**DATA SCIENCE WITH PYTHON**

**Student Name:M.CHAMESWARI ID Number: N190455 Class:CSE-1**

**DEPARTMENT OF COMPUTER SCIENCE AND**

**ENGINEERING RGUKT-NUZ-AP**

**E2-SEMESTER-II, AY-2022-23**

**TABLE OF CONTENTS:**

|  |  |
| --- | --- |
| **Assignment** | **Assignment Name Page Number Date** |
| **number** |  |
|  |  |
| **1** | **Python Basics** |
|  |  |
| **2** | **Python Data** |
|  | **Structures** |
|  |  |
| **3** | **Python Programming** |
|  | **Fundamentals** |
|  |  |

**4**

**5**

**6**

**7**

**Working with data in**

**python**

**Working Numpy**

**Arrays**

**Importing and datasets**

**and cleaning and**

**preparing the data**

**Model Development**

**8 9**

**Introduction to**

**Model Evaluation Visualization tools**

**10 Basic Visualization**

**11 Specialized**

**visualization tools**

**12 Advanced**

**Visualization Tools**

**LAB -01**

**Python Basics:** Your first program,Types Expressions and Variables string Operations.

**Code:**

name="chamu"

print(name)

num1=5

num2=2.0

num3=3+2j

print(type(num1),type(num2),type(num3))

lang=['swift','c++','python']

print(lang)

print(lang[1])

print(type(lang))

tuple=('chamu',2,4.5)

print(type(tuple))

set={1,1,2,2,3,4,4}

print(set)

print(type(set))

set1=(1,2,3,5,6,7)

print(set.union(set1))

print(set.intersection(set1))

print(set.difference(set1))

print(set.issubset(set1))

print(set.symmetric\_difference(set1))

print(set.issuperset(set1))

print(set.isdisjoint(set1))

dict={1:'chamu',"nepal":"kammam",3:'tomato'}

print(type(dict))

print(dict.keys())

print(dict.items())

print(dict.values())

print(dict["nepal"])

#python list methods

a=['apple' ,5.5, 6]

a.append("Orange")

print(a)

x=a.count(6)

print(x)

a.extend(['volvo'])

print(a)

a.append(['volvo'])

print(a)

a.insert(1,"kiran")

print(a)

z=a.remove(6)

print(z)

y=a.pop(2)

print(y)

a.reverse()

print(a)

lava=[1,2,3]

kush=lava.copy()

print("copy of a list:",kush)

print("clear of list:",lava.clear())

print("######String Methods:######")

txt="string methods begin from here."

print("capatalize the first word:",txt.capitalize())

p=txt.casefold()

print(p)

q=txt.center(40)

print("It returns a centered string:",q)

r=txt.count("string")

print("count the particular word:",r)

t=txt.endswith('s')

print("Ends with:",t)

print("Encode:",txt.encode())

s="H\te\tl\tl\to"

print(s)

print("Expandtabs:",s.expandtabs())

a,b,c=5,3.3,'Anusha'

print("Asssgning different values to different variables:", a,b,c)

a=b=c="Anusha"

print("SAme value assisgned to different variables:",a,b,c)

rge=range(2,22,2)

for n in age:

print(n)

num=5

Num=55

print(num,Num)

**Output:**

chamu

<class 'int'> <class 'float'> <class 'complex'>

['swift', 'c++', 'python']

c++

<class 'list'>

<class 'tuple'>

{1, 2, 3, 4}

<class 'set'>

{1, 2, 3, 4, 5, 6, 7}

{1, 2, 3}

{4}

False

{4, 5, 6, 7}

False

False

<class 'dict'>

dict\_keys([1, 'nepal', 3])

dict\_items([(1, 'chamu'), ('nepal', 'kamam'), (3, 'tomato')])

dict\_values(['Yamuna', 'kamam', 'tomato'])

['apple', 5.5, 6, 'Orange']

1

['apple', 5.5, 6, 'Orange', 'volvo']

['apple', 5.5, 6, 'Orange', 'volvo', ['volvo']]

['apple', 'kiran', 5.5, 6, 'Orange', 'volvo', ['volvo']]

None

5.5

[['volvo'], 'volvo', 'Orange', 'kiran', 'apple']

copy of a list: [1, 2, 3]

clear of list: None

######String Methods:######

capatalize the first word: String methods begin from here. string methods begin from here.

It returns a centered string: string methods begin from here.

count the particular word: 1

Ends with: False

Encode: b'string methods begin from here.'

H e l l o

Expandtabs: H e l l o

Asssgning different values to different variables: 5 3.3 Yamuna

SAme value assisgned to different variables: Yamuna Yamuna Yamuna

2

4

6

8

10

12

14

16

18

20

5 55

**LAB - 02**

**Python Data Structures:** Lists and Tuples Sets, and Dictionaries

**Code:**

site\_name = 'programiz.pro'

print(site\_name)

site\_name = 'programiz.pro'

print(site\_name)

site\_name = 'apple.com'

print(site\_name)

a, b, c = 5, 3.2, 'Hello'

print(a)

print(b)

print(c)

site1 = site2 = 'programiz.com'

print(site1)

print(site2)

string1="python programming"

print(string1)

string2='python programming'

print(string2)

name="python"

print(name)

message="i love python"

print(message)

great="hello"

print(great[1])

print(great[-4])

print(great[1:4])

print(great[:4])

Message='Hola Amigos'

#Message[0]='H'

print(Message)

Message='Hello friends'

print(Message)

Message="""Never gona give you up Never gona give you down"""

print(Message)

str1="hello world"

str2=" i love python"

str3="hello world"

print(str1==str2)

print(str1==str3)

greet="hello"

name="jack"

result=greet+name

print(result)

for letter in greet:

print(letter)

print(len(greet))

print('a' in 'program')

print('at' not in 'battle')

num=[1,2,5]

print(num)

lan=["python","swift","c++"]

print(lan[0])

print(lan[2])

print(lan[-1])

print(lan[-3])

my\_list=['p','r','o','g','m','i','z']

print(my\_list[2:5])

print(my\_list[5:])

print(my\_list[:])

num.append(32)

print(num)

num.insert(1,35)

print(num)

numbers=[4,5,6]

num.extend(numbers)

print(num)

languages=['python','swit','c']

languages[2]='c'

print(languages)

del languages[1]

print(languages)

del languages[-1]

print(languages)

languages.remove('python')

print(languages)

language=['python','swit','c']

print('c' in language)

print('python' in language)

print(len(language))

numbers=[number\*number for number in range(1,6)]

print(numbers)

my\_tuple=()

print(my\_tuple)

my\_tuple=(1,2,3)

print(my\_tuple)

my\_tuple=(1,"hello",3.4)

print(my\_tuple)

my\_tuple=("mouse",[8,4,6],(1,2,3))

print(my\_tuple)

var1=("hello")

print(type(var1))

var2=("hello",)

print(type(var2))

letters=('p','r','o','g','m','i','z')

print(letters[-1])

print(letters[-3])

my\_tuple=('p','r','o','g','m','i','z')

print(my\_tuple[1:4])

print(my\_tuple[:-7])

print(my\_tuple[7:])

print(my\_tuple[:])

print(my\_tuple.count('p'))

print(my\_tuple.index('i'))

capital\_city={"nepal":"kathmandu","england":"london"} print("initial dictionary:",capital\_city) capital\_city["japan"]="tokyo" print("updated dictionary:",capital\_city)

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print("Initial Dictionary: ", student\_id)

student\_id[112] = "Stan"

print("Updated Dictionary: ", student\_id)

student\_id = {111: "Eric", 112: "Kyle", 113: "Butters"}

print(student\_id[111])

print(student\_id[113])

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

print(1 in squares)

print(2 not in squares)

print(49 in squares)

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

for i in squares:

print(squares[i])

A= {1, 3, 5}

B= {0, 2, 4}

print('Union using |:', A | B)

print('Union using union():', A.union(B))

print('Intersection using &:', A & B)

print('Intersection using intersection():', A.intersection(B))

print('Difference using &:', A - B)

print('Difference using difference():', A.difference(B))

print('using ^:', A ^ B)

print('using symmetric\_difference():', A.symmetric\_difference(B))

if A == B:

print('Set A and Set B are equal')

else:

print('Set A and Set B are not equal')

"""greet="Hello",name="Jack"

result = greet+name

print(result)"""

numbers = [1, 2, 5]

print(numbers)

prime\_numbers = [2, 3, 5]

print("List1:", prime\_numbers)

even\_numbers = [4, 6, 8]

print("List2:", even\_numbers)

prime\_numbers.extend(even\_numbers)

print("List after append:", prime\_numbers)

**OUTPUT:-**

programiz.pro

programiz.pro

apple.com

5

3.2

Hello

programiz.com

programiz.com

python programming

python programming

python

i love python

e

e

ell

hell

Hola Amigos

Hello friends

Never gona give you up Never gona give you down

False

True

hellojack

h

e

l

l

o

5

True

False

[1, 2, 5]

python

c++

c++

python

['o', 'g', 'm']

['i', 'z']

['p', 'r', 'o', 'g', 'm', 'i', 'z']

[1, 2, 5, 32]

[1, 35, 2, 5, 32]

[1, 35, 2, 5, 32, 4, 5, 6]

['python', 'swit', 'c']

['python', 'c']

['python']

[]

True

True

3

[1, 4, 9, 16, 25]

()

(1, 2, 3)

(1, 'hello', 3.4)

('mouse', [8, 4, 6], (1, 2, 3))

<class 'str'>

<class 'tuple'>

z

m

('r', 'o', 'g')

()

()

('p', 'r', 'o', 'g', 'm', 'i', 'z')

1

5

initial dictionary: {'nepal': 'kathmandu', 'england': 'london'}

updated dictionary: {'nepal': 'kathmandu', 'england': 'london', 'japan': 'tokyo'}

Initial Dictionary: {111: 'Eric', 112: 'Kyle', 113: 'Butters'}

Updated Dictionary: {111: 'Eric', 112: 'Stan', 113: 'Butters'}

Eric

Butters

True

True

False

1

9

25

49

81

Union using |: {0, 1, 2, 3, 4, 5}

Union using union(): {0, 1, 2, 3, 4, 5}

Intersection using &: set()

Intersection using intersection(): set()

Difference using &: {1, 3, 5}

Difference using difference(): {1, 3, 5}

using ^: {0, 1, 2, 3, 4, 5}

using symmetric\_difference(): {0, 1, 2, 3, 4, 5}

Set A and Set B are not equal

[1, 2, 5]

List1: [2, 3, 5]

List2: [4, 6, 8]

List after append: [2, 3, 5, 4, 6, 8]

**LAB- 03**

**Python Programming Fundamentals:** Conditions and Branching

Loops, Functions,Objects and Classes

**if-else**

number=int(input("Enter a Number:"))

if number>10:

print('Number is greater than 10')

else:

print('Number is less than 10')

**Output:**

Enter a number:11

Number is greater than 10

**If-elif-else**

num=int(input('Enter a Number:'))

if num>0:

print('Positive Number')

elif num<0:

print('Negative Number')

else:

print('Positive Number')

print('This statement is always executed')

**Output:**

Enter a number:10

Positive Number

**nested-if**

num=int(input('Enter a Number:'))

if(num>=0):

if num==0:

print('Number is 0')

else:

print('Number is positive')

else:

print('Number is Negative')

**output:**

Enter a Number:15

Number is positive

**short-hand-if**

a=10;

b=20;

if a<b: print('This is if')

**Output:**

This is if

**shorthand-if-else**

a=30;

b=20;

print('This is if') if a<b else print('this is else')'''

**Output:**

This is else

**for-loop**

lang=['swift','c','python','c++']

for x in lang:

print(x)

**range function**

a=range(6)

for x in a:

print(x)

a=range(1,6)

for x in a:

print(x)

a=range(2,22,2)

for x in a:

print(x)

for i in range(1,1001):

for j in range (1,11):

print(i\*j,end=" ")

print()

**for loops with else**

digits=[0,1,2]

for i in digits:

print(i)

else:

print("No items left.")

**while loop**

i=1

n=5

while i<=n:

print(i)

i=i+1

**Python oops Concept**

**python inheritence**

class Animal:

def speak(self):

print("Animal Speaking")

class Dog(Animal):

def bark(self):

print("dog barking")

class DogChild(Dog):

def eat(self):

print("Eating bread...")

d=DogChild()

d.speak()

d.bark()

d.eat()

**Output:**

Dog Barking

Animal Speaking

Eating Bread

**Method overriding**

'class Animal:

def speak(self):

print("Speaking")

class Dog(Animal):

def speak(self):

print("Not Speaking")

class Cat(Dog):

def speak(self):

print("Is this a cat")

d=Cat()

d.speak()

**Output:**

Speaking

Not Speaking

Is this a cat

**Data Abstraction**

class Employee:

\_\_count=0;

def \_\_init\_\_(self):

Employee.\_\_count=Employee.\_\_count+1

def display(self):

print("The number of Employees",Employee.\_\_count)

emp=Employee()

try:

print(emp.\_\_count)

finally:

emp.display()

**Output:**

Number of Employees:3

**Abstract Method**

from abc import ABC, abstractmethod

class Car(ABC):

def mileage(self):

pass

class Tesla(Car):

def mileage(self):

print("The mileage is 30kmph")

class Suzuki(Car):

def mileage(self):

print("The mileage is 25kmph ")

class Duster(Car):

def mileage(self):

print("The mileage is 24kmph ")

class Renault(Car):

def mileage(self):

print("The mileage is 27kmph ")

* Driver code t= Tesla () t.mileage()

r = Renault()

r.mileage()

s = Suzuki()

s.mileage()

d = Duster()

d.mileage()

**Output**

The mileage is 30kmph

The mileage is 25kmph

The mileage is 24kmph

The mileage is 27kmph

**LAB - 04**

**Working with python in data:**Reading files with python,Writing Files with open,Loading with pandas

# 1.create a dataframe from the list

import pandas as pd

friuts=['mango','papaya','grapes','pine-apple','banana','apple']

print(pd.DataFrame(friuts))

print("\n\n")

# list of strings

lst = ['first', 'second', 'third', 'four',

'five', 'six', 'eight']

* Calling DataFrame constructor on list df = pd.DataFrame(lst)

print(df)

print("\n\n\n")

* 2. Dataframe using list with index and column names import pandas as pd name=['yammu','chamu','sandy','sowji',ram','sri'] df=pd.DataFrame(name,index=[101,102,103,104,105,106],columns=['Names']) print(df)

print("\n\n")

import pandas as pd

# list of strings

lst = ['one', 'is', 'always', 'greater', 'then', 'two', 'three']

* Calling DataFrame constructor on list
* with indices and columns specified

df = pd.DataFrame(lst, index =['a', 'b', 'c', 'd', 'e', 'f', 'g'], columns =['Names'])

print(df)

print("\n\n\n")

#3 Using zip() for zipping two lists

name=['raji','devi','vaishu','yammu','chamu','annita']

rol=['a', 'b', 'c', 'd', 'e', 'f', 'g']

df=pd.DataFrame(list(zip(name,rol)),columns=['Name','Rol'])

print(df)

print("\n\n")

# list of strings

lst = ['one', 'is', 'always', 'greater', 'then', 'two', 'three']

# list of int

lst2 = [11, 22, 33, 44, 55, 66, 77]

* Calling DataFrame constructor after zipping
* both lists, with columns specified

df = pd.DataFrame(list(zip(lst, lst2)),

columns =['Name', 'val'])

print(df)

print("\n\n\n")

#4 Creating DataFrame using multi-dimensional list name=[['raji',20],['sandy',20],['madhu',20],['vaishu',19],['devi',20]] df=pd.DataFrame(name,columns=['Name','Age'])

print(df)

print("\n\n")

# List1

lst = [['cherry', 5], ['puppy', 3],

['sweety', 6], ['honey', 2]]

df = pd.DataFrame(lst, columns =['Name', 'Age'])

print(df)

print("\n\n\n")

#5 Using multi-dimensional list with column name and dtype specified.

cakes=[['1','Vanilla cupcakes',50],['2','Chocolate cupcakes',60],['3','Banana cake',30],['4','walnut cake',40],['5',' coconut cake',25]]

df=pd.DataFrame(cakes,columns=['cake place','cake Name','price'])

print(df)

print("\n\n")

# List1

lst = [['cherry', 'gadu',5 ], ['puppy', 'gadu', 3],

['sweety', 'papa', 6], ['honey', 'papa', 6]]

df = pd.DataFrame(lst, columns =['FName', 'LName', 'Age'])

print(df)

print("\n\n\n")

#6 Using lists in dictionary to create dataframe

movies=['Akkada Ammayi Ikkada Abbayi','bheemla nayak','Thammudu','Gabbar Singh','Kushi','Tholi Prema']

place=['1', '27', '5', '9', '7', '4']

dic={'Movies':movies,'Place':place}

df=pd.DataFrame(dic)

print(df)

print("\n\n")

# list of name, degree, score

name = ["ram", "vaishu", "devi", "yamuna"]

deg = ["puc", "puc", "puc", "puc"]

scr = [9.02, 9.8, 8.8, 9.0]

print("\n\n")

* dictionary of lists dict = {'name': name, 'degree': deg,

'score': scr}

df = pd.DataFrame(dict)

print(dict)

print(df)

print("\n\n\n")

df=pd.read\_csv('C:\\Users\\Chameswari\\Desktop\\image\\grade.csv',index\_col=("Na me")) f=df.loc["Yamuna"]

s=df.loc["Ambica"]

a=df["At-02"]

print(f ,"\n\n\n",s)

print("\n\n")

print(a)

print("\n")

sf=df.iloc[3]

print(sf)

print("\n\n")

#iteration over rows and columns

# dictionary of lists

dict = {'name':["Adinarayana", "Lakshmi", "Srinu", "Rajeswari"], 'degree': ["10th", "no", "MCA", "BTECH"],

'score':[100, 80, 98, 95]}

df=pd.DataFrame(dict)

for i,j in df.iterrows():

print(i,j)

print()

print("\n\n\n")

# dictionary of lists

dict = {'name':["Adinarayana", "Lakshmi", "Srinu", "Rajeswari"], 'degree': ["10th", "no", "MCA", "BTECH"],

'score':[100, 80, 98, 95]}

* creating a dataframe from a dictionary df = pd.DataFrame(dict)

print(df)

columns=list(df)

for i in columns:

print(df[i][0])

**Output:**

0

0 mango 1

papaya

2 grapes

* pine-apple

4 banana

5 apple

0

* first

1 second

2 third

3 four

4 five

5 six

6 eight

Names

1. yammu
2. chamu
3. sandy

1. sowji
2. ram

106 sri

Names

* one b is

c always d greater e then

f two

g three

Name Rol

0 raji a

1 devi b

2 vaishu c

3 yammu d

4 chamu e

5 annita f

Name val

0 one 11 1 is

22

2 always 33 3

greater 44 4

then 55 5 two

66 6 three 77

Name Age

0 raji 20

1 sandy 20

2 madhu 20

3 vaishu 19

4 devi 20

Name Age

0 cherry 5

1 puppy 3

2 sweety 6

3 honey 2

cake place cake Name price 0 1

Vanilla cupcakes 50 1 2 Chocolate

cupcakes 60 2 3 Banana cake 30 3

4

walnut cake 40 4 5 coconut cake 25

FName LName Age

0 cherry gadu 5

1 puppy gadu 3

2 sweety papa 6

3 honey papa 6

Movies Place

0 Akkada Ammayi Ikkada Abbayi 1 1

bheemla nayak 27 2 Thammudu 5 3

Gabbar Singh 9 4 Kushi 7

5 Tholi Prema 4

{'name': ['yammu', 'vaishu', 'devi', 'sanjay’], 'degree': ['puc', 'puc', 'puc', 'puc'], 'score':

[9.02, 9.8, 8.8, 9.0]}

name degree score

0 yammu puc 9.02

1 vaishu puc 9.80

2 devi puc 8.80

3 sanjay puc 9.00

0 name Adinarayana

degree 10th

score 100

Name: 0, dtype: object

1 name Lakshmi

degree no

score 80

Name: 1, dtype: object

2 name Srinu

degree MCA

score 98

Name: 2, dtype: object

3 name Rajeswari

degree BTECH

score 95

Name: 3, dtype: object

name degree score

0 Adinarayana 10th 100

1 Lakshmi no 80

2 Srinu MCA 98

3 Rajeswari BTECH 95

Adinarayana

10th

100

**LAB - 05**

**Working with Numpy Arrays:**Numpy 1d Arrays,Numpy 2D Arrays

**Code:**

import numpy as np

#arrange

arr=np.arange(20)

print(arr)

#shape

arr.shape

print("Shape of array:",arr)

print(arr[4])

#asisgning a value

arr[7]=777

print(arr)

#reshape the existing array

arr=np.arange(20).reshape(4,5)

print("Rearranging the array:",arr)

print(arr.shape)

print(arr[1][2])

array=np.arange(27).reshape(3,3,3)

print(array)

#zero function

print(np.zeros((2,4)))

#ones function

print(np.ones((2,4)))

#empty function

print(np.empty((2,2)))

#full function

print(np.full((4,3),7))

#eye function

print(np.eye(3,3))

#linespace

print(np.linspace(0, 100, num=5))

#conversion from list to array

list=[4,5,6]

print(list)

array=np.array(list)

print(array)

print(type(array))

#random funcion

print(np.random.random((2,2)))

print(np.shape(array))

print(np.size(array))

print(np.dtype(float))

array1=np.array([1,2,3,4,5])

print(array1[1:5])

print(array1[:])

print(array1[3:])#copying the array

myarray=np.copy(array1)

print(myarray)

myarray[2]=8

print(myarray,array1)

#view function

arrayv=array1.view()

print(arrayv)

arrayv[2]=9

print(arrayv,array1)

yammu=np.array(['A','B'])

ary=np.array([11,22,33,44])

print(ary)

print(np.delete(ary,2))

#atack function

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

b=np.array([5,6,7,8])

c=np.stack((a,b),axis=1)

print(c)

#concatenate

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

y=np.array([[12,30]])

r=np.array([[33,44]])

z=np.concatenate((x,y),axis=0)

print(z)

print(np.vstack((x,y)))

print(np.hstack((y,r)))

print(np.dstack((y,r)))

split=np.array([11,22,33,44,55,66])

newarr=np.array\_split(split,3)

print(newarr)

#where function

t=np.arange(12)

s=np.where(a<6,a,5\*a)

print(s)

fun=np.array([1,11,2,22,3,33])

print(np.max(fun))

print(np.min(fun))

print(np.mean(fun))

print(np.median(fun))

print(np.var(fun))

print(np.std(fun))

**Output:**

* 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19] Shape of array: [ 0 1

2345678910111213141516171819]4

* 01234567778910111213141516171819] Rearranging the array: [[ 0 1 2 3 4]

[56789]

[10 11 12 13 14]

[15 16 17 18 19]] (4, 5)

7

[[[012]

* 345]
* 678]]

[[ 9 10 11]

[12 13 14]

[15 16 17]]

[[18 19 20]

[21 22 23]

[24 25 26]]]

[[0. 0. 0. 0.]

[0. 0. 0. 0.]]

[[1. 1. 1. 1.]

[1. 1. 1. 1.]]

[[2.12199579e-314 4.67296746e-307]

[1.11658836e-320 1.04614393e-311]]

[[7 7 7]

[777]

[777]

[7 7 7]]

[[1. 0. 0.]

[0. 1. 0.]

[0. 0. 1.]]

* 0. 25. 50. 75. 100.] [4, 5, 6] [456]

<class 'numpy.ndarray'> [[0.04181302 0.70924674]

[0.96165792 0.17146781]]

(3,)

3

float64

[2345]

[12345]

[4 5]

[12345]

[12845][12345]

[12345]

[12945][12945]

[11 22 33 44]

[11 22 44]

[[1 5]

[2 6]

[3 7]

[4 8]]

[[12]

* 3 4]

[12 30]] [[12]

* 3 4]

[12 30]]

[[12 30 33 44]]

[[[12 33] [30 44]]]

[array([11, 22]), array([33, 44]), array([55, 66])] [1234] 33 1

12.0

7.0

140.66666666666666

11.86029791643813

**LAB -06**

**Cleaning and preparing the data:**Identify the Handle Missing Values

**Code:**

import pandas as pd

data=pd.read\_csv("C:\\Users\\Chameswari\\Downloads\\toyota.csv",index \_col=0, na\_values=['??','NaN'])

print(data)

new\_data=data.dropna()

print(new\_data)

fill=data.fillna(5000)

print(fill)

null=data.isnull()

print(null)

**Output:**

Price Age KM FuelType ... Automatic CC Doors Weight 0 13500 23.0 46986.0 Diesel ... 0 2000 three 1165 1 13750 23.0 72937.0 Diesel ... 0 2000 3 1165 2 13950 24.0 41711.0 Diesel ... 0 2000 3 1165 3 14950 26.0 48000.0 Diesel ... 0 2000 3 1165 4 13750 30.0 38500.0 Diesel ... 0 2000 3 1170 ... ... ... ... ... ... ... ... ... 1431 7500 NaN 20544.0 Petrol ... 0 1300 3 1025 1432 10845 72.0 NaN Petrol ... 0 1300 3 1015 1433 8500 NaN 17016.0 Petrol ... 0 1300 3 1015 1434 7250 70.0 NaN NaN ... 0 1300 3 1015 1435 6950 76.0 1.0 Petrol ... 0 1600 5 1114

[1436 rows x 10 columns]

Price Age KM FuelType ... Automatic CC Doors Weight 0 13500 23.0 46986.0 Diesel ... 0 2000 three 1165 1 13750 23.0 72937.0 Diesel ... 0 2000 3 1165 3 14950 26.0 48000.0 Diesel ... 0 2000 3 1165 4 13750 30.0 38500.0 Diesel ... 0 2000 3 1170 5 12950 32.0 61000.0 Diesel ... 0 2000 3 1170 ... ... ... ... ... ... ... ... ... 1423 7950 80.0 35821.0 Petrol ... 1 1300 3 1015 1424 7750 73.0 34717.0 Petrol

... 0 1300 3 1015 1429 8950 78.0 24000.0 Petrol ... 1 1300 5 1065 1430 8450 80.0 23000.0 Petrol ... 0 1300 3 1015 1435 6950 76.0 1.0 Petrol ... 0 1600 5 1114

[1099 rows x 10 columns]

Price Age KM FuelType ... Automatic CC Doors Weight 0 13500

23.0 46986.0 Diesel ... 0 2000 three 1165 1 13750 23.0 72937.0 Diesel

... 0 2000 3 1165 2 13950 24.0 41711.0 Diesel ... 0 2000 3 1165 3 14950 26.0 48000.0 Diesel ... 0 2000 3 1165 4 13750 30.0 38500.0 Diesel ... 0 2000 3 1170 ... ... ... ... ... ... ... ... ...

1431 7500 5000.0 20544.0 Petrol ... 0 1300 3 1025 1432 10845 72.0 5000.0 Petrol ... 0 1300 3 1015 1433 8500 5000.0 17016.0 Petrol ... 0 1300 3 1015 1434 7250 70.0 5000.0 5000 ... 0 1300 3 1015 1435 6950 76.0 1.0 Petrol ... 0 1600 5 1114

[1436 rows x 10 columns]

Price Age KM FuelType ... Automatic CC Doors Weight 0 False False False False ... False False False False 1 False False False False

... False False False False 2 False False False False ... False False False False

3 False False False False ... False False False False 4 False False False False ... False False False False ... ... ... ... ... ... ... ...

...

1431 False True False False ... False False False False 1432 False False True False ... False False False False 1433 False True False False ... False False False False 1434 False False True True ...

False False False False 1435 False False False False ... False False False False

[1436 rows x 10 columns]

**LAB - 07**

**Model Development:**Simple and Multiple Linear Regression ,Model Evaluation Using Visualization

**Simple Linear**

* -\*- coding: utf-8 -\*-

"""

Created on Sun Jun 18 13:03:25 2023

@author: chamu

"""

import numpy as np

**import pandas as pd**

**df=pd.read\_csv("lin.csv")**

**print(df)**

**df.head()**

**data\_=df.loc[:,['product','cost']]**

**print(data\_.head(6))**

**#showing the data in matplotlib**

**#to use we need to first install matplotlib**

**import matplotlib.pyplot as plt**

**df.plot(x='product',y='cost',style='o')**

**plt.xlabel('product')**

**plt.ylabel('cost')**

**plt.show()**

**#dividing the variables into dependent and**

**independent**

**x=pd.DataFrame(df['product'])**

**y=pd.DataFrame(df['cost'])**

**#split the data into train and test sets**

**from sklearn.model\_selection import**

**train\_test\_split**

**x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,tes t\_size=0.2,random\_state=1)**

**#knowning the shapes of the test and train**

**print(x\_train.shape)**

**print(x\_test.shape)**

**print(y\_train.shape)**

**print(y\_test.shape)**

**#train the algorithm**

**from sklearn.linear\_model import**

**LinearRegression**

**regressor=LinearRegression()**

**regressor.fit(x\_train,y\_train)**

**#retriving the intercept**

**print(regressor.intercept\_)**

**#retriving the slope**

**print(regressor.coef\_)**

**#predicting the test results**

**y\_pred=regressor.predict(x\_test)**

**y\_test**

**print(y\_pred)**

**print(y\_test)**

**#evaluting the algorithm**

**from sklearn import metrics**

**import numpy as np**

**print('Mean Absolute**

**Error:',metrics.mean\_absolute\_error(y\_test,y\_pred**

**))**

**print('Mean Squared**

**Error:',metrics.mean\_squared\_error(y\_test,y\_pred)**

**)**

**print('Root Mean Squared Error:',np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))**

**#plot for the train train set**

**plt.scatter(x\_train,y\_train,color='red')#plotting the**

**observation line**

**plt.plot(x\_train,regressor.predict(x\_train),color='bl**

**ue')#plotting the regression line**

**plt.title("product vs cost")**

**plt.xlabel("product")**

**plt.ylabel("cost")**

**plt.show()#specifies end of the graph**

**#plot the test set**

**plt.scatter(x\_test,y\_test,color='red')**

**plt.plot(x\_train,regressor.predict(x\_train),color='bl**

**ue')#plotting the regresion line**

**plt.title("product vs cost")**

**plt.xlabel("product")**

**plt.ylabel("cost")**

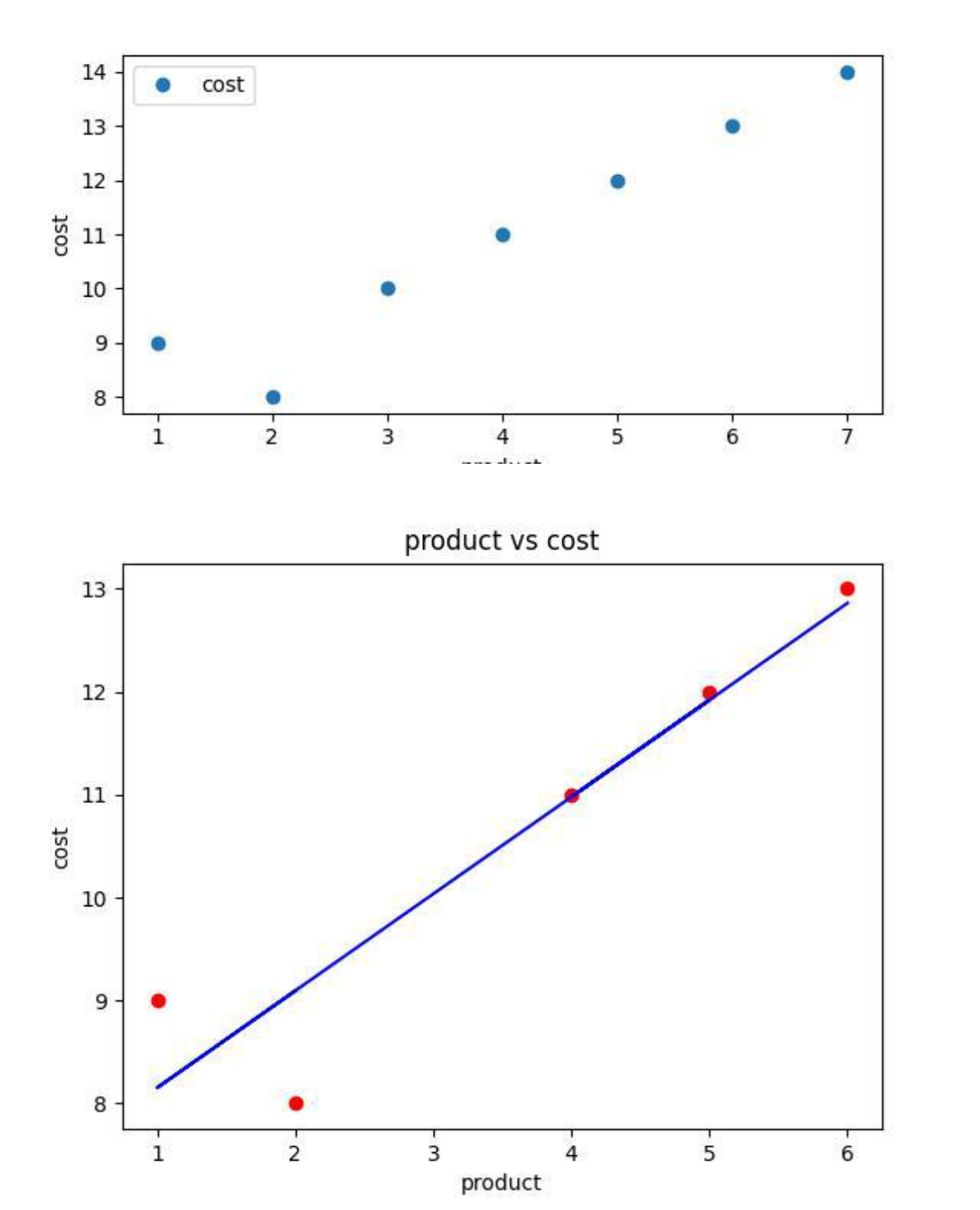
**plt.show()**

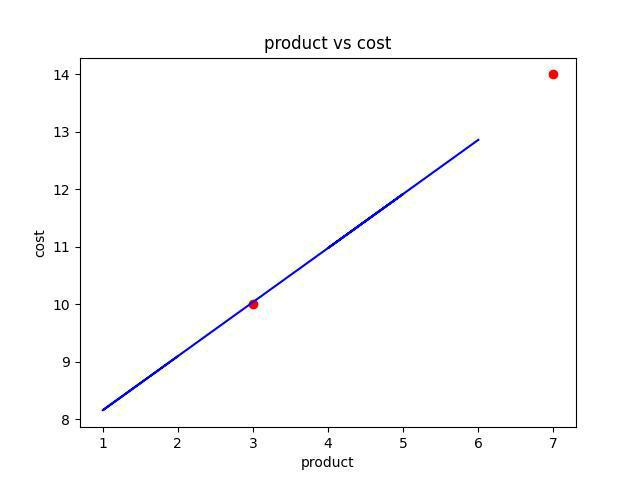
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output:- | |  |  |  |
|  | sno product cost | | | rating |
| 0 | 1 | 1 | 9 | 2 |
| 1 | 2 | 2 | 8 | 3 |
| 2 | 3 | 3 | 10 | 4 |
| 3 | 4 | 4 | 11 | 5 |
| 4 | 5 | 5 | 12 | 6 |
| 5 | 6 | 6 | 13 | 7 |
| 6 | 7 | 7 | 14 | 8 |
|  | product | cost | |  |

* 1 9
* 2 8
* 3 10
* 4 11

4512

5 6 13





(5, 1)

(2, 1)

(5, 1)

(2, 1)

[7.20930233]

[[0.94186047]]

[[13.80232558]

[10.03488372]]

cost

* 14

2 10

Mean Absolute Error:

0.11627906976744118 Mean Squared Error:

0.020146024878312466 Root Mean Squared Error: 0.14193669320620536

**Multiple Regression**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

#from sklearn.linear\_model import LinearRegression as lr

# Load the dataset

data = pd.read\_csv('pra.csv')

* Extract the independent variables (features) X = data[['$cost','rating']].values
* Extract the dependent variable

y = data['discount'].values

# Create and fit the multiple regression model

from sklearn.linear\_model import LinearRegression model = LinearRegression()

model.fit(X, y)

* Generate predicted values predicted\_y = model.predict(X) print("Coefficients: ", model.coef\_) print("Intercept: ", model.intercept\_) print("Predicted values: ", predicted\_y)
* Plot the original data points and the predicted values plt.scatter( X[:, 1], y, c='blue', label='Original data') plt.plot( X[:, 1], predicted\_y, c='red', label='Multipleregression') plt.xlabel('matchpoints')

plt.ylabel('rank') plt.title('Multiple Regression') plt.legend()

plt.show()

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(X[:, 0], X[:, 1], y, c='blue', label='Original data')

ax.plot(X[:, 0], X[:, 1], predicted\_y, c='red', label='Multipleregression')

ax.set\_xlabel('$cost')

ax.set\_ylabel('rating')

ax.set\_zlabel('discount')

ax.set\_title('Multiple Regression')

ax.legend()

plt.show()

**Output**:

Coefficients: [-0.02574741 -0.28450588]

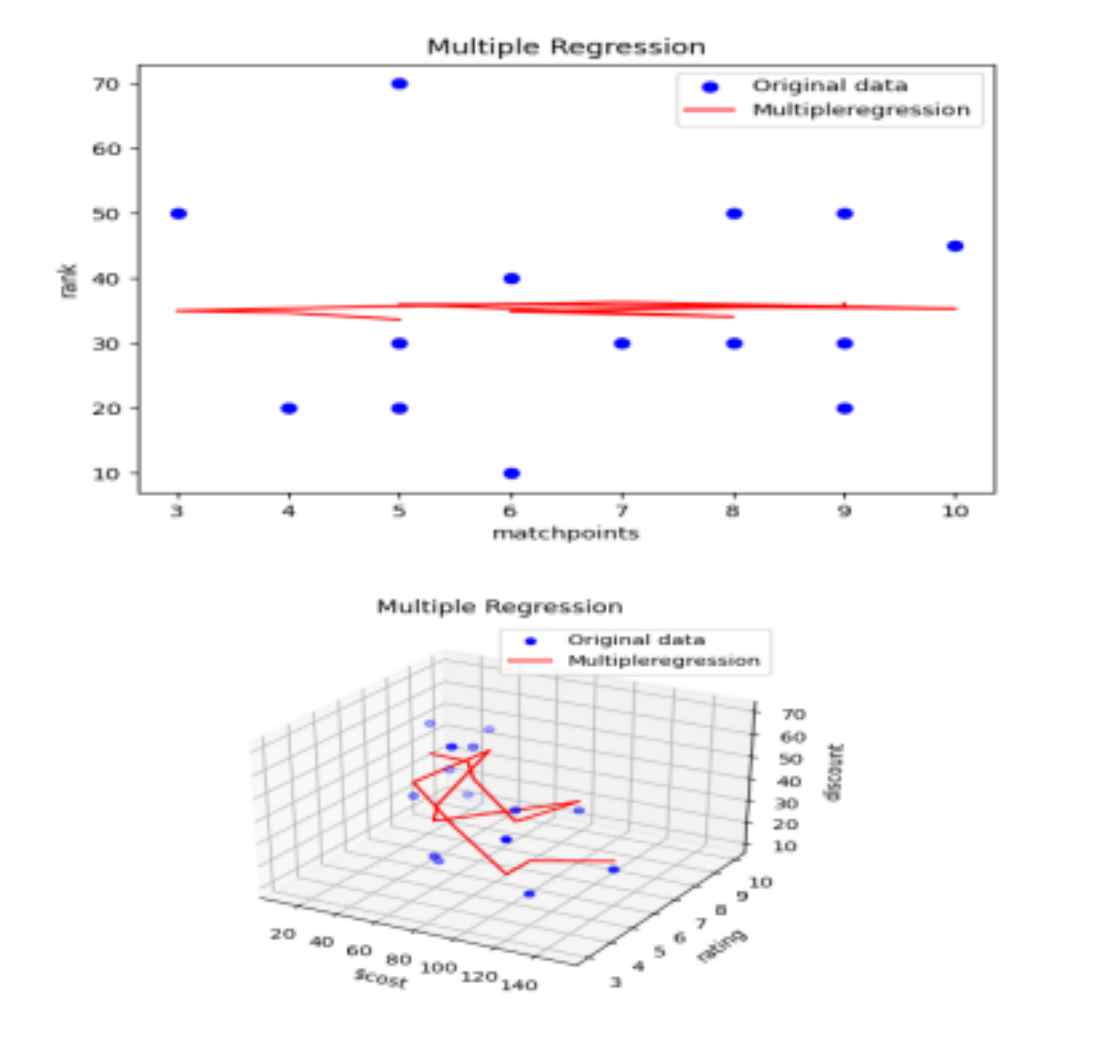
Intercept: 38.95936897090988

Predicted values: [33.67472874 34.73165679 35.01616266 35.73452119 36.32414269 35.34188804

35.96496343 35.99199525 34.10858139 34.9350672 35.5246887 35.62639391

35.88386797 36.14134203]

**Graphs**:



**Polynomial Regression**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

dataset=pd.read\_csv("poly1.csv")

dataset.head()

x=dataset.iloc[:,1:-1].values

y=dataset.iloc[:,-1].values

#plotting the data points

plt.scatter(x,y,color="red")

plt.xlabel("position level")

plt.ylabel("salary")

plt.show()

#Linear regression

from sklearn.linear\_model import LinearRegression lin\_reg=LinearRegression()

lin\_reg.fit(x,y)

#polynomial regression

from sklearn.preprocessing import PolynomialFeatures poly\_reg=PolynomialFeatures(degree=4) x\_poly=poly\_reg.fit\_transform(x) lin\_reg\_2=LinearRegression() lin\_reg\_2.fit(x\_poly,y)

print(x\_poly)

#plotting linear regression

plt.scatter(x,y,color="red")

plt.plot(x,lin\_reg.predict(x),color="blue")

plt.title("Linear Regression")

plt.xlabel("level")

plt.ylabel("Employee need")

plt.show()

#plotting of polynomial regression

plt.scatter(x,y,color="red")

plt.plot(x,lin\_reg\_2.predict(poly\_reg.fit\_transform(x)),color="blue")

plt.title("Polynomial Regression")

plt.xlabel("level")

plt.ylabel("Employee need")

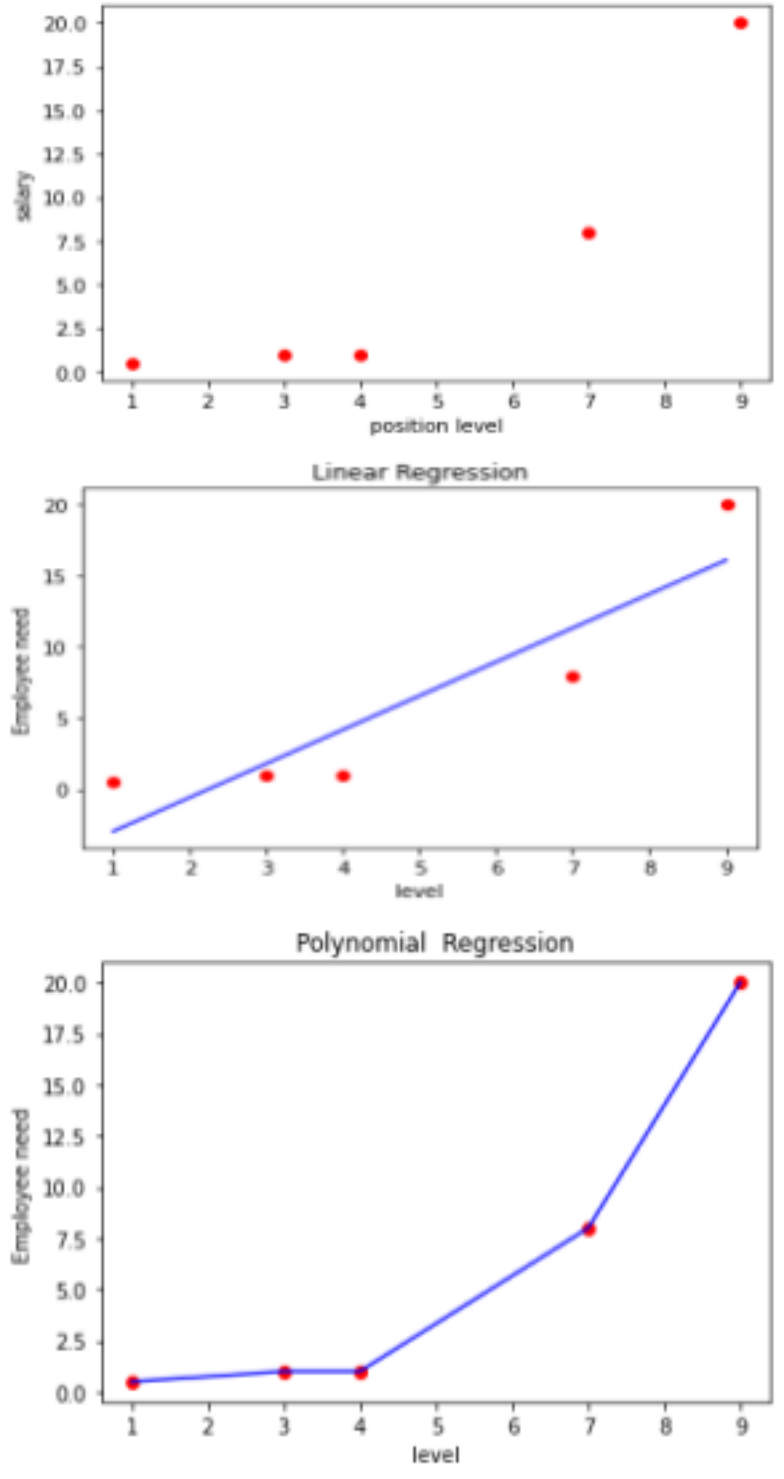
plt.show()

* predicting the final result with the linear Regression model: lin\_pred=lin\_reg.predict([[6.5]])

print(lin\_pred)

* predicting the final result with the polynomial Regression model: poly\_pred=lin\_reg\_2.predict(poly\_reg.fit\_transform([[6.5]])) print(poly\_pred)

**Output:**



[[1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00]

[1.000e+00 3.000e+00 9.000e+00 2.700e+01 8.100e+01]

[1.000e+00 4.000e+00 1.600e+01 6.400e+01 2.560e+02]

[1.000e+00 7.000e+00 4.900e+01 3.430e+02 2.401e+03]

[1.000e+00 9.000e+00 8.100e+01 7.290e+02 6.561e+03]]

[10.14583333]

[5.87706163]

**LAB -09**

**Model Evaluation:**Over-fitting ,under-fitting, Ridge Regression

**Ridge Regression**

from sklearn.linear\_model import Ridge

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

from sklearn.datasets import load\_boston

from sklearn.preprocessing import StandardScaler

* loading boston dataset boston = load\_boston() X = boston.data[:, :13] y = boston.target

print ("Boston dataset keys : \n", boston.keys())

print ("\nBoston data : \n", boston.data)

# scaling the inputs

scaler = StandardScaler()

scaled\_X = scaler.fit\_transform(X)

# Train Test split will be used for both models

X\_train, X\_test, y\_train, y\_test = train\_test\_split(scaled\_X, y, test\_size = 0.3)

# training model with 0.5 alpha value

model = Ridge(alpha = 0.5, normalize = False, tol = 0.001, \ solver ='auto', random\_state = 42)

model.fit(X\_train, y\_train)

# predicting the y\_test

y\_pred = model.predict(X\_test)

# finding score for our model

score = model.score(X\_test, y\_test)

print("\n\nModel score : ", score)

Output:

Boston dataset keys :

dict\_keys(['feature\_names', 'DESCR', 'data', 'target'])

Boston data :

[[6.3200e-03 1.8000e+01 2.3100e+00 ... 1.5300e+01 3.9690e+02 4.9800e+00] [2.7310e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9690e+02 9.1400e+00]

[2.7290e-02 0.0000e+00 7.0700e+00 ... 1.7800e+01 3.9283e+02 4.0300e+00] ...

[6.0760e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 5.6400e+00]

[1.0959e-01 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9345e+02 6.4800e+00]

[4.7410e-02 0.0000e+00 1.1930e+01 ... 2.1000e+01 3.9690e+02 7.8800e+00]] **Model score : 0.6819292026260749**

**Overfitting and underfitting Problem:**

Import numpy as np

Import matplotlib.pypplot as plt

from sklearn.pipeline import Pipeline from sklearn.preprocessing import PolynomialFeatures from sklearn.linear\_model import LinearRegression #this allows us to create a random dataset X = np.sort(np.random.rand(100)) #Lets create a true function

true\_f = lambda X: np.cos(3.5 \* np.pi \* X)

y = true\_f(X) + np.random.randn(100) \* 0.1

degrees = [1,15]

plt.figure(figsize=(15, 10))

for i in range(len(degrees)):

ax = plt.subplot(1, len(degrees), i+1)

plt.setp(ax, xticks=(), yticks=()) polynomial\_features = PolynomialFeatures(degree=degrees[i], include\_bias=False) linear\_regression = LinearRegression() pipeline=Pipeline([("polynomial\_features",polynomial\_features),("linear\_regression", linear\_regression)])

pipeline.fit(X[:, np.newaxis], y) #Testing

X\_test = np.linspace(0, 1, 100)

hat = pipeline.predict(X\_test[:, np.newaxis])

plt.plot(X\_test, hat,label="Model")

plt.plot(X\_test, true\_f(X\_test), label="True function") plt.scatter(X, y, label="Samples")

plt.xlabel("x") plt.ylabel("y")

plt.xlim((0, 1))

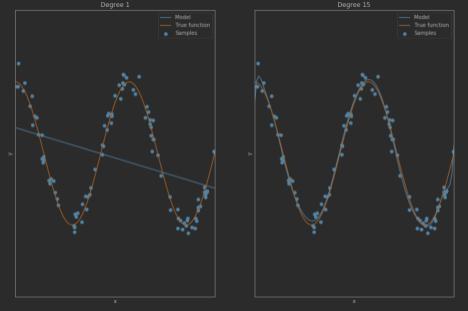
plt.ylim((-2, 2))

plt.legend(loc="best")

plt.title("Degree %d" % degrees[i])

plt.show()

**Output:**



**LAB - 09**

**Introduction to Visualization Tools**

**code:**

import pandas as pd

import matplotlib.pyplot as plt

data=pd.read\_csv(r"C:\Users\YAMUNA370\Downloads\toy\_dataset\toy\_dataset.csv",index \_col=0) print(data)

#index

print(data.index)

#prints the first 20 Records

print(data.head(20))

#Prints the last 20 Records

print(data.tail(20))

#prints the dimension of dataframe

print(data.ndim)

x=(data[['City']])

print(x)

y=(data[['Income']])

print(y)

data.plot(kind='scatter',x='City',y='Gender',color='red')

plt.title('City vs Income')

plt.show()

data.plot(kind='scatter',x='Age',y='Gender',color='red')

plt.title('Age vs Income')

plt.show()

**Output:**

Number City Gender Age Income Illness

149981 Austin Female 53 89384.0 Yes

149982 Austin Male 50 77134.0 No

149983 Austin Female 47 92157.0 No

149984 Austin Female 25 92482.0 No

149985 Austin Male 51 99075.0 No

149986 Austin Female 25 74947.0 No

149987 Austin Female 63 80381.0 No

149988 Austin Female 55 62501.0 No

149989 Austin Female 26 77823.0 No

149990 Austin Male 52 83688.0 No 149991

Austin Female 26 82163.0 No 149992

Austin Male 51 97510.0 No 149993 Austin

Male 37 88408.0 No 149994 Austin Male 64

89906.0 No

149995 Austin Female 37 106097.0 No

149996 Austin Male 48 93669.0 No 149997

Austin Male 25 96748.0 No 149998 Austin

Male 26 111885.0 No 149999 Austin Male 25

111878.0 No 150000 Austin Female 37

87251.0 No 2

City

Number

* Dallas

2 Dallas

3 Dallas

4 Dallas

5 Dallas

...

149996 Austin

149997 Austin

149998 Austin

149999 Austin

150000 Austin

[150000 rows x 1 columns]

Income

Number

* 40367.0

2 45084.0

3 52483.0

4 40941.0

5 50289.0

...

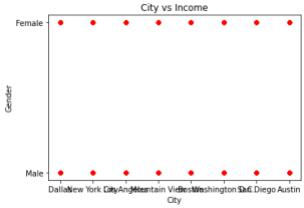
149996 93669.0

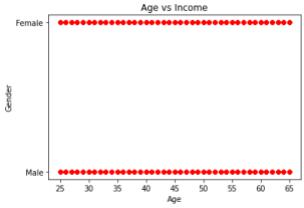
149997 96748.0

149998 111885.0

149999 111878.0

150000 87251.0





**LAB - 10**

**Basic Visualization Tools:**Area plots,Histogram,Bar Charts

BAR PLOT:-

import matplotlib.pyplot as plt

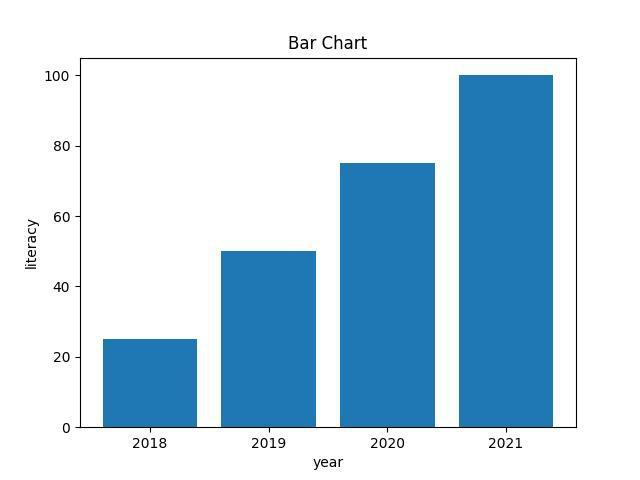
# Sample data

year= ['2018', '2019', '2020', '2021']

literacy = [25, 50, 75, 100]

* Create a bar chart plt.bar(year, literacy)
* Add labels and title plt.xlabel('year') plt.ylabel('literacy') plt.title('Bar Chart')

* Display the chart plt.show() Output:-



LINE PLOT:-

#lineplot

# Sample data

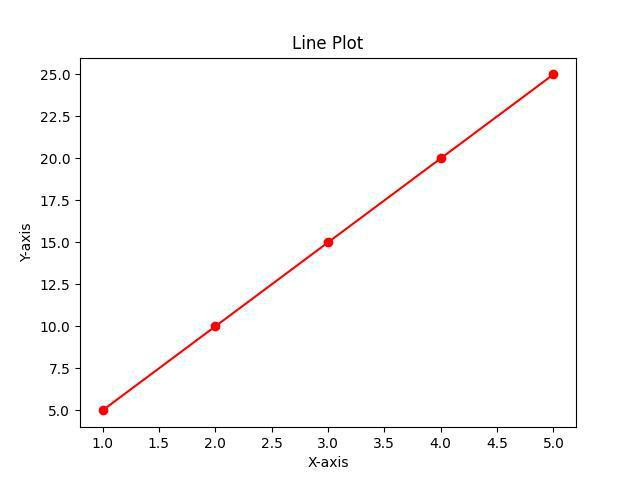
x = [1, 2, 3, 4, 5]

y = [5, 10, 15, 20, 25]

* Create a figure and axis fig, ax = plt.subplots()
* Plot the data

ax.plot(x, y, marker='o', linestyle='-', color='r')

* Customize the plot ax.set\_xlabel('X-axis') ax.set\_ylabel('Y-axis') ax.set\_title('Line Plot')
* Display the plot plt.show() OUTPUT:-



AREA PLOT:-

#area plot

# Sample data

years = [2010, 2011, 2012, 2013, 2014, 2015]

level1 = [2, 5, 8, 6, 9, 3]

level2 = [1, 3, 2, 7, 5, 4]

level3 = [6, 8, 7, 3, 2, 5]

# Plotting

plt.stackplot(years, level1, level2, level3, labels=['level 1', 'level 2', 'level 3'])

plt.legend(loc='upper left')

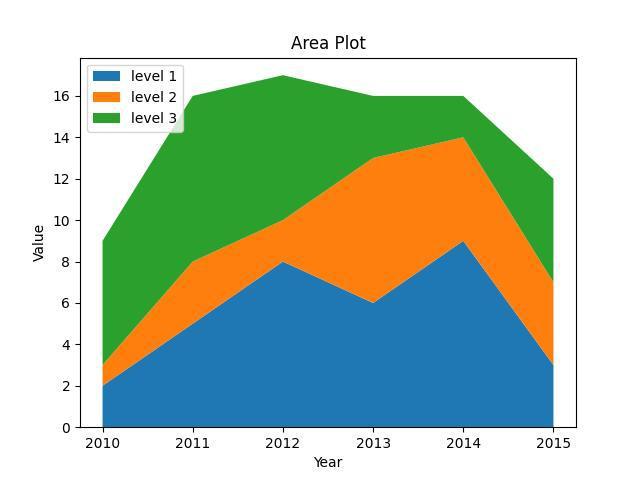
plt.xlabel('Year')

plt.ylabel('Value')

plt.title('Area Plot')

plt.show()

OUTPUT:-



HISTOGRAM:-

#histogram

import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, alpha=0.5, color='steelblue')

plt.xlabel('year')

plt.ylabel('literacy')

plt.title('Histogram')

plt.show()

#pie chart

x=[2010,2012,2013,2014,2015]

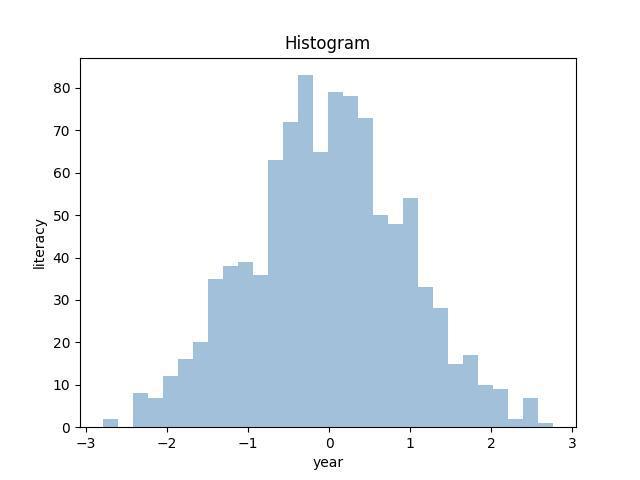
y=[20,30,40,50,60]

plt.pie(y,labels=x,autopct='%1.1f%%')

plt.axis("equal")

plt.show()

OUTPUT:-



**#pie chart**

**x=[2010,2012,2013,2014,2015]**

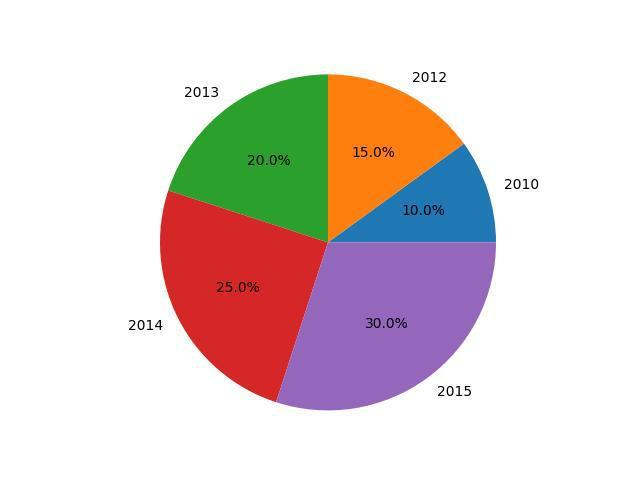
**y=[20,30,40,50,60]**

**plt.pie(y,labels=x,autopct='%1.1f%%')**

**plt.axis("equal")**

**plt.show()**

Output:-



**Bubble plot:**

**#Bubble plot**

**x=np.random.rand(50)**

**y=np.random.rand(50)**

**sizes=np.random.rand(50)\*100**

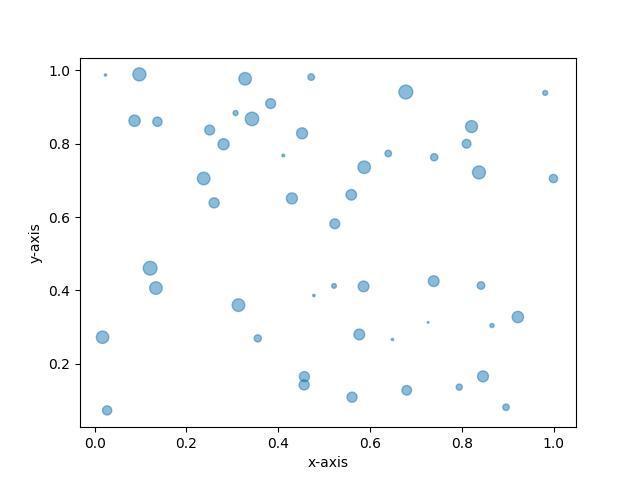
**plt.scatter(x,y,s=sizes,alpha=0.5,marker='o')**

**plt.xlabel("x-axis")**

**plt.ylabel("y-axis")**

**plt.show()**

**Output:**



**LAB -12**

**Advanced Visualization Tools:**Waffle charts,Word Clouds Seaborn and Regression plots

**Waffle charts:**

**import pandas as pd**

**import matplotlib.pyplot as plt from pywaffle import Waffle data={'vehicles':['car', 'lorry','bus','bike','bicycle'],**

**'stock':[25,20,15,10,5]}**

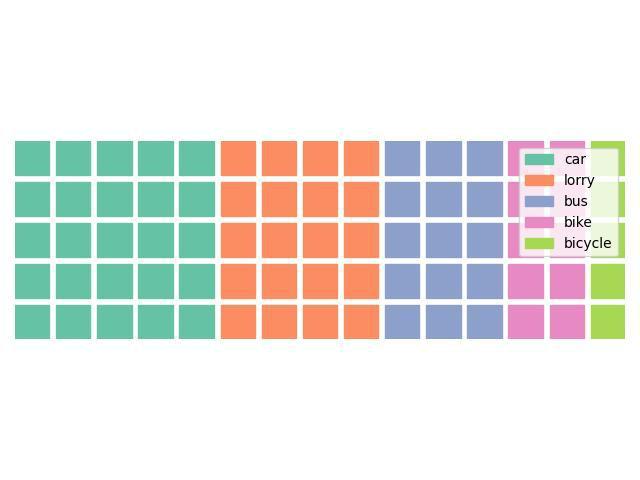
**df=pd.DataFrame(data)**

**fig=plt.figure(FigureClass=Waffle,rows=5,values=df.stock,**

**labels=list(df.vehicles))**

**plt.show()**

Output:-



**Word clouds**

#word cloud

from wordcloud import WordCloud

import matplotlib.pyplot as plt

text="hi hello welcome to word cloud it is very easy to understand"

wc=WordCloud().generate(text)

plt.imshow(wc)

plt.axis("off")

plt.show()

**Output:**



**Seaborn :**

**import seaborn as sns**

**sns.set(style="ticks")**

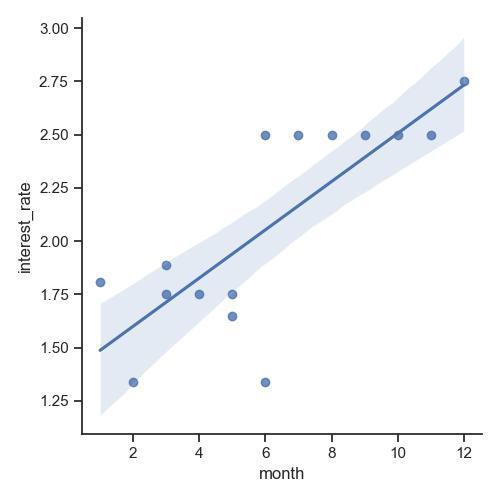
**df=pd.read\_csv("cham.csv")**

**print(df)**

**sns.lmplot(x="month",y="interest\_rate",data=df)**

**plt.show()**

**Output:**



**Barplot:**

**import seaborn as sns**

**df=pd.read\_csv("cham.csv")**

**sns.barplot(x="month",y="interest\_rate",data=df)**

**plt.show()**

**Output:-**

