Assignment 3

Assignment Date	03 October 2022
Team ID	PNT2022TMID45005
Student Name	Sivaranjani P
Student RollNumber	811219205015
Project Name	AI Based Discourse For Banking Industry
Maximum Marks	2 Marks

Problem Statement: Abalone Age Prediction

- **Download the dataset: Dataset**
- **Load the dataset into the tool.**

```
import numpy as
np import pandas
as pd

ds=pd.read_csv("abalone.csv")

# Rings / integer / -- / +1.5 gives the age in years
ds['Age']=ds["Rings"]+1

.5ds.head(5)
```

Sex weight	Length \	Diameter	Height	Whole	weight	Shucked	weight	Viscera
• M 0.1010 • M	0.455 0.350	0.365 0.265	0.095 0.090		0.5140 0.2255		0.2245 0.0995	
0.0485 2 F	0.530	0.420	0.135		0.6770		0.2565	
0.1415 3 M	0.440	0.365	0.125		0.5160		0.2155	
0.1140 4 I	0.330	0.255	0.080		0.2050		0.0895	
0.0395								

	Shell weight	Rings	Age
0	0.150	15	16.5
1	0.070	7	8.5
2	0.210	9	10.5
3	0.155	10	11.5
4	0.055	7	8.5

- **Perform Below Visualizations.**

- **Univariate Analysis**

- **Bi-Variate Analysis**

- **Multi-Variate Analysis**

```
# univarient analysis

#frequency table for

age



ft = ds1['Age'].value_counts()

print("Frequency table for Age is given below")
print("{}\n\n\n".format(ft))

# mean

print("Mean, Median, std \n")
ma=ds1['Age'].mean() #mean of
age
mh = ds1['Height'].mean() #mean of height
mel = ds1['Length'].median() #median value of length
stw = ds1['Whole weight'].std() #standard devation of whole weight


#chart

import matplotlib.pyplot as plt # library for plot or graph
import seaborn as sns

plt.subplot(1,2,1)
ch = ds1.boxplot(column='Diameter',grid=True,color ='red')
plt.title('Box plot')

plt.subplot(1,2,2)
DC =
sns.kdeplot(ds1['Diameter'])
plt.title('Density Curve')
```

```
print("1-mean of age = ",ma)
print("2-mean of height =
",mh)
print("3-median value of length = ",mel)#
print("4-standard devation of whole weight = ",stw)
print("5-frequency table for rings = \n {}"
.format(fre))print("\nChart\n\n6-boxplot of
Diameter",flush=True)
```

Frequency table for Age is given below

11.5	32
10.5	28
8.5	20
9.5	18
13.5	17
12.5	16
14.5	13
15.5	11
16.5	10
17.5	7
6.5	6

7.5	5
21.5	4
5.5	4
20.5	3
19.5	3
22.5	2
18.5	1
Name:	Age, dtype: int64

Mean, Median, std

1-mean of age = 12.235
2-mean of height =
0.13482500000000033-median value of
length = 0.53
4-standard devation of whole weight =
0.48292552690013145-frequency table for rings =

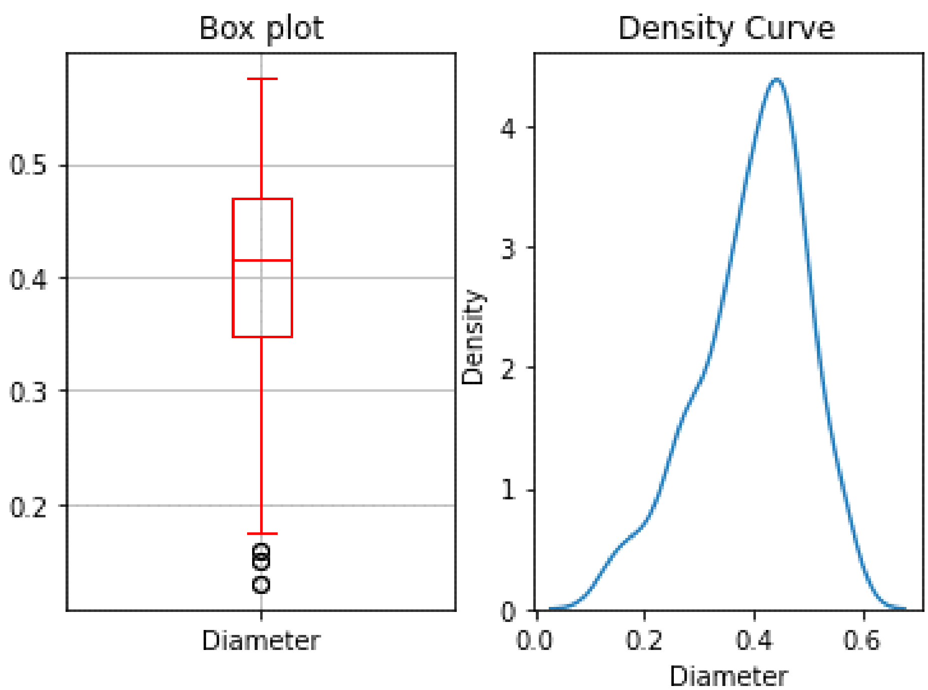
10	32
9	28
7	20
8	18
12	17
11	16
13	13
14	11

15	10
16	7
5	6
6	5
20	4
4	4
19	3
18	3
21	2
17	1

Name: Rings, dtype:

int64Chart

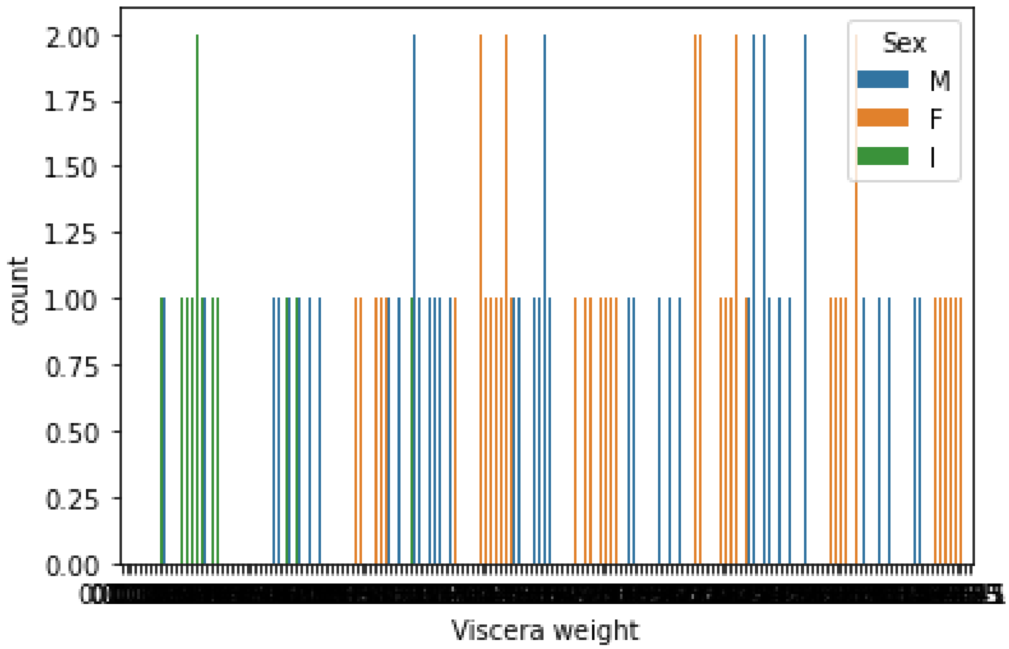
6-boxplot of Diameter




```
#multi-varient analysis

import matplotlib.pyplot as
pltimport seaborn as sns

ds1=ds.head(200)
df=sns.countplot(x="Viscera weight",hue='Sex',data=ds1)

print(df)
```

AxesSubplot(0.125,0.125;0.775x0.75

5)



- **Perform descriptive statistics on the dataset.**

```
ds.describe()
```

```
               Length     Diameter       Height  Whole weight  Shucked
weight  \
count  4177.000000  4177.000000  4177.000000   4177.000000
4177.000000
```

mean 0.359367 std	0.523992 0.120093	0.407881 0.099240	0.139516 0.041827	0.828742 0.490389
0.221963 min	0.075000	0.055000	0.000000	0.002000
0.001000				
25%	0.450000	0.350000	0.115000	0.441500
0.186000				
50%	0.545000	0.425000	0.140000	0.799500
0.336000				
75%	0.615000	0.480000	0.165000	1.153000
0.502000 max	0.815000	0.650000	1.130000	2.825500
1.488000				

	Viscera weight	Shell weight	Rings	Age
count	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.180594	0.238831	9.933684	11.433684
std	0.109614	0.139203	3.224169	3.224169
min	0.000500	0.001500	1.000000	2.500000
25%	0.093500	0.130000	8.000000	9.500000

50%	0.171000	0.234000	9.000000	10.500000
75%	0.253000	0.329000	11.000000	12.500000
max	0.760000	1.005000	29.000000	30.500000

- **Check for Missing values and deal with them.**

```
ds.info()

<class
'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
 •   Sex             4177 non-null   object
 •   Length          4177 non-null   float64
 •   Diameter        4177 non-null   float64
 •   Height          4177 non-null   float64
 •   Whole weight    4177 non-null   float64
 •   Shucked weight  4177 non-null   float64
 •   Viscera weight  4177 non-null   float64
 •   Shell weight    4177 non-null   float64
 •   Rings           4177 non-null   int64
 •   Age             4177 non-null
     float64dtypes: float64(8), int64(1),
object(1) memory usage: 326.5+ KB
ds.isnull().sum()

Sex               0

Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
Age               0
```

	Sex	Length	Diameter	Height	Wholeweight	Shuckedweigh	\

						t	
0	True	True	True	True	True	True	
1	True	True	True	True	True	True	
2	True	True	True	True	True	True	
3	True	True	True	True	True	True	
4	True	True	True	True	True	True	
...	...	...	...	...	...	...	
4172	True	True	True	True	True	True	
4173	True	True	True	True	True	True	
4174	True	True	True	True	True	True	
4175	True	True	True	True	True	True	

dtype:

int64

ds.notnull

()

```
4176   True     True       True     True            True               True
```

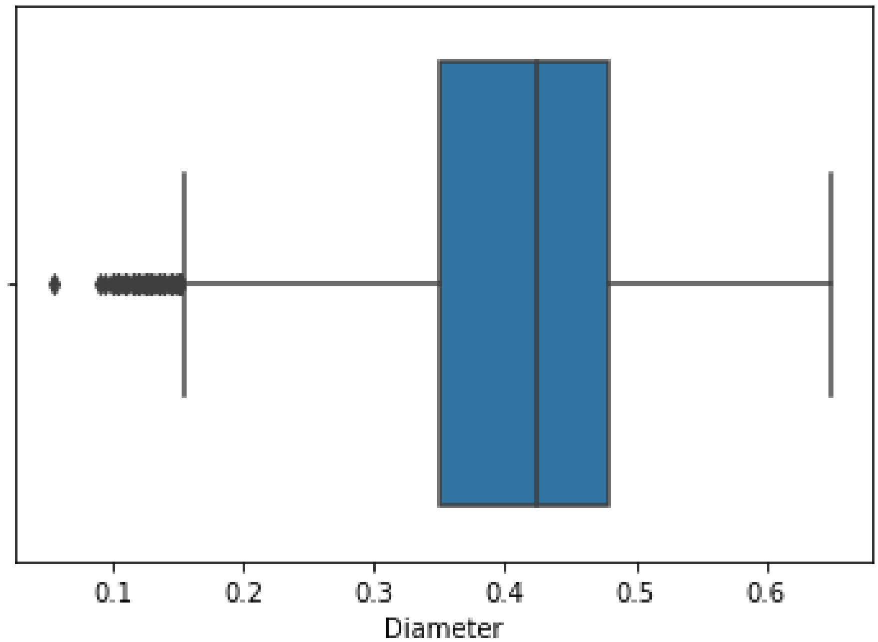
	Viscera	weight	Shell weight	Rings	Age
0		True	True	True	True
1		True	True	True	True
2		True	True	True	True
3		True	True	True	True
4		True	True	True	True
...		...	...	...	...
4172		True	True	True	True
4173		True	True	True	True
4174		True	True	True	True
4175		True	True	True	True
4176		True	True	True	True

```
[4177 rows x 10 columns]
```

- **Find the outliers and replace them outliers**

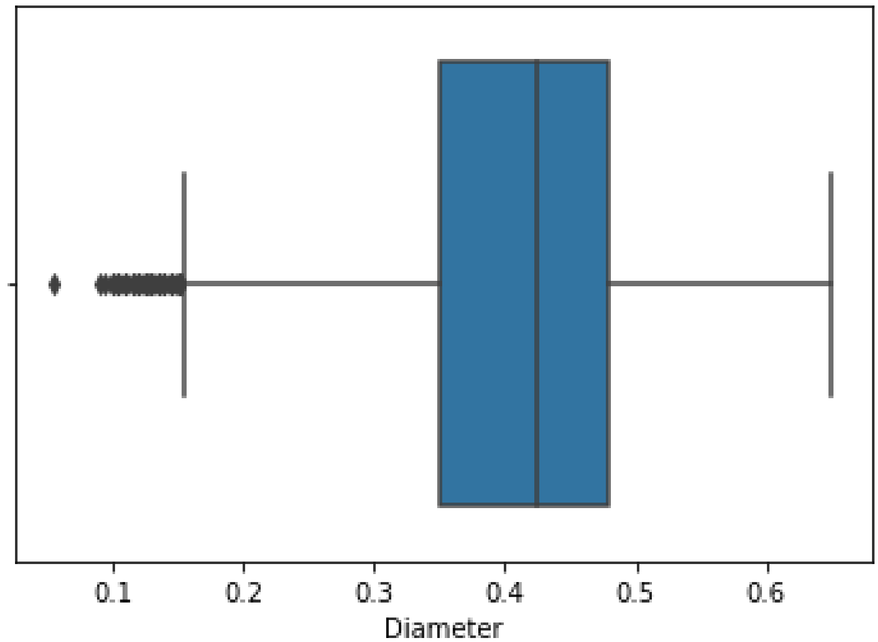
```
#occurence of outliers
#a data point in a data set that is distant from all other
observations

sns.boxplot(ds.Diameter)
```

```
/home/lokesh/anaconda3/lib/python3.9/site-packages/seaborn/
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional
argumentwill be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(

<AxesSubplot:xlabel='Diameter'>
```




```
Q1=
ds.Diameter.quantile(0.25)
Q3=ds.Diameter.quantile(0.75
)

IQR=Q3-Q1   #spread the middle values are

upper_limit =Q3 +
1.5*IQRlower_limit =Q1
- 1.5*IQR

ds['Diameter'] =
np.where(ds['Diameter']>upper_limit,30,ds['Diameter'])

sns.boxplot(ds.Diameter)
```

```
/home/lokesh/anaconda3/lib/python3.9/site-packages/seaborn/
_decorators.py:36: FutureWarning: Pass the following variable as a
keyword arg: x. From version 0.12, the only valid positional
argumentwill be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(

<AxesSubplot:xlabel='Diameter'>
```



- **Check for Categorical columns and perform encoding.**

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

ds1['Sex'] = le.fit_transform(ds1['Sex'])
ds1

        # 0 = female, 1 = infant, 2 = male
```

	Sex	Length	Diameter	Height	Whole	weight	Shucked	weight	\
0	2	0.455	0.365	0.095		0.5140		0.2245	
1	2	0.350	0.265	0.090		0.2255		0.0995	
2	0	0.530	0.420	0.135		0.6770		0.2565	
3	2	0.440	0.365	0.125		0.5160		0.2155	
4	1	0.330	0.255	0.080		0.2050		0.0895	
..	...	...	...	...		...		...	
195	2	0.500	0.405	0.155		0.7720		0.3460	
196	0	0.505	0.410	0.150		0.6440		0.2850	
197	2	0.640	0.500	0.185		1.3035		0.4445	
198	2	0.560	0.450	0.160		0.9220		0.4320	
199	2	0.585	0.460	0.185		0.9220		0.3635	

	Viscera weight	Shell	weight	Rings	Age
0	0.1010		0.150	15	16.5
1	0.0485		0.070	7	8.5
2	0.1415		0.210	9	10.5
3	0.1140		0.155	10	11.5
4	0.0395		0.055	7	8.5

..	...	...	...	...
195	0.1535	0.245	12	13.5
196	0.1450	0.210	11	12.5
197	0.2635	0.465	16	17.5
198	0.1780	0.260	15	16.5
199	0.2130	0.285	10	11.5

[200 rows x 10 columns]

- **Split the data into dependent and independent variables.**

#Splitting the Dataset into the Independent Feature Matrix

```
x = ds1.iloc[:,
0:9]x
```

	Sex	Length	Diameter	Height	Whole	weight	Shucked	weight	\
0	2	0.455	0.365	0.095		0.5140		0.2245	
1	2	0.350	0.265	0.090		0.2255		0.0995	
2	0	0.530	0.420	0.135		0.6770		0.2565	
3	2	0.440	0.365	0.125		0.5160		0.2155	
4	1	0.330	0.255	0.080		0.2050		0.0895	
..	...	...	...	...		...		...	
195	2	0.500	0.405	0.155		0.7720		0.3460	
196	0	0.505	0.410	0.150		0.6440		0.2850	
197	2	0.640	0.500	0.185		1.3035		0.4445	
198	2	0.560	0.450	0.160		0.9220		0.4320	
199	2	0.585	0.460	0.185		0.9220		0.3635	

	Viscera	weight	Shell	weight	Rings
0		0.1010		0.150	15
1		0.0485		0.070	7
2		0.1415		0.210	9
3		0.1140		0.155	10
4		0.0395		0.055	7
..		...		...	...
195		0.1535		0.245	12
196		0.1450		0.210	11
197		0.2635		0.465	16
198		0.1780		0.260	15
199		0.2130		0.285	10

[200 rows x 9 columns]

#Extracting the Dataset to Get the Dependent Vector

```
y =
ds1.iloc[:,9:10]
print(y)
```

```
     Age
0    16.5
```

1	8.5	
2	10.5	
3	11.5	
4	8.5	
..	...	
195	13.5	
196	12.5	
197	17.5	
198	16.5	
199	11.5	
[200	rows	x 1 columns]

- **Scale the independent variables**

```
#scaling the independent variables using scale and MinMaxScaler

from sklearn.preprocessing import scale
from sklearn.preprocessing import MinMaxScaler

mm = MinMaxScaler()

x_scaled =
mm.fit_transform(x)y_scaled
= mm.fit_transform(y)

x_scaled

array([[1.        , 0.51351351, 0.52808989, ..., 0.17680075,
0.14070352,
        0.64705882],
       [1.        , 0.32432432, 0.30337079, ..., 0.07857811,
0.06030151,
        0.17647059],
       [0.        , 0.64864865, 0.65168539, ..., 0.2525725 ,
0.20100503,
        0.29411765],
       ...,
       [1.        , 0.84684685, 0.83146067, ..., 0.4808232 ,
0.45728643,
        0.70588235],
       [1.        , 0.7027027 , 0.71910112, ..., 0.32086062,
0.25125628,
        0.64705882],
       [1.        , 0.74774775, 0.74157303, ..., 0.38634238,
0.27638191,
        0.35294118]])
```

```
y_scaled

array([[0.64705882

],

       [0.17647059],

       [0.29411765],
       [0.35294118],
       [0.17647059],
       [0.23529412],
       [0.94117647],
       [0.70588235],
       [0.29411765],
       [0.88235294],
       [0.58823529],
       [0.35294118],
       [0.41176471],
       [0.35294118],
       [0.35294118],
       [0.47058824],
       [0.17647059],
       [0.35294118],
       [0.17647059],
       [0.29411765],
       [0.41176471],
       [0.35294118],
       [0.47058824],
       [0.29411765],
       [0.35294118],
       [0.41176471],
       [0.41176471],
       [0.47058824],
       [0.64705882],
       [0.41176471],
       [0.35294118],
       [0.64705882],
       [0.82352941],
       [0.88235294],
       [0.52941176],
       [0.23529412],
       [0.70588235],
       [0.23529412],
       [0.41176471],
       [0.29411765],
       [0.29411765],
       [0.58823529],
       [0.05882353],
       [0.05882353],
```

```
[0.        ],
[0.17647059],
[0.29411765],
[0.17647059],
[0.11764706],
[0.29411765],
[0.23529412],
[0.17647059],

[0.35294118],
[0.35294118],
[0.17647059],
[0.23529412],
[0.23529412],
[0.23529412],
[0.        ],
[0.17647059],
[0.17647059],
[0.29411765],
[0.35294118],
[0.17647059],
[0.23529412],
[0.23529412],
[0.47058824],
[0.52941176],
[0.35294118],
[0.11764706],
[0.52941176],
[0.23529412],
[0.94117647],
[0.41176471],
[0.52941176],
[0.64705882],
[0.29411765],
[0.35294118],
[0.41176471],
[0.58823529],
[0.29411765],
[0.47058824],
[0.70588235],
[1.        ],
[0.58823529],
[0.47058824],
[0.52941176],
[0.35294118],
[0.29411765],
[0.47058824],
[0.64705882],
[0.47058824],
[0.52941176],
```

```
[0.35294118],
[0.64705882],
[0.58823529],
[0.29411765],
[0.23529412],
[0.17647059],
[0.35294118],
[0.17647059],
[0.64705882],

[0.64705882],
[0.35294118],
[0.47058824],
[0.47058824],
[0.41176471],
[0.35294118],
[0.29411765],
[0.29411765],
[0.29411765],
[0.29411765],
[0.29411765],
[0.29411765],
[0.41176471],
[0.41176471],
[0.41176471],
[0.35294118],
[0.29411765],
[0.23529412],
[0.29411765],
[0.17647059],
[0.58823529],
[0.11764706],
[0.11764706],
[0.05882353],
[0.11764706],
[0.23529412],
[0.88235294],
[0.82352941],
[0.76470588],
[0.29411765],
[0.17647059],
[0.17647059],
[0.17647059],
[0.23529412],
[0.17647059],
[0.29411765],
[0.29411765],
[0.29411765],
[0.35294118],
[0.35294118],
```

```
[0.70588235],
[0.41176471],
[0.35294118],
[0.35294118],
[0.35294118],
[0.29411765],
[0.05882353],
[0.        ],
[0.64705882],
[0.29411765],

[0.35294118],
[0.35294118],
[0.47058824],
[0.35294118],
[0.52941176],
[0.70588235],
[0.52941176],
[0.52941176],
[0.52941176],
[0.52941176],
[0.47058824],
[0.82352941],
[0.70588235],
[0.58823529],
[0.94117647],
[0.94117647],
[0.58823529],
[0.47058824],
[0.58823529],
[0.17647059],
[0.23529412],
[0.23529412],
[0.05882353],
[0.17647059],
[0.05882353],
[0.23529412],
[0.        ],
[0.41176471],
[0.58823529],
[1.        ],
[0.35294118],
[0.35294118],
[0.47058824],
[0.52941176],
[0.47058824],
[0.35294118],
[0.41176471],
[0.29411765],
[0.52941176],
```

```
        [0.47058824],
        [0.58823529],
        [0.23529412],
        [0.35294118],
        [0.47058824],
        [0.41176471],
        [0.70588235],
        [0.64705882],
        [0.35294118]])
```

- ### Split the data into training and testing

```
from sklearn.model_selection import train_test_split # library for
split the data into training and testing

x_train,x_test,y_train,y_test =
train_test_split(x_scaled,y_scaled,train_size=0.80,test_size =
0.20,random_state=0)

x_train

array([[0.5       , 0.17117117, 0.15730337, ..., 0.0261927 ,
0.01809045,
         0.17647059],
        [0.        , 0.71171171, 0.69662921, ..., 0.34985968,
0.31155779,
         0.47058824],
        [0.        , 0.73873874, 0.71910112, ..., 0.49672591,
0.27638191,
         0.41176471],
        ...,
        [1.        , 0.48648649, 0.47191011, ..., 0.16651076,
0.15577889,
         0.35294118],
        [0.        , 0.52252252, 0.5505618 , ..., 0.19363891,
0.14070352,
         0.17647059],
        [1.        , 0.63963964, 0.68539326, ..., 0.42376052,
0.27638191,
         0.23529412]])
y_train

array([[0.17647059

],

        [0.47058824],
        [0.41176471],
        [0.29411765],
        [0.58823529],
```

```
[0.17647059],
[0.29411765],
[0.64705882],
[0.29411765],
[0.41176471],
[0.23529412],
[0.11764706],
[0.47058824],
[0.23529412],
[0.        ],
[0.35294118],
[0.35294118],

[0.52941176],
[0.29411765],
[0.23529412],
[0.29411765],
[0.29411765],
[1.        ],
[0.29411765],
[0.35294118],
[0.52941176],
[0.17647059],
[0.82352941],
[0.17647059],
[0.52941176],
[0.29411765],
[0.64705882],
[0.29411765],
[0.64705882],
[0.35294118],
[0.47058824],
[0.29411765],
[0.35294118],
[0.47058824],
[0.35294118],
[0.35294118],
[0.29411765],
[0.29411765],
[0.47058824],
[0.29411765],
[0.35294118],
[0.29411765],
[0.17647059],
[0.17647059],
[0.70588235],
[0.05882353],
[0.58823529],
[0.35294118],
[0.41176471],
```

```
[0.41176471],
[0.        ],
[0.17647059],
[0.11764706],
[0.35294118],
[0.29411765],
[0.52941176],
[0.47058824],
[0.23529412],
[0.64705882],
[0.64705882],
[0.29411765],
[0.58823529],

[0.23529412],
[0.94117647],
[0.58823529],
[0.11764706],
[0.29411765],
[0.11764706],
[0.47058824],
[0.35294118],
[0.52941176],
[0.29411765],
[0.47058824],
[0.23529412],
[0.41176471],
[0.47058824],
[0.41176471],
[0.47058824],
[0.35294118],
[0.17647059],
[0.29411765],
[0.35294118],
[0.41176471],
[0.70588235],
[0.64705882],
[0.94117647],
[0.35294118],
[0.58823529],
[0.17647059],
[0.35294118],
[0.17647059],
[0.52941176],
[0.47058824],
[0.35294118],
[0.35294118],
[0.23529412],
[0.64705882],
[0.23529412],
```

```
[0.23529412],
[0.23529412],
[0.17647059],
[0.29411765],
[0.47058824],
[0.05882353],
[0.47058824],
[0.17647059],
[0.23529412],
[0.35294118],
[0.41176471],
[0.17647059],
[0.35294118],
[0.70588235],
```

x_test

```
[0.88235294],
[0.52941176],
[0.64705882],
[0.41176471],
[0.29411765],
[0.64705882],
[0.94117647],
[0.23529412],
[0.05882353],
[0.82352941],
[0.70588235],
[0.47058824],
[0.29411765],
[0.41176471],
[0.35294118],
[0.70588235],
[0.58823529],
[0.41176471],
[0.05882353],
[0.23529412],
[0.94117647],
[0.35294118],
[0.41176471],
[0.58823529],
[0.47058824],
[0.41176471],
[0.05882353],
[0.52941176],
[0.29411765],
[0.        ],
[0.35294118],
[0.29411765],
[0.52941176],
[0.35294118],
[0.70588235],
[0.35294118],
[0.88235294],
[0.35294118],
[0.52941176],
```

[0.58823529],
[0.35294118],
[0.17647059],
[0.23529412]])

array([[1.   , 0.35135135, 0.37078652, 0.21052632, 0.08948413,

0.08160377, 0.06828812,	0.09045226, 0.17647059],
[1. , 0.94594595,	0.94382022, 0.92105263, 0.76448413,
0.66226415, 1. ,	0.58291457, 0.58823529],
[0. , 0.59459459,	0.60674157, 0.44736842, 0.25297619,

0.23386342,	0.21105528,	0.35294118],	
0.54054054,	0.53932584,	0.47368421,	0.19543651,
0.23666978,	0.15577889,	0.17647059],	
0.26	0.25842697,	0.23	0.04

126126,		684211,	503968,
0.07670072,	0.04020101,	0.23529412],	
0.7027027,	0.71910112,	0.63157895,	0.39424603,
0.48924228,	0.29145729,	0.35294118],	
0.4594594	0.38202247,	0.2894736	0.1275793

6,		8,	7,
0.13283442,	0.11055276,	0.23529412],	
0.52252252,	0.49438202,	0.42105263,	0.19246032,
0.1898971,	0.14723618,	0.35294118],	
0.57657658,	0.56179775,	0.5,	0.20297619,
0.1	0.1809045	0.4	

655753,	2,	1176471],	
0.83783784,	0.86516854,	0.78947368,	0.53234127,
0.55846586,	0.44221106,	0.35294118],	
0.6036036,	0.61797753,	0.36842105,	0.23611111,
0.287184	0.16582915,	0.294117	

28,		65],	
0.18018018,	0.14606742,	0.10526316,	0.01706349,
0.03180543,	0.0201005,	0.05882353],	
0.72072072,	0.78651685,	0.73684211,	0.3609127,
0.36202058,	0.28643216,	0.58823529],	
0	0.7	0	0

.71171171,	1910112,	.5 ,	.38035714,
0.26753976,	0.30150754,	0.47058824],	
0.72972973,	0.70786517,	0.52631579,	0.36150794,
0.45930776,	0.27638191,	0.29411765],	
0.6756	0.66292135,	0.4473	0.2928

7568,		6842,	5714,
0.26753976,	0.25125628,	0.70588235],	
0.91891892,	0.94382022,	0.71052632,	0.7015873,
0.72217025,	0.44723618,	0.88235294],	
0.76576577,	0.78651685,	0.65789474,	0.48888889,

0.517305889,	0.40201005,	0.76470588],	
0.504504 5 ,	0.50561798,	0.3421052 6,	0.19543651,
0.205799 81,	0.13567839,	0.23529412],	
0.78378378,	0.71910112,	0.81578947,	0.42380952,
0.529	0.30653266,	0.529	

46679,		41176],	
0.81081081,	0.7752809,	0.71052632,	0.39146825,
0.38821328,	0.31658291,	0.35294118],	
0.57657658,	0.56179775,	0.44736842,	0.20595238,
0.18896165	0.16482412,	0.35294118	

,		],	
0.3963964,	0.37078652,	0.28947368,	0.06865079,
0.07202993,	0.06532663,	0.17647059],	
0.72972973,	0.74157303,	0.65789474,	0.43412698,
0.32179607,	0.4321608,	0.52941176],	
0.5	0.4831460	0.3	0.1

045045,	7,	4210526,	5138889,
0.19831618,	0.12562814,	0.176470 59],	
0.3603603 6,	0.30337079,	0.18421053,	0.07301587,
0.08325538,	0.06030151,	0.11764706],	
0.738738	0.7752809,	0.578947	0.373015

74,		37,	87,
0.3928905 5,	0.3417085 4,	0.411764 71],	
0.8108108 1,	0.8089887 6,	0.7894736 8,	0.4714285 7,

```
 0.23632075,
[1.        ,
 0.17971698,
[0.5       ,
 0.04009434,
[0.       ,
 0.39481132,
[0.5       ,
 0.12311321,
[1.        ,
 0.20141509,
[1.        ,
 0.19528302,
[0.       ,
 0.46792453,
[1.        ,
 0.27783019,
[0.5       ,
 0.01698113,
[1.        ,
 0.36650943,
[0.       ,
 0.35518868,
```

[0.   ,
 0.35283019,
[0.   ,
 0.26745283,
[0.   ,
 0.75896226,
[1.   ,
 0.44622642,
[0.   ,
 0.21367925,
[0.   ,
 0.44386792,
[0.   ,
 0.4009434 ,
[0.   ,
 0.22122642,
[0.5   ,
 0.07264151,
[0.   ,
 0.27169811,
[1.   ,
 0.15990566,
[1.   ,
 0.075   ,
[1.   ,
 0.35330189,
[1.   ,

0.50471698,	0.54256314,	0.34673367,	0.52941176],
[0. ,	0.62162162,	0.66292135,	0.52631579, 0.29206349,
0.27688679,	0.31057063,	0.24623116,	0.58823529],
[0.5 ,	0.07207207,	0.04494382,	0.05263158, 0.0047619 ,
0.00660377,	0.01122544,	0.00502513,	0.],
[0.5 ,	0.33333333,	0.33707865,	0.23684211, 0.10337302,
0.07971698,	0.06173994,	0.10552764,	0.17647059],
[0. ,	0.59459459,	0.60674157,	0.52631579, 0.25059524,
0.23207547,	0.31618335,	0.21105528,	0.23529412],
[1. ,	0.75675676,	0.7752809 ,	0.55263158, 0.40595238,
0.40660377,	0.47801684,	0.31658291,	0.64705882],
[1. ,	0.53153153,	0.51685393,	0.34210526, 0.15912698,
0.15235849,	0.18802619,	0.16582915,	0.29411765],
[0. ,	0.71171171,	0.69662921,	0.60526316, 0.3609127 ,
0.39339623,	0.38821328,	0.26130653,	0.47058824],
[0. ,	0.74774775,	0.74157303,	0.68421053, 0.35813492,
0.33443396,	0.494855 ,	0.28140704,	0.29411765],
[1. ,	0.97297297,	0.92134831,	0.65789474, 0.76547619,
0.71320755,	0.47614593,	0.77386935,	0.82352941],
[0.5 ,	0.28828829,	0.28089888,	0.21052632, 0.06944444,
0.0745283 ,	0.06173994,	0.04522613,	0.17647059],

[1. ,	0.76576577,	0.7752809 ,	0.63157895, 0.5109127 ,
0.375 ,	0.42563143,	0.57286432,	1.],
[0. ,	0.67567568,	0.6741573 ,	0.65789474, 0.30634921,

```
 0.26698113, 0.33021515, 0.27135678, 0.41176471]])
```

```
y_test
```

```
array([[0.17647059]
                   ,
       [0.58823529],
       [0.35294118],
       [0.17647059],
       [0.23529412],
       [0.35294118],
       [0.23529412],
       [0.35294118],
       [0.41176471],
       [0.35294118],
       [0.29411765],
       [0.05882353],
       [0.58823529],
       [0.47058824],
       [0.29411765],
       [0.70588235],
       [0.88235294],
       [0.76470588],
       [0.23529412],
       [0.52941176],
       [0.35294118],
       [0.35294118],
       [0.17647059],

       [0.52941176],
       [0.17647059],
       [0.11764706],
       [0.41176471],
       [0.52941176],
       [0.58823529],
       [0.        ],
       [0.17647059],
       [0.23529412],
       [0.64705882],
       [0.29411765],
       [0.47058824],
       [0.29411765],
       [0.82352941],
       [0.17647059],
       [1.        ],
       [0.41176471]])
```

```
print(x_scaled.shap
e)
print(y_scaled.shap
e)
print(x_train.shape
)
print(y_train.shape
)
print(x_test.shape)
print(y_test.shape)
```

```
(200, 9)
(200, 1)
(160, 9)
(160, 1)
(40, 9)
(40, 1)
```

- **Build the Model**

```
from sklearn.linear_model import LinearRegression

mlr = LinearRegression()

mlr.fit(x_train,y_trai

n)LinearRegression()
```

- **Train the Model**

- **Test the Model**

```
prediction                    =

mlr.predict(x_test)prediction

array([[1.76470588e-
       01],
       [5.88235294e-
       01],
       [3.52941176e-
       01],


       [1.76470588e-
       01],
       [2.35294118e-
       01],
       [3.52941176e-
       01],
       [2.35294118e-
       01],
       [3.52941176e-
```

01],
[4.11764706e-
01],
[3.52941176e-
01],
[2.94117647e-
01],
[5.88235294e-
02],
[5.88235294e-
01],
[4.70588235e-
01],
[2.94117647e-
01],
[7.05882353e-
01],
[8.82352941e-
01],
[7.64705882e-
01],
[2.35294118e-
01],
[5.29411765e-
01],
[3.52941176e-
01],
[3.52941176e-
01],
[1.76470588e-
01],
[5.29411765e-
01],
[1.76470588e-
01],
[1.17647059e-
01],
[4.11764706e-
01],
[5.29411765e-
01],
[5.88235294e-
01],
[2.20691474e-
16],
[1.76470588e-
01],
[2.35294118e-
01],
[6.47058824e-

```
        01],
        [2.94117647e-
        01],
        [4.70588235e-
        01],
        [2.94117647e-
        01],
        [8.23529412e-
        01],
        [1.76470588e-
        01],
        [1.00000000e+00]
        , [4.11764706e-
        01]])
```

```
prediction.astype(int)
```

```
array([[0]
         ,
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],

        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
        [0],
```

```
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [1],
       [0]])
y_test.astype(in

t)array([[0],

       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],

       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
```

```
       [0],
       [1],
       [0]])
```

- ## Measure the performance using Metrics.

```
from sklearn.metrics import
r2_score
r2_score(prediction,y_test)

1.0

from sklearn.preprocessing import PolynomialFeatures
plr = PolynomialFeatures(degree=2)
x_poly =

plr.fit_transform(x)x_poly

array([[1.00000e+00, 2.00000e+00, 4.55000e-01, ..., 2.25000e-02,
        2.25000e+00, 2.25000e+02],
       [1.00000e+00, 2.00000e+00, 3.50000e-01, ..., 4.90000e-
        03,4.90000e-01, 4.90000e+01],
       [1.00000e+00, 0.00000e+00, 5.30000e-01, ..., 4.41000e-02,
        1.89000e+00, 8.10000e+01],
       ...,
       [1.00000e+00, 2.00000e+00, 6.40000e-01, ..., 2.16225e-01,
        7.44000e+00, 2.56000e+02],
       [1.00000e+00, 2.00000e+00, 5.60000e-01, ..., 6.76000e-02,
        3.90000e+00, 2.25000e+02],
       [1.00000e+00, 2.00000e+00, 5.85000e-01, ..., 8.12250e-02,
        2.85000e+00, 1.00000e+02]])
```

Abalone Age Prediction

- ## LinearRegression

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_poly,y)

LinearRegression

()

lr.predict(plr.transform([[1,0.350,0.410,0.185,1.3035,0.3635,0.1010,0
.285,16]]))

/home/lokesh/anaconda3/lib/python3.9/site-packages/sklearn/
base.py:450: UserWarning: X does not have valid feature names, but
PolynomialFeatures was fitted with feature names
  warnings.war

n(
```

```
array([[17.5]]

)
```

- ## Ridge

```
from sklearn.linear_model import
Ridger = Ridge()
r.fit(x,

y)

Ridge()

r.predict([[1,0.350,0.410,0.185,1.3035,0.3635,0.1010,0.285,16]])

/home/lokesh/anaconda3/lib/python3.9/site-packages/sklearn/
base.py:450: UserWarning: X does not have valid feature names, but
Ridge was fitted with feature names
  warnings.warn(

array([[17.49624459]

])
```

- ## Lasso

```
from sklearn.linear_model import
Lassol = Lasso()
l.fit(x,

y)

Lasso()

l.predict([[1,0.350,0.410,0.185,1.3035,0.3635,0.1010,0.285,16]])

/home/lokesh/anaconda3/lib/python3.9/site-packages/sklearn/
base.py:450: UserWarning: X does not have valid feature names, but
Lasso was fitted with feature names
  warnings.warn(

array([17.08721342

])
```