CS3362 DATA SCIENCE LABORATORY

LIST OF EXPERIMENTS

NAME: **REG.NO:**

EXP. NO.	DATE	NAME OF THE EXPERIMENT	MARKS	SIGN
1		Download, install and explore the features of NumPy, SciPy, Jupyter, Statsmodels and Pandas packages.		
2		Working with Numpy arrays		
3		Working with Pandas data frames		
4		Reading data from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set.		
5		Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following: a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis. b. Bivariate analysis: Linear and logistic regression modeling c. Multiple Regression analysis d. Also compare the results of the above analysis for the two data sets.		
6		Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following: a. Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis. b. Bivariate analysis: Linear and logistic regression modeling c. Multiple Regression analysis d. Also compare the results of the above analysis for the two data sets.		
7		Visualizing Geographic Data with Basemap		

EX NO: **Download, install and explore the features of NumPy, SciPy, Jupyter, Statsmodels and Pandas packages**

DATE:

Aim:

To Verify the phyton software download,install and explore the features of numpy,scipy,jupyter ,stats models and pandas packages

Algorithm:

```
Step 1: Start.
Step 2: Select version of phyton to install
Step 3: Download phyton executable installer
Step 4: Run excecutable installer
Step 5: verfy phyton was installed on Windows
Step 6: verify pip was installed
Step 7: Add phyton path to environment variables
Step 8: install virtual nv
Step 9: install all the packages through pip
Step 10: then install the jupyeter notebook,py pip install jupyter noyebook
Step 11: Stop.
```

Python is a high-level and general-purpose programming language with data science and machine learning packages. Use the video below to install on Windows, MacOS, or Linux. As a first step, install Python for Windows, MacOS, or Linux**.**

Install Python Packages

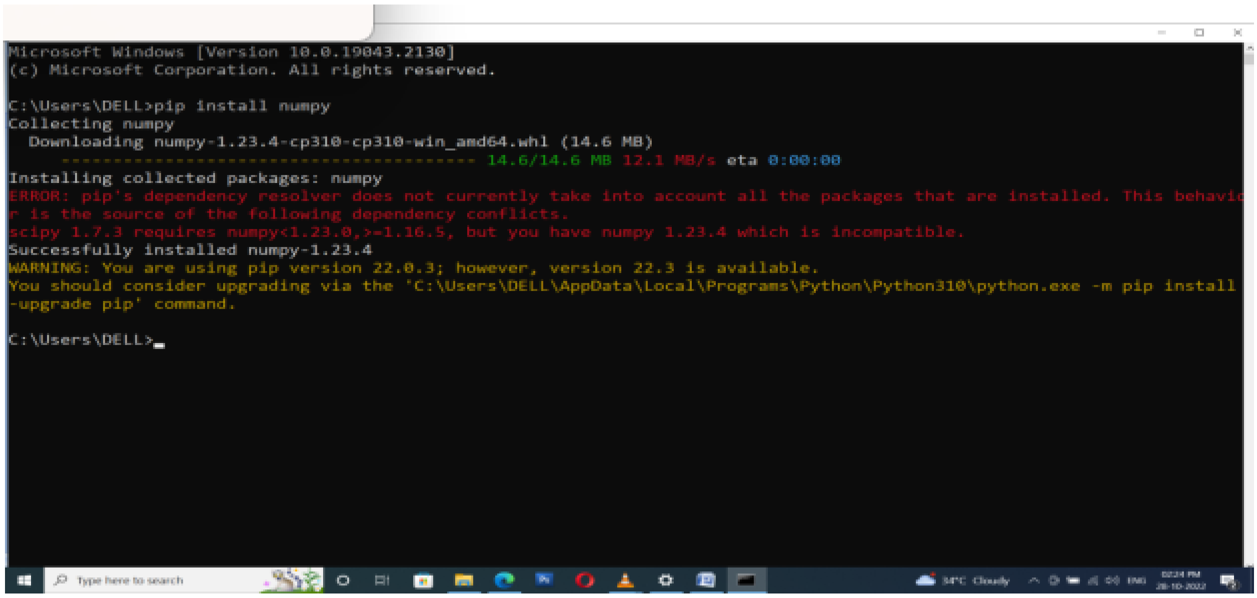
The power of Python is in the packages that are available either through the pip or conda package managers. This page is an overview of some of the best packages for machine learning and data science and how to install them. We will explore the Python packages that are commonly used for data science and machine learning. You may need to install the packages from the terminal, Anaconda prompt, command prompt, or from the Jupyter Notebook. If you have multiple versions of Python or have specific dependencies then use an environment manager such as pyenv. For most users, a single installation is typically sufficient. The Python package manager pip has all of the packages (such as NumPy, SciPy) that we need for this course. If there is an administrative access error, install to the local profile with the --user flag**.**

Install Method

Numpy

Numpy is a numerical computing package for mathematics, science, and engineering. Many data science packages use Numpy as a dependency. Ex : pip install NumPy

Output:



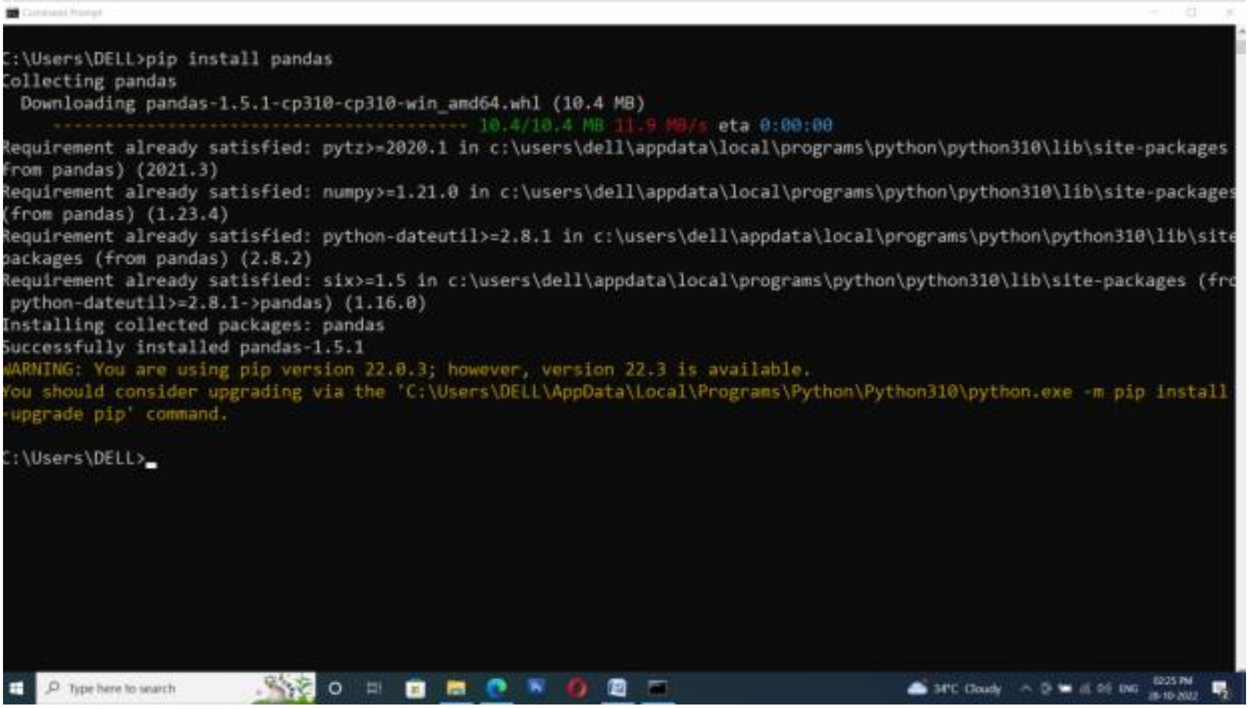
```
Microsoft Windows [Version 10.0.19043.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>pip install numpy
Collecting numpy
  Downloading numpy-1.23.4-cp310-cp310-win_amd64.whl (14.6 MB)
     ---------------------------------------- 14.6/14.6 MB 12.1 MB/s eta 0:00:00
Installing collected packages: numpy
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behavio
r is the source of the following dependency conflicts.
scipy 1.7.3 requires numpy<1.23.0,>=1.16.5, but you have numpy 1.23.4 which is incompatible.
Successfully installed numpy-1.23.4
WARNING: You are using pip version 22.0.3; however, version 22.3 is available.
You should consider upgrading via the 'C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe -m pip install
-upgrade pip' command.

C:\Users\DELL>_
```

PANDAS

Pandas visualizes and manipulates data tables. There are many functions that allow efficient manipulation for the preliminary steps of data analysis problems.

Ex: pip install pandas

Output:



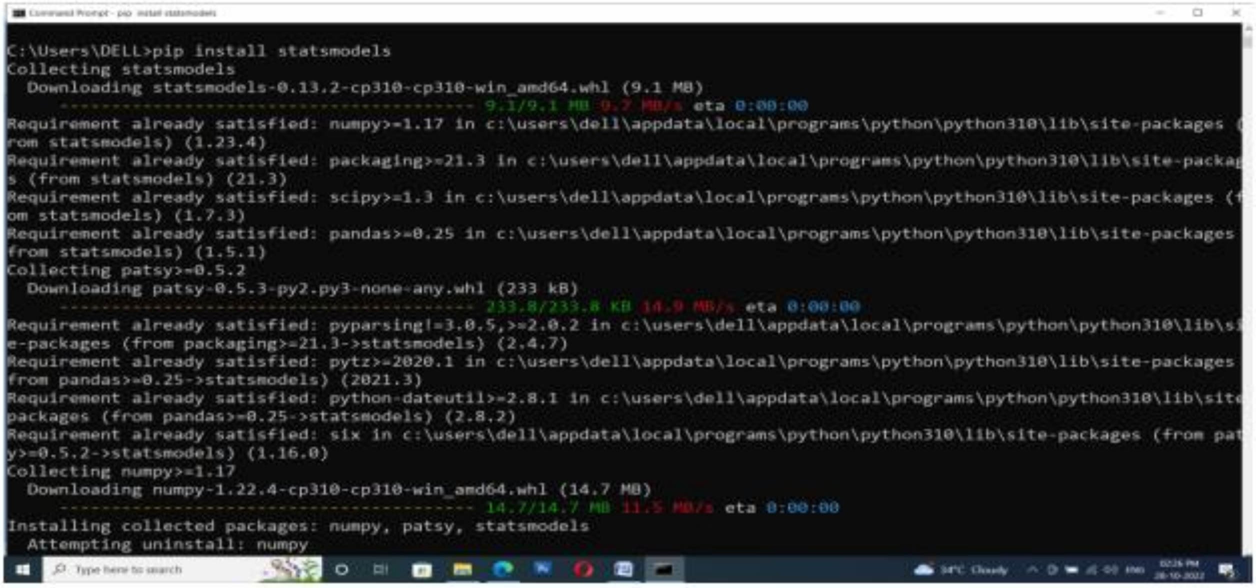
```
C:\Users\DELL>pip install pandas
Collecting pandas
  Downloading pandas-1.5.1-cp310-cp310-win_amd64.whl (10.4 MB)
     ---------------------------------------- 10.4/10.4 MB 11.9 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages
from pandas) (2021.3)
Requirement already satisfied: numpy>=1.21.0 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages
(from pandas) (1.23.4)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\dell\appdata\local\programs\python\python310\lib\site
packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (fro
 python-dateutil>=2.8.1->pandas) (1.16.0)
Installing collected packages: pandas
Successfully installed pandas-1.5.1
WARNING: You are using pip version 22.0.3; however, version 22.3 is available.
You should consider upgrading via the 'C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe -m pip install
-upgrade pip' command.

C:\Users\DELL>
```

Statsmodel

Statsmodels is a package for exploring data, estimating statistical models, and performing statistical tests. It include descriptive statistics, statistical tests, plotting functions, and result statistics.

Ex: pip install statsmodels

Output:

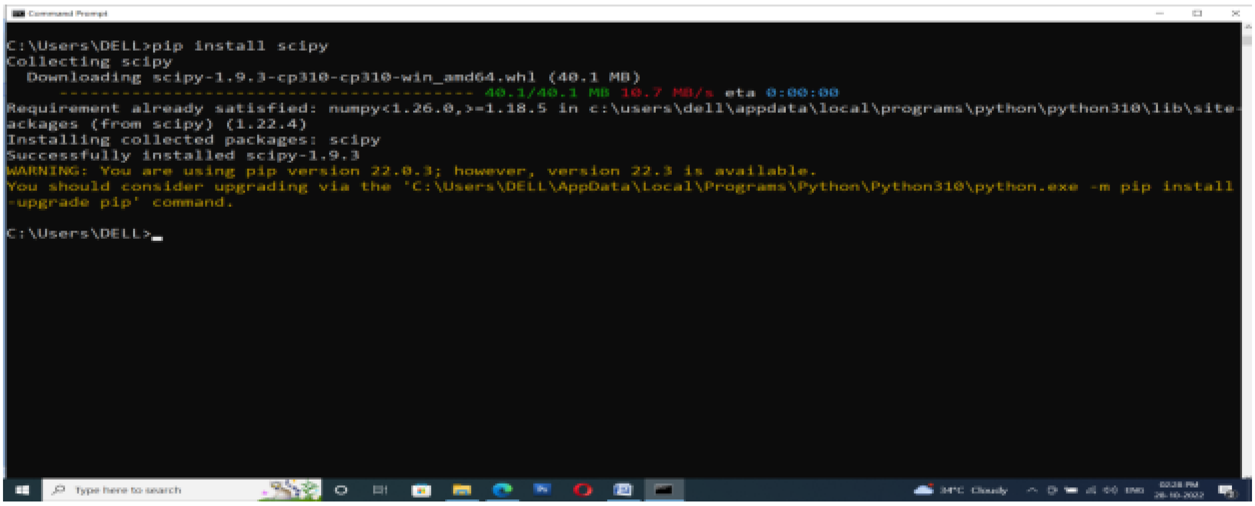


```
C:\Users\DELL>pip install statsmodels
Collecting statsmodels
  Downloading statsmodels-0.13.2-cp310-cp310-win_amd64.whl (9.1 MB)
     ---------------------------------------- 9.1/9.1 MB 9.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (
rom statsmodels) (1.23.4)
Requirement already satisfied: packaging>=21.3 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packag
s (from statsmodels) (21.3)
Requirement already satisfied: scipy>=1.3 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (f
om statsmodels) (1.7.3)
Requirement already satisfied: pandas>=0.25 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages
from statsmodels) (1.5.1)
Collecting patsy>=0.5.2
  Downloading patsy-0.5.3-py2.py3-none-any.whl (233 kB)
     ---------------------------------------- 233.8/233.8 KB 14.9 MB/s eta 0:00:00
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\dell\appdata\local\programs\python\python310\lib\si
e-packages (from packaging>=21.3->statsmodels) (2.4.7)
Requirement already satisfied: pytz>=2020.1 in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages
from pandas>=0.25->statsmodels) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\dell\appdata\local\programs\python\python310\lib\site
packages (from pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: six in c:\users\dell\appdata\local\programs\python\python310\lib\site-packages (from pat
y>=0.5.2->statsmodels) (1.16.0)
Collecting numpy>=1.17
  Downloading numpy-1.22.4-cp310-cp310-win_amd64.whl (14.7 MB)
     ---------------------------------------- 14.7/14.7 MB 11.5 MB/s eta 0:00:00
Installing collected packages: numpy, patsy, statsmodels
  Attempting uninstall: numpy
```

Scipy:

SciPy is a general-purpose package for mathematics, science, and engineering and extends the base capabilities of NumPy.

Ex: pip install scipy



```
C:\Users\DELL>pip install scipy
Collecting scipy
  Downloading scipy-1.9.3-cp310-cp310-win_amd64.whl (40.1 MB)
     ---------------------------------------- 40.1/40.1 MB 10.7 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.26.0,>=1.18.5 in c:\users\dell\appdata\local\programs\python\python310\lib\site-
ackages (from scipy) (1.22.4)
Installing collected packages: scipy
Successfully installed scipy-1.9.3
WARNING: You are using pip version 22.0.3; however, version 22.3 is available.
You should consider upgrading via the 'C:\Users\DELL\AppData\Local\Programs\Python\Python310\python.exe -m pip install
-upgrade pip' command.

C:\Users\DELL>_
```

Viva Questions:

Result:

Thus we have successfully installed all the packages and verfied

EX NO:
DATE:

Working with Numpy arrays

Aim:

To Verify the working of NumPy Arrays.

Algorithm:

```
Step 1: Start.
Step 2: import the numpy package
Step 3:Assign the variable name
Step 4:create a numpy array
Step 5: Print the array
Step 6: Print the shape of the Array
Step 7: Stop.
```

Program:

One Dimensional Array:

```
# importing numpy module
import numpy as np
# creating list list = [1, 2, 3, 4]
# creating numpy array
sample_array = np.array(list1)
print("List in python : ", list)
print("Numpy Array in python :", sample_array)
```

Multi-Dimensional Array:

```
# importing numpy module
import numpy as np
# creating list
list_1 = [1, 2, 3, 4]
list_2 = [5, 6, 7, 8]
list_3 = [9, 10, 11, 12]
# creating numpy array
sample_array = np.array([list_1, list_2, list_3])
print("Numpy multi dimensional array in python\n", sample_array)
```

Viva Questions:

Result:

Thus, the Working of numpy array have been Successfully executed and the output is verified

EX NO: **Working with Pandas data frames**
DATE:

Aim:

To write a phyton program in data frames using Pandas Module

Algorithm:

```
Step 1: Start
Step 2: import the Pandas module as pd
Step 3: Declare the array in column and low
Step 4: Call the function inside the data frame
Step 5: Print the Function
Step 6: Stop
```

Program:

Creating a Pandas DataFrame

```
import pandas as pd
import numpy as np
sas=pd.Series([1,3,5,np.nan,6])
sas
```

Dealing with Rows and Columns

```
import pandas as pd
data={'apple': [3,2,0], 'orange' : [3,8,9]}
purchase=pd.DataFrame(data)
purchase
```

Viva Questios:

1)What are the three principal components?
2) What is pandas' data frame?
3)How Pandas data frame can be created?

Result:

Thus the Program is successfully executed and the output is verified.

EX NO: **Reading data from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set.**

DATE:

AIM

To Reading the datas from text files, Excel and the web and exploring various commands for doing descriptive analytics on the Iris data set.

ALGORITHM

```
Step 1: Start
Step 2: Download the dataset from www.kaggle.com
Step 3: download the Iris.csv file from the above link
Step 4: use the Pandas library to load this CSV file, and convert it into the dataframe
Step 5: read_csv() method is used to read CSV files.
Step 6:Display the result
Step 7:Stop the program
```

PROGRAM

```
import pandas as pd
data1=pd.read_csv("Iris.csv")
data1.head()
data1.info()
data1.describe()
data1.isnull().sum()
data1.shape
data = data1.drop_duplicates(subset ="Species",)
data
```

VIVA QUESTIONS

RESULT

Thus, the program is executed successfully and the output is verified.

EX NO: **Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following**

DATE:

AIM

To Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following

ALGORITHM

Step 1: Start

Step 2: Import Numpy and Pandas packages

Step 3: Download and import the Pima Indian Diabeties Dataset from any website

Step 4: Read the file by read_csv method using pandas

Step 5: Gathering information about this dataset

Step 6: then gathering mean,skewness,variance,median,mode,frequency by various commands

Step7: Display the result

Step 8: Stop the program

PROGRAM

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

```
import pandas as pd
import numpy as np
import statistics as st
# Load the data
df = pd.read_csv("diabetes.csv")
print(df.shape)
print(df.info())
```

Mean

Code:

```
df.mean()
print(df.loc[:,'Age'].mean())
print(df.loc[:,'Income'].mean())
```

Median

Code:

```
df.median()
```

Mode

Code:

```
df.mode()
```

Standard Deviation

Code:

```
df.std()
```

Variance

Code:

```
df.var()
```

Interquartile Range (IQR)

Code:

```
from scipy.stats import iqr
```

```
iqr(df['Age'])
```

Skewness

Code:

```
print(df.skew())
```

b. Bivariate analysis: Linear and logistic regression modeling

Code:

```
 import pandas as pd
df = pd.read_csv(diabetes.csv')
df.head()
```

Code:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='whitegrid', context='notebook')
cols =
['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigree
Function','Age']
```

Code:

```
import numpy as np
cm = np.corrcoef(df[cols].values.T)
sns.set(font_scale=1.5)
hm = sns.heatmap(cm,cbar=True,annot=True,square=True,fmt='.2f',annot_kws={'size':
15},yticklabels=cols,xticklabels=cols)
plt.show()
```

Code:

```
class LinearRegressionGD(object):
def __init__(self, eta=0.001, n_iter=20):
 self.eta = eta
self.n_iter = n_iter
def fit(self, X, y):
self.w_ = np.zeros(1 + X.shape[1])
self.cost_ = []
for i in range(self.n_iter):
output = self.net_input(X)
errors = (y - output)
self.w_[1:] += self.eta * X.T.dot(errors)
self.w_[0] += self.eta * errors.sum()
cost = (errors**2).sum() / 2.0
self.cost_.append(cost)
return self
def net_input(self, X):
return np.dot(X, self.w_[1:]) + self.w_[0]
def predict(self, X):
return self.net_input(X)
X = df[['Age']].values
y = df['Pregnancies'].values
from sklearn.preprocessing import StandardScaler sc_x = StandardScaler() sc_y =
StandardScaler()
X_std = sc_x.fit_transform(X)
 y_std = sc_y.fit_transform(y)
lr = LinearRegressionGD()
lr.fit(X_std, y_std)
plt.plot(range(1, lr.n_iter+1), lr.cost_)
plt.ylabel('SSE')
plt.xlabel('Epoch')
plt.show()
```

Code:

```
def lin_regplot(X, y, model):
plt.scatter(X, y, c='blue')
plt.plot(X, model.predict(X), color='red')
return None
lin_regplot(X_std, y_std, lr)
plt.xlabel('Age (standardized)')
plt.ylabel('Pregnancies(standardized)')
plt.show()
```

Code:

```
age_std = sc_x.transform([20])
pregnancy_std = lr.predict(age_std)
print("Pregnancy: %.3f" %sc_y.inverse_transform(price_std))
print('Slope: %.3f' % lr.w_[1])
```

c. **Multiple Regression analysis:**

Code:

```
from sklearn.model_selection import
train_test_split train_x, test_x, train_y, test_y =
train_test_split(X,Y,test_size=0.3,random_state=99)
train_x.shape, train_y.shape
from sklearn.linear_model import MultipleRegression
le = MultipleRegression()
le.fit(train_x,train_y)
y_pred = le.predict(test_x)
y_pred
result = pd.DataFrame({'Actual': test_y, 'Predict' : y_pred})
result
```

Code:

```
print ('coefficient', le.coef_)
print('intercept', le.intercept_)
```

d. **Also compare the results of the above analysis for the two data sets**

pip install datacompy

Code: import datacompy

```
compare = datacompy.Compare(df1,df2,join_columns=’acct_id’, abs_tol=0.0001,
rel_tol=0,df1_name=’olddiabetes’,df2_name=’newdiabetes’)
print(compare.report())
```

VIVA QUESTIONS

RESULT

Thus, the program is executed successfully and the output is verified.

EX NO:

Apply and explore various plotting functions on UCI data sets

DATE:

AIM

To Apply and explore various plotting functions on UCI data sets

ALGORITHM

Step 1: Start

Step 2: Create a Series of Data

Step 3: Calculate mean and Standard Deviation

Step 4: Apply function to the data

Step 5: Display the result

Step 6: End the Program

PROGRAM

a. **Normal curves**

Code:

```
import numpy as np
import matplotlib.pyplot as plt
# Creating a series of data of in range of 1-50.
 x = np.linspace(1,50,200)
#Creating a Function.
def normal_dist(x , mean , sd):
prob_density = (np.pi*sd) * np.exp(-0.5*((x-mean)/sd)**2)
return prob_density
#Calculate mean and Standard deviation.
mean = np.mean(x)
sd = np.std(x)
#Apply function to the data.
pdf = normal_dist(x,mean,sd)
#Plotting the Results
plt.plot(x,pdf , color = 'red')
plt.xlabel('Data points')
plt.ylabel('Probability Density)
```

b. **Density and contour plots**

Code:

```
import matplotlib.pyplot as plt
import numpy as np
feature_x = np.arange(0, 50, 2)
feature_y = np.arange(0, 50, 3)
# Creating 2-D grid of features
[X, Y] = np.meshgrid(feature_x, feature_y)
fig, ax = plt.subplots(1, 1)
Z = np.cos(X / 2) + np.sin(Y / 4)
# plots contour lines
ax.contour(X, Y, Z)
ax.set_title('Contour Plot')
ax.set_xlabel('feature_x')
ax.set_ylabel('feature_y')
```

```
plt.show()
```

c. Correlation and scatter plots

Code:

```
import pandas as pd
con = pd.read_csv('concrete.csv')
con
list(con.columns)
con.head()
con['cement'] = con['cement'].astype('category')
con.describe(include='category')
import seaborn as sns

sns.scatterplot(x="water", y="coarseagg", data=con);


ax = sns.scatterplot(x="water", y="coarseagg", data=con)
ax.set_title("Concrete Strength vs. Fly ash")
ax.set_xlabel("coarseagg");

sns.lmplot(x="water", y="coarseagg", data=con);
```

d. Histograms:

Code:

```
from matplotlib import pyplot as plt
import numpy as np
 # Creating dataset
 a = np.array([22, 87, 5, 43, 56, 73, 55, 54, 11, 20, 51, 5, 79, 31, 27])
# Creating histogram
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(a, bins = [0, 25, 50, 75, 100])
# Show plot
plt.show()
```

Code:

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import colors
from matplotlib.ticker import PercentFormatter
# Creating dataset
np.random.seed(23685752)
N_points = 10000
n_bins = 20
# Creating distribution.
x = np.random.randn(N_points)
y = .8 ** x + np.random.randn(10000) + 25
# Creating histogram
fig, axs = plt.subplots(1, 1,figsize =(10, 7),tight_layout = True)
axs.hist(x, bins = n_bins) #
Show plot
plt.show()
```

E.<u>Three-dimensional plotting</u>

Code:

```
from mpl_toolkits import mplot3d
import numpy as np
```

```
 import matplotlib.pyplot as plt
fig = plt.figure()
 # syntax for 3-D projection
ax = plt.axes(projection ='3d')
 # defining axes
 z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)
c = x + y
ax.scatter(x, y, z, c = c)
# syntax for plotting
ax.set_title('3d Scatter plot')
plt.show()
```

VIVA QUESTIONS

RESULT

Thus the output is successfully executed and the output is verified

EX NO:

Visualizing Geographic Data with Basemap

DATE:

AIM

To Visualizing Geographic Data with Basemap

ALGORITHM

Step 1: Start

Step 2: Import numpy as np

Step 3: Import seaborn

Step 4:Import shape File as shp as Shapely.geomentry

Step 5: Display the Result

Step 6: Stop the Program

PROGRAM

CODE:

```
%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);
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=12);
from mpl_toolkits.basemap import Basemap
import matplotlib.pyplot as plt
fig = plt.figure(figsize = (12,12))
m = Basemap()
m.drawcoastlines()
m.drawcoastlines(linewidth=1.0, linestyle='dashed', color='red')
plt.title("Coastlines", fontsize=20)
plt.show()
```

CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
import shapefile as shp from shapely.geometry
import Point sns.set_style('whitegrid')
fp = r'Maps_with_python\india-polygon.shp'
```

```
map_df = gpd.read_file(fp)
map_df_copy = gpd.read_file(fp)
plt.plot(map_df , markersize=5)
```

VIVA QUESTIONS

RESULT

Thus the output is successfully executed and the output is verified.