

IBM ASSIGNMENT 4

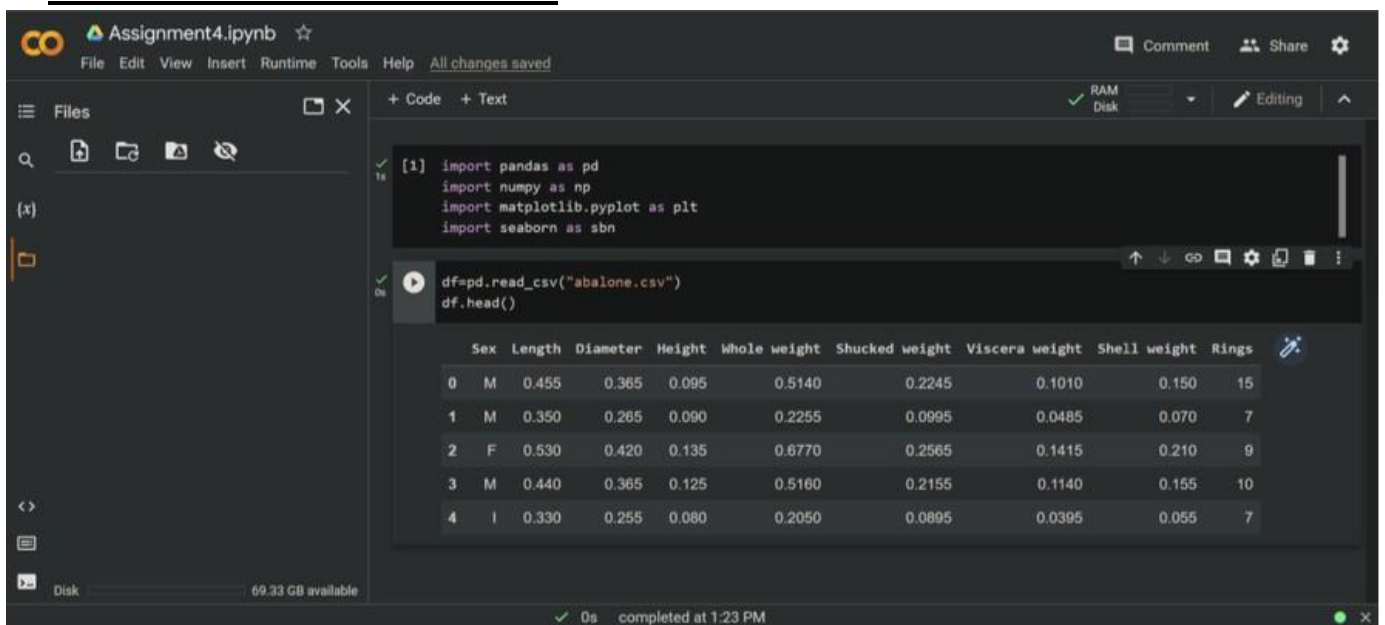
Name: P.R.Prithiv singh

Register No: 961819104066

CHALLENGE:

Abalone Age Prediction

LOADING THE DATASET:



The screenshot shows a Jupyter Notebook interface with the file 'Assignment4.ipynb' open. The code cell contains the following Python code:

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

df=pd.read_csv("abalone.csv")
df.head()
```

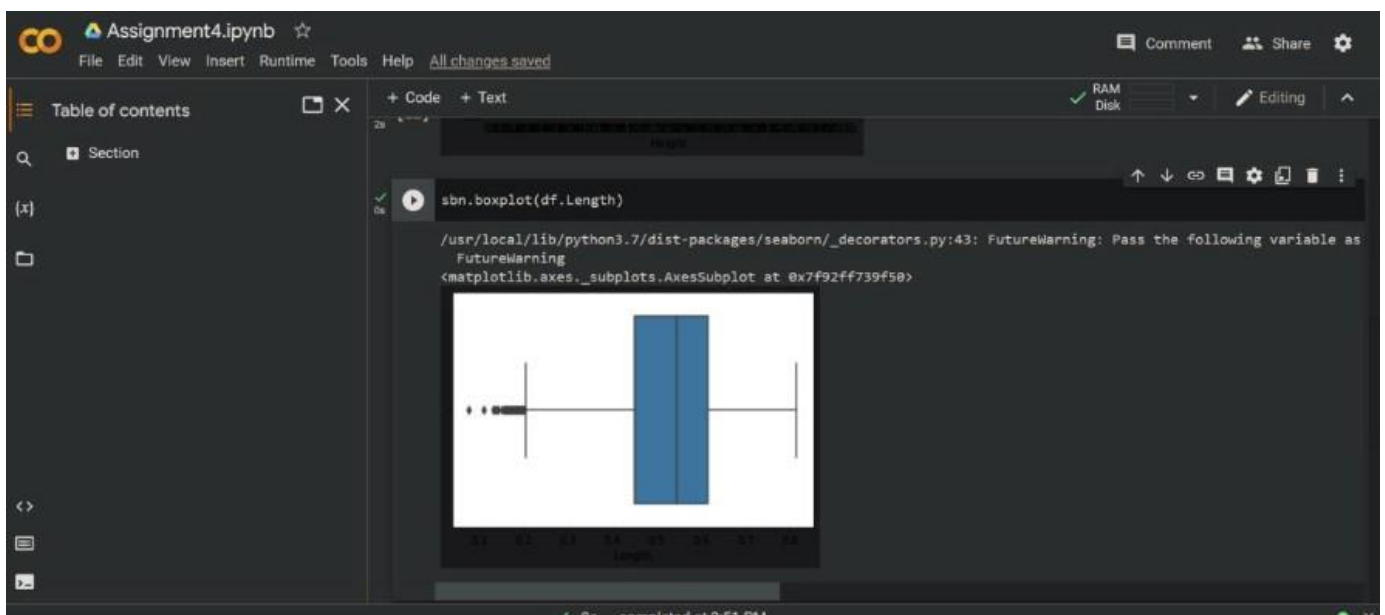
The output of the code is a preview of the first five rows of the 'abalone.csv' dataset:

	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:

Univariate Analysis

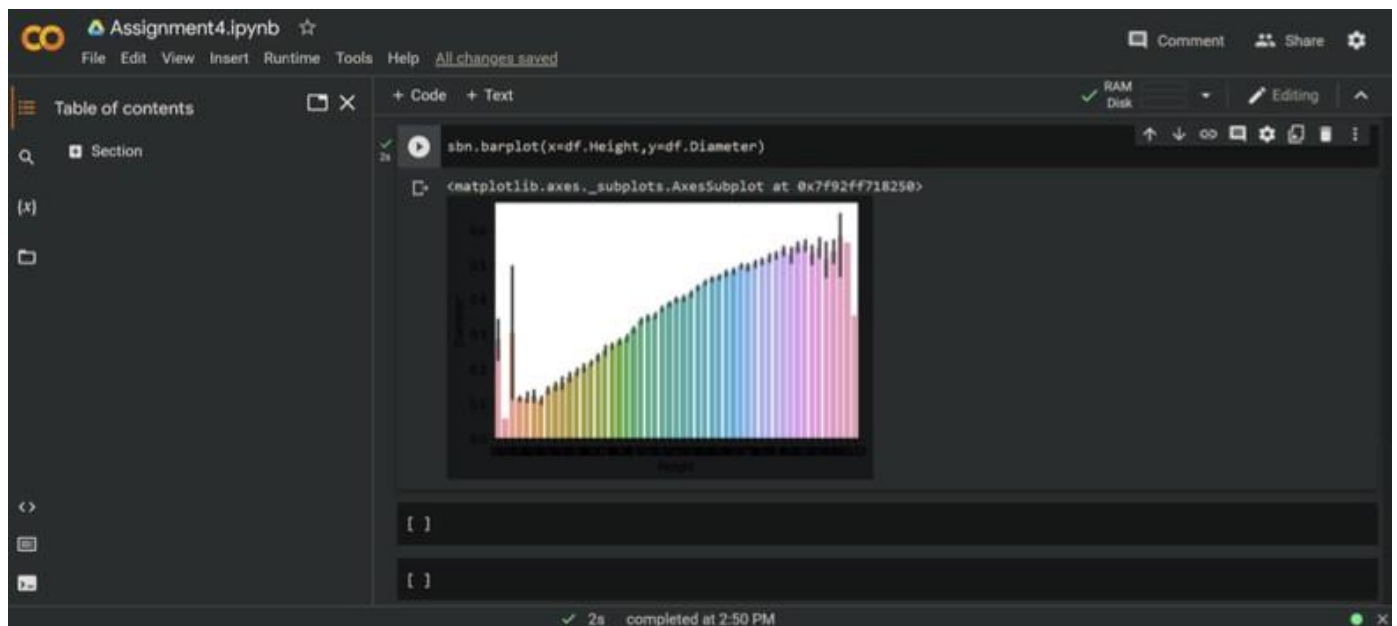
Bi-Variate Analysis



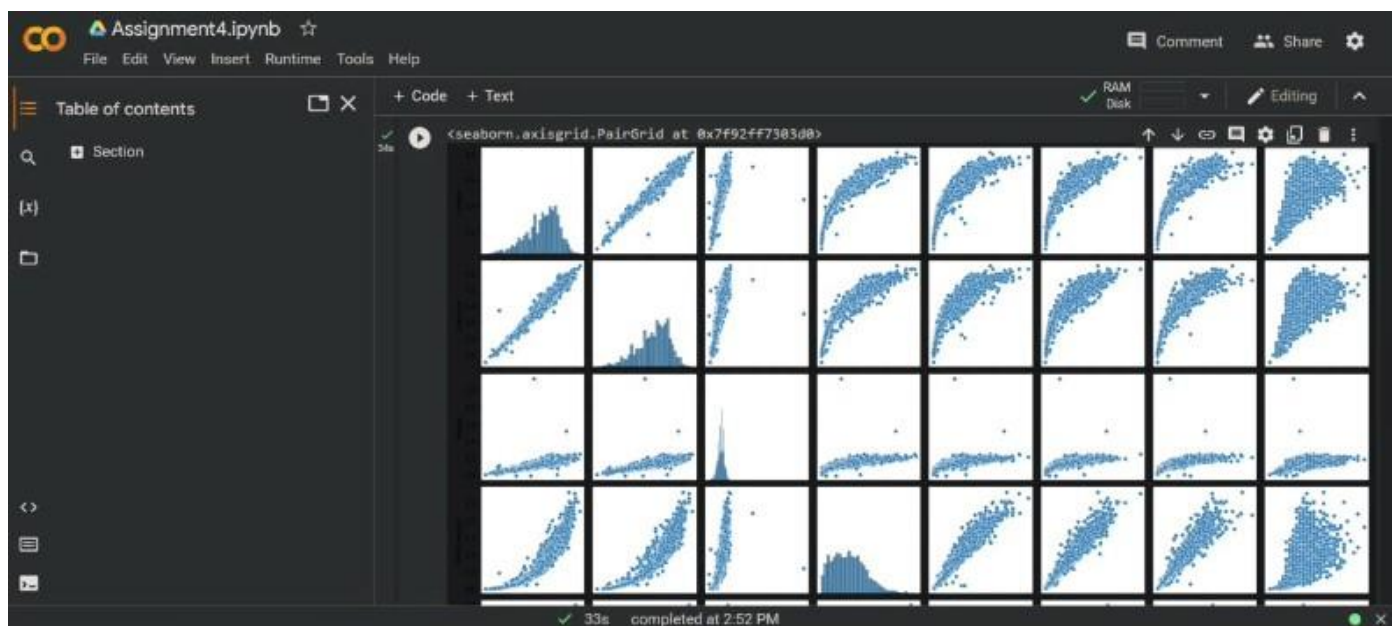
The screenshot shows a Jupyter Notebook interface with the file 'Assignment4.ipynb' open. The code cell contains the following Python code:

```
2s sbn.boxplot(df.Length)
```

The output of the code is a boxplot visualization of the 'Length' variable. The x-axis is labeled 'Length' and ranges from 0.1 to 0.8. The y-axis represents the frequency of observations. The boxplot shows a distribution of 'Length' values, with a median around 0.45 and a range from approximately 0.1 to 0.8.



Multi-Variate Analysis



Perform descriptive Analysis on datasets

This screenshot shows a Jupyter Notebook titled 'Assignment4.ipynb'. The left sidebar contains a 'Table of contents' and a search bar. The main area displays three code cells. The first cell, [15], shows the mode of the 'Length' variable, which is 0.550 for category 0 and 0.625 for category 1. The second cell, [17], shows the mean of the 'Height' variable, which is approximately 0.1395. The third cell, [20], shows the count of each variable, with all variables having a count of 4177. The status bar at the bottom indicates the notebook is completed at 2:56 PM.

```
[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
```

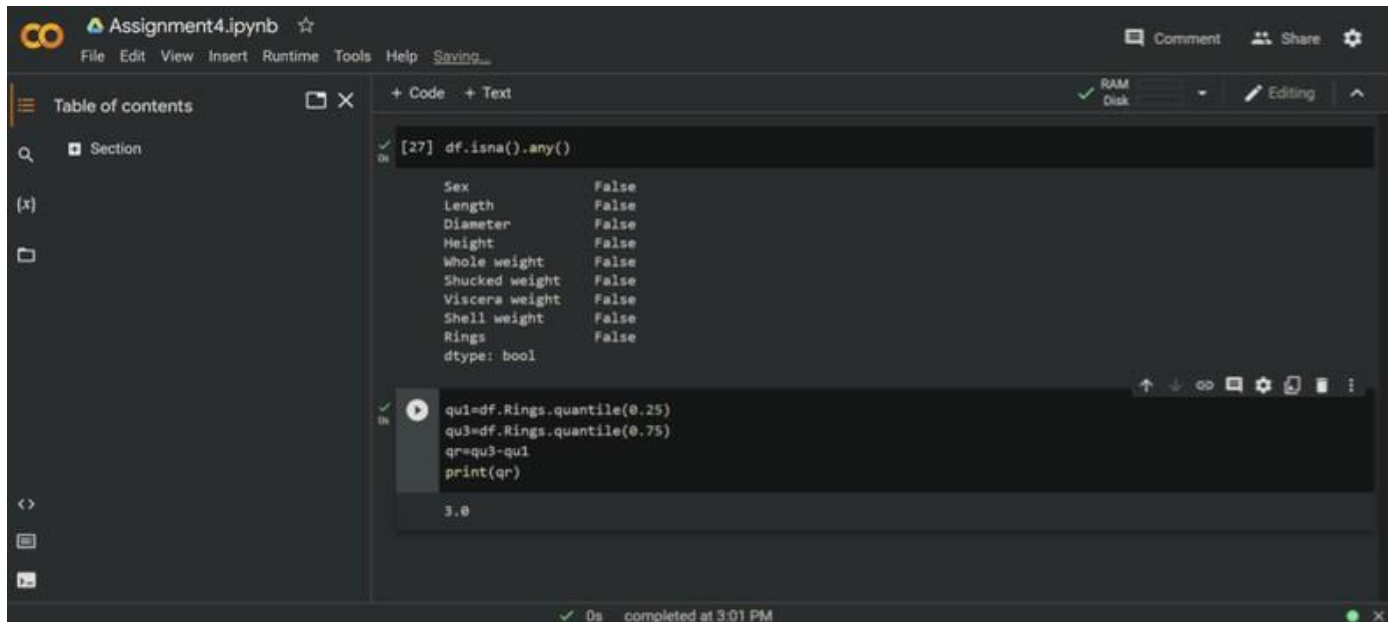
This screenshot shows the same Jupyter Notebook at a later stage. The left sidebar is the same. The main area displays three more code cells. The first cell, [23], shows the sum of the 'Shell weight' variable, which is 997.5964999999999. The second cell, [24], shows the product of the 'Rings' variable, which is 0. The third cell, [25], shows the maximum of the 'Whole weight' variable, which is 2.8255. The status bar at the bottom indicates the notebook is completed at 2:59 PM.

```
[23] df['Shell weight'].sum()
997.5964999999999

[24] df['Rings'].product()
0

[25] df['Whole weight'].max()
2.8255
```

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



The 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 two code cells. The first cell, labeled '[27]', contains the code `df.isna().any()`, which has been executed to check for missing values. The output is a series of boolean values for each column: Sex (False), Length (False), Diameter (False), Height (False), Whole weight (False), Shucked weight (False), Viscera weight (False), Shell weight (False), and Rings (False). The data type for the 'Rings' column is shown as 'dtype: bool'. The second cell contains code to calculate the interquartile range (IQR) for the 'Rings' column: `qu1=df.Rings.quantile(0.25)`, `qu3=df.Rings.quantile(0.75)`, `qr=qu3-qu1`, and `print(qr)`. The output of this cell is the value '3.0'. The bottom status bar indicates the notebook is 'completed at 3:01 PM'.

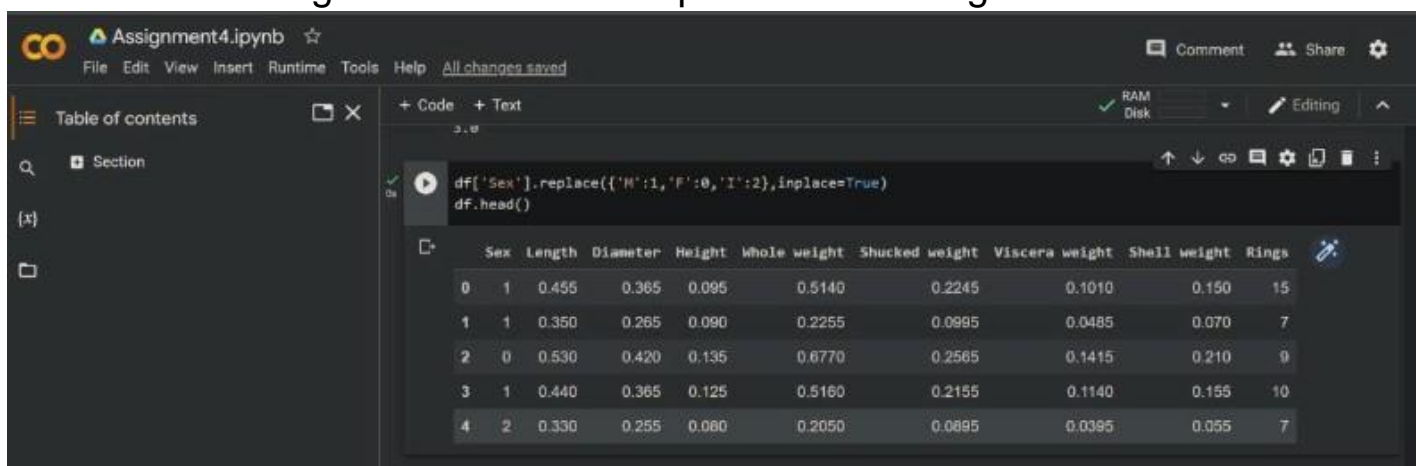
```
[27] df.isna().any()

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

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

3.0
```

Check for categorical columns and perform encoding

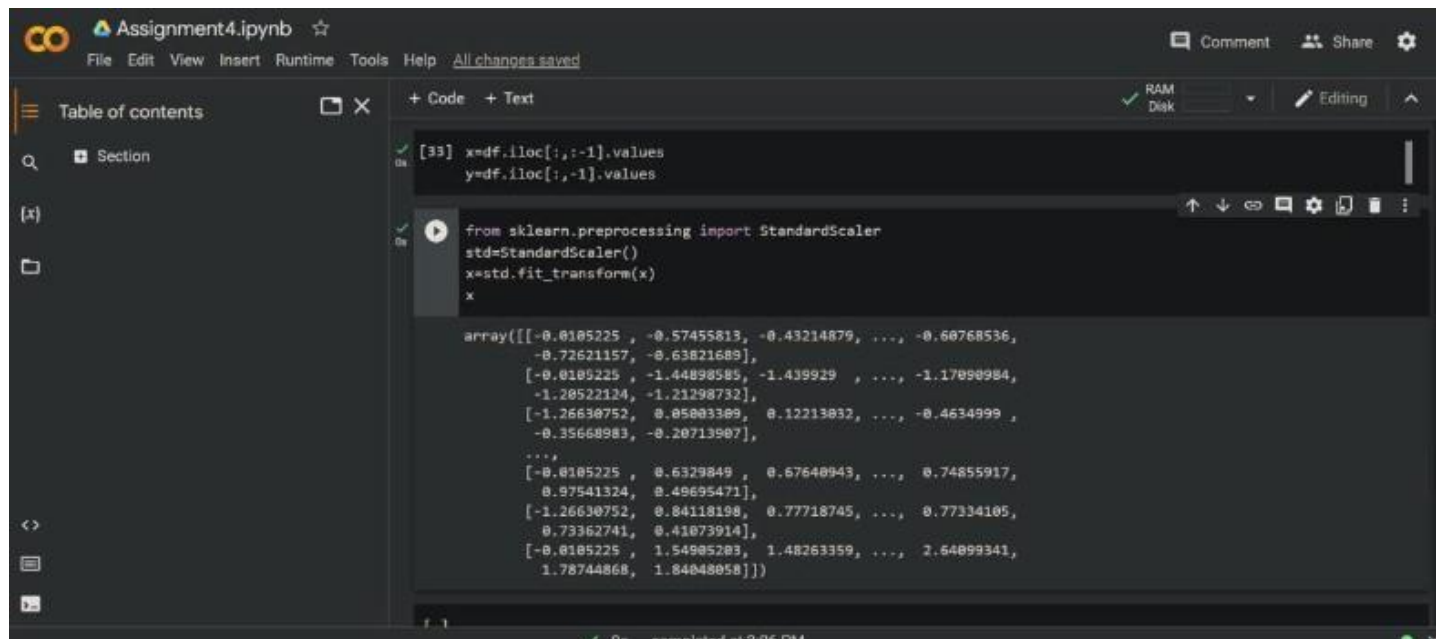


The screenshot shows the same Jupyter Notebook interface. The first code cell now contains the code `df['Sex'].replace({'M':1,'F':0,'I':2},inplace=True)` followed by `df.head()` to perform label encoding on the 'Sex' column. The output of the `df.head()` command is displayed as a table showing the first five rows of the dataset. The 'Sex' column has been successfully encoded with numerical values: 1 for 'M', 0 for 'F', and 2 for 'I'.

```
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 titled "Assignment4.ipynb". The left sidebar contains a "Table of contents" and a "Section" dropdown. The main area displays two code cells. The first cell, labeled [33], contains the following code:

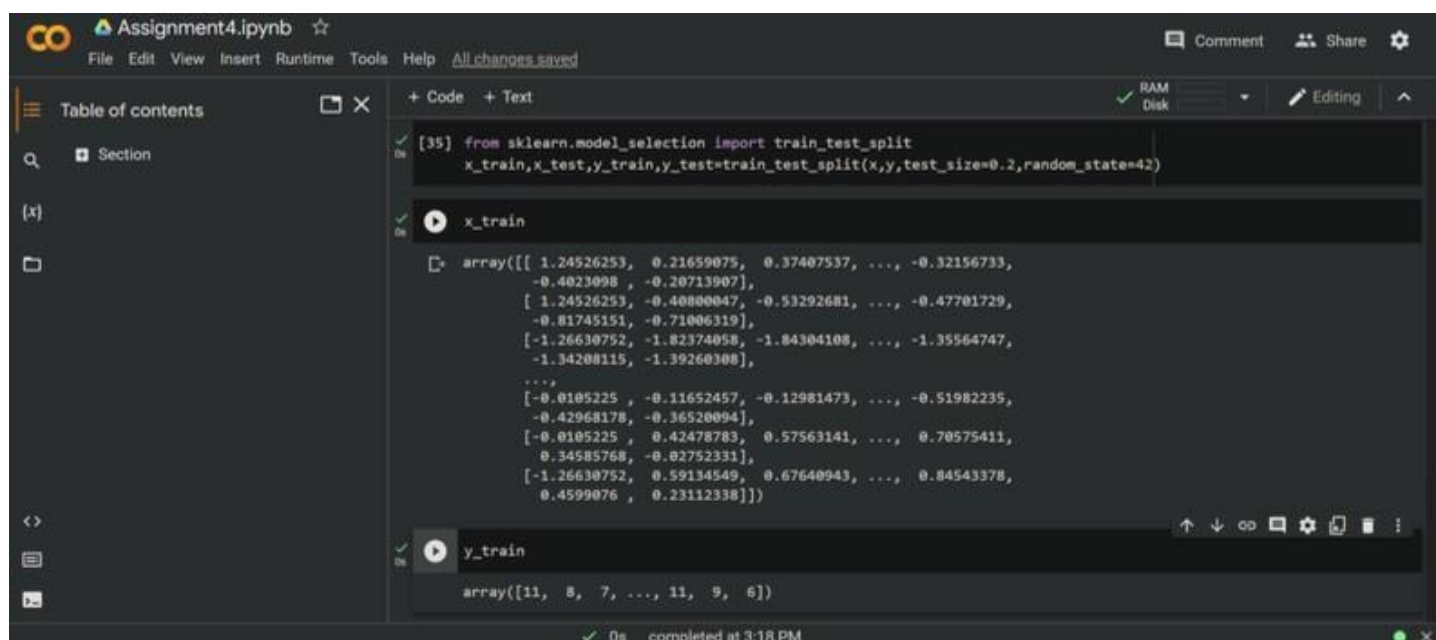
```
x=df.iloc[:, :-1].values
y=df.iloc[:, -1].values
```

The second cell, labeled [34], contains the following code:

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

The output of the second cell is a large NumPy array of standardized features. The status bar at the bottom indicates "completed at 3:06 PM".

Split the data into training and testing



The screenshot shows the same Jupyter Notebook "Assignment4.ipynb". The first code cell, labeled [35], contains the following 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 second cell, labeled [36], contains the following code:

```
x_train
```

The output of the second cell is a large NumPy array of training features. The third cell, labeled [37], contains the following code:

```
y_train
```

The output of the third cell is a small NumPy array of training labels: `array([11, 8, 7, ..., 11, 9, 6])`. The status bar at the bottom indicates "completed at 3:18 PM".

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 Test 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)
```


The screenshot shows a Jupyter Notebook titled "Assignment4.ipynb". The left sidebar contains a "Table of contents" and a "Section" dropdown. The main area is in "Code" mode, displaying a single code cell with the following code:

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

The output of the code cell is a large array of 50 numerical values, arranged in 10 rows and 5 columns. The values range from approximately 6.5 to 15.8. The status bar at the bottom indicates the code was "completed at 3:45 PM".

Measure the performance using metrics

The screenshot shows the same Jupyter Notebook interface. The code cell now contains the following code to calculate the R-squared score:

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

The output of the code cell is the value 0.5565430591634158. The status bar at the bottom indicates the code was "completed at 3:45 PM".