

# IBM ASSIGNMENT 4

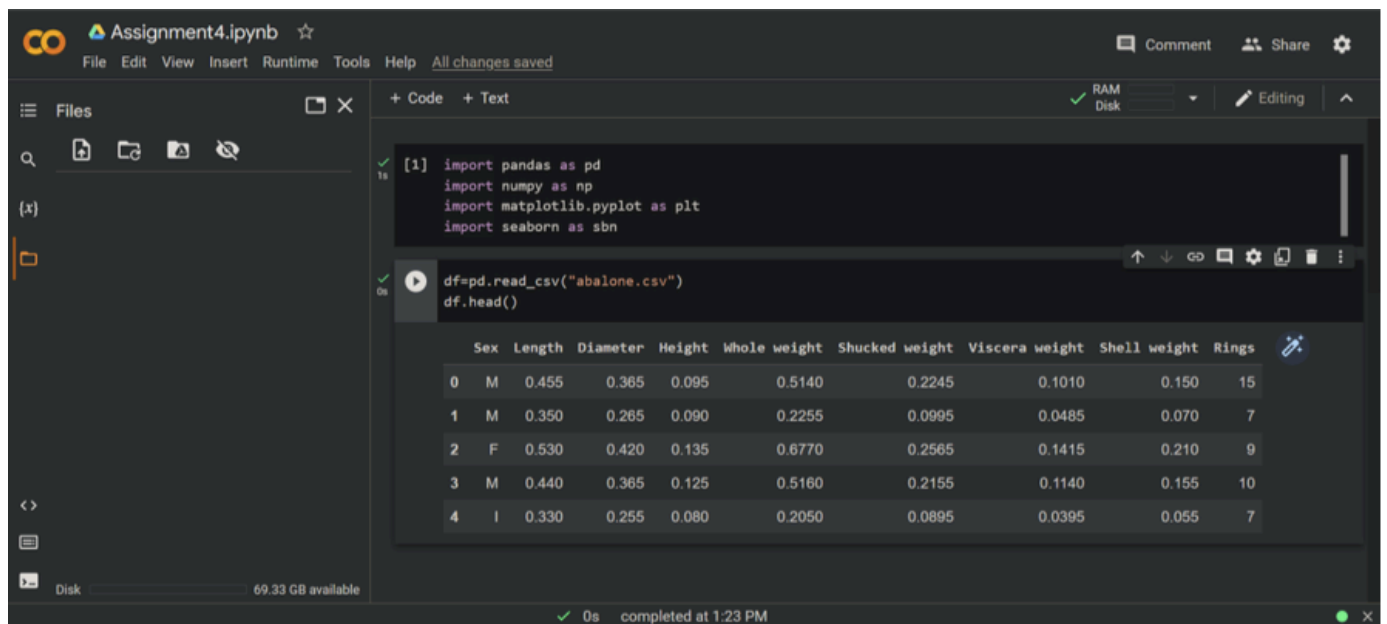
Name: Shamyukta Bose

Register No: 711319CS146

## CHALLENGE:

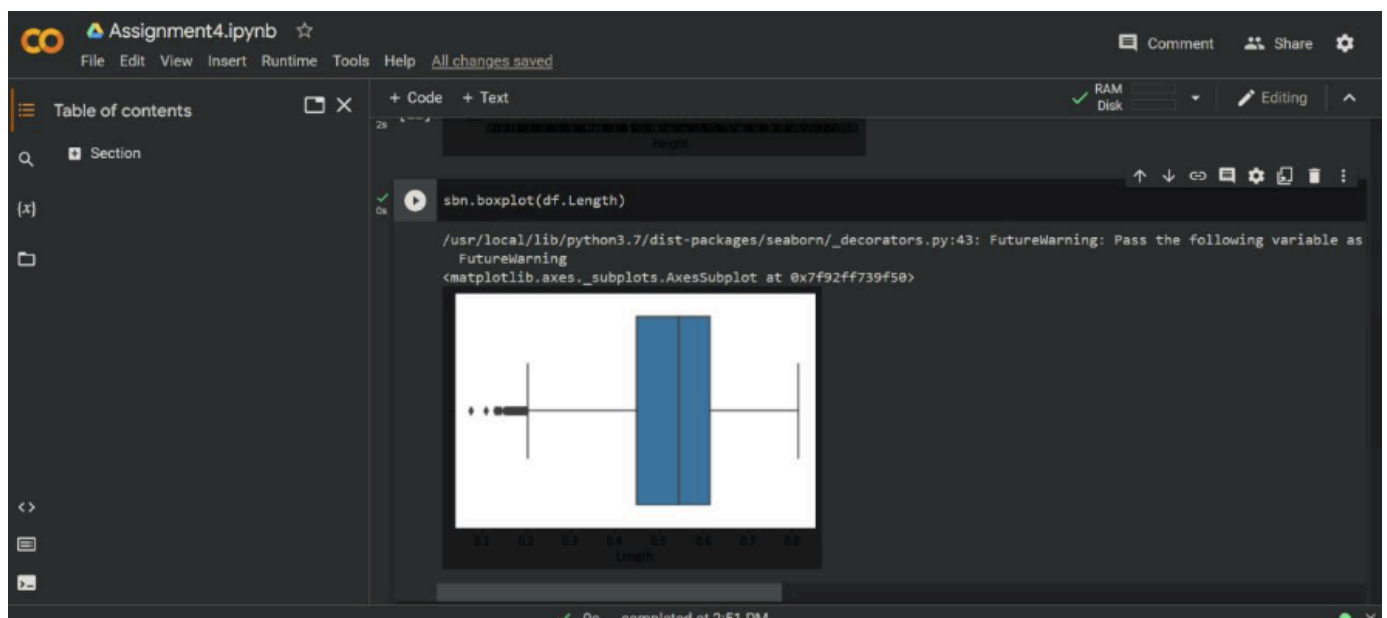
Abalone Age Prediction

## LOADING THE DATASET:

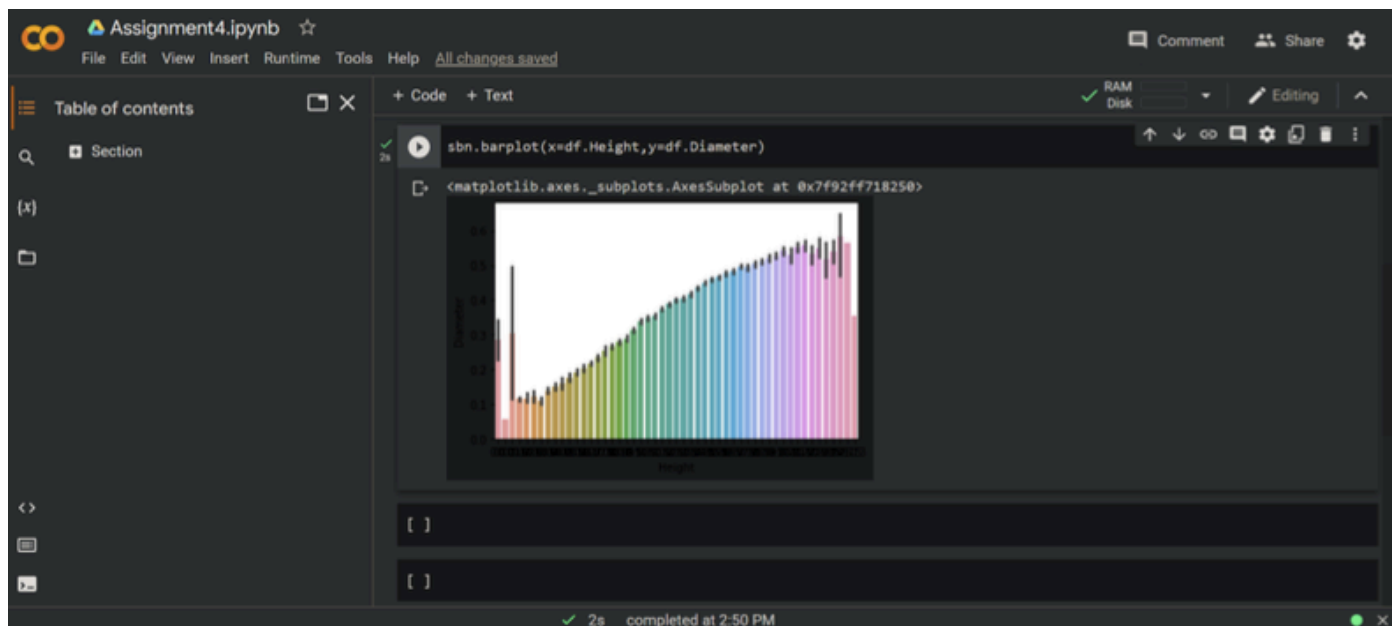


## VISUALIZATIONS:

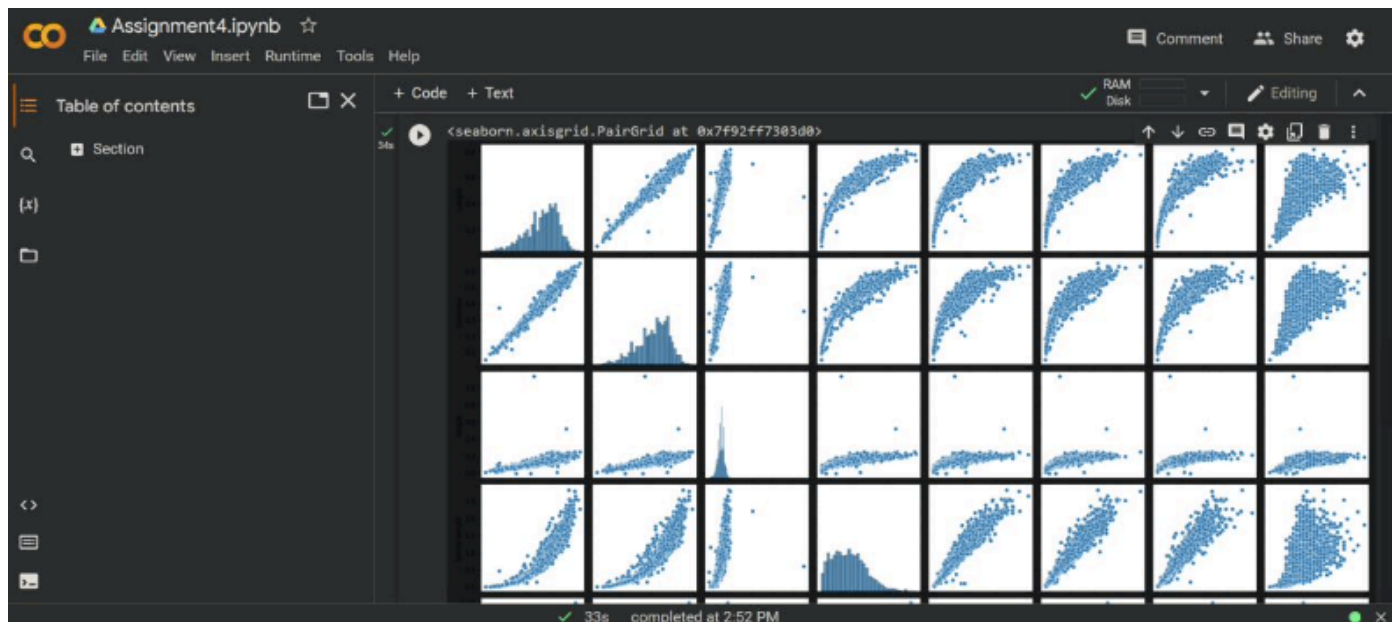
Univariate Analysis



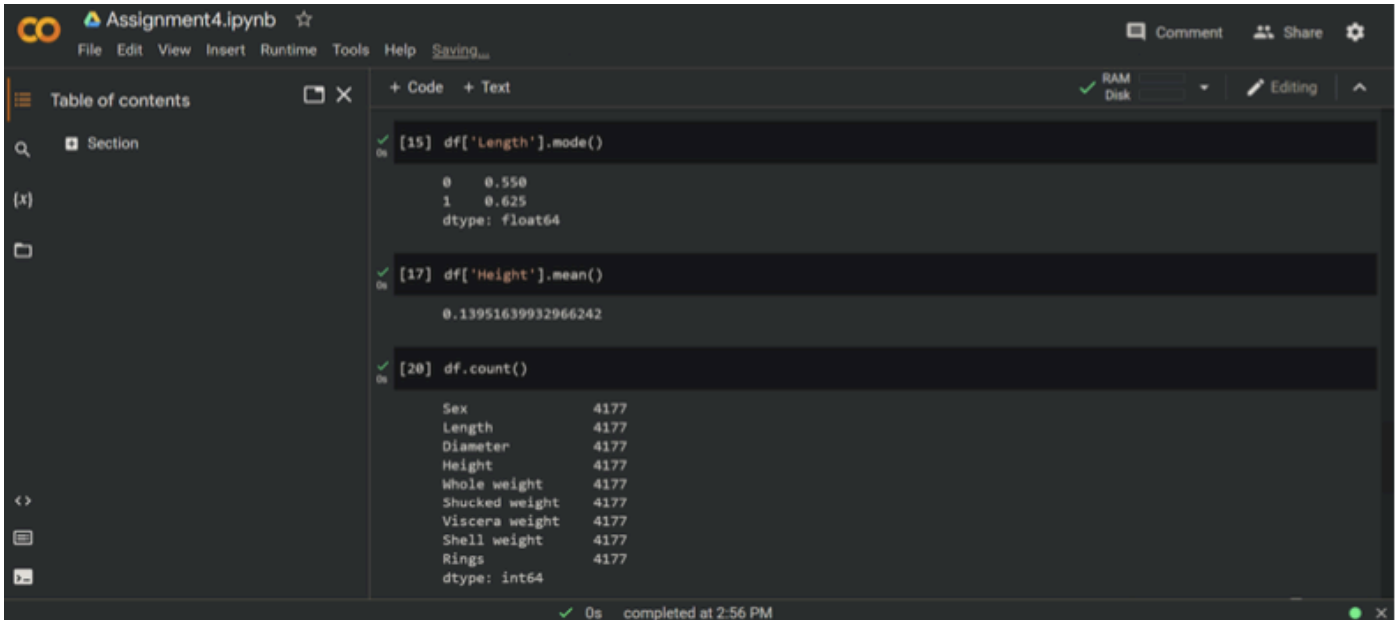
## Bi-Variate Analysis



## Multi-Variate Analysis



## Perform descriptive Analysis on datasets



This screenshot shows a Jupyter Notebook interface with the file 'Assignment4.ipynb'. The left sidebar contains a 'Table of contents' and a search bar. The main area displays three code cells, each with a green checkmark indicating successful execution. The first cell calculates the mode of the 'Length' variable, the second calculates the mean of the 'Height' variable, and the third displays the count of each variable in the dataset.

```
[15] df['Length'].mode()

0    0.550
1    0.625
dtype: float64

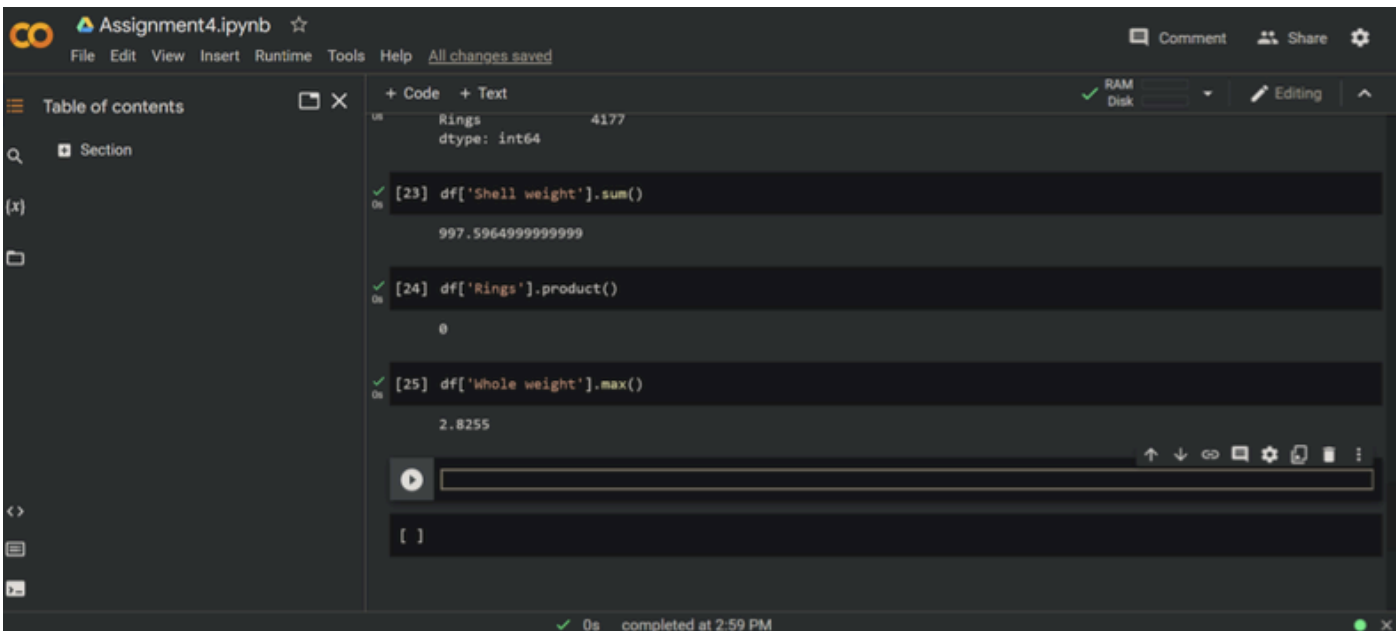
[17] df['Height'].mean()

0.13951639932966242

[20] df.count()

Sex          4177
Length       4177
Diameter     4177
Height       4177
Whole weight 4177
Shucked weight 4177
Viscera weight 4177
Shell weight 4177
Rings        4177
dtype: int64
```

The status bar at the bottom indicates '0s completed at 2:56 PM'.



This screenshot shows the continuation of the Jupyter Notebook. The left sidebar remains the same. The main area displays three more code cells, each with a green checkmark. The first cell calculates the sum of 'Shell weight', the second calculates the product of 'Rings', and the third calculates the maximum of 'Whole weight'. Below the last cell, there is an empty input field and a list representation '[ ]'.

```
[23] df['Shell weight'].sum()

997.5964999999999

[24] df['Rings'].product()

0

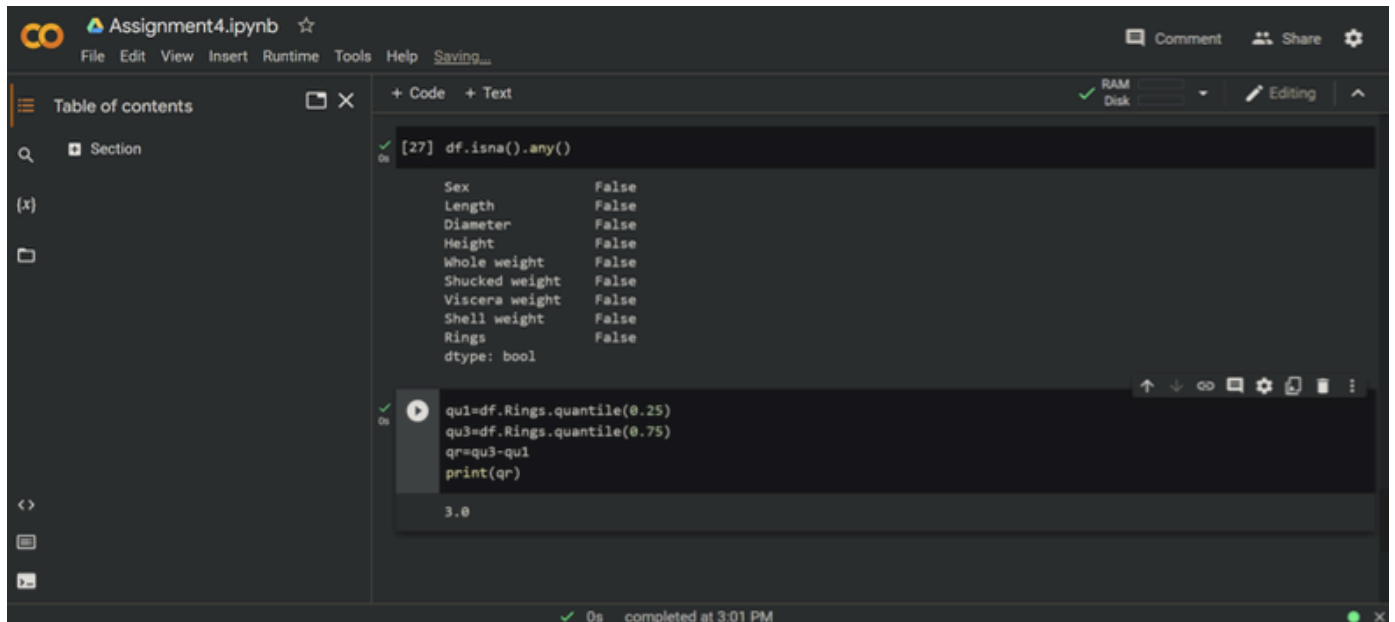
[25] df['Whole weight'].max()

2.8255

[ ]
```

The status bar at the bottom indicates '0s completed at 2:59 PM'.

Checking for missing values and deal with them , Finding the outliers and replace them outliers



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell checks for missing values in the 'Rings' column. The second cell calculates the interquartile range (IQR) for the 'Rings' column to identify outliers.

```
[27] df.isna().any()
```

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

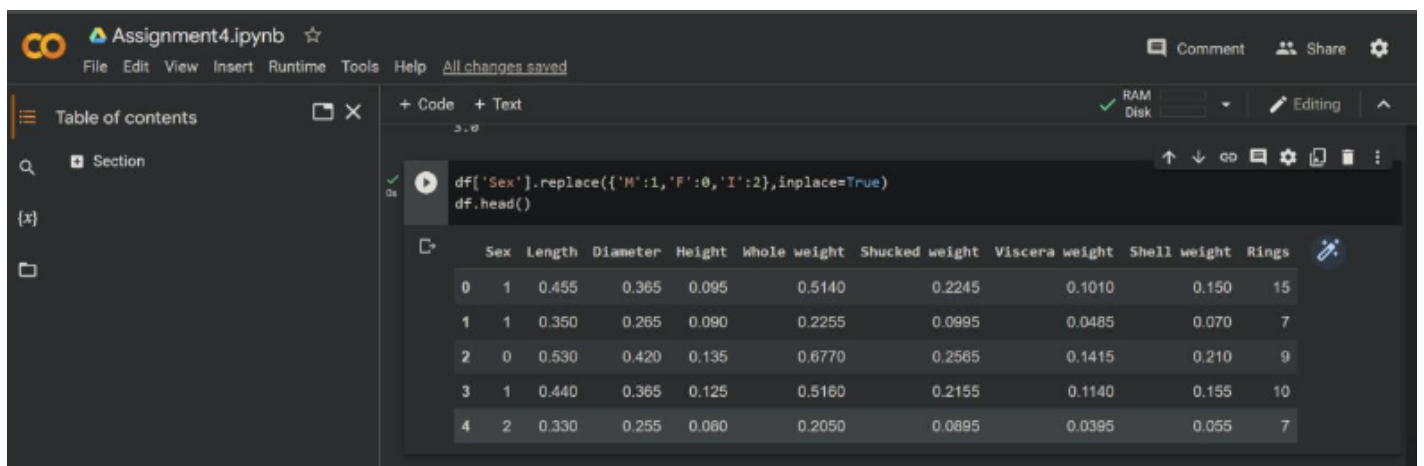
```
dtype: bool
```

```
qu1=df.Rings.quantile(0.25)
qu3=df.Rings.quantile(0.75)
qr=qu3-qu1
print(qr)
```

3.0

completed at 3:01 PM

Check for categorical columns and perform encoding

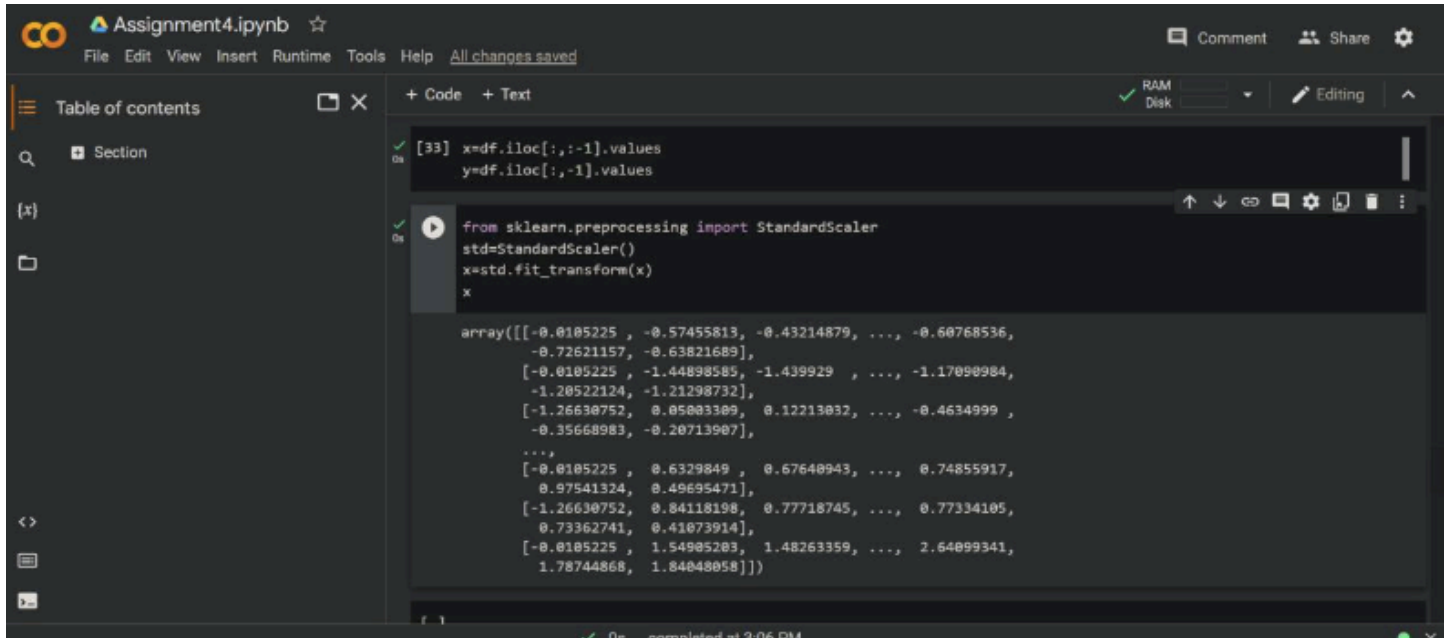


The screenshot shows a Jupyter Notebook interface with a code cell that performs one-hot encoding on the 'Sex' column. Below the code cell, the first five rows of the resulting DataFrame are displayed.

```
df['Sex'].replace({'M':1,'F':0,'I':2},inplace=True)
df.head()
```

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

Split the data into dependent and independent variables, Scale the independent variable



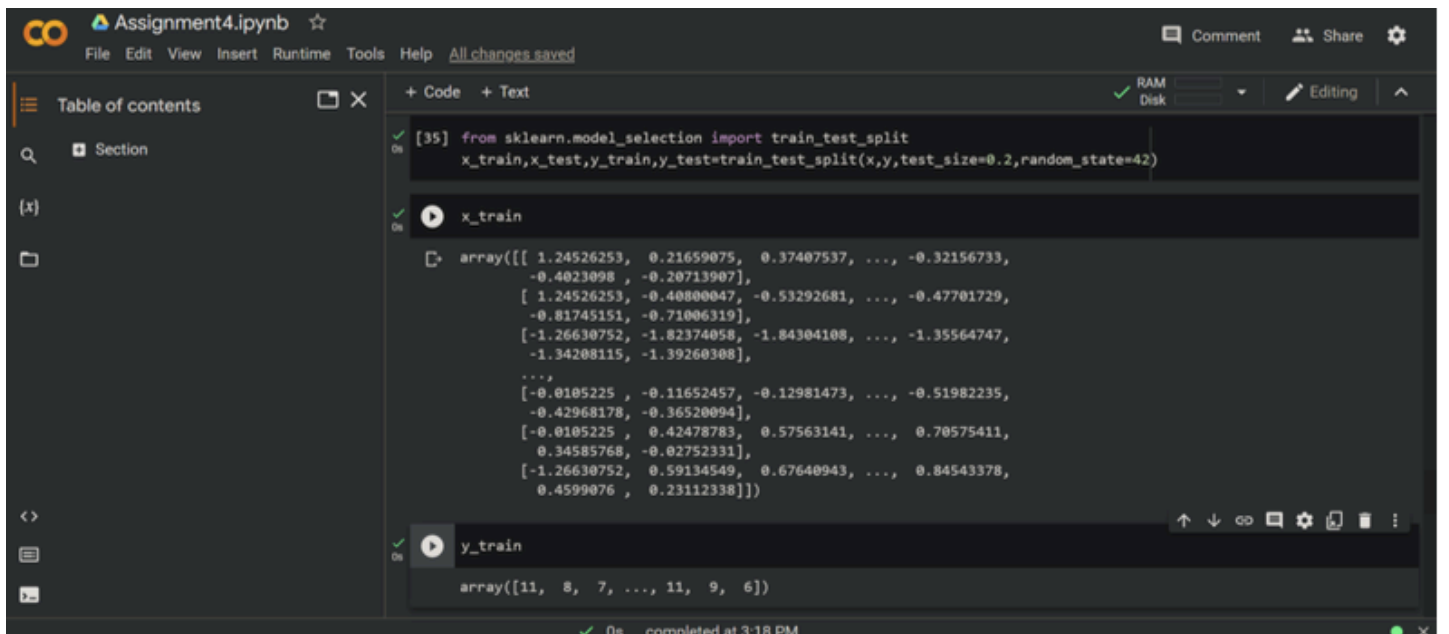
The screenshot shows a Jupyter Notebook interface for 'Assignment4.ipynb'. The code cell [33] contains the following Python code:

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

from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x=std.fit_transform(x)
x
```

The output of the code is a large NumPy array representing the scaled features. The array is a 10x10 matrix of floating-point values, where each row represents a data point and each column represents a feature. The values are centered around zero and have a standard deviation of one.

Split the data into training and testing



The screenshot shows a Jupyter Notebook interface for 'Assignment4.ipynb'. The code cell [35] contains the following Python code:

```
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=42)
```

The output of the code is two variables: `x_train` and `y_train`. `x_train` is a 10x10 matrix of floating-point values, representing the scaled features for the training set. `y_train` is a 1D array of integers, representing the target values for the training set.

Assignment4.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

Section

+ Code + Text

RAM Disk

Editing

[38] x\_test

```
array([[ -0.0105225,  0.67462432,  0.47485339, ...,  0.27770351,
         1.10314916,  0.61909342],
       [ -0.0105225,  0.54970607,  0.32368636, ...,  0.12450645,
         0.3139237,   0.04432299],
       [ -1.26630752,  0.29986958,  0.37407537, ..., -0.2449688,
         0.40060164,  0.69093973],
       ...,
       [ 1.24526253,  0.17495134,  0.22290834, ..., -0.0309435,
        -0.20614393, -0.22150833],
       [ 1.24526253, -0.4912793, -0.53292681, ..., -0.47025859,
        -0.81288951, -0.39393946],
       [ 1.24526253, -1.3240676, -1.33915098, ..., -1.17766853,
        -1.30558517, -1.17706417]])
```

Assignment4.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

Section

+ Code + Text

RAM Disk

Editing

y\_test

```
array([ 9,  8, 16,  9, 14, 11,  7,  6,  7, 10, 22,  7, 15,  9,  8, 18, 11,
        14, 13,  9, 20, 12, 12, 11, 10,  7, 11,  8,  9, 10,  9, 10,  6, 10,
         8,  9,  5,  3,  6,  6, 12, 12, 18,  8, 12, 13, 10, 10, 18,  4,  6,
        22,  8,  5,  7, 10, 15, 21, 10,  9, 10, 13, 11,  7,  9, 11,  4,  5,
         7,  9, 10, 11, 10,  7,  9, 12, 23, 14, 15,  9, 15, 13, 10,  6,  7,
        13,  9, 10, 19, 10, 10,  9, 11, 11, 10, 10,  6, 15,  7,  7, 15, 11,
        11, 13,  7,  9, 10,  8,  9, 14, 18,  8, 13,  9, 12,  5,  9, 12, 11,
        13, 11, 10,  8, 14,  9, 20,  9,  9,  9, 10,  9,  9, 10,  5,  8,  8,
        10, 10,  5, 12,  8, 11,  7,  8, 10, 15, 10, 14, 10, 10, 10,  8, 11,
        11,  8, 11, 12,  7,  8,  6,  9,  6, 10, 12,  7, 10, 17, 11,  8,  8,
        10, 12,  9,  8,  8,  7,  9, 11,  9, 10, 13,  7,  8,  8,  7, 10,  8,
        11,  9,  5,  9,  8, 16, 13, 11, 17, 10, 11, 12,  9,  8, 17, 11, 12,
         9, 12, 11,  9,  8, 10,  5,  9, 12,  6,  8, 11, 11,  7,  9, 12, 13,
         9, 12, 11,  9,  8,  7, 13,  9, 12,  5, 10, 10, 12,  7, 10, 10,  7,
         4, 10,  8, 11, 10,  9, 10,  8,  9,  7,  7,  6,  7,  9,  9,  7, 15,
        11,  9,  5, 12, 14, 19, 16,  9,  9,  7,  6,  7, 14, 12,  6,  9,  8,
         6, 12,  8, 18, 10, 16,  9,  6, 15,  9, 13,  8,  5,  9, 10,  5, 10,
        10, 11,  4, 15,  9, 15,  8,  5, 14,  7, 11, 10, 10,  7, 10,  9, 10,
        18,  8,  6,  5,  8,  6,  7, 14, 12, 10,  5, 23,  9,  9, 12,  7,  8,
         8, 13,  6, 13, 17,  7,  8,  8,  7,  7,  9, 14, 10,  9, 13,  8, 10,
        10,  9, 10,  9,  8,  8, 10, 13, 10,  9,  8, 10,  8, 11, 10,  3,  7,
         6,  3,  8,  8, 13, 15,  6, 14,  8,  9, 12,  8,  8, 15, 11,  9,  6,
        10, 13, 13,  7,  7,  9,  9,  8,  7, 10, 11,  5, 10, 12,  8,  7,  9,
         6,  8, 13,  7,  7, 10, 11, 23,  9, 11, 10,  8,  8,  7,  7, 10,  9,
```

Build the model, Train the Model

Assignment4.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Table of contents

Section

+ Code + Text

RAM Disk

Editing

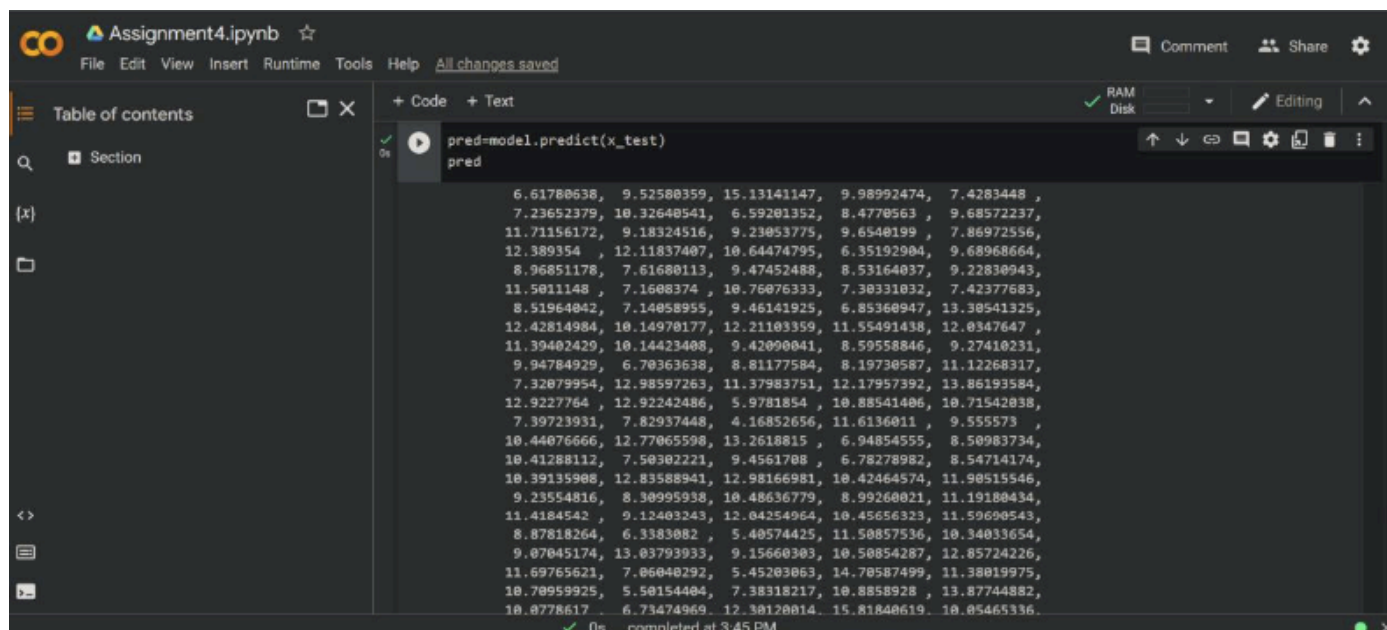
[40] from sklearn.ensemble import RandomForestRegressor

```
model = RandomForestRegressor(n_estimators = 1000, oob_score = True, n_jobs=-1, min_samples_split = 6, min_samples_leaf = 4, max_features = 'sqrt', max_depth = 120, bootstrap=True)
```

[41] model.fit(x\_train,y\_train)

```
RandomForestRegressor(max_depth=120, max_features='sqrt', min_samples_leaf=4, min_samples_split=6, n_estimators=1000, n_jobs=-1, oob_score=True)
```

## Test the Model

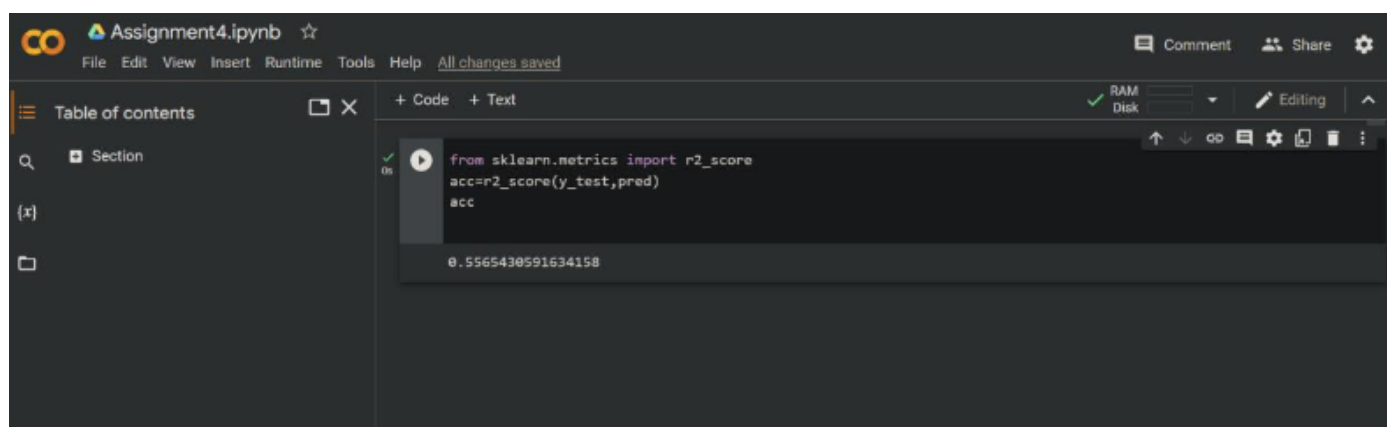


A screenshot of a Jupyter Notebook titled "Assignment4.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a status bar indicating "All changes saved". On the left, a "Table of contents" sidebar shows a "Section" and a search icon. The main area displays a code cell with the following code:

```
pred=model.predict(x_test)
pred
```

The output of the code cell is a large array of numerical values, representing the model's predictions for the test set. The array is displayed in a grid-like format, with values ranging from approximately 6.6 to 15.8. The status bar at the bottom indicates that the code was "completed at 3:45 PM".

## Measure the performance using metrics



A screenshot of a Jupyter Notebook titled "Assignment4.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", along with a status bar indicating "All changes saved". On the left, a "Table of contents" sidebar shows a "Section" and a search icon. The main area displays a code cell with the following code:

```
from sklearn.metrics import r2_score
acc=r2_score(y_test,pred)
acc
```

The output of the code cell is a single numerical value, representing the R-squared score for the model's performance on the test set. The value is displayed as 0.5565430591634158. The status bar at the bottom indicates that the code was "completed at 3:45 PM".