

**DEPARTMENT OFARTIFICIAL INTELLIGENCE AND DATA SCIENCE FOR IV SEM / II YEAR**

**LABORATORY MANUAL**

**19ADE401 - FUNDAMENTAL OF DATA SCIENCE AND ANALYTICS LABORATORY**

**[R-2019]**

# Prepared by

# S.SANTHI PRIYA ,AP/AI&DS

**Name : Register No. : Semester : Department :**

**STAFF INCHARGE ACADEMIC CO-ORDINATOR HOD**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DEAN ACADEMICS** |  |  | **PRINCIPAL** | **PRINCIPAL** |
|  |  |  |  |  |

## 19GEE101 COMPUTER FUNDAMENTALS AND PYTHON PROGRAMMING L T P C

(Lab Embedded Theory Course) **3 0 2 4**

(Common to All Branches)

## OBJECTIVES

* + To understand the techniques and processes of data science
  + To apply descriptive data analytics
  + To visualize data for various applications
  + To understand inferential data analytics
  + To analysis and build predictive models from data
  + To Provide a practical knowledge about various data analytics.

## LIST OF EXPERIMENTS

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

**OUTCOMES TOTAL: 15 PERIODS**

* + Explain the data analytics pipeline
  + Describe and visualize data
  + Perform statistical inferences from data
  + Analyze the variance in the data
  + Build models for predictive analytics

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

**AIM:**

To write a python Program to implementWorking With Pandas Data Frames.

**ALGORITHM:**

**Step1**: Import necessary libraries: Pandas and NumPy.

**Step 2**: Generate random data:

* + Generate random arrays values\_1 and values\_2 using NumPy's random.randint() function.
  + Create an array of years from 2015 to 2019 using NumPy's arange() function.
  + Define groups as ['A', 'A', 'B', 'B', 'C'].
  + Create a DataFrame df using Pandas, containing columns for group, year, value\_1, and value\_2.

**Step 3**: Print the DataFrame df.

**Step 4**: Insert a new column named 'new\_col' filled with random values from a standard normal distribution at position 2 in the DataFrame.

**Step 5**: Print the updated DataFrame df.

**Step 6**: Calculate cumulative sum of 'value\_2' column grouped by 'group' and store it in a new column named 'cumsum\_2'.

**Step 7**: Print the DataFrame with the new column.

**Step 8**: Sample 3 random rows from the DataFrame and print them.

**Step 9**: Replace negative values in 'new\_col' with 0 and print the result.

**Step10**: Filter DataFrame rows where the 'year' column matches certain years ('2010', '2014', '2017') and print them.

**Step11**: Use iloc and loc to select specific rows and columns from the DataFrame and print the results.

**Step12**: Calculate the percentage change of 'value\_1' column and print the result.

**Step13**:Calculate ranks for 'value\_1' column and add a new column 'rank\_1' to the DataFrame.

**Step14**: Print the DataFrame with the new 'rank\_1' column.

**Step15**: Count the number of unique values in each column of the DataFrame and print the results.

**Step16**: Infer the data types of columns in the DataFrame and print them.

**Step17**: Create a large DataFrame with random data and calculate its memory usage in megabytes.

**Step18**: Print the memory usage of the large DataFrame.

**Step19**: Print the summary statistics of the DataFrame using the describe() function.

**Step 20**: Select columns of type 'int64' and print them.

**Step 21**: Select columns excluding type 'int64' and print them.

**Step 22**: Replace occurrences of 'A' with 'A\_1' in the DataFrame and print the result.

**PROGRAM**

import pandas as pd import numpy as np #1. Query

values\_1 = np.random.randint(10, size=5)

values\_2 = np.random.randint(10, size=5)

years = np.arange(2015,2020)

groups = ['A','A','B','B','C']

df = pd.DataFrame({'group':groups, 'year':years, 'value\_1':values\_1, 'value\_2':values\_2})

print(df) print("\n")

#Query with certain conditions print(df.query('value\_1 < value\_2')) print("\n")

#2. Insert

#new column

new\_col = np.random.randn(5) #insert the new column at position 2 df.insert(2, 'new\_col', new\_col) print(df)

print("\n")

#3. Cumsum

df['cumsum\_2'] = df[['value\_2','group']].groupby('group').cumsum() print(df)

print("\n")

#4. Sample

sample1 = df.sample(n=3) print(sample1)

print("\n")

#5. Where

print(df['new\_col'].where(df['new\_col'] > 0 , 0)) print("\n")

#6. Isin

years = ['2010','2014','2017']

print(df[df.year.isin(years)]) print("\n")

#7. Loc and iloc print(df.iloc[:3,:2]) print("\n")

print(df.loc[:2,['group','year']]) print("\n")

#8. Pct\_change

print(df.value\_1.pct\_change()) print("\n")

#9. Rank

df['rank\_1'] = df['value\_1'].rank() print(df)

print("\n") #10. Nunique

print(df.nunique()) print("\n")

#11. Infer\_objects print(df.infer\_objects().dtypes) print("\n")

#12. Memory\_usage

df\_large = pd.DataFrame({'A': np.random.randn(1000000),'B': np.random.randint(100, size=1000000)})

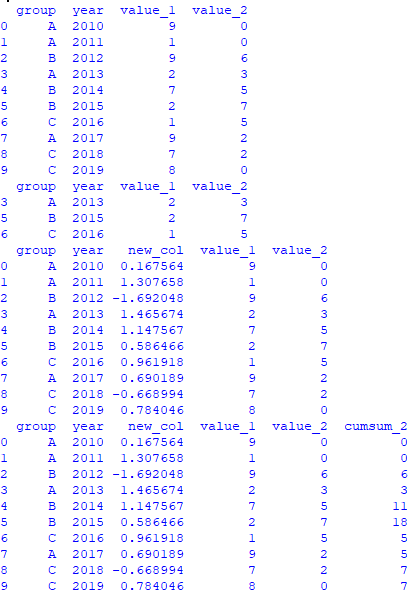
print(df\_large.memory\_usage().sum() / (1024\*\*2)) #converting to megabytes print("\n")

#13.Describe print(df.describe()) print("\n") #14.Select\_dtypes

print(df.select\_dtypes(include='int64')) print("\n") print(df.select\_dtypes(exclude='int64')) print("\n")

Replace print(df.replace('A', 'A\_1')) print("\n")

**SAMPLE OUTPUT:**



**RESULT:**

Thus the working with Numpy arrays was successfully completed.

**EX.NO. 2 WORKING WITH PANDAS DATA FRAMES**

#### Aim:

To work with Pandas data frames

### ALGORITHM

Step1:Start

Step2: import numpyand pandas module Step3:Createadataframeusingthedictionary Step4: Print the output

Step5:Stop

### PROGRAM

import numpy as np

import pandas as pd

data=np.array([['','Col1','Col2'], ['Row1',1,2],

['Row2',3,4]])

print(pd.DataFrame(data=data[1:,1:],

index = data[1:,0], columns=data[0,1:]))

#Takea2DarrayasinputtoyourDataFrame my\_2darray = np.array([[1, 2, 3], [4, 5, 6]]) print(pd.DataFrame(my\_2darray))

#TakeadictionaryasinputtoyourDataFrame my\_dict = {1: ['1', '3'], 2: ['1', '2'], 3: ['2', '4']}

print(pd.DataFrame(my\_dict))

#TakeaDataFrameasinputtoyourDataFrame

my\_df=pd.DataFrame(data=[4,5,6,7],index=range(0,4),columns=['A']) print(pd.DataFrame(my\_df))

#TakeaSeries as inputtoyourDataFrame

my\_series=pd.Series({"UnitedKingdom":"London","India":"NewDelhi","United States":"Washington", "Belgium":"Brussels"})

print(pd.DataFrame(my\_series))

df=pd.DataFrame(np.array([[1,2,3],[4,5,6]]))

#Usethe`shape`property print(df.shape)

Orusethe`len()`functionwiththe`index`property print(len(df.index))

#### Output:

Col1Col2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row1 | | | 1 | 2 |
| Row2 | | | 3 | 4 |
| 0 | 1 | 2 |  |  |
| 0 | 1 | 2 | 3 |  |
| 1 | 4 | 5 | 61 | 23 |
| 0 | 1 | 1 | 2 |  |
| 1 | 3 | 2 | 4A |  |
| 0 | 4 |  |  |  |
| 1 | 5 |  |  |  |
| 2 | 6 |  |  |  |
| 3 | 7 |  |  |  |
| 0 |  |  |  |  |

United Kingdom London India New Delhi United StatesWashington Belgium Brussels

(2, 3)

2

**RESULT**

**EX.NO. 3**  **BASIC PLOTS USING MAT PLOT LIB**

**AIM :**

.

To write a python Program to implement Basic Plots Using Matplotlib.

**ALGORITHM :**

**Step1** : Start the program.

**Step2 :** Divide matrices A and B in 4 sub-matrices of size N/2 x N/2 as shown in the below diagram.

**Step3 :** Calculate following values recursively. ae + bg, af + bh, ce + dg and cf + dh.

**Step 4:** Stop the program.

**PROGRAM**

1. import matplotlib.pyplot as plt

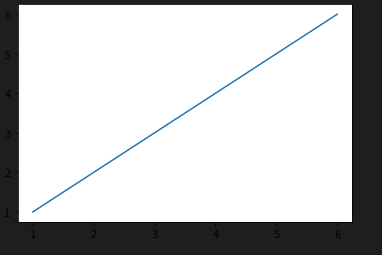
import numpy as np

xpoints =[1,2,3,4,5,6]

ypoints =[1,2,3,4,5,6]

plt.plot(xpoints, ypoints)

plt.show()

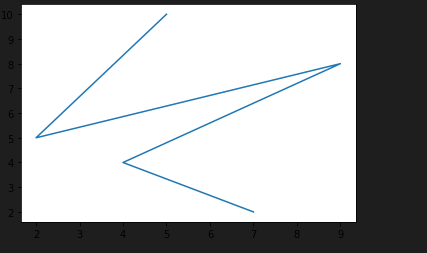


**b) x = [5, 2, 9, 4, 7]**

**y = [10, 5, 8, 4, 2]**

**plt.plot(x,y)**

**plt.show()**



**c) import matplotlib.pyplot as plt**

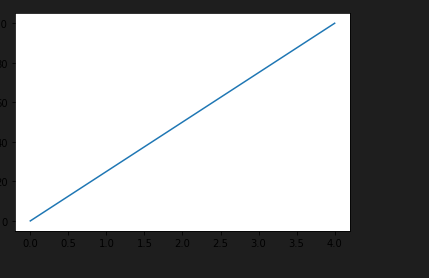
**import numpy as np**

**x = np.array([0, 4])**

**y = np.array([0, 100])**

**plt.plot(x, y)**

**plt.show()**

****

**d) import matplotlib.pyplot as plt**

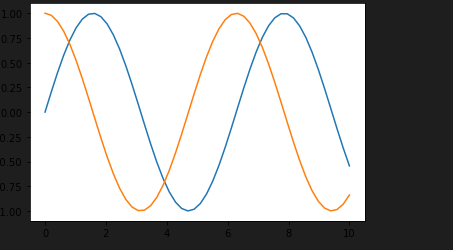
**import numpy as np**

**x = np.linspace(0, 10)**

**plt.plot(x, np.sin(x))**

**plt.plot(x, np.cos(x))**

**plt.show()**

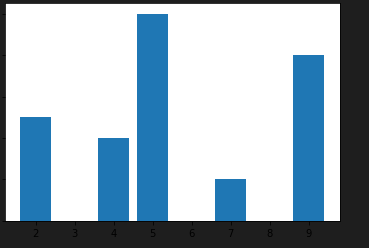
****

**e) x = [5, 2, 9, 4, 7]**

**y = [10, 5, 8, 4, 2]**

**plt.bar(x,y)**

**plt.show()**

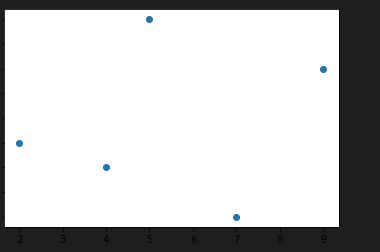
****

**f) x = [5, 2, 9, 4, 7]**

**y = [10, 5, 8, 4, 2]**

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

**plt.show()**

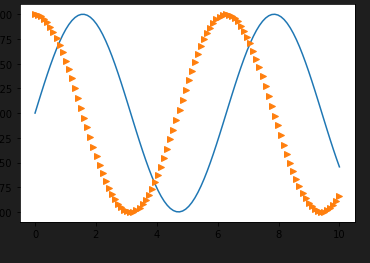
****

**g) x = np.linspace(0, 10, 100)**

**fig = plt.figure()**

**plt.plot(x, np.sin(x), '-')**

**plt.plot(x, np.cos(x), '>');**

****

**EX.NO.**  **4 FREQUENCY DISTRIBUTIONS, AVERAGES, VARIABILITY**

**AIM:**

To write a python Program to implement Frequency distributions, Averages, Variability.

**ALGORITHM :**

**Step1** : Start The Program

**Step 2 :** Import necessary libraries: NLTK (Natural Language Toolkit)

**Step 3 :** Tokenization: Tokenize the input text into individual words using NLTK's word\_tokenize function

**Step4 :** Stop The Program

**PROGRAM**

**A)**

from nltk.tokenize import word\_tokenize

from nltk.corpus import gutenberg

sample = gutenberg.raw("blake-poems.txt")

token = word\_tokenize(sample)

wlist = []

for i in range(50):

wlist.append(token[i])

wordfreq = [wlist.count(w) for w in wlist]

print("Pairs\n" + str(zip(token, wordfreq)))

**OUTPUT :**

[([', 1), (Poems', 1), (by', 1), (William', 1), (Blake', 1), (1789', 1), (]', 1), (SONGS', 2), (OF', 3), (INNOCENCE', 2), (AND', 1), (OF', 3), (EXPERIENCE', 1), (and', 1), (THE', 1), (BOOK', 1), (of', 2), (THEL', 1), (SONGS', 2), (OF', 3), (INNOCENCE', 2), (INTRODUCTION', 1), (Piping', 2), (down', 1), (the', 1), (valleys', 1), (wild', 1), (,', 3), (Piping', 2), (songs', 1), (of', 2), (pleasant', 1), (glee', 1), (,', 3), (On', 1), (a', 2), (cloud', 1), (I', 1), (saw', 1), (a', 2), (child', 1), (,', 3), (And', 1), (he', 1), (laughing', 1), (said', 1), (to', 1), (me', 1), (:', 1), (``', 1)]

**B)**  # Python code to demonstrate the working of

# variance() function of Statistics Module

 # Importing Statistics module

import statistics

 # Creating a sample of data

sample = [2.74, 1.23, 2.63, 2.22, 3, 1.98]

 # Prints variance of the sample set

 # Function will automatically calculate

# it's mean and set it as xbar

print("Variance of sample set is % s" %(statistics.variance(sample)))

**OUTPUT :**

Variance of sample set is 0.40924

**RESULT :**

**EX.NO. 5**  **NORMAL CURVES**

**AIM:**

To write a python Program to implement normal curves .

**ALGORITHM :**

**Step1** :Start the program

**Step 2** :Import the numpy and pandas and matplotlib.

**Step 3** :Declare the x value and read csv data for the website.

**Step 4** :Plot the point for mean and d

**Step 5** :Draw a graph for given plot point.

**Step 6** :Stop the program

**PROGRAM**

import numpy as np import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

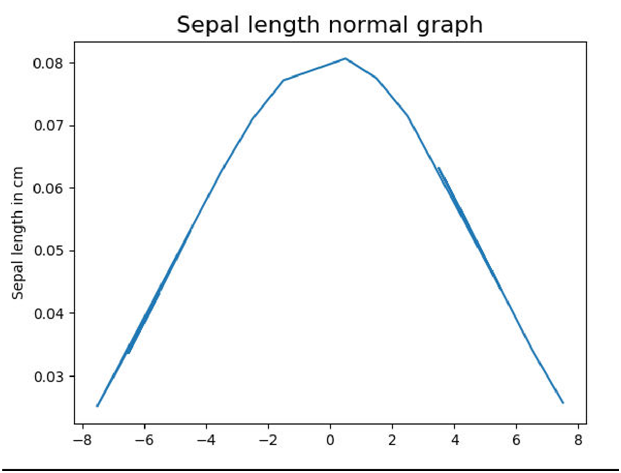
import statistics

from scipy.stats import norm x\_data=[]

iris\_data = pd.read\_csv("iris.csv") df=iris\_data.head(15) x\_data=list(df['sepallength']) mean=statistics.mean(x\_data) sd=statistics.stdev(x\_data) plt.plot(x\_data,norm.pdf(x\_data,mean,sd)) plt.title("Sepal length normal graph",fontsize=16) plt.ylabel('Sepal length in cm')

plt.show()

**0UTPUT**

****

**RESULT :**

**EX.NO. 6**

**REGRESSION**

**AIM:**

To write a python Program to implement Regression

**ALGORITHM**

**Step 1** :Start the program

**Step 2** :Importing Necessary Python Package

**Step 3** : Importing Dataset

**Step 4** :Organizing Data Into Training & Testing Sets

**Step 5** :Model Evaluation & Prediction

**Step 6** :Plot & Visualization

**Step 7** :Stop the program

**PROGRAM**

import numpy as np

import pandas as pd

# statsmodel is used to build statistical models

import statsmodels.api as sm

# To split the data into train and test data. The function is called train\_test\_split() from sklearn.model\_selection

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

# Load the dataset

cook = pd.read\_csv('Chapter 9 - Case Study Dataset - Cookingrange.csv')

# Understanding the dataset cook.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 400 entries, 0 to 399

Non-Null Count Dtype

Data columns (total 11 columns

**OUTPUT:**

**# Column**

0

Sales

400 **non-null**

int64

**1**

**CompPrice**

400 **non-null**

int64

**2**

**Income**

400 **non-null**

**int64**

**3**

**Promotion** 400 **non-null**

**int64**

**RESULT:**

The above program and output was executed successfully and verified.

**EX.NO.**  **7 Z-TEST**

**AIM**:

To write a python Program to implement Z-test.

**ALGORITHM :**

**Step 1 :** Start the program

**Step 2 :** First, identify the null and alternate hypotheses.

**Step 3 :** Determine the level of significance (∝).

**Step 4 :** Find the critical value of z in the z-test using

**Step 5 :** Calculate the z-test statistics. Below is the formula for calculating the z-test statistics.

**Step 6**  : Stop the program

**PROGRAM**

# imports import math

import numpy as np

from numpy.random import randn

from statsmodels.stats.weightstats import z test

 # Generate a random array of 50 numbers having mean 110 and sd 15

# similar to the IQ scores data we assume above

mean\_iq = 110

sd\_iq = 15/math.sqrt(50)

alpha =0.05

null\_mean =100

data = sd\_iq\*randn(50)+mean\_iq

# print mean and sd

print('mean=%.2f stdv=%.2f' % (np.mean(data), np.std(data)))

  # now we perform the test. In this function, we passed data, in the value parameter

# we passed mean value in the null hypothesis, in alternative hypothesis we check whether the

# mean is larger

  ztest\_Score, p\_value= ztest(data,value = null\_mean, alternative='larger')

# the function outputs a p\_value and z-score corresponding to that value, we compare the

# p-value with alpha, if it is greater than alpha then we do not null hypothesis

# else we reject it.

  if(p\_value <  alpha):

  print("Reject Null Hypothesis")

else:

  print("Fail to Reject NUll Hypothesis")

**OUTPUT:**

Reject Null Hypothesis

**RESULT:**

**EX.NO. 8**  **T-TEST**

**AIM:**

To write a python Program to implement T-test.

**ALGORITHM:**

**Step 1:** Start the program.

**Step 2:** State a hypothesis. A hypothesis is classified as a null hypothesis ( H0) and an alternative hypothesis (Ha) that rejects the null hypothesis. The null and alternate hypotheses are defined according to the type of test being performed.

**Step 3 :** Collect sample data.

**Step 4 :** Conduct the test.

**Step 5 :** Reject or fail to reject your null hypothesis H0.

**Step 6 :** Stop the program

**PROGRAM:**

a)  # Python program to display variance of data groups

 # Import library

import scipy.stats as stats

 # Creating data groups

data\_group1 = np.array([14, 15, 15, 16, 13, 8, 14,

                        17, 16, 14, 19, 20, 21, 15,

                        15, 16, 16, 13, 14, 12])

data\_group2 = np.array([15, 17, 14, 17, 14, 8, 12,

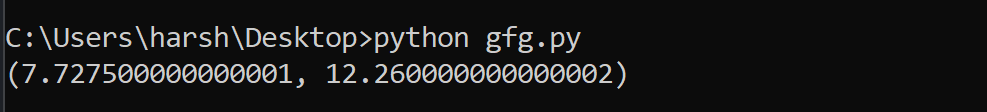
                        19, 19, 14, 17, 22, 24, 16,

                        13, 16, 13, 18, 15, 13])

 # Print the variance of both data groups

print(np.var(data\_group1), np.var(data\_group2))

**SAMPLE OUTPUT:**



b) # Python program to demonstrate how to

**# perform two sample T-test**

# Import the library

import scipy.stats as stats

 # Creating data groups

data\_group1 = np.array([14, 15, 15, 16, 13, 8, 14,

                        17, 16, 14, 19, 20, 21, 15,

                        15, 16, 16, 13, 14, 12])

data\_group2 = np.array([15, 17, 14, 17, 14, 8, 12,

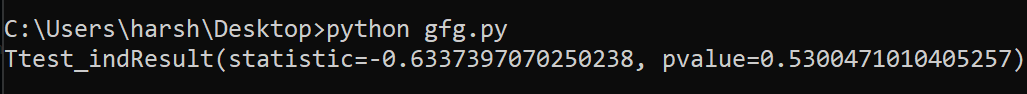
                        19, 19, 14, 17, 22, 24, 16,

                       13, 16, 13, 18, 15, 13])

# Perform the two sample t-test with equal variances

stats.ttest\_ind(a=data\_group1, b=data\_group2, equal\_var=True)

**0UTPUT**



**RESULT :**