Assignment -3
Python Programming

Assignment Date	30 September 2022
Student Name	POOJA LAKSHMI.S
Student Roll Number	311419205026
Maximum Marks	2 Marks

import libraries

InÂ [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

InÂ [2]:
```
df = pd.read_csv('../input/abalone.csv')
```

InÂ [3]:
```
df.head()
```

Out[3]:

	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

InÂ [4]:
```
df.describe()
```

Out[4]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

In [5]:

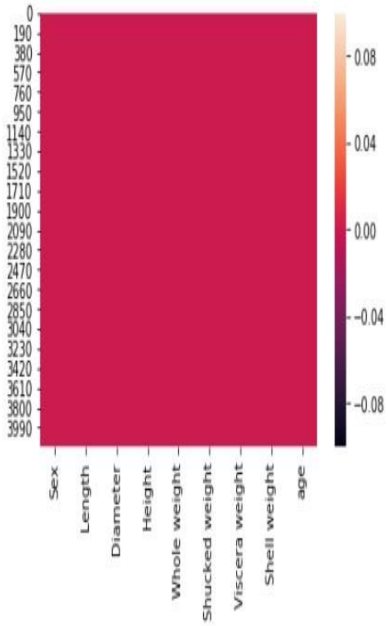
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

EDA

In [6]:

```
sns.heatmap(df.isnull())
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcc468da358>

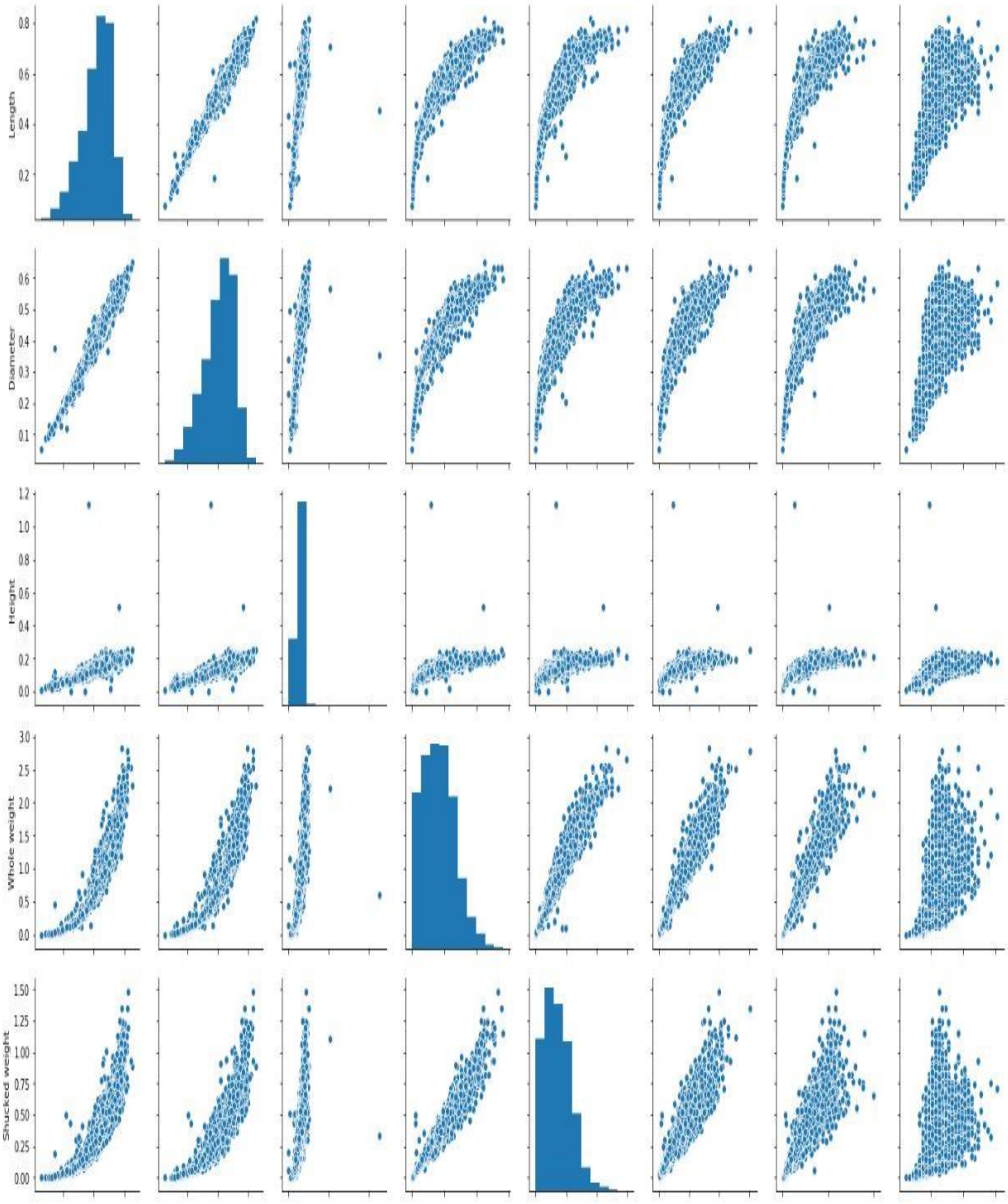


InÂ [7]:

```
sns.pairplot(df)
```

Out[7]:

```
<seaborn.axisgrid.PairGrid at 0x7fcc3caa8160>
```



InÂ [9]:

```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

InÂ [10…

```
numerical_features
```

Out[10]:

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

InÂ [11…

```
categorical_features
```

Out[11]:

```
Index(['Sex'], dtype='object')
```

InÂ [12…

```
plt.figure(figsize = (20,7))
sns.heatmap(df[numerical_features].corr(),annot = True)
```

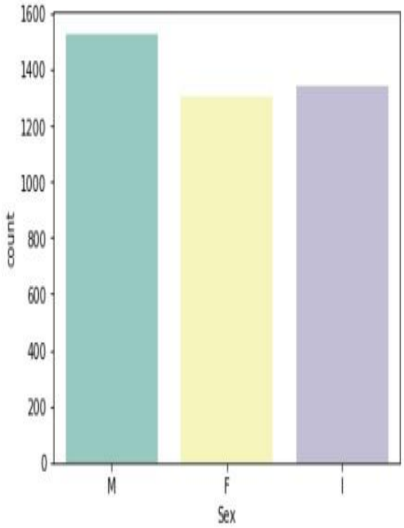
Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcc29714dd8>
```

InÂ [13…

```
sns.countplot(x = 'Sex', data = df, palette = 'Set3')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcc26ba6748>



Male : age majority lies in between 7.5 years to 19 years

Female: age majority lies in between 8 years to 19 years

Immature: age majority lies in between 6 years to < 10 years

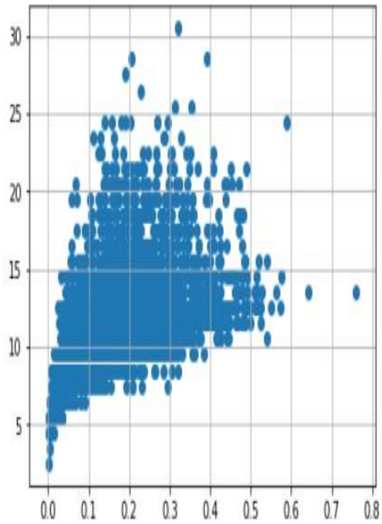
Data Preprocessing

InÂ [15…

```
# outlier handling
df = pd.get_dummies(df)
dummy_df = df
```

InÂ [16…

```
var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```

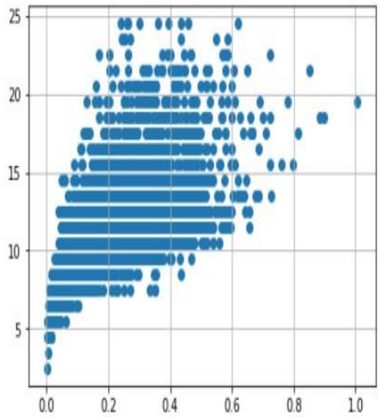


InÂ [17…

```
df.drop(df[(df['Viscera weight'] > 0.5) &
        (df['age'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<0.5) & (
df['age'] > 25)].index, inplace = True)
```

InÂ [18…

```
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```

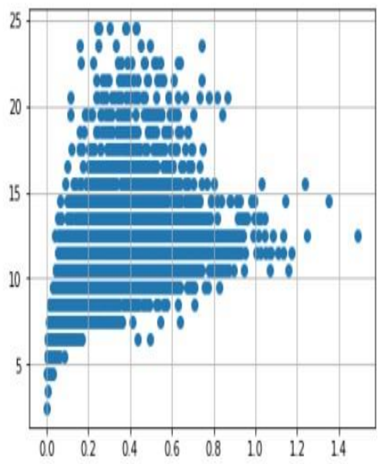


InÂ [19…

```
df.drop(df[(df['Shell weight'] > 0.6) &
          (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Shell weight']<0.8) & (
df['age'] > 25)].index, inplace = True)
```

InÂ [20…

```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```

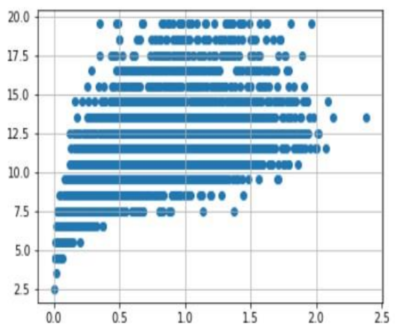


InÂ [21…

```
df.drop(df[(df['Shucked weight'] >= 1) &
          (df['age'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<1) & (
df['age'] > 20)].index, inplace = True)
```

InÂ [22…

```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```

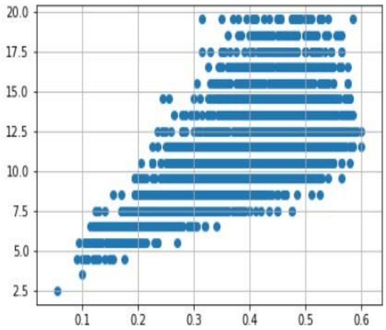


InÂ [23…

```
df.drop(df[(df['Whole weight'] >= 2.5) &
          (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (
df['age'] > 25)].index, inplace = True)
```

InÂ [24…

```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```

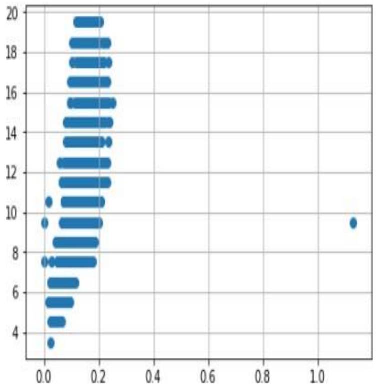


InÂ [25…

```
df.drop(df[(df['Diameter'] <0.1) &
          (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Diameter']>=0.6) & (
df['age'] < 25)].index, inplace = True)
```

InÂ [26…

```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```

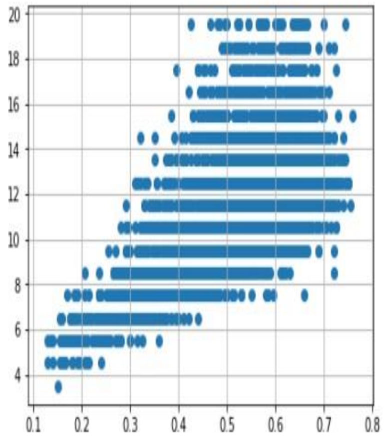


InÂ [27…

```
df.drop(df[(df['Height'] > 0.4) &
         (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['age'] > 25)].index, inplace = True)
```

InÂ [28…

```
var = 'Length'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



Model Selection

1)Linear regression

InÂ [33…

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

InÂ [34…

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

Out[34]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

InÂ [35…

```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

InÂ [36…

```
 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 :3.551893
Mean Squared error of testing set :3.577687
```

InÂ [37…

```
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:0.54
R2 Score of testing set:0.53
```

2)Ridge

InÂ [38…

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

InÂ [39…

```
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)
```

Out[39]:

```
0.5307346478347332
```

InÂ [40…

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

Out[40]:

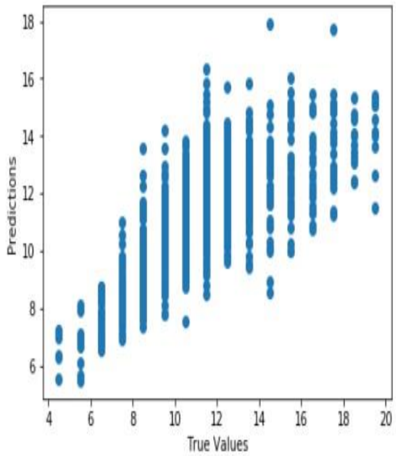
```
0.5272608729607438
```

InÂ [41…

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

Out[41]:

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



3) RandomForestRegression

InÂ [46…

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

InÂ [47…

```
regr = RandomForestRegressor(max_depth=2, random_state=0,
                             n_estimators=100)
```

InÂ [48…

```
regr.fit(X_train, y_train)
regr.fit(X_test, y_test)
```

Out[48]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

InÂ [49…

```
y_train_pred = regr.predict(X_train)
y_test_pred = regr.predict(X_test)

regr.score(X_train, y_train)
```

Out[49]:

```
0.4287379777803546
```

InÂ [50…

```
regr.score(X_test, y_test)
```

Out[50]:

```
0.43753106247261264
```