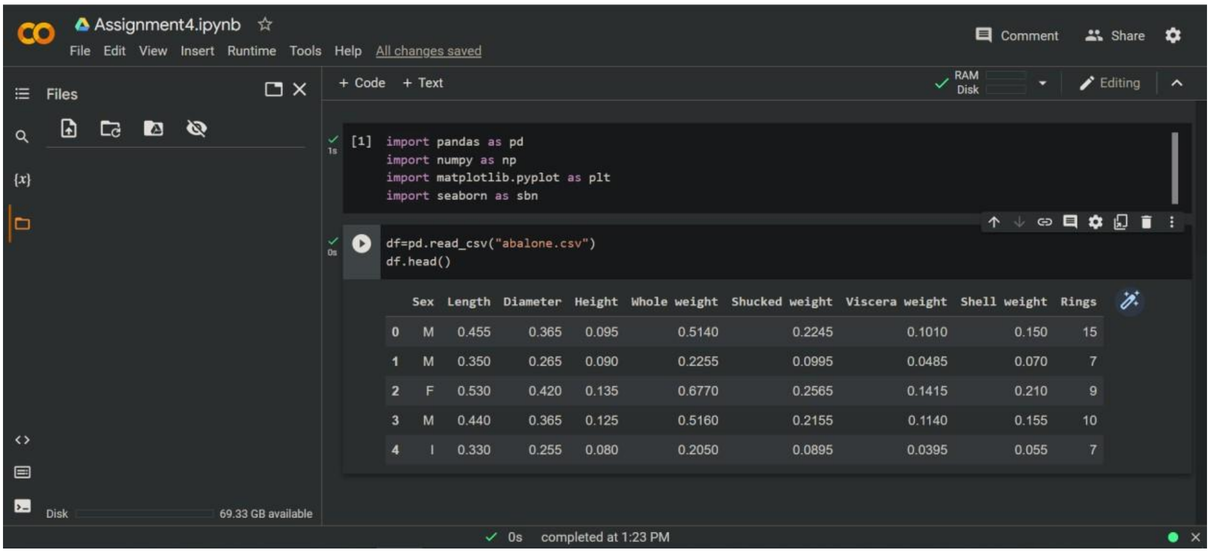
ASSIGNMENT - 4

Loading the dataset:



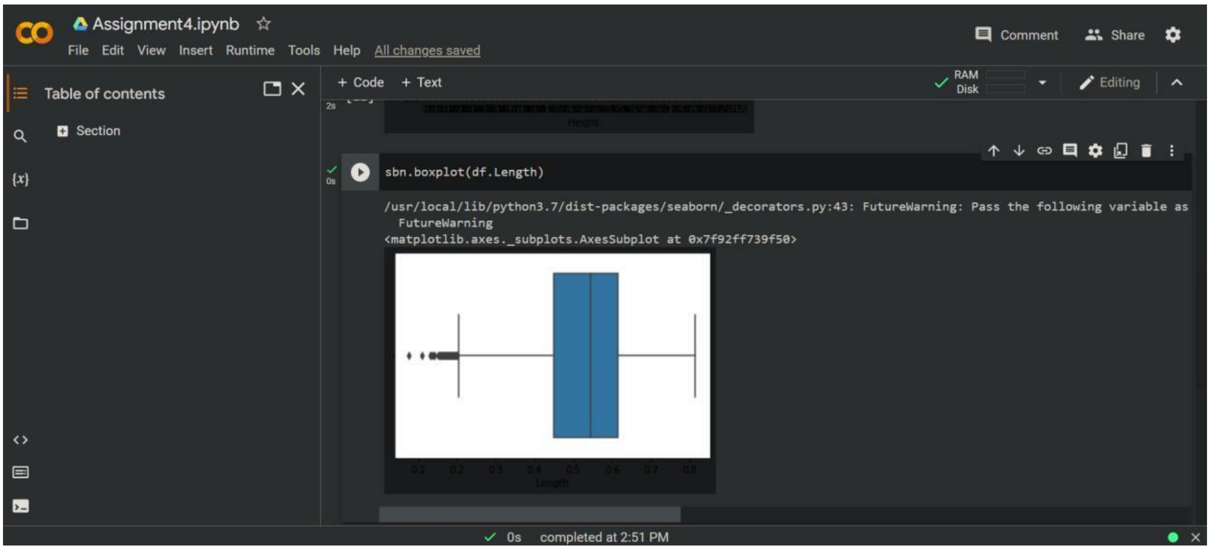
```
[1] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sbn
```

```
df=pd.read_csv("abalone.csv")
df.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

Perform Below Visualizations:

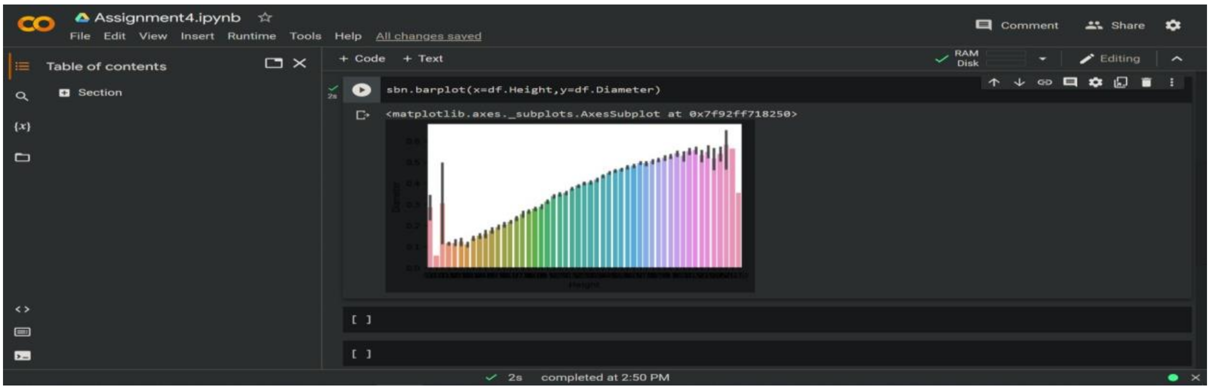
Univariate Analysis



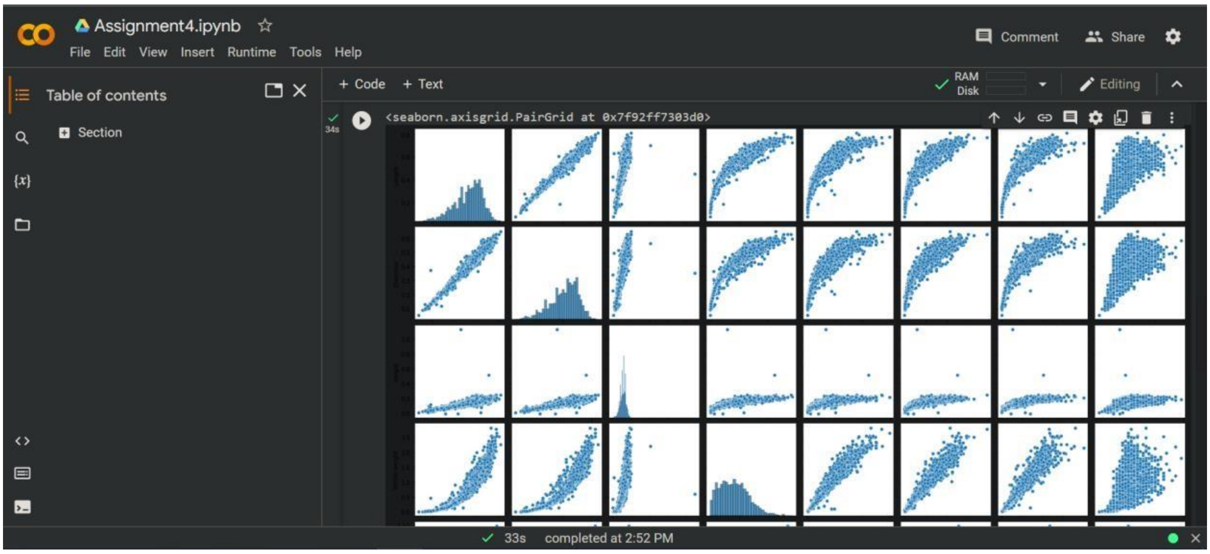
```
sbn.boxplot(df.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f92ff739f50>
```

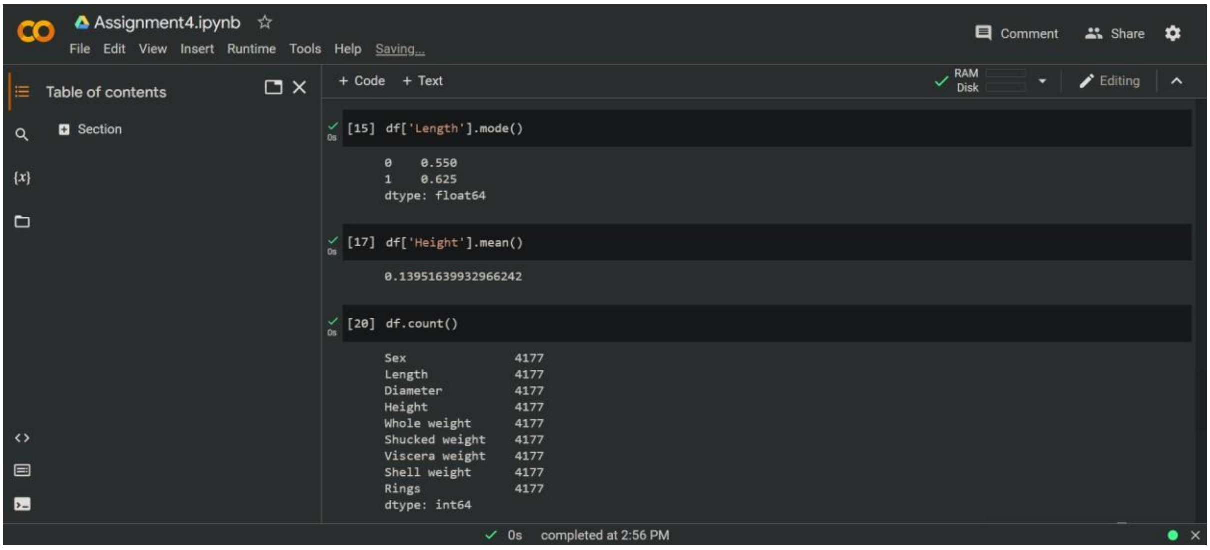
Bi-Variate Analysis

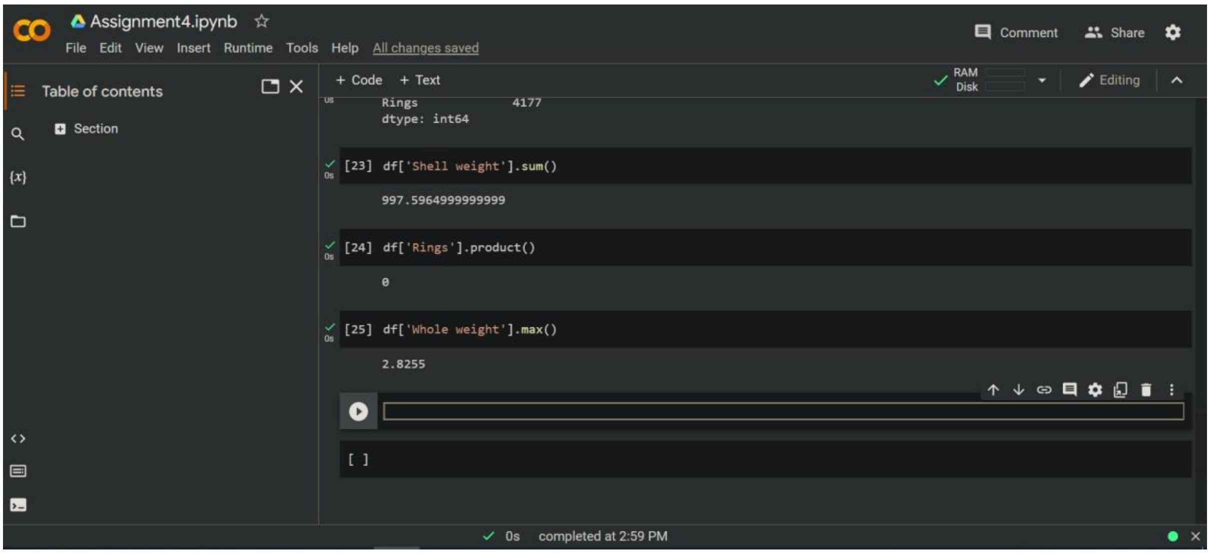


Multi-Variate Analysis

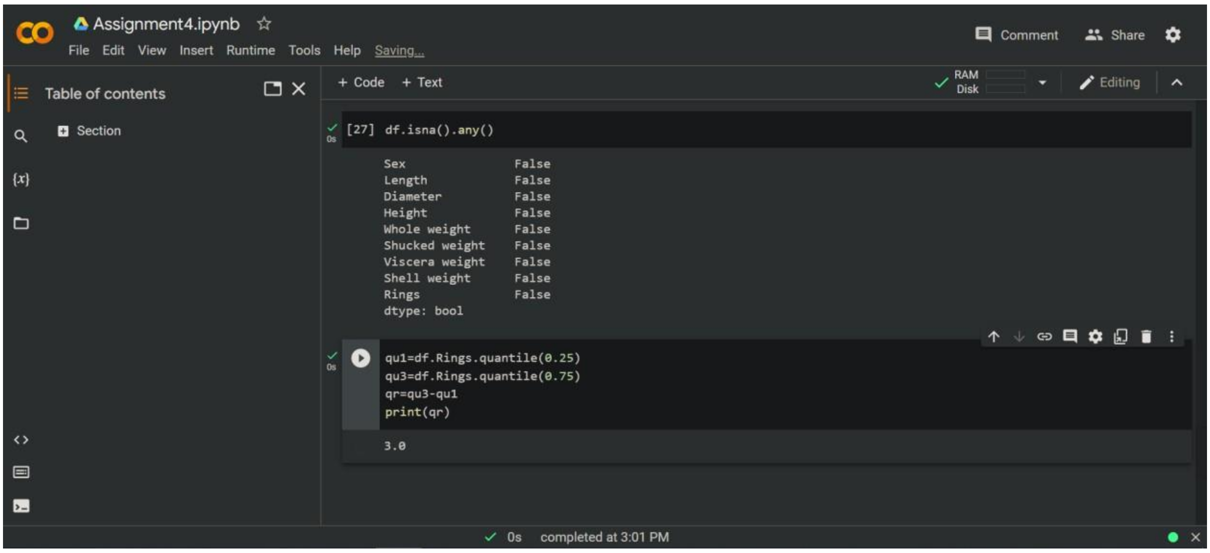


Perform descriptive analytics on the dataset





Check for Missing values and deal with them, Find the outliers and replace them outliers



Check for Categorical columns and perform encoding

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

	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	15
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

Split the data into dependent and independent variables, Scale the independent variables

```
[33] x=df.iloc[:,:-1].values
     y=df.iloc[:,-1].values
```

```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x=std.fit_transform(x)
x
```

```
array([[-0.0105225 , -0.57455813, -0.43214879, ..., -0.60768536,
        -0.72621157, -0.63821689],
       [-0.0105225 , -1.44898585, -1.439929  , ..., -1.17090984,
        -1.20522124, -1.21298732],
       [-1.26630752,  0.05003309,  0.12213032, ..., -0.4634999 ,
        -0.35668983, -0.20713907],
       ...,
       [-0.0105225 ,  0.6329849 ,  0.67640943, ...,  0.74855917,
         0.97541324,  0.49695471],
       [-1.26630752,  0.84118198,  0.77718745, ...,  0.77334105,
         0.73362741,  0.41073914],
       [-0.0105225 ,  1.54905203,  1.48263359, ...,  2.64099341,
         1.78744868,  1.84048058]])
```

Split the data into training and testing

```
[35] 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=42)
```

```
x_train
```

```
array([[ 1.24526253,  0.21659075,  0.37407537, ..., -0.32156733,
        -0.4023098 , -0.20713907],
       [ 1.24526253, -0.40800047, -0.53292681, ..., -0.47701729,
        -0.81745151, -0.71006319],
       [-1.26630752, -1.82374058, -1.84304108, ..., -1.35564747,
        -1.34208115, -1.39260308],
       ...,
       [-0.0105225 , -0.11652457, -0.12981473, ..., -0.51982235,
        -0.42968178, -0.36520094],
       [-0.0105225 ,  0.42478783,  0.57563141, ...,  0.70575411,
         0.34585768, -0.02752331],
       [-1.26630752,  0.59134549,  0.67640943, ...,  0.84543378,
         0.4599076 ,  0.23112338]])
```

```
y_train
```

```
array([11,  8,  7, ..., 11,  9,  6])
```

```
[38] x_test

array([[-0.0105225 ,  0.67462432,  0.47485339, ...,  0.27770351,
         1.10314916,  0.61909342],
       [-0.0105225 ,  0.54970607,  0.32368636, ...,  0.12450645,
         0.3139237 ,  0.04432299],
       [-1.26630752,  0.29986958,  0.37407537, ..., -0.2449688 ,
         0.40060164,  0.69093973],
       ...,
       [ 1.24526253,  0.17495134,  0.22290834, ..., -0.0309435 ,
        -0.20614393, -0.22150833],
       [ 1.24526253, -0.4912793 , -0.53292681, ..., -0.47025859,
        -0.81288951, -0.39393946],
       [ 1.24526253, -1.3240676 , -1.33915098, ..., -1.17766853,
        -1.30558517, -1.17706417]])
```

```
y_test

array([ 9,  8, 16,  9, 14, 11,  7,  6,  7, 10, 22,  7, 15,  9,  8, 18, 11,
       14, 13,  9, 20, 12, 12, 11, 10,  7, 11,  8,  9, 10,  9, 10,  6, 10,
        8,  9,  5,  3,  6,  6, 12, 12, 18,  8, 12, 13, 10, 10, 18,  4,  6,
       22,  8,  5,  7, 10, 15, 21, 10,  9, 10, 13, 11,  7,  9, 11,  4,  5,
        7,  9, 10, 11, 10,  7,  9, 12, 23, 14, 15,  9, 15, 13, 10,  6,  7,
       13,  9, 10, 19, 10, 10,  9, 11, 11, 10, 10,  6, 15,  7,  7, 15, 11,
       11, 13,  7,  9, 10,  8,  9, 14, 18,  8, 13,  9, 12,  5,  9, 12, 11,
       13, 11, 10,  8, 14,  9, 20,  9,  9,  9, 10,  9,  9, 10,  5,  8,  8,
       10, 10,  5, 12,  8, 11,  7,  8, 10, 15, 10, 14, 10, 10, 10,  8, 11,
       11,  8, 11, 12,  7,  8,  6,  9,  6, 10, 12,  7, 10, 17, 11,  8,  8,
       10, 12,  9,  8,  8,  7,  9, 11,  9, 10, 13,  7,  8,  8,  7, 10,  8,
       11,  9,  5,  9,  8, 16, 13, 11, 17, 10, 11, 12,  9,  8, 17, 11, 12,
        9, 12, 11,  9,  8, 10,  5,  9, 12,  6,  8, 11, 11,  7,  9, 12, 13,
        9, 12, 11,  9,  8,  7, 13,  9, 12,  5, 10, 10, 12,  7, 10, 10,  7,
        4, 10,  8, 11, 10,  9, 10,  8,  9,  7,  7,  6,  7,  9,  9,  7, 15,
       11,  9,  5, 12, 14, 19, 16,  9,  9,  7,  6,  7, 14, 12,  6,  9,  8,
        6, 12,  8, 18, 10, 16,  9,  6, 15,  9, 13,  8,  5,  9, 10,  5, 10,
       10, 11,  4, 15,  9, 15,  8,  5, 14,  7, 11, 10, 10,  7, 10,  9, 10,
       18,  8,  6,  5,  8,  6,  7, 14, 12, 10,  5, 23,  9,  9, 12,  7,  8,
        8, 13,  6, 13, 17,  7,  8,  8,  7,  7,  9, 14, 10,  9, 13,  8, 10,
       10,  9, 10,  9,  8,  8, 10, 13, 10,  9,  8, 10,  8, 11, 10,  3,  7,
        6,  3,  8,  8, 13, 15,  6, 14,  8,  9, 12,  8,  8, 15, 11,  9,  6,
       10, 13, 13,  7,  7,  9,  9,  8,  7, 10, 11,  5, 10, 12,  8,  7,  9,
        6,  8, 13,  7,  7, 10, 11, 23,  9, 11, 10,  8,  8,  7,  7, 10,  9,
```

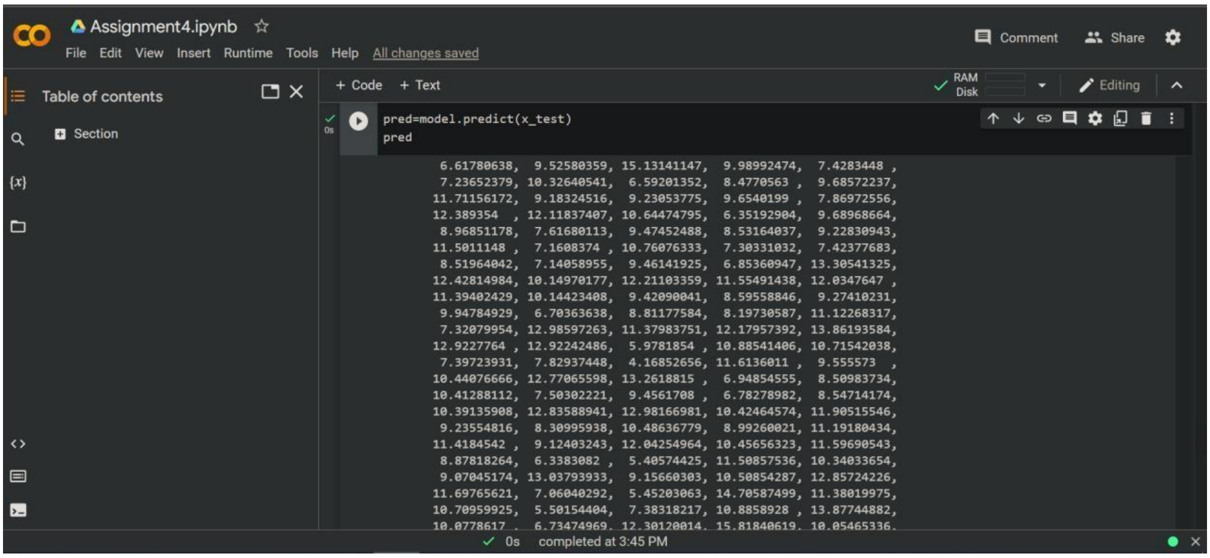
Build the Model, Train the Model

```
[40] from sklearn.ensemble import RandomForestRegressor
     model = RandomForestRegressor(n_estimators = 1000, oob_score = True,n_jobs=-1,min_samples_split = 6, min_samples_leaf= 4, max_features = 'sqrt', max_depth= 120,
     bootstrap=True)
```

```
[41] model.fit(x_train,y_train)

RandomForestRegressor(max_depth=120, max_features='sqrt', min_samples_leaf=4,
                      min_samples_split=6, n_estimators=1000, n_jobs=-1,
                      oob_score=True)
```

Test the Model



Measure the performance using Metrics