In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn.preprocessing import scale
import warnings
warnings.filterwarnings('ignore')
```

Load the Dataset into the tool

In [11]:
```python
df=pd.read_csv("abalone.csv")
df.head()
```

Out[11]:

	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

In [13]:
```python
df.shape
```

Out[13]:
```
(4177, 9)
```

In [15]:
```python
Age=1.5+df.Rings
df["Age"]=Age
df=df.rename(columns = {'Whole weight':'Whole_weight','Shucked weight': 'Shucked_weight'
                        'Shell weight': 'Shell_weight'})
df=df.drop(columns=["Rings"],axis=1)
df.head()
```

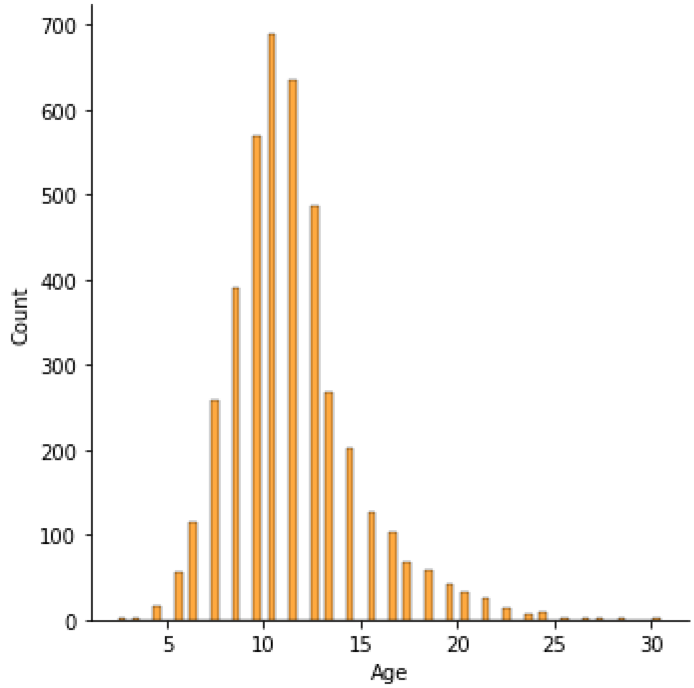
Out[15]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

Univariate Analysis

In [16]:
```python
sns.displot(df["Age"], color='darkorange')
```

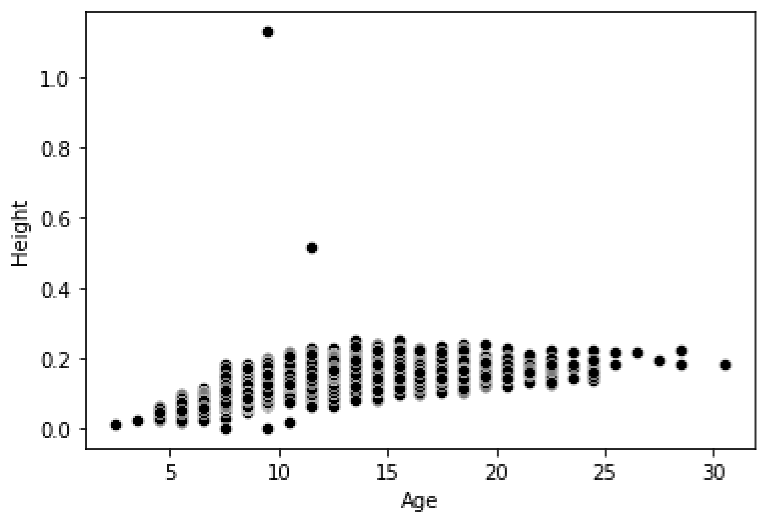
Out[16]:
```
<seaborn.axisgrid.FacetGrid at 0x2629a2418d0>
```



Bi - Variate Analysis

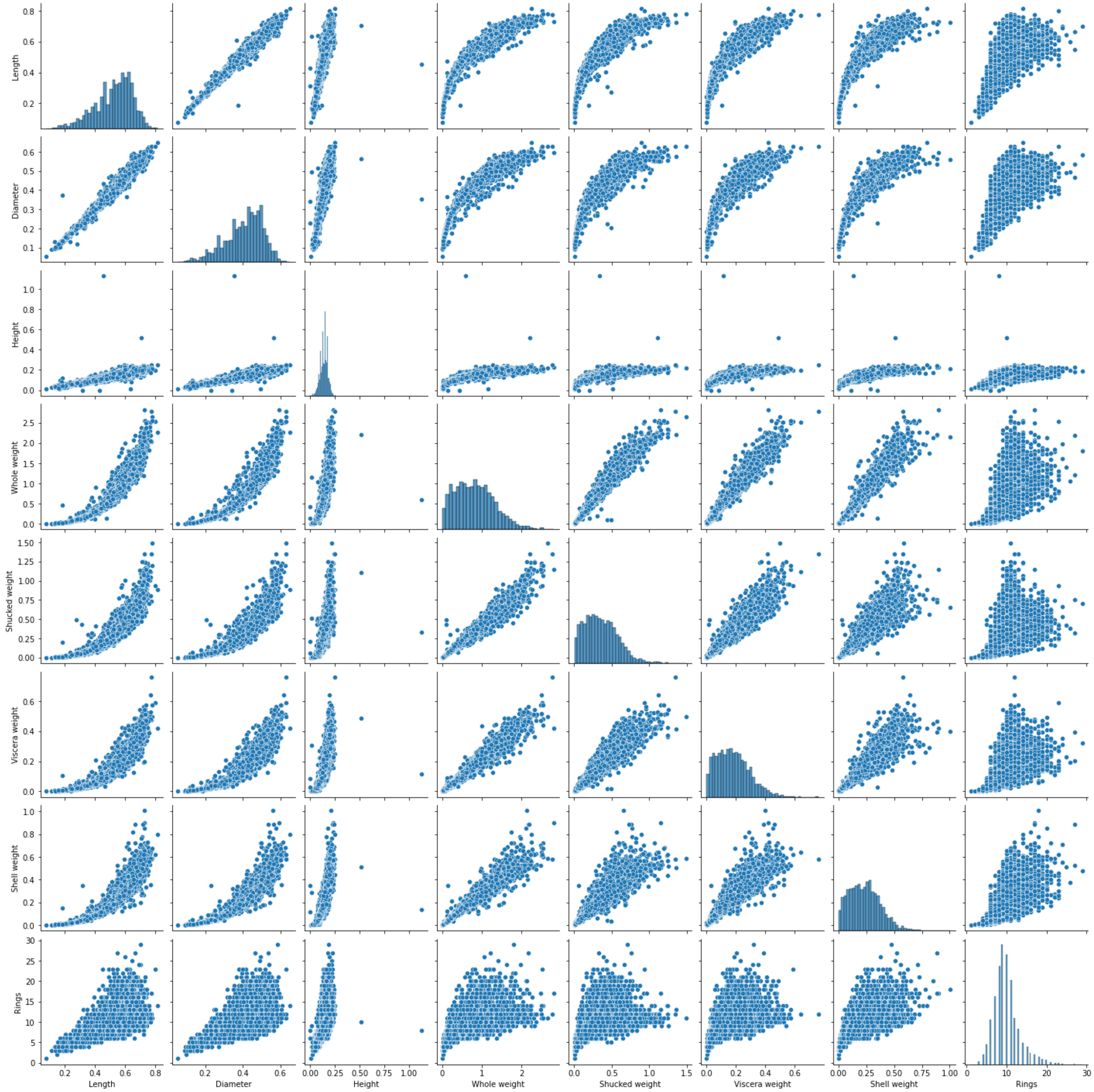
In [17]:
```
sns.scatterplot(x=df.Age,y=df.Height,color='black')
```

Out[17]:
```
<AxesSubplot: xlabel='Age', ylabel='Height'>
```



In [8]:
```
sns.pairplot(df)
```

Out[8]:
```
<seaborn.axisgrid.PairGrid at 0x2629577ce20>
```



Perform descriptive statistics on the dataset

In [18]:
```
df.describe()
```

Out[18]:

	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	4
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 [19]:

```
df.isna().sum()
```

Out[19]:

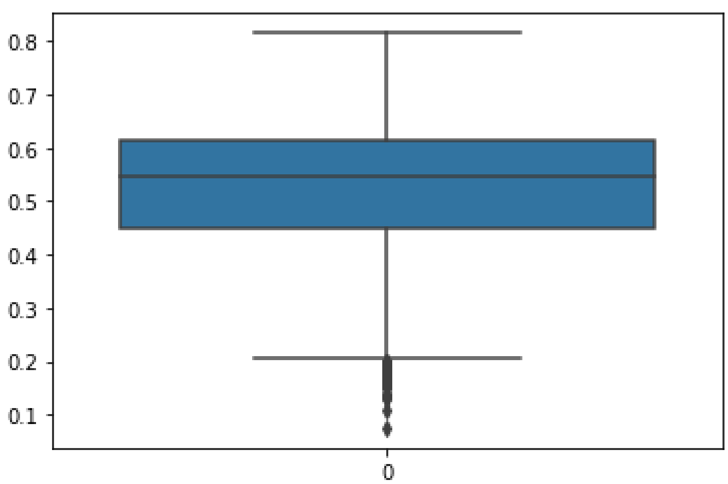
```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
Age               0
dtype: int64
```

Find the outliers and replace them outliers.

In [20]:

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

Out[20]: <AxesSubplot: >



Check for Categorical columns and perform encoding.

In [21]:

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

```
df.tail()
```

Out[21]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	12.5
4173	1	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	11.5
4174	1	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	10.5
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	11.5
4176	1	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	13.5

Split the data into dependent and independent variables.

In [24]:

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

Out[24]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0.25	0.0	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	9.5
0.75	2.0	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	12.5

In [25]:

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

Out[25]:

```
Sex                1.0000
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
```

In [27]:

```
from sklearn.preprocessing import LabelEncoder

lab = LabelEncoder()
df.Sex = lab.fit_transform(df.Sex)

df.head()
```

Out[27]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

Scale the independent variables

In [28]:
```
x=df.drop(columns=['Length','Height'])
print(x)
```

```
      Sex  Diameter  Whole_weight  Shucked_weight  Viscera_weight  \
0       1     0.365        0.5140          0.2245          0.1010   
1       1     0.265        0.2255          0.0995          0.0485   
2       0     0.420        0.6770          0.2565          0.1415   
3       1     0.365        0.5160          0.2155          0.1140   
4       2     0.255        0.2050          0.0895          0.0395   
...   ...       ...           ...             ...             ...   
4172    0     0.450        0.8870          0.3700          0.2390   
4173    1     0.440        0.9660          0.4390          0.2145   
4174    1     0.475        1.1760          0.5255          0.2875   
4175    0     0.485        1.0945          0.5310          0.2610   
4176    1     0.555        1.9485          0.9455          0.3765   

      Shell_weight   Age  
0           0.1500  16.5  
1           0.0700   8.5  
2           0.2100  10.5  
3           0.1550  11.5  
4           0.0550   8.5  
...            ...   ...  
4172        0.2490  12.5  
4173        0.2605  11.5  
4174        0.3080  10.5  
4175        0.2960  11.5  
4176        0.4950  13.5  

[4177 rows x 7 columns]
```

In [29]:
```
x=scale(x)
print(x)
```

```
[[-0.0105225  -0.43214879 -0.64189823 ... -0.72621157 -0.63821689
   1.57154357]
 [-0.0105225  -1.439929   -1.23027711 ... -1.20522124 -1.21298732
  -0.91001299]
 [-1.26630752  0.12213032 -0.30946926 ... -0.35668983 -0.20713907
  -0.28962385]
 ...
 [-0.0105225   0.67640943  0.70821206 ...  0.97541324  0.49695471
  -0.28962385]
 [-1.26630752  0.77718745  0.54199757 ...  0.73362741  0.41073914
   0.02057072]
 [-0.0105225   1.48263359  2.28368063 ...  1.78744868  1.84048058
   0.64095986]]
```

Split the data into training and testing.

In [42]:
```
y = df["Sex"]
y.head()
```

Out[42]:
```
0    1
1    1
2    0
3    1
4    2
Name: Sex, dtype: int64
```

In [36]:
```python
x=df.drop(columns=["Sex"],axis=1)
x.head()
```

Out[36]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

In [43]:
```python
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=0)
x_train.shape
```

Out[43]:
```
(3341, 8)
```

In [44]:
```python
x_test.shape
```

Out[44]:
```
(836, 8)
```

In [45]:
```python
y_test.shape
```

Out[45]:
```
(836,)
```

Build the Model.

In [46]:
```python
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier()
df = pd.get_dummies(df, drop_first=True)
df.head()
```

Out[46]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

In [47]:
```python
X = df.drop('Height', axis=1)
y = df['Height']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)

from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

X_trains = ss.fit_transform(X_train)
X_tests = ss.transform(X_test)
```

Base model

In [48]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(X_trains, y_train)
pred = lr.predict(X_tests)

from sklearn.metrics import r2_score, roc_auc_score, mean_squared_error
rmse = np.sqrt(mean_squared_error(y_test, pred))
r2 = r2_score(y_test, pred)

print("The root mean Sq error calculated from the base model is:",rmse)
print("The r2-score is:",r2)
```

```
The root mean Sq error calculated from the base model is: 0.017389350580888849
The r2-score is: 0.8102454999039992
```

selecting best feautre

In [49]:
```
from sklearn.feature_selection import RFE
lr = LinearRegression()
n = [{'n_features_to_select':list(range(1,10))}]
rfe = RFE(lr)

from sklearn.model_selection import GridSearchCV
gsearch = GridSearchCV(rfe, param_grid=n, cv=3)
gsearch.fit(X, y)

gsearch.best_params_
```

Out[49]:
```
{'n_features_to_select': 8}
```

In [50]:
```
lr = LinearRegression()
rfe = RFE(lr, n_features_to_select=8)
rfe.fit(X,y)

pd.DataFrame(rfe.ranking_, index=X.columns, columns=['Class'])
```

Out[50]:

	Class
Sex	1
Length	1
Diameter	1
Whole_weight	1
Shucked_weight	1
Viscera_weight	1
Shell_weight	1
Age	1

Measure the performance using Metrics

In [51]:
```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import  RandomForestRegressor
from sklearn.linear_model import LinearRegression
ensemble import  GradientBoostingRegressor
```

```python
from sklearn.linear_model import  Ridge
from sklearn.svm import SVR
from sklearn import model_selection
from sklearn.model_selection import cross_val_predict

models = [   SVR(),
             RandomForestRegressor(),
             GradientBoostingRegressor(),
             KNeighborsRegressor(n_neighbors = 4)]
results = []
names = ['SVM','Random Forest','Gradient Boost','K-Nearest Neighbors']
for model,name in zip(models,names):
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold)
    rmse = np.sqrt(mean_squared_error(y, cross_val_predict(model, X , y, cv=3)))
    results.append(rmse)
    names.append(name)
    msg = "%s: %f" % (name, rmse)
    print(msg)
```

```
SVM: 0.036681
Random Forest: 0.025132
Gradient Boost: 0.023635
K-Nearest Neighbors: 0.023851
```

In [ ]: