ASSIGNMENT-3

REGRESSION MODEL

DATE	04-10-2022
TEAM_ID	**PNT2022TMID46440**
PROJECT_NAME	**EFFICIENT WATER QUALITY ANALYSIS AND PREDICTION USING MACHINE LEARNING**
STUDENT_NAME	**R.KARTHICKRAJA**
MAXIMUM_MARKS	2 Marks

Problem Statement: Abalone Age Prediction

Description:- Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

Load the dataset into the tool.

```
[1] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns

[2] data=pd.read_csv(r'D:\ADS_Assignment\abalone.csv',encoding='ISO-88591',
low_memory='False')
data.head()
```

Output:

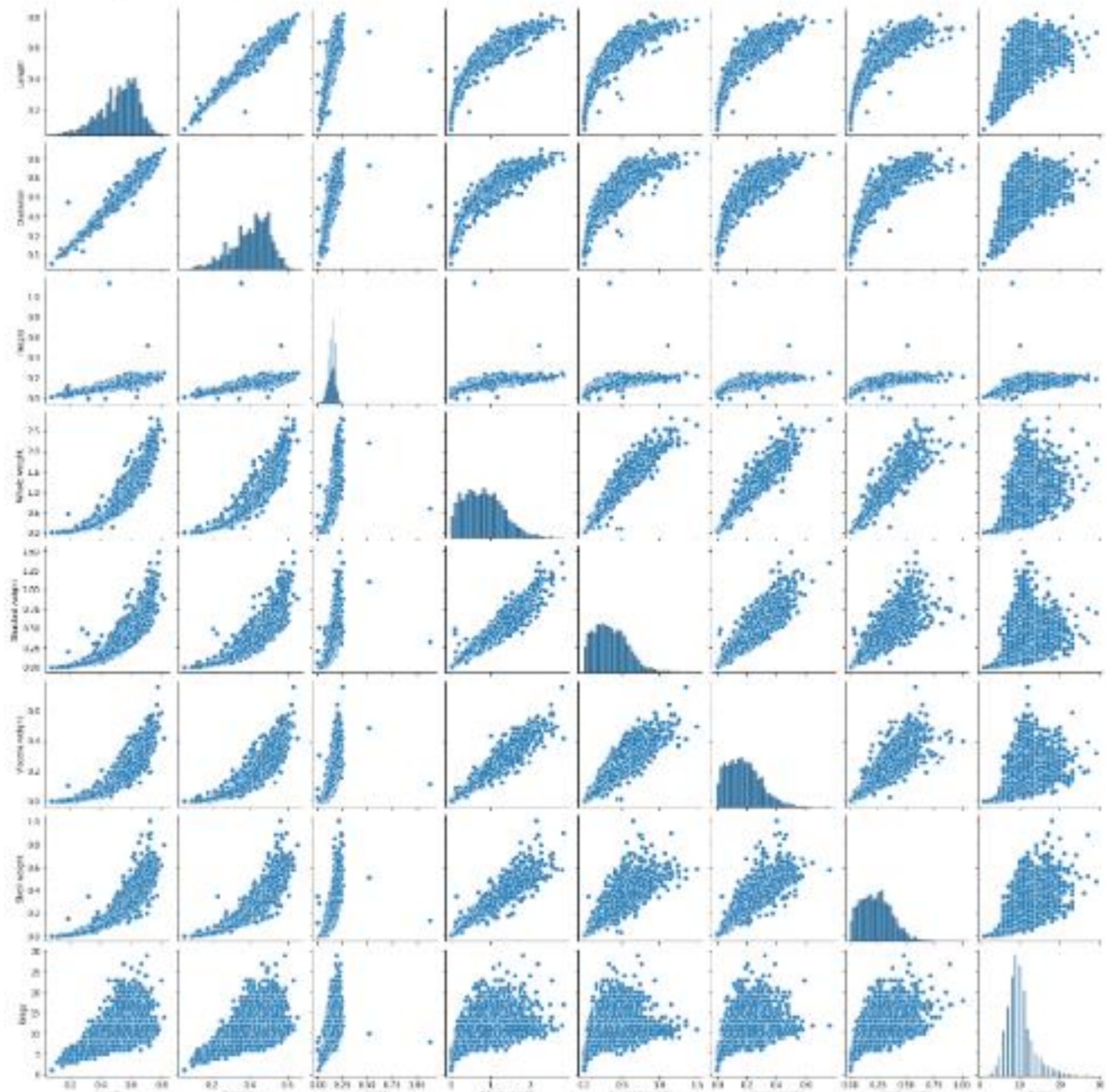
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330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

Visualizations.

[1] sns.pairplot(data)

output:

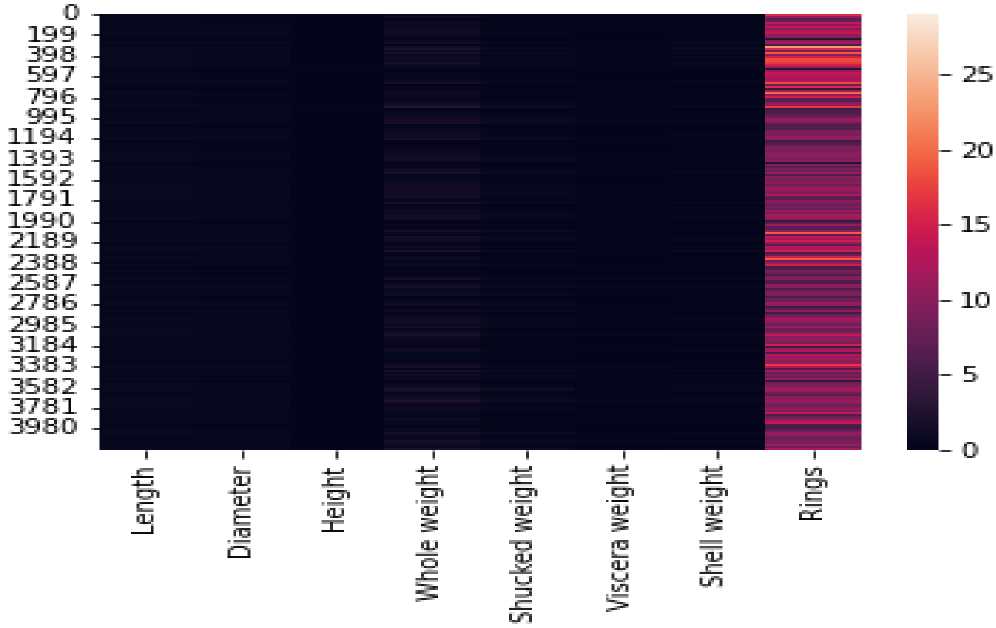
<seaborn.axisgrid.PairGrid at 0x1ddc7e691c0>



[2] sns.heatmap(data[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight', 'Rings']])

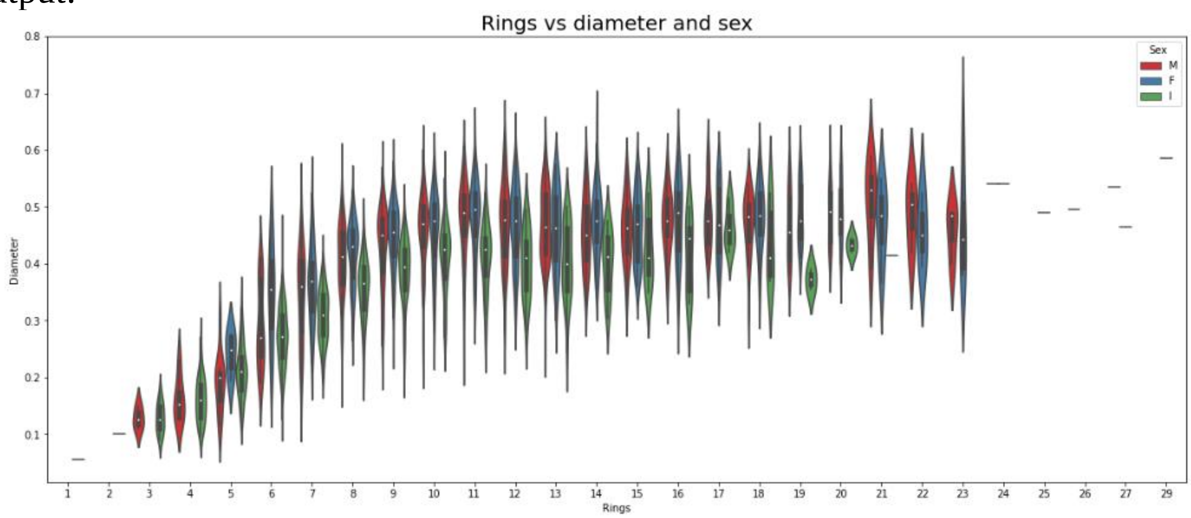
Output:

```
<AxesSubplot:>
```



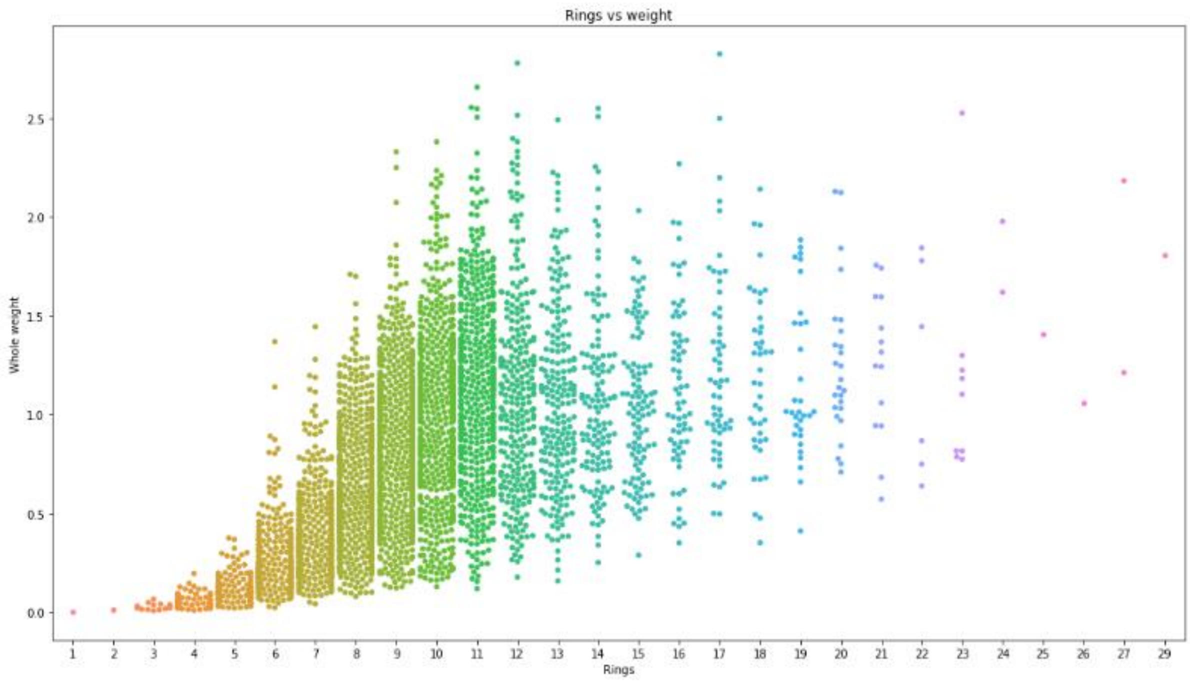

```
[3] plt.rcParams['figure.figsize'] = (20, 8)
    sns.violinplot(data['Rings'], data['Diameter'], hue = data['Sex'], palette = 'Set1')
    plt.title('Rings vs diameter and sex', fontsize = 20)
```

Output:



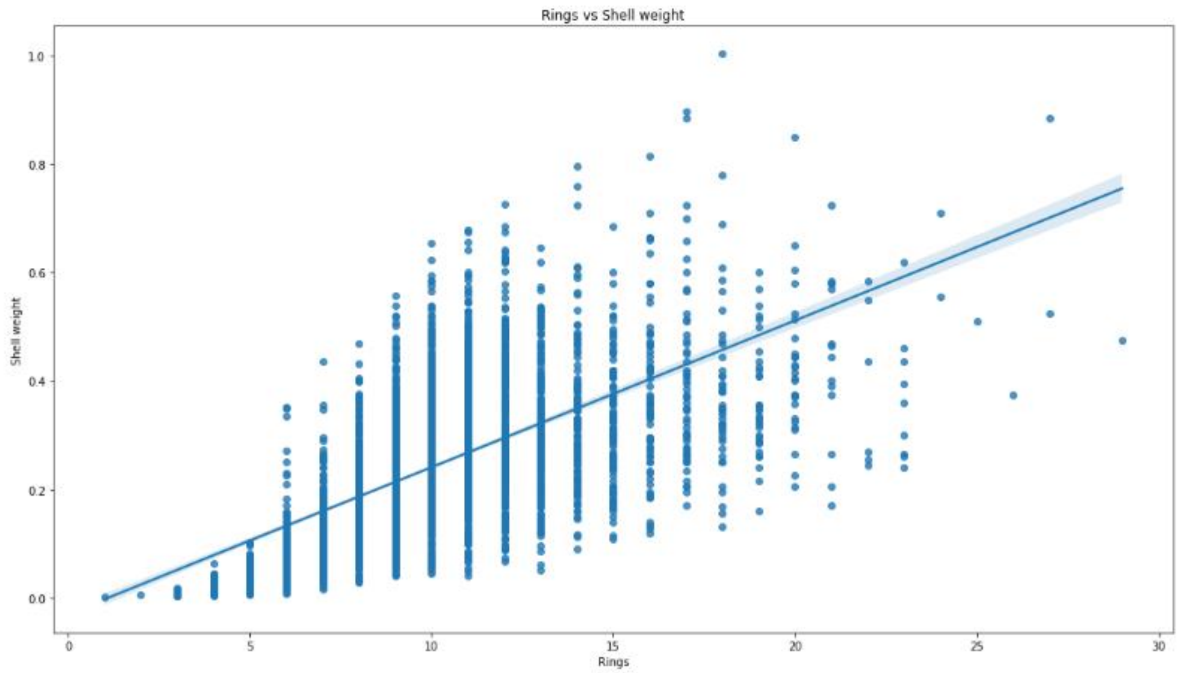
```
[4]  plt.rcParams['figure.figsize'] = (18, 10)
     sns.swarmplot(data['Rings'], data['Whole weight'])
     plt.title('Rings vs weight')
```

Output:




```
[5]  plt.rcParams['figure.figsize'] = (18, 10)
     sns.regplot(data['Rings'], data['Shell weight'])
     plt.title('Rings vs Shell weight')
```

Output:



Check for Missing values and deal with them.

[1] data.isnull().sum()

Output:

```
:  Sex               0
   Length            0
   Diameter          0
   Height            0
   Whole weight      0
   Shucked weight    0
   Viscera weight    0
   Shell weight      0
   Rings             0
   dtype: int64
```

Check for Categorical columns and perform encoding.

[1] data.columns

Output:

Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight', 'Rings'], dtype='object')

```
[2] '''from sklearn.preprocessing import LabelEncoder
    le = LabelEncoder()
    data['Sex'] = le.fit_transform(data['Sex'])
    data['Sex'].value_counts()'''
    data = pd.get_dummies(data)
    data.head()
```

Output:

	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	0.150	15	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	0	1	0

Split the data into dependent and independent variables.

```
[1] y = data['Rings']
    data = data.drop(['Rings'], axis = 1)
    x = data

    # getting the shapes
    print("Shape of x:", x.shape)
    print("Shape of y:", y.shape)
```

output:

```
Shape of x: (4177, 10)
Shape of y: (4177,)
```

Split the data into training and testing.

```
[1] 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)
```

```
# getting the shapes
print("Shape of x_train :", x_train.shape)
print("Shape of x_test :", x_test.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of y_test :", y_test.shape)
```

output:

```
Shape of x_train : (3341, 10)
Shape of x_test : (836, 10)
Shape of y_train : (3341,)
Shape of y_test : (836,)
```

Build the Model and Measure the performance using Metrics.

```
[1] from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import mean_squared_error
    from sklearn.metrics import r2_score

    model = RandomForestClassifier()
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    # evaluation
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    print("RMSE :", rmse)

    # r2 score
    r2 = r2_score(y_test, y_pred)
    print("R2 Score :", r2)
```

Output:

```
RMSE : 2.5157399246386842
R2 Score : 0.4172406942189353
```

```
[2]  #for purmutation importance
     import eli5
     from eli5.sklearn import PermutationImportance
     perm = PermutationImportance(model, random_state = 0).fit(x_test, y_test)
     eli5.show_weights(perm, feature_names = x_test.columns.tolist())
```

Output:

Weight	Feature
0.0340 ± 0.0350	Shell weight
0.0330 ± 0.0146	Shucked weight
0.0132 ± 0.0134	Viscera weight
0.0100 ± 0.0134	Length
0.0084 ± 0.0127	Sex_I
0.0045 ± 0.0051	Whole weight
0.0026 ± 0.0096	Height
0.0014 ± 0.0141	Sex_M
-0.0043 ± 0.0070	Sex_F
-0.0067 ± 0.0227	Diameter