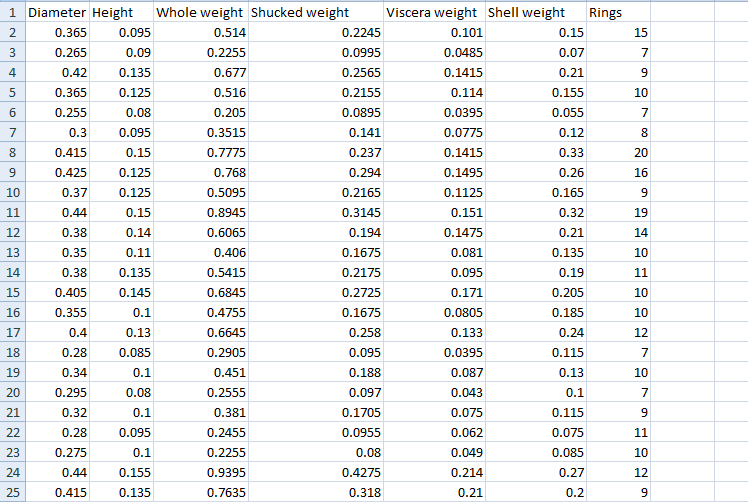
**Abalone Age Prediction**

**Assignment -3**

|  |  |
| --- | --- |
| Assignment Date | 03 October 2022 |
| Team ID | PNT2022TMID27812 |
| Project Name | Smart Lender-Application Credibility  Prediction for loan Approval |
| Student Name | Adnan Ahmed . S |
| Student Roll Number | 311519104005 |
| Maximum Marks | 2 Marks |

**Question-1.**Download dataset

**Solution:**



**Question-2.**Load the dataset

**Solution:**

**import numpy as np**

**import pandas as pd**

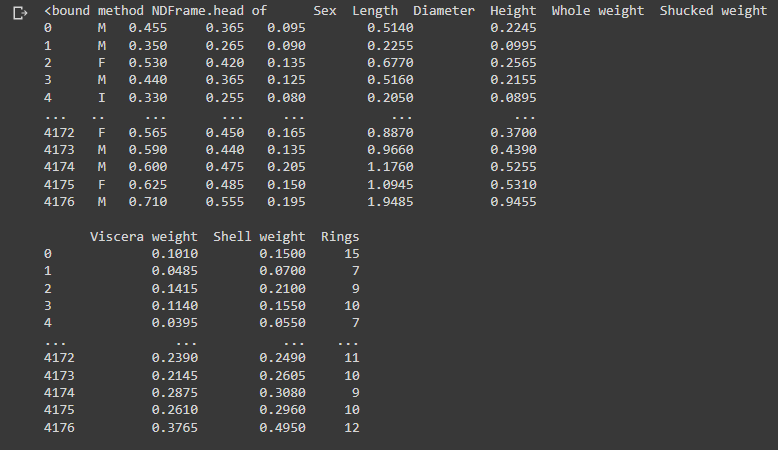
**import seaborn as sns**

**import matplotlib.pyplot as plt**

**import sklearn**

**data = pd.read\_csv(r"abalone.csv")**

**data.head**

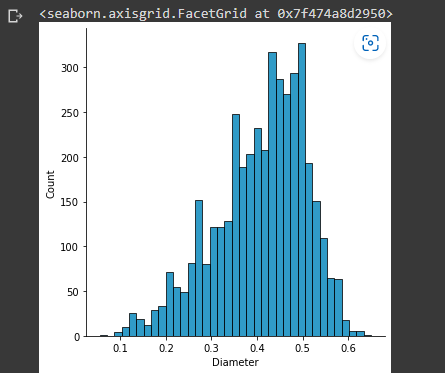
****

**Question-3.**Perform Below Visualizations.

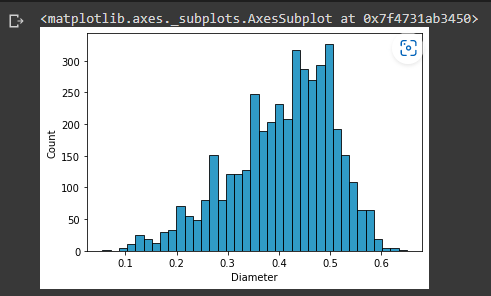
***3.1 Univariate Analysis***

**Solution:**

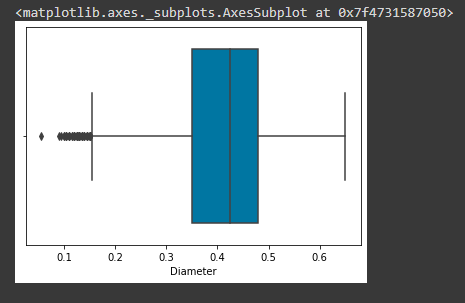
**sns.displot(data['** **Diameter'])**

****

**sns.histplot(data[' Diameter '])**

****

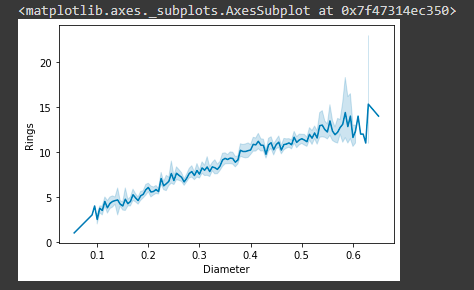
**sns.boxplot(x = data[' Diameter '])**

****

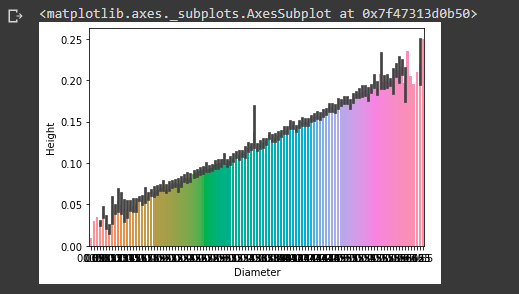
***3.2 Bivariate Analysis***

**Solution:**

**sns.lineplot(data = data, x = 'Diameter', y = 'Rings')**

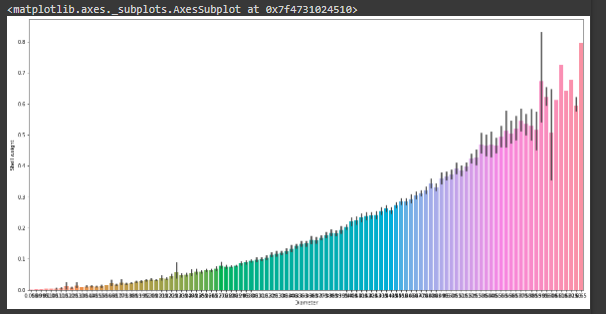
****

**sns.barplot(data = data, x = 'Diameter',y = 'Height')**

****

**plt.figure(figsize=(20,20))**

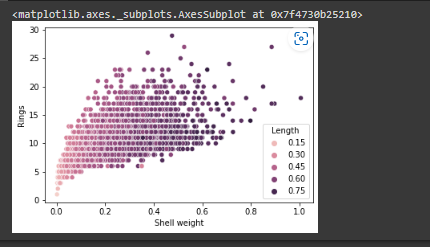
**sns.barplot(data = data, x = 'Diameter', y = 'Shell weight)**



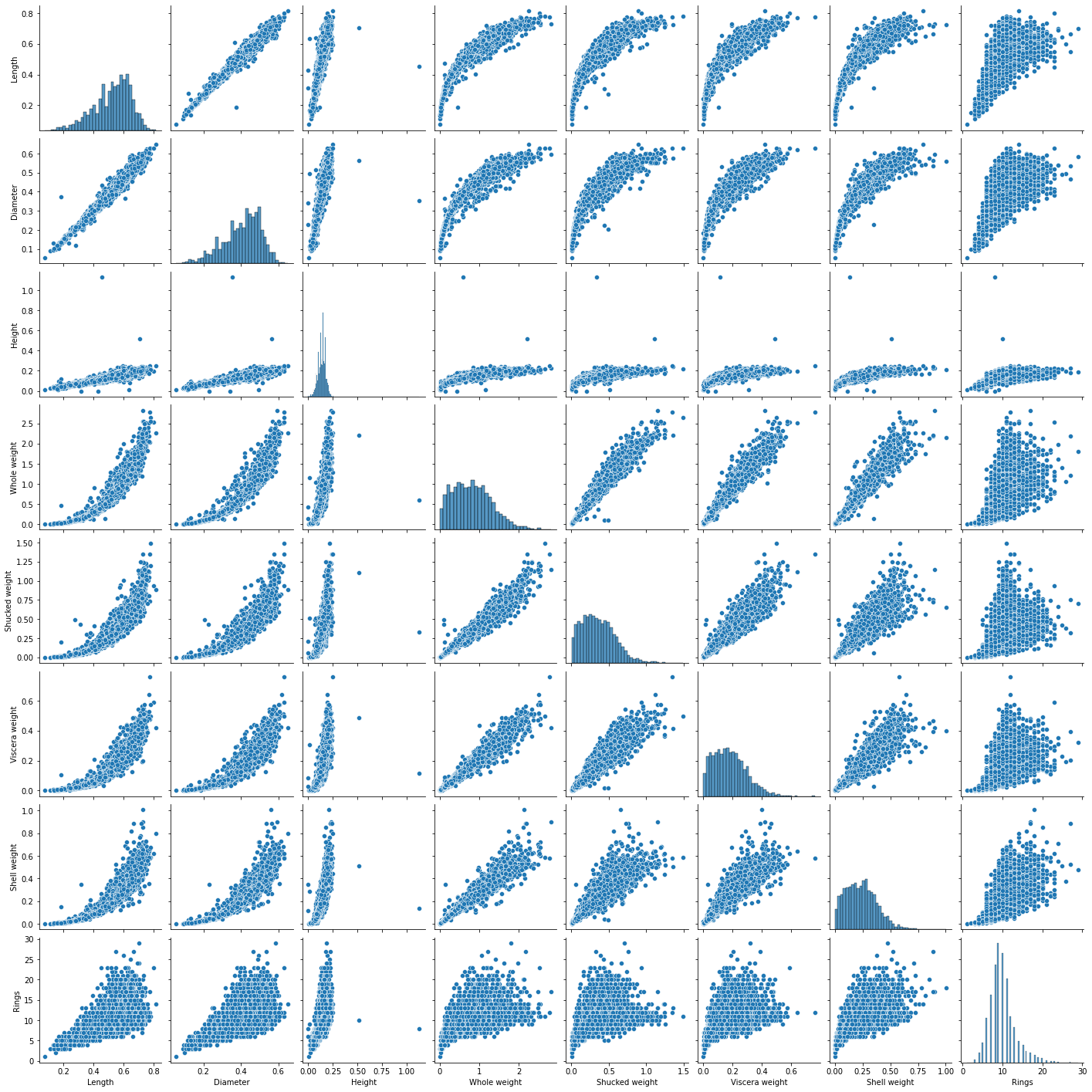
***3.3 Multivariate Analysis***

**Solution:**

**sns.scatterplot(data = data x = 'Shell weight', y = 'Rings', hue = 'Length')**

****

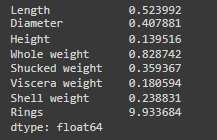
**sns.pairplot(data)**

****

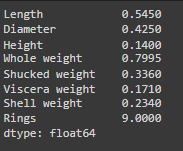
**Question-4.**Perform descriptive statistics on the dataset.

**Solution:**

**data.mean(numeric\_only = True)**



**data.median(numeric\_only = True)**

****

**data['Whole weight'].mode()**

0 0.2225 dtype: float64

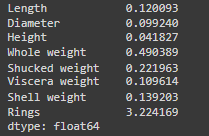
**data['Length'].mode()**

0 0.550 1 0.625 dtype: float64

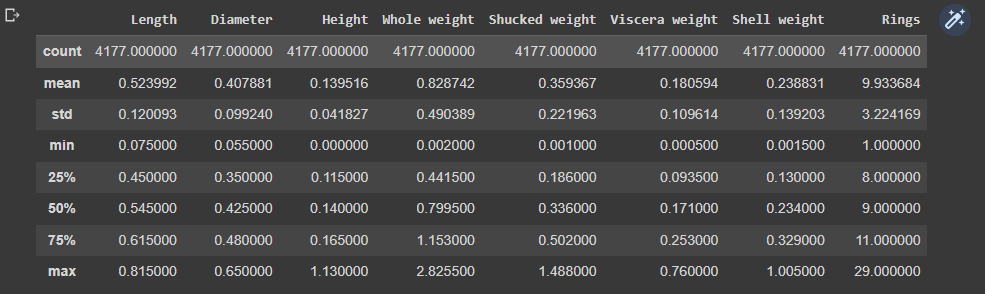
**data['Rings'].unique()**

array([15, 7, 9, 10, 8, 20, 16, 19, 14, 11, 12, 18, 13, 5, 4, 6, 21, 17, 22, 1, 3, 26, 23, 29, 2, 27, 25, 24])

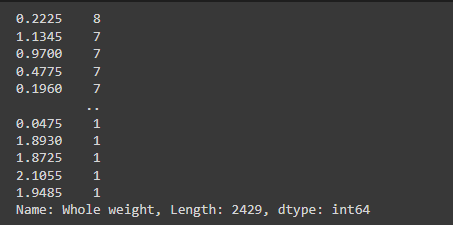
**data.std(numeric\_only=True)**

****

**data.describe()**

****

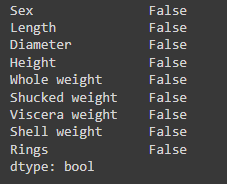
**data['Whole weight'].value\_counts()**



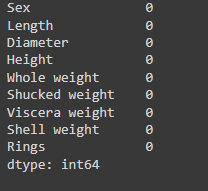
**Question-5.**Handle the Missing values.

**Solution:**

**data.isnull().any()**



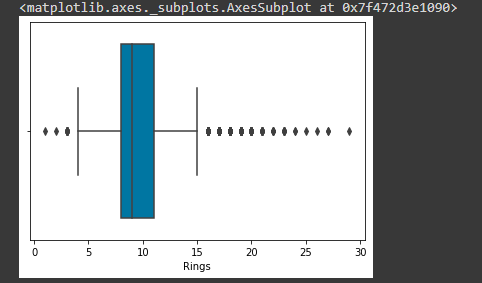
**data.isnull().sum()**



**Question-6.**Find the outliers and replace the outliers

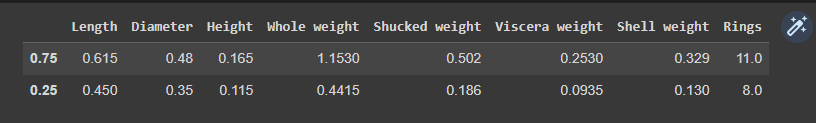
**Solution:**

**sns.boxplot(x = data['Rings'])**

****

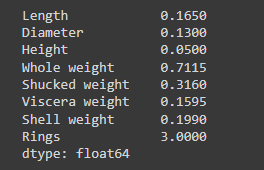
**q = data.quantile([0.75,0.25])**

**q**

****

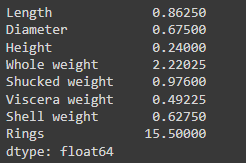
**iqr = q.iloc[0] - q.iloc[1]**

**iqr**

****

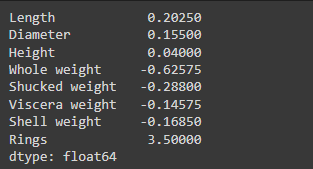
**u = q.iloc[0] + (1.5\*iqr)**

**u**

****

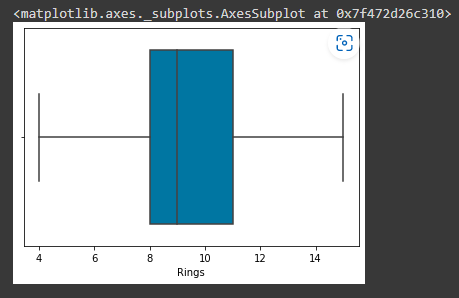
**l = q.iloc[1] - (1.5\*iqr)**

**l**

****

**data['Rings'] = np.where(np.logical\_or(data['Rings']>15, data['Rings']<4), 9, data['Rings'])**

**sns.boxplot(x = data['Rings'])**

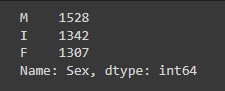
****

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

**Solution:**

**data['Age'] = data['Rings'] + 1.5**

**data['Sex'].value\_counts()**

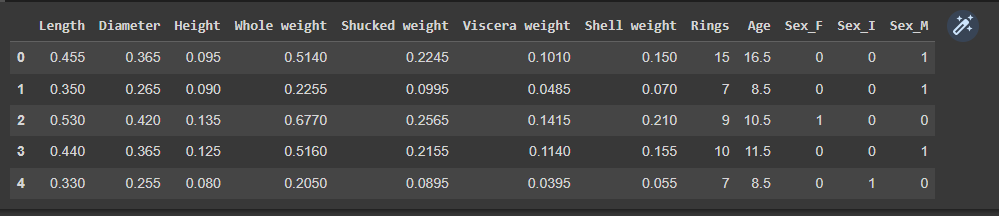
****

**from sklearn.preprocessing import LabelEncoder, OneHotEncoder**

**le = LabelEncoder()**

**data = pd.get\_dummies(data)**

**data.head()**

****

**Question-8.**Split the data into dependent and independent variables split the data in X and Y

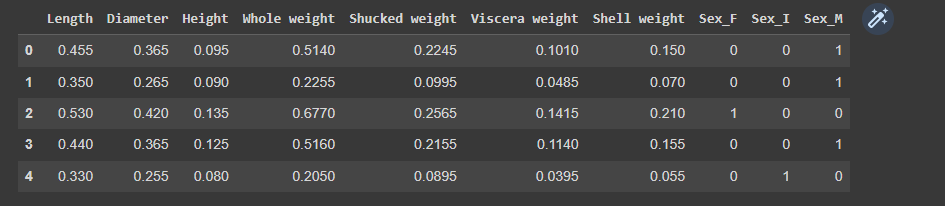
**Solution:**

**y = data['Age']**

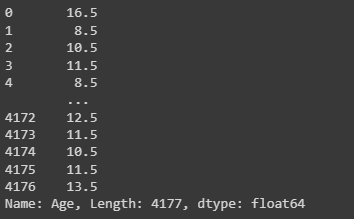
**data = data.drop(['Rings','Age'], axis=1)**

**x = data**

**data.head**

****

**y**

****

**Question-9.**Scale the independent variables

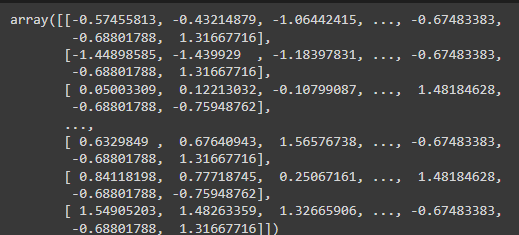
**Solution:**

**from sklearn.preprocessing import StandardScaler, MinMaxScaler**

**sc = StandardScaler()**

**x\_scaled = sc.fit\_transform(x)**

**x\_scaled**

****

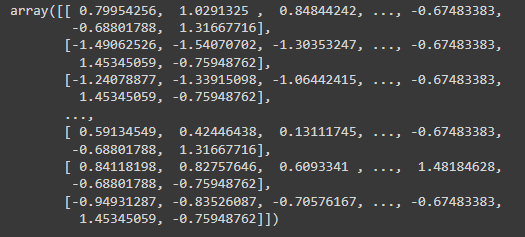
**Question-10.**Split data into Training and Testing

**Solution:**

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

**x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_scaled, y, test\_size = 0.3, random\_state = 0)**

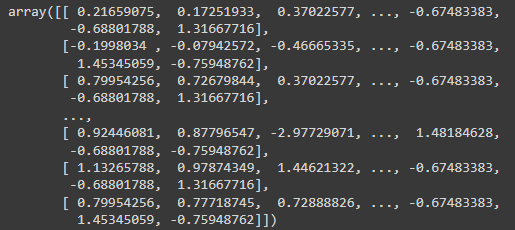
**x\_train**

****

**x\_train.shape**

(2923, 10)

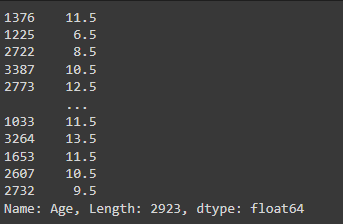
**x\_test**

****

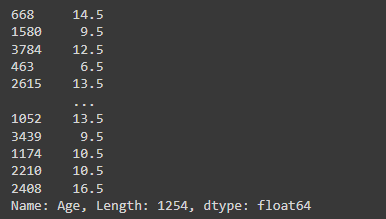
**x\_test.shape**

(1254, 10)

**y\_train**

****

**y\_test**

****

**Question-11.** Build the model

**Solution:**

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

**lr = LinearRegression()**

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

LinearRegression()

**predict = lr.predict(x\_test)**

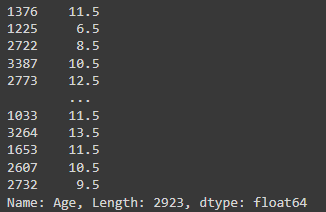
**predict**

****

**Question-12.** Train the model

**Solution:**

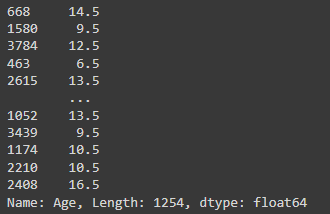
**y\_train**

****

**Question-13.** Test the model

**Solution:**

**y\_test**

****

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

**Solution:**

**from sklearn.metrics import r2\_score, mean\_squared\_error**

**mse = mean\_squared\_error(y\_test, predict)**

**rmse = np.sqrt(mse)**

**print("mse = ", mse)**

**print("rmse = ", rmse)**

**r2\_score(y\_test,predict)**

****