ASSIGNMENT - 4

IMPORTING LIBRARIES

In [308…
```
# Import libraries
import pandas as pd
import numpy as np
#data visualization
import seaborn as sns
import matplotlib.pyplot as plt
```

In [309…
```
#Import the dataset
import os
os.chdir("C:/Users/Lenovo/Desktop/Dataset")
```

In [310…
```
#1.Load the dataset into the tool
#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 [311…
```
df=pd.read_csv('abalone.csv')
```

In [312…
```
df
```

Out[312]:

	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

In [313…
```
5.#check for missing values in the dataset and deal with them
df.isnull().sum()
```

Out[313]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

In [314…
```
data.describe()
```

Out[314]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shel weigh
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	0.23883
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.13920
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.00150
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.13000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.23400
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.32900
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.00500

In [316…
```
data['age'].isnull().sum()
```

Out[316]:
```
0
```

In [317…
```
data['age'].mean()
```

Out[317]:
```
11.433684462532918
```

In [318…
```
data['age'].replace(np.NaN , data['age'].mean()).head(15)
```

Out[318]:
```
0     16.5
1      8.5
2     10.5
3     11.5
4      8.5
5      9.5
6     21.5
7     17.5
8     10.5
9     20.5
10    15.5
11    11.5
12    12.5
13    11.5
14    11.5
Name: age, dtype: float64
```

In [319…
```
data['age'].median()
```

Out[319]:
```
10.5
```

In [320…]
```
data['age'].mode()
```

Out[320]:
```
0    10.5
Name: age, dtype: float64
```

In [ ]:
```
# 7.Check for categorical columns and perform encoding
```

In [226…]
```
#preprocess our categorical data from words to number to make it easier for the co
#understand
```

In [227…]
```
from sklearn.preprocessing import OneHotEncoder
```

In [228…]
```
encoder = OneHotEncoder(sparse=False)
cat_cols = ['sex']
```

In [229…]
```
from sklearn.preprocessing import StandardScaler

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

In [230…]
```
scaler = StandardScaler()
num_cols = ['Rings', 'Shell weight', 'Viscera weight', 'Shucked weight', 'Whole we
```

In [231…]
```
df_ready.head()
```

Out[231]:

	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 [232…]
```
#Pre-process our categorical data from words to number to make it easier for the c
#understand.
```

In [233…]
```
from sklearn.preprocessing import OneHotEncoder
```

In [234…]
```
encoder = OneHotEncoder(sparse=False)
cat_cols = ['Sex']
```

In [235…]
```
# 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\Lenovo\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Futur
eWarning: 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 [236…]
```
# Replace Categotical Data with Encoded Data
df_ready = df_ready.drop(cat_cols ,axis=1)
df_ready = pd.concat([df_encoded, df_ready], axis=1)
```

In [237…]
```
# Encode target value
```

```
df_ready['Rings'] = df_ready['Rings'].apply(lambda x: 1 if x == 'yes' else 0)
```

In [238…]
```
print('Shape of dataframe:', df_ready.shape)
```

```
Shape of dataframe: (4177, 11)
```

In [239…]
```
df_ready.head()
```

Out[239]:

	Sex_F	Sex_I	Sex_M	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0.0	0.0	1.0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	0
1	0.0	0.0	1.0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	0
2	1.0	0.0	0.0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	0
3	0.0	0.0	1.0	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	0
4	0.0	1.0	0.0	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	0

In [ ]:
```
# 10.split the data into training and testing
# 12.train the model
# 13.test the model
```

In [240…]
```
#Split Dataset for Training and Testing
# Select Features
feature = df_ready.drop('Rings', axis=1)
```

In [241…]
```
# Select Target
target = df_ready['Rings']
```

In [242…]
```
# 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 [243…]
```
# 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, 10)
Shape of testing feature: (836, 10)
Shape of training label: (3341,)
Shape of training label: (836,)
```

In [244…]
```
X_train
```

Out[244]:

	Sex_F	Sex_I	Sex_M	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
666	0.0	0.0	1.0	0.455	0.350	0.120	0.4835	0.1815	0.1440	0.1600
2813	0.0	1.0	0.0	0.255	0.195	0.055	0.0725	0.0285	0.0170	0.0210
1862	0.0	1.0	0.0	0.520	0.410	0.110	0.5185	0.2165	0.0915	0.1840
3684	0.0	1.0	0.0	0.620	0.470	0.155	0.9660	0.4470	0.1710	0.2840
551	0.0	1.0	0.0	0.615	0.490	0.155	0.9885	0.4145	0.1950	0.3450
...	...	...	...	...	...	...	...	...	...	...
2895	0.0	1.0	0.0	0.540	0.415	0.110	0.6190	0.2755	0.1500	0.1765
2763	0.0	1.0	0.0	0.550	0.425	0.135	0.6560	0.2570	0.1700	0.2030
905	0.0	1.0	0.0	0.320	0.240	0.090	0.1575	0.0700	0.0265	0.0425
3980	1.0	0.0	0.0	0.525	0.410	0.115	0.7745	0.4160	0.1630	0.1800
235	0.0	1.0	0.0	0.295	0.225	0.080	0.1240	0.0485	0.0320	0.0400

3341 rows × 10 columns

In [245…
```
y_train
```

Out[245]:
```
666     0
2813    0
1862    0
3684    0
551     0
       ..
2895    0
2763    0
905     0
3980    0
235     0
Name: Rings, Length: 3341, dtype: int64
```

In [246…
```
X_train.shape
```

Out[246]:
```
(3341, 10)
```

In [247…
```
y_train.shape
```

Out[247]:
```
(3341,)
```

In [248…
```
X_train = X_train.values.reshape((-1,1))
```

In [249…
```
X_train
```

Out[249]:
```
array([[0.    ],
       [0.    ],
       [1.    ],
       ...,
       [0.0485],
       [0.032 ],
       [0.04  ]])
```

In [250…
```
y_train
```

Out[250]:
```
666     0
2813    0
1862    0
3684    0
551     0
       ..
2895    0
2763    0
905     0
3980    0
235     0
Name: Rings, Length: 3341, dtype: int64
```

In [251…
```
X_test
```

Out[251]:

	Sex_F	Sex_I	Sex_M	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
17	1.0	0.0	0.0	0.440	0.340	0.100	0.4510	0.1880	0.0870	0.1300
1131	0.0	0.0	1.0	0.565	0.435	0.150	0.9900	0.5795	0.1825	0.2060
299	0.0	0.0	1.0	0.370	0.280	0.105	0.2340	0.0905	0.0585	0.0750
1338	0.0	0.0	1.0	0.580	0.455	0.135	0.7955	0.4050	0.1670	0.2040
2383	1.0	0.0	0.0	0.525	0.390	0.135	0.6005	0.2265	0.1310	0.2100
…	…	…	…	…	…	…	…	…	…	…
1787	0.0	1.0	0.0	0.545	0.420	0.165	0.8935	0.4235	0.2195	0.2280
3075	1.0	0.0	0.0	0.680	0.520	0.185	1.4940	0.6150	0.3935	0.4060
2766	1.0	0.0	0.0	0.555	0.445	0.175	1.1465	0.5510	0.2440	0.2785
1410	1.0	0.0	0.0	0.665	0.530	0.180	1.4910	0.6345	0.3420	0.4350
2529	1.0	0.0	0.0	0.600	0.500	0.155	1.3320	0.6235	0.2835	0.3500

836 rows × 10 columns

In [252…
```
y_test
```

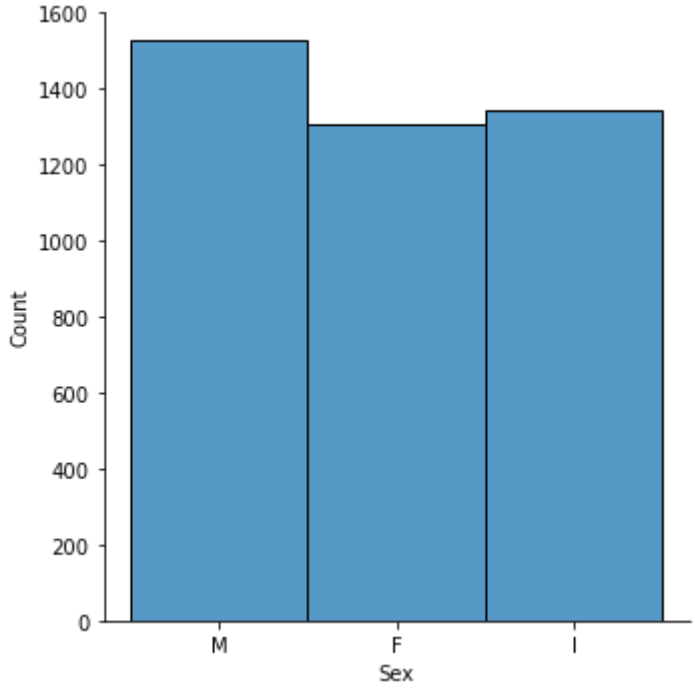
Out[252]:
```
17      0
1131    0
299     0
1338    0
2383    0
       ..
1787    0
3075    0
2766    0
1410    0
2529    0
Name: Rings, Length: 836, dtype: int64
```

In [ ]:
```
# 3.perform Visualization
```

In [ ]:
```
#Univarient analysis
```

In [253…
```
sns.displot(df['Sex'])
```

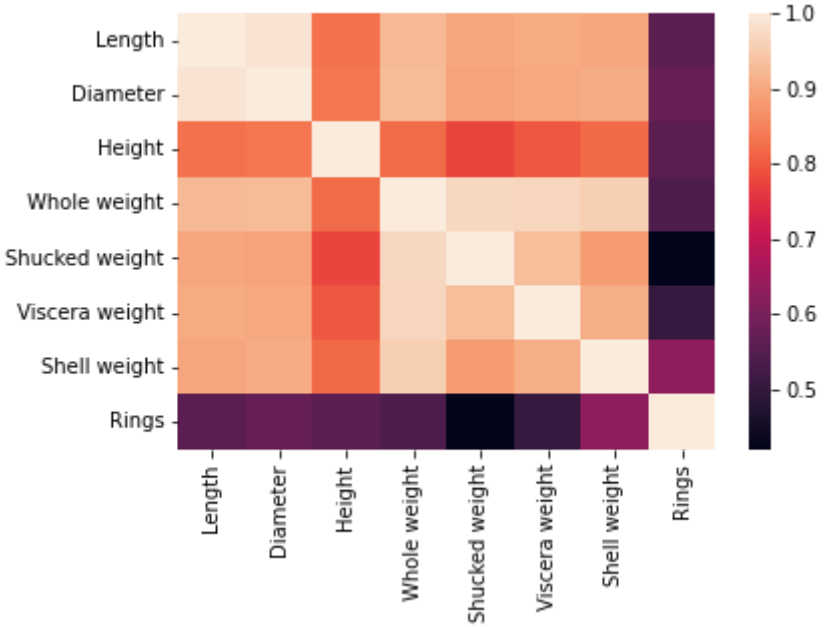
Out[253]:
```
<seaborn.axisgrid.FacetGrid at 0x2024bd218e0>
```



In [254…
```
#Multivariant analysis
```

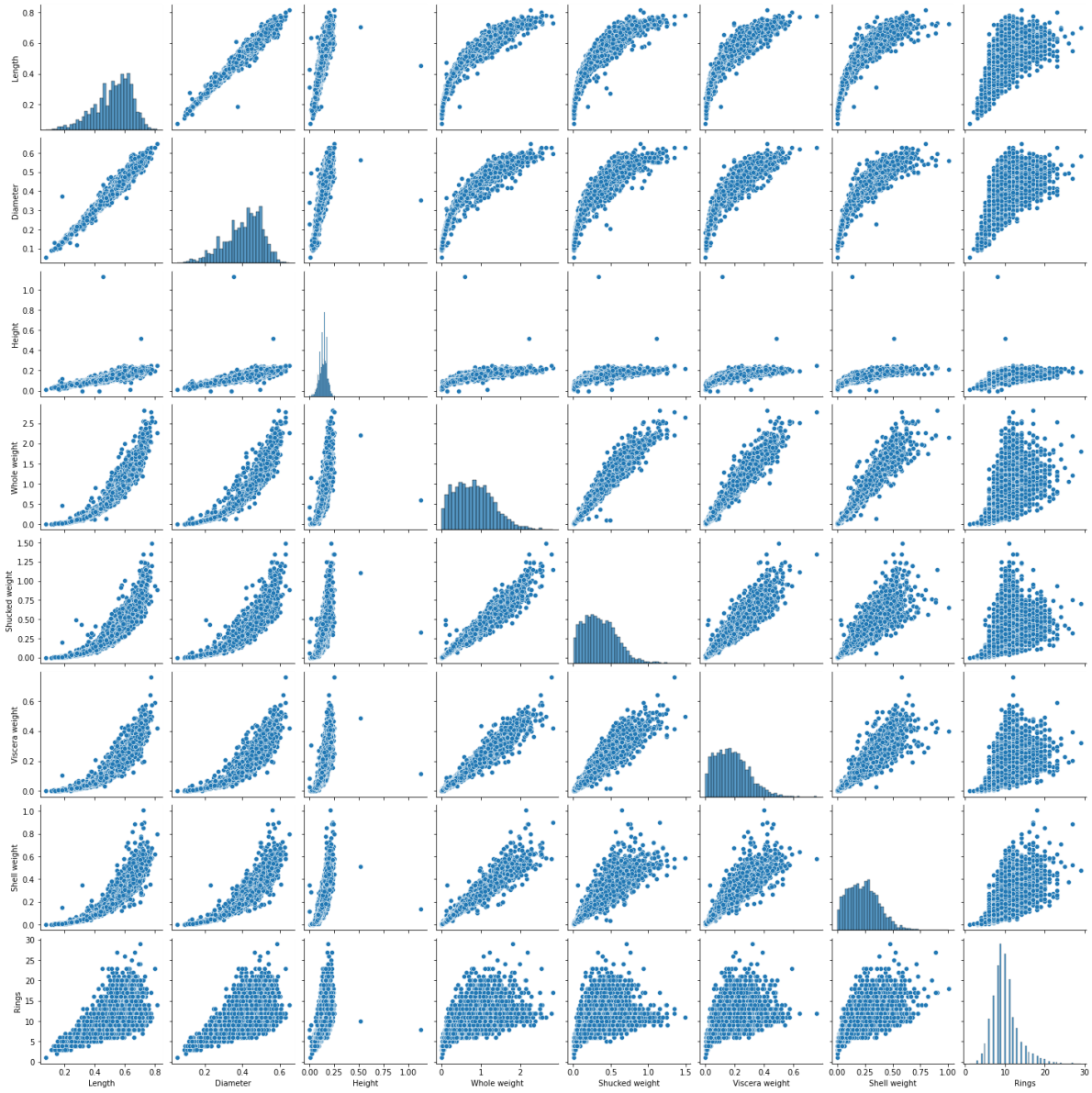
In [255…
```
corr = df.corr()
sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns)
```

Out[255]:
```
<AxesSubplot:>
```



In [256…
```
#Bi-variant analysis
sns.pairplot(df)
```

Out[256]:
```
<seaborn.axisgrid.PairGrid at 0x20255e826d0>
```



In [257…

```
# 4.Discriptive statistics on the dataset
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 [258…

```
data.describe()
```

Out[258]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shel weigh
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.00000(
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.23883
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.13920
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.00150(
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.13000(
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.23400(
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.32900(
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.00500(

In [259…

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

In [260…

```
from collections import Counter
def detection(df,features):
    outlier_indices=[]

    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 detection                                    #Outlier indexes
        outlier_list_col=df[  (df[c] < lower_range) | (df[c] > upper_range)  ].ind

        #Store indexes
        outlier_indices.extend(outlier_list_col)

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

    return multiple_outliers
```

In [261…

```
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 [262…

```
outliers=detection(df,["Length","Whole weight","Height","Diameter"])

df.loc[outliers]
```

Out[262]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_M
236	0.075	0.055	0.010	0.0020	0.0010	0.0005	0.0015	1	0	1	0
237	0.130	0.100	0.030	0.0130	0.0045	0.0030	0.0040	3	0	1	0
238	0.110	0.090	0.030	0.0080	0.0025	0.0020	0.0030	3	0	1	0
239	0.160	0.120	0.035	0.0210	0.0075	0.0045	0.0050	5	0	1	0
306	0.165	0.120	0.030	0.0215	0.0070	0.0050	0.0050	3	0	1	0
694	0.165	0.110	0.020	0.0190	0.0065	0.0025	0.0050	4	0	1	0
718	0.180	0.125	0.035	0.0265	0.0095	0.0055	0.0085	4	0	1	0
719	0.150	0.100	0.025	0.0150	0.0045	0.0040	0.0050	2	0	1	0
720	0.160	0.110	0.025	0.0180	0.0065	0.0055	0.0050	3	0	1	0
1429	0.140	0.105	0.035	0.0140	0.0055	0.0025	0.0040	3	0	1	0
1987	0.160	0.110	0.025	0.0195	0.0075	0.0050	0.0060	4	0	1	0
2114	0.130	0.095	0.035	0.0105	0.0050	0.0065	0.0035	4	0	1	0
2169	0.165	0.115	0.015	0.0145	0.0055	0.0030	0.0050	4	0	1	0
2171	0.190	0.130	0.030	0.0295	0.0155	0.0150	0.0100	6	0	1	0
2381	0.155	0.115	0.025	0.0240	0.0090	0.0050	0.0075	5	0	0	1
2711	0.190	0.140	0.030	0.0315	0.0125	0.0050	0.0105	3	0	1	0
3190	0.200	0.145	0.025	0.0345	0.0110	0.0075	0.0100	5	0	1	0
3837	0.170	0.105	0.035	0.0340	0.0120	0.0085	0.0050	4	0	1	0
3899	0.140	0.105	0.035	0.0145	0.0050	0.0035	0.0050	4	0	1	0
3902	0.160	0.120	0.020	0.0180	0.0075	0.0045	0.0050	4	0	1	0

In [263…

```
df=df.drop(outliers,axis=0).reset_index(drop = True)
```

```
df
```

Out[263]:

	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
...	...	...	...	...	...	...	...	...	...	...	...
4152	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11	1	0	0
4153	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10	0	0	1
4154	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9	0	0	1
4155	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10	1	0	0
4156	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12	0	0	1

4157 rows × 11 columns

In [ ]:
```
# 8.split the data into dependent and independent variables
# 9.scale independent variable
```

In [ ]:
```
# x-independent variable & y-dependent variable
```

In [264…
```
x=df.iloc[:,:1]
```

In [265…
```
x
```

Out[265]:

	Length
0	0.455
1	0.350
2	0.530
3	0.440
4	0.330
...	...
4152	0.565
4153	0.590
4154	0.600
4155	0.625
4156	0.710

4157 rows × 1 columns

In [266…
```
df=pd.read_csv('abalone.csv')
```

```
df
y=df.iloc[:,1:]
```

In [267…]
```
y
```

Out[267]:

	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

In [268…]
```
# 11.Build the model
```

In [269…]
```
transformed_sex_feature = OneHotEncoder().fit_transform(df['Sex'].values.reshape(-1
df_sex_encoded = pd.DataFrame(transformed_sex_feature, columns = ["Sex_"+str(int(i
df = pd.concat([df, df_sex_encoded], axis=1)
```

In [270…]
```
df.head()
```

Out[270]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_0	Sex_1	Sex_
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	0.0	0.0	1
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	0.0	0.0	1
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	1.0	0.0	0
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	0.0	0.0	1
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	0.0	1.0	0

In [271…]
```
# 14. Measure the performance using Metrics
```

In [298…]
```
df['Age'] = df['Rings'] + 1.5
df['Age'].head(5)
```

Out[298]:
```
0    16.5
1     8.5
2    10.5
3    11.5
4     8.5
Name: Age, dtype: float64
```

In [299…
```
#'''Sex and Age Visulization'''
plt.figure(figsize = (20,7))
sns.swarmplot(x = 'Sex', y = 'Age', data = df, hue = 'Sex')
sns.violinplot(x = 'Sex', y = 'Age', data = df)
```

```
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarni
ng: 56.2% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarni
ng: 52.2% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Lenovo\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarni
ng: 58.5% of the points cannot be placed; you may want to decrease the size of the
markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[299]:
```
<AxesSubplot:xlabel='Sex', ylabel='Age'>
```