EX.NO:1 INSTALLATION OF PACKAGES

# AIM:

To download, install and explore the features of NumPy, SciPy, Jupyter, Stasmodels and

Pandas packages.

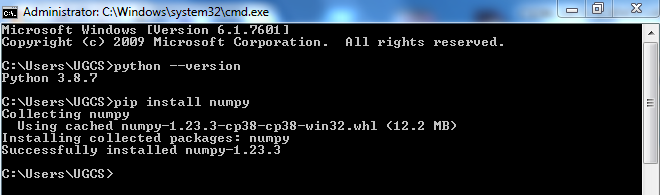
# INSTALLATION OF PACKAGES:

**PRE-REQUISITES:**

Operating System : Windows 7 Professional (Service Pack 1) Software : Python 3.8.7

# NUMPY:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.



# Features:

* High-performance N-dimensional array object.
* It contains tools for integrating code from C/C++ and FORTRAN.
* It contains a multidimensional container for generic data.
* Additional linear algebra, Fourier transforms, and random number capabilities.
* It consists of broadcasting functions.
* It had data type definition capability to work with varied databases.

# Sample Program:

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

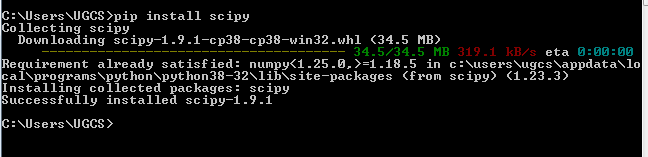
# Output:

[1 2 3]

# SCIPY:

[SciPy](https://docs.scipy.org/doc/scipy/reference/) is a python library that is useful in solving many mathematical equations and algorithms. It is designed on the top of Numpy library that gives more extension of finding scientific mathematical formulae like Matrix Rank, Inverse, polynomial equations, LU Decomposition, etc. Using its high level functions will significantly reduce the complexity of the code and helps in better analyzing the data. SciPy is an interactive Python session used as a data-

processing library that is made to compete with its rivalries such as MATLAB, Octave, R- Lab,etc. It has many user-friendly, efficient and easy-to-use functions that helps to solve problems like numerical integration, interpolation, optimization, linear algebra and statistics.



# Sample Program:

from scipy import constants print(constants.pi)

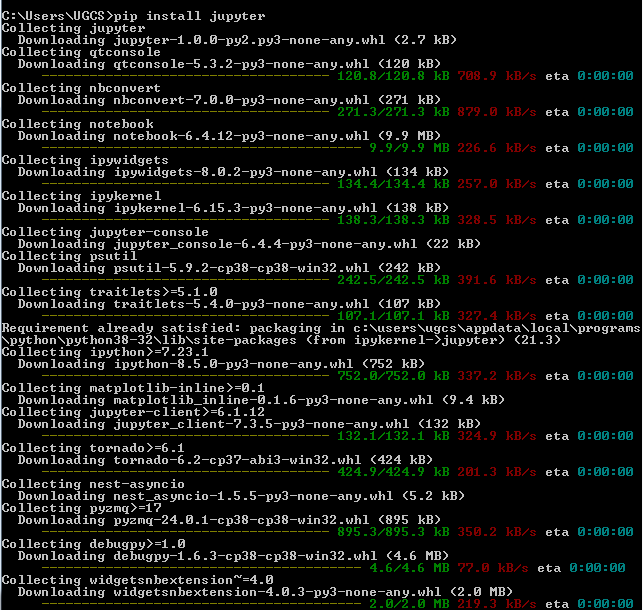
# Output:

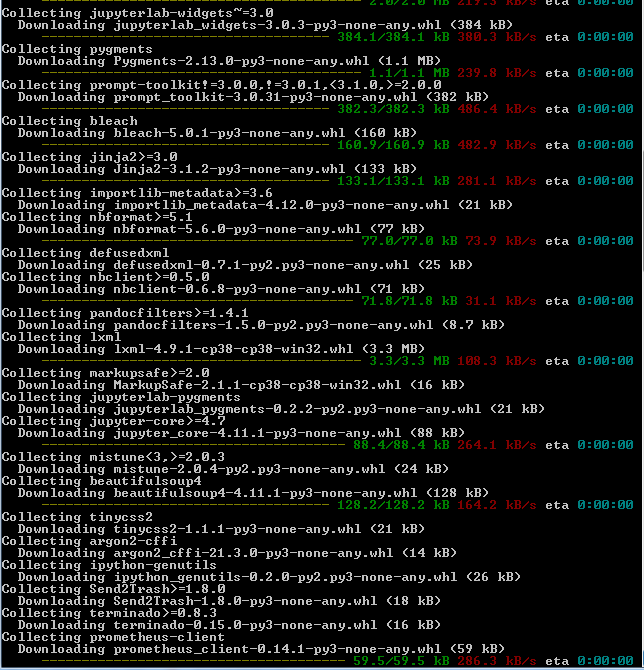
3.141592653589793

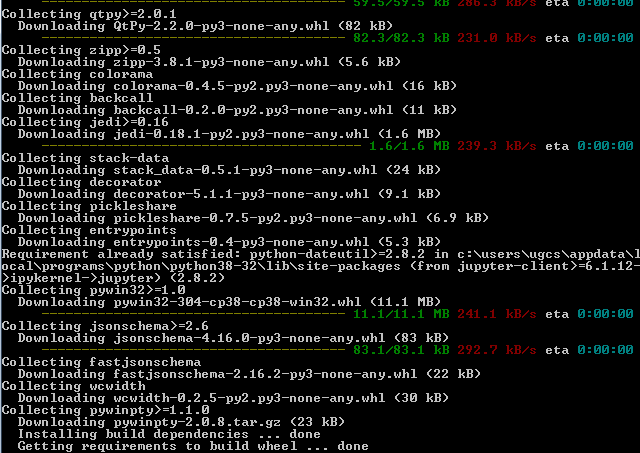
# JUPYTER:

The IPython Notebook concept was expanded upon to allow for additional programming languages and was therefore renamed "Jupyter". "Jupyter" is a loose acronym meaning **Julia, Python and R**, but today, the notebook technology supports many programming languages. An IDE normally consists of at least a source code editor, build automation tools and a debugger. **Jupyter Notebook is an IDE for Python** that allows its users to create documents containing both rich text and code. It also supports the programming languages Julia, and R.

Jupyter Notebook allows users to compile all aspects of a data project in one place making it easier to show the entire process of a project to your intended audience. Through the web-based application, users can create data visualizations and other components of a project to share with others via the platform.



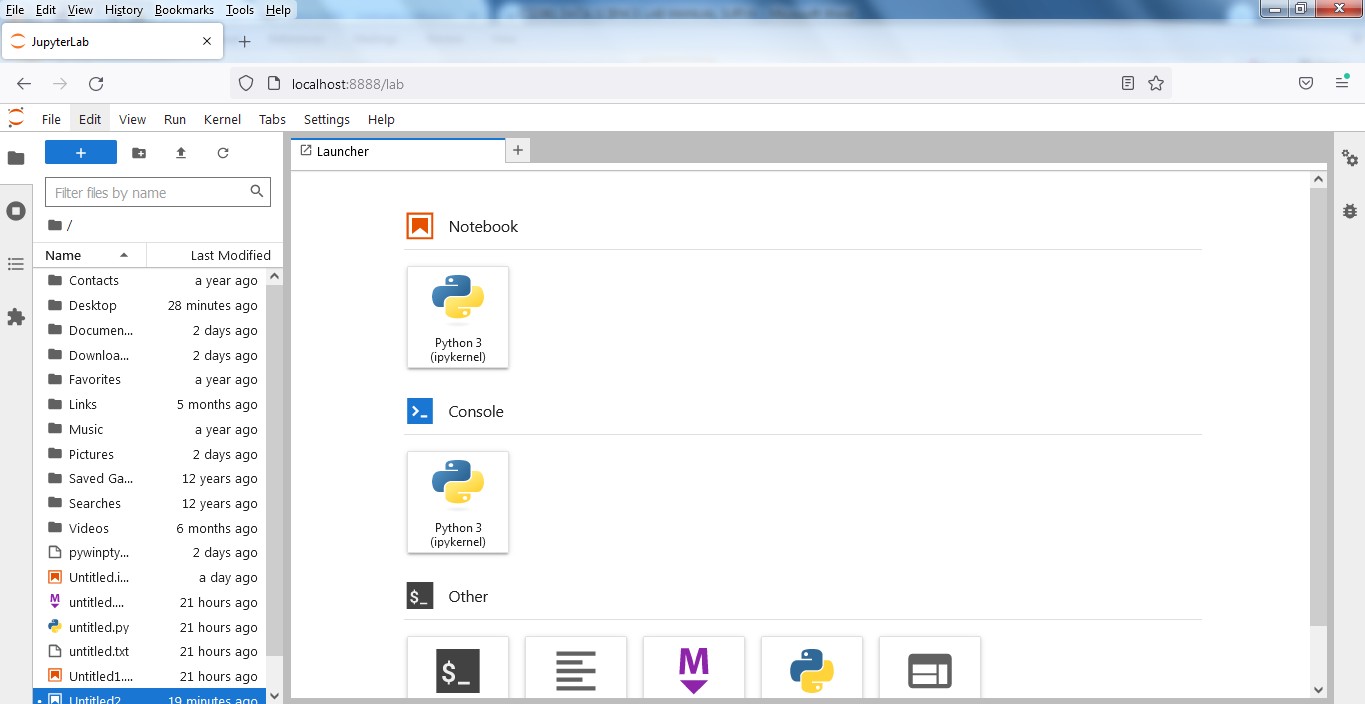




To open jupyter-lab:

Open command prompt and type jupyter-lab.

Then after initializing all the necessary packages, it will open as follows:



Click on new notebook, then the new file will be opened with .ipynb file extension. Then type python code and execute the code using Shift+Enter.

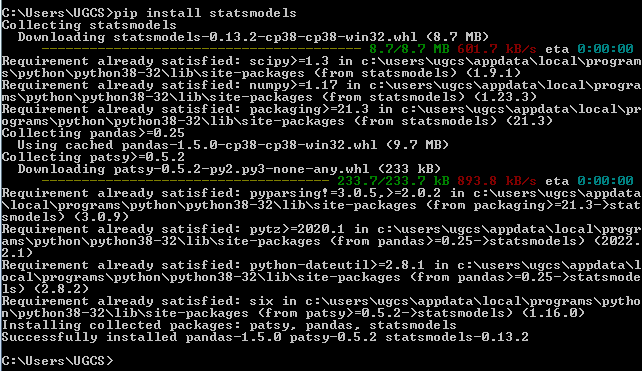
# Sample Program and Output:



**STASMODELS:**

[statsmodels](https://www.statsmodels.org/stable/about.html#about-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. An extensive list of result statistics are available for each estimator. The results are tested against existing statistical packages to ensure that they are correct. statsmodels supports specifying models using R-style formulas and pandas DataFrames.

statsmodels is a Python package that provides a complement to scipy for statistical computations including descriptive statistics and estimation and inference for statistical models.



# Sample Program:

import numpy as np import pandas as pd

import statsmodels.formula.api as smf

df = pd.read\_csv(r"C:\Users\UGCS\Desktop\headbrain11.csv") print(df.head())

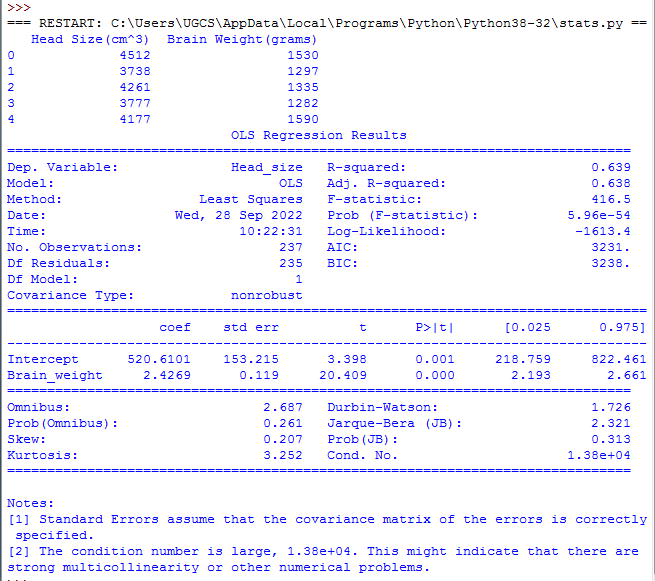
# fitting the model

df.columns = ['Head\_size', 'Brain\_weight']

model = smf.ols(formula='Head\_size ~ Brain\_weight', data=df).fit() # model summary

print(model.summary())

# Output:



**PANDAS:**

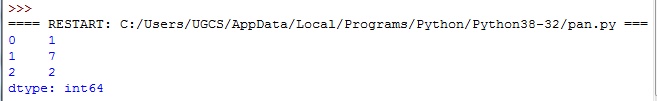
Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. Pandas allow us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant. Relevant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



**Sample Program:** import pandas as pd a = [1, 7, 2]

myvar = pd.Series(a) print(myvar)

# Output:



**EX.NO:2 WORKING WITH NUMPY ARRAYS**

**AIM:**

To write a python code to work with numpy arrays.

# ALGORITHM:

1. Import the numpy package.
2. Create the array using numpy.array()
3. Indexing can be done like this: [*start*:*end*].
4. The NumPy array object has a property called dtype that returns the data type of the array.
5. To deal with iteration to multi-dimensional arrays in numpy, we can do this using basic for loop of python.
6. To join two arrays, the concatenate() function along with the axis can be used.
7. Use array\_split() for splitting arrays, we pass it the array we want to split and the number of splits.
8. To search an array, use the where() method.
9. The NumPy ndarray object has a function called sort(), that will sort a specified array.
10. In NumPy, you filter an array using a *boolean index list*.
    1. If the value at an index is True that element is contained in the filtered array
    2. if the value at that index is False that element is excluded from the filtered array.

# PROGRAM:

**#Create a 0-D array:**

import numpy as np arr = np.array(42) print(arr)

# OUTPUT:

42

# #Create a 1-D array:

import numpy as np

arr = np.array([1, 2, 3, 4, 5]) print(arr)

# OUTPUT:

[1,2,3,4,5]

# #Create a 2-D array:

import numpy as np

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

# OUTPUT:

[[1 2 3]

[4 5 6]]

# #Create a 3-D array:

import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(arr)

# OUTPUT:

[[[1 2 3]

[4 5 6]]

[[1 2 3]

[4 5 6]]]

# #Check how many dimensions the arrays have:

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

# #Accessing Array Elements:

import numpy as np

arr = np.array([1, 2, 3, 4]) print(arr[1])

# OUTPUT:

2

# #Accessing 2-D Arrays:

import numpy as np

arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])

print('2nd element on 1st row: ', arr[0, 1])

# OUTPUT:

2nd element on 1st row: 2

# #Array Slicing:

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7]) print(arr[4:])

# OUTPUT:

[5 6 7]

# #Slicing 2-D Arrays:

import numpy as np

arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])

# OUTPUT:

[7 8 9]

# #Getting the data type of an array:

import numpy as np

arr = np.array(['apple', 'banana', 'cherry']) print(arr.dtype)

# OUTPUT:

<U6

# #Iterate on the elements of 1-D array:

import numpy as np arr = np.array([1, 2, 3]) for x in arr:

print(x)

# OUTPUT:

1

2

3

# #Iterating 2-D Arrays:

import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]]) for x in arr:

print(x)

# OUTPUT:

[1 2 3]

[4 5 6]

**#Join two arrays:** import numpy as np arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2)) print(arr)

# OUTPUT:

[1 2 3 4 5 6]

# #Splitting the array:

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6]) newarr = np.array\_split(arr, 3) print(newarr)

# OUTPUT:

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

# #Searching Arrays:

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 4, 4]) x = np.where(arr == 4)

print(x)

# OUTPUT:

(array([3, 5, 6], dtype=int32),)

# #Sorting Arrays:

import numpy as np

arr = np.array([3, 2, 0, 1]) print(np.sort(arr))

OUTPUT:

[0 1 2 3]

# #Filtering Arrays:

import numpy as np

arr = np.array([41, 42, 43, 44]) x = [True, False, True, False] newarr = arr[x]

print(newarr)

# OUTPUT:

[41 43]

# RESULT:

Thus the python code to work with numpy arrays has been implemented and executed successfully.

# EX.NO:3 WORKING WITH PANDAS DATA FRAMES

# **AIM:**

To write a python program to work with pandas data frames.

# ALGORITHM:

1. Pandas is a Python library used for working with data sets.
2. It has functions for analyzing, cleaning, exploring, and manipulating data.
3. Dataframes can be created using list or dictionary.
4. Dataframes can also be used to load any other .csv or .xslx files.
5. It can be used to replace the null values with other values.
6. It can also perform data and its statistical analyzing.

**PROGRAM:**

# #[Creating a dataframe using List](https://www.geeksforgeeks.org/create-a-pandas-dataframe-from-lists/):

import pandas as pd

lst = [‘Anna’, ‘University, ‘Chennai’, ‘Sri’, ‘Ramakrishna’, ‘College’, ‘of’,’Engineering’] df = pd.DataFrame(lst)

print(df)

# OUTPUT:

**#[Creating DataFrame from dict of ndarray/lists](https://www.geeksforgeeks.org/python-create-a-pandas-dataframe-from-a-dict-of-equal-length-lists/):**

import pandas as pd

data = {'Name':['Tom', 'nick', 'krish', 'jack'], 'Age':[20, 21, 19, 18]} df = pd.DataFrame(data)

print(df)

# OUTPUT:

**#[Column Selection](https://www.geeksforgeeks.org/how-to-select-multiple-columns-in-a-pandas-dataframe/):**

import pandas as pd

data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'], 'Age':[27, 24, 22, 32],

'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],

'Qualification':['Msc', 'MA', 'MCA', 'Phd']} df = pd.DataFrame(data)

print(df[['Name', 'Qualification']])

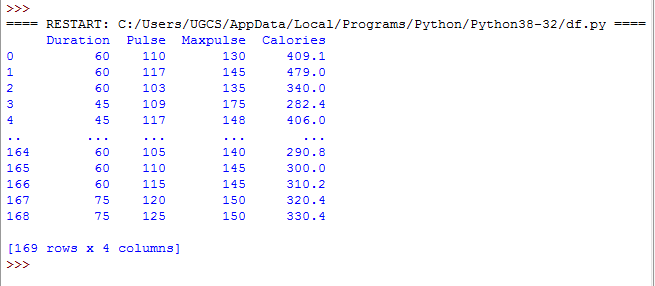
# OUTPUT:

**#Load Files Into a DataFrame:**

import pandas as pd

df = pd.read\_csv(r"C:\Users\UGCS\Desktop\data.csv") print(df)

# OUTPUT:



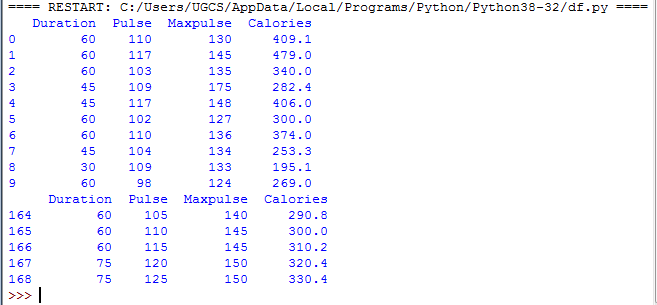
**# Viewing the Data**

import pandas as pd

df = pd.read\_csv(r"C:\Users\UGCS\Desktop\data.csv") print(df.head(10))

print(df.tail(5))

# OUTPUT:



**#Replacing Nullvalues:**

import pandas as pd

df = pd.read\_csv(r"C:\Users\UGCS\Desktop\data.csv") df.fillna(130, inplace = True)

print(df)

# OUTPUT:

**#Checking for missing values using isnull() and notnull() :**

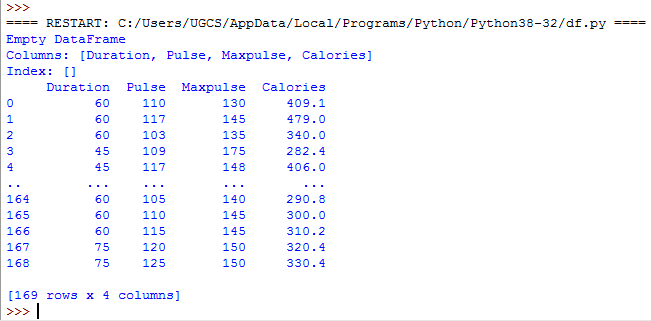
import pandas as pd

df = pd.read\_csv(r"C:\Users\UGCS\Desktop\data.csv") bool\_series = pd.isnull(df["Pulse"])

print(df[bool\_series])

bool\_series = pd.notnull(df["Pulse"]) print(df[bool\_series])

# OUTPUT:



# RESULT:

Thus the python program to work with pandas data frames have been implemented and executed successfully.

**EX.NO:4 READING DATA FROM TEXT FILES, EXCEL AND THE WEB AND EXPLORING VARIOUS COMMANDS**

**AIM:**

To read the data from text files, Excel and the web and exploring various commands for

doing descriptive analytics on the Iris data set.

# PRE-REQUISITES:

pip install xlrd

pip install openpyxl pip install requests

pip install beautifulsoup4

# ALGORTIHM:

1. Open the file to be written using open() function.
2. The file can opened with read/write/append/… mode.
3. Write the file using write() or writelines() function.
4. seek(n) takes the file handle to the nth byte from the beginning.
5. Close the file using close().
6. To read the data from the excel, install pandas.
7. Create a dataframe using read\_excel()
8. To read the data from the web, install requests and beautifulsoup4.
9. The content from the web can be accessed using the function requests.get(url).
10. To perform descriptive analytics on a dataset, install seaborn, matplotlib and pandas to explore various functions.

# PROGRAM:

**#Reading data from text file:**

# Program to show various ways to read and write data in a file. file1 = open("myfile.txt","w")

L = ["This is Python \n","This is datascience \n","This is jupyter \n"] # \n is placed to indicate EOL (End of Line)

file1.write("Hello \n") file1.writelines(L)

file1.close() #to change file access modes file1 = open("myfile.txt","r+") print("Output of Read function is ") print(file1.read())

print()

# seek(n) takes the file handle to the nth byte from the beginning. file1.seek(0)

print( "Output of Readline function is ") print(file1.readline())

print()

file1.seek(0)

# To show difference between read and readline print("Output of Read(9) function is ") print(file1.read(9))

print()

file1.seek(0)

print("Output of Readline(9) function is ") print(file1.readline(9))

file1.seek(0)

# readlines function

print("Output of Readlines function is ") print(file1.readlines())

print() file1.close()

# OUTPUT:

**#Reading data from excel:**

# Create a new excel file import pandas as pd

# read by default 1st sheet of an excel file dataframe1 = pd.read\_excel('excel.xlsx') print(dataframe1)

# OUTPUT:

**#Reading data from the web:**

import requests

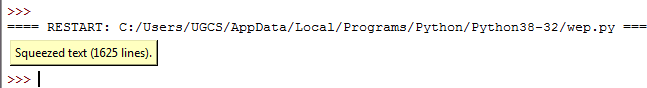
from bs4 import BeautifulSoup import time

url ="https://sriramakrishna.ac.in/srce-about-college.php" page = requests.get(url)

soup = BeautifulSoup(page.content,'html.parser')

#result = soup.find(id="mosaic-provider-jobcards") #job\_elements = result.find\_all("div", class\_="job\_seen\_beacon") print(soup)

# OUTPUT:



**# descriptive analytics on the Iris data set**

import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt # Reading the CSV file

df = pd.read\_csv("Iris.csv") df.shape

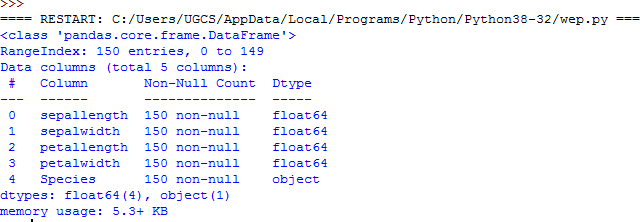
df.info() df.describe()

data = df.drop\_duplicates(subset ="Species",) data

df.value\_counts("Species")

sns.countplot(x='Species', data=df, ) plt.show()

# OUTPUT:



# RESULT:

Thus the python code to read the data from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set.

**EX.NO:5A UNIVARIATE ANALYSIS USING DIABETES DATASET**

**AIM:**

To perform Univariate analysis such as Frequency, Mean, Median, Mode, Variance,

Standard Deviation, Skewness and Kurtosis on the diabetes dataset.

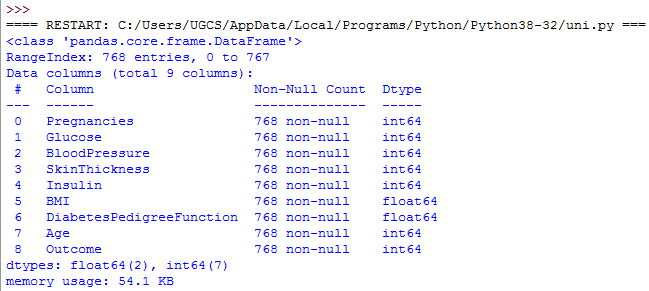
# ALGORITHM:

1. Install pandas.
2. To find the frequency of a single variable on a dataset, use the value\_counts() function.
3. To find the mean of a single variable on a dataset, use the mean() function.
4. To find the median of a single variable on a dataset, use the median() function.
5. To find the mode of a single variable on a dataset, use the mode() function.
6. To find the variance of a single variable on a dataset, install and import statistics and use the statistics.variance() function.
7. To find the standard deviation of a single variable on a dataset, use the std() function.
8. To find the skewness of a single variable on a dataset, install and import scipy and use the scipy.stats.skew() function.
9. To find the kurtosis of a single variable on a dataset, install and import scipy and use the scipy.stats.kurtosis() function.

**PROGRAM:**

**#Reading dataset** import pandas as pd #create DataFrame

df = pd.read\_csv("diabetes.csv") df.info()

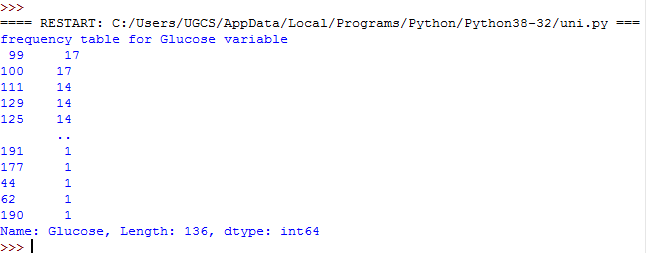
df.describe()

# #Finding Frequency

import pandas as pd #create DataFrame

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

#create frequency table for 'Glucose' variable f1=df['Glucose'].value\_counts() print('frequency table for Glucose variable\n',f1)

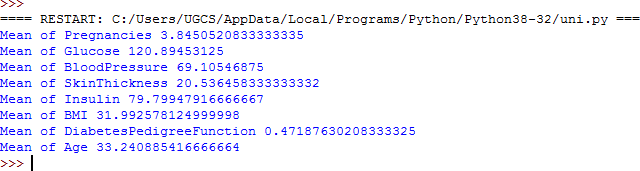


**#Finding Mean** import pandas as pd #create DataFrame

df = pd.read\_csv("diabetes.csv") m1=df['Pregnancies'].mean() print('Mean of Pregnancies',m1) m2=df['Glucose'].mean() print('Mean of Glucose',m2) m3=df['BloodPressure'].mean() print('Mean of BloodPressure',m3) m4=df['SkinThickness'].mean() print('Mean of SkinThickness',m4) m5=df['Insulin'].mean() print('Mean of Insulin',m5) m6=df['BMI'].mean()

print('Mean of BMI',m6) m7=df['DiabetesPedigreeFunction'].mean() print('Mean of DiabetesPedigreeFunction',m7) m8=df['Age'].mean()

print('Mean of Age',m8)



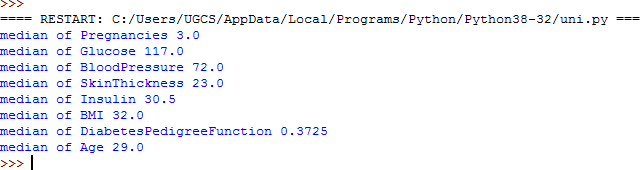
# #Finding Median

import pandas as pd

df = pd.read\_csv("diabetes.csv") m1=df['Pregnancies'].median() print('median of Pregnancies',m1) m2=df['Glucose'].median() print('median of Glucose',m2) m3=df['BloodPressure'].median() print('median of BloodPressure',m3) m4=df['SkinThickness'].median() print('median of SkinThickness',m4) m5=df['Insulin'].median() print('median of Insulin',m5) m6=df['BMI'].median() print('median of BMI',m6)

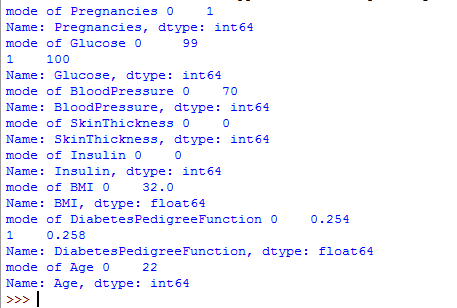
m7=df['DiabetesPedigreeFunction'].median() print('median of DiabetesPedigreeFunction',m7) m8=df['Age'].median()

print('median of Age',m8)



**#Finding Mode** import pandas as pd #create DataFrame df = pd.read\_csv("diabetes.csv") m1=df['Pregnancies'].mode() print('mode of Pregnancies',m1) m2=df['Glucose'].mode() print('mode of Glucose',m2) m3=df['BloodPressure'].mode() print('mode of BloodPressure',m3) m4=df['SkinThickness'].mode() print('mode of SkinThickness',m4) m5=df['Insulin'].mode() print('mode of Insulin',m5) m6=df['BMI'].mode()

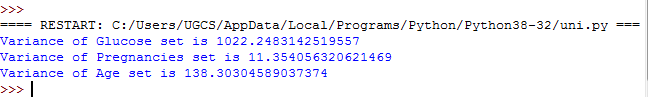
print('mode of BMI',m6) m7=df['DiabetesPedigreeFunction'].mode() print('mode of DiabetesPedigreeFunction',m7) m8=df['Age'].mode()

print('mode of Age',m8)

**#Finding Variance** import pandas as pd import statistics #create DataFrame

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

print("Variance of Glucose set is % s"%(statistics.variance(df.Glucose))) print("Variance of Pregnancies set is % s"%(statistics.variance(df.Pregnancies))) print("Variance of Age set is % s"%(statistics.variance(df.Age)))



# #Finding Standard Deviation

import pandas as pd #create DataFrame

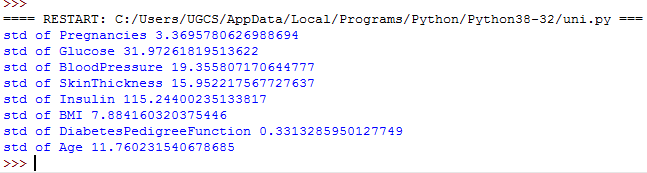
df = pd.read\_csv("diabetes.csv") s1=df['Pregnancies'].std() print('std of Pregnancies',s1) s2=df['Glucose'].std()

print('std of Glucose',s2) s3=df['BloodPressure'].std() print('std of BloodPressure',s3) s4=df['SkinThickness'].std() print('std of SkinThickness',s4) s5=df['Insulin'].std()

print('std of Insulin',s5) s6=df['BMI'].std() print('std of BMI',s6)

s7=df['DiabetesPedigreeFunction'].std() print('std of DiabetesPedigreeFunction',s7) s8=df['Age'].std()

print('std of Age',s8)

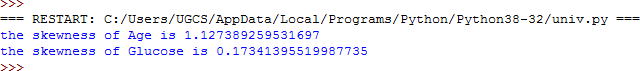


# #Finding Skewness

import scipy

import pandas as pd #create DataFrame

df = pd.read\_csv("diabetes.csv") s1=scipy.stats.skew(df.Age, axis=0, bias=True) print('the skewness of Age is',s1) s2=scipy.stats.skew(df.Glucose, axis=0, bias=True) print('the skewness of Glucose is',s2)

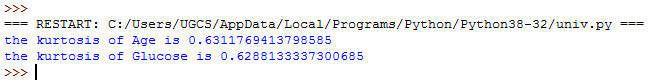


# #Finding Kurtosis

import scipy

import pandas as pd #create DataFrame

df = pd.read\_csv("diabetes.csv") k1=scipy.stats.kurtosis(df.Age, axis=0, bias=True) print('the kurtosis of Age is',k1) k2=scipy.stats.kurtosis(df.Glucose, axis=0, bias=True) print('the kurtosis of Glucose is',k2)



# RESULT:

Thus the Univariate analysis such as Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis on the diabetes dataset have been performed successfully.

# EX.NO:5B BIVARIATE ANALYSIS USING DIABETES DATASET

**AIM:**

To perform Bivariate analysis such as Linear and logistic regression modeling on the

diabetes dataset.

# ALGORITHM:

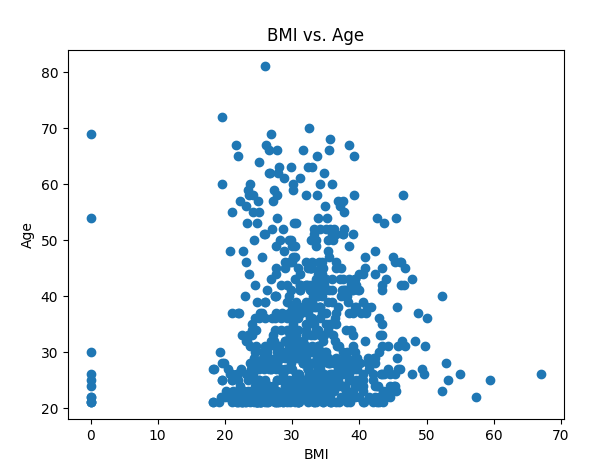
1. Linear regression uses the relationship between the data-points to draw a straight line through all them.
2. This line can be used to predict future values.
3. Import scipy and draw the line of Linear Regression
4. Define response and explanatory variable.
5. Add constant to predictor variables.
6. Create the model using, sm.OLS(y, x).fit().
7. View the model using summary().
8. To construct the correlation matrix, use corr().
9. To model the logistic regression, Install scikit-learn of version 0.24.2.
10. Read and explore the data.
11. Split the Dataset as Train and Test dataset
12. Train the model using, LogisticRegression()
13. Visualize the performance of logistic regression model.

**PROGRAM:**

# #creating scatterplots

import pandas as pd

import matplotlib.pyplot as plt df = pd.read\_csv("diabetes.csv") plt.scatter(df.BMI, df.Age) plt.title('BMI vs. Age') plt.xlabel('BMI') plt.ylabel('Age')

plt.show()

# #simple linear regression

import pandas as pd

import matplotlib.pyplot as plt import statsmodels.api as sm

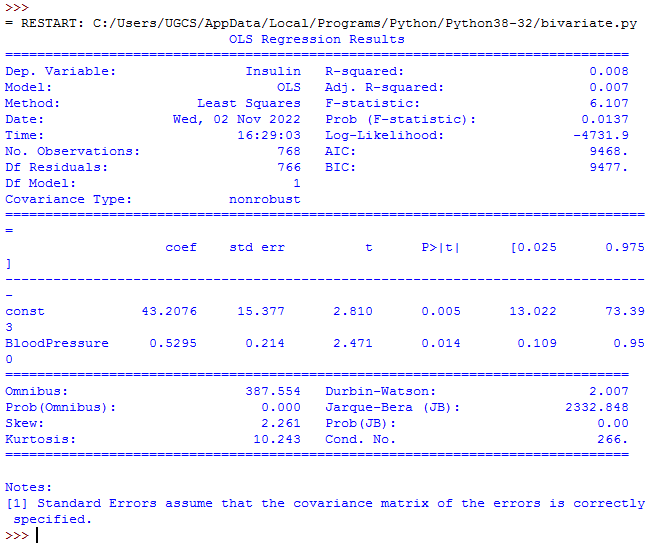
df = pd.read\_csv("diabetes.csv") #define response variable

y = df['Insulin']

#define explanatory variable x = df[['BloodPressure']]

#add constant to predictor variables x = sm.add\_constant(x)

#fit linear regression model model = sm.OLS(y, x).fit() #view model summary print(model.summary())

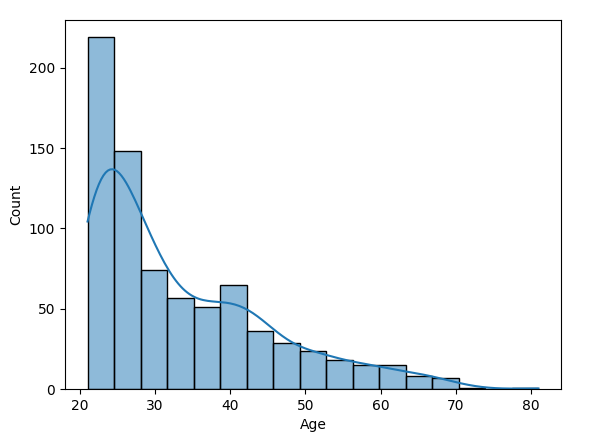


# #creating histogram

import pandas as pd

import matplotlib.pyplot as plt import statsmodels.api as sm import seaborn as sns

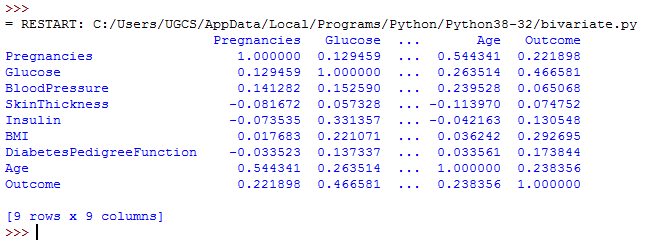
df = pd.read\_csv("diabetes.csv") sns.histplot(df.Age,kde=True) plt.show()



# #constructing correlation matrix:

import pandas as pd

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



# #LOGISTIC REGRESSION MODELING:

**PRE-REQUISITES:**

Install scikit-learn of version 0.24.2 in the command prompt as follows:

# pip install scikit-learn==0.24.2

**#Import Libraries** import pandas as pd import numpy as np

import matplotlib.pyplot as plt

# #Read and Explore the data

dataset = pd.read\_csv("diabetes.csv") # input

x = dataset.iloc[:, [2, 3]].values # output

y = dataset.iloc[:, 4].values

# #Splitting The Dataset: Train and Test dataset

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

xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size = 0.25, random\_state = 0) from sklearn.preprocessing import StandardScaler

sc\_x = StandardScaler()

xtrain = sc\_x.fit\_transform(xtrain) xtest = sc\_x.transform(xtest)

print (xtrain[0:10, :])

# #Train The Model

from sklearn.linear\_model import LogisticRegression classifier = LogisticRegression(random\_state = 0) classifier.fit(xtrain, ytrain)

y\_pred = classifier.predict(xtest)

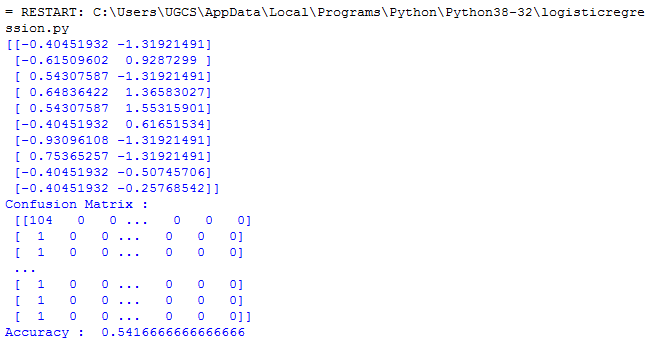
# #Evaluation Metrics

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

print ("Confusion Matrix : \n", cm)

from sklearn.metrics import accuracy\_score

print ("Accuracy : ", accuracy\_score(ytest, y\_pred))



# #Visualizing the performance of logistic regression model

from matplotlib.colors import ListedColormap X\_set, y\_set = xtest, ytest

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1,

stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1,

stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(

np.array([X1.ravel(), X2.ravel()]).T).reshape(

X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green'))) plt.xlim(X1.min(), X1.max())

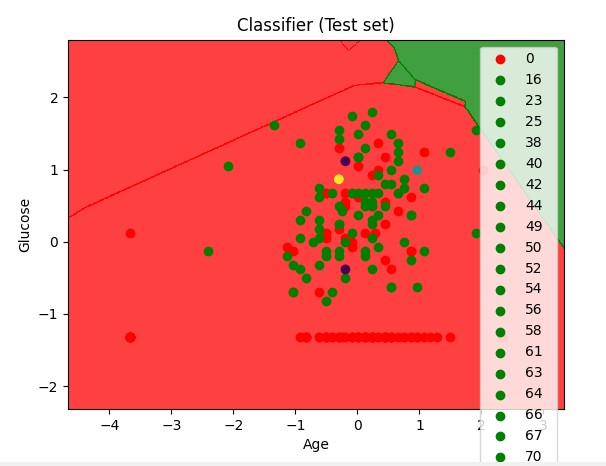
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)): plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Classifier (Test set)') plt.xlabel('Age')

plt.ylabel('Glucose') plt.legend() plt.show()



# RESULT:

Thus the Bivariate analysis such as Linear and logistic regression modeling on the diabetes dataset have been performed and analyzed successfully.

# EX.NO:5C MULTIPLE REGRESSION ANALYSIS USING DIABETES DATASET

# AIM:

To perform multiple regression analysis using diabetes dataset.

# ALGORITHM:

1. Multiple regression is like [linear regression,](https://www.w3schools.com/python/python_ml_linear_regression.asp) but with more than one independent value, meaning that we try to predict a value based on two or more variables.
2. Import pandas, numpy and matplotlib packages.
3. Install and import sklearn(scikit-learn) package.
4. Import linear\_model from scikit-learn.
5. Plot the graph using scatter()
6. Generate training and testing data from the dataset.
7. Model the dataset using, regr.fit()
8. Analyze the coefficients and intercepts.

**PROGRAM:**

from mpl\_toolkits.mplot3d import Axes3D import pandas as pd

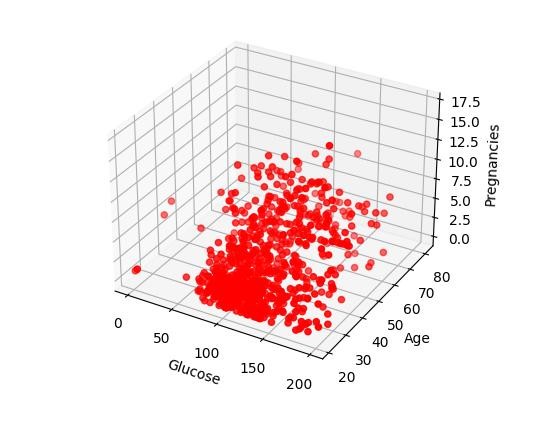
import matplotlib.pyplot as plt import numpy as np

from sklearn import linear\_model np.random.seed(19680801) data=pd.read\_csv("diabetes.csv") data.head(210)

data = data[["Glucose","Age","Pregnancies"]] fig=plt.figure() ax=fig.add\_subplot(111,projection='3d') n=100

ax.scatter(data["Glucose"],data["Age"],data["Pregnancies"],color="red") ax.set\_xlabel("Glucose")

ax.set\_ylabel("Age") ax.set\_zlabel("Pregnancies") plt.show()



# #Generating training and testing data from our data:

train = data[:(int((len(data)\*0.8)))]

test = data[(int((len(data)\*0.8))):]

# # Modeling:Using sklearn package to model data :

regr = linear\_model.LinearRegression() train\_x = np.array(train[["Glucose"]]) train\_y = np.array(train[["Age"]]) regr.fit(train\_x,train\_y)

ax.scatter(data["Glucose"],data["Age"],data["Pregnancies"],color="red") plt.plot(train\_x, regr.coef\_\*train\_x + regr.intercept\_, '-r') ax.set\_xlabel("Glucose")

ax.set\_ylabel("Age") ax.set\_zlabel("Pregnancies") print ("coefficients : ",regr.coef\_) #Slope

print ("Intercept : ",regr.intercept\_)



# RESULT:

Thus the multiple regression analysis using diabetes dataset have been implemented and executed successfully.

# EX.NO:6 EXPLORING VARIOUS PLOTTING FUNCTIONS USING ANY DATASET

**AIM:**

To apply and explore various plotting functions such as Normal curves, Density and

contour plots, Correlation and scatter plots, Histograms and Three dimensional plotting on UCI data sets.

# ALGORITHM:

1. Import numpy, matplotlib, scipy and pandas.
2. Create the dataframe.
3. Find mean and standard deviation from the dataset.
4. Find the normal curve snd using, stats.norm()
5. Generate 1000 randomvalues and plot the normalcurve.
6. Install and import seaborn package.
7. Draw the density plot using distplot().
8. Draw the contour plot using kdeplot().
9. Construct the correlation matrix using, con.corr().
10. Display the coefficient of correlation using stats.pearsonr()
11. Plot the histogram using hist().
12. To model 3D plotting, import Axes3D.

**PROGRAM:**

# #NORMAL CURVES:

import numpy as np

import matplotlib.pyplot as plt from scipy import stats

import pandas as pd #create DataFrame

df = pd.read\_csv("diabetes.csv") mu=df['Pregnancies'].mean() std=df['Pregnancies'].std()

snd = stats.norm(mu, std)

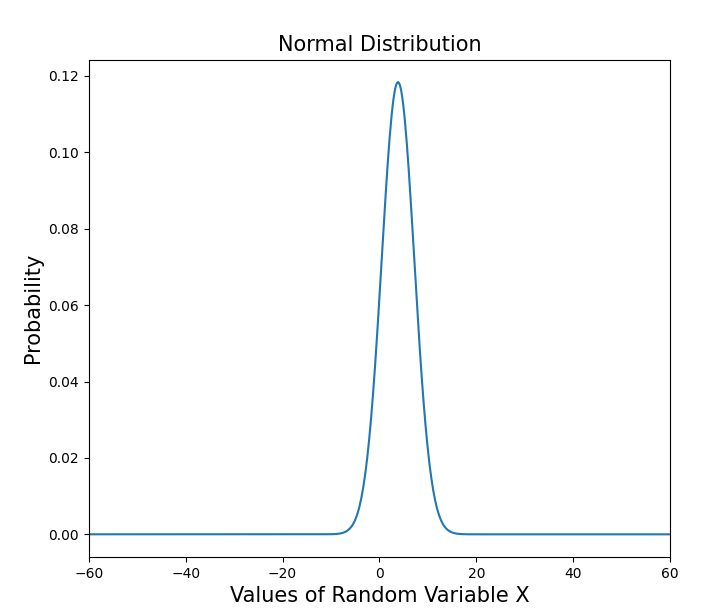
# Generate 1000 random values between -100, 100

x = np.linspace(-100, 100, 1000) plt.figure(figsize=(7.5,7.5)) plt.plot(x, snd.pdf(x))

plt.xlim(-60, 60)

plt.title('Normal Distribution', fontsize='15') plt.xlabel('Values of Random Variable X', fontsize='15') plt.ylabel('Probability', fontsize='15')

plt.show()



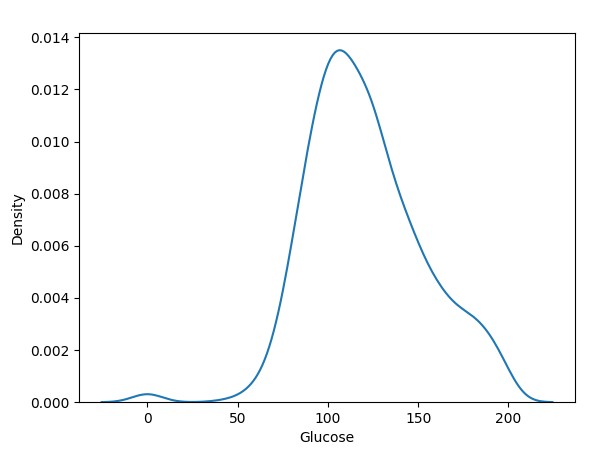
# #DENSITY AND CONTOUR PLOTS:

**#density plot:**

import seaborn as sns

import matplotlib.pyplot as plt import pandas as pd

df = pd.read\_csv("diabetes.csv") sns.distplot(a=df.Glucose, hist=False) plt.show()



# #contour plot:

import seaborn as sns

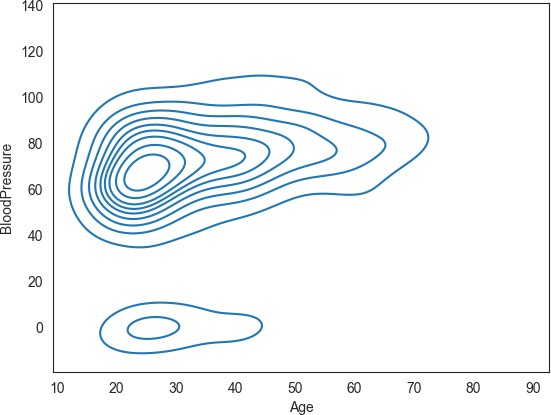
import matplotlib.pyplot as plt import pandas as pd

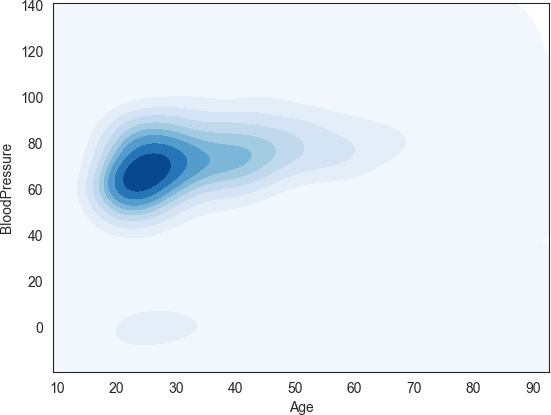
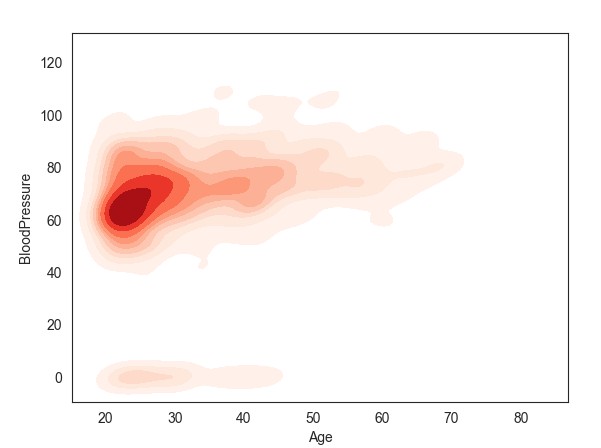
df = pd.read\_csv("diabetes.csv") sns.set\_style("white") sns.kdeplot(x=df.Age, y=df.BloodPressure) plt.show()

sns.kdeplot(x=df.Age, y=df.BloodPressure, cmap="Reds", shade=True, bw\_adjust=.5) plt.show()

sns.kdeplot(x=df.Age, y=df.BloodPressure, cmap="Blues", shade=True, thresh=0) plt.show()







# #CORRELATION AND SCATTER PLOTS:

import pandas as pd

import matplotlib.pyplot as plt con = pd.read\_csv('diabetes.csv') print(con)

import seaborn as sns

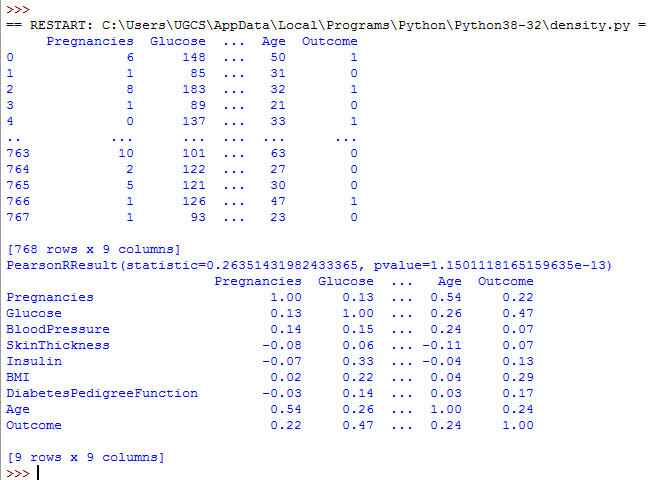
sns.scatterplot(x="Age", y="Glucose", data=con); plt.show()

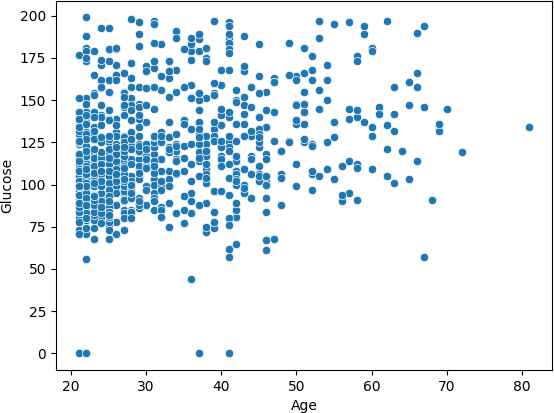
sns.lmplot(x="Age", y="Glucose", hue="BMI", data=con); plt.show()

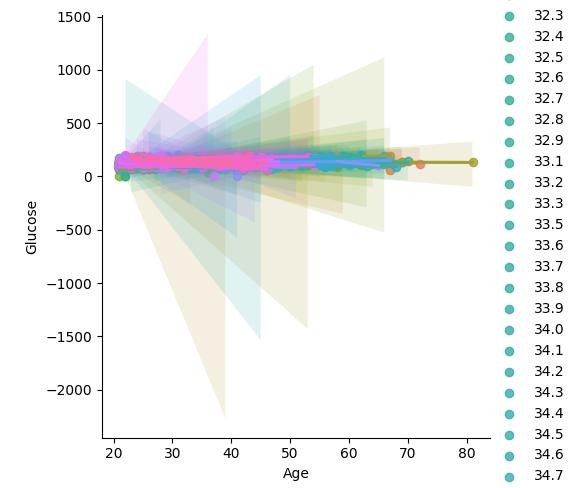
# #coefficient of correlation

from scipy import stats cr=stats.pearsonr(con['Glucose'], con['Age']) print(cr)

**#correlation matrix** cormat = con.corr() print(round(cormat,2))







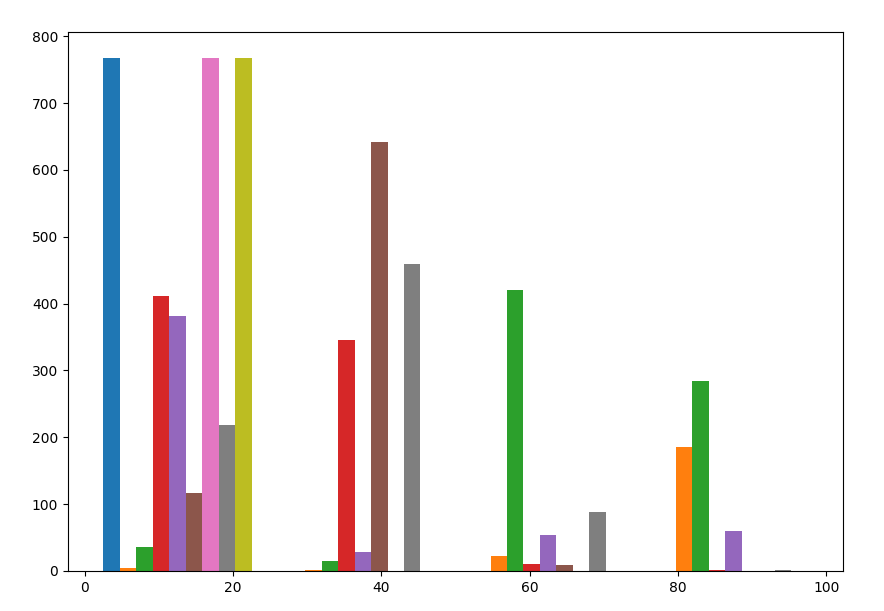
# #HISTOGRAMS:

import pandas as pd import numpy as np

import matplotlib.pyplot as plt a = pd.read\_csv('diabetes.csv') # Creating histogram

fig, ax = plt.subplots(figsize =(10, 7))

ax.hist(a, bins = [0, 25, 50, 75, 100]) plt.show()



# #THREE DIMENSIONAL PLOTTING:

from mpl\_toolkits.mplot3d import Axes3D import pandas as pd

import matplotlib.pyplot as plt import numpy as np

from sklearn import linear\_model np.random.seed(19680801) data=pd.read\_csv("diabetes.csv") data.head(210)

data = data[["BMI","BloodPressure","Insulin"]] fig=plt.figure() ax=fig.add\_subplot(111,projection='3d')

n=100

ax.scatter(data["BMI"],data["BloodPressure"],data["Insulin"],color="red") ax.set\_xlabel("BMI")

ax.set\_ylabel("BloodPressure") ax.set\_zlabel("Insulin") plt.show()

# RESULT:

Thus the various plotting functions such as Normal curves, Density and contour plots, Correlation and scatter plots, Histograms and three dimensional plotting have been explored successfully on UCI data sets.

# EX.NO:7 VISUALIZING GEOGRAPHIC DATA WITH BASEMAP

**AIM:**

To implement visualization of geographic data with basemap.

# PRE-REQUISITIES:

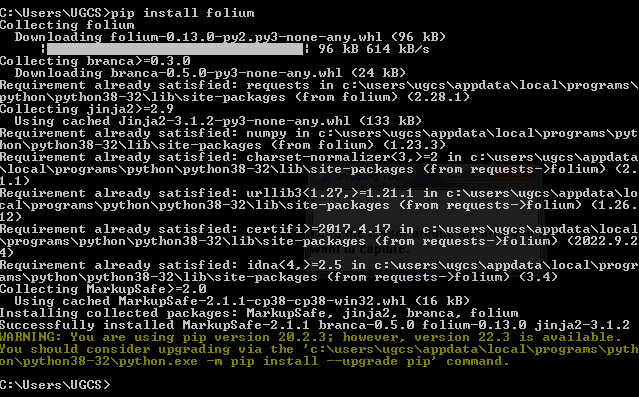
Install folium.

# ALGORITHM:

1. Import folium and pandas libraries.
2. Initialize the map and store it in a m object
3. Use the function, folium.Map()
4. Save the map using save() function.
5. Open and view the file using any browser.

**PROGRAM:**

Installation of Folium:



# import the folium, pandas libraries import folium

import pandas as pd

# initialize the map and store it in a m object m = folium.Map(location = [40, -95],

zoom\_start = 4)

# show the map m.save('my\_map.html')

# OUTPUT:

# 