROGRAM:

We are using Superstore sales data .

```
import warnings
import itertools
import numpy as np
import matplotlib.pyplot as plt
warnings.filterwarnings("ignore")
plt.style.use('fivethirtyeight')
import pandas as pd
import statsmodels.api as sm
import matplotlibmatplotlib.rcParams['axes.labelsize'] = 14
matplotlib.rcParams['xtick.labelsize'] = 12
matplotlib.rcParams['ytick.labelsize'] = 12 matplotlib.rcParams['text.color'] = 'k'
```

We start from time series analysis and forecasting for furniture sales.

```
df=pd.read_excel("Superstore.xls")
furniture = df.loc[df['Category'] == 'Furniture'] A good 4-
year furniture sales data.
furniture['Order Date'].min(), furniture['Order Date'].max() Timestamp('2014–
01–06 00:00:00'), Timestamp('2017–12–30
00:00:00')
```

DATA PREPROCESSING

This step includes removing columns we do not need, check missing values, aggregate sales by date and so on.

```
cols = ['Row ID', 'Order ID', 'Ship Date', 'Ship Mode', 'Customer ID',
'Customer Name', 'Segment', 'Country', 'City', 'State', 'Postal Code', 'Region',
'Product ID', 'Category', 'Sub-Category', 'Product Name', 'Quantity',
'Discount', 'Profit']

furniture.drop(cols,axis=1,inplace=True) furniture=furniture.sort_values('Order
Date')furniture.isnull().sum() furniture=furniture.groupby('OrderDate')
['Sales'].sum().reset_ index()
```

OUTPUT:

Figure 1

```
Order Date          0
Sales dtype: int64  0
```

INDEXING WITH TIME SERIES DATA

furniture=furniture.set_index('OrderDate') furniture.index

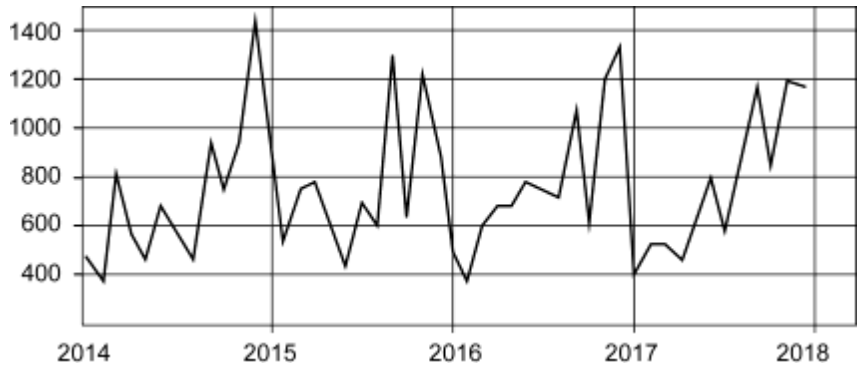
```
DatetimeIndex(['2014-01-06', '2014-01-07', '2014-01-10', '2014-01-11',
               '2014-01-13', '2014-01-14', '2014-01-16', '2014-01-19',
               '2014-01-20', '2014-01-21',
               ...
               '2017-12-18', '2017-12-19', '2017-12-21', '2017-12-22',
               '2017-12-23', '2017-12-24', '2017-12-25', '2017-12-28',
               '2017-12-29', '2017-12-30'],
              dtype='datetime64[ns]', name='Order Date', length=889, freq=None)
```

```
Order Date
2017-01-01     397.602133
2017-02-01     528.179800
2017-03-01     544.672240
2017-04-01     453.297905
2017-05-01     678.302328
2017-06-01     826.460291
2017-07-01     562.524857
2017-08-01     857.881889
2017-09-01    1209.508583
2017-10-01     875.362728
2017-11-01    1277.817759
2017-12-01    1256.298672
Freq: MS, Name: Sales, dtype: float64
```

Figure 3

Visualizing Furniture Sales Time Series Data

y.plot (figsize=(15,6)) plt.show()



RESULT:

Thus the python program for performing time series analysis is executed Successfully.