In [1]:
```
# Import libraries
import pandas as pd
import numpy as np
#data visualization
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [3]:
```
#1Import the dataset
import os
os.chdir("C:/Users")
```

In [4]:
```
#2.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 [5]:
```
df=pd.read_csv('abalone.csv')
```

In [6]:
```
df
```

Out[6]:

	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 [7]:
```
5.#check for missing values in the dataset and deal with them
df.isnull().sum()
```

Out[7]:
```
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 [8]:
```
data.describe()
```

Out[8]:

	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	0.139203
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

In [9]:
```
data['age'].isnull().sum()
```

Out[9]:
```
0
```

In [10]:
```
data['age'].mean()
```

Out[10]:
```
11.433684462532918
```

In [11]:
```
data['age'].replace(np.NaN , data['age'].mean()).head(15)
```

Out[11]:
```
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 [12]:
```
data['age'].median()
```

Out[12]:
```
10.5
```

In [13]:
```
data['age'].mode()
```

Out[13]:
```
0    10.5
Name: age, dtype: float64
```

In [14]:
```
# 7.Check for categorical columns and perform encoding
from sklearn.preprocessing import OneHotEncoder
```

In [15]:
```
encoder = OneHotEncoder(sparse=False)
cat_cols = ['sex']
```

In [16]:
```
from sklearn.preprocessing import StandardScaler
# copying original dataframe
df_ready = df.copy()
```

In [17]:
```
from sklearn.preprocessing import StandardScaler
# copying original dataframe
df_ready = df.copy()
```

In [19]:
```
scaler = StandardScaler()
num_cols = ['Rings', 'Shell weight', 'Viscera weight', 'Shucked weight', 'Whole wei
```

In [20]:
```
df_ready.head()
```

Out[20]:

	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 [21]:
```
from sklearn.preprocessing import OneHotEncoder
```

In [22]:
```
encoder = OneHotEncoder(sparse=False)
cat_cols = ['Sex']
```

In [23]:
```
# Encode Categorical Data
df_encoded = pd.DataFrame(encoder.fit_transform(df_ready[cat_cols]))
df_encoded.columns = encoder.get_feature_names(cat_cols)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Future
Warning: Function get_feature_names is deprecated; get_feature_names is deprecated
in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [24]:
```
# 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 [25]:
```
df_ready['Rings'] = df_ready['Rings'].apply(lambda x: 1 if x == 'yes' else 0)
```

In [26]:
```
print('Shape of dataframe:', df_ready.shape)
```

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

In [27]:
```
df_ready.head()
```

Out[27]:

	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 [28]:
```
# 10.split the data into training and testing
# 12.train the model
# 13.test the model
#Split Dataset for Training and Testing
# Select Features
feature = df_ready.drop('Rings', axis=1)
```

In [29]:
```
# Select Target
target = df_ready['Rings']
```

In [30]:
```
# 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 [31]:
```
# 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 [32]:
```
X_train
```

Out[32]:

	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 [33]:
```
y_train
```

Out[33]:
```
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 [34]:
```
X_train.shape
```

Out[34]:
```
(3341, 10)
```

In [35]:
```
y_train.shape
```

Out[35]:
```
(3341,)
```

In [36]:
```
X_train = X_train.values.reshape((-1,1))
```

In [37]:
```
X_train
```

Out[37]:
```
array([[0.    ],
       [0.    ],
       [1.    ],
       ...,
       [0.0485],
       [0.032 ],
       [0.04  ]])
```

In [38]:
```
y_train
```

Out[38]:
```
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 [39]:
```
X_test
```

Out[39]:

	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

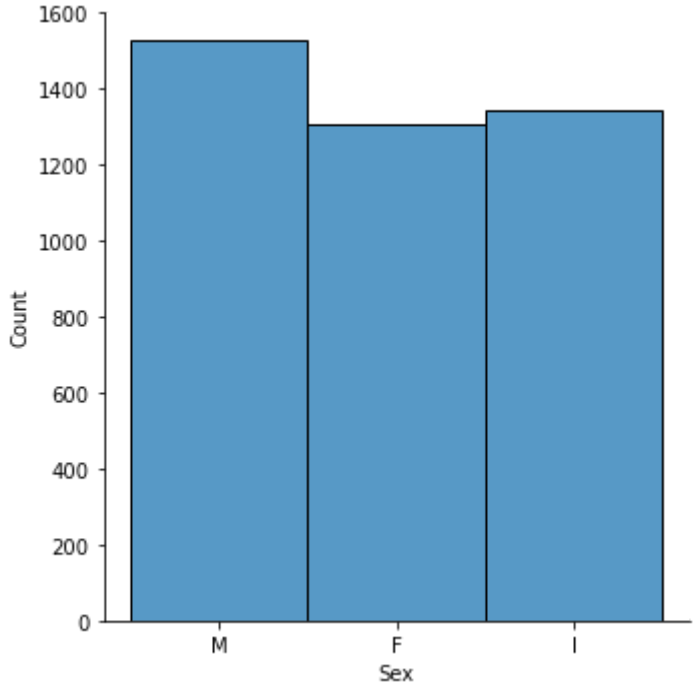
836 rows × 10 columns

In [40]:
```
y_test
```

Out[40]:
```
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
```

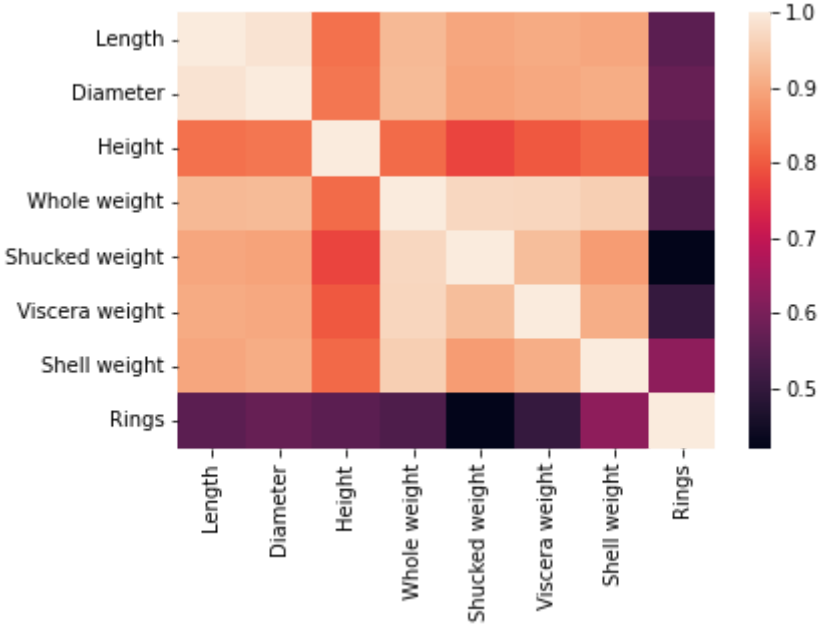
In [41]:
```
# 3.perform Visualization
#Univarient analysis
sns.displot(df['Sex'])
```

Out[41]:
```
<seaborn.axisgrid.FacetGrid at 0x24865593e20>
```



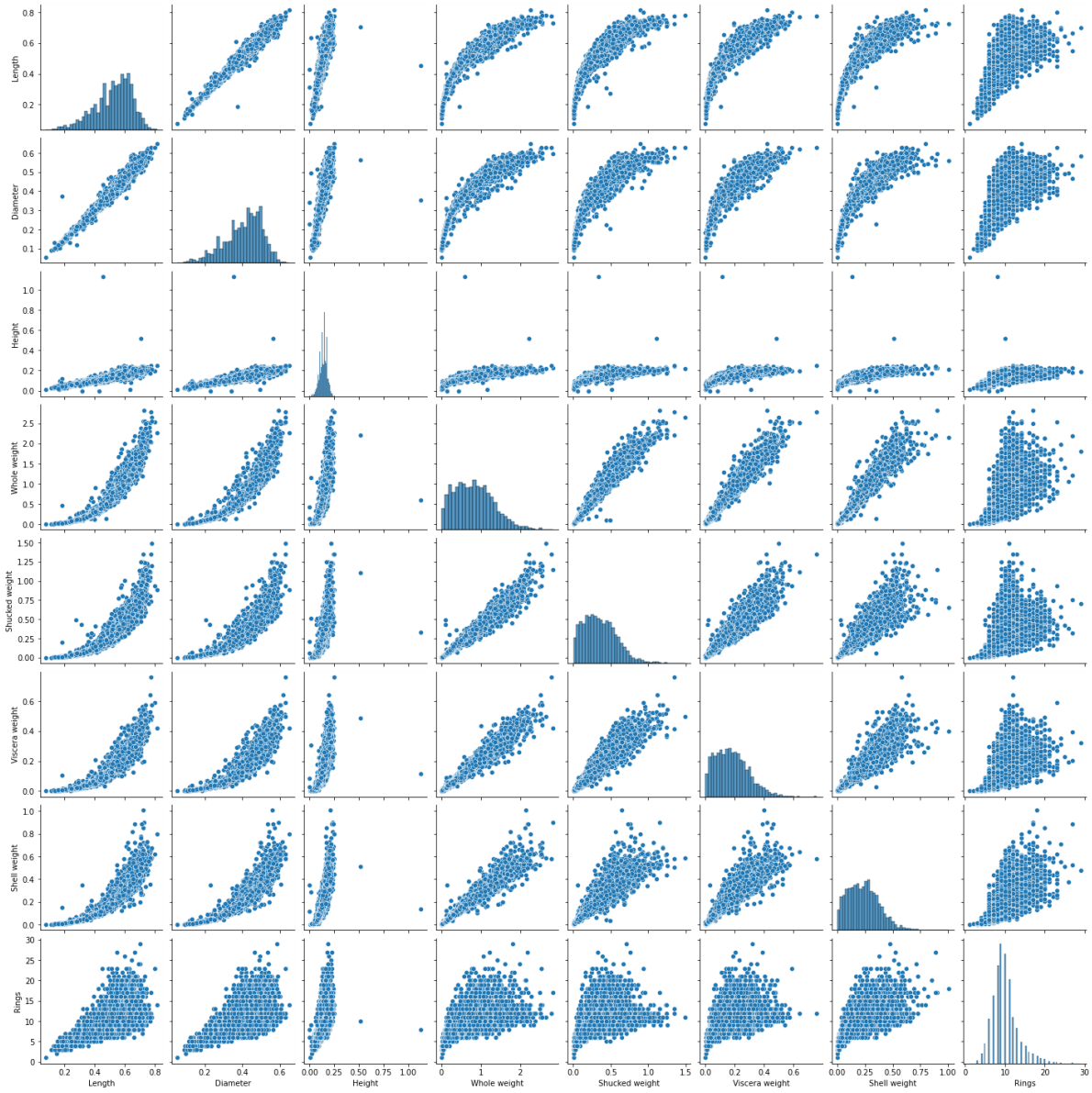
In [42]:
```
#Multivariant analysis
corr = df.corr()
sns.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns)
```

Out[42]:
```
<AxesSubplot:>
```



In [44]:
```
#Bi-variant analysis
sns.pairplot(df)
```

Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x24869b4b520>
```



In [45]:
```
# 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 [46]:
```
data.describe()
```

Out[46]:

	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 [47]:
```
# 6.outlier handling
df = pd.get_dummies(df)
dummy_df = df
```

In [50]:
```
import numpy as np
from collections import Counter


def detect_outliers(df, n, features):
    """
    Takes a dataframe df of features and returns a list of the indices
    corresponding to the observations containing more than n outliers according
    to the Tukey method.
    """
    outlier_indices = []

    # iterate over features(columns)
    for col in features:
        # 1st quartile (25%)
        Q1 = np.percentile(df[col], 25)
        # 3rd quartile (75%)
        Q3 = np.percentile(df[col], 75)
        # Interquartile range (IQR)
        IQR = Q3 - Q1

        # outlier step
        outlier_step = 1.5 * IQR

        # Determine a list of indices of outliers for feature col
        outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + outl

        # append the found outlier indices for col to the list of outlier indices
        outlier_indices.extend(outlier_list_col)

    # select observations containing more than 2 outliers
    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list(k for k, v in outlier_indices.items() if v > n)

    return multiple_outliers
```

In [51]:
```
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 [52]:
```
# 8.split the data into dependent and independent variables
# 9.scale independent variable
# x-independent variable & y-dependent variable
x=df.iloc[:,:1]
```

In [53]:
```
x
```

Out[53]:

	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 [54]:
```
df=pd.read_csv('abalone.csv')
```

In [55]:
```
df
y=df.iloc[:,1:]
```

In [56]:
```
y
```

Out[56]:

	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 [57]:
```
# 11.Build the model
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)
```

```
  Input In [57]
    df_sex_encoded = pd.DataFrame(transformed_sex_feature, columns = ["Sex_"+str(i
nt(i)) fo

^
SyntaxError: invalid syntax
```

In [58]:
```
df.head()
```

Out[58]:

	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 [59]:
```
# 14. Measure the performance using Metrics
df['Age'] = df['Rings'] + 1.5
df['Age'].head(5)
```

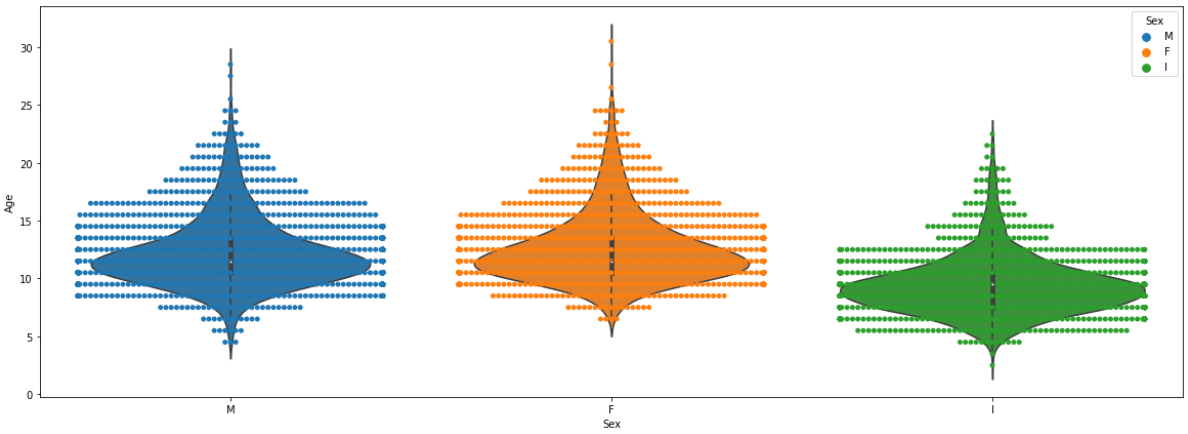
Out[59]:
```
0    16.5
1     8.5
2    10.5
3    11.5
4     8.5
Name: Age, dtype: float64
```

In [60]:
```
#'''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:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 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:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 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:\ProgramData\Anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 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[60]: <AxesSubplot:xlabel='Sex', ylabel='Age'>



In [ ]: