DATA SCIENCE WITH PYTHON

STUDENT NAME: DASARI GANGADHAR

ID NUMBER: N190302

CLASS: CSE-03

DEPARTMENT OF COMPUTER SCIENCE AND ENGINERING

RGUKT-NUZ-AP E2-SEMISTER-

II, AY-2022-23

Table of Contents :

LAB-01

AIM : a) Python Basics: Your first program, Types Expressions and Variables String Operations

Code:

```
print("hello world")

color="green"
print(type(color))

a=3
print(a,type(a))

b=-3.5
print(b,type(b))

c=2+3j
print(type(c))

d,e,f=2,3,-4
print(f)
print(e)

print(d)

h=j=k="RAJA"
print(h,j,k)

id1='How are you?'
print(id1[1:7])

x=0b11
print(type(x))

val=None
print(val)

#python string
id1="Mariya babu"
print(id1[1])

#negative indexing
print(id1[-3])
```

```
#id1[3]=q

#multiline strings
string="""mariya babu is the roommate of durgaprasa
Hari is friend of mariya"""
print(string)

#python string operation
id2=" is the roommate of Durgaprasad"
print(id1+id2)

id3="babu"
id4="babu"
print(id3==id4)

id3="babu"
id4="babu1"
print(id3==id4)


#iterton
gr='welcome'
for letter in gr:
    print(letter)

gr='welcome'
for letter in gr:
    print(gr)
print(len(gr))

#membership
print("a" in gr)
print("a" not in gr)

print(gr.upper())
print(gr.lower())
print(gr.startswith("h"))

id='name'
name='Gangadhar'
print(f'my {id} is {name}')


#escape sequence
ex="he said,\"what's is there?\""
print(ex)
```

output:

```
hello world
<class 'str'>
3 <class 'int'>
-3.5 <class 'float'>
<class 'complex'>
-4
3
2
RAJA RAJA RAJA
ow are
<class 'int'>
None

a
a
mariya babu is the roommate of durgaprasa
Hari is friend of mariya
Mariya babu is the roommate of Durgaprasad
True
False
w
e
l
c
o
m
e
welcome
welcome
welcome
welcome
welcome
welcome
welcome
7
False
True
WELCOME
welcome
False
my name is Gangadhar
he said,"what's is there?"
```

LAB-02

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

CODE

""" list,tuple,dic,set"""

```
a=[2,'a','aba','aaa']
print(a)
num=(1,5,3)
print(num)
b={'a':3,'ba':456,'a':4}
print(b)
c={1,4,3,2,5,}
print(c)
d={2,'a','aba','aaa'}
print(d)
lan=["telugu","tamil","kannada"]
print(lan[2])
print(type(lan))
e={2,2,2,3}
print(e)
a=True
print(a)
b=False
print(b)

#list
a=[4,6,7]
print(a)

print(a[0])
print(a[-3])

print(a[0:2])

#append


a.append(2)
print(a)

#extend
```

```
b=[8,9,7]
a.extend(b)
print(a)

a[0]=0
print(a)

#del
del b[1]
print(b)
a.remove(0)

a.sort()
print(a)
a.reverse()
print(a)

a.pop(2)
print(a)
#checking
print(1 in a)
print(len(a))

#list comprehension
c=[]
for x in range(1,6):
    c.append(x*x)
print(c)
```

#tuple

```
print("tuples")
a=(3,4,5)
print(a)


b="hello",
print(type(b))

c=("hello")
print(type(c))

#tuple accessing
print(a[-1])
print(a[1])
print(a[0:2])
```

```
#tuple methods
d=(6,5,7,7,7,8,4,9,0)

print(d.count(7))
print(d.index(6))

#iteration

for x in d:
   print(x)

print(7 in d)
```

#sets

```
a={3,5,6,7,8,9,4,5,6}
b={10,20,30,40}

print("set")
print(a)
print(type(a))


a.add(10)
print(a)
#min
print(min(a))

#max
print(max(a))

#len
print(len(a))

#all
print(all(a))

#any
print(any(a))

#enumerate
print(enumerate(a))

#sum
print(sum(a))
```

```
#sorted
print(sorted(a))


#union
print(a|b)


print(a.union(b))
#intersection


print(a&b)
print(a.intersection(b))


#symmetric difference
print(a^b)

#equal
print(a==b)
```

#dictonary

```
dic={1:"a",2:"b",3:"c",4:"d",5:"e"}
print(dic)
print(type(dic))

#adding
dic[6]="f"
print(dic)

#changing
dic[3]="C"
print(dic)

#accessing
print(dic[3])

#remove

del dic[6]
print(dic)
```

```
# sorted
sorted(c)
print(dic)

#membership

print(1 in dic)
print(4 not in dic)
```

output:

```
[2, 'a', 'aba', 'aaa']
(1, 5, 3)
{'a': 4, 'ba': 456}
{1, 2, 3, 4, 5}
{2, 'aba', 'a', 'aaa'}
kannada
<class 'list'>
{2, 3}
True
False
 [4, 6, 7]
4
4
[4, 6]
[4, 6, 7, 2]
[4, 6, 7, 2, 8, 9, 7]
[0, 6, 7, 2, 8, 9, 7]
[8, 7]
[2, 6, 7, 7, 8, 9]
[9, 8, 7, 7, 6, 2]
[9, 8, 7, 6, 2]
False
5
[1, 4, 9, 16, 25]
tuples
(3, 4, 5)
<class 'tuple'>
<class 'str'>
5
4
(3, 4)
3
0
```

```
6
5
7
7
7
8
4
9
0
True
set
{3, 4, 5, 6, 7, 8, 9}
<class 'set'>
{3, 4, 5, 6, 7, 8, 9, 10}
3
10
8
True
True
<enumerate object at 0x000001803DC96E80>
52
[3, 4, 5, 6, 7, 8, 9, 10]
{3, 4, 5, 6, 7, 8, 9, 10, 40, 20, 30}
{3, 4, 5, 6, 7, 8, 9, 10, 40, 20, 30}
{10}
{10}
{3, 4, 5, 6, 7, 40, 8, 9, 20, 30}
False
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
<class 'dict'>
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
{1: 'a', 2: 'b', 3: 'C', 4: 'd', 5: 'e', 6: 'f'}
C
{1: 'a', 2: 'b', 3: 'C', 4: 'd', 5: 'e'}
{1: 'a', 2: 'b', 3: 'C', 4: 'd', 5: 'e'}
True
False
```

LAB-03

AIM: Python Programming Fundamentals: Conditions and Branching Loops, Functions,Objects and Classes

Code:

```
a=30
b=60
c=40
```

#if else

```
if(a<b):
   print(a)
else:
   print(b)
```

#if elif else

```
if(a>b):
   print("greater",a)
elif(b>c):
   print("greater",b)
else:
   print("greater",c)
```

#nested if

```
d=60
if(a>b):
   if(a>c):
      if(a>d):
         print("greater",a)
elif(b>a):
   if(b>c):
      if(b>d):
         print(" greater",b)
elif(c>a):
   if(c>b):
      if(c>d):
         print("greatest",c)
```

```
else:
   print("greater",c)
#shorthand if
print("a is less than b") if(a<b) else print("b is less")
```

output:

```
30
greater 60
a is less than b
```

 # FOR LOOP

```
print("for loop in python")
name="DASARI GANGADHAR"
print(name)
print("printing each character in python")
for i in name:
   print(i)
print("Printing numbers from 1 to 21 with difference of 2 using for loop")
for i in range (1,22,2):
   print(i,end="  ")
print()
```

#while loop

```
print("while loop in python:")
i=1
while i<=4:
   print("dasari",   end="")
   j=1
   while j<=3:
      print("gangadhar",     end="")
      j+=1
   i+=1
   print()
```

outputs:

```
for loop in python
DASARI GANGADHAR
printing each character in python
D
A
```

S
A
R
I

G
A
N
G
A
D
H
A
R
Printing numbers from 1 to 21 with difference of 2 using for loop
1 3 5 7 9 11 13 15 17 19 21
while loop in python:
dasarigangadhargangadhargangadhar
dasarigangadhargangadhargangadhar
dasarigangadhargangadhargangadhar
dasarigangadhargangadhargangadhar

Classes and objects

Code:

```
        class animal:
    def speak(self):
        print("i am speaking")
class dog(animal):
    def bark(self):
        print("i am barking")
d=dog()
d.bark()
d.speak()
```

Output:

```
        i am barking
i am speaking
```

//Built in class objects

Code:

```
        class student:
    def __init__(self,name,id,age) :
        self.name=name
        self.id=id
```

```
        self.age=age
s=student("mani",45,18)
print(getattr(s,'name'))
setattr(s,"age",19)
print(getattr(s,"age"))
print(hasattr(s,'id'))
delattr(s,'age')
```

Output:

```
            mani
19
True
```

INHERITANCE:

Code:

```
class animal:
    def speak(self):
        print("i am speaking")
class dog(animal):
    def bark(self):
        print("i am barking")
class dogchild(dog):
    def eat(self):
        print("i am eating")
d=dogchild()
d.eat()
d.bark()
d.speak()
```

Output:

```
            i am eating
i am barking
i am speaking
```

//Multiple inheritance

Code:

```
class fam:
    def speak(self):
        print("hi i am mani")
class ram(fam):
    def eat(self):
        print("i am eating")
class raj(ram):
    def sleep(self):
        print("i am sleeping ")
```

```
d=raj()
d.sleep()
d.eat()
d.speak()
```

Output:

```
        i am sleeping
i am eating
hi i am mani
```

//Abstarct classes

Code:

```
    from abc import ABC,abstractmethod
class car(ABC):
  def mileage(self):
    pass
class maruthi(car):
  def mileage(self):
    print("the mileage is:30kmph")
class suzuki(car):
  def mileage(self):
    print("the mileage is:25kmph")
class bazaz(car):
  def mileage(self):
    print("the mileage is:35kmph")
m=maruthi()
m.mileage()
s=suzuki()
s.mileage()
b=bazaz()
b.mileage()
```

Output:

```
 the mileage is:30kmph
the mileage is:25kmph
the mileage is:35kmph
base class: 123
```

LAB-04

AIM: Working with Data in Python: Reading files with open, Writing files with open,Loading data with Pandas, Working with and Saving data with Pandas

CODE:

```
import pandas as pd
import numpy as np
print(pd.__version__)
b=[1,2,3,4]
c=pd.Series(b)

print(c)
b=['s','d']
c=pd.Series(b[-1])

print(c)
d=np.array(['a','b','c','d'])
s=pd.Series(d)
r=pd.DataFrame(d)
print(s)
print(r)
print(len(s))
s=pd.Series(d,index=[101,103,103,104])
j=pd.Series(d,index=["x","y","z","w"])
print(s)
print(j)

dataset={'icecreams':['vanila','strawberry','badam','pista'],
        'rating':[4.5,3.8,4.2,4.6]
}

ds=pd.DataFrame(dataset)
print(ds)

ds=pd.Series(dataset)
print(ds)
```

Output:

```
2.0.1
0    1
```

```
1    2
2    3
3    4
dtype: int64
0    d
dtype: object
0    a
1    b
2    c
3    d
dtype: object
   0
0  a
1  b
2  c
3  d
4
101    a
103    b
103    c
104    d
dtype: object
x    a
y    b
z    c
w    d
dtype: object
    icecreams  rating
0      vanila     4.5
1  strawberry     3.8
2       badam     4.2
3       pista     4.6
icecreams    [vanila, strawberry, badam, pista]
rating                 [4.5, 3.8, 4.2, 4.6]
dtype: object
```

Attribute of series

```
import pandas as pd
import numpy as np

ds=np.array(['a','b','c','d'])
d=pd.Series(ds)
print(d)
```

```
d=pd.Series(ds ,index=[101,102,103,"e"])
print(d)
print(d[103])
ds1={'d1':100,'d2':200,'d3':300}
d=pd.Series(ds1)
print(d)
j=pd.Series(ds1,index=['d1','d2'])
print(j)
print(j.name)
print(j.values)
print(j.size)
print(d.shape)
print(d.ndim)
print(d.nbytes)
print(d.memory_usage)
print(j.empty)
j.name='raj'
print(j.name)
```

output:

```
0    a
1    b
2    c
3    d
dtype: object
101    a
102    b
103    c
e      d
dtype: object
c
d1    100
d2    200
d3    300
dtype: int64
d1    100
d2    200
dtype: int64
None
[100 200]
2
(3,)
1
24
<bound method Series.memory_usage of d1    100
d2    200
```

```
d3    300
dtype: int64>
False
raj
```

Multiplication of series :

```
import pandas as pd
import numpy as np

ds1=np.array([1,1,2,3,4])
d1=pd.Series(ds1)
ds2=np.array([2,2,3,4,5])
d2=pd.Series(ds2)
a=d1.add(d2)
print(a)
b=d1.sub(d2)
print(b)
c=d1.mul(d2)
print(c)
d=d1.multiply(4)
print(d)
e=d1.div(d2)
print(e)
f=d2.mod(d1)
print(f)
g=d2.pow(3)
print(g)
h=d2.le(d1)
print(h)
i=d2.gt(d1)
print(i)
j=d2.equals(d1)
print(j)
```

output:

```
0    3
1    3
2    5
3    7
4    9
dtype: int32
0   -1
1   -1
2   -1
```

```
3   -1
4   -1
dtype: int32
0     2
1     2
2     6
3    12
4    20
dtype: int32
0     4
1     4
2     8
3    12
4    16
dtype: int32
0    0.500000
1    0.500000
2    0.666667
3    0.750000
4    0.800000
dtype: float64
0    0
1    0
2    1
3    1
4    1
dtype: int32
0      8
1      8
2     27
3     64
4    125
dtype: int32
0    False
1    False
2    False
3    False
4    False
dtype: bool
0    True
1    True
2    True
3    True
4    True
dtype: bool
False
```

LAB-05

Aim: Working with Numpy Arrays: Numpy 1d Arrays, Numpy 2d Arrays

Code:

```
import numpy as np
from numpy import random

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

b=np.array([[1,2,3,4,5],[6,7,8,9,0]])
print(b)

c=np.array([[[1,2,3],[4,5,6],[7,0,9]]])
print(c)

d=np.array(32)
print(d)

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)

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

f=np.array([5,6],ndmin=3)
print(f)
print(f.ndim)

print(b[1,2])
#slicing
print(a[0:2])
print(a[2:])
print(a[:3])
print(a[-4:-2])
print(a[1:4:2])
print(a[1:4:3])
print(a[::1])
print(b[1,0:3:2])

g=np.array([1,2,3,4],dtype='S')
print(g)
```

```
print(b[1,0::3])

print(type(g))
print(g.dtype)

i=np.array([1.1,2.2,3.3,4.4])
print(i)
j=i.astype('i')
print(j)
print(i)


a=([1,3,4],[5,6,7])
b=np.asarray(a,order='f')
print(b)
for i in np.nditer(b):
    print(i)

a=np.zeros((5,2 ),dtype=int)
print(a)
b=np.full([2,3],56 ,dtype=float)
print(b)

c=np.ones(([4,2]),dtype=int)
print(c)
x=random.randint(10000)
print(x)
for i in range(1,5):
    x=random.randint(10)
    print(x)

d=np.eye(5,3 ,dtype=int, k=-1)
print(d)
a=np.eye(3,3, dtype=int)
print(a)
b=np.asarray(a,order='f')
for i in np.nditer(b):
    print(i)

#captcha
x=random.randint(10000)
print(x)
c=int(input('enter the capctha'))
while(c!=x):
    print("invalid captcha")
    c=int(input('enter'))
```

```
print("valid")

c=random.rand(3,2)
print(c)
d=random.ranf([3,2])
print(d)
```

output:

```
[1 2 3 4]
[[1 2 3 4 5]
 [6 7 8 9 0]]
[[[1 2 3]
  [4 5 6]
  [7 0 9]]]
32
1
2
3
0
[[[[[1 2 3 4]]]]]
[[[5 6]]]
3
8
[1 2]
[3 4]
[1 2 3]
[1 2]
[2 4]
[2]
[1 2 3 4]
[6 8]
[b'1' b'2' b'3' b'4']
[6 9]
<class 'numpy.ndarray'>
|S1
[1.1 2.2 3.3 4.4]
[1 2 3 4]
[1.1 2.2 3.3 4.4]
[[1 3 4]
 [5 6 7]]
1
5
3
6
4
```

```
7
[[0 0]
 [0 0]
 [0 0]
 [0 0]
 [0 0]]
[[56. 56. 56.]
 [56. 56. 56.]]
[[1 1]
 [1 1]
 [1 1]
 [1 1]]
5425
7
2
4
3
[[0 0 0]
 [1 0 0]
 [0 1 0]
 [0 0 1]
 [0 0 0]]
[[1 0 0]
 [0 1 0]
 [0 0 1]]
1
0
0
0
1
0
0
0
1
7479
enter the capctha7479
valid
[[0.14702871 0.94097438]
 [0.80805663 0.52615084]
 [0.45495018 0.4452953 ]]
[[0.99567496 0.61726301]
 [0.44050543 0.35901677]
 [0.69665999 0.3356309 ]]
```

LAB-06

Aim: Importing Datasets: Learning Objectives, Understanding the Domain, Understandingthe Dataset, Python package for data science, Importing and Exporting Data in Python, BasicInsights from Datasets
Cleaning and Preparing the Data: Identify and Handle Missing Values, Data Formatting, Data Normalization Sets, Binning, Indicator variables

Code:

Importing datasets and preparing the data

```
import pandas as pd
df=pd.read_csv(r'C:\Users\DASARI GANGADHAR\Desktop\DSP\data1.csv')
d=pd.DataFrame(df)
print(d)
d=df.loc[4]
print(d)
d=df.loc[2:3]
print(d)
print(df.loc[1,"Name"])
print(df.loc[0:4,["Name","marks"]])
print(df.loc[4:8,"Name":"marks"])

"""ILOC"""

print(df.iloc[3])
print(df.iloc[3:8])
print(df.iloc[3:8,1])
print(df.iloc[5:9,1:3])
print(df.iloc[[2,4,6,7]])
```

output:

```
Unnamed: 0       Name    id   marks
0           1     Dasari  R1254     14
1           2  Gangadhar   R1255     14
2           3       Sree  R1256     13
3           4        Raj  R1257     12
4           5        Ram  R1258     15
5           6       Roja  R1259     13
6           7      Rahul  R1260     14
7           8      Ramya  R1261     11
8           9       Siri    NaN     12
```

```
9         10       Lava  R1263     10
Unnamed: 0       5
Name          Ram
id          R1258
marks          15
Name: 4, dtype: object
  Unnamed: 0  Name    id   marks
2          3  Sree  R1256     13
3          4   Raj  R1257     12
Gangadhar
        Name  marks
0     Dasari     14
1  Gangadhar     14
2       Sree     13
3        Raj     12
4        Ram     15
   Name    id   marks
4   Ram  R1258     15
5   Roja  R1259     13
6  Rahul  R1260     14
7  Ramya  R1261     11
8   Siri    NaN     12
Unnamed: 0       4
Name          Raj
id          R1257
marks          12
Name: 3, dtype: object
  Unnamed: 0   Name    id   marks
3          4    Raj  R1257     12
4          5    Ram  R1258     15
5          6   Roja  R1259     13
6          7  Rahul  R1260     14
7          8  Ramya  R1261     11
3     Raj
4     Ram
5    Roja
6   Rahul
7   Ramya
Name: Name, dtype: object
   Name    id
5   Roja  R1259
6  Rahul  R1260
7  Ramya  R1261
8   Siri    NaN
  Unnamed: 0   Name    id   marks
2          3   Sree  R1256     13
```

```
4          5    Ram  R1258     15
6          7  Rahul  R1260     14
7          8  Ramya  R1261     11
```

Data cleaning

dropna()

```
import pandas as pd
import numpy as np
df=pd.read_csv(r'C:\Users\DASARI GANGADHAR\Desktop\DSP\data1.csv')
print(df)
d=df.dropna()
print(d)
print(df)
print(df.loc[:,["marks","Name"]].dropna())
d=df.dropna(inplace=True)
print(d)
print(df)
```

output:

```
 Unnamed: 0        Name    id   marks
0          1      Dasari  R1254     14
1          2  Gangadhar   R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
8          9       Siri    NaN     12
9         10       Lava  R1263     10
   Unnamed: 0        Name    id   marks
0          1      Dasari  R1254     14
1          2  Gangadhar   R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
9         10       Lava  R1263     10
   Unnamed: 0        Name    id   marks
0          1      Dasari  R1254     14
1          2  Gangadhar   R1255     14
```

```
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
8          9       Siri    NaN     12
9         10       Lava  R1263     10
   marks       Name
0     14     Dasari
1     14  Gangadhar
2     13       Sree
3     12        Raj
4     15        Ram
5     13       Roja
6     14      Rahul
7     11      Ramya
8     12       Siri
9     10       Lava
None
   Unnamed: 0       Name    id   marks
0          1     Dasari  R1254     14
1          2  Gangadhar   R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
9         10       Lava  R1263     10
```

fillna()

```
import pandas as pd
df=pd.read_excel(r"C:\Users\DASARI GANGADHAR\Desktop\DSP\data2.xlsx")
print(df)
d=df.fillna("missing")
print(d)
df.fillna("missing",inplace=True)
print(df)
```

output:

```
name  gender   age  weight
0   John      M  48.0   128.6
```

```
1  Peter    NaN  58.0   158.3
2    Liz      F   NaN   115.5
3    Joe      M  28.0   170.1
  name    gender     age  weight
0   John       M     48.0   128.6
1  Peter  missing     58.0   158.3
2    Liz       F  missing   115.5
3    Joe       M     28.0   170.1
  name    gender     age  weight
0   John       M     48.0   128.6
1  Peter  missing     58.0   158.3
2    Liz       F  missing   115.5
3    Joe       M     28.0   170.1
```

isnull()

```
import pandas as pd
import numpy as np
df=pd.read_excel(r"data2.xlsx")

print(df)
print(df.isnull())
print(df.notnull())
d=df.replace(to_replace="Liz",value="Loe")
print(d)
di=df.interpolate(method="linear",limit_direction="forward")
print(di)
```

output:

```
 Unnamed: 0       Name    id   marks
0          1     Dasari  R1254     14
1          2  Gangadhar   R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
8          9       Siri    NaN     12
9         10       Lava  R1263     10
   Unnamed: 0       Name    id   marks
0          1     Dasari  R1254     14
1          2  Gangadhar   R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
```

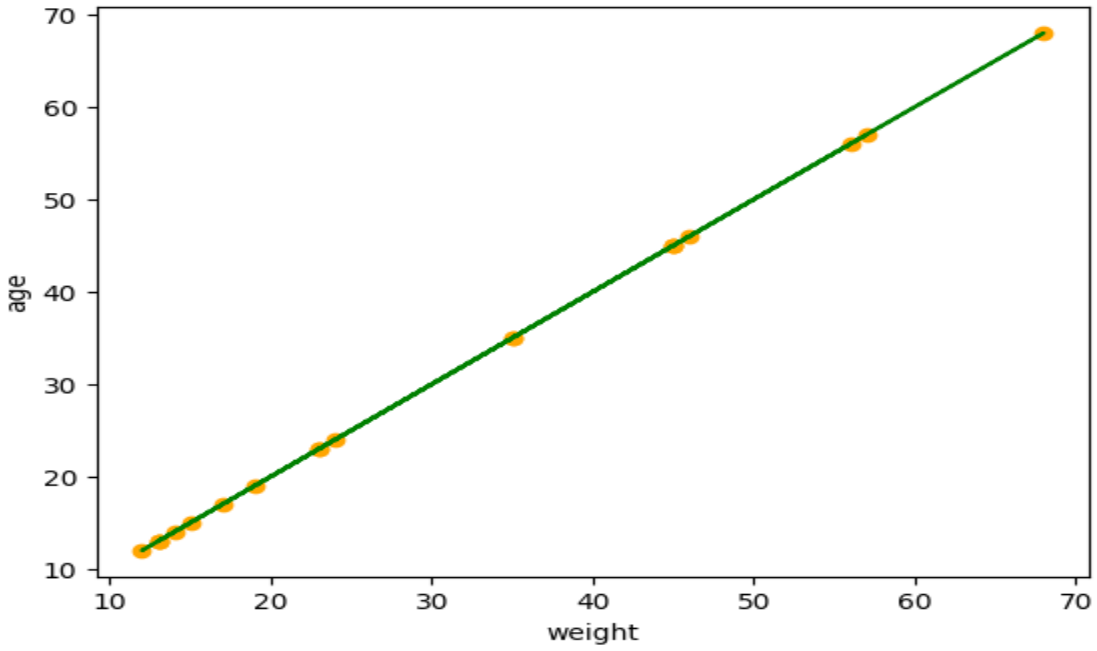
```
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
9         10       Lava  R1263     10
   Unnamed: 0       Name    id   marks
0          1     Dasari  R1254     14
1          2  Gangadhar  R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
8          9       Siri    NaN     12
9         10       Lava  R1263     10
   marks       Name
0     14     Dasari
1     14  Gangadhar
2     13       Sree
3     12        Raj
4     15        Ram
5     13       Roja
6     14      Rahul
7     11      Ramya
8     12       Siri
9     10       Lava
None
   Unnamed: 0       Name    id   marks
0          1     Dasari  R1254     14
1          2  Gangadhar  R1255     14
2          3       Sree  R1256     13
3          4        Raj  R1257     12
4          5        Ram  R1258     15
5          6       Roja  R1259     13
6          7      Rahul  R1260     14
7          8      Ramya  R1261     11
9         10       Lava  R1263     10
```

LAB-07

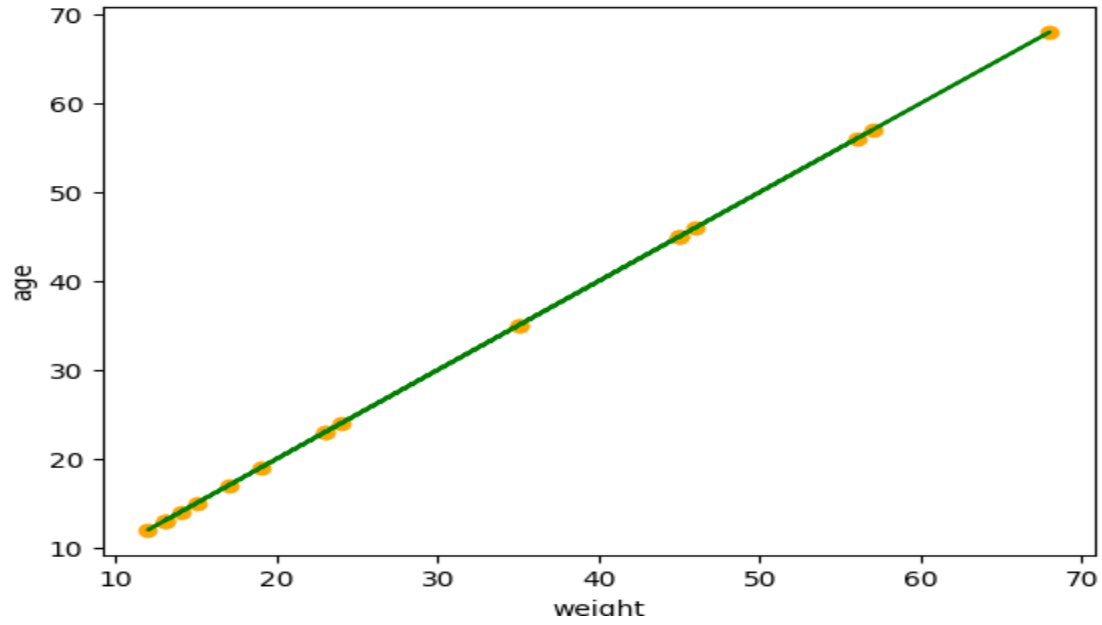
Aim: Model Development: Simple and Multiple Linear Regression, Model EvaluationUsingVisualization, Polynomial Regression and Pipelines, R-squared and MSE for In-Sample Evaluation, Prediction and Decision Making

CODE: simple linear regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

dataset=pd.read_excel('aw.xlsx')

dataset.head()

dataset.isna().sum()

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

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(x_train,y_train)

y_pred=regressor.predict(x_test)

plt.scatter(x_train,y_train,color="orange")
plt.plot(x_train,regressor.predict(x_train),color="green")
plt.xlabel('weight')
plt.ylabel('age')
plt.show()
```




```
plt.scatter(x_test,y_test,color="orange")
plt.plot(x_test,regressor.predict(x_test),color="green")
plt.xlabel('weight')
plt.ylabel('age')
plt.show()
```



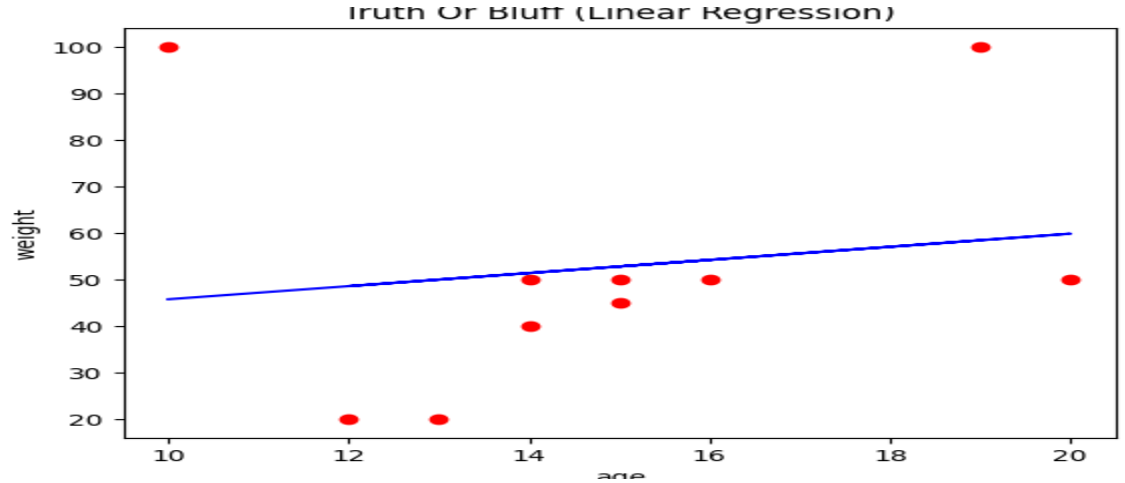
Code: Multiple linear regression

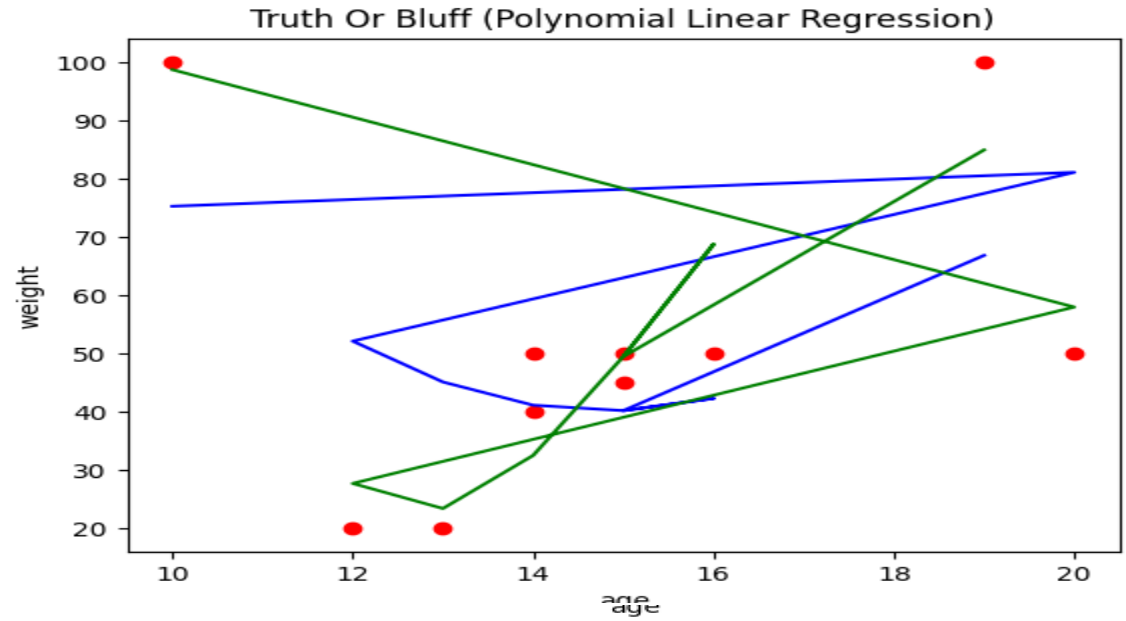
```
#importing pandas
import pandas as pd
#importing data set
df=pd.read_csv("class1.csv")
#making list of independent variales as x and dependent variable as y
X= df[['Height','Age']]
y = df['Weight']
#to import this sklearn  pip install -U scikit-learn
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X, y)
predictedCO2 = regr.predict([[2300, 1300]])
print(predictedCO2)
print(regr.coef_)
predictedCO2 = regr.predict([[3300, 1300]])
print(predictedCO2)
```

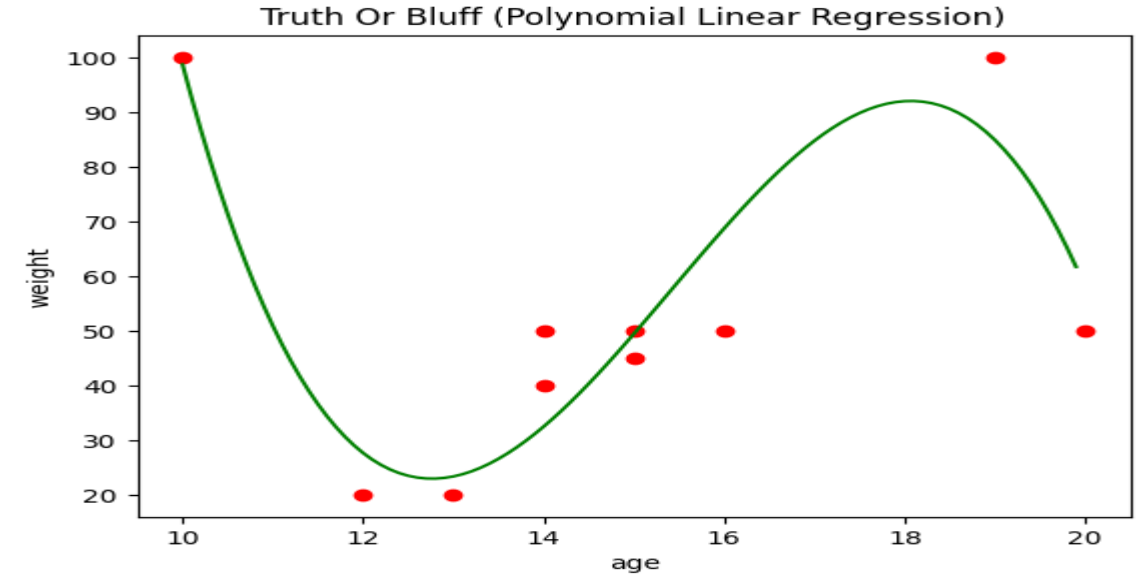
CODE: polynomial regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('ageweight.csv')

df

df.describe()

X=df.iloc[:,1:2].values


y=df.iloc[:,2].values


from sklearn.linear_model import LinearRegression
lin_reg=LinearRegression()
lin_reg.fit(X,y)


from sklearn.preprocessing import PolynomialFeatures
poly_reg2=PolynomialFeatures(degree=2)
```

```
X_poly=poly_reg2.fit_transform(X)
lin_reg_2=LinearRegression()
lin_reg_2.fit(X_poly,y)

poly_reg3=PolynomialFeatures(degree=3)
X_poly3=poly_reg3.fit_transform(X)
lin_reg_3=LinearRegression()
lin_reg_3.fit(X_poly3,y)


plt.scatter(X,y,color='red')
plt.plot(X,lin_reg.predict(X),color='blue')
plt.title('Truth Or Bluff (Linear Regression)')
plt.xlabel('age')
plt.ylabel('weight')
plt.show()


plt.scatter(X,y,color='red')
plt.plot(X,lin_reg_2.predict(poly_reg2.fit_transform(X)),color='blue')
plt.plot(X,lin_reg_3.predict(poly_reg3.fit_transform(X)),color='green')
plt.title('Truth Or Bluff (Polynomial Linear Regression)')
plt.xlabel('age')
plt.ylabel('weight')
plt.show()

X_grid=np.arange(min(X),max(X),0.1) # This will give us a vector.We will have to convert this
into a matrix
X_grid=X_grid.reshape((len(X_grid),1))
plt.scatter(X,y,color='red')
plt.plot(X_grid,lin_reg_3.predict(poly_reg3.fit_transform(X_grid)),color='green')
#plt.plot(X,lin_reg_3.predict(poly_reg3.fit_transform(X)),color='green')
plt.title('Truth Or Bluff (Polynomial Linear Regression)')
plt.xlabel('age')
plt.ylabel('weight')
plt.show()


lin_reg.predict([[6.5]])

lin_reg_2.predict(poly_reg2.fit_transform([[6.5]]))

lin_reg_3.predict(poly_reg3.fit_transform([[6.5]]))
```







array([539.06379994])

LAB-08

Aim: Model Evaluation: Model Evaluation, Over-fitting, Under-fitting and Model Selection,Ridge Regression, Grid Search, Model Refinement

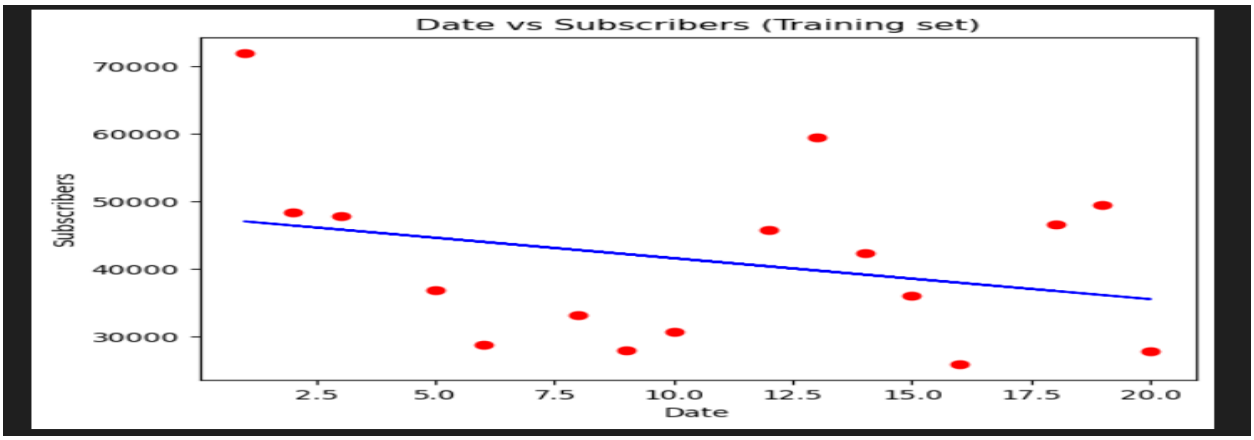
Code: Ridge regression

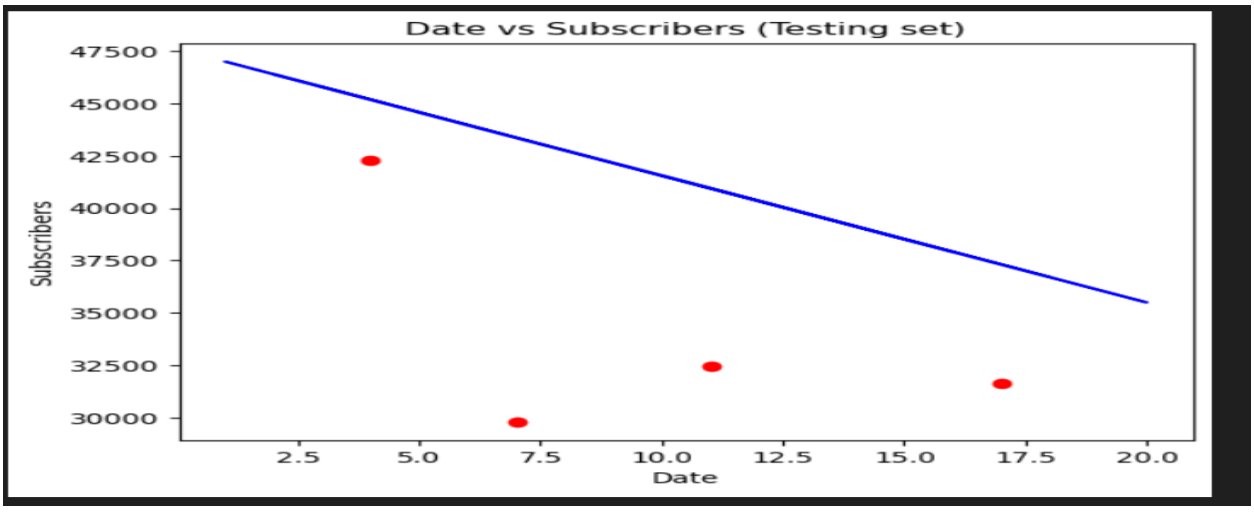
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn import metrics
import numpy as np

df=pd.read_csv("PewDiePie.csv")

#dividing the variables into dependent and independent
X=pd.DataFrame(df['Date'])
y=pd.DataFrame(df['Subscribers'])

#Split the data into train and test sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)

#train the algorithm
ridge=Ridge(alpha=1.0)
ridge.fit(X_train,y_train)

#retriving the intercept
print(ridge.intercept_)

#retriving the slope
print(ridge.coef_)

#predecting the test results
y_pred = ridge.predict(X_test)

#evaluting the algorithm
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 set
plt.scatter(X_train, y_train, color='red') # plotting the observation line
plt.plot(X_train, ridge.predict(X_train), color='blue') # plotting the regression line
plt.title("Date vs Subscribers (Training set)") # stating the title of the graph
```

```
plt.xlabel("Date") # adding the name of x-axis
plt.ylabel("Subscribers") # adding the name of y-axis
plt.show() # specifies end of graph

#plot for the test set
plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, ridge.predict(X_train), color='blue') # plotting the regression line
plt.title("Date vs Subscribers (Testing set)")
plt.xlabel("Date")
plt.ylabel("Subscribers")
plt.show()
```

output:

```
[47611.65464541]
[[-605.65189665]]
Mean Absolute Error: 7670.798653106103
Mean Squared Error: 74374253.37775256
Root Mean Squared Error: 8624.05086822617
```

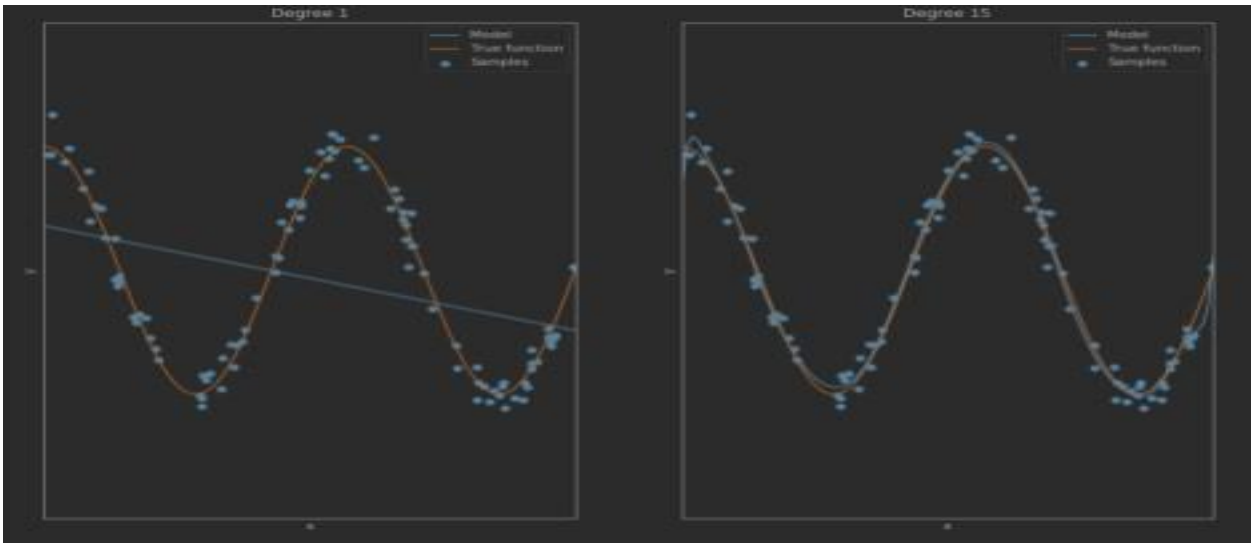




Overfitting and underfitting Problem:

```
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-9

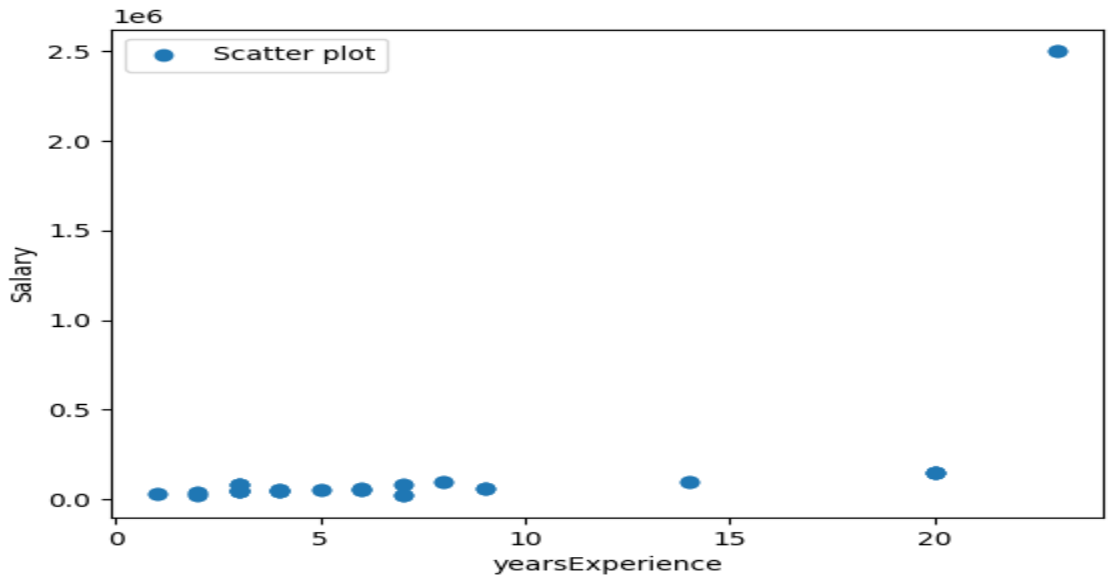
Aim: Introduction to Visualization Tools: Introduction to Data Visualization,Introduction to Matplotlib

CODE:

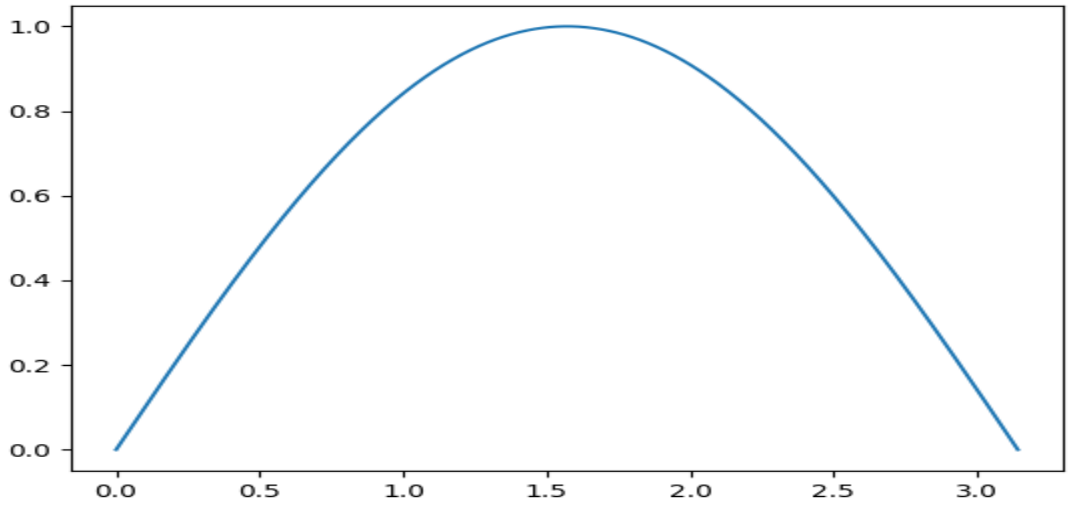
#scatterplot

```
d1=df.head(50)
x_scatter=d1['yearsExperience']
y_scatter=d1['salary']
plt.xlabel('yearsExperience')
plt.ylabel('Salary')
plt.scatter(x_scatter,y_scatter,label="Scatter plot")
plt.legend()
plt.show()
```

output:



```
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,1*np.pi,10000)
y=np.sin(x)
fig, ax=plt.subplots()
ax.plot(x,y)
plt.show()
```



LAB-10

AIM: Basic Visualization Tools: Area Plots,Histograms,Bar Charts

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r"C:\Users\DASARI GANGADHAR\Desktop\DSP\salary.csv")
```

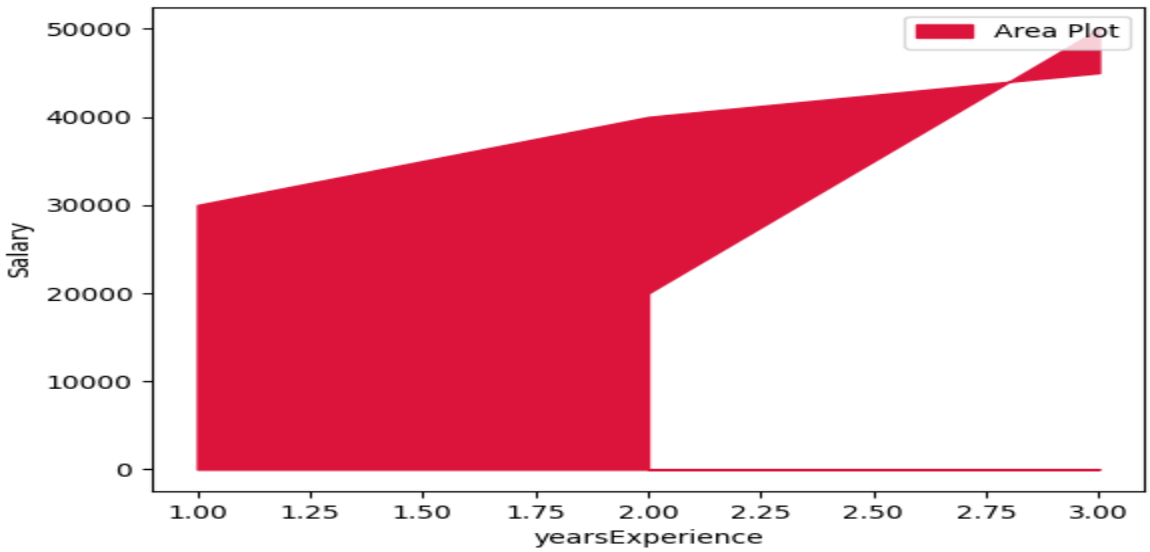
#line plots

```
x=df["yearsExperience"]
y=df["salary"]

plt.xlabel("yearsExperience")
plt.ylabel("salary")
plt.plot(x,y,linestyle="solid",label="Employee data" )
plt.title("yearsExperience vs salary")
plt.grid()
plt.legend()
plt.show()
```

#areaplots

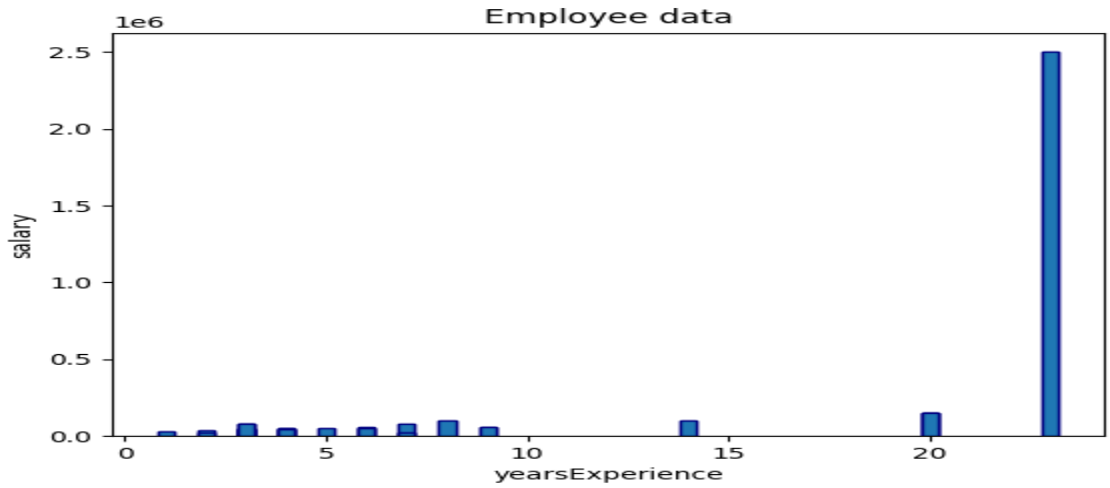
```
d2=df.head()
print(d2['yearsExperience'])
x_area=d2['yearsExperience']
```

```
y_area=d2['salary']
plt.xlabel('yearsExperience')
plt.ylabel('Salary')
plt.fill_between(x_area,y_area,label="Area Plot",color="crimson")
plt.legend()
plt.show()
```



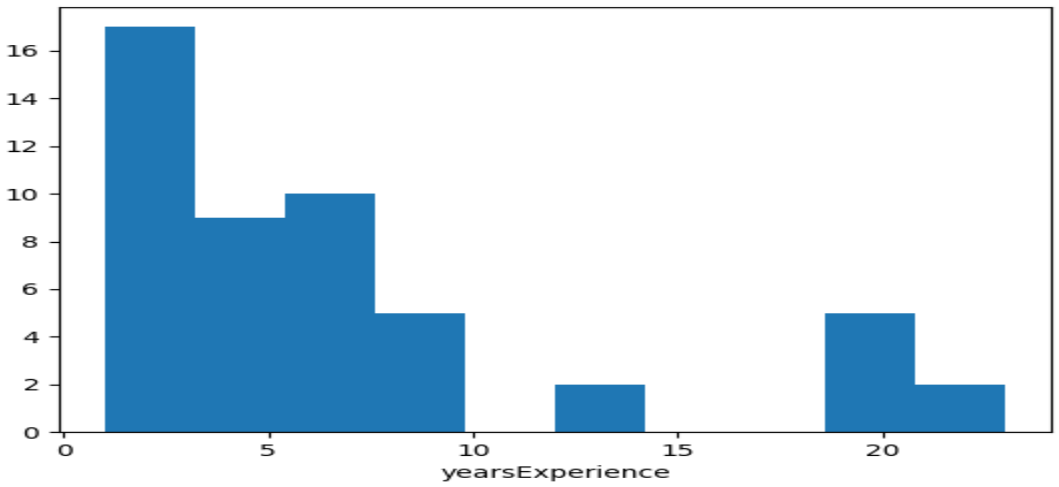
#Bar plots

```
x_bar=df['yearsExperience']
y_bar=df['salary']
plt.bar(x_bar,y_bar,label='yearsExperience',width=0.4,edgecolor="navy")
plt.title('Employee data')
plt.xlabel('yearsExperience')
plt.ylabel('salary')
plt.show()
```



#histogram

```
x_h=df['yearsExperience']
plt.hist(x_h,bins=10)
plt.xlabel('yearsExperience')
plt.show()
```



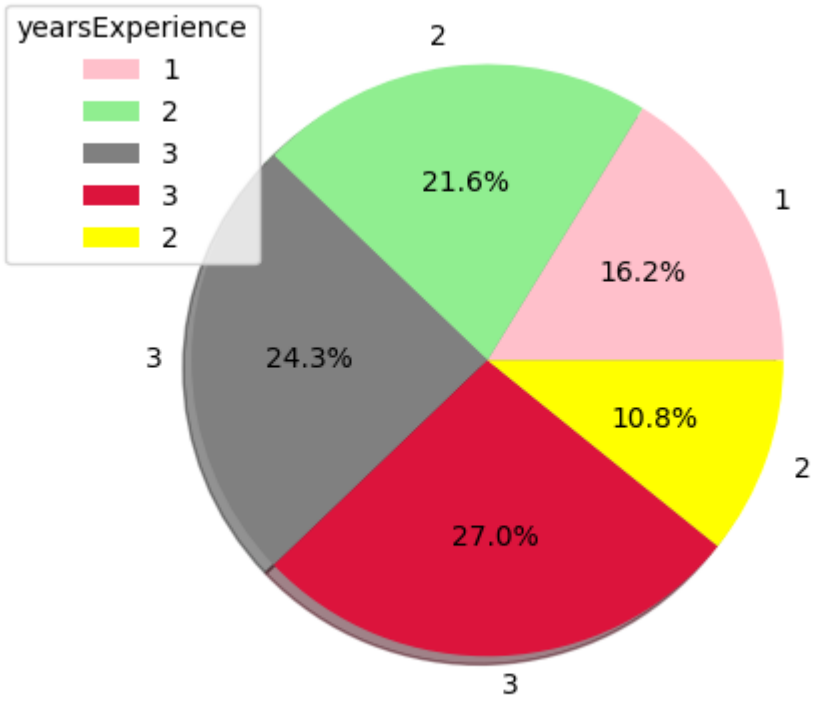
LAB-11

Aim: Specialized visualization tools pie charts ,boxplots

Code:

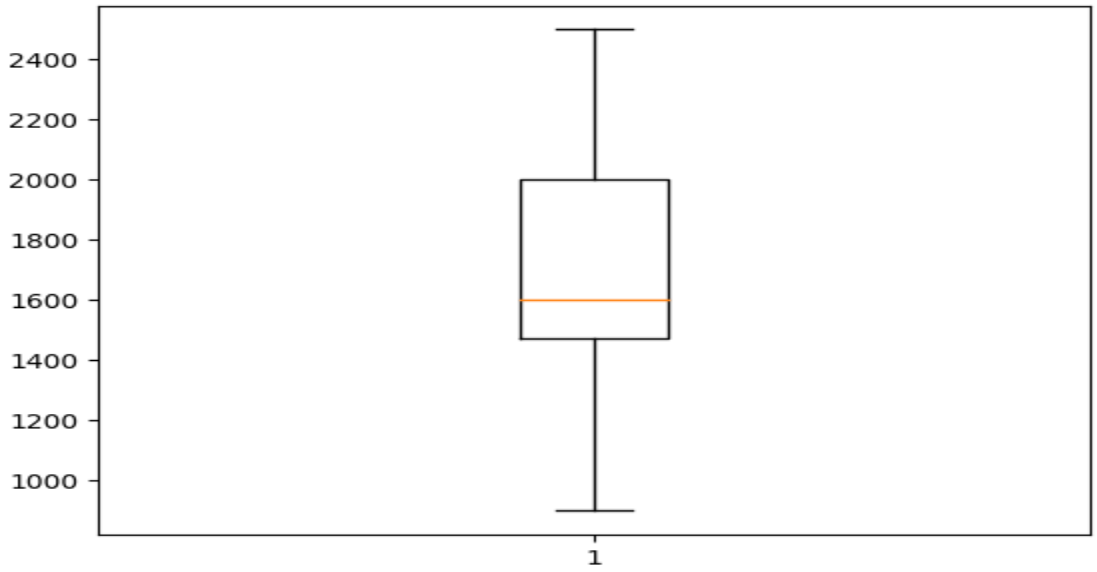
#pie

```
dt=df.head()
x_pie=dt['yearsExperience']
y_pie=dt['salary']
plt.axis('equal')
plt.pie(y_pie,labels=x_pie,colors=['pink','lightgreen','grey','crimson','yellow'],shadow=True,autop
ct='%2.1f%%')
plt.legend(title="yearsExperience",loc="best")
plt.show()
```



#box plot

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
data = pd.read_csv("data.csv")
data.head()
x = data.Volume
plt.boxplot(x)
plt.show()
```



LAB-12

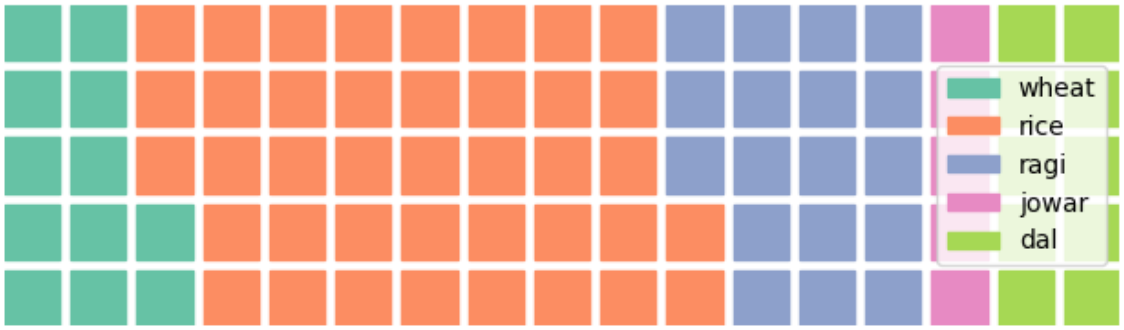
Aim:

Advanced Visualization Tools: Waffle Charts, Word Clouds,Seaborn and Regression Plots

CODE:

WAFFLE CHART

```
# python program to generate Waffle Chart
# importing all necessary requirements
import pandas as pd
import matplotlib.pyplot as plt
from pywaffle import Waffle
# creation of a dataframe
data ={'grossary': ['wheat', 'rice','ragi',
'jowar', 'dal'],
'stock': [12, 40, 18, 5, 10]}
df = pd.DataFrame(data)
# To plot the waffle Chart
fig = plt.figure(FigureClass = Waffle,
rows = 5,values = df.stock,
labels = list(df.grossary))
```

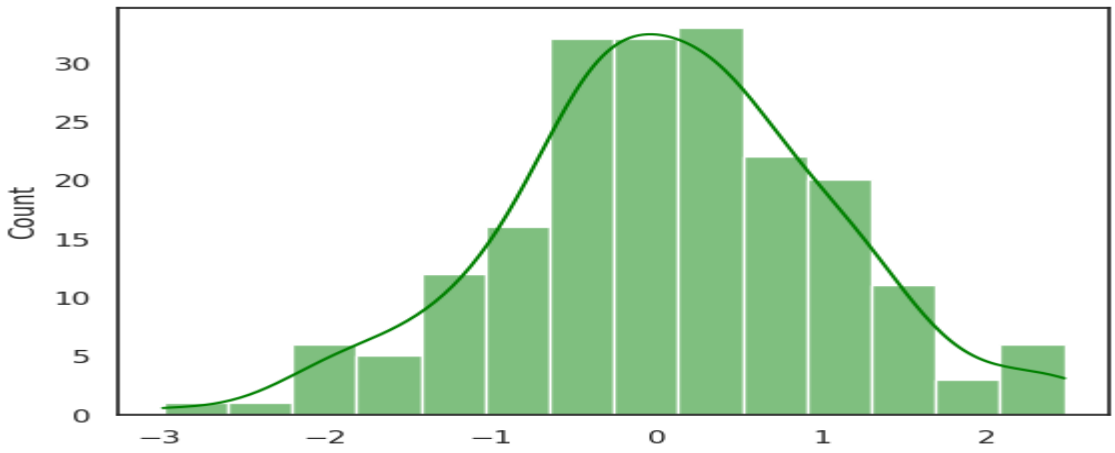


#word cloud

```
#python code for Word cloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
text="DASARI GANGADHAR n190302"
wc=WordCloud().generate(text)
plt.imshow(wc)
plt.axis("off")
plt.show()
```

#SEABORN

```
import numpy as np
import seaborn as sns
sns.set(style="white")
# Generate a random univariate dataset
rs = np.random.RandomState(10)
d = rs.normal(size=200)
# Plot a simple histogram and kde
sns.histplot(d, kde=True, color="green")
```



#maps

```
# import the library
import folium
# Make an empty map
```

```
m = folium.Map(location=[20,0], tiles="OpenStreetMap", zoom_start=2)
# Import the pandas library
import pandas as pd
# Make a data frame with dots to show on the map
data = pd.DataFrame({
'lon':[-58, 20.5937, 145, 30.32, -4.03, -73.57, 36.82, -38.5],
'lat':[-34, 78.9629, -38, 59.93, 5.33, 45.52, -1.29, -12.97],
'name':['Buenos Aires', 'norway', 'melbourne', 'St Petersbourg', 'Abidjan',
'Montreal', 'Nairobi', 'Salvador'],
'value':[10, 12, 40, 70, 23, 43, 100, 43]
}, dtype=str)
# add marker one by one on the map
for i in range(0,len(data)):
      folium.Marker(
      location=[data.iloc[i]['lat'], data.iloc[i]['lon']],
      popup=data.iloc[i]['name'],
   ).add_to(m)
# Show the map again
M
```