ICP4

Student Name: **Srinivas Musinuri**
Student id:700758813

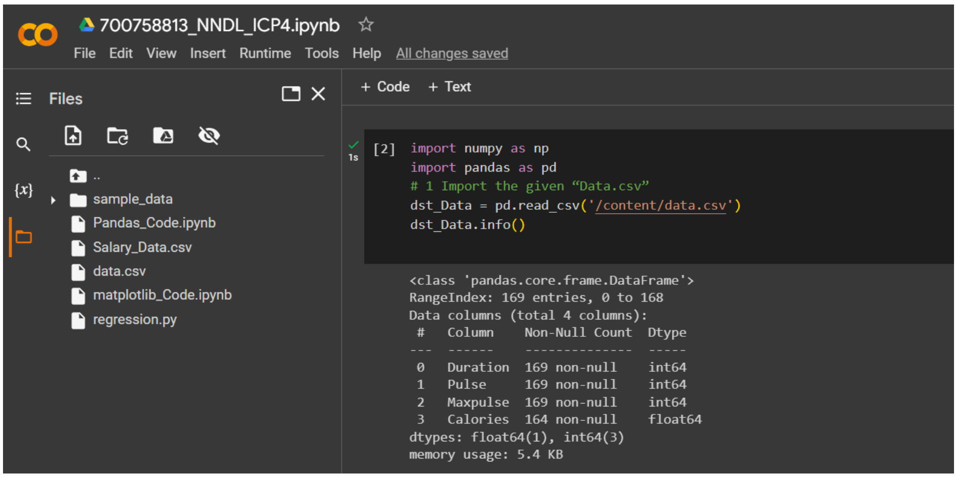
GitHub Link: https://github.com/srinivasmusinuri/700758813_NNDL_ICP4

Video Link:
https://drive.google.com/file/d/1tdcyMJxzvThn85YoVXFr4buGFZMn_8sJ/view?usp=sharing

1. Data Manipulation

a. Read the provided CSV file ‘data.csv’.

b. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing



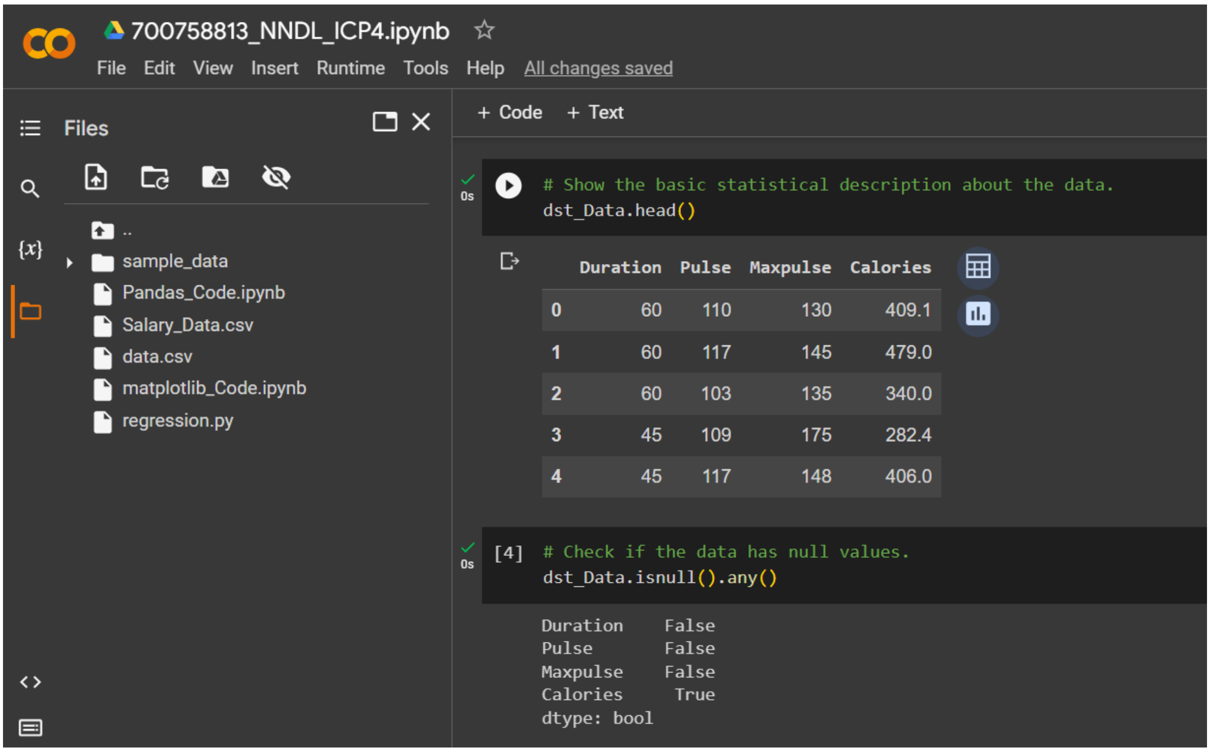
```
import numpy as np
import pandas as pd
# 1 Import the given “Data.csv”
dst_Data = pd.read_csv('/content/data.csv')
dst_Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```

c. Show the basic statistical description about the data.

d. Check if the data has null values.

i. Replace the null values with the mean.

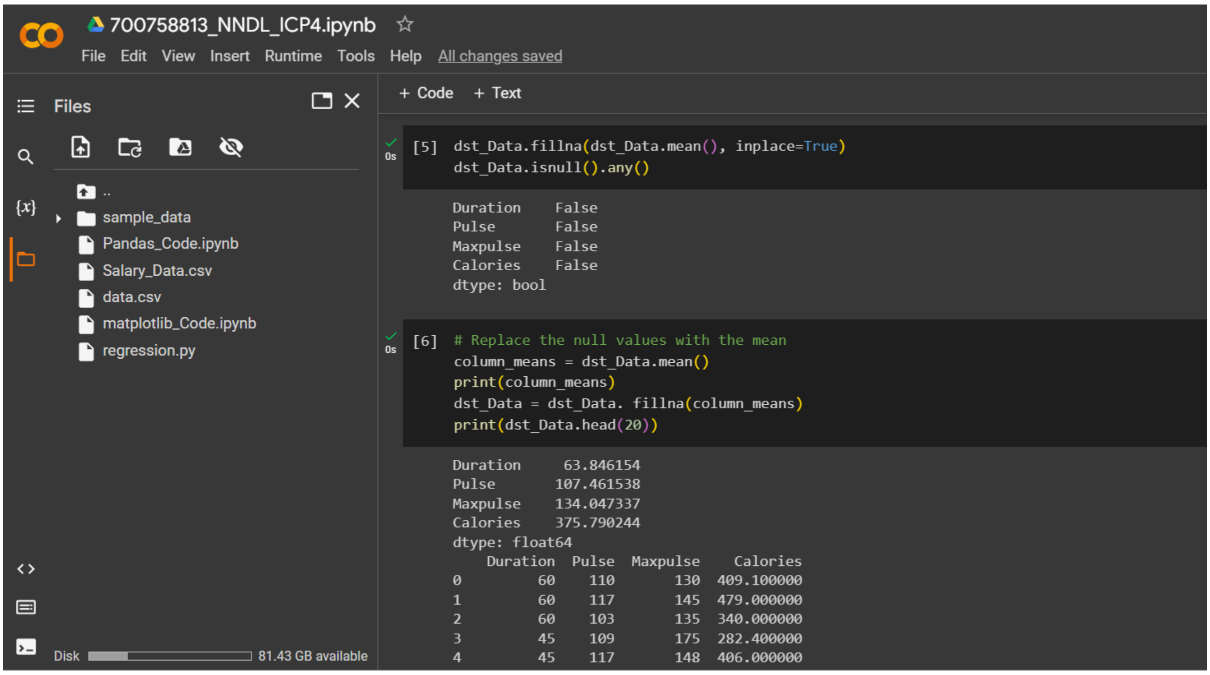


```
# Show the basic statistical description about the data.
dst_Data.head()
```

	Duration	Pulse	Maxpulse	Calories
0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0

```
[4] # Check if the data has null values.
    dst_Data.isnull().any()
```

```
Duration     False
Pulse        False
Maxpulse     False
Calories      True
dtype: bool
```



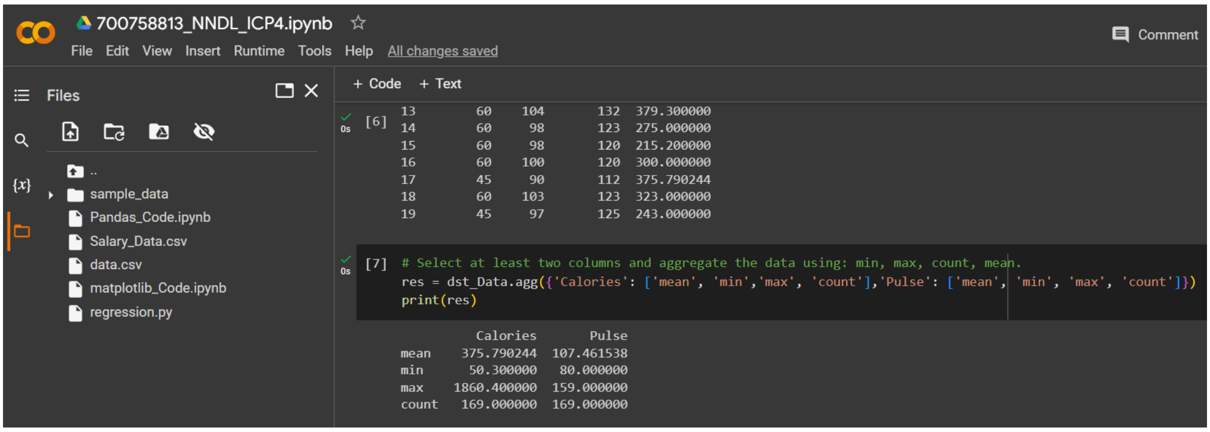
```
[5] dst_Data.fillna(dst_Data.mean(), inplace=True)
    dst_Data.isnull().any()
```

```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

```
[6] # Replace the null values with the mean
    column_means = dst_Data.mean()
    print(column_means)
    dst_Data = dst_Data. fillna(column_means)
    print(dst_Data.head(20))
```

```
Duration     63.846154
Pulse       107.461538
Maxpulse    134.047337
Calories    375.790244
dtype: float64
     Duration  Pulse  Maxpulse    Calories
0          60    110       130  409.100000
1          60    117       145  479.000000
2          60    103       135  340.000000
3          45    109       175  282.400000
4          45    117       148  406.000000
```

e. Select at least two columns and aggregate the data using: min, max, count, mean.



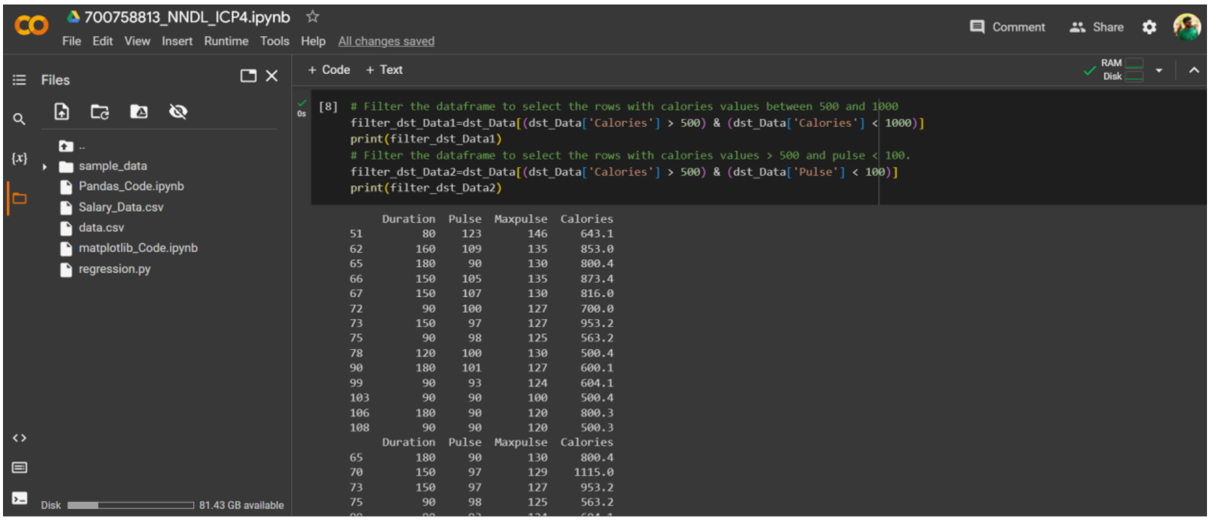
```
[6]
13        60    104       132  379.300000
14        60     98       123  275.000000
15        60     98       120  215.200000
16        60    100       120  300.000000
17        45     90       112  375.790244
18        60    103       123  323.000000
19        45     97       125  243.000000
```

```
[7] # Select at least two columns and aggregate the data using: min, max, count, mean.
    res = dst_Data.agg({'Calories': ['mean', 'min','max', 'count'],'Pulse': ['mean', 'min', 'max', 'count']})
    print(res)

          Calories       Pulse
mean    375.790244  107.461538
min      50.300000   80.000000
max    1860.400000  159.000000
count   169.000000  169.000000
```

f. Filter the dataframe to select the rows with calories values between 500 and1000.

g. Filter the dataframe to select the rows with calories values > 500 and pulse.

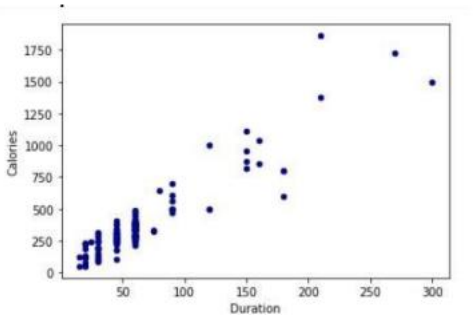


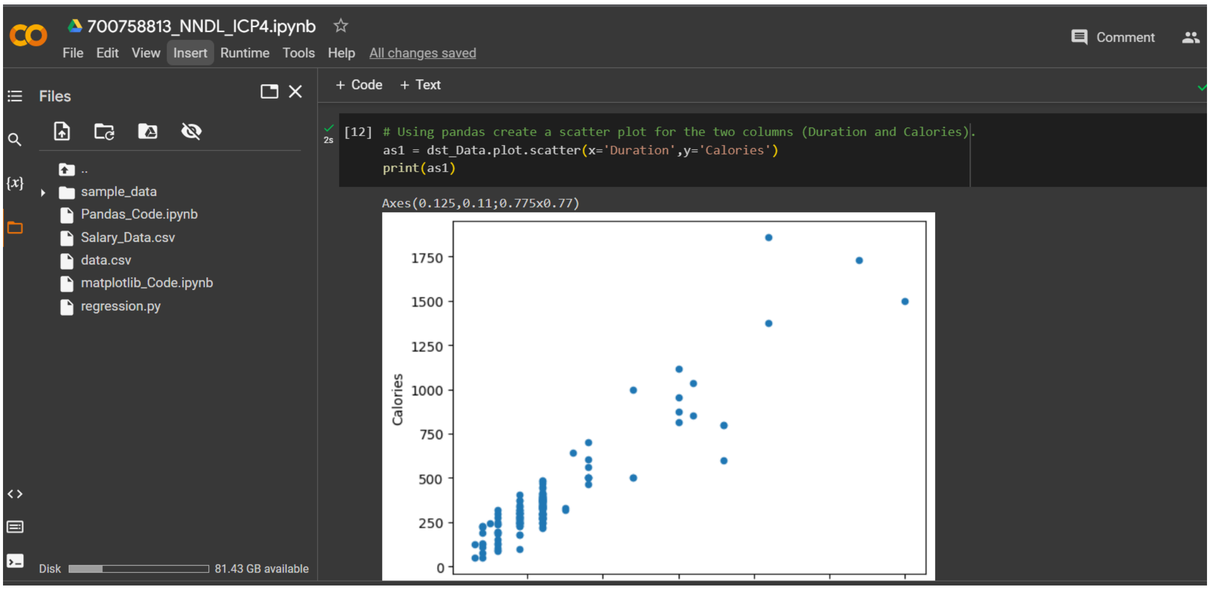
```
[8] # Filter the dataframe to select the rows with calories values between 500 and 1000
    filter_dst_Data1=dst_Data[(dst_Data['Calories'] > 500) & (dst_Data['Calories'] < 1000)]
    print(filter_dst_Data1)
    # Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
    filter_dst_Data2=dst_Data[(dst_Data['Calories'] > 500) & (dst_Data['Pulse'] < 100)]
    print(filter_dst_Data2)

     Duration  Pulse  Maxpulse  Calories
51         80    123       146     643.1
62        160    109       135     853.0
65        180     90       130     800.4
66        150    105       135     873.4
67        150    107       130     816.0
72         90    100       127     700.0
73        150     97       127     953.2
75         90     98       125     563.2
78        120    100       130     500.4
90        180    101       127     600.1
99         90     93       124     604.1
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
     Duration  Pulse  Maxpulse  Calories
65        180     90       130     800.4
70        150     97       129    1115.0
73        150     97       127     953.2
75         90     98       125     563.2
```

k. Using pandas create a scatter plot for the two columns (Duration and Calories).

Example





```
[12] # Using pandas create a scatter plot for the two columns (Duration and Calories).
     as1 = dst_Data.plot.scatter(x='Duration',y='Calories')
     print(as1)

Axes(0.125,0.11;0.775x0.77)
```

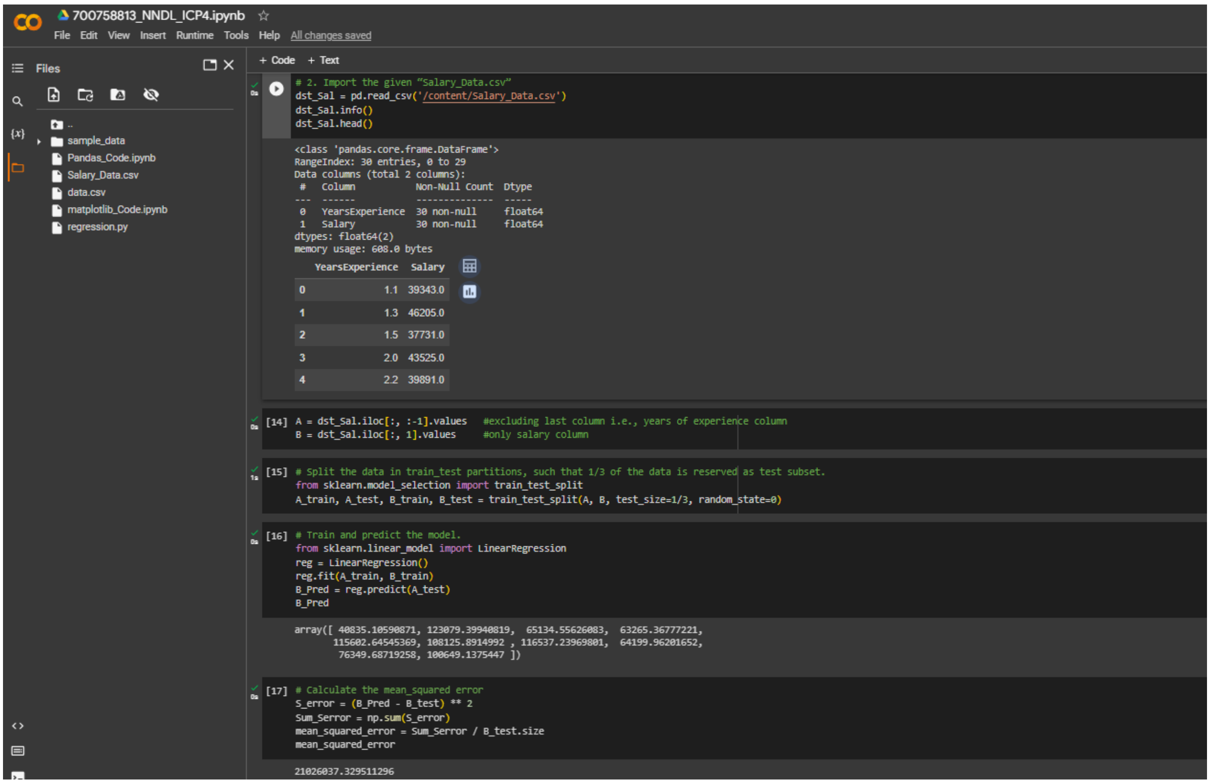
2. Linear Regression

a) Import the given “Salary_Data.csv”

b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.

c) Train and predict the model.

d) Calculate the mean_squared error



```
# 2. Import the given “Salary_Data.csv”
dst_Sal = pd.read_csv('/content/Salary_Data.csv')
dst_Sal.info()
dst_Sal.head()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
[14] A = dst_Sal.iloc[:, :-1].values   #excluding last column i.e., years of experience column
     B = dst_Sal.iloc[:, 1].values    #only salary column
```

```
[15] # Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
     from sklearn.model_selection import train_test_split
     A_train, A_test, B_train, B_test = train_test_split(A, B, test_size=1/3, random_state=0)
```

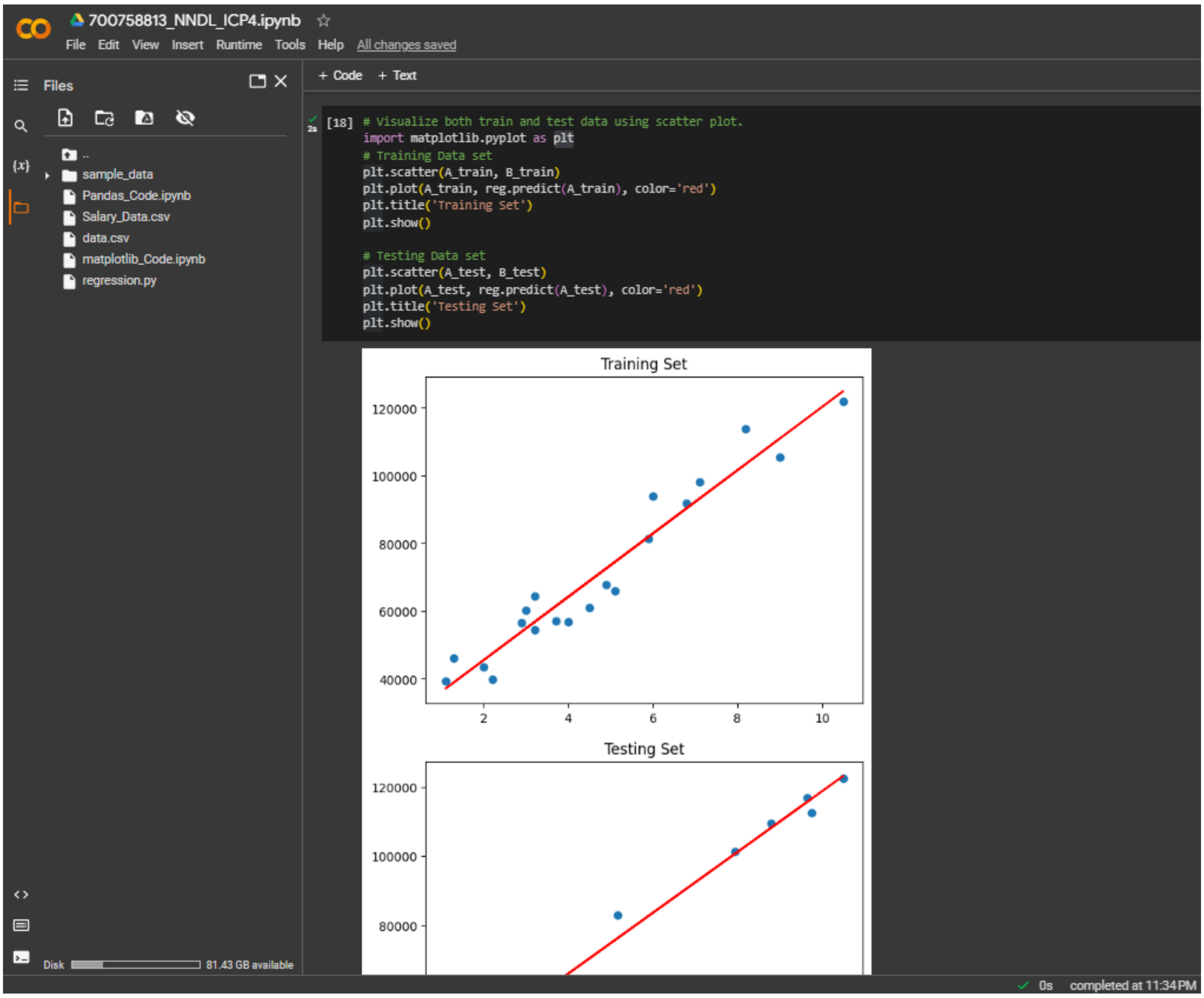
```
[16] # Train and predict the model.
     from sklearn.linear_model import LinearRegression
     reg = LinearRegression()
     reg.fit(A_train, B_train)
     B_Pred = reg.predict(A_test)
     B_Pred

array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

```
[17] # Calculate the mean_squared error
     S_error = (B_Pred - B_test) ** 2
     Sum_Serror = np.sum(S_error)
     mean_squared_error = Sum_Serror / B_test.size
     mean_squared_error

21026037.329511296
```

e) Visualize both train and test data using scatter plot.



```
[18] # Visualize both train and test data using scatter plot.
     import matplotlib.pyplot as plt
     # Training Data set
     plt.scatter(A_train, B_train)
     plt.plot(A_train, reg.predict(A_train), color='red')
     plt.title('Training Set')
     plt.show()

     # Testing Data set
     plt.scatter(A_test, B_test)
     plt.plot(A_test, reg.predict(A_test), color='red')
     plt.title('Testing Set')
     plt.show()
```