EX. N0.:1 PACKAGES FOR DATA SCIENCE IN PYTHON

DATE

AIM:

To download, install and explore the features of NumPy , SciPy, Jupyter, Statsmodels, and Pandas packages in Python.

ALGORITHM:

Python is an open-source, object-oriented , and cross-platform programming language. Compared to programming languages like C++ or Java, Python is very concise. It allows us to build a working software prototype in a very short time. It has become the most used language in the data scientist's toolbox. It is also a generalpurpose language, and it is very flexible due to a variety of available packages that solve a wide spectrum of problems and necessities . To install the necessary packages , use 'pip'.

Anaconda [(http://continuum.io/downloads) i](http://continuum.io/downloads))s a Python distribution offered by Continuum Analytics that includes nearly 200 packages , which comprises NumPy, SciPy, pandas , Jupyter, Matplotlib , Scikit-learn, and NLTK. It is a cross• platform distribution (Windows, Linux , and Mac OS X) that can be installed on machines with other existing Python distributions and versions. Its base version is free; instead, add-ons that contain advanced features are charged separately. Anaconda introduces 'conda', a binary package manager , as a command-line tool to manage your package installations . Anaconda's goal is to provide enterprise• ready Python distribution for large-scale processing , predictive analytics, and scientific computing .

1. Download Anaconda
2. Install Anaconda
3. Start Anaconda
4. Install data science packages
5. Download Anaconda

This step downloads the Anaconda Python package for the Windows platform.

Anaconda is a free and easy-to-use environment for scientific Python .

* 1. Visit the Anaconda homepage .
  2. Click "Anaconda" from the menu and click "Download" to go to the download page.
  3. Choose the download suitable for your platform (Windows, OSX, or Linux):
     + Choose Python 3.5
     + Choose the Graphical Installer

1. **Install Anaconda**

This step installs the Anaconda Python software on the system.

This step assumes that sufficient administrative privileges are contained to install software on the system.

* 1. Double click the downloaded file.
  2. Follow the installation wizard .

1. **Start Anaconda**

Anaconda comes with a suite of graphical tools called Anaconda Navigator. Start Anaconda Navigator by opening it from the application launcher.

First, start with the Anaconda command line environment called conda.

Conda is fast, and simple, it's hard for error messages to hide, and you can quickly confirm your environment is installed and working correctly.

* 1. Open a terminal (command line window) .
  2. Confirm conda is installed correctly, by typing : conda -V



* 1. Confirm Python is installed correctly by typing : python -V



* 1. **Install packages**

With pip, a package is installed . To install the < package-name > generic package, run this command :

$> **pip install < package-name** >

To install the <package-name > generic package , you just need to run the following command:

$> **conda install <package-name>**

To install a particular version of the package

$> **conda install <package-name>=l.11.0**

To install multiple packages at once by listing all their names :

$> **conda install <package-name-1> <package-name-2>**

To update a package that you previously installed, you can keep on using conda:

$> **conda update <package-name>**

To update all the available packages simply by using the --all argument

$> **conda update -all**

To uninstall packages using conda:

$> **conda remove <package-name>**

**Installing Data Science Packages**

1. **NumPy**

NumPy is the true analytical workhorse of the Python language. It provides the user with multidimensional arrays, along with a large set of functions to operate a multiplicity of mathematical operations on these arrays. Arrays are blocks of data arranged along multiple dimensions , which implement mathematical vectors and matrices. Characterized by optimal memory allocation , arrays are useful not just for storing data, but also for fast matrix operations (vectorization), which are indispensable when solving adhoc data science problems.

$> **conda install numpy**

1. **SciPy**

SciPy completes NumPy's functionalities , offering a larger variety of scientific **algorithms for linear algebra, sparse matrices, signal and image processing ,** optimization, fast Fourier transformation , and much more.

$> **conda install scipy**

1. **Statsmodels**

Statsmodels is a complement to SciPy's statistical functions . It features generalized linear models , discrete choice models, time series analysis, and a series of descriptive statistics as well as parametric and nonparametric tests.

$> **conda install -c conda-forge statsmodels**

1. **Pandas**

The pandas package deals with everything that NumPy and SciPy cannot do. Thanks to its specific data structures, namely DataFrames and Series, pandas allow us to handle complex tables of data of different types and time series. It enables the easy and smooth loading of data from a variety of sources . The data can then be sliced, diced, handled with missing elements , added, renamed , aggregated , reshaped , and finally visualized.

$> **conda install pandas**

1. **Jupyter**

A scientific approach requires the fast experimentation of different hypotheses in a reproducible fashion. Initially named!Python and limited to working only with the Python language, Jupyter was created to address the need for an interactive command shell for several languages (based on the shell, web browser , and application interface) , featuring graphical integration, customizable commands, rich history (in the JSON format), and computational parallelism for enhanced performance .

**Steps to install Jupyter using Anaconda**

* Launch Anaconda Navigator
* Click on the Install Jupyter Notebook Button

EX. N0.:2.a WORKING WITH NUMPY ARRAYS BASIC NUMPY OPERATIONS PROGRAM:

**(i) Creation of different types of Numpy arrays and displaying basic information**

# Importing numpy

import numpy as np

# Defining 1D array mylDArray = np.array([l , 8, 27, 64] print(myl DArray)

# Defining and printing 2D array my2DArray = np .array([[l , 2, 3, 4], [2, 4, 9, 16], [4, 8, 18, 32]]) print(my2DArray)

#Defining and printing 3D array my3Darray = np.array([[[ **1,** 2 , 3 , 4],[ 5 , 6 , 7 ,8]], [[ **1,** 2, 3, 4],[ 9, 10, 11,

12]]]) print(my3Darray)

# Print out memory address print(my2DArray.data) # Print the shape of array print(my2DArray .shape)

# Print out the data type of the array print(my2DArray .dtype)

# Print the stride of the array.

print(my2DArray .strides)

**(ii) Creation of an array using built-in NumPy functions**

# Array of ones ones = np.ones((3 ,4))

print( ones)

# Array of zeros zeros = np.zeros((2 ,3,4),dtype=np.int16)

print(zeros)

# Array with random values np.random.random((2,2))

# Empty array emptyArray = np.empty((3 ,2))

print( emptyArray)

# Full array fullArray = np.full((2 ,2),7)

print(fullArray)

# Array of evenly-spaced values

evenSpacedArray = np .arange( l0,25,5) print( evenSpacedArray)

# Array of evenly-spaced values evenSpacedArray2 = np.linspace(0 ,2,9) print( evenSpacedArray2)

**(iii) Performing file operations with NumPy arrays** import numpy as np

#initialize an array

arr = np.array([[[ll , 11, 9, 9], [11, 0, 2, O]], [[10, 14, 9, 14], [O, **1,** 11, 11]]])

# open a binary file in write mode file = open("arr", "wb")

# save array to the file np .save(file, arr)

# close the file file.close

# open the file in read binary mode

file = open("arr", "rb")

#read the file to numpy array

arr l = np.load(file) #close the file print( arr1) OUTPUT:

(i) Creation of different types of Numpy arrays and displaying basic information

[ 1 8 27 64]

[[ 1 2 3 4]

[ 2 4 9 16]

[ 4 8 18 32]]

[[[ 1 2 3 4]

[ 5 6 7 8]]

[[ 1 2 3 4] [ 9 10 11 12]]]

<memory at Ox0000024 7AE2AOAOO >

(3, 4)

int32

(16, 4)

(ii) Creation of an array using built-in NumPy functions

[[l. 1. 1. l.]

[l. 1. 1. l.]

[1. 1. 1. 1.]]

[[[O 00 O]

[O 00O]

[O 00 O]]

O 00OJ



[O 0 0 OJ

[O 0 0 OJ]]

[[O. O.]

[O. O.] [O. O.]]

[[7 7]

[7 7]]

[10 15 20]

[O. 0.25 0.5 0.75 1. 1.25 1.5 1.75 2. ]

(iii) Performing ftle operations with NumPy arrays

[[[11 11 9 9]

[11 0 2 OJ]

[[10 14 9 14]

[ O 1 11 11]]]

**2.b BASIC ARITHMETIC OPERATIONS WITH NUMPY ARRAYS**

**PROGRAM:**

import numpy as np

a = np .arange(9 , dtype = np.float\_).reshape(3 ,3)

print ('First array:') print (a) print ('\n')print ('Second array:') b = np .array([l0,10, 10])

print (b ) print ('\n')print ('Add the two arrays:') print (np.add(a ,b)) print ('\n')print ('Subtract the two arrays:') print (np.subtract(a ,b))

print ('\n')

print ('Multiply the two arrays:') print (np.multiply(a ,b)) print ('\n')

print ('Divide the two arrays:') print (np.divide(a,b))

**OUTPUT:**

First array :

[[ 0. 1. 2.] [ 3. 4. 5.]

[ 6. 7. 8.]]

Second array: [10 10 10]

Add the two arrays:

[[ 10. 11. 12.]

[ 13. 14. 15.]

[ 16. 17. 18.]]

Subtract the two arrays:

[[-10. -9. -8.]

[ -7. -6. -5 .] [ -4. -3. -2.]]

Multiply the two arrays:

[[ 0. 10. 20.]

[ 30. 40. 50.]

[ 60. 70 . 80.]]

Divide the two arrays:

[[ 0. 0.1 0.2]

[ 0.3 0.4 0.5]

[ 0.6 0.7 0.8]]

**EX. N0.:3 WORKING WITH PANDAS DATAFRAMES PROGRAM:**

**(i) CREATION OF A DATAFRAME FROM A SERIES** import numpy as np import pandas as pd

print("Pandas Version :", pd .\_version\_) pd.set\_option('display.max\_columns', 500) pd.set\_option('display.max\_rows' , 500)

series = pd.Series([2 , 3, 7, 11, 13, 17, 19, 23])

print( series)

series\_df = pd .DataFrame( {

'A': range(l , 5),

'B': pd.Timestamp('20190526') ,

'C': pd .Series(5 , index=list(range(4)) , dtype='float64') ,

'D': np.array([3] \* 4, dtype='int64'),

'E': pd.Categorical(["Depression" , "Social Anxiety", "Bipolar Disorder", "Eating Disorder"]),

'F': 'Mental health', 'G': 'is challenging'

})

print( series\_df)

1. **CREATION OF A DATAFRAME FROM DICTIONARY** import numpy as np import pandas as pd

dict\_df = [{'A': 'Apple', 'B': 'Ball'},{'A': 'Aeroplane', 'B':'Bat', 'C': 'Cat'}] diet\_df = pd .DataFrame( diet\_df)

print( diet\_df)

1. **CREATION OF A DATAFRAME FROM N-DIMENSIONAL ARRAYS** import numpy as np import pandas as pd

sdf = {'County':['Ostfold', 'Hordaland' , 'Oslo', 'Hedmark', 'Oppland', 'Buskemd'] ,

'ISO-Code' :[l,2,3,4,5,6],

'Area': [4180.69, 4917.94 , 454.07 , 27397.76 , 25192 .10, 14910.94],

'Administrative centre' : ["Sarpsborg", "Oslo" "City of Oslo" "Hamar",

' '

"Lillehammer", "Drammen"]} sdf = pd.DataFrame(sdf) print(sdf)

1. **LOADING A DATASET FROM AN EXTERNAL SOURCE INTO A PANDASDATAFRAME**

import numpy as np import pandas as pd

columns=['age' , 'workclass', 'fnlwgt', 'education', 'education\_num' , 'marital\_status' , 'occupation', 'relationship', 'ethnicity' , 'gender', 'capital\_gain' , 'capital\_loss',

'hours\_per\_week', 'country\_of\_origin','income']

df=pd.read\_csv(['http ://archive.ics.uci.edu/ml/machine-leaming-](http://archive.ics.uci.edu/ml/machine-leaming-) databases/adult/adult.data' ,names=columns) df.head(lO)

OUTPUT:

* 1. Creation of a dataframe from a series Pandas Version : 1.3.4

* 1. 2
  2. 3
  3. 7
  4. 11
  5. 13
  6. 17
  7. 19
  8. 23

dtype: int64

A B C D E F G

* 1. 1 2019-05-26 5.0 3 Depression Mental health is challenging
  2. 2 2019-05-26 5.0 3 Social Anxiety Mental health is challenging
  3. 3 2019-05-26 5.0 3 Bipolar Disorder Mental health is challenging
  4. 4 2019-05-26 5.0 3 Eating Disorder Mental health is challenging

* 1. Creation of a dataframe from a dictionary

## A B c

1. Apple Ball NaN
2. Aeroplane Bat Cat

(iii) Creation of a dataframe from n-dimensional array

|  |  |  |  |
| --- | --- | --- | --- |
| County ISO-Code | | Area | Administrative centre |
|  |  |
| 0 | Ostfold | 1 4180.69 | Sarpsborg |
| 1 | Hordaland | 2 4917 .94 | Oslo |
| 2 | Oslo | 3 454.07 | City of Oslo |
| 3 | Hedmark | 4 27397.76 | Hamar |
| 4 | Oppland | 5 25192.10 | Lillehamme |
| 5 | Buskerud | 6 14910.94 | Drammen |
|  |  |  |  |

EX.N0.4. READING DATA FROM TEXT FILES, EXCEL AND THE WEB

**PROGRAM:**

**DATA INPUT AND OUTPUT**

This notebook is the reference code for getting input and output, pandas can read a variety of file types using its pd.read\_ methods. Let's take a look at the most common data types :

import numpy as np import pandas as pd **csv**

**CSV INPUT:** df = pd.read\_csv('exam ple') df

[**a b c** **d**](#_Toc81460)

[**0** 0 1 2 3](#_Toc81461)

[**1** 4 5 6 7](#_Toc81462)

[**2** 8 9 10 11](#_Toc81463)

[**3** 12 13 14 15](#_Toc81464)

**CSV OUTPUT:**

df.to\_csv('example',index=False)

**EXCEL**

Pandas can read and write excel files, keep in mind, this only imports data. Not formulas or images, having images or macros may cause this read\_excel method to crash.

**EXCEL INPUT :**

pd.read\_excel('Excel\_Sample.xlsx' ,sheetname='Sheet 1')

**a b c d**

# **0** 0 1 2 3

# **1** 4 5 6 7

# **2** 8 9 10 11

# 3 12 13 14 15

**EXCEL OUTPUT** :

df.to\_excel('Excel\_Sample.xlsx' ,sheet\_name='Sheet 1')

**HTML**

You may need to install htmllib5, lxml, and Beautifu1Soup4. In your terminal/command prompt run:

pip install lxml pip install html5lib== 1.1 pip install

Beautifu1Soup

4

Then restart Jupyter Notebook. (or use conda install) Pandas can read table tabs off of html.

For example:

**HTML INPUT**

Pandas read\_html function will read tables off of a webpage and return a list of DataFrame objects :

Downloaded by Jegatheeswari ic37721 (ic3f71@imail.iitm.ac .in)

url = https [://www.fdic.gov/resources/resolutions/bank- failures/failed-bank-list d](http://www.fdic.gov/resources/resolutions/bank-failures/failed-bank-list)f = pd.read\_html(url) df[O]

match = "Metcalf Bank" df\_list = pd.read\_html(url, match=match) df\_list[O]

HTML OUTPUT:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bank Name | City | ST CERT | Ac quiring Institution | Closing Date | Updated Date | Loss  Share  Type | Ag reement  Termi nated | Termination  Date |

First-Citizens Bank &

0 First Cornerstone Bank King of Prussia PA 35312 May 6, 2016 July 12, 20 16 none NaN NaN

Trust Company

The Bank of Fayette August 4,

Trust Company Bank Memphis TN 9956 Apnl 29, 2016 none NaN Na N

County 2016

1. North Milwaukee SBatatnek M ilwaukee WI 20364 F irst-CTitruizestn Cos Bmapnakn &y Marc2h0 1116, June 16, 2016 none NaN NaN

October 2 ,

Hometown National Bank Longview WA 35156 Twin City Bank April 13, 20 16 none NaN NaN

1. 2015

October 2 ,

The Bank of Georgia Peachtree City GA 35259 Fidelity Bank April 13, 20 16 none NaN NaN

1. 2015
2. Premier Bank Denver co 34 112 United Fidelity Bank, fsb July 10, 2015 July 12, 20 16 none NaN NaN

1. Edgebrook Bank Chicago I L 57772 Republic Bank of Chicago May B, 2015 July 12, 2016 none NaN NaN

Banco Popular de Puerto February 27 ,

1. Doral BankEn Espanol San Juan PR 32102 Ri cr1 -?01<; May 13, 2015 none NaN Na

**EX NO 4. EXPLORING VARIOUS COMMANDS FOR DOING DESCRIPTIVE ANALYTICS ON THE IRIS DATA SET.**

**PROGRAM:**

import pandas as pd

from pandas import DataFrame

from skleam.datasets import load\_iris

# skleam.datasetsincludes common example datasets

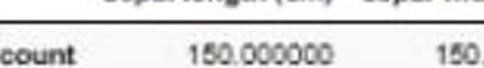
# A function to load in the iris dataset iris\_obj = load\_iris() # Dataset preview iris\_obj.data

iris = DataFrame(iris\_obj.data, columns=iris\_obj.feature\_names,index=pd.Index([i for i in range(iris\_obj.data.shape[O]) ])).join(DataFrame(iris\_obj .target, columns=pd.Index([" species"]), index=pd.Index([i for i in range(iris\_obj.target.shape[O])])))

iris # prints iris data

Commands

iris\_obj.feature\_names iris.count() iris.mean() iris.median() iris.var() iris.std() iris.max() iris.min() iris.describe() **OUTPUT:**



**5.A UNIVARIATE ANALYSIS: FREQUENCY, MEAN, MEDIAN, MODE, VARIANCE, STANDARD DEVIATION, SKEWNESS AND KURTOSIS PROGRAM:**

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns sns.set\_style('darkgrid') %matplotlib inline from matplotlib.ticker import FormatStrFormatter import warnings

warnings .filterwarnings('ignore')

df = pd.read\_csv('C:/Users/kirub/Documents/Learning/Untitled Folder/diabetes.csv') df.head() df.shape df.dtypes

df['Outcome']=df['Outcome'] .astype('bool') df.dtypes['Outcome']

df.info() df.describe().T

# Frequency# finding the unique count dfl = df['Outcome'].value\_counts()

# displaying dfl

print(dfl)

#mean df.mean() #median df.median()

#mode df.mode()

#Variance df.var()

#standard deviation df.std()

#

#kurtosis df.kurtosis( axis=O,skipna=True)

df['Outcome'].kurtosis(axis=O,skipna= True)

#skewness

# skewness along the index axis df.skew(axis = 0, skipna = True)

# skip the na values # find skewness in each row df.skew(axis = 1, skipna = True) #Pregnancy variable

preg\_proportion = np.array( df['Pregnancies']. value\_counts()) preg\_month = np.array( df['Pregnancies']. value\_counts().index) preg\_proportion\_perc =

np.array(np.round(preg\_proportion/ sum(preg\_proportion ),3)\*100,dtype=int)

preg =

pd.DataFrame( {'month':preg\_month,'count\_of\_preg\_prop' :preg\_proportion,'percentage\_pro portion':preg\_proportion\_perc}) preg.set\_index(['month'] ,inplace=True) preg.head( 10)

sns.countplot(data=df['Outcome']) sns.distplot(df['Pregnancies']) sns.boxplot( data=df['Pregnancies '])

OUTPUT:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| " |  | coun n st | lll1n 25% | 59 75X | x |
|  | agt | 1025 0 9072290 | 29.0 .0 | SG.O 61 0 | 77 0 |
|  | Hlt | 1 5.0 0.695610 0-160373 | 0.0 0.0 | 10 1D | 10 |
|  | cp | 1or D 9J2 9 1 .029 1 | 0 0 0 0 | 1 0 2 0 | JO |
|  | UISU!ps | 1 5.0 131.61 707 17.5 67 6 | 9 .0 0.0 | 130.0 140. | 00 0 |
|  | chol | 1oro 2..! 000000 515925 0 | 1260 211 0 | 24 00 275 0 | 5 0 |
|  | bs | 1 .O .1 l9 68 .3565 7 | 0.0 0.0 | 0. | 1 0 |
|  | restecg | 1025.0 0 529756 0.527878 | 0 0 0 0 | 1 0 1 0 | 2 0 |
|  | lhlilch | 1or o 1-1'3 1 -11 6 23 00572 | 7 0 132 0 | 1570  166 0 | °" 0 |
|  | txng | 1025.0 0.336585 0 472m | 0.0 0.0 | 0.0 1 0 | 10 |
|  | oldpea | 1or o 0 1512 1 1ro-3 | 0 0 0 0 | 08 1 8 | ? |
|  | slope | .0 .385366o. 1nss | 0.0 1.0 | 1.0 0 | 0 |
|  | ca | 0 0 1-1 1 0 7 s | 0 0 0 0 | 0 0 1 0 | 0 |
|  | tnal | 1DT D 2.323902 0 620660 | 0 0 2.0 | 2 0 30 | JO |

**EX.NO:5. B) BIVARIATE ANALYSIS: LINEAR AND LOGISTIC REGRESSION**

**MODELING**

**PROGRAM:**

**BIVARIATE ANALYSIS GENERAL PROGRAM**

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns sns.set\_style('darkgrid') %matplotlib inline from matplotlib.ticker import FormatStrFormatter import warnings warnings .filterwarnings('ignore') df = pd.read\_csv('C:/Users/diabetes.csv') df.head() df.shape df.dtypes df['Outcome']=df['Outcome'] .astype('bool') fig,axes = plt.subplots(nrows=3,ncols=2,dpi= l20,figsize = (8,6)) plotOO=sns .countplot('Pregnancies' ,data=df,ax=axes [0][0],color='green')

axes[O][0].set\_title('Count',fontdict= {'fontsize':8}) axes[O] [0].set\_xlabel('Month of Preg.',fontdict= {'fontsize' :7}) axes[O] [0].set\_ylabel('Count',fontdict={ 'fontsize':7})

plt.tight\_layout()

plotO 1=sns.countplot('Pregnancies' ,data=df,hue='Outcome' ,ax=axes [0][1])

axes[O][l].set\_title('Diab. VS Non-Diab.',fontdict={ 'fontsize':8})

axes[O][ 1].set\_xlabel('Month of Preg.',fontdict= {'fontsize' :7})

axes[O] [l].set\_ylabel('Count',fontdict={ 'fontsize':7})

plotO l .axes.legend(loc= 1)

plt.setp( axes[O] [1].get\_legend() .get\_texts(), fontsize='6')

plt.setp( axes[O] [1].get\_legend().get\_title(), fontsize='6') plt.tight\_layout()

plot 10 = sns.distplot( df['Pregnancies '],ax=axes[ 1][0])

axes[ 1][0].set\_title('Pregnancies Distribution' ,fontdict= {'fontsize': 8})

axes[l] [0].set\_xlabel('Pregnancy Class',fontdict={ 'fontsize':7}) axes[l] [0].set\_ylabel('Freq/Dist',fontdict={ 'fontsize':7}) plt.tight\_layout()

plot 11 = df[df['Outcome']==False] ['Pregnancies'] .plot.hist(ax=axes[ l][l],label='Non• Diab.') plot l 1\_2=df[df['Outcome']==True] ['Pregnancies'] .plot.hist(ax=axes[ l][l],label='Diab. ')

axes[l][l] .set\_title('Diab. VS Non-Diab.',fontdict={ 'fontsize':8}) axes[l] [l].set\_xlabel('Pregnancy Class',fontdict={ 'fontsize':7}) axes[l] [l].set\_ylabel('Freq/Dist',fontdict={ 'fontsize':7})

plot 11.axes.legend(loc= 1) plt.setp(axes[l][ l].get\_legend().get\_texts(), fontsize='6') # for legend text plt.setp(axes[l][ l].get\_legend().get\_title(), fontsize='6') # for legend title plt.tight\_layout()

plot20 = sns.boxplot(df['Pregnancies'] ,ax=axes [2][0],orient='v')

axes[2][0] .set\_title('Pregnancies' ,fontdict={ 'fontsize':8}) axes[2] [0].set\_xlabel('Pregnancy' ,fontdict={ 'fontsize':7})

axes[2][0].set\_ylabel('Five Point Summary',fontdict={ 'fontsize':7}) plt.tight\_layout()

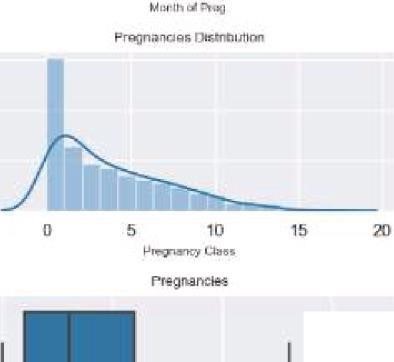
plot2 l = sns.boxplot(x='Outcome' ,y='Pregnancies' ,data=df,ax=axes [2][1])

axes[2][1].set\_title('Diab. VS Non-Diab.',fontdict={ 'fontsize':8}) axes[2] [l].set\_xlabel('Pregnancy' ,fontdict={ 'fontsize':7})

axes[2][1].set\_ylabel('Five Point Summary',fontdict={ 'fontsize' :7}) plt.xticks( ticks=[O, 1],labels=['Non-Diab. ','Diab.'],fontsize=7) plt.tight\_layout() plt.show()

**OUTPUT:**

COun< **D1U:!J. VS NcnDmb** -



l

'

**11111**

**......\_**

**\_**

**\_**

1

00

,..

.....

l

ru•

0

1

2

:>

4

5

15

,

*1*

8

!

i

10

1

1

12

1

14

15

1

7

0

1

2

3

*A*

5

6

7

a

9

1

0

1

1

12

13

14

1

5

17

**Pi.bri**

**lh**

**of**

**P**

**ru!J**

15

-

0

*B*

100

0

i

1

5

i

•

ii

,

O

'

5

J

0

0

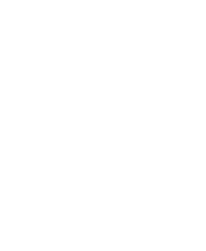
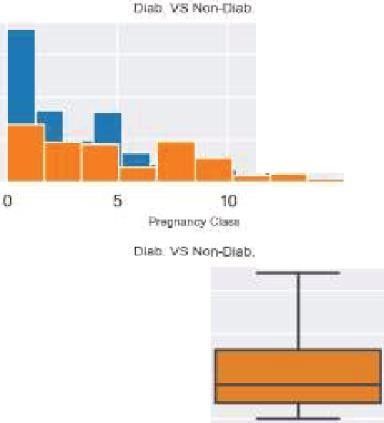
5

10

1

5

**Ff"eQncr:i::y**



### LINEAR REGRESSION MODELLING ON HOUSING DATASET

# Data manipulation libraries

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns

%matplotlib inline

USAhousing = pd.read\_csv('USA\_Housing.csv')

USAhousing.head()

USAhousing.info()

USAhousing.describe()

USAhousing.colUinns sns.pairplot(USAhousing)

sns.distplot(USAhousing['Price']) sns.heatmap(USAhousing.corr())

X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',

'Avg. Area Number of Bedrooms', 'Area Population']] y = USAhousing['Price'] from skleam.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state= lOl) from skleam.linear\_model import LinearRegression lm = LinearRegression() lm.fit(X\_train,y\_train) # print the intercept print(lm.intercept\_)

coeff\_df = pd.DataFrame(lm.coef\_,X.columns,columns=['Coefficient']) coeff\_df

predictions = lm.predict(X\_test) plt.scatter(y \_test,predictions) sns.distplot((y\_test-predictions ),bins=50);

from skleam import metrics

print('MAE:', metrics.mean\_absolute\_error(y \_test, predictions)) print('MSE: ', metrics.mean\_squared\_error(y \_test, predictions))

print('RMSE:', np.sqrt(metrics.mean\_sq uared\_error(y\_test, predictions)))

MAE: 82288.2225191

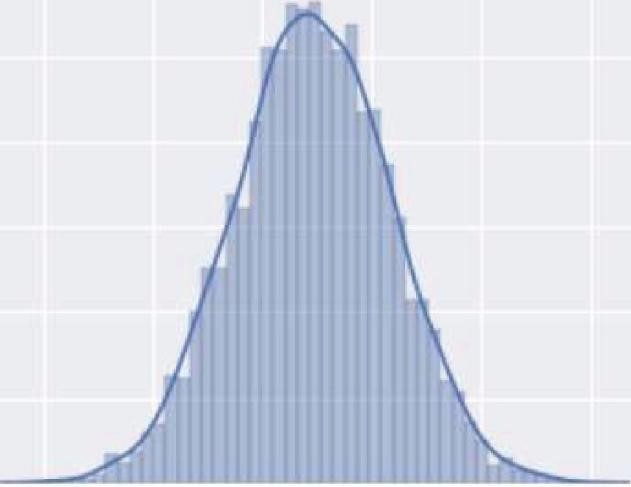
MSE: 10460958907.2

RMSE : 102278.829223

OUTPUT:

00000012

-500000c moooo 1000000 1000000 21!lOOOOO 2500000 3XlOOOO



00000010

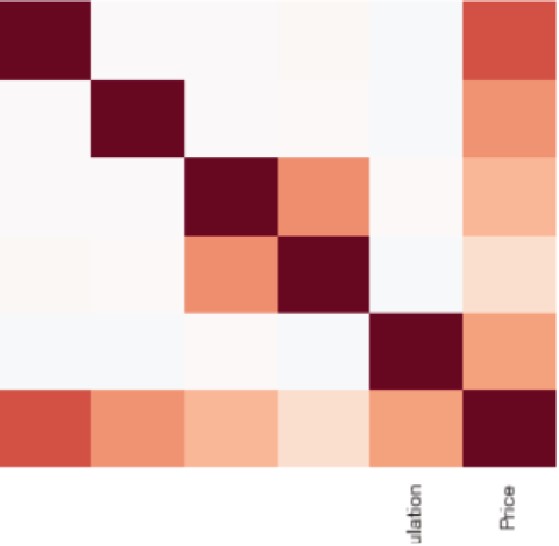
0:0000006

Ol0000002

00000000



Price



0::



Avg.

A

r

e

n

come

/lo/g

.

Area

Number

of

Room!I

/Wg

.

Area

N

mbe1

o

B

oons

A

-

ea

Populabon

0.4

0.0

-

l

<

.

4

Pnce

-

(

I

!!

«(

Ill

lli

li

m



di o!(I!! 1Ill jI

.f z

I!! zE

*1*

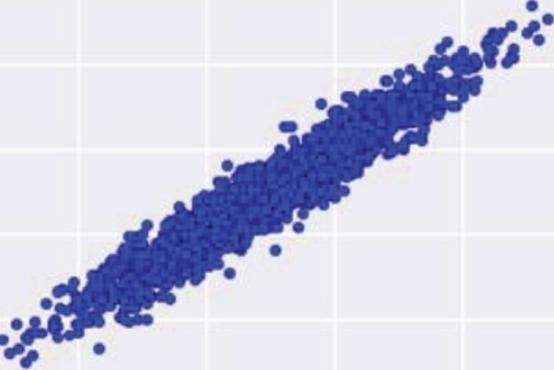
o!(*1* JCD

*$*

Coefficiient.

|  |  |
| --- | --- |
| Avg.AreaIncome | 21.528276 |
| Avg. A1rea House Age | 164883.282027 |
| Avg..A rea Number of Rooms | 122368.1678027 |
| Avg .Area INumber of Bedrooms | 2233.801864 |
| Area Po1 pula1ii on | 15.150420 |

2500000



Z>OOOOO

1500000

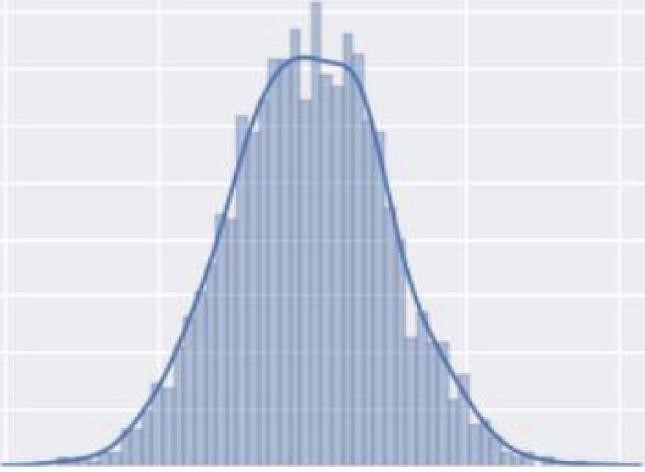
1000000

5JOOOO

### 0 •••••

-500000 0 !iXlOOO 1000000 1500000 Z>OOOOO 2500000

01 0000015



00000040

0

L

00000l5

0

.

0000030

0

.

0000025

0l0000020

0

0000015

0

L

0000010

ClLOOOOOD!i

0

.

0000000

--

400000

-

200000

0

3)(1000

'300000

Price

LOGISTIC REGRESSION MODELLING ON PIME DIABETIES

# Data manipulation libraries import numpy as np import pandas as pd

#scikit Learn Modules needed for Logistic Regression from skleam.linear\_model import LogisticRegression from skleam.model\_selection import train\_test\_split,GridSearchCV from skleam.preprocessing import

LabelEncoder ,MinMaxScaler, OneHotEncoder ,StandardScaler from skleam.metrics import confusion\_matrix from skleam.impute import Simplelmputer from skleam.pipeline import Pipeline

from skleam.compose import ColumnTransformer

#for plotting import matplotlib.pyplot as plt import seaborn as sns %matplotlib inline sns.set(color\_codes=True) import warnings warnings .filterwarnings('ignore') df=pd.read\_csv('C:/Users/diabetes.csv') df.head()

df.tail()

df.isnull().sum() df.describe(include='all') df.corr() sns.heatmap(df.corr(),annot= True)

plt.show() df.hist() plt.show()

sns.countplot(x=df['Outcome']) scaler=StandardScaler() df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age']]=scaler.fit\_transform( df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age']]) df\_new = df

# Train & Test split x\_train, x\_test, y\_train, y\_test = train\_test\_split( df\_new[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',

'BMI', 'DiabetesPedigreeFunction', 'Age']],

df\_new['Outcome'],test\_size=0.20,

random\_state=21)

print(' Shape of Training Xs:{ }'.format(x\_train.shape)) print('Shape of Test Xs:{}'.format(x\_test.shape)) print(' Shape of Training y:{}'.format(y\_train.shape)) print(' Shape of Test y: { }'.format(y\_test.shape))

|  |  |  |
| --- | --- | --- |
| Shape of | T r aining X s : ( 6 14 , | 8 ) |
| Shape of  Shape of | T est X s : ( 154 , 8)  T r a ining y : ( 6 14 , ) |  |
| Shape of | T est y : ( 154 , ) |  |

# Build Model model = LogisticRegression() model.fit(x\_train, y\_train) y\_predicted = model.predict(x\_test)

score=model.score(x\_test,y \_test);

print( score)

0 . 7337 662 337 6 62 337

#Confusion Matrix # Compute confusion matrix cnf\_matrix = confusion\_matrix(y \_test, y\_predicted) np.set\_printoptions(precision=2)

cnf\_matri

**OUTPUT:**

Pregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeF unction Age Outcome

0 6 148 72 35 0 33\_ 6 0.627 50

85 66 29 0 26\_ 6 0. 351 310

1. 8 183 64 0 0 23\_3 0.672 32

1. 89 66 23 94 28\_ 1 0 167 21 0 **4** 0 137 40 35 168 43\_ 1 2.288 33

Piregnancies Glucose BloodPressure SkinThickness Insulin BMI DiabetesPedigreeFunct ion Age Outcome

1. 10 101 76 48 180 32\_ 9 0.171 63 0
2. 2 122 70 27 0 36\_ 8 0.340 27 0
3. 5 121 72 23 112 26\_ 2 0.245 30 0
4. 126 60 0 0 30\_1 0.349 47

767 93 70 31 0 30\_4 0.315 230

Pregnancies 0

Glucose 0

BloodPressure 0 skinThickness 0

Insu lin 0

BM I 0

DiabetesPedigreeF unction 0

Age 0 outcome 0 dltype : int64

Pregnancies Glucose BloodPressure Sk inThickness Insulin BMI DiabetesPedigreeFunctio n A ge Outcome

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 768.000000 768.000000 | 768.000000 | 768.000000 768.000000 | 768.000000 |  | 768 .000000 | 768.000000 768.000000 | |
| mean | 3.845052 120.894531 | 69.105469 | 20.536458 79.799479 | 31\_ 992578 |  | 0.471876 | 33.240885 | 0. 348958 |
| std | 3.369578 31.972618 | 19.355807 | 15.952218 115.244002 | 7-884160 |  | 0.33 1329 | 11.760232 | 0.476951 |
| **min** | 0 000000 0 000000 | 0 000000 | 0 000000 0 000000 | 0\_ 000000 |  | 0.078000 | 21 000000 | 0 000000 |
| 25% | 1.000000 99.000000 | 62 .000000 | 0 000000 0 000000 | 27-300000 |  | 0.243750 | 24 .000000 | 0 000000 |
| 50 % | 3 000000 117000000 | 72 000000 | 23 000000 30 500000 | 32 000000 |  | 0 372500 | 29 000000 | 0 000000 |
| 75% | 6 000000 140 250000 | 80 000000 | 32 000000 127 250000 | 36 600000 |  | 0 626250 | 41 000000 | 1 000000 |
| max | 17.000000 199.000000 | 122 .000000 | 99.000000 846.000000 | 67\_ 100000 |  | 2.420000 | 81.000000 | 1.000000 |
|  | **Pregnancies** | **Glucose** | **BloodPressure Sk inThickness** | **Insulin** | **BMI** | **DiabetesPedig reeFunction Age** | | **Outcome** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Pregnancies** | 1.000000 0. 129459 | 0.14 1282 | -0.081672 | -0 . 073535 0.017683 | -0 033523 0.54434 1 0-221898 |
| Glucose | 0. 129459 1.000000 | 0.152590 | 0.057328 | 0. 331357 0.221071 | 0. 137337 0.263514 0-466581 |
| **Blood Pressure** | 0. 141282 0. 152590 | 1 .000000 | 0.20737 1 | 0 . 088933 0.281805 | 0.04 1265 0.239528 0\_ 065068 |
| **SkinThickness** | -0 081672 0057328 | 0 207371 | 1 000000 | 0 436783 0 392573 | 0 183928 -0 113970 0 074 752 |
| **Insulin** | -0.073535 0.331357 | 0.088933 | 0.436783 | 1. 000000 0.197859 | 0. 185071 -0 .042 163 0\_ 130548 |
| BMI | 0 017683 0221071 | 0 281805 | 0 392573 | 0 197859 1 000000 | 0 140647 0 036242 0292695 |
| DiabetesPedigreeFunction | -0.033523 0. 137337 | 0.041265 | 0.183928 | 0 . 185071 0.140647 | 1.000000 0.033561 0\_ 173844 |
| Age | 0 544341 0263514 | 0 239528 | -0 113970 -0 042163 0 036242 | | 0 033561 1 000000 0238356 |
| **Outcome** | 0.221898 0.466581 | 0.065068 | 0.074752 0 . 130548 0.292695 | | 0. 173844 0.238356 1-000000 |

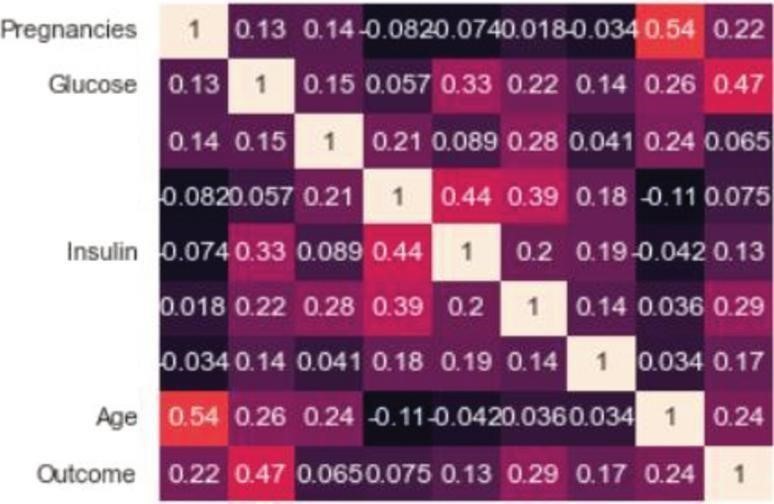
-10

-0.8

0.6

0.4

DlabetesPedigreefunction 02



BloodPressure

SkmThickness

BMI

00

u :e:I I '5 ti *If*I) E c

m*l*c l§:!:lI a*e*.. :"c" 5 c;::I *a*8I)

u..

I- II!)!

Ci:! Cl

*:i*

1mi

Pregnancies

Glucose

BloodPrressure

BM

I

100

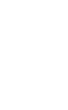
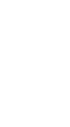
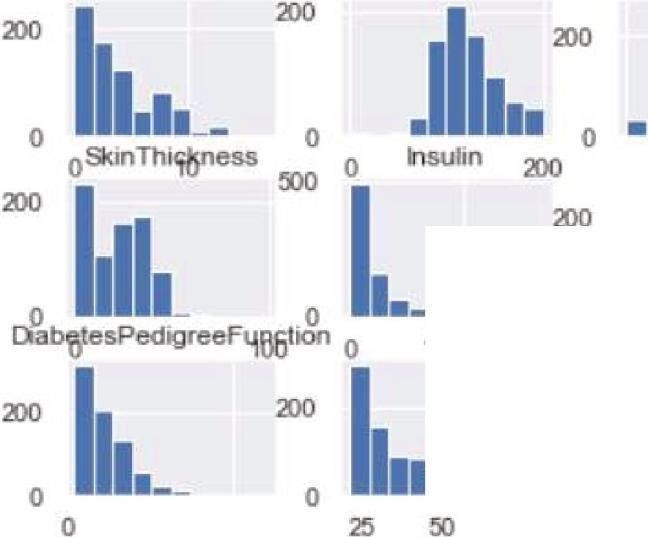
I.

\_

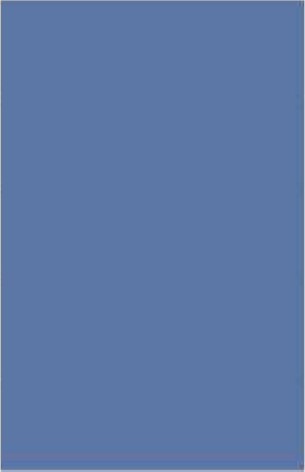
tco

().1

I



500



0

200

0

0

**MULTIPLE REGRESSION ANALYSIS**

**PROGRAM:**

# Data manipulation libraries

import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns %matplotlib inline

USAhousing = pd.read\_csv('USA\_Housing.csv')

USAhousing.head()

USAhousing.info()

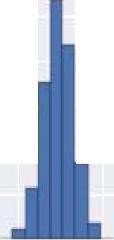
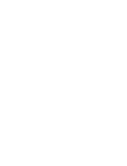
USAhousing.describe()

USAhousing.columns sns.pairplot(USAhousing)

**OUTPUT:**

110000

100000 u E .85 lllOOO



;

* lllOOO

<(

2llOOO 40000

10

..

<(

I0,  

* ;

...

*}*

'

.

***,:a***

"·

.

..

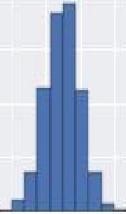


<(

f

* 12 E

0 10



! Q

• , z

•

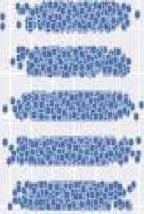
<(

r

"r•o



~~;~~



0

E 4 f 1

•

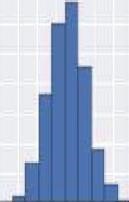
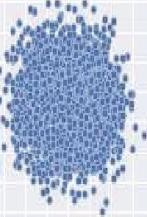
2

1

<(

lllOOO

70000 < lllOOO



0

: !ilOOO i0 40000

11. 31000

2llOOO

10000

J

-10000

3100000 2i00000

2000000

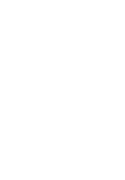
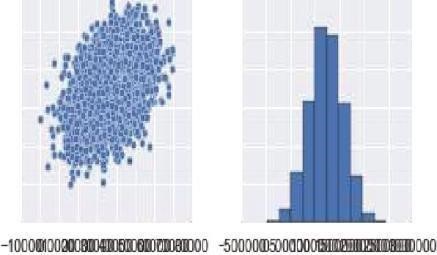
15110000

"

1000000

!'00000

0 2llOOlJXJOIKIJOIEOOlllJOO\IOOOO 1 4 5 6 *1* B 9 10 2 4 6 B 10 12 1 2 1 4 5 6 *1*



Avg Area lnoome Avg. Area House Age Avg Area Number of Rooms Avg. Area Number of Bedrooms Area Population Prire

**EX.NO:5.D) ALSO COMPARE THE RESULTS OF THE ABOVE ANALYSIS FOR THE**

**TWO DATA SETS.**

**PROGRAM:**

# Glucose Variable df.Glucose.describe()

#sns.set\_style('darkgrid')

fig,axes = plt.subplots(nrows=2,ncols=2,dpi=120,figsize = (8,6))

plotOO=sns.distplot( df['Glucose'],ax=axes[0][0],color='green') axes[O] [0].yaxis.set\_major\_formatter(FormatStrFormatter('% .3f )) axes[O] [0].set\_title('Distribution of Glucose',fontdict= {'fontsize' :8}) axes[O] [0].set\_xlabel('Glucose Class',fontdict= {'fontsize' :7}) axes[O] [0].set\_ylabel('Count/Dist. ',fontdict= {'fontsize' :7}) plt.tight\_layout()

plotO l=sns.distplot(df[df['Outcome']==False]['Glucose'],ax=axes[O][l],color='green',label=' Non Diab.') sns.distplot(df[df.Outcome==True]['Glucose'],ax=axes[O] [1],color='red',label='Diab')

axes[O] [1].set\_title('Distribution of Glucose',fontdict= {'fontsize' :8}) axes[O] [1].set\_xlabel('Glucose Class',fontdict= {'fontsize' :7})

axes[O] [1].set\_ylabel('Count/Dist.',fontdict= {'fontsize' :7}) axes[O][l].yaxis.set\_major\_formatter(FormatStrFormatter('%.3f )) plotO 1.axes.legend(loc= 1) plt.setp(axes[O] [1].get\_legend().get\_texts(), fontsize='6') plt.setp( axes[O] [1].get\_legend() .get\_title(), fontsize='6') plt.tight\_layout()

plot lO=sns.boxplot(df['Glucose'],ax=axes[l][O],orient='v')

axes[l][O].set\_title('Numerical Summary',fontdict={ 'fontsize' :8}) axes[ 1][0].set\_xlabel('Glucose',fontdict= {'fontsize' :7}) axes[l] [0].set\_ylabel(r'Five Point Summary(Glucose)',fontdict={ 'fontsize':7})

plt.tight\_layout()

plot 11=sns.boxplot(x='Outcome' ,y='Glucose',data=df,ax=axes [1][1]) axes[l][l].set\_title(r'Numerical Summary (Outcome)',fontdict={ 'fontsize':8}) axes[l] [1].set\_ylabel(r'Five Point Summary(Glucose)',fontdict={ 'fontsize':7})

plt.xticks( ticks=[O, 1],labels=['Non-Diab. ','Diab.'],fontsize=7)

axes[l] [1].set\_xlabel('Category',fontdict={ 'fontsize':7})

plt.tight\_layout() plt.show() fig,axes = plt.subplots(nrows=l,ncols=2,dpi=120,figsize = (8,4))

plotO=sns.distplot( df[df['Glucose']!=0]['Glucose'],ax=axes[O],color='green')

axes[O].yaxis.set\_major\_formatter(FormatStrFormatter('%.3f )) axes[O] .set\_title('Distribution of Glucose',fontdict= {'fontsize': 8})

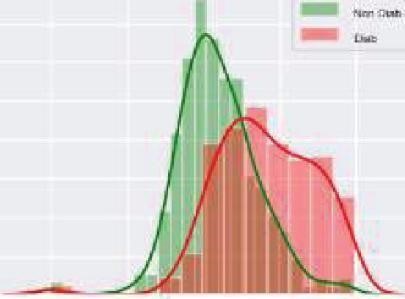
axes[O].set\_xlabel('Glucose Class',fontdict= {'fontsize' :7}) axes[O].set\_ylabel('Count/Dist.',fontdict= {'fontsize' :7}) plt.tight\_layout()

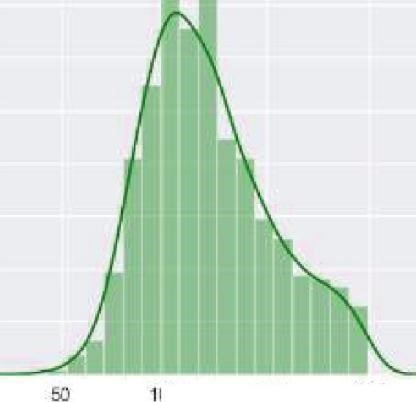
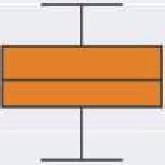
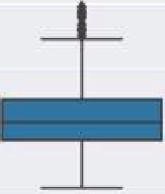
plot 1=sns.boxplot(df[df['Glucose'] !=0]['Glucose'],ax=axes[ 1],orient='v')

axes[1].set\_title('Numerical Summary',fontdict= {'fontsize': 8 }) axes[ 1].set\_xlabel('Glucose' ,fontdict= {'fontsize' :7})

axes[l].set\_ylabel(r'Five Point Summary(Glucose)',fontdict={ 'fontsize' :7}) plt.tight\_layout()

**OUTPUT:**





**EX.N0:6.**

**APPLY AND EXPLORE VARIOUS PLOTTING FUNCTIONS ON UCI DATA SETS.**

**PROGRAM:**

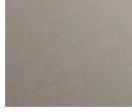
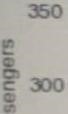
**A. NORMAL CURVES**

#seaborn package import seaborn as sns

flights = sns.load\_dataset("flights ") flights.head() may\_flights = flights.query("month == 'May"')

sns.lineplot(data=may\_flights, x="year", y="passengers")

**OUTPUT:**



**200**



1

1

'



#### B. DENSITY AND CONTOUR PLOTS

iris = sns.load\_dataset("iris") sns.kdeplot(data=iris)

**OUTPUT:**



025

* **sepal\_length**
* **pal\_w1dth**

##### -pelal\_length



020

015

-

**p**

**e**

**t**

**a**

**l**

**\_w•**

**d**

**t**

**h**

010

I

#### C.CORRELATION AND SCATTER PLOTS

#correlation visualized using heatmap function df = sns.load\_dataset("titanic") ax = sns.heatmap(df annot=True, fmt="d")

#scatter plots of categorical variable df = sns.load\_dataset("titanic")

sns.catplot(data=df, x="age", y="class")

**OUTPUT:**

**ValueEr ror: Unknown format code 'd ' for object o type oat**

* 1.0 - 1.0

* 0.8 - 0.8

pclass

0.6 0.6

age

0.4 0.4

sobsp

0.20.2

parch

0.0 0.0

fare

-0.2 -0.2

adult\_male

E' 0. **dlone**

I i7i

" "'' *L* 1' "'"' c"'

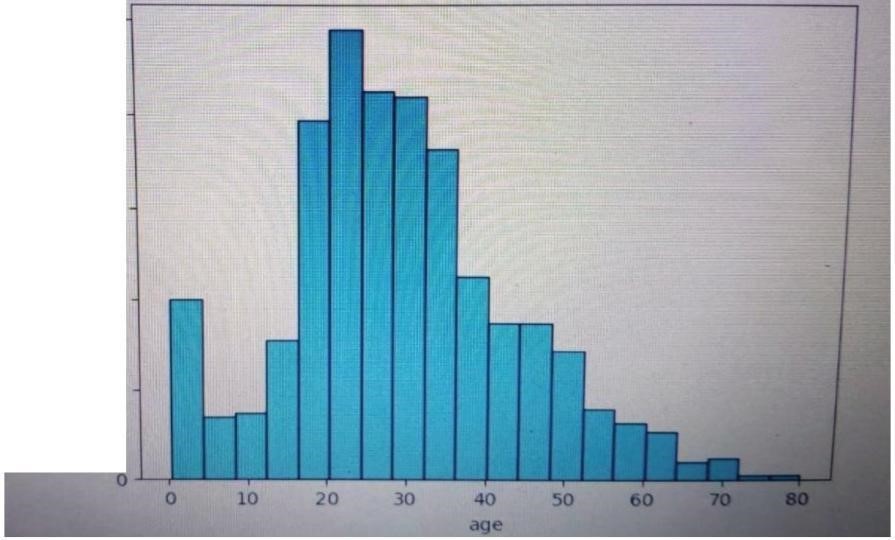
*<*L.'.*A* "' .c t' Jl!

##### D.HISTOGRAMS

#histogram of datafra,e

df = sns.load\_dataset("titanic") sns.histplot(data=df, x="age")

**OUTPUT:**



20

##### E.THREE DIMENSIONAL PLOTTING

#3d plotting using ploty package import plotly as px

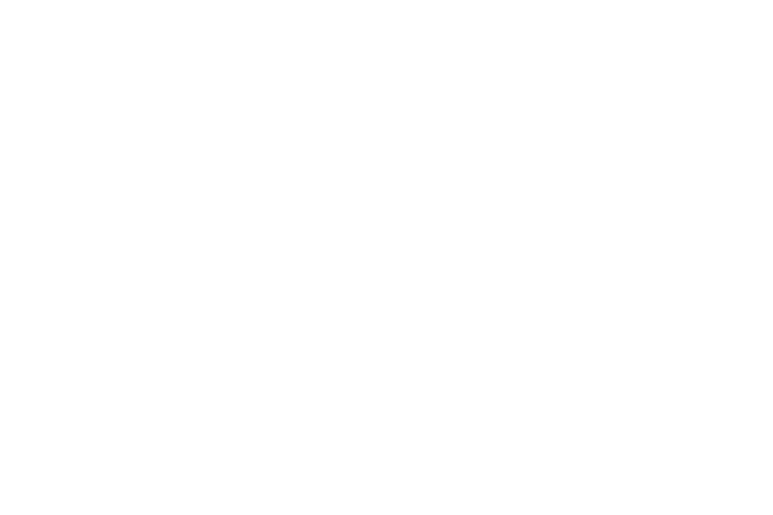
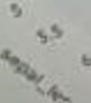
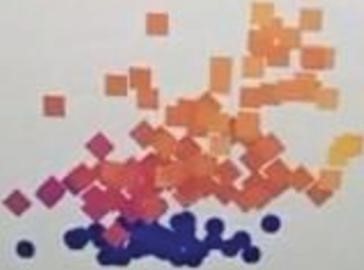
df = sns.load\_dataset("iris")

px.scatter\_3d( df, x="PetalLengthCm", y="PetalWidthCm", z="SepalWidthCm",

size="SepalLengthCm",

color="Species", color\_discrete\_map = {"Joly": "blue", "Bergeron": "violet", "Coderre":"pink"})

**OUTPUT:**



2

.'S

EX.N0:7. VISUALIZING GEOGRAPHIC DATA WITH BASEMAP

**PROGRAM:**

%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.basemap

import Basemap

plt.figure(figsize=(8, 8)) m = Basemap(projection='ortho', resolution=None, lat\_0=50, lon\_0=-100) m.bluemarble(scale=0.5);

**OUTPUT:**



fig = plt.figure(figsize=(8, 8))

m = Basemap(projection='lcc', resolution=None, width=8E6, height=8E6, lat\_0=45, lon\_0=-100,)

m.etopo(scale=0.5, alpha=0.5)

# Map (long, lat) to (x, y) for plotting x, y

= m(-122.3, 47.6) plt.plot(x, y, 'ok', markersize=5) plt.text(x, y, ' Seattle', fontsize= l2);

**OUTPUT:**

