**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler

**import** statsmodels.api **as** sma

**from** statsmodels.stats.outliers\_influence **import** variance\_inflation\_factor

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.svm **import** SVC

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.model\_selection **import** cross\_val\_score

**from** sklearn.model\_selection **import** GridSearchCV

**from** sklearn.metrics **import** classification\_report

**from** sklearn.metrics **import** confusion\_matrix

**import** warnings

warnings**.**filterwarnings('ignore')

**1. Download the dataset: Dataset**

**2. Load the dataset into the tool.**

In [38]:

df **=** pd**.**read\_csv("nalaiya thiran\\assignment 3\\abalone.csv")

**3. Perform Below Visualizations.**

**∙ Univariate Analysis**

**∙ Bi-Variate Analysis**

**∙ Multi-Variate Analysis**

In [39]:

*#rename output variable*

df**.**rename(columns**=**{"Sex":"sex", "Length":"length", "Diameter":"diameter",

"Height":"height", "Whole weight":"whole\_weight",

"Shucked weight":"shucked\_weight", "Viscera weight":"viscera\_weight",

"Shell weight":"shell\_weight", "Rings":"rings"}, inplace **=** **True**)

In [40]:

df[df['height'] **==** 0] *#need to drop these rows.*

df**.**drop(index**=**[1257,3996], inplace **=** **True**)

df**.**shape

Out[40]:

(4175, 9)

In [41]:

df['age'] **=** df['rings']**+**1.5 *#AS per the problem statement*

df**.**drop('rings', axis **=** 1, inplace **=** **True**)

df**.**head()

*#categorical features*

temp **=** pd**.**concat([df['age'], df['sex']], axis**=**1)

f, ax **=** plt**.**subplots(figsize**=**(8, 6))

fig **=** sns**.**boxenplot(x**=**'sex', y**=**"age", data**=**df)

fig**.**axis(ymin**=**0, ymax**=**30);

**ANALYSIS**

* There is no difference in age of rings for male and female (8-19). But in infants, it lies between (5-10)

**Count Plot**

In [42]:

sns**.**countplot('sex', data**=**df)

plt**.**title('Distributed Classes', fontsize**=**14)

plt**.**show()

**Histograms: Understanding the Distribution of the Numerical Features**

In [43]:

df**.**hist(figsize **=** (20,10), layout **=** (2,4))

Out[43]:

array([[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F69C8940>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F69EEAC0>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F6A1CEB0>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F6A56340>],

[<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F6A82790>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F6AB25E0>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F6AB26D0>,

<matplotlib.axes.\_subplots.AxesSubplot object at 0x00000206F6ADCEB0>]],

dtype=object)

**ANALYSIS**

* Skewness of the height is too high. (need to normalise later...)
* Need to check skewness for all varibles

**Skewness of the Variables**

In [44]:

df**.**skew()**.**sort\_values(ascending **=** **False**)

Out[44]:

height 3.166364

age 1.113754

shucked\_weight 0.718735

shell\_weight 0.621081

viscera\_weight 0.591455

whole\_weight 0.530549

diameter -0.610182

length -0.640993

dtype: float64

**ANALYSIS:**

* Skewness is close to 0 for Normal distribution curve.
* Height has the highest skewness of 3.17.
  + May be there are outliers in height, we need to check that and remove them before modeling.
  + Will check the coorelation with the dependent variable (Rings)
  + Will use IQR algorithm to remove outliers.

**Coorelation Plot**

In [45]:

corr **=** df**.**corr()

plt**.**figure(figsize **=** (10,10))

ax **=** sns**.**heatmap(corr, vmin **=** **-**1, center **=** 0, annot **=** **True**, cmap **=** 'mako')

**ANALYSIS**

* No Negative correlation found
* High coorelation between **Length & Diameter**
* High corelation between **shucked weight, viscera weight Vs Whole\_weight** & **Shell weight vs Whole\_weight**

In [46]:

upper\_tri **=** corr**.**where(np**.**triu(np**.**ones(corr**.**shape),k**=**1)**.**astype(np**.**bool))

columns\_to\_drop **=** [column **for** column **in** upper\_tri**.**columns **if** any(upper\_tri[column] **>** 0.95)] *#highly correlated variables to be removed.*

print("Columns to drop:\n", columns\_to\_drop)

Columns to drop:

['diameter', 'shucked\_weight', 'viscera\_weight', 'shell\_weight']

**ANALYSIS**

* We will remove the above columns, before proceeding any further.

**4. Perform descriptive statistics on the dataset.**

In [47]:

df**.**head()

Out[47]:

|  | **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 |

In [48]:

df**.**shape

Out[48]:

(4175, 9)

In [49]:

df**.**describe()

Out[49]:

|  | **length** | **diameter** | **height** | **whole\_weight** | **shucked\_weight** | **viscera\_weight** | **shell\_weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 4175.000000 | 4175.00000 | 4175.000000 | 4175.000000 | 4175.000000 | 4175.000000 | 4175.000000 | 4175.000000 |
| **mean** | 0.524065 | 0.40794 | 0.139583 | 0.829005 | 0.359476 | 0.180653 | 0.238834 | 11.435090 |
| **std** | 0.120069 | 0.09922 | 0.041725 | 0.490349 | 0.221954 | 0.109605 | 0.139212 | 3.224227 |
| **min** | 0.075000 | 0.05500 | 0.010000 | 0.002000 | 0.001000 | 0.000500 | 0.001500 | 2.500000 |
| **25%** | 0.450000 | 0.35000 | 0.115000 | 0.442250 | 0.186250 | 0.093500 | 0.130000 | 9.500000 |
| **50%** | 0.545000 | 0.42500 | 0.140000 | 0.800000 | 0.336000 | 0.171000 | 0.234000 | 10.500000 |
| **75%** | 0.615000 | 0.48000 | 0.165000 | 1.153500 | 0.502000 | 0.253000 | 0.328750 | 12.500000 |
| **max** | 0.815000 | 0.65000 | 1.130000 | 2.825500 | 1.488000 | 0.760000 | 1.005000 | 30.500000 |

In [50]:

df**.**info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 4175 entries, 0 to 4176

Data columns (total 9 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 sex 4175 non-null object

1 length 4175 non-null float64

2 diameter 4175 non-null float64

3 height 4175 non-null float64

4 whole\_weight 4175 non-null float64

5 shucked\_weight 4175 non-null float64

6 viscera\_weight 4175 non-null float64

7 shell\_weight 4175 non-null float64

8 age 4175 non-null float64

dtypes: float64(8), object(1)

memory usage: 486.2+ KB

**5. Check for Missing values and deal with them.**

In [51]:

df[df**.**duplicated()]

Out[51]:

|  | **sex** | **length** | **diameter** | **height** | **whole\_weight** | **shucked\_weight** | **viscera\_weight** | **shell\_weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

In [52]:

df**.**isna()**.**sum()

Out[52]:

sex 0

length 0

diameter 0

height 0

whole\_weight 0

shucked\_weight 0

viscera\_weight 0

shell\_weight 0

age 0

dtype: int64

**there is no missing values and duplicates in dataframe**

**6. Find the outliers and replace them outliers**

In [53]:

**for** i **in** df:

**if** df[i]**.**dtype**==**'int64' **or** df[i]**.**dtypes**==**'float64':

q1**=**df[i]**.**quantile(0.25)

q3**=**df[i]**.**quantile(0.75)

iqr**=**q3**-**q1

upper**=**q3**+**1.5**\***iqr

lower**=**q1**-**1.5**\***iqr

df[i]**=**np**.**where(df[i] **>**upper, upper, df[i])

df[i]**=**np**.**where(df[i] **<**lower, lower, df[i])

**After removing outliers, boxplot will be like**

In [54]:

**import** matplotlib.pyplot **as** mtp

In [55]:

**def** box\_scatter(data, x, y):

fig, (ax1, ax2) **=** plt**.**subplots(nrows**=**2, ncols**=**1, figsize**=**(16,6))

sns**.**boxplot(data**=**data, x**=**x, ax**=**ax1)

sns**.**scatterplot(data**=**data, x**=**x,y**=**y,ax**=**ax2)

In [56]:

**for** i **in** df:

**if** df[i]**.**dtype**==**'int64' **or** df[i]**.**dtypes**==**'float64':

mtp**.**boxplot(df[i])

mtp**.**show()

**7. Check for Categorical columns and perform encoding.**

In [57]:

df**.**head()

Out[57]:

|  | **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 |

In [58]:

**from** sklearn.preprocessing **import** LabelEncoder

encoder**=**LabelEncoder()

df['sex']**=**encoder**.**fit\_transform(df['sex'])

In [59]:

df**.**head()

Out[59]:

|  | **sex** | **length** | **diameter** | **height** | **whole\_weight** | **shucked\_weight** | **viscera\_weight** | **shell\_weight** | **age** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 16.5 |
| **1** | 2 | 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** | 2 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 11.5 |
| **4** | 1 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 8.5 |

**8. Split the data into dependent and independent variables.**

In [66]:

x**=**df**.**iloc[:,:**-**1]

x**.**head()

Out[66]:

|  | **sex** | **length** | **diameter** | **height** | **whole\_weight** | **shucked\_weight** | **viscera\_weight** | **shell\_weight** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2 | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 |
| **1** | 2 | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 |
| **2** | 0 | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 |
| **3** | 2 | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 |
| **4** | 1 | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 |

In [67]:

y**=**df**.**iloc[:,**-**1]

y**.**head()

Out[67]:

0 16.5

1 8.5

2 10.5

3 11.5

4 8.5

Name: age, dtype: float64

**9. Scale the independent variable**

In [68]:

**from** sklearn.preprocessing **import** StandardScaler

scaler**=**StandardScaler()

x**=**scaler**.**fit\_transform(x)

**10. Split the data into training and testing**

In [69]:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.33)

In [70]:

x\_train**.**shape

Out[70]:

(2797, 8)

In [71]:

x\_test**.**shape

Out[71]:

(1378, 8)

**11. Build the Model**

In [74]:

**from** sklearn.ensemble **import** RandomForestRegressor

reg**=**RandomForestRegressor()

**12. Train the Model**

In [76]:

reg**.**fit(x\_train,y\_train)

Out[76]:

RandomForestRegressor()

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.  
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

**13. Test the Model**

In [77]:

y\_pred**=**reg**.**predict(x\_test)

**14. Measure the performance using Metrics.**

In [78]:

**from** sklearn.metrics **import** mean\_squared\_error

**import** math

print(math**.**sqrt(mean\_squared\_error(y\_test,y\_pred)))

1.8081309009200963