Assignment - 4
ABALONE AGE PREDICTION

Assignment Date	01 September 2022
Student Name	Suraj Kumar. R
Student Roll Number	1902241
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

Attribute Information:

Given is the attribute name, attribute type, measurement unit, and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description

1- Sex / nominal / -- / M, F, and I (infant)

2- Length / continuous / mm / Longest shell measurement

3- Diameter / continuous / mm / perpendicular to length

4- Height / continuous / mm / with meat in shell

5- Whole weight / continuous / grams / whole abalone

6- Shucked weight / continuous / grams / weight of meat

7- Viscera weight / continuous / grams / gut weight (after bleeding)

8- Shell weight / continuous / grams / after being dried

9- Rings / integer / -- / +1.5 gives the age in years

Building a Regression Model

1. Download the dataset: Dataset
2. Load the dataset into the tool.

3. Perform Below Visualizations.

· Univariate Analysis

· Bi-Variate Analysis

· Multi-Variate Analysis

4. Perform descriptive statistics on the dataset.
5. Check for Missing values and deal with them.
6. Find the outliers and replace them outliers
7. Check for Categorical columns and perform encoding.
8. Split the data into dependent and independent variables.
9. Scale the independent variables
10. Split the data into training and testing
11. Build the Model
12. Train the Model
13. Test the Model
14. Measure the performance using Metrics

CODING:

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

from google.colab import files
upload=files.upload()
df = pd.read_csv('abalone.csv')
```

df.describe()

OUTPUT:

Length Diameter Height Whole weight Shucked weight Viscera weight Shell weight

Rings

count 4177.000000 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
0.238831 9.933684

std 0.120093 0.099240 0.041827 0.490389 0.221963 0.109614
0.139203 3.224169

min 0.075000 0.055000 0.000000 0.002000 0.001000 0.000500

0.001500 1.000000

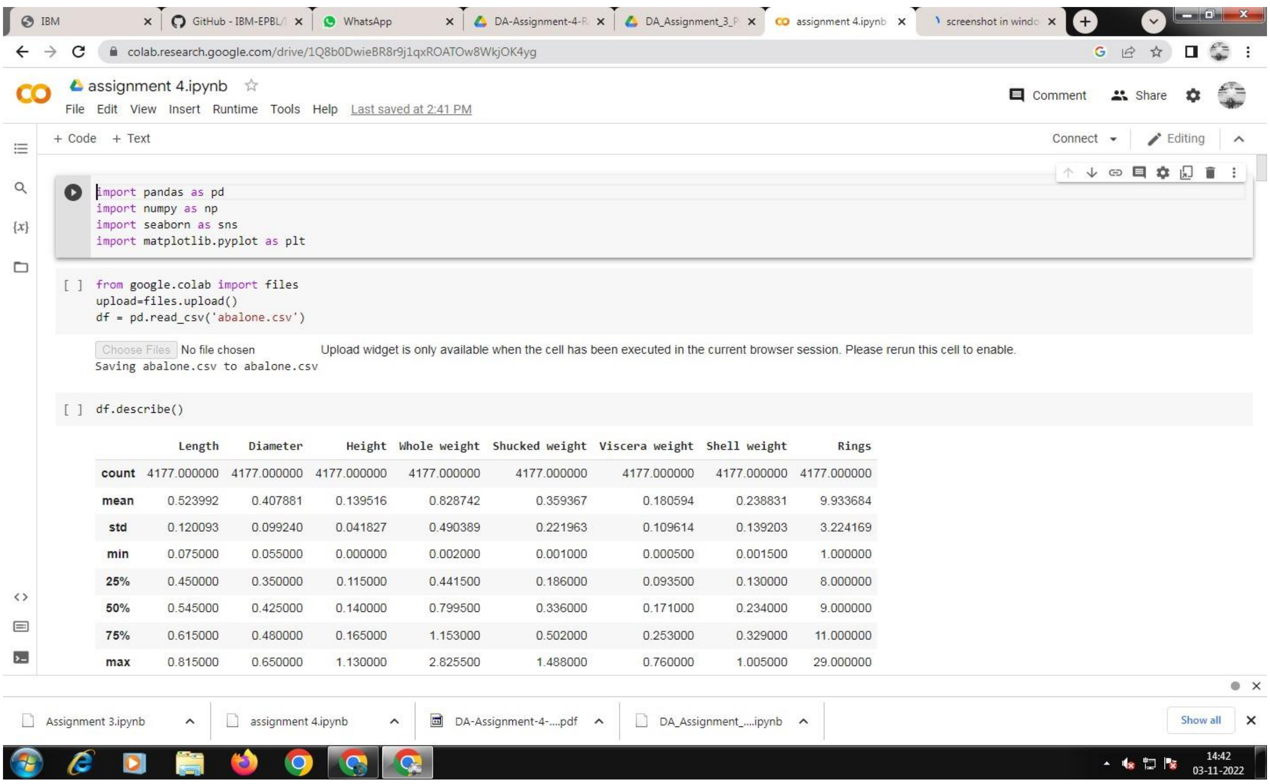
25% 0.450000 0.350000 0.115000 0.441500 0.186000 0.093500
0.130000 8.000000

50% 0.545000 0.425000 0.140000 0.799500 0.336000 0.171000
0.234000 9.000000

75% 0.615000 0.480000 0.165000 1.153000 0.502000 0.253000

0.329000 11.000000

max 0.815000 0.650000 1.130000 2.825500 1.488000 0.760000
1.005000 29.000000



CODING:

```
df.head()
```

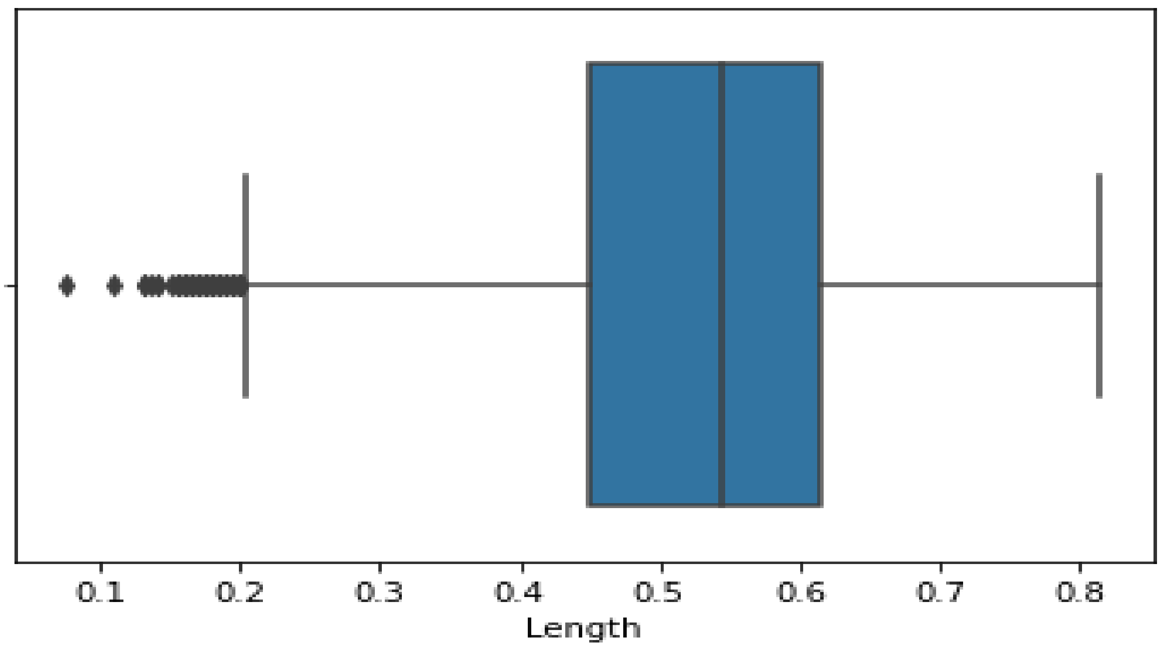
OUTPUT:

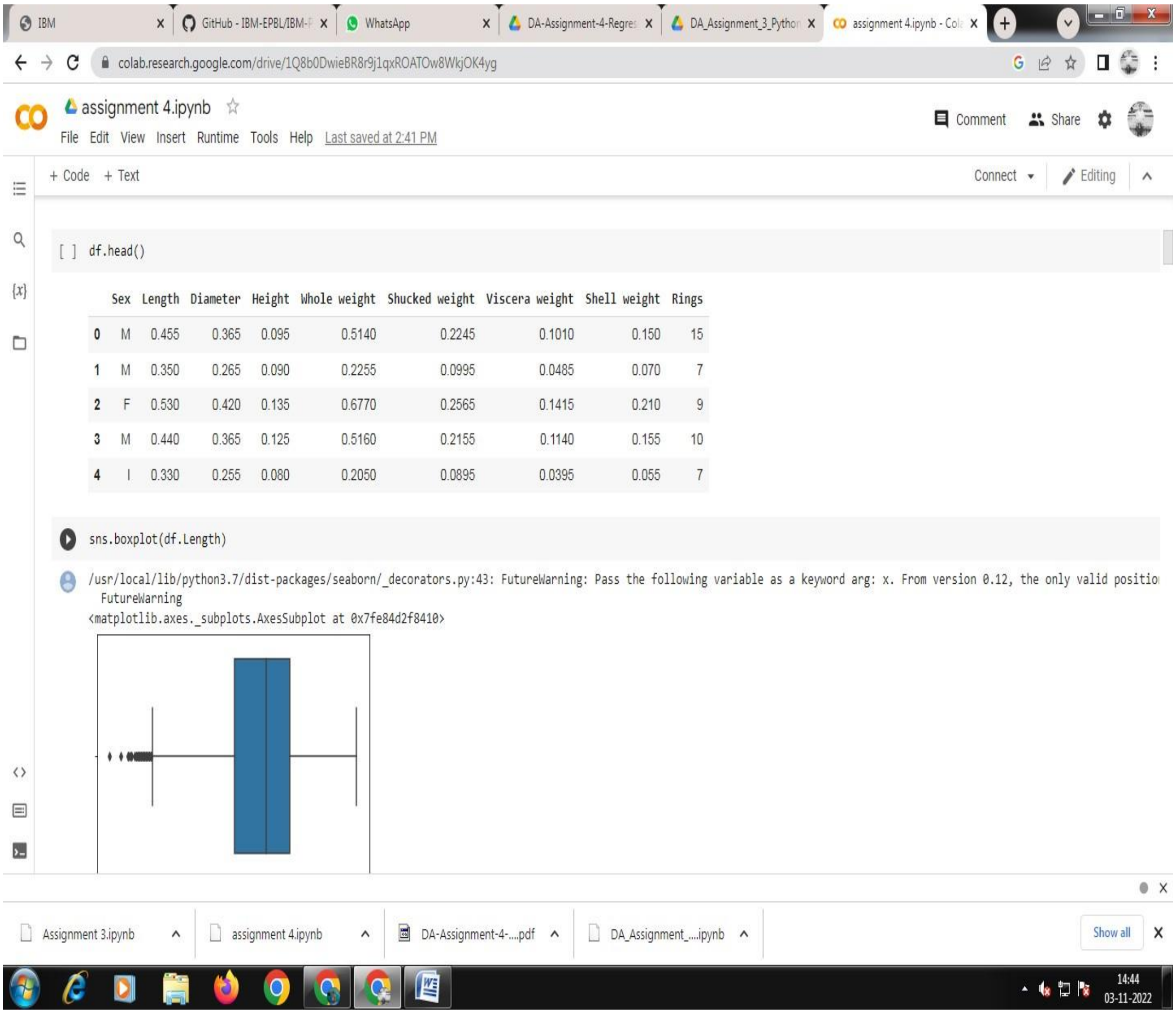
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330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

CODING:

```
sns.boxplot(df.Length)
```

OUTPUT:

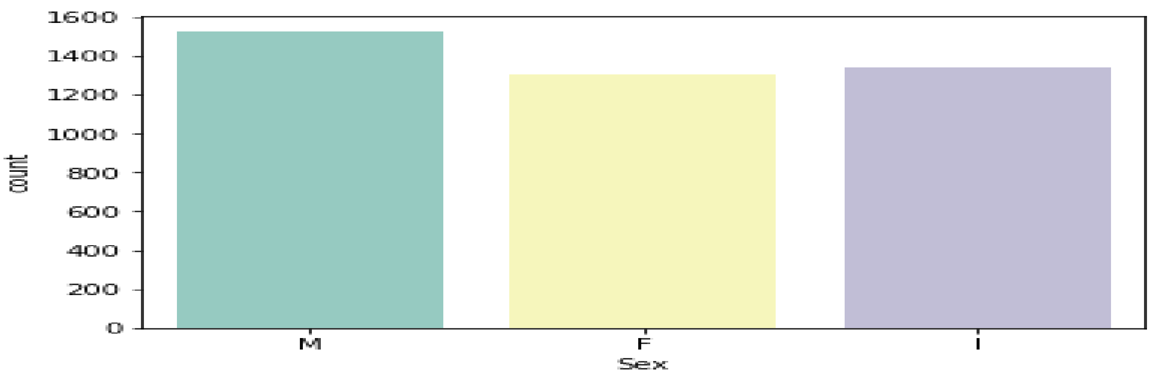


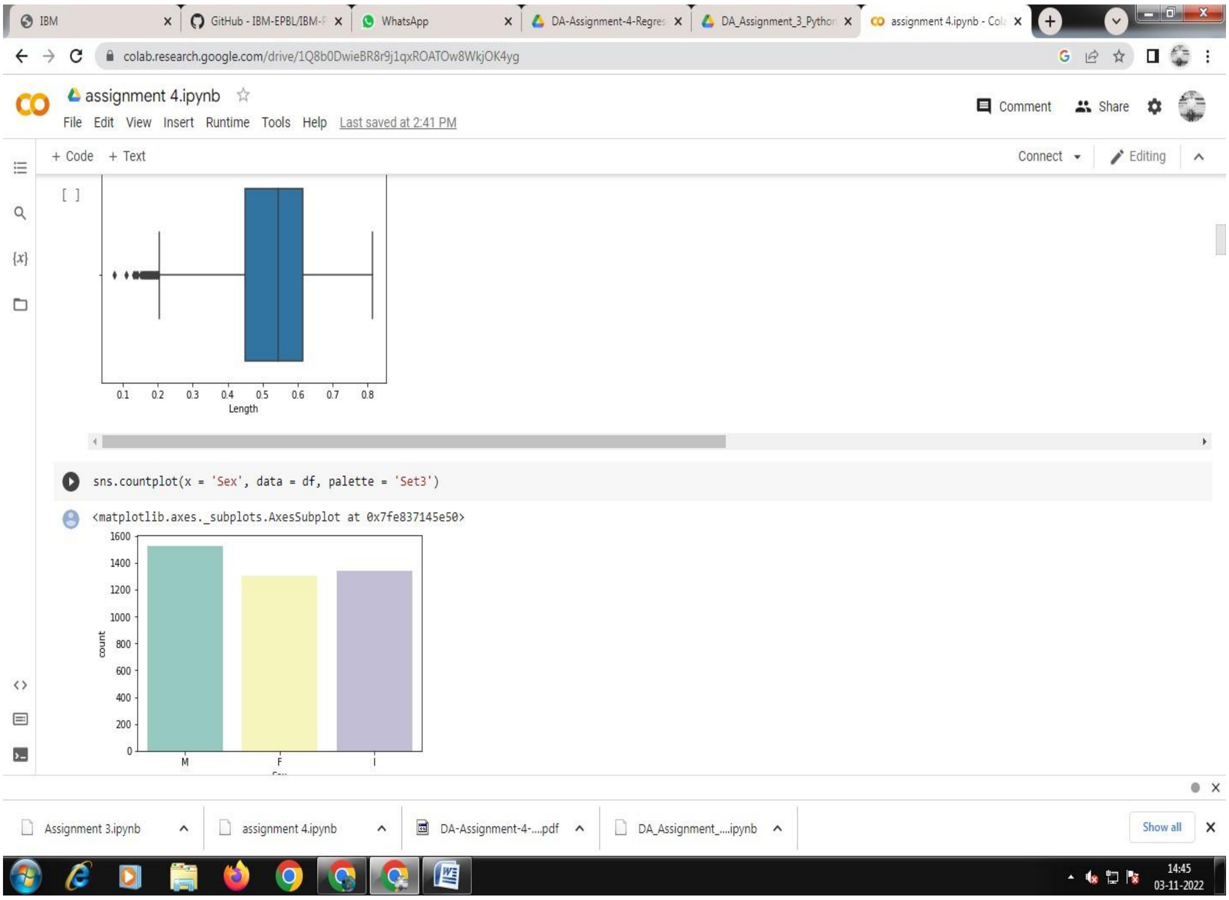


CODING:

```
sns.countplot(x = 'Sex', data = df, palette = 'Set3')
```

OUTPUT:



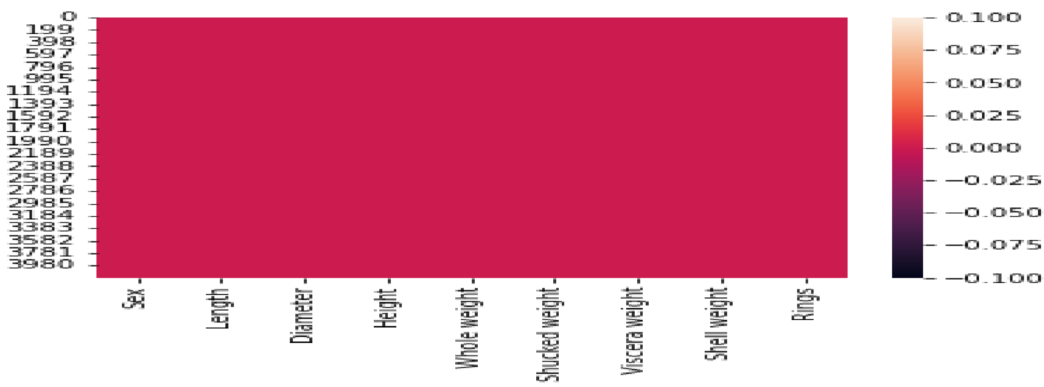


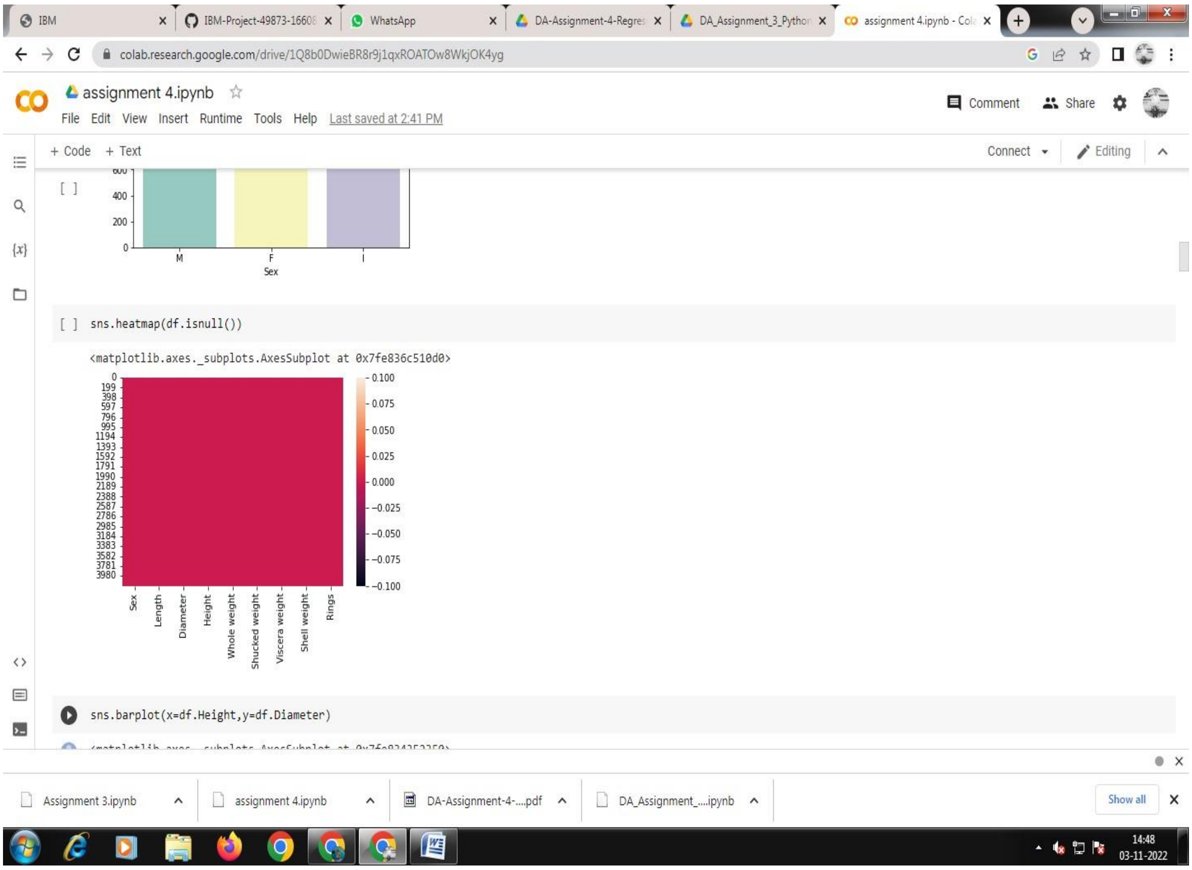
CODING:

sns.heatmap(df.isnull())

OUTPUT:

<matplotlib.axes._subplots.AxesSubplot at 0x7fe836c510d0>



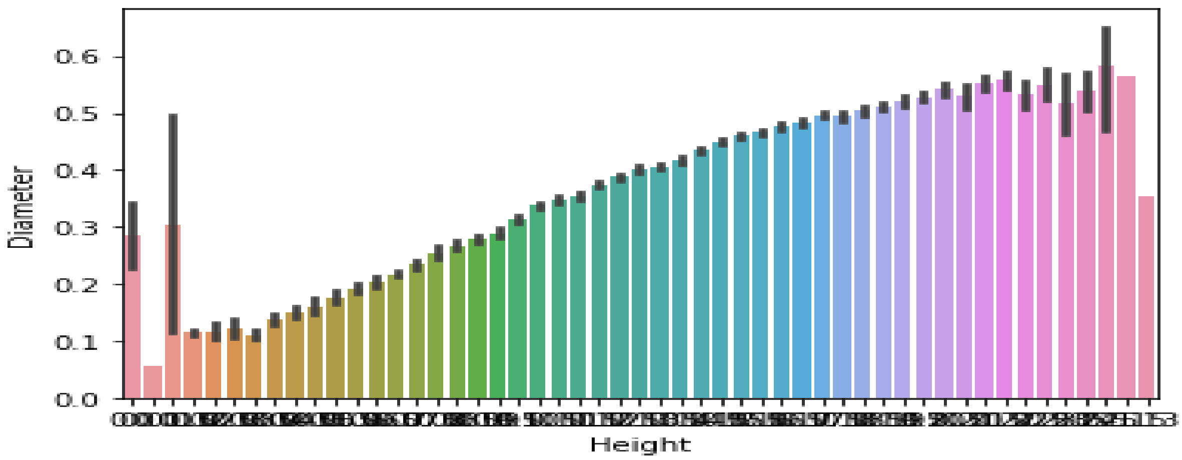


CODING:

sns.barplot(x=df.Height,y=df.Diameter)

OUTPUT:

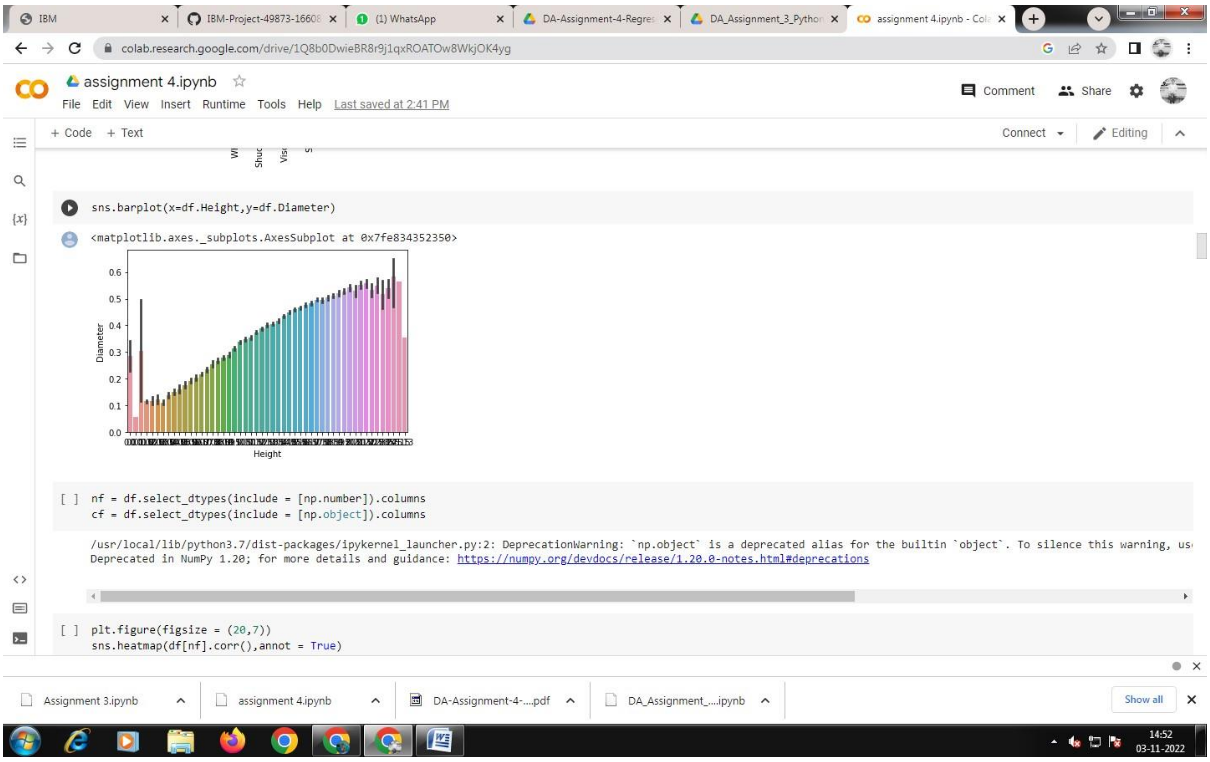
<matplotlib.axes._subplots.AxesSubplot at 0x7fe834352350>



CODING:

nf = df.select_dtypes(include = [np.number]).columns cf

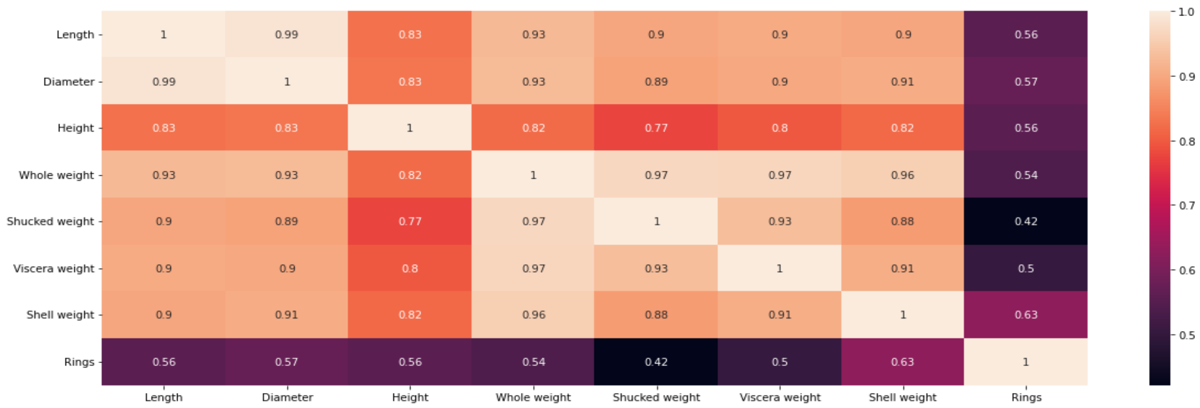
= df.select_dtypes(include = [np.object]).columns



plt.figure(figsize = (20,7)) sns.heatmap(df[nf].corr(),annot

= True)

OUTPUT:

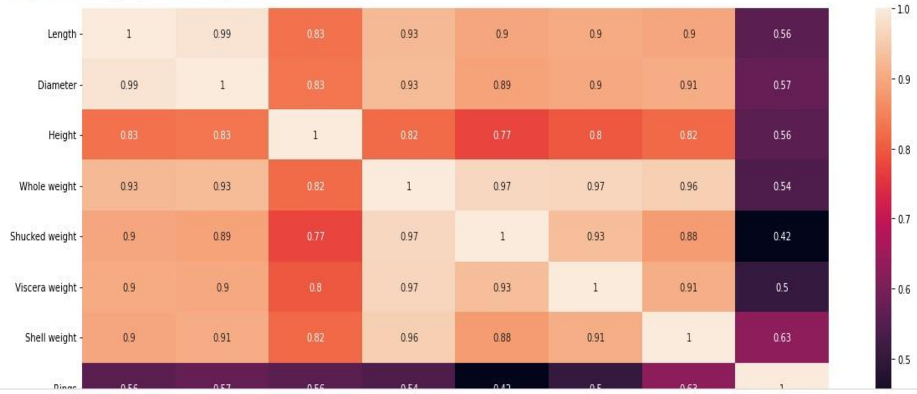


```
nf = df.select_dtypes(include = [np.number]).columns
cf = df.select_dtypes(include = [np.object]).columns
```

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warning, us
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations

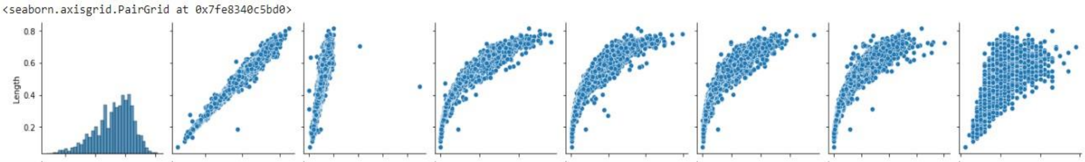
```
plt.figure(figsize = (20,7))
sns.heatmap(df[nf].corr(),annot = True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe834062250>




```
sns.pairplot(df)
```

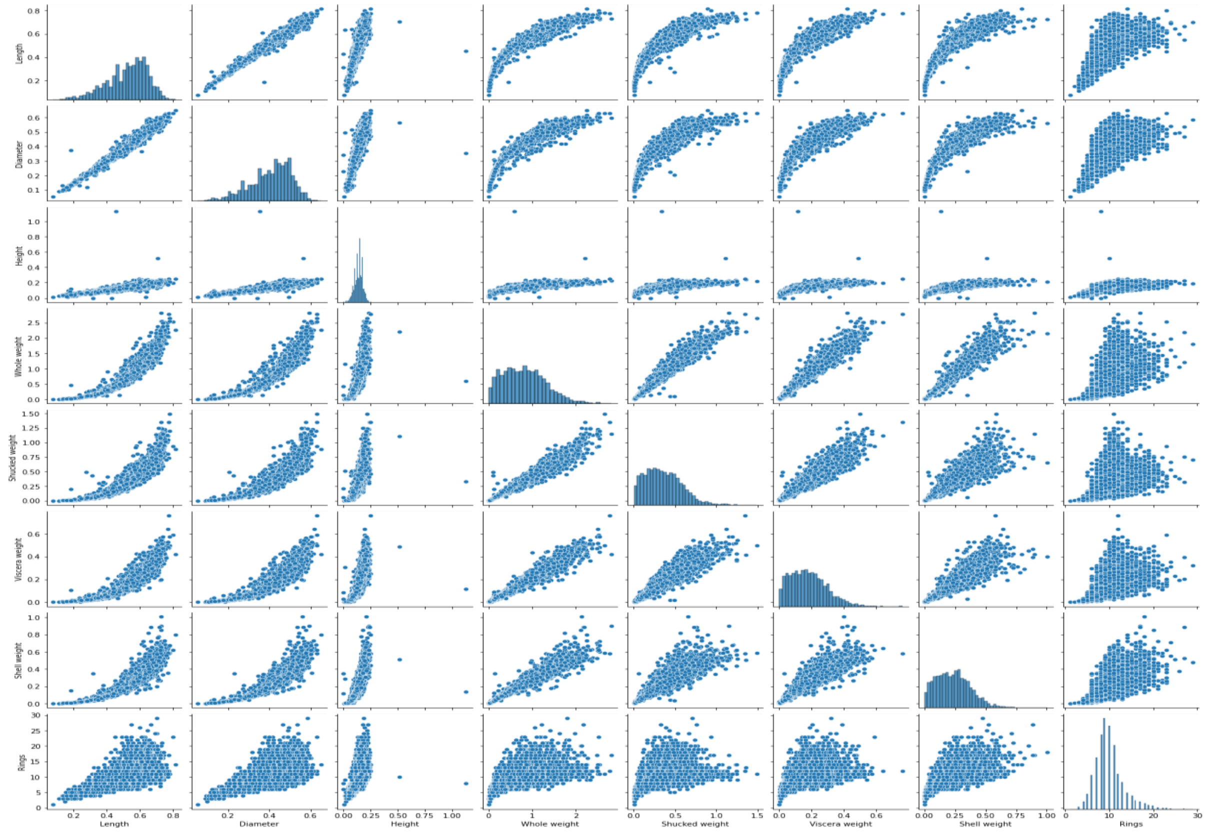
<seaborn.axisgrid.PairGrid at 0x7fe8340c5bd0>

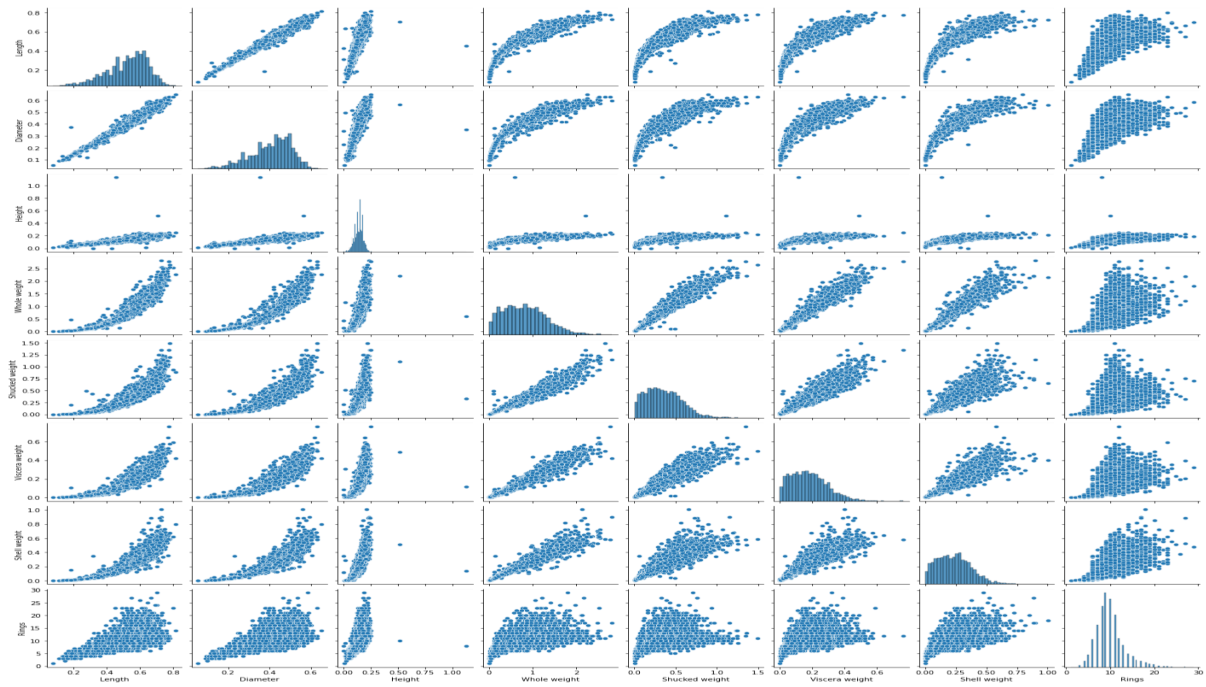


CODING:

sns.pairplot(df)

OUTPUT:





CODING:

df['Height'].describe()

OUTPUT:

count 4177.000000

mean 0.139516

std 0.041827 min

0.000000

25% 0.115000

50% 0.140000 75%

0.165000 max

1.130000

Name: Height, dtype: float64

CODING:

df['Height'].mean()

OUTPUT:

0.13951639932966242

CODING:

df.max()

OUTPUT:

Sex M

Length 0.815

Diameter 0.65

Height 1.13

Whole weight 2.8255

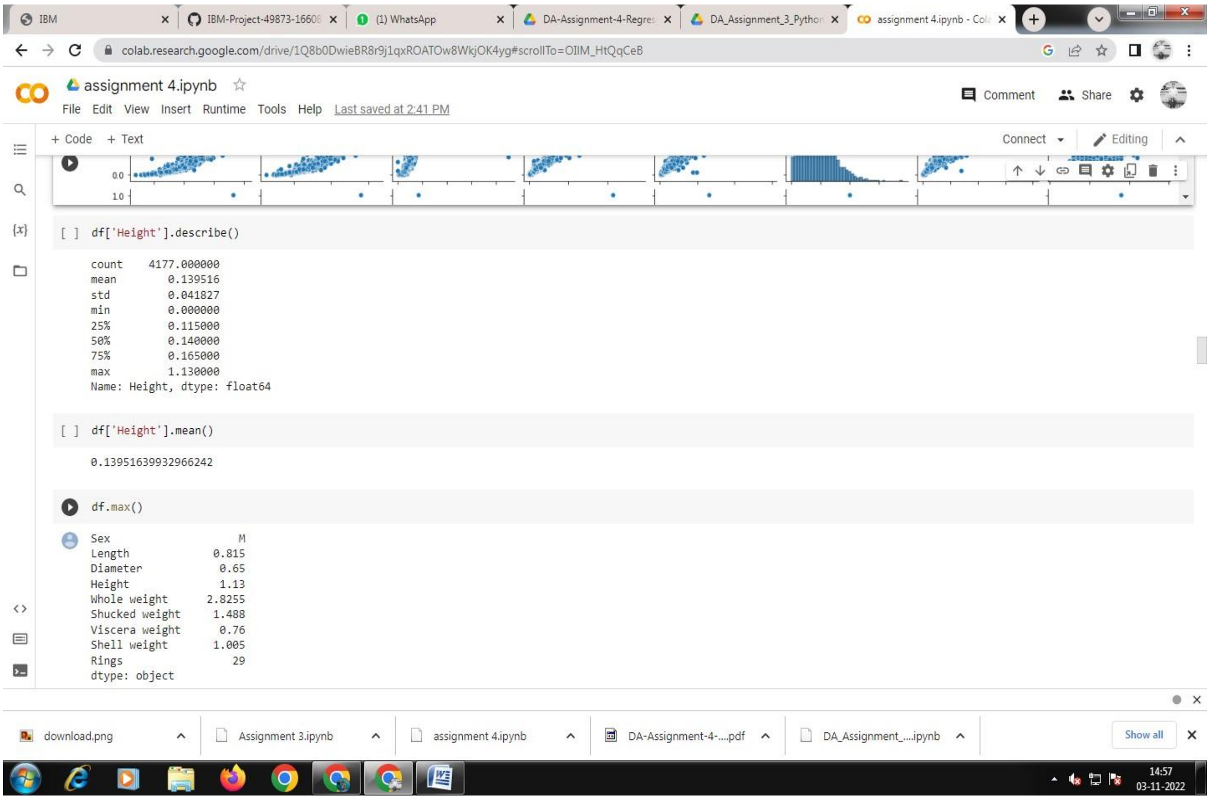
Shucked weight 1.488

Viscera weight 0.76

Shell weight 1.005

Rings 29

dtype: object



CODING:

df['Sex'].value_counts()

OUTPUT:

M 1528

I 1342

F 1307

Name: Sex, dtype: int64

CODING:

df[df.Height == 0]

OUTPUT:

Sex Length Diameter Height Whole weight Shucked weight Viscera weight Shell weight Rings

1257 I 0.430 0.34 0.0 0.428 0.2065 0.0860 0.1150 8 3996

I 0.315 0.23 0.0 0.134 0.0575 0.0285 0.3505 6

CODING:

df['Shucked weight'].kurtosis()

OUTPUT:

0.5951236783694207

CODING:

df['Diameter'].median()

OUTPUT:

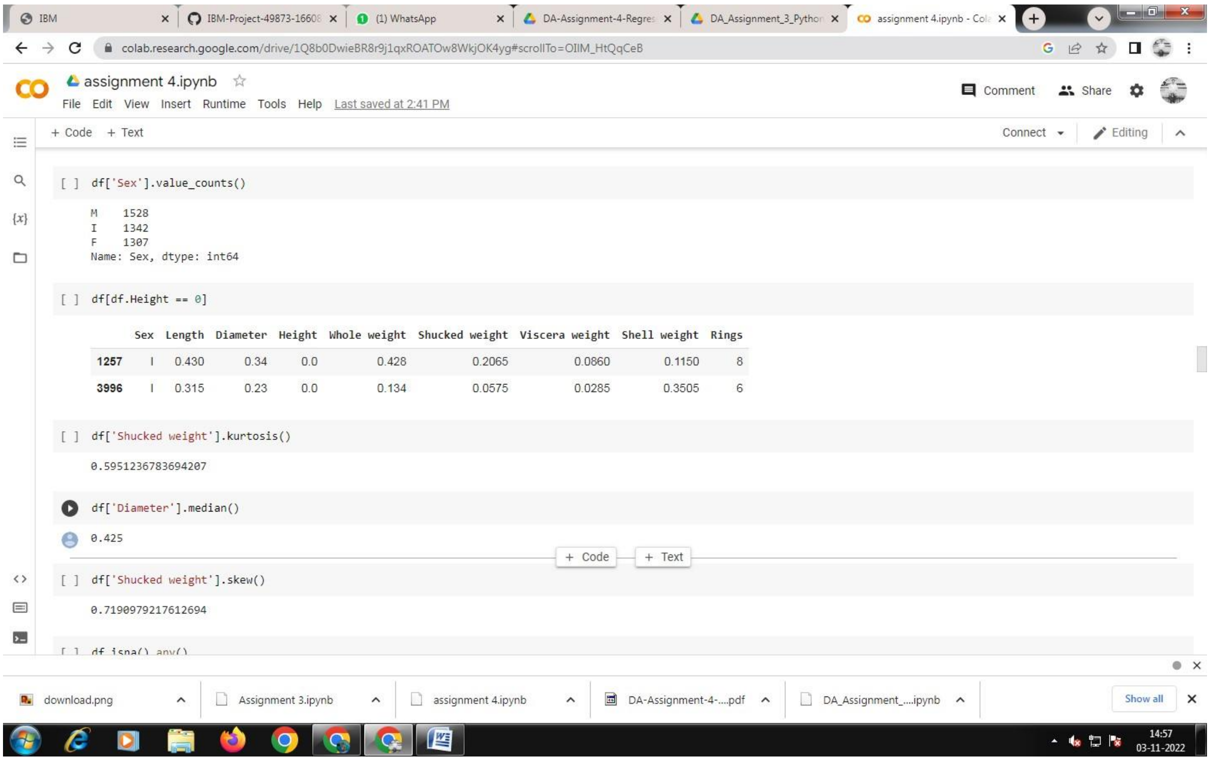
0.425

CODING:

df['Shucked weight'].skew()

OUTPUT:

0.7190979217612694



CODING:

```
df.isna().any()
```

OUTPUT:

```
Sex             False
Length          False
Diameter        False
Height          False
Whole weight    False
Shucked weight  False
Viscera weight  False
Shell weight    False
Rings           False
dtype: bool
```

CODING:

missing_values = df.isnull().sum().sort_values(ascending = False) percentage_missing_values

= (missing_values/len(df))*100

pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing values', '% Missing'
])

OUTPUT:

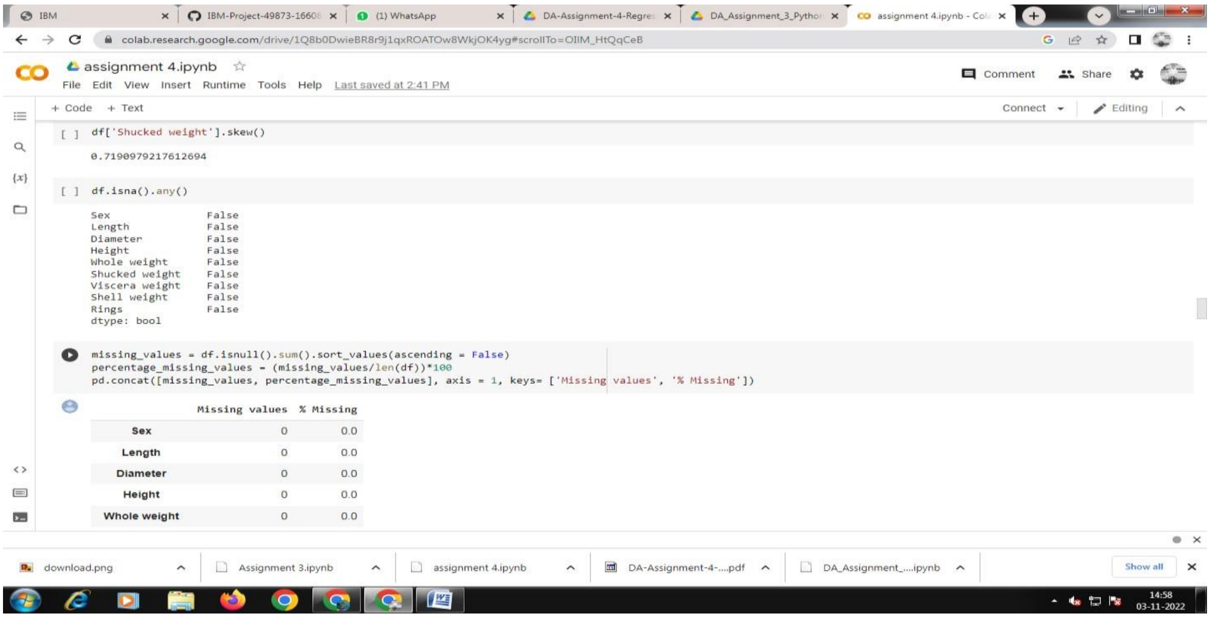
	Missing values	% Missing
Sex	0	0.0
Length	0	0.0
Diameter	0	0.0
Height	0	0.0
Whole weight	0	0.0
Shucked weight	0	0.0
Viscera weight	0	0.0
Shell weight	0	0.0
Rings	0	0.0

CODING:

q1=df.Rings.quantile(0.25)

q2=df.Rings.quantile(0.75)

iqr=q2-q1 print(iqr)



OUTPUT:

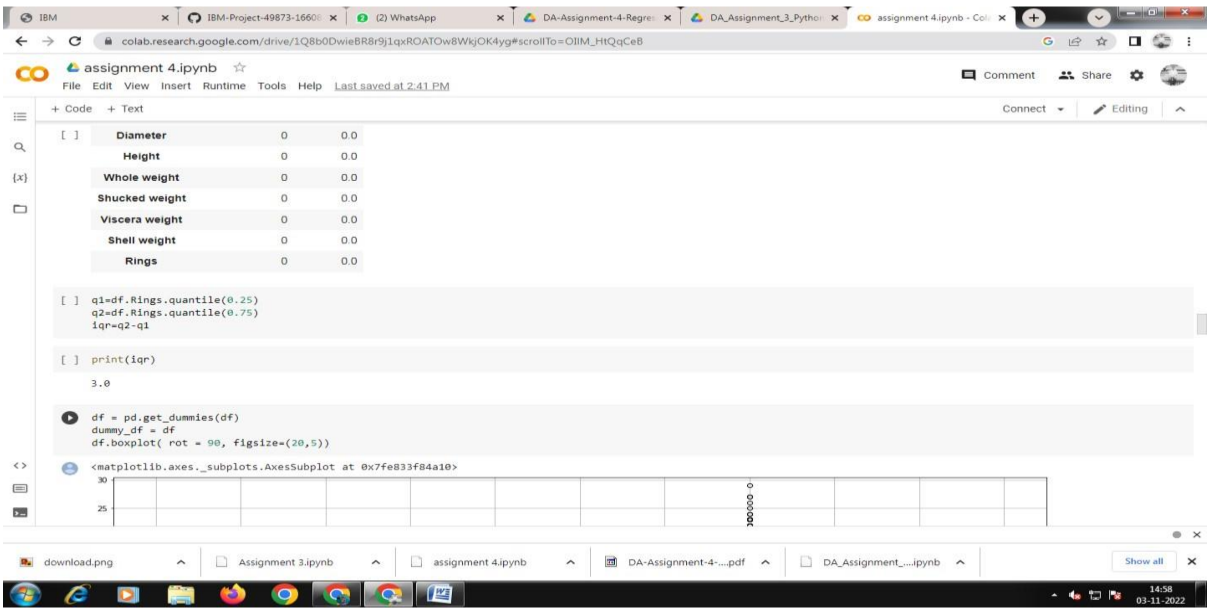
3.0

CODING:

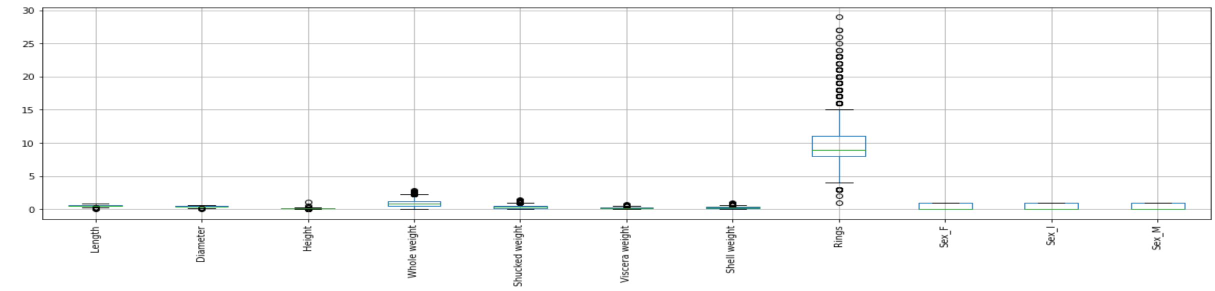
df = pd.get_dummies(df) dummy_df

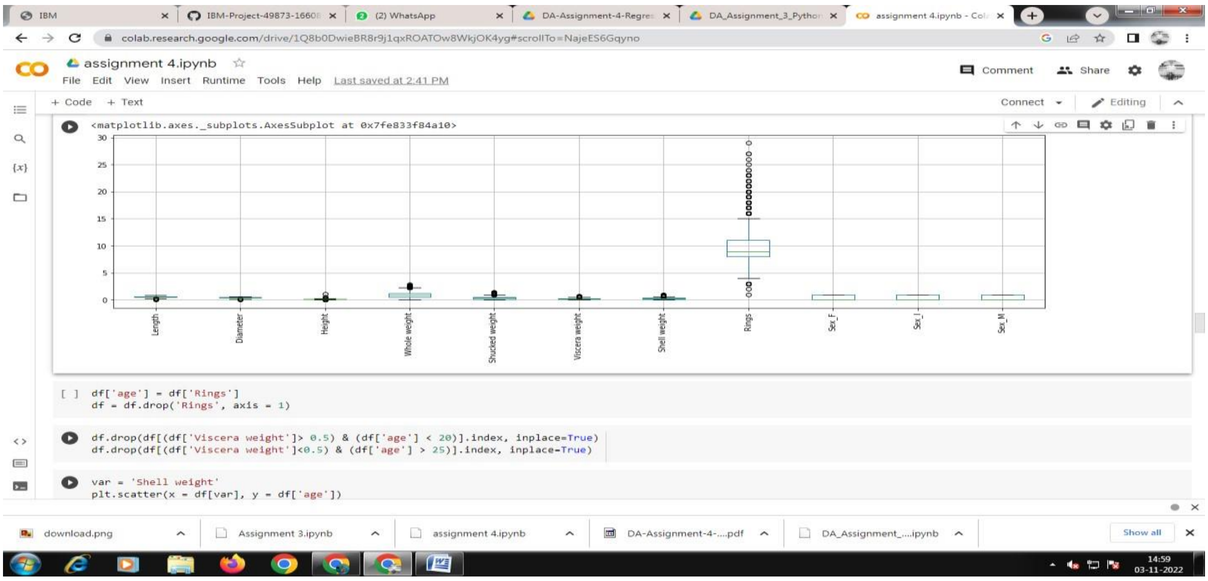
= df

df.boxplot(rot = 90, figsize=(20,5))



OUTPUT:

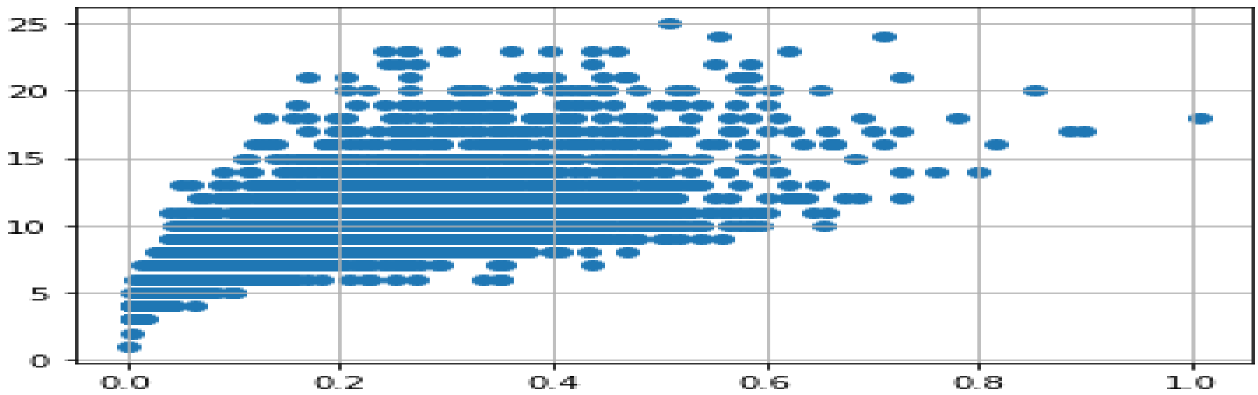


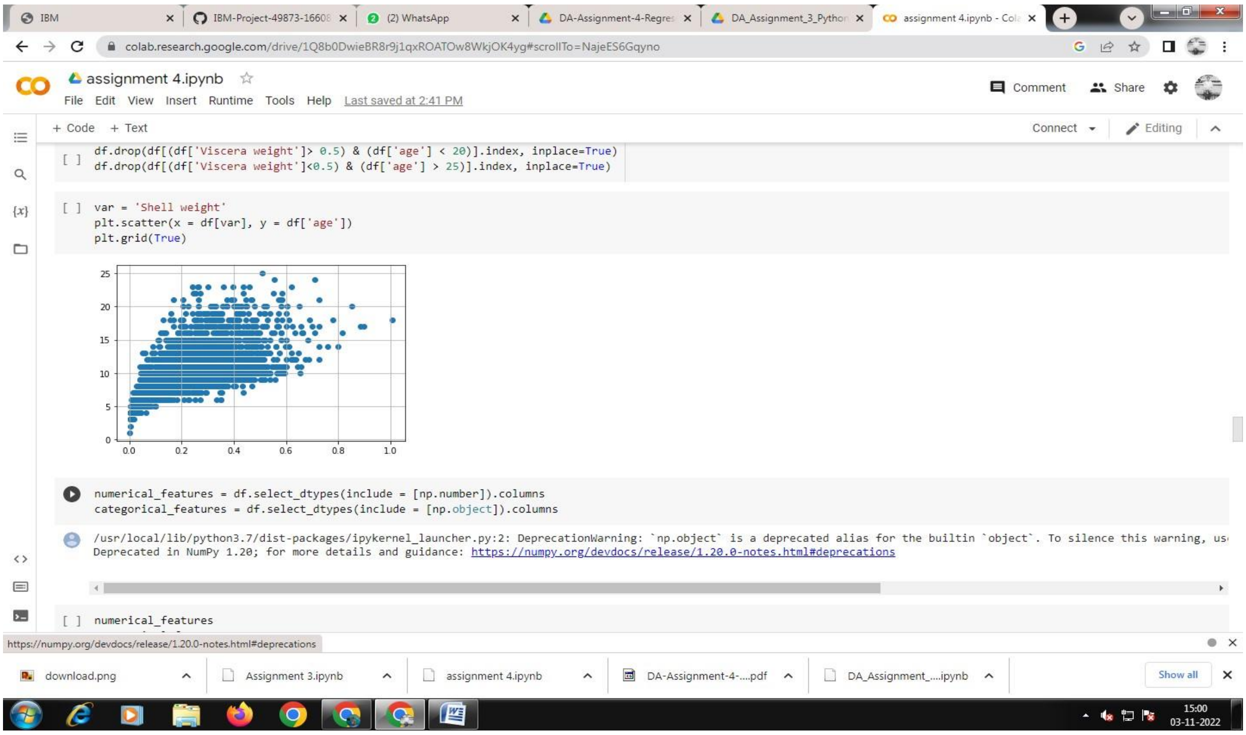


CODING:

df['age'] = df['Rings'] df = df.drop('Rings', axis = 1) df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True) df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True) var = 'Shell weight' plt.scatter(x = df[var], y = df['age']) plt.grid(True)

OUTPUT:





CODING:

numerical_features = df.select_dtypes(include = [np.number]).columns categorical_features

= df.select_dtypes(include = [np.object]).columns

OUTPUT:

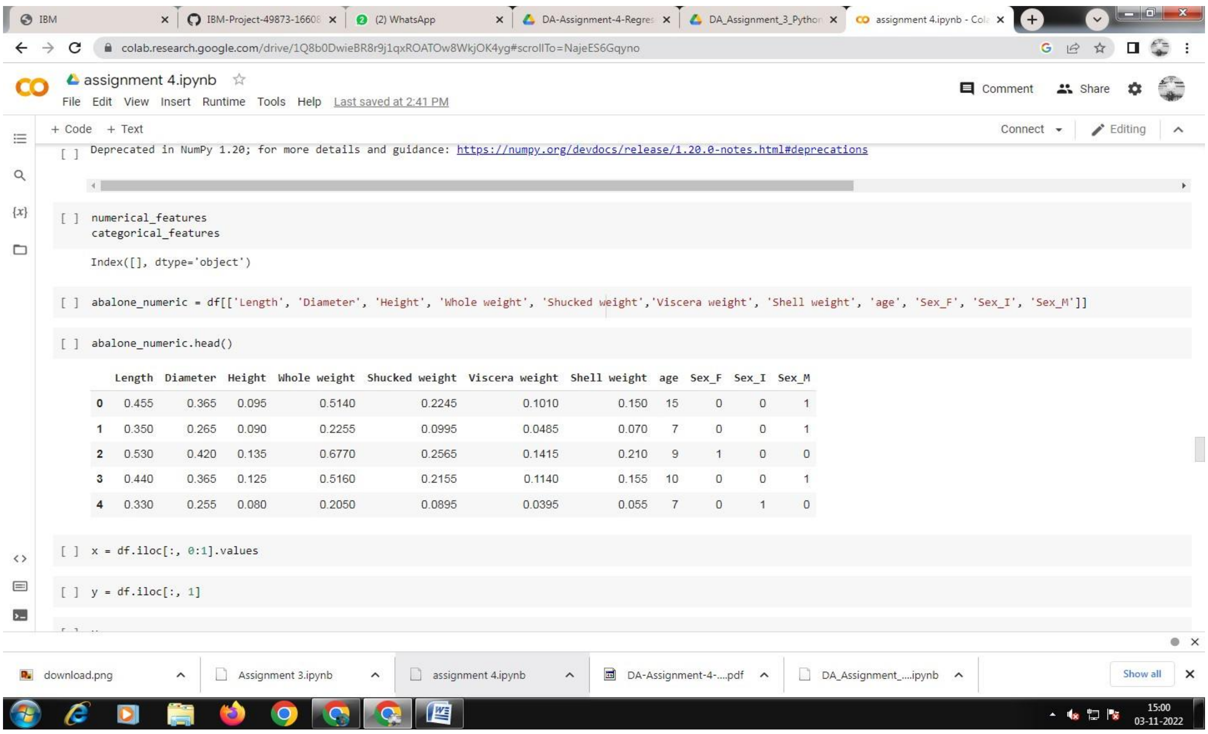
Index([], dtype='object')

CODING:

abalone_numeric = df[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight','Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M']] abalone_numeric.head()

OUTPUT:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age	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



CODING:

x = df.iloc[:, 0:1].values y

= df.iloc[:, 1]

OUTPUT:

0 0.365

1 0.265

2 0.420

3 0.365

4 0.255

4172 0.450

4173 0.440

4174 0.475

4175 0.485

4176 0.555

Name: Diameter, Length: 4150, dtype: float64

CODING:

```
print ("\n ORIGINAL VALUES: \n\n", x,y)
```

OUTPUT:

ORIGINAL VALUES:

[[0.455]

[0.35]

[0.53]

[0.6]

[0.625]

[0.71]] 0 0.365

1 0.265

2 0.420

3 0.365

4 0.255

4172 0.450

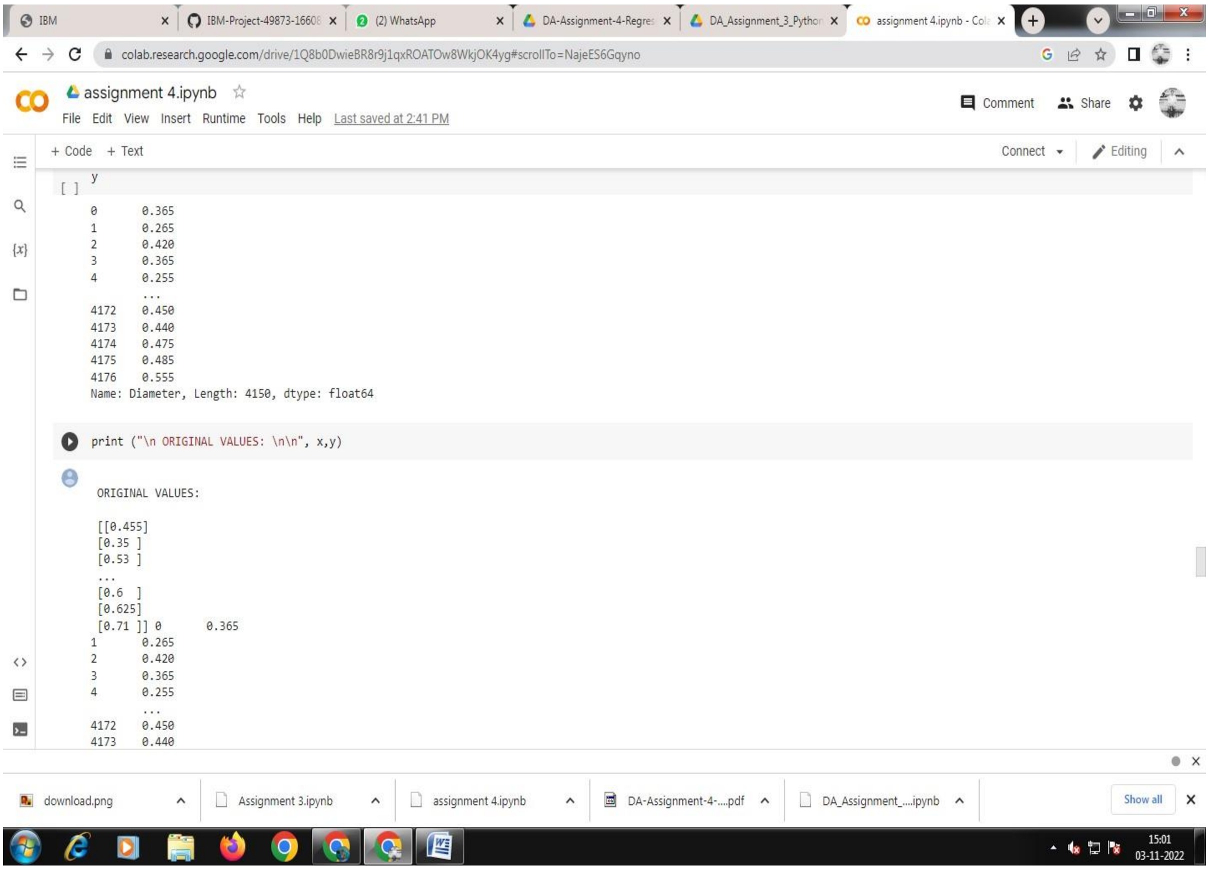
4173 0.440

4174 0.475

4175 0.485

4176 0.555

Name: Diameter, Length: 4150, dtype: float64



CODING:

from sklearn import preprocessing min_max_scaler =

preprocessing.MinMaxScaler(feature_range =(0, 1)) new_y=

min_max_scaler.fit_transform(x,y) print ("\n VALUES AFTER MIN

MAX SCALING: \n\n", new_y)

OUTPUT:

VALUES AFTER MIN MAX SCALING:

[[0.51351351]

[0.37162162]

[0.61486486]

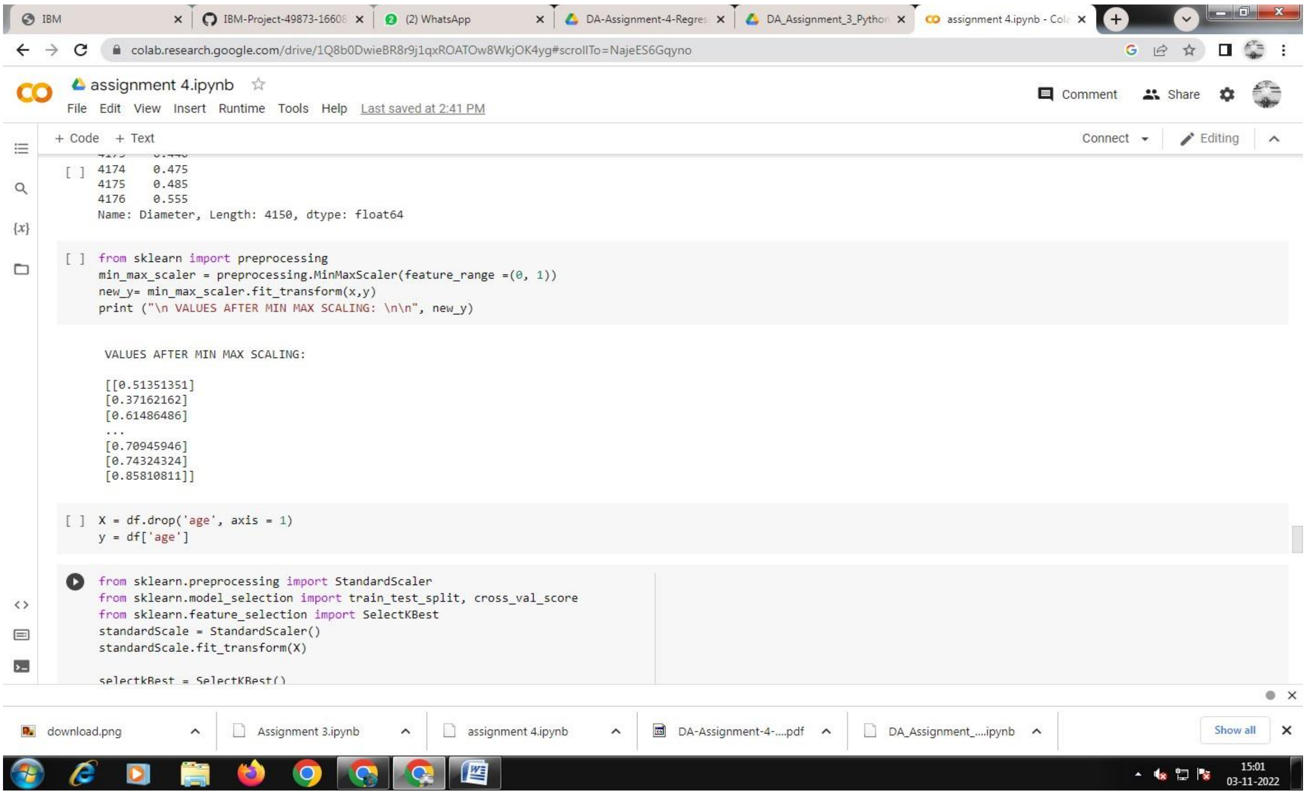
[0.70945946]

[0.74324324]

[0.85810811]]

CODING:

```
X = df.drop('age', axis = 1) y = df['age'] from sklearn.preprocessing

import StandardScaler from sklearn.model_selection import

train_test_split, cross_val_score from sklearn.feature_selection

import SelectKBest standardScale = StandardScaler()

standardScale.fit_transform(X)


selectkBest = SelectKBest()

X_new = selectkBest.fit_transform(X, y)


X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)

X_train
```



OUTPUT:

array([[0.255, 0.185, 0.06 , ..., 0.   , 1.   , 0.   ],

[0.655, 0.505, 0.165, ..., 1.   , 0.   , 0.   ],

[0.355, 0.26 , 0.09 , ..., 0.   , 1.   , 0.   ],

...,

[0.635, 0.495, 0.015, ..., 1.   , 0.   , 0.   ],

[0.335, 0.245, 0.09 , ..., 0.   , 1.   , 0.   ],

[0.65 , 0.5 , 0.17 , ..., 1.   , 0.   , 0.   ]])

CODING:

y_train

OUTPUT:

813   5

3150   10

2485   8

2307 16

844 8

1298 10

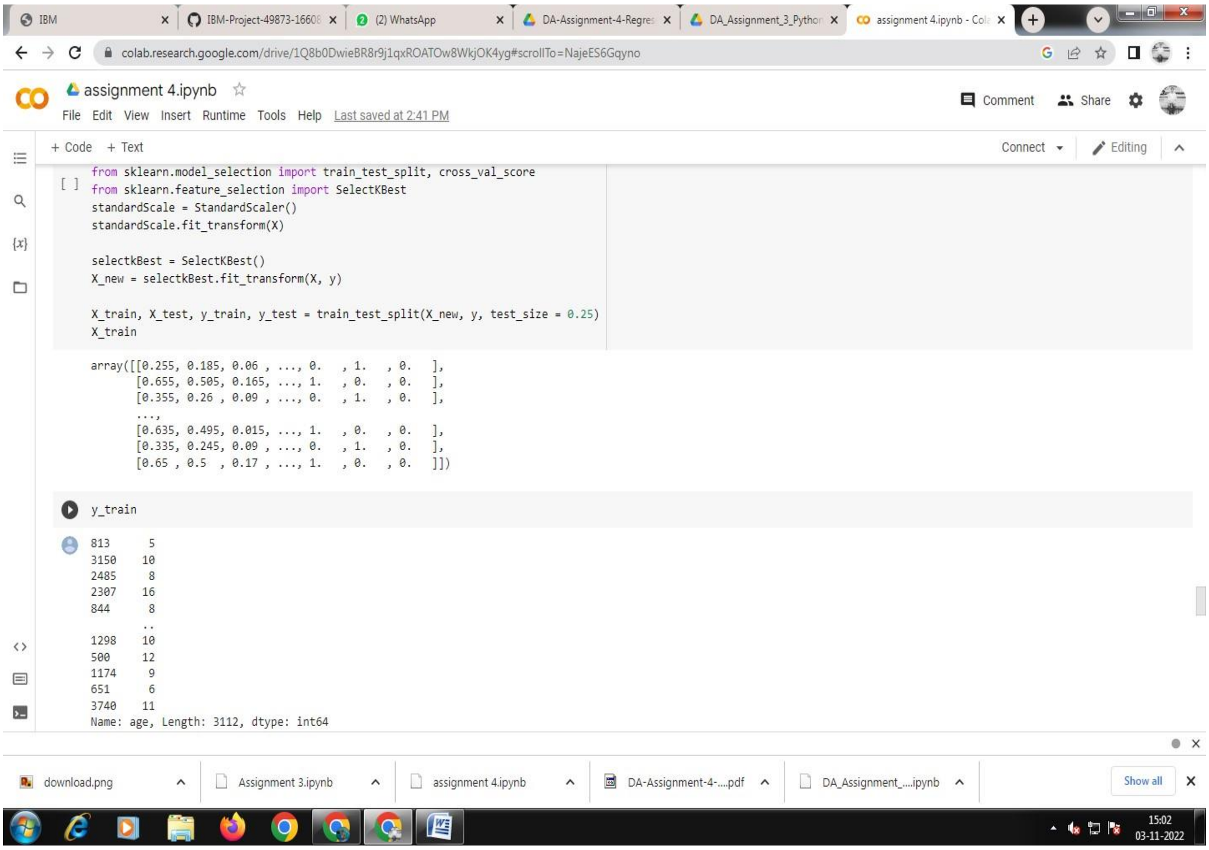
500 12

1174 9

651 6

3740 11

Name: age, Length: 3112, dtype: int64



CODING:

from sklearn import linear_model as lm from

sklearn.linear_model import LinearRegression

model=lm.LinearRegression()

results=model.fit(X_train,y_train) ccuracy =

model.score(X_train, y_train) print('Accuracy of the

model:', accuracy)

OUTPUT:

Accuracy of the model: 0.5345933867890345

CODING:

lm = LinearRegression()

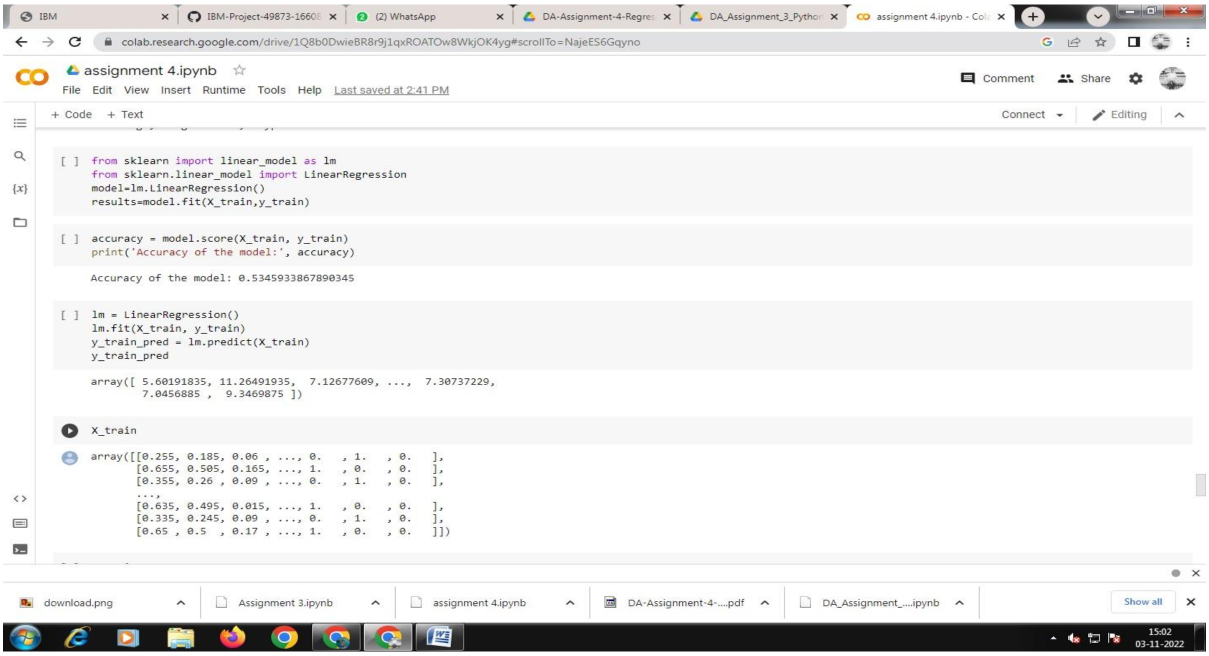
lm.fit(X_train, y_train) y_train_pred

= lm.predict(X_train) y_train_pred

OUTPUT:

array([5.60191835, 11.26491935, 7.12677609, ..., 7.30737229,

      7.0456885 , 9.3469875])



CODING:

X_train

OUTPUT:

array([[0.255, 0.185, 0.06 , ..., 0.   , 1.   , 0.   ],

[0.655, 0.505, 0.165, ..., 1.   , 0.   , 0.   ],

[0.355, 0.26 , 0.09 , ..., 0.   , 1.   , 0.   ],

...,

[0.635, 0.495, 0.015, ..., 1.   , 0.   , 0.   ],

[0.335, 0.245, 0.09 , ..., 0.   , 1.   , 0.   ],

[0.65 , 0.5   , 0.17 , ..., 1.   , 0.   , 0.   ]])

CODING:

y_train from sklearn.metrics import mean_absolute_error,

mean_squared_error s = mean_squared_error(y_train, y_train_pred)

print('Mean Squared error :%2f'%s) **OUTPUT:**

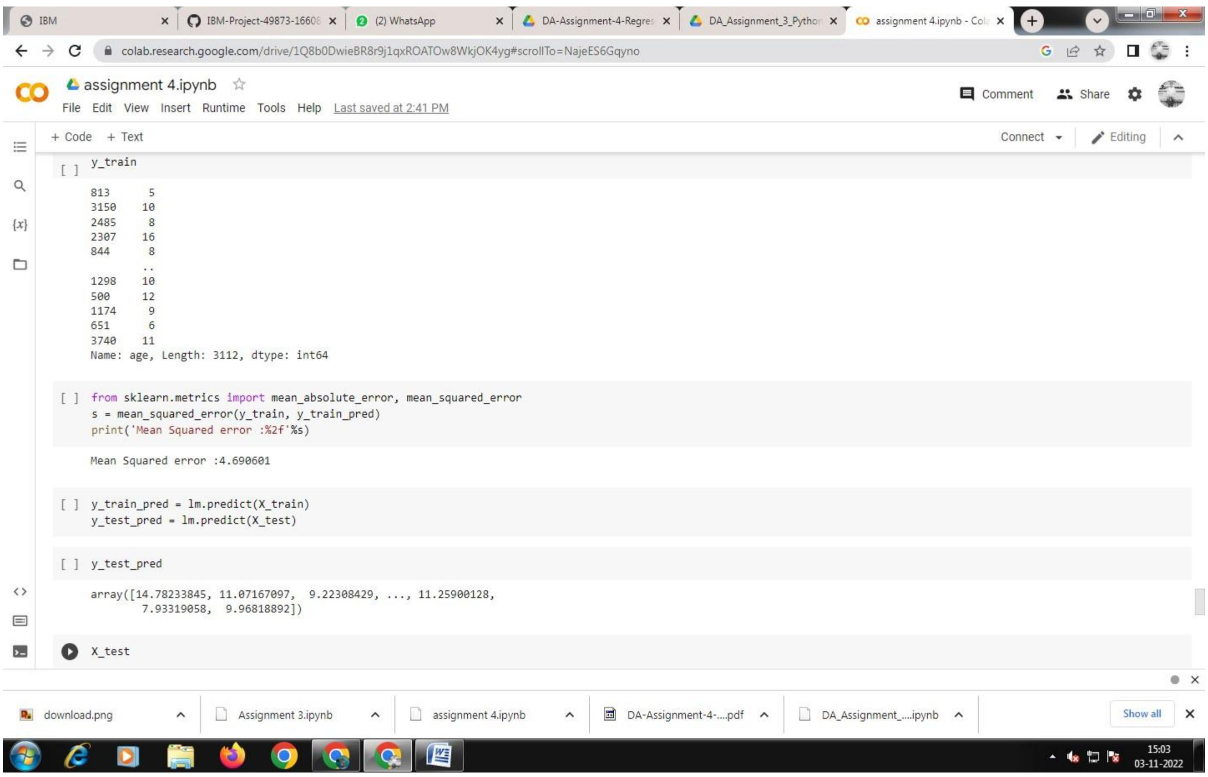
Mean Squared error :4.690601 **CODING:**

y_train_pred = lm.predict(X_train)

y_test_pred = lm.predict(X_test) y_test_pred

OUTPUT:

array([14.78233845, 11.07167097, 9.22308429, ..., 11.25900128, 7.93319058, 9.96818892])



CODING:

X_test

OUTPUT:

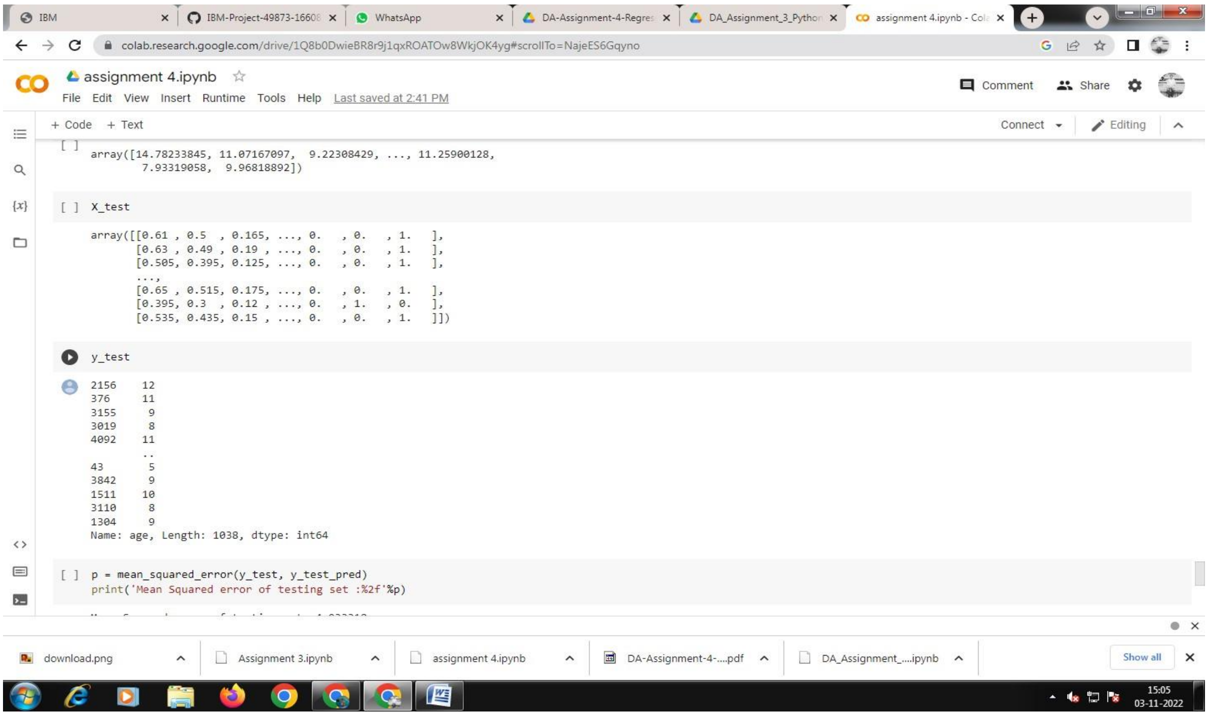
```
array([[0.61 , 0.5  , 0.165, ..., 0.   , 0.   , 1.   ],
[0.63 , 0.49 , 0.19 , ..., 0.   , 0.   , 1.   ],
    [0.505, 0.395, 0.125, ..., 0.   , 0.   , 1.   ],
  [0.65 , 0.515, 0.175, ..., 0.   , 0.   , 1.   ],
    [0.395, 0.3  , 0.12 , ..., 0.   , 1.   , 0.   ],
    [0.535, 0.435, 0.15 , ..., 0.   , 0.   , 1.   ]])
```

CODING:

y_test

OUTPUT:

```
2156    12
376     11
3155     9
3019     8
4092    11
43       5
3842     9
1511    10
3110     8
1304     9
Name: age, Length: 1038, dtype: int64
```



CODING:

p = mean_squared_error(y_test, y_test_pred) print('Mean

Squared error of testing set :%2f'%p)

OUTPUT:

Mean Squared error of testing set :4.933318

CODING:

from sklearn.metrics import r2_score s

= r2_score(y_train, y_train_pred)

print('R2 Score of training set:%.2f'%s)

OUTPUT:

R2 Score of training set:0.53

CODING:

from sklearn.metrics import r2_score p

= r2_score(y_test, y_test_pred)

print('R2 Score of testing set:%.2f'%p)

OUTPUT:

R2 Score of testing set:0.52