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

02.Load the Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
```

Read the Dataset

```
mydata=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/abalone.csv')
```

```
mydata.shape
```

```
(4177, 9)
```

```
mydata.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
mydata.tail()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12


```
mydata.columns
```

```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
'Viscera weight', 'Shell weight', 'Rings'],
 dtype='object')
```

```
mydata.describe()
```

Length Diameter Height Whole weight Shucked weight Viscera weight

count 4177.000000 4177.000000 4177.000000 4177.000000 4177.000000 4177.000000 4 **mean** 0.523992 0.407881 0.139516 0.828742 0.359367 0.180594 **std** 0.120093 0.099240 0.041827 0.490389 0.221963 0.109614 **min** 0.075000 0.055000 0.000000 0.002000 0.001000 0.000500 **25%** 0.450000 0.350000 0.115000 0.441500 0.186000 0.093500 **50%** 0.545000 0.425000 0.140000 0.799500 0.336000 0.171000 **75%** 0.615000 0.480000 0.165000 1.153000 0.502000 0.253000

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 # Column Non-Null Count Dtype
--- ------ -------------- -----
 0 Sex 4177 non-null object
 1 Length 4177 non-null float64
 2 Diameter 4177 non-null float64
 3 Height 4177 non-null float64
 4 Whole weight 4177 non-null float64
 5 Shucked weight 4177 non-null float64
 6 Viscera weight 4177 non-null float64
 7 Shell weight 4177 non-null float64
 8 Rings 4177 non-null int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

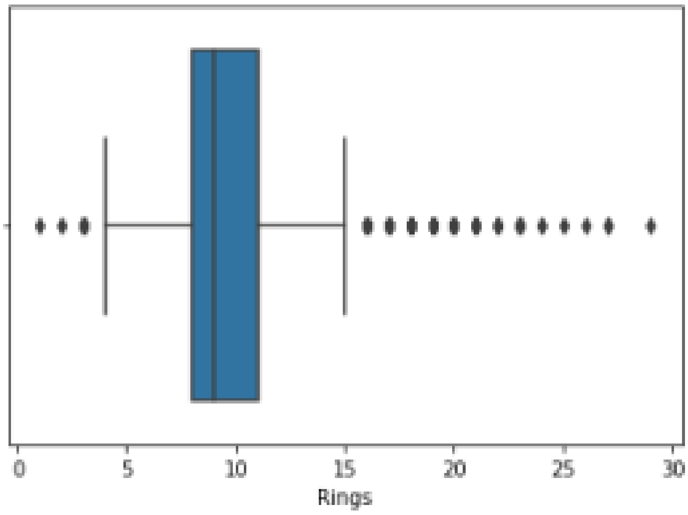
```
mydata.dtypes
```

```
Sex object
Length float64
Diameter float64
Height float64
Whole weight float64
Shucked weight float64
Viscera weight float64
Shell weight float64
Rings int64
dtype: object
```


03.Perform Virtualization

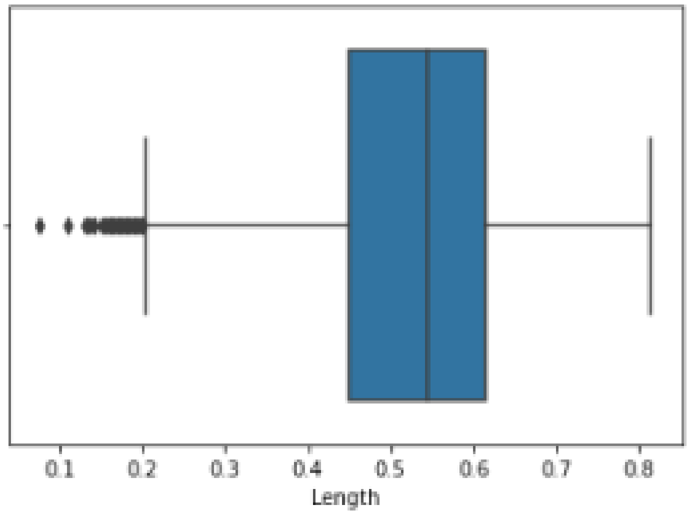
```
sns.boxplot(mydata['Rings'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f30c649e0d0>
```



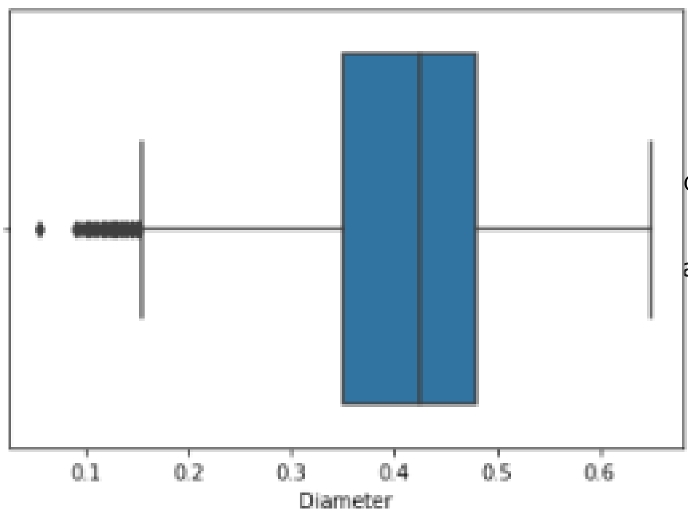

```
sns.boxplot(mydata['Length'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f30c647c410>
```




```
sns.boxplot(mydata['Diameter'])
```

Copy

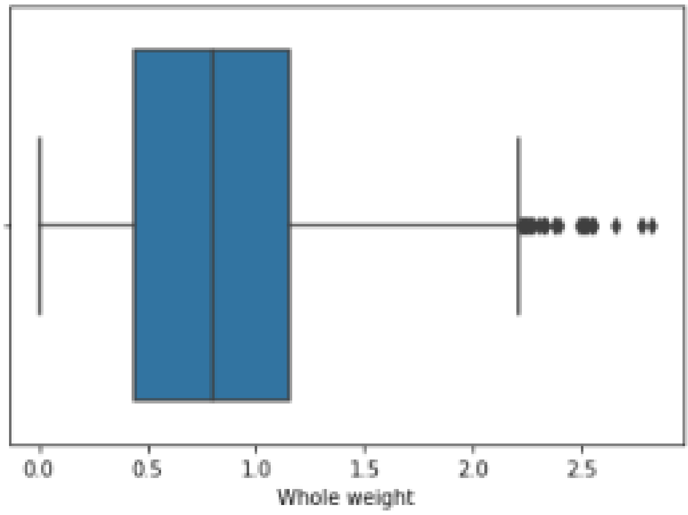


OI7LuIEeqRG6dVM4_NOdk#printMode=true

```
aborn/_decorators.py:43: FutureWarning: Pas
 0x7f30c63f23d0>
```

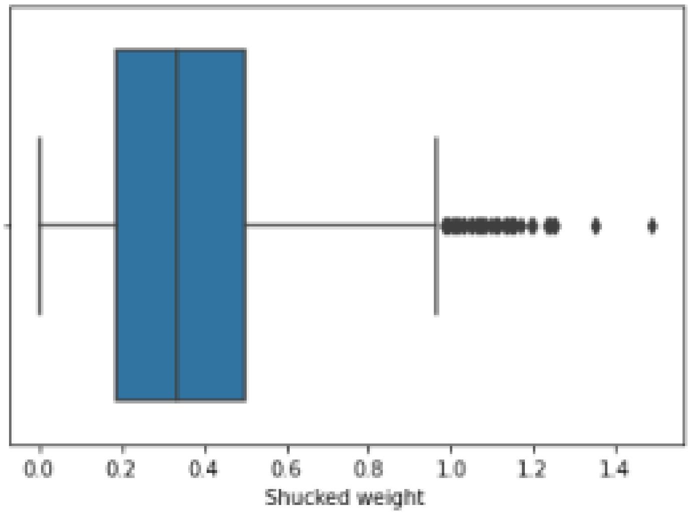
```
sns.boxplot(mydata['Whole weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f30c63d15d0>
```



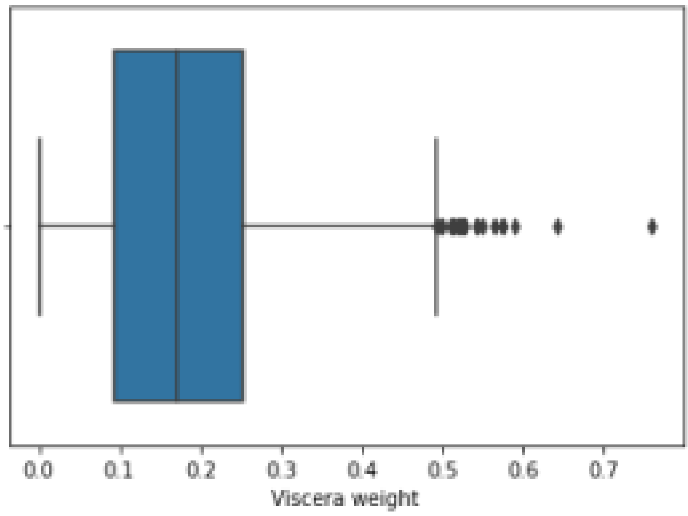

```
sns.boxplot(mydata['Shucked weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f30c6336250>
```



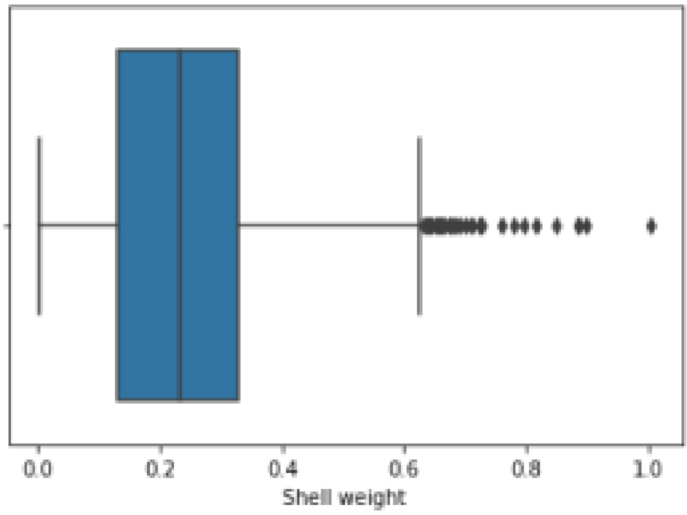

```
sns.boxplot(mydata['Viscera weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f30c62b7510>
```



```
sns.boxplot(mydata['Shell weight'])
```

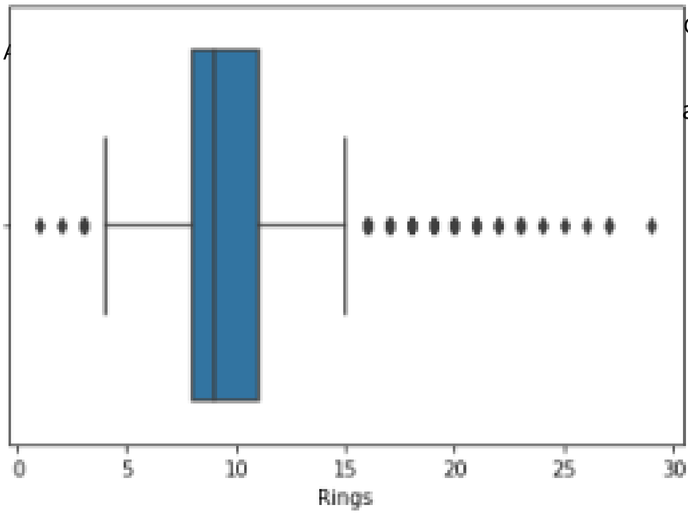
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f30c6290bd0>
```




```
sns.boxplot(mydata['Rings'])
```

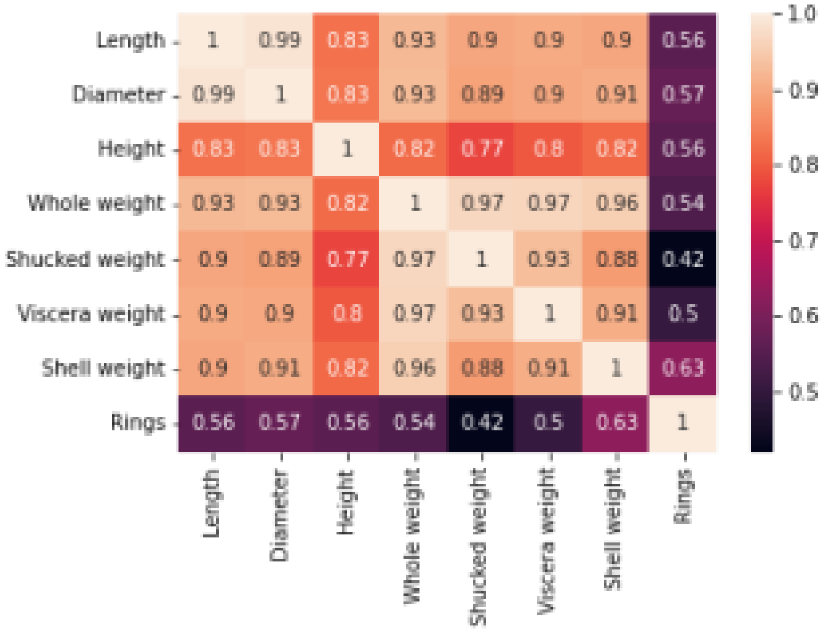


```
aborn/_decorators.py:43: FutureWarning: Pas
0x7f30c61f2910>
```




```
sns.heatmap(mydata.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f30c6184490>
```

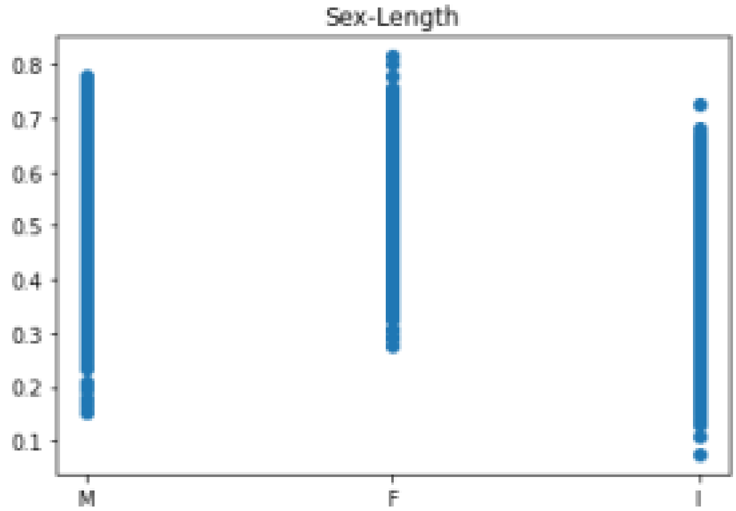


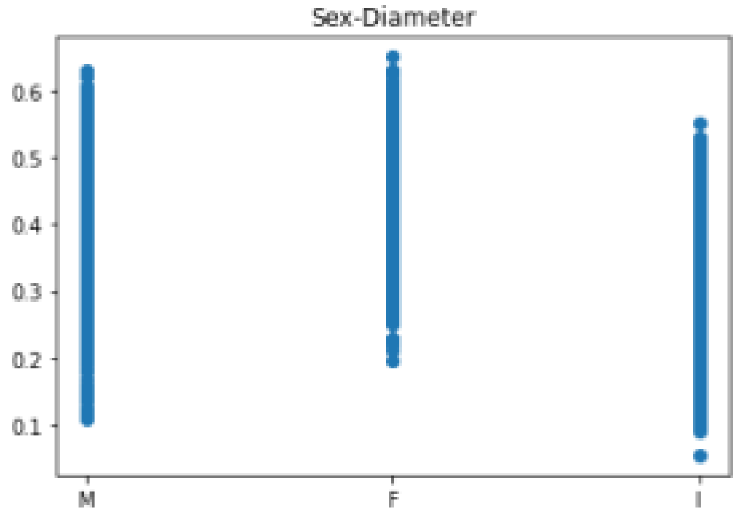

```
mydata.iloc[:,-4:-1].sum().sum(), len(mydata)
```

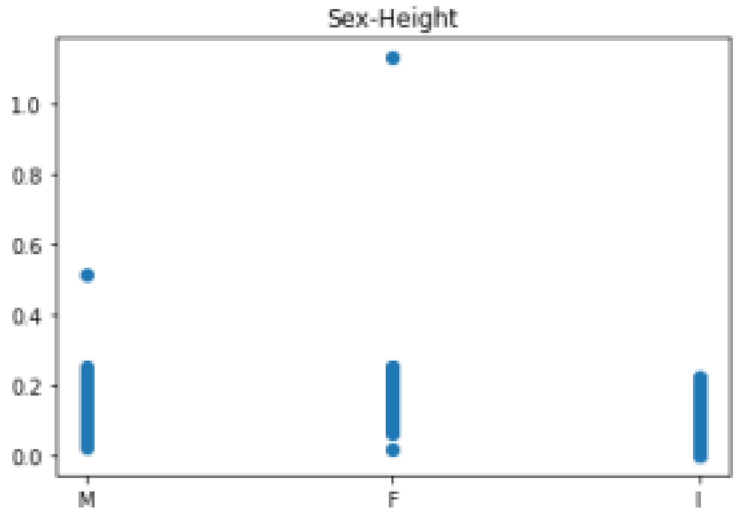
```
(3253.014, 4177)
```

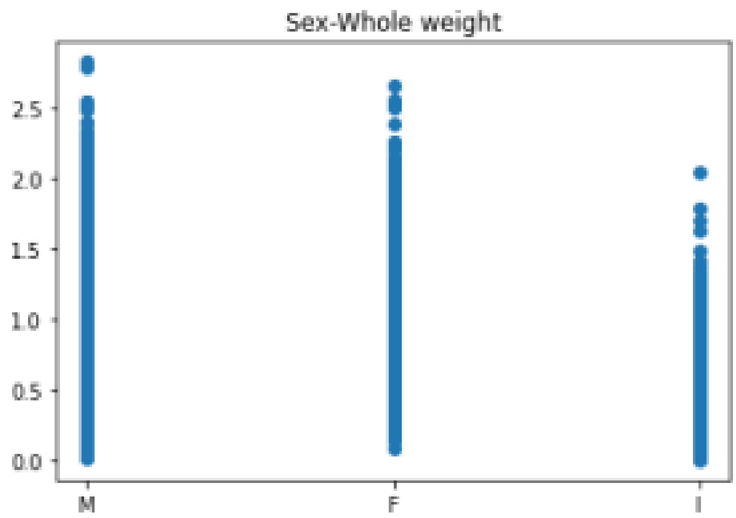
Bivariate Analysis

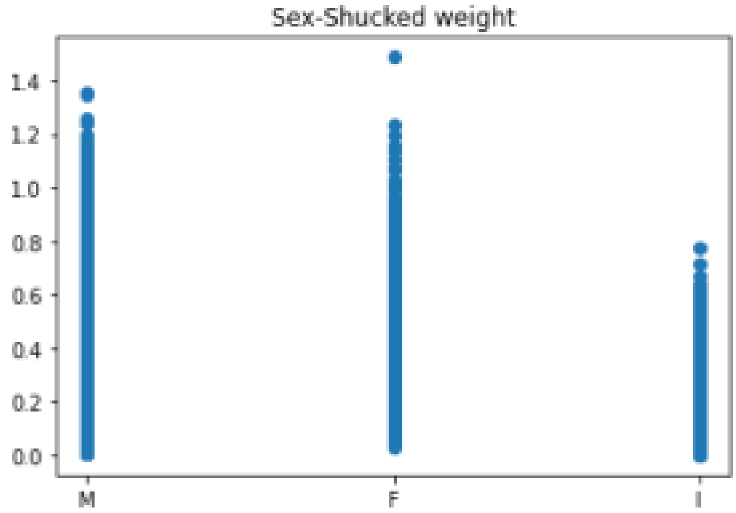
```
cols = list(mydata.iloc[:,:7].columns)
for i in range(len(cols)-1):
    for j in range(i+1, len(cols)):
        plt.scatter(mydata[cols[i]], mydata[cols[j]])
        plt.title(cols[i]+'-'+cols[j])
        plt.show()
```

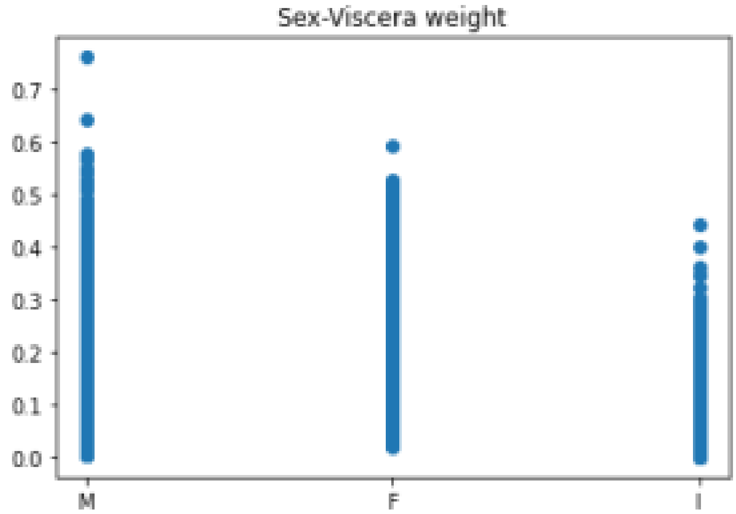

Sex-Length
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
M
F
I

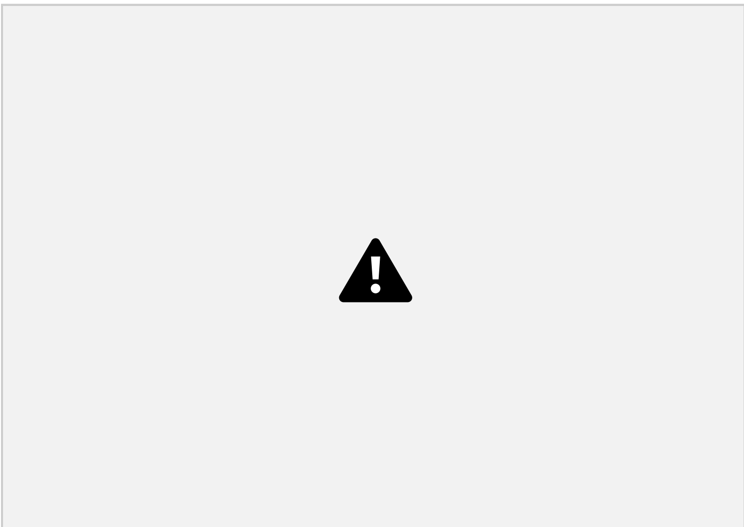

Sex-Diameter
0.6
0.5
0.4
0.3
0.2
0.1
M
F
I

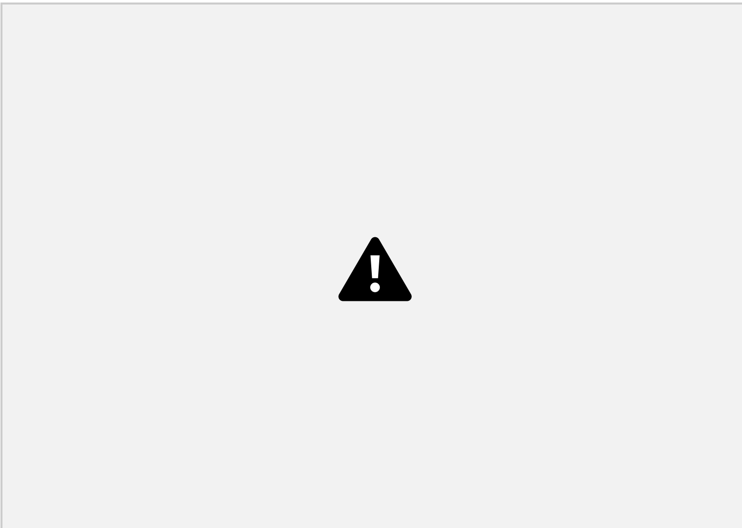

Sex-Height
1.0
0.8
0.6
0.4
0.2
0.0
M
F
I

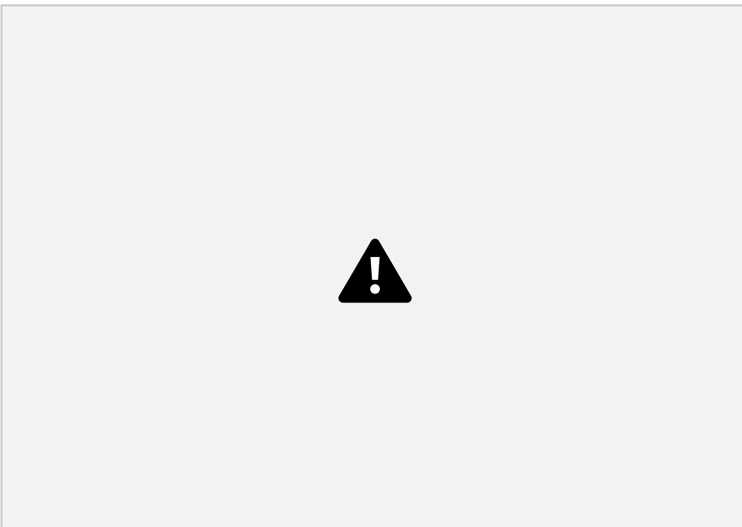

Sex-Whole weight
2.5
2.0
1.5
1.0
0.5
0.0
M
F
I

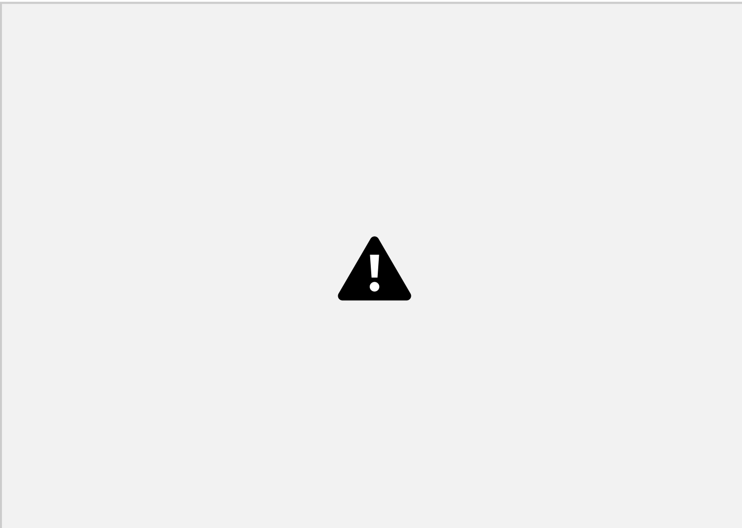

Sex-Shucked weight
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
M
F
I

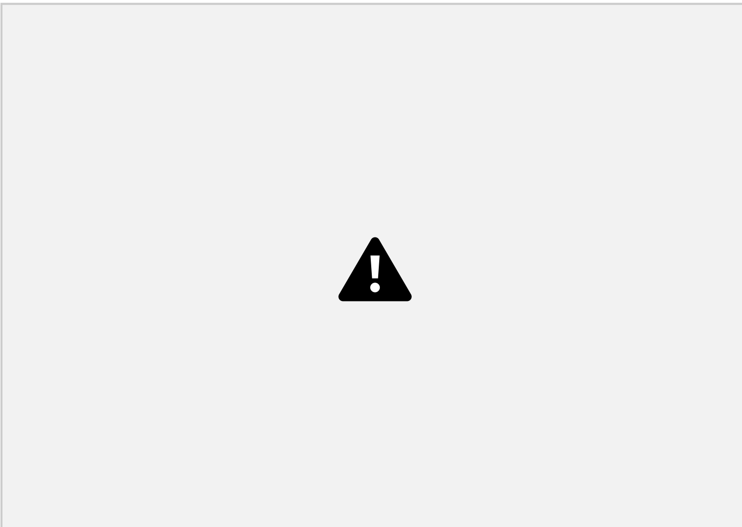

Sex-Viscera weight
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
M
F
I

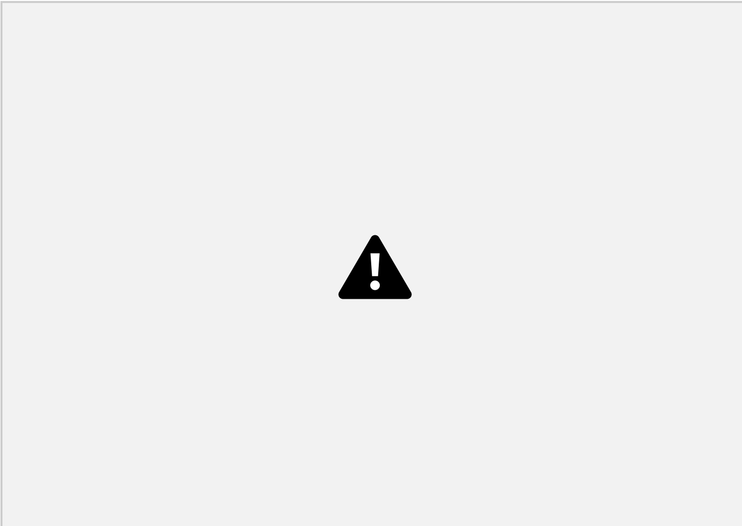


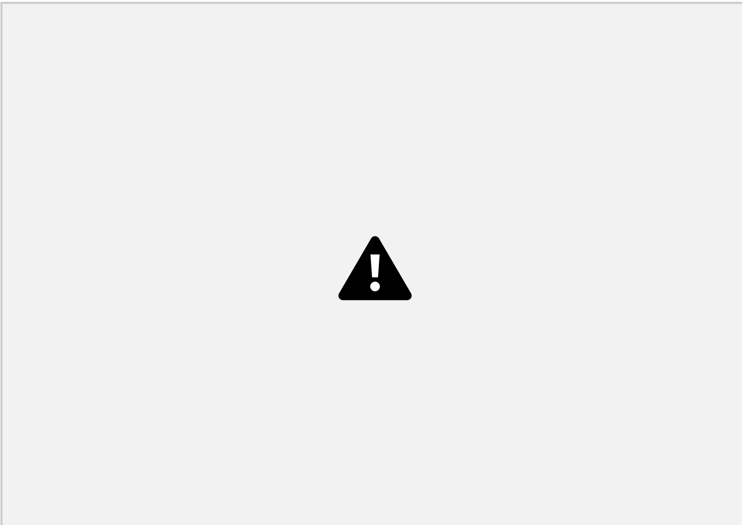


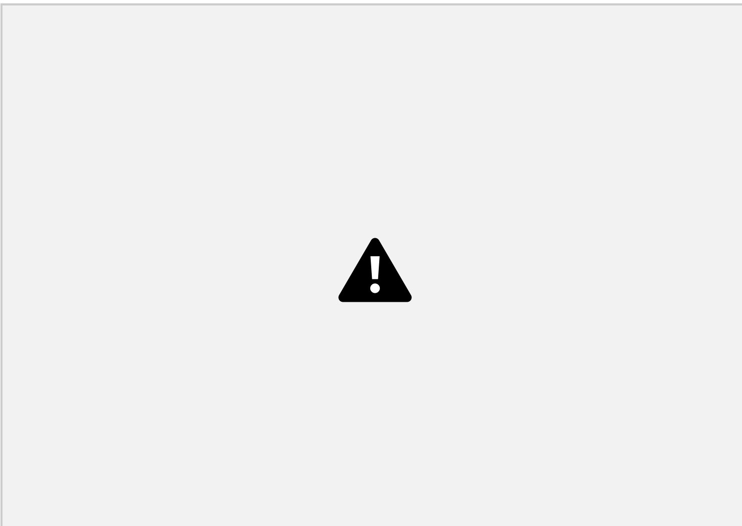


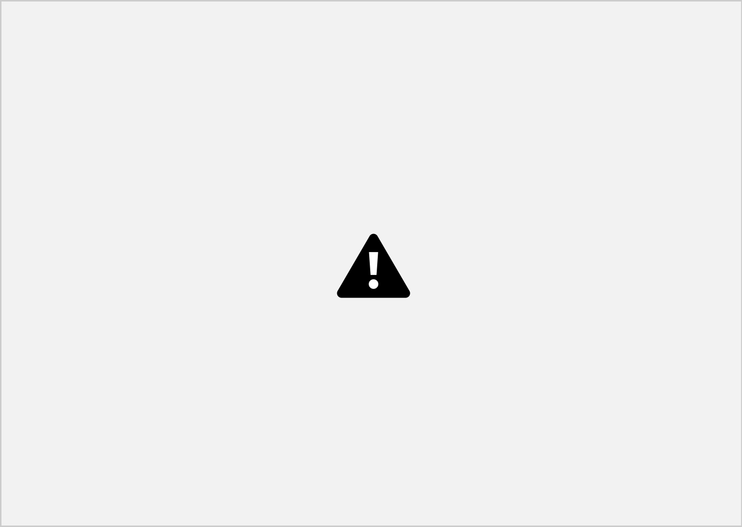


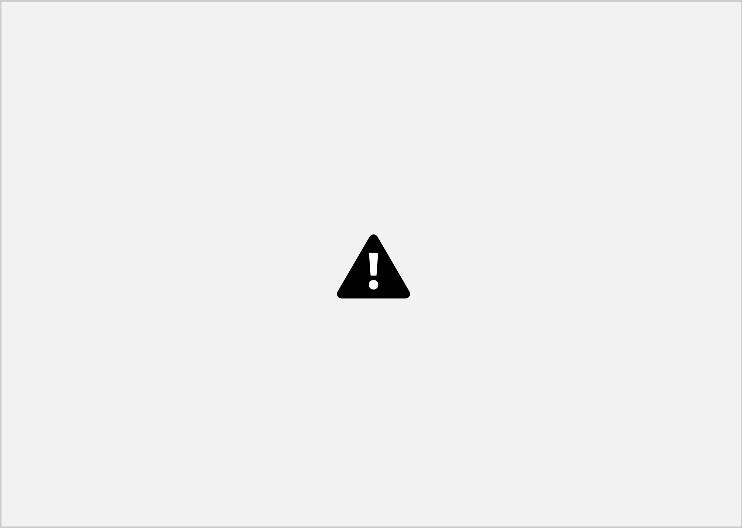












04.Perform Descriptive Statistics

```
mydata = pd.DataFrame()
```

```
mydata.sum()
```

```
Sex     MMFMIIFFMFFMMFFMIFMMMIFFFFFMMMMFMFFMFFFMFFIIII...     Length
2188.715  Diameter  1703.72  Height  582.76  Whole  weight  3461.656
Shucked  weight  1501.078  Viscera  weight  754.3395  Shell  weight
997.5965 Rings 41493 dtype: object
```

```
mydata.sum(1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropp
"""Entry point for launching an IPython kernel.
0 16.9045
1 8.1485
2 11.3700
3 11.9305
4 8.0540
 ...
4172 13.9250
4173 13.0450
4174 12.5770
4175 13.4425
4176 17.2255
Length: 4177, dtype: float64
```

```
mydata.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropp
"""Entry point for launching an IPython kernel.
Length 0.523992
Diameter 0.407881
Height 0.139516
Whole weight 0.828742
Shucked weight 0.359367
Viscera weight 0.180594
Shell weight 0.238831
```



```
Rings 9.933684
dtype: float64
```

```
mydata.std()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropp
```

```
"""Entry point for launching an IPython kernel.
Length 0.120093
Diameter 0.099240
Height 0.041827
Whole weight 0.490389
Shucked weight 0.221963
Viscera weight 0.109614
Shell weight 0.139203
Rings 3.224169
dtype: float64
```

```
mydata.count()
```

```
Sex 4177
Length 4177
Diameter 4177
Height 4177
Whole weight 4177
Shucked weight 4177
Viscera weight 4177
Shell weight 4177
Rings 4177
dtype: int64
```

```
mydata.min()
```

```
Sex F
Length 0.075
Diameter 0.055
Height 0.0
Whole weight 0.002
Shucked weight 0.001
Viscera weight 0.0005
Shell weight 0.0015
Rings 1
dtype: object
```

```
mydata.describe
```

```
<bound method NDFrame.describe of Sex Length Diameter Height Whole weight  Shucked
weight \
0 M 0.455 0.365 0.095 0.5140 0.2245
1 M 0.350 0.265 0.090 0.2255 0.0995
2 F 0.530 0.420 0.135 0.6770 0.2565
3 M 0.440 0.365 0.125 0.5160 0.2155
4 I 0.330 0.255 0.080 0.2050 0.0895
... .. ... ... ... ... ...
4172 F 0.565 0.450 0.165 0.8870 0.3700
4173 M 0.590 0.440 0.135 0.9660 0.4390
```



```
4174 M 0.600 0.475 0.205 1.1760 0.5255  4175 F 0.625 0.485 0.150
1.0945 0.5310  4176 M 0.710 0.555 0.195 1.9485 0.9455

 Viscera weight Shell weight Rings
0 0.1010 0.1500 15
1 0.0485 0.0700 7
2 0.1415 0.2100 9
3 0.1140 0.1550 10
```

```
4 0.0395 0.0550 7
... ... ... ...
4172 0.2390 0.2490 11
4173 0.2145 0.2605 10
4174 0.2875 0.3080 9
4175 0.2610 0.2960 10
4176 0.3765 0.4950 12

[4177 rows x 9 columns]>
```

05.Handling Missig Values

```
mydata.duplicated().sum()
```

```
0
```

```
mydata.isna().sum()
```

```
Sex 0
Length 0
Diameter 0
Height 0
Whole weight 0
Shucked weight 0
Viscera weight 0
Shell weight 0
Rings 0
dtype: int64
```

```
mydata.nunique()
```

```
Sex 3
Length 134
Diameter 111
Height 51
Whole weight 2429
Shucked weight 1515
Viscera weight 880
Shell weight 926
Rings 28
dtype: int64
```

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```



```
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 # Column Non-Null Count Dtype
--- ------ -------------- -----
 0 Sex 4177 non-null object
 1 Length 4177 non-null float64
 2 Diameter 4177 non-null float64
 3 Height 4177 non-null float64
 4 Whole weight 4177 non-null float64
 5 Shucked weight 4177 non-null float64
```

```
 6 Viscera weight 4177 non-null float64
 7 Shell weight 4177 non-null float64
 8 Rings 4177 non-null int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
mydata.drop(columns=['Whole weight','Shucked weight','Viscera weight','Shell weight']).des
```

	Length	Diameter	Height	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	9.933684
std	0.120093	0.099240	0.041827	3.224169
min	0.075000	0.055000	0.000000	1.000000
25%	0.450000	0.350000	0.115000	8.000000
50%	0.545000	0.425000	0.140000	9.000000
75%	0.615000	0.480000	0.165000	11.000000
max	0.815000	0.650000	1.130000	29.000000

```
qnt=mydata.drop(columns=['Shucked weight','Viscera weight','Shell weight'])
```

06.Find Outliers

```
qnt=mydata.drop(columns=['Sex','Viscera weight','Shucked weight']).quantile(q=[0.015,0.050
qnt
```

	Length	Diameter	Height	Whole weight	Shell weight	Rings
0.015	0.215	0.1582	0.050	0.04932	0.0150	4.0
0.050	0.295	0.2200	0.075	0.12590	0.0384	6.0
0.080	0.335	0.2500	0.080	0.17954	0.0550	6.0

```
Q1=qnt.iloc[0]
Q4=qnt.iloc[1]
Q7=qnt.iloc[2]
```



```
iqr=Q4-Q1
iqr
```

```
Length 0.08000 Diameter
0.06180  Height  0.02500
Whole   weight   0.07658
Shell   weight   0.02340
Rings   2.00000   dtype:
float64
```

```
iqr=Q7-Q1
iqr
```

```
Length 0.12000 Diameter
0.09180  Height  0.03000
Whole   weight   0.13022
Shell   weight   0.04000
Rings   2.00000   dtype:
float64
```

```
upper=qnt.iloc[2]+1.5*iqr
upper
```

```
Length 0.51500 Diameter
0.38770  Height  0.12500
Whole   weight   0.37487
Shell   weight   0.11500
Rings   9.00000   dtype:
float64
```

```
lower=qnt.iloc[0]-1.5*iqr
lower
```

```
Length 0.03500 Diameter
0.02050  Height  0.00500
Whole   weight  -0.14601
Shell   weight  -0.04500
Rings   1.00000   dtype:
float64
```

```
medium=qnt.iloc[1]-1.5*iqr
medium
```

```
Length 0.11500 Diameter
0.08230  Height  0.03000
Whole   weight  -0.06943
Shell   weight  -0.02160
Rings   3.00000   dtype:
float64
```


Replace Outliers

```
mydata['Rings']= np.where(mydata['Rings']>11.2,9.933684,mydata['Rings']) mydata['Whole
weight']= np.where(mydata['Whole weight']>3.1,2.825500,mydata['Whole weight'
```

07.Categorical Columns

```
mydata['Sex'].replace({'M': 1, 'F':0,'I':2}, inplace=True)
mydata.head(10)
```

Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings

0 1 0.455 0.365 0.095 0.5140 0.2245 0.1010 0.150 9.933684 **1** 1 0.350 0.265 0.090 0.2255 0.0995 0.0485 0.070 7.000000 **2** 0 0.530 0.420 0.135 0.6770 0.2565 0.1415 0.210 9.000000 **3** 1 0.440 0.365 0.125 0.5160 0.2155 0.1140 0.155 10.000000 **4** 2 0.330 0.255 0.080 0.2050 0.0895 0.0395 0.055 7.000000 **5** 2 0.425 0.300 0.095 0.3515 0.1410 0.0775 0.120 8.000000 **6** 0 0.530 0.415 0.150 0.7775 0.2370 0.1415 0.330 9.933684 **7** 0 0.545 0.425 0.125 0.7680 0.2940 0.1495 0.260 9.933684 **8** 1 0.475 0.370 0.125 0.5095 0.2165 0.1125 0.165 9.000000 **9** 0 0.550 0.440 0.150 0.8945 0.3145 0.1510 0.320 9.933684

Perform Encoding

```
mydata_all = mydata.drop(columns="Rings")
target = mydata['Rings']
```

```
mydata_all = pd.get_dummies(mydata_all)
mydata_all.columns
```

```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
'Viscera weight', 'Shell weight'],
 dtype='object')
```

```
mydata_all.shape
```

```
(4177, 8)
```

```
target.value_counts()
```

```
9.933684 960
9.000000 689
```



```
10.000000 634
8.000000 568
11.000000 487
7.000000 391
6.000000 259
5.000000 115
4.000000 57
3.000000 15
1.000000 1
2.000000 1
Name: Rings, dtype:
int64
```

Dropping Unwanted Columns

```
mydata
=mydata.drop(columns=[
'Length'])
mydata.head()
```

Sex	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings

0 1 0.365 0.095 0.5140 0.2245 0.1010 0.150 9.933684 **1** 1 0.265 0.090 0.2255 0.0995 0.0485 0.070 7.000000 **2** 0 0.420 0.135 0.6770 0.2565 0.1415 0.210 9.000000 **3** 1 0.365 0.125 0.5160 0.2155 0.1140 0.155 10.000000 **4** 2 0.255 0.080 0.2050 0.0895 0.0395 0.055 7.000000

08.Split the Data into Depenent and Indepenent Variable

```
Y= mydata['Rings']
mydata = mydata.drop(['Rings'], axis = 1)
Y= mydata
```

```
X=mydata.iloc[:,:-1]
X.head()
```

Sex Diameter Height Whole weight Shucked weight Viscera weight 0 1 0.365 0.095 0.5140 0.2245 0.1010 **1** 1 0.265 0.090 0.2255 0.0995 0.0485 **2** 0 0.420 0.135 0.6770 0.2565 0.1415 **3** 1 0.365 0.125 0.5160 0.2155 0.1140 **4** 2 0.255 0.080 0.2050 0.0895 0.0395

```
Y=mydata.iloc[:,-1]
Y.head()
```

```
0 0.150
```



```
1 0.070
2 0.210
3 0.155
4 0.055
Name: Shell weight, dtype: float64
```

09.Scale The Independent Variables

```
from sklearn.preprocessing import StandardScaler
```

```
cls=StandardScaler()
X=cls.fit_transform(X)
```

```
X
```

```
array([[-0.0105225 , -0.43214879, -1.06442415, -0.64189823, -0.60768536,
-0.72621157],
 [-0.0105225 , -1.439929 , -1.18397831, -1.23027711, -1.17090984,
-1.20522124],
 [-1.26630752, 0.12213032, -0.10799087, -0.30946926, -0.4634999 ,
-0.35668983],
 ...,
 [-0.0105225 , 0.67640943, 1.56576738, 0.70821206, 0.74855917,
0.97541324],
 [-1.26630752, 0.77718745, 0.25067161, 0.54199757, 0.77334105,
```

```
0.73362741],
 [-0.0105225 , 1.48263359, 1.32665906, 2.28368063, 2.64099341,
1.78744868]])
```

10.Split Data Into Training and Testing

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.4,random_state=0)
print('train data points :', len(X_train))
print('test data points :', len(X_test))
```

```
train data points : 2506
test data points : 1671
```

```
X_train.shape
```

```
(2506, 6)
```

```
X_test.shape
```

```
(1671, 6)
```

```
X_train
```



```
array([[ 1.24526253, -2.14537514, -1.78174911, -1.47399037, -1.37592355,
-1.46525506],
 [-0.0105225 , -0.73448285, -0.34709919, -0.77752109, -0.64373173,
-0.49811173],
 [-0.0105225 , 0.82757646, 0.72888826, 0.7051529 , 0.74855917,
0.84311534],
 ...,
 [-0.0105225 , 0.42446438, 0.13111745, 0.26565325, 0.46694694,
0.23636976],
 [-1.26630752, 0.82757646, 0.6093341 , 0.60827942, 0.53002808,
0.51008957],
 [ 1.24526253, -0.83526087, -0.70576167, -1.02531323, -1.02221858,
-0.96343541]])
```

```
X_test
```

```
array([[-0.0105225 , 0.21659075, 0.17251933, ..., 0.18101643,
-0.36887819, 0.56939553],
 [ 1.24526253, -0.1998034 , -0.07942572, ..., -0.43387519,
-0.44322382, -0.34300384],
 [-0.0105225 , 0.79954256, 0.72679844, ..., 0.87034766,
0.75531787, 1.7646387 ],
 ...,
 [ 1.24526253, 0.67462432, 0.62602042, ..., 0.22486442,
-0.09402464, 0.18618779],
 [-0.0105225 , 0.46642724, 0.47485339, ..., -0.06779544,
0.20561078, -0.12402799],
 [ 1.24526253, -1.61554351, -1.69187405, ..., -1.3577422 ,
```

```
-1.28355474, -1.34664314]])
```

11.Build the Model

1.Linear Regression

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

```
LinearRegression()
```

```
Y_train_pred = lm.predict(X_train)
Y_test_pred = lm.predict(X_test)
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
s = mean_squared_error(Y_train, Y_train_pred)
print('Mean Squared error of training set :%2f'%s)

p = mean_squared_error(Y_test, Y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```



```
Mean Squared error of training set :0.000995
Mean Squared error of testing set :0.000791
```

```
from sklearn.metrics import r2_score
s = r2_score(Y_train, Y_train_pred)
print('R2 Score of training set:%.2f'%s)

p = r2_score(Y_test, Y_test_pred)
print('R2 Score of testing set:%.2f'%p)
```

```
R2 Score of training set:0.95
R2 Score of testing set:0.96
```

2.Ridge

```
from sklearn.linear_model import Ridge
```

```
ridge_mod = Ridge(alpha=0.01, normalize=True)
ridge_mod.fit(X_train, Y_train)
ridge_mod.fit(X_test, Y_test)
ridge_model_pred = ridge_mod.predict(X_test)
ridge_mod.score(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_base.py:145: FutureWarn
```

```
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing

from sklearn.pipeline import make_pipeline

model = make_pipeline(StandardScaler(with_mean=False), Ridge())

 If you wish to pass a sample_weight parameter, you need to pass it as a fit paramete

kwargs = {s[0] + '__sample_weight': sample_weight for s in model.steps}
model.fit(X, y, **kwargs)

Set parameter alpha to: original_alpha * n_samples.
 FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_base.py:145: FutureWarn
If you wish to scale the data, use Pipeline with a StandardScaler in a preprocessing

from sklearn.pipeline import make_pipeline

model = make_pipeline(StandardScaler(with_mean=False), Ridge())

 If you wish to pass a sample_weight parameter, you need to pass it as a fit paramete

kwargs = {s[0] + '__sample_weight': sample_weight for s in model.steps}
model.fit(X, y, **kwargs)

Set parameter alpha to: original_alpha * n_samples.
 FutureWarning,
0.9290023156453548
```

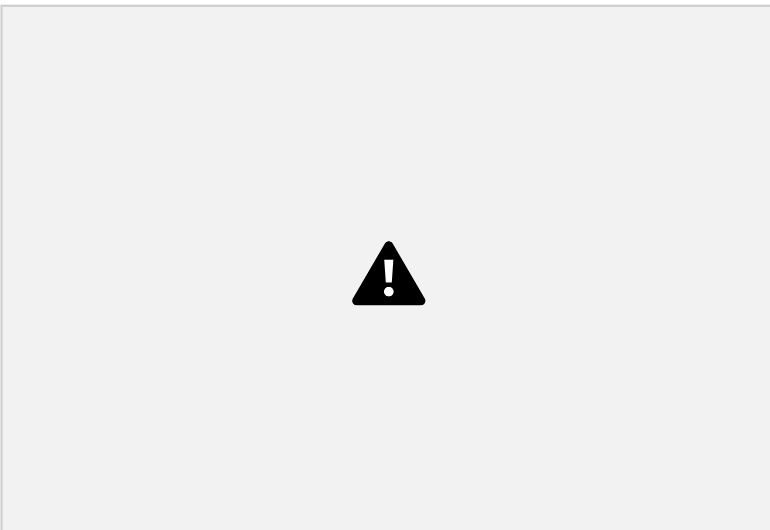


```
ridge_mod.score(X_test, Y_test)
```

```
0.9519921522913208
```

```
plt.scatter(Y_test, ridge_model_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
```

```
Text(0, 0.5, 'Predictions')
```



3.Support Vector Regression

```
from sklearn.svm import SVR
```

```
svr = SVR(kernel = 'linear')
svr.fit(X_train, Y_train)
svr.fit(X_test, Y_test)
```

```
SVR(kernel='linear')
```

```
Y_train_pred = svr.predict(X_train)
Y_test_pred = svr.predict(X_test)
svr.score(X_train, Y_train)
```

```
0.8922092465754603
```

```
svr.score(X_test, Y_test)
```

```
0.93330364267072
```

4.Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
```