BUILDING A REGRESSION MODEL

In [ ]:
```
# Import libraries
import numpy as np
import pandas as pd

## Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt
```

DOWNLOAD THE DATASET

In [7]:
```
#Import the dataset
import os
os.chdir("C:/Users/BE HAPPY/Downloads")
```

In [8]:
```
df=pd.read_csv('abalone.csv')
```

In [9]:
```
df
```

Out[9]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
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

4177 rows × 9 columns

LOAD THE DATASET INTO THE TOOL

In [92]:
```
#add target(age) to dataset [rings+1.5=age]
data=pd.read_csv('abalone.csv')
data['age']=data.Rings+1.5

#remove rings variable
data.drop('Rings',axis=1,inplace=True)
print("Data loaded successfully!")
```

```
Data loaded successfully!
```

In [10]:
```
data.sample(5)
```

Out[10]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
699	F	0.395	0.315	0.105	0.3515	0.1185	0.0910	0.1195	17.5
2468	F	0.370	0.275	0.080	0.2270	0.0930	0.0625	0.0700	9.5
2871	I	0.375	0.280	0.085	0.2145	0.0855	0.0485	0.0720	8.5
747	M	0.520	0.385	0.140	0.6595	0.2485	0.2035	0.1600	10.5
927	I	0.435	0.325	0.120	0.3995	0.1815	0.0610	0.1125	9.5

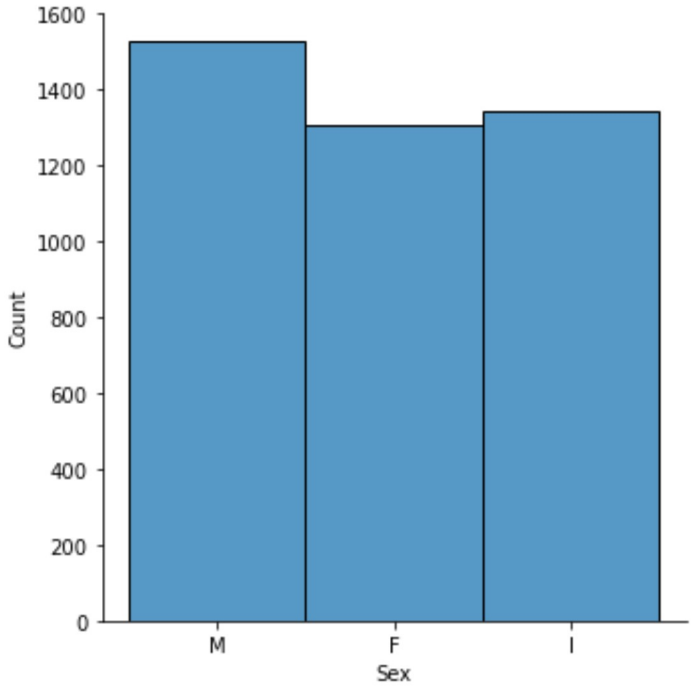
PERFORM BELOW VISUALIZATIONS

Univariate Analysis

In [101…
```
import seaborn as sns
from scipy import stats
```

In [94]:
```
sns.displot(df['Sex'])
```

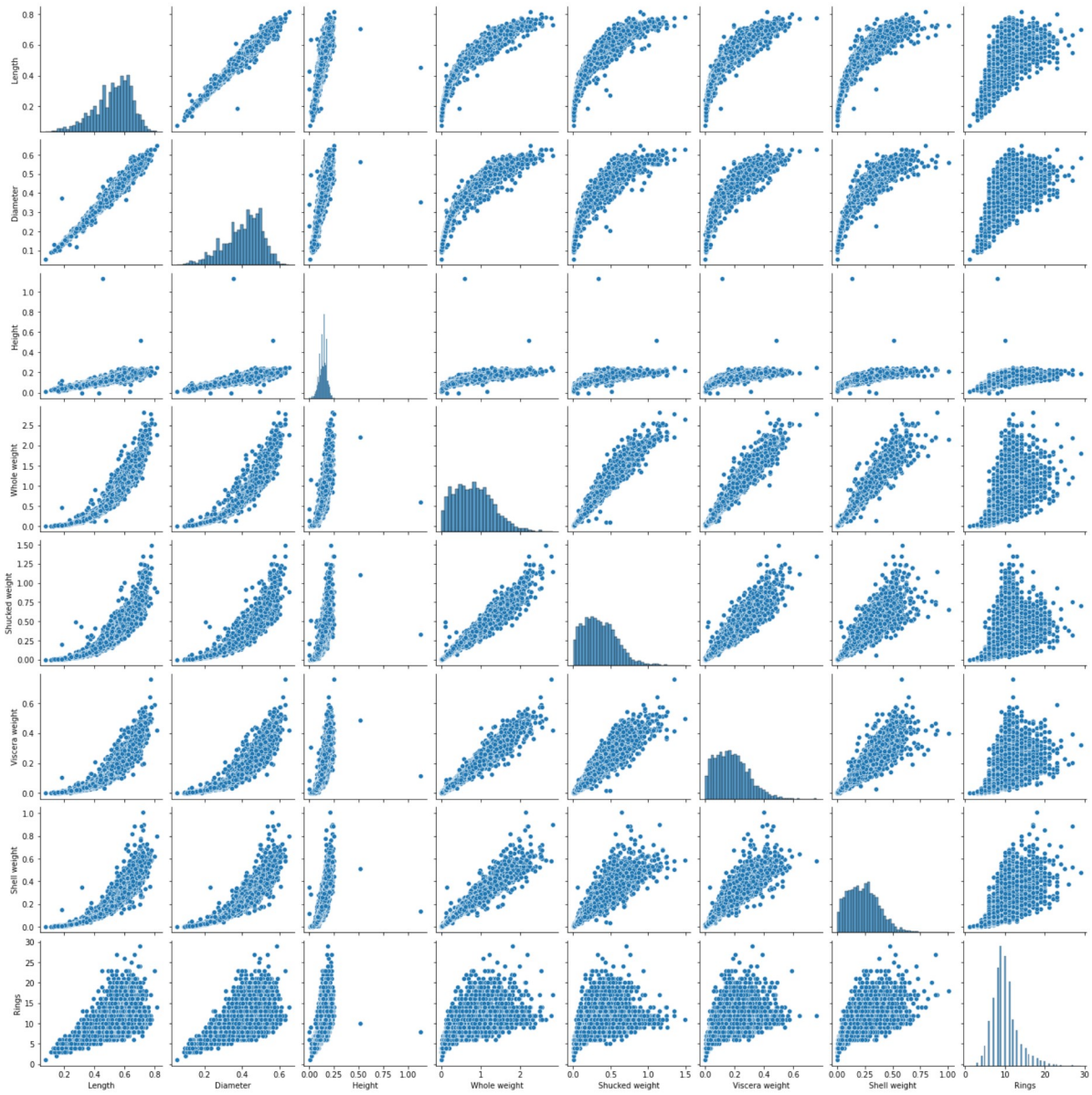
Out[94]:
```
<seaborn.axisgrid.FacetGrid at 0x1e71032caf0>
```



Bi-variate Analysis

In [96]:
```
sns.pairplot(df)
```

Out[96]:
```
<seaborn.axisgrid.PairGrid at 0x1e713ffe190>
```

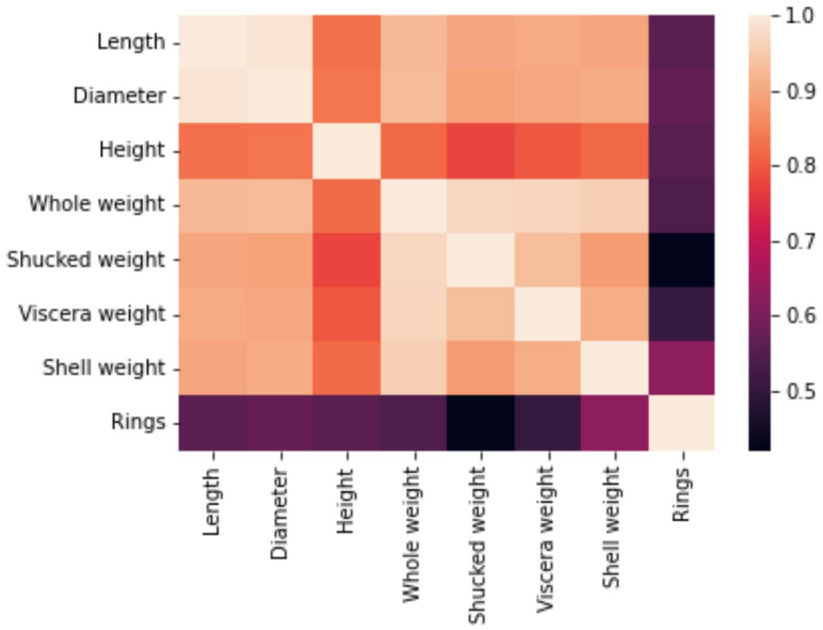


Multi-variate Analysis

In [98]:
```
challenger=pd.read_csv("abalone.csv")
```

In [99]:
```
corr = challenger.corr()
sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns)
```

Out[99]:
```
<AxesSubplot:>
```



PERFORM DESCRIPTIVE STATISTICS ON THE DATASET

In [102…
```
#Discriptive statistics on the data
data.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   age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

In [103…
```
#Statistical summary of features
data.describe()
```

Out[103]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
count	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
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000

CHECK FOR MISSING VALUES AND DEAL WITH THEM

In [11]:
```
#Check for Missing values in the dataset
df.isnull().sum()
```

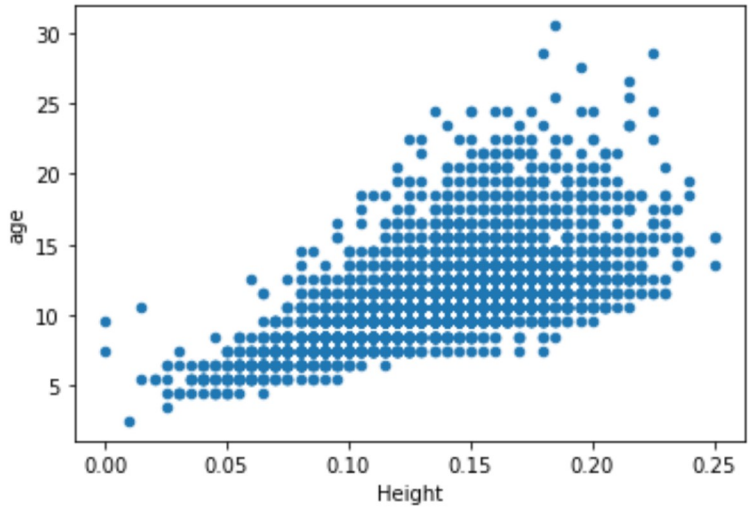
Out[11]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

FIND THE OUTLIERS AND REPLACE THEM OUTLIERS

In [104…
```
#Removing the outliers value from our data
dataset=data.drop(data[(data['Height']>0.4)&(data['Height']<1.4)].index)

#Visualising again to know those outliers removed or not
data_plot=pd.concat([dataset['Height'],dataset['age']],axis=1)
data_plot.plot.scatter(x='Height',y='age')
```

Out[104]:
```
<AxesSubplot:xlabel='Height', ylabel='age'>
```



In [105…
```
#outliers handling
df  = pd.get_dummies(df)
dummy_df = df
```

In [106…
```
from collections import Counter
def detection(df,features):
    outliers_indicies=[]

    for c in features:
        #1st quartile
        Q1 = np.percentile(df[c],25)

        #3rd quartile
        Q3 = np.percentile(df[c],75)

        #IQR calculation
        IQR = Q3 - Q1
        outlier_step = IQR * 1.5
        lower_range = Q1 - (outlier_step)
        upper_range = Q3 + (outlier_step)

        #outlier delection                              #outlier indexes
        outlier_list_col=df[  (df[c] < lower_range) | (df[c] > upper_range)  ].inde

        #store indexes
        outlier_indices.extend(outlier_list_col)

    outlier_indices=Counter(outlier_indices)
    #number of outliers
    #If we have more than 2 outliers in a sample,this sample LL be drop
    multiple_outliers = list(i for i,v in outlier_indices.item()if v>2)
    #We are taking indexes

    return multiple_outliers
```

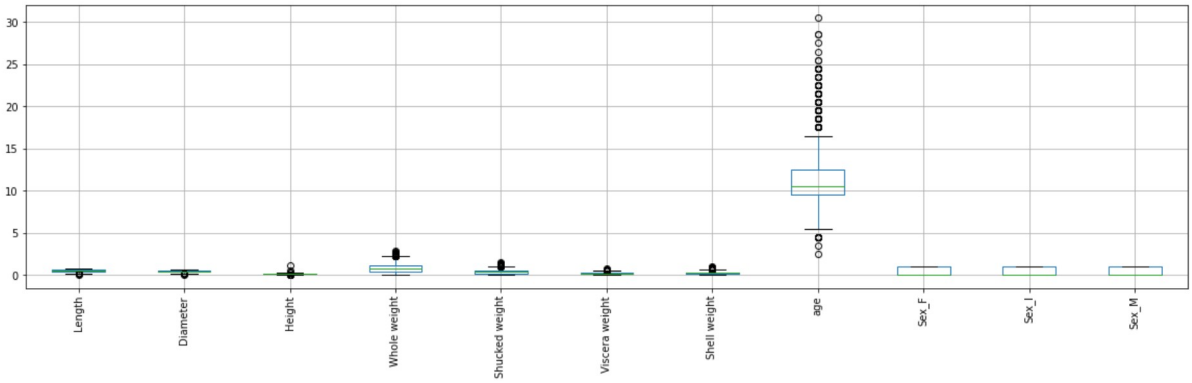
In [107…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Length          4177 non-null   float64
 1   Diameter        4177 non-null   float64
 2   Height          4177 non-null   float64
 3   Whole weight    4177 non-null   float64
 4   Shucked weight  4177 non-null   float64
 5   Viscera weight  4177 non-null   float64
 6   Shell weight    4177 non-null   float64
 7   Rings           4177 non-null   int64  
 8   Sex_F           4177 non-null   uint8  
 9   Sex_I           4177 non-null   uint8  
 10  Sex_M           4177 non-null   uint8  
dtypes: float64(7), int64(1), uint8(3)
memory usage: 273.4 KB
```

In [108…
```
data = pd.get_dummies(data)
dummy_data = data.copy()
```

In [109…
```
data.boxplot(rot = 90,figsize= (20,5))
```

Out[109]:
```
<AxesSubplot:>
```



CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING

In [12]:
```
#Pre-process our categorical data from words to number to make it easier for the co
#understand.
```

In [13]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [14]:
```
encoder = OneHotEncoder(sparse=False)
cat_cols = ['Sex']
```

In [15]:
```
from sklearn.preprocessing import StandardScaler

# Copying original dataframe
df_ready = df.copy()
```

In [16]:
```
# Encode Categorical Data
df_encoded = pd.DataFrame(encoder.fit_transform(df_ready[cat_cols]))
df_encoded.columns = encoder.get_feature_names(cat_cols)
```

```
C:\Users\BE HAPPY\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Futu
reWarning: Function get_feature_names is deprecated; get_feature_names is deprecate
d in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [17]:
```
df_encoded
```

Out[17]:

	Sex_F	Sex_I	Sex_M
0	0.0	0.0	1.0
1	0.0	0.0	1.0
2	1.0	0.0	0.0
3	0.0	0.0	1.0
4	0.0	1.0	0.0
...	...	...	...
4172	1.0	0.0	0.0
4173	0.0	0.0	1.0
4174	0.0	0.0	1.0
4175	1.0	0.0	0.0
4176	0.0	0.0	1.0

4177 rows × 3 columns

In [56]:
```
abalone = pd.get_dummies(challenger)
```

In [57]:
```
abalone
```

Out[57]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11	1	0	0
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10	0	0	1
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9	0	0	1
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10	1	0	0
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12	0	0	1

4177 rows × 11 columns

Split the data into depandent and independent variables

In [110…
```
# x=independent variables and y=depandent
```

In [111…
```
x=df.iloc[:,:1]
```

In [112…
```
x
```

Out[112]:

	Length
0	0.455
1	0.350
2	0.530
3	0.440
4	0.330
...	...
4172	0.565
4173	0.590
4174	0.600
4175	0.625
4176	0.710

4177 rows × 1 columns

In [113...

```
df=pd.read_csv('abalone.csv')
df
y=df.iloc[:,1:]
```

In [114...

```
y
```

Out[114]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 8 columns

SCALE THE INDEPENDENT VARIABLES

In [115…
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
num_cols = ['Length','Diameter','Height','Whole weight','Shucked weight','Viscera w
df_ready[num_cols] = scaler.fit_transform(df[num_cols])
#df_ready.head()
print(df_ready[num_cols])
```

```
        Length  Diameter    Height  Whole weight  Shucked weight  \
0    -0.574558 -0.432149 -1.064424     -0.641898       -0.607685
1    -1.448986 -1.439929 -1.183978     -1.230277       -1.170910
2     0.050033  0.122130 -0.107991     -0.309469       -0.463500
3    -0.699476 -0.432149 -0.347099     -0.637819       -0.648238
4    -1.615544 -1.540707 -1.423087     -1.272086       -1.215968
...        ...       ...       ...           ...             ...
4172  0.341509  0.424464  0.609334      0.118813        0.047908
4173  0.549706  0.323686 -0.107991      0.279929        0.358808
4174  0.632985  0.676409  1.565767      0.708212        0.748559
4175  0.841182  0.777187  0.250672      0.541998        0.773341
4176  1.549052  1.482634  1.326659      2.283681        2.640993

      Viscera weight  Shell weight     Rings
0          -0.726212     -0.638217  1.571544
1          -1.205221     -1.212987 -0.910013
2          -0.356690     -0.207139 -0.289624
3          -0.607600     -0.602294  0.020571
4          -1.287337     -1.320757 -0.910013
...              ...           ...       ...
4172        0.532900      0.073062  0.330765
4173        0.309362      0.155685  0.020571
4174        0.975413      0.496955 -0.289624
4175        0.733627      0.410739  0.020571
4176        1.787449      1.840481  0.640960

[4177 rows x 8 columns]
```

SPLIT THE DATA INTO TRAINING AND TESTING

In [20]:
```
# Select Features
feature = df_ready.drop('Sex',axis=1)
```

In [21]:
```
# Select Target
target = df_ready['Sex']
```

In [22]:
```
# Set Training and Testing Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(feature , target,
                                                    shuffle = True,
                                                    test_size=0.2,
                                                    random_state=1)
```

In [23]:
```
# Show the Training and Testing Data
print('Shape of training feature:', X_train.shape)
print('Shape of testing feature:', X_test.shape)
print('Shape of training label:', y_train.shape)
print('Shape of training label:', y_test.shape)
```

```
Shape of training feature: (3341, 8)
Shape of testing feature: (836, 8)
Shape of training label: (3341,)
Shape of training label: (836,)
```

BUILD THE MODEL

In [116…
```
from sklearn.linear_model import LinearRegression
```

In [118…
```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

Out[118]:
```
LinearRegression()
```

In [ ]:
```
y_train_pred=lm.predict(X_train)
y_test_pred=lm.predict(X_test)
```

TRAIN THE MODEL

In [24]:
```
X_train
```

Out[24]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
666	0.455	0.350	0.120	0.4835	0.1815	0.1440	0.1600	11
2813	0.255	0.195	0.055	0.0725	0.0285	0.0170	0.0210	4
1862	0.520	0.410	0.110	0.5185	0.2165	0.0915	0.1840	8
3684	0.620	0.470	0.155	0.9660	0.4470	0.1710	0.2840	11
551	0.615	0.490	0.155	0.9885	0.4145	0.1950	0.3450	13
…	…	…	…	…	…	…	…	…
2895	0.540	0.415	0.110	0.6190	0.2755	0.1500	0.1765	10
2763	0.550	0.425	0.135	0.6560	0.2570	0.1700	0.2030	10
905	0.320	0.240	0.090	0.1575	0.0700	0.0265	0.0425	5
3980	0.525	0.410	0.115	0.7745	0.4160	0.1630	0.1800	7
235	0.295	0.225	0.080	0.1240	0.0485	0.0320	0.0400	9

3341 rows × 8 columns

In [25]:
```
y_train
```

Out[25]:
```
666     M
2813    I
1862    I
3684    I
551     I
       ..
2895    I
2763    I
905     I
3980    F
235     I
Name: Sex, Length: 3341, dtype: object
```

In [26]:
```
X_train = X_train.values.reshape((-1,1))
```

In [27]:
```
X_train
```

Out[27]:
```
array([[0.455],
       [0.35 ],
       [0.12 ],
       ...,
       [0.032],
       [0.04 ],
       [9.   ]])
```

TEST THE MODEL

In [28]:
```
X_test
```

Out[28]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
17	0.440	0.340	0.100	0.4510	0.1880	0.0870	0.1300	10
1131	0.565	0.435	0.150	0.9900	0.5795	0.1825	0.2060	8
299	0.370	0.280	0.105	0.2340	0.0905	0.0585	0.0750	9
1338	0.580	0.455	0.135	0.7955	0.4050	0.1670	0.2040	10
2383	0.525	0.390	0.135	0.6005	0.2265	0.1310	0.2100	16
...	...	...	...	...	...	...	...	...
1787	0.545	0.420	0.165	0.8935	0.4235	0.2195	0.2280	8
3075	0.680	0.520	0.185	1.4940	0.6150	0.3935	0.4060	11
2766	0.555	0.445	0.175	1.1465	0.5510	0.2440	0.2785	8
1410	0.665	0.530	0.180	1.4910	0.6345	0.3420	0.4350	10
2529	0.600	0.500	0.155	1.3320	0.6235	0.2835	0.3500	8

836 rows × 8 columns

In [29]:
```
y_test
```

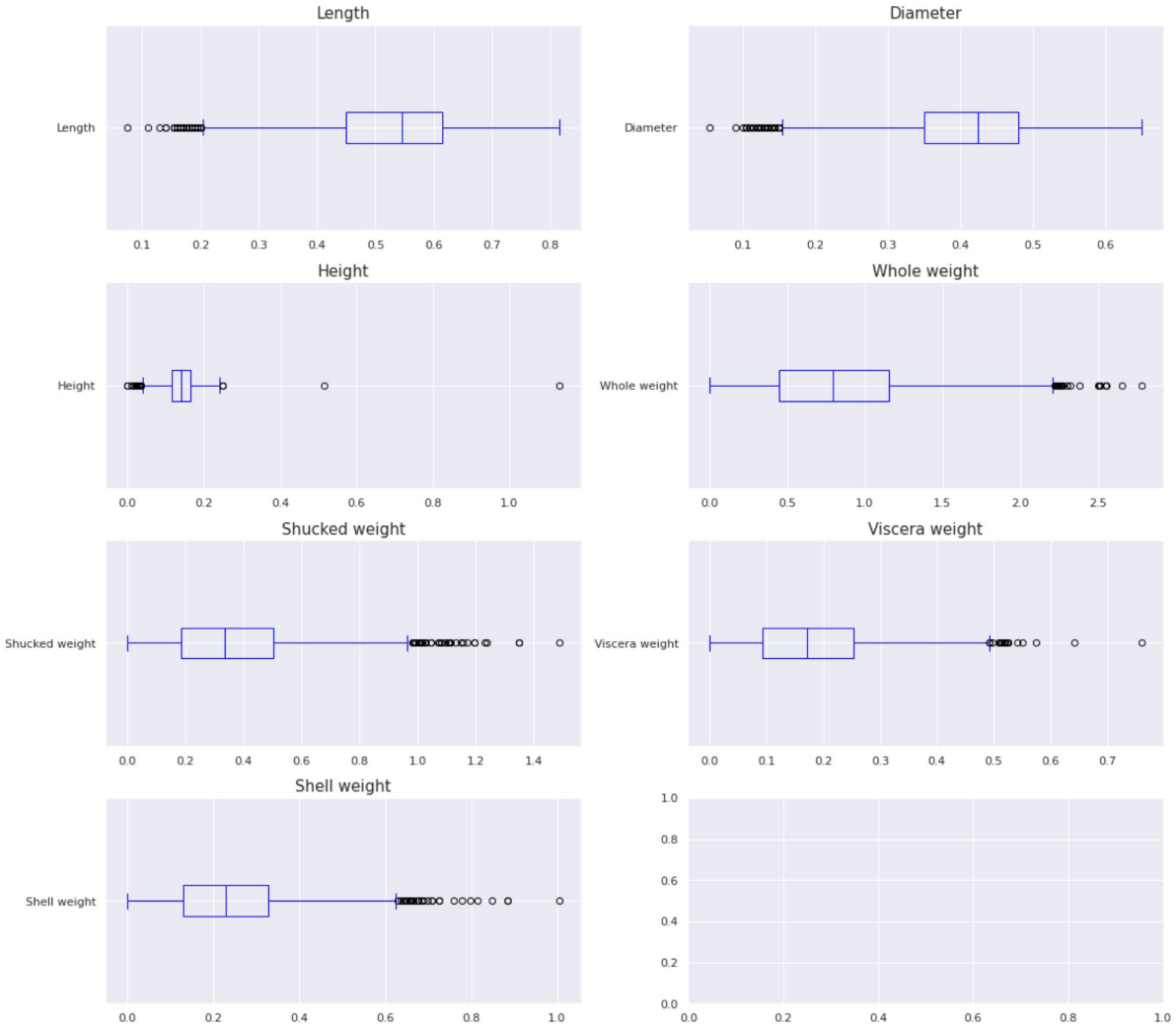
Out[29]:
```
17      F
1131    M
299     M
1338    M
2383    F
       ..
1787    I
3075    F
2766    F
1410    F
2529    F
Name: Sex, Length: 836, dtype: object
```

BOX PLOT(OUTLIERS)

In [12]:
```
fig, axes = plt.subplots(4,2,figsize=(16, 14))
axes = np.ravel(axes)

for i, c in enumerate(numerical_features):
    hist = train[c].plot(kind = 'box', ax=axes[i],color='blue', vert=False)
    axes[i].set_title(c, fontsize=15)

plt.tight_layout()
plt.show()
```



MEASURE THE PERFORMANCE USING METRICS

In [20]:
```
models = {'linear_regression':LinearRegression(),

          'lasso':Lasso(random_state=1),

          'decision_tree':DecisionTreeRegressor(random_state=1),

          'random_forest':RandomForestRegressor(random_state=1),

          'xgboost':XGBRegressor(random_state=1),
         }
```

In [21]:
```
#
for key, regressor in models.items():
    print(key)
    eval_model(regressor, X_train, y_train, X_test, y_test)
    print("\n----------------------------------------")
```

```
linear_regression
Train rmse :  2.1601637766834694
Test rmse :  2.1993326495103673

----------------------------------------
lasso
Train rmse :  3.1425445775484584
Test rmse :  3.2071190143034873

----------------------------------------
decision_tree
Train rmse :  0.0
Test rmse :  2.8672378052018894

----------------------------------------
random_forest
Train rmse :  0.7983734867135102
Test rmse :  2.1456051220373515

----------------------------------------
xgboost
Train rmse :  0.7048710489942694
Test rmse :  2.2560493392094654

----------------------------------------
```