## Abalone Age Prediction

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

## Load the dataset

```
data=pd.read_csv("/content/sample_data/abalone.csv")
data.head()
```

```
  Sex  Length  Diameter  Height  Whole weight  Shucked weight  Viscera
weight  \
0   M   0.455     0.365   0.095        0.5140          0.2245
0.1010
1   M   0.350     0.265   0.090        0.2255          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
0         0.150     15
1         0.070      7
2         0.210      9
3         0.155     10
4         0.055      7
```

```
Age=1.5+data.Rings
data["Age"]=Age
data=data.rename(columns = {'Whole weight':'Whole_weight','Shucked
weight': 'Shucked_weight','Viscera weight': 'Viscera_weight',
                            'Shell weight': 'Shell_weight'})
data=data.drop(columns=["Rings"],axis=1)
data.head()
```

```
  Sex  Length  Diameter  Height  Whole_weight  Shucked_weight
Viscera_weight  \
0   M   0.455     0.365   0.095        0.5140          0.2245
0.1010
1   M   0.350     0.265   0.090        0.2255          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
```

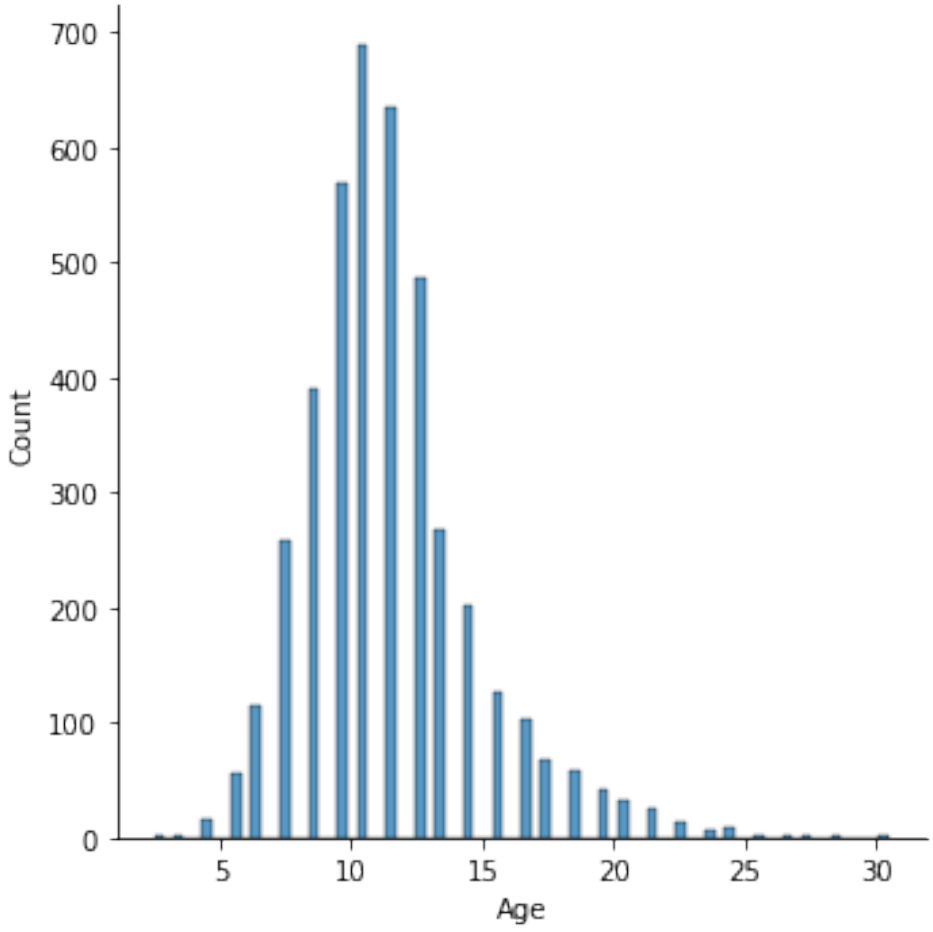
```
0.1140
4   I   0.330     0.255   0.080        0.2050          0.0895
0.0395
```

```
   Shell_weight   Age
0         0.150  16.5
1         0.070   8.5
2         0.210  10.5
3         0.155  11.5
4         0.055   8.5
```

#Univariate Analysis

```
sns.displot(data["Age"])
```

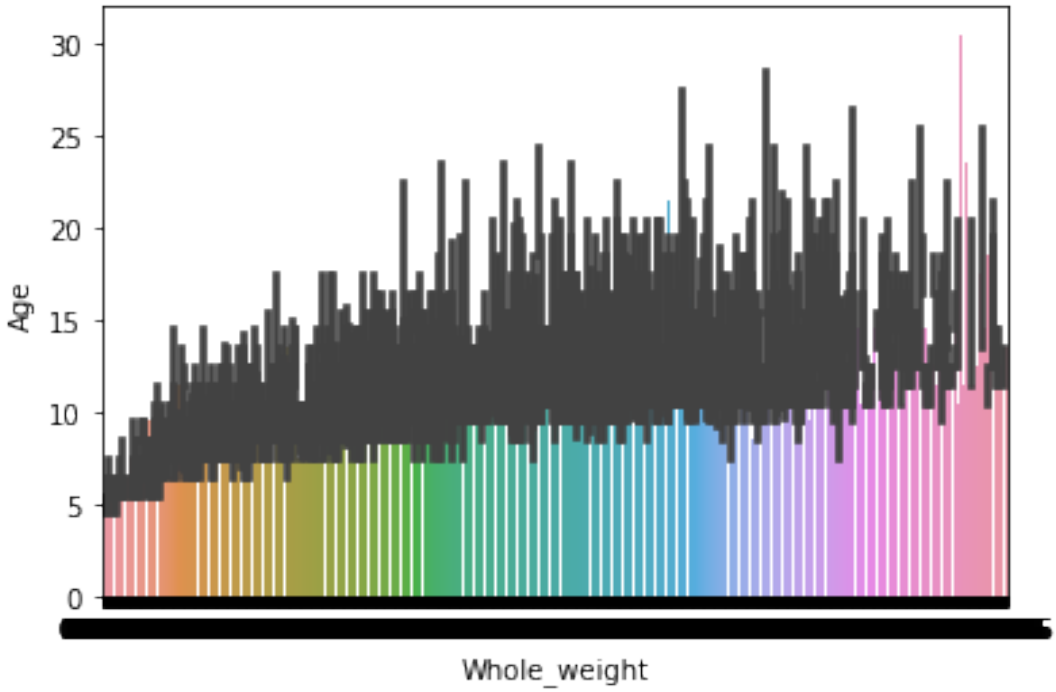
```
<seaborn.axisgrid.FacetGrid at 0x7fd9225619d0>
```




#Bi-Variate Analysis

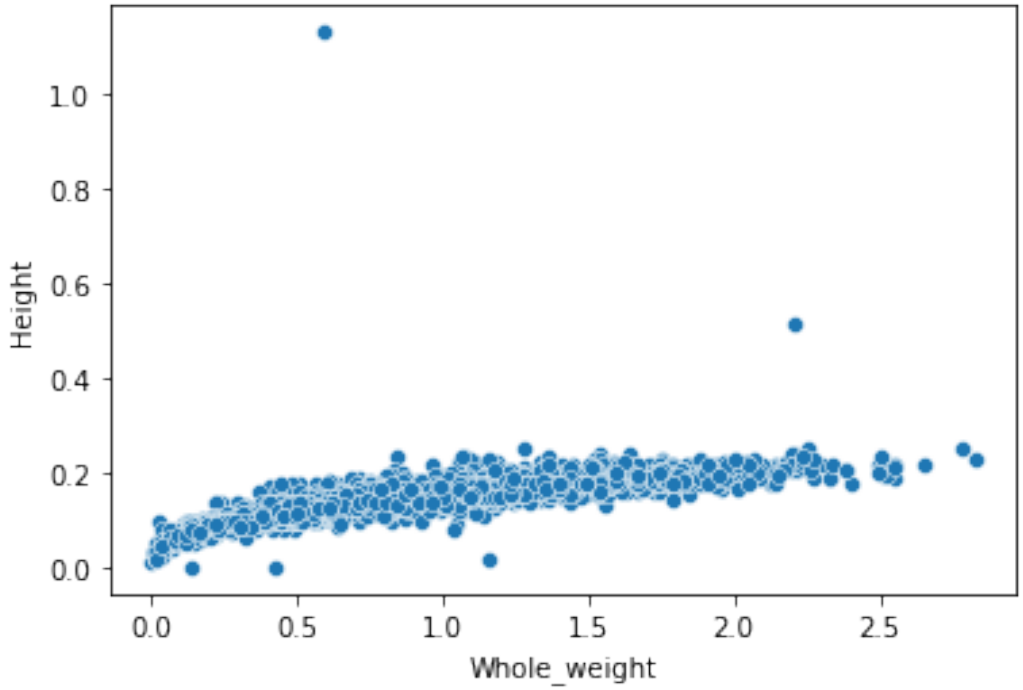
```
sns.barplot(x=data.Whole_weight,y=data.Age)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd91fbc3950>
```




```
sns.scatterplot(x=data.Whole_weight,y=data.Height)
```

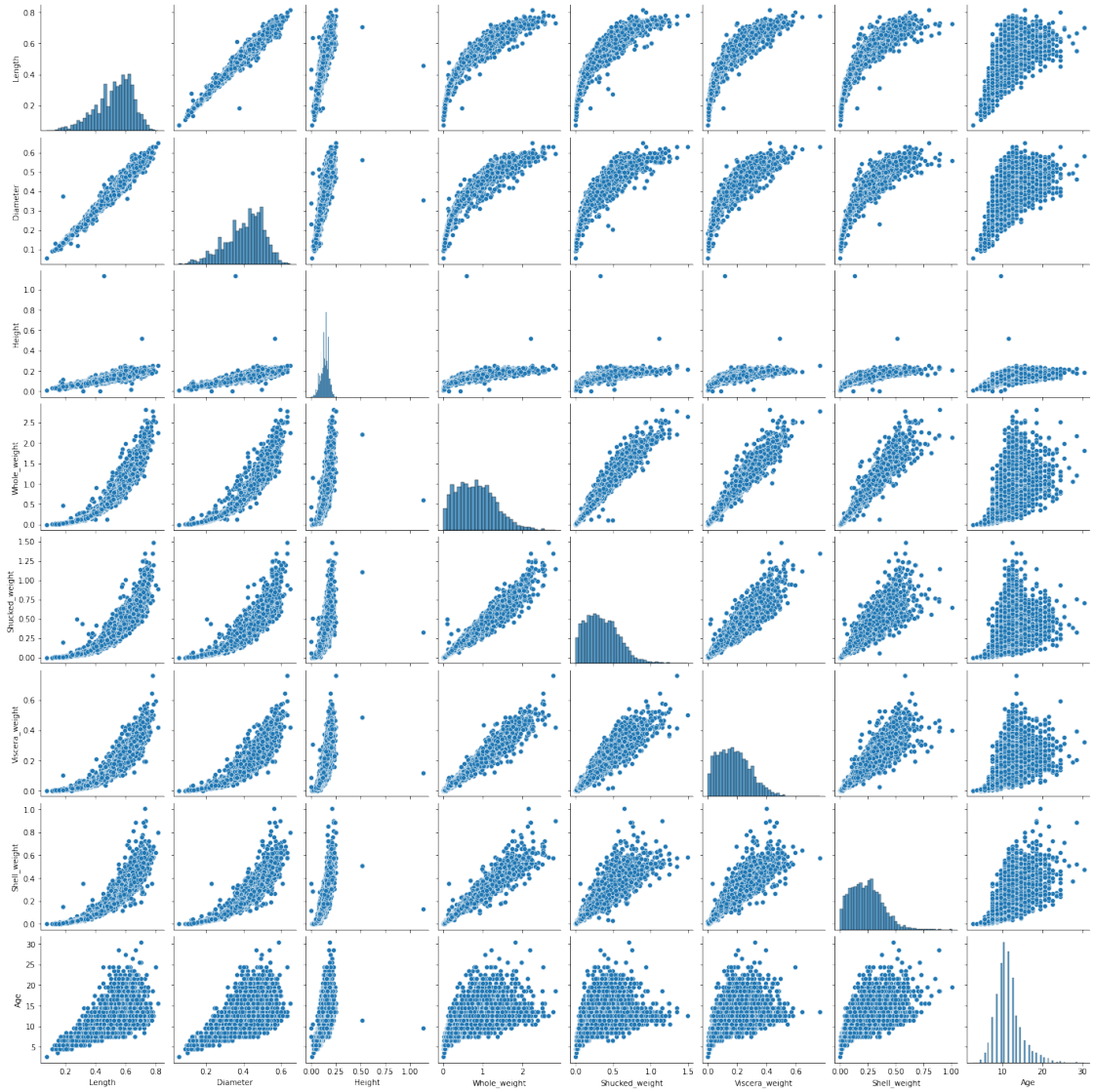
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd91f1651d0>
```




#Multi-variate Analysis

```
sns.pairplot(data=data[["Length","Diameter","Height","Whole_weight","S
hucked_weight","Viscera_weight","Shell_weight","Age",]])
```

```
<seaborn.axisgrid.PairGrid at 0x7fd91fbe93d0>
```




#Descriptive statistics

```
data.describe(include='all')
```

```
          Sex       Length     Diameter       Height  Whole_weight  \
count    4177  4177.000000  4177.000000  4177.000000   4177.000000
unique      3          NaN          NaN          NaN           NaN
top         M          NaN          NaN          NaN           NaN
freq     1528          NaN          NaN          NaN           NaN
mean      NaN     0.523992     0.407881     0.139516      0.828742
std       NaN     0.120093     0.099240     0.041827      0.490389
min       NaN     0.075000     0.055000     0.000000      0.002000
25%       NaN     0.450000     0.350000     0.115000      0.441500
50%       NaN     0.545000     0.425000     0.140000      0.799500
75%       NaN     0.615000     0.480000     0.165000      1.153000
```

```
max      NaN     0.815000     0.650000     1.130000      2.825500
```

```
        Shucked_weight  Viscera_weight  Shell_weight          Age
count      4177.000000     4177.000000   4177.000000  4177.000000
unique             NaN             NaN           NaN          NaN
top                NaN             NaN           NaN          NaN
freq               NaN             NaN           NaN          NaN
mean          0.359367        0.180594      0.238831    11.433684
std           0.221963        0.109614      0.139203     3.224169
min           0.001000        0.000500      0.001500     2.500000
25%           0.186000        0.093500      0.130000     9.500000
50%           0.336000        0.171000      0.234000    10.500000
75%           0.502000        0.253000      0.329000    12.500000
max           1.488000        0.760000      1.005000    30.500000
```

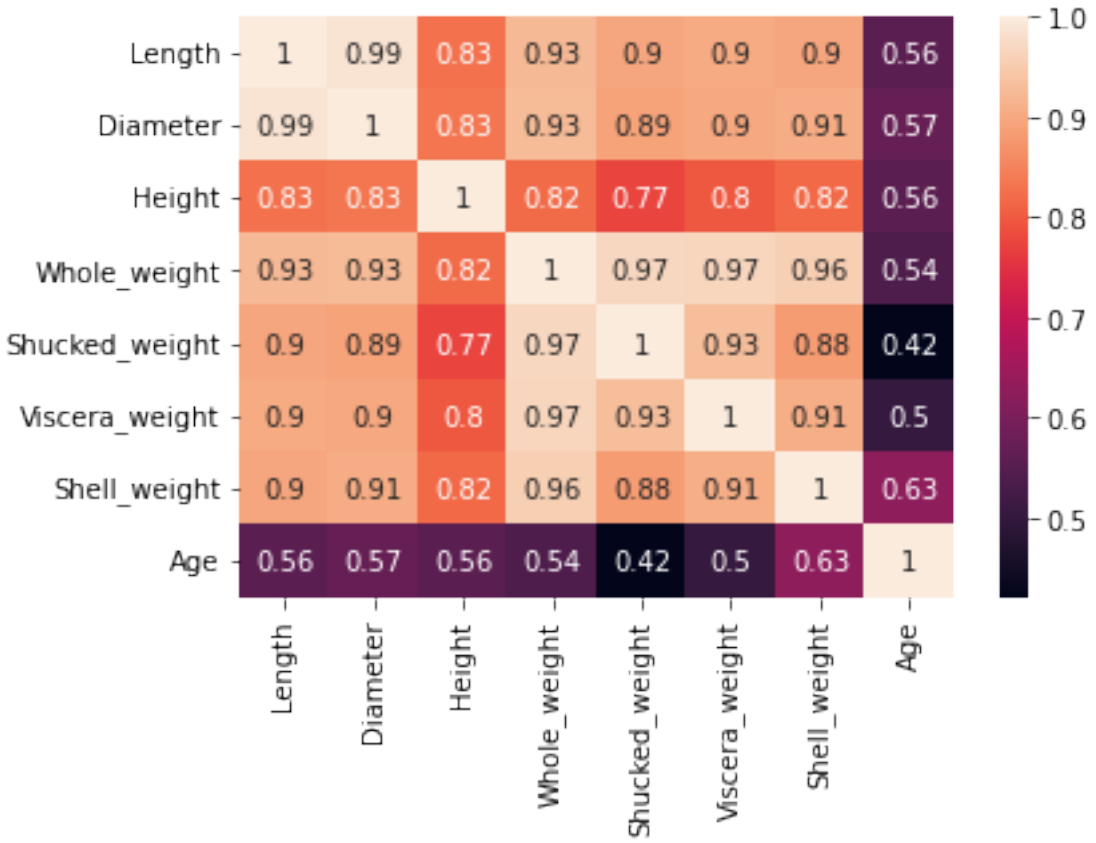
```
data.corr()
```

```
                  Length  Diameter    Height  Whole_weight
Shucked_weight  \
Length          1.000000  0.986812  0.827554      0.925261
0.897914
Diameter        0.986812  1.000000  0.833684      0.925452
0.893162
Height          0.827554  0.833684  1.000000      0.819221
0.774972
Whole_weight    0.925261  0.925452  0.819221      1.000000
0.969405
Shucked_weight  0.897914  0.893162  0.774972      0.969405
1.000000
Viscera_weight  0.903018  0.899724  0.798319      0.966375
0.931961
Shell_weight    0.897706  0.905330  0.817338      0.955355
0.882617
Age             0.556720  0.574660  0.557467      0.540390
0.420884

                Viscera_weight  Shell_weight       Age
Length                0.903018      0.897706  0.556720
Diameter              0.899724      0.905330  0.574660
Height                0.798319      0.817338  0.557467
Whole_weight          0.966375      0.955355  0.540390
Shucked_weight        0.931961      0.882617  0.420884
Viscera_weight        1.000000      0.907656  0.503819
Shell_weight          0.907656      1.000000  0.627574
Age                   0.503819      0.627574  1.000000
```

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd9180453d0>
```




#Handle The Missing values

```
data.isnull().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
Age               0
dtype: int64
```

#Find the outliers

```
outliers=data.quantile(q=(0.25,0.75))
outliers
```

```
      Length  Diameter  Height  Whole_weight  Shucked_weight
Viscera_weight  \
0.25   0.450      0.35   0.115        0.4415           0.186
0.0935
0.75   0.615      0.48   0.165        1.1530           0.502
```

```
0.2530
```

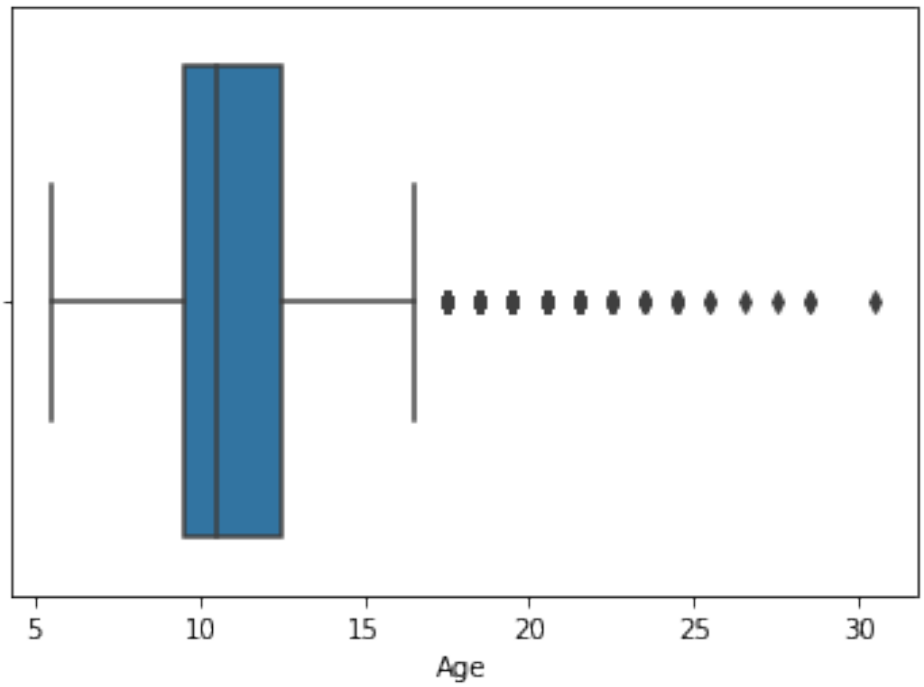
```
      Shell_weight   Age
0.25         0.130   9.5
0.75         0.329  12.5
```

```
a = data.Age.quantile(0.25)
b = data.Age.quantile(0.75)
c = b - a
lower_limit = a - 1.5 * c
data.median(numeric_only=True)
```

```
Length             0.5450
Diameter           0.4250
Height             0.1400
Whole_weight       0.7995
Shucked_weight     0.3360
Viscera_weight     0.1710
Shell_weight       0.2340
Age               10.5000
dtype: float64
```

```
data['Age'] = np.where(data['Age'] < lower_limit, 7, data['Age'])
sns.boxplot(x=data.Age)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd916270e50>
```




#Categorical colums and encoding

```
data.head()
```

```
  Sex  Length  Diameter  Height  Whole_weight  Shucked_weight
Viscera_weight  \
0   M   0.455     0.365   0.095        0.5140          0.2245
0.1010
1   M   0.350     0.265   0.090        0.2255          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   Age
0         0.150  16.5
1         0.070   8.5
2         0.210  10.5
3         0.155  11.5
4         0.055   8.5
```

```
from sklearn.preprocessing import LabelEncoder
labenc = LabelEncoder()
data.Sex = labenc.fit_transform(data.Sex)
data.head()
```

```
   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

   Viscera_weight  Shell_weight   Age
0          0.1010         0.150  16.5
1          0.0485         0.070   8.5
2          0.1415         0.210  10.5
3          0.1140         0.155  11.5
4          0.0395         0.055   8.5
```

#Split the data into dependent and independent variables.

```
x=data.drop(columns=["Sex"],axis=1)
y = data["Sex"]
print(x.head())
print(y.head())
```

```
   Length  Diameter  Height  Whole_weight  Shucked_weight
Viscera_weight  \
0   0.455     0.365   0.095        0.5140          0.2245
0.1010
1   0.350     0.265   0.090        0.2255          0.0995
```

```
0.0485
2   0.530     0.420   0.135        0.6770          0.2565
0.1415
3   0.440     0.365   0.125        0.5160          0.2155
0.1140
4   0.330     0.255   0.080        0.2050          0.0895
0.0395

   Shell_weight   Age
0         0.150  16.5
1         0.070   8.5
2         0.210  10.5
3         0.155  11.5
4         0.055   8.5
0    2
1    2
2    0
3    2
4    1
Name: Sex, dtype: int64
```

#Scale the independent variables

```
from sklearn.preprocessing import scale
X_Scaled = pd.DataFrame(scale(x), columns=x.columns)
X_Scaled.head()
```

```
     Length  Diameter    Height  Whole_weight  Shucked_weight
Viscera_weight  \
0 -0.574558 -0.432149 -1.064424     -0.641898       -0.607685       -
0.726212
1 -1.448986 -1.439929 -1.183978     -1.230277       -1.170910       -
1.205221
2  0.050033  0.122130 -0.107991     -0.309469       -0.463500       -
0.356690
3 -0.699476 -0.432149 -0.347099     -0.637819       -0.648238       -
0.607600
4 -1.615544 -1.540707 -1.423087     -1.272086       -1.215968       -
1.287337

   Shell_weight       Age
0     -0.638217  1.577830
1     -1.212987 -0.919022
2     -0.207139 -0.294809
3     -0.602294  0.017298
4     -1.320757 -0.919022
```

#Split the 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_Scaled, y,
```

```
test_size=0.2, random_state=0)
X_Train.shape,X_Test.shape
```

```
((3341, 8), (836, 8))
```

```
Y_Train.shape,Y_Test.shape
```

```
((3341,), (836,))
```

```
X_Train.head()
```

```
        Length  Diameter    Height  Whole_weight  Shucked_weight  \
3141 -2.864726 -2.750043 -1.423087     -1.622870       -1.553902
3521 -2.573250 -2.598876 -2.020857     -1.606554       -1.551650
883   1.132658  1.230689  0.728888      1.145672        1.041436
3627  1.590691  1.180300  1.446213      2.164373        2.661269
2106  0.591345  0.474853  0.370226      0.432887        0.255175

      Viscera_weight  Shell_weight       Age
3141       -1.583867     -1.644065 -1.543234
3521       -1.565619     -1.626104 -1.387181
883         0.286552      1.538726  1.577830
3627        2.330326      1.377072  0.017298
2106        0.272866      0.906479  1.265723
```

```
X_Test.head()
```

```
        Length  Diameter    Height  Whole_weight  Shucked_weight  \
668   0.216591  0.172519  0.370226      0.181016       -0.368878
1580 -0.199803 -0.079426 -0.466653     -0.433875       -0.443224
3784  0.799543  0.726798  0.370226      0.870348        0.755318
463  -2.531611 -2.447709 -2.020857     -1.579022       -1.522362
2615  1.007740  0.928354  0.848442      1.390405        1.415417

      Viscera_weight  Shell_weight       Age
668         0.569396      0.690940  0.953617
1580       -0.343004     -0.325685 -0.606915
3784        1.764639      0.565209  0.329404
463        -1.538247     -1.572219 -1.543234
2615        1.778325      0.996287  0.641511
```

```
Y_Train.head(),Y_Test.head()
```

```
(3141    1
 3521    1
 883     2
 3627    2
 2106    2
 Name: Sex, dtype: int64, 668     2
 1580    1
 3784    2
 463     1
```

```
2615    2
Name: Sex, dtype: int64)
```

## Build the Model

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=10,criterion='entropy')
model.fit(X_Train,Y_Train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

```
y_predict = model.predict(X_Test)
```

```
y_predict_train = model.predict(X_Train)
```

## Training and Testing Model

```
#train the model
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
print('Training accuracy: ',accuracy_score(Y_Train,y_predict_train))
```

```
Training accuracy:  0.9823406165818617
```

```
#test the model
print('Testing accuracy: ',accuracy_score(Y_Test,y_predict))
```

```
Testing accuracy:  0.5394736842105263
```

#Measure the performance using metrics

```
pd.crosstab(Y_Test,y_predict)
```

```
col_0    0    1    2
Sex
0      115   30  104
1       41  216   34
2      122   54  120
```

```
print(classification_report(Y_Test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.41      0.46      0.44       249
           1       0.72      0.74      0.73       291
           2       0.47      0.41      0.43       296

    accuracy                           0.54       836
   macro avg       0.53      0.54      0.53       836
```

| | | | | |
|---|---|---|---|---|
| weighted avg | 0.54 | 0.54 | 0.54 | 836 |