Neural Networks and Deep Learning
ICP4

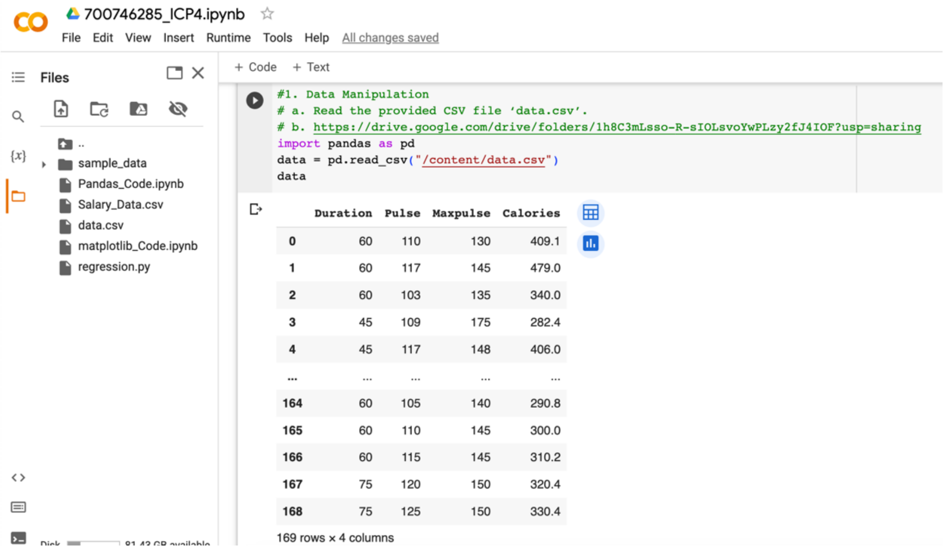
Student Name: Sravani Lankala
Student Id: 700746285

GitHub Link: https://github.com/sravanilankala/NNDL_ICP4_Fall2023

Video Link: https://drive.google.com/file/d/1x8z1x5ml5F91KMepjyEj0QMeLfsKsUoG/view?usp=sharing

1. Data Manipulation

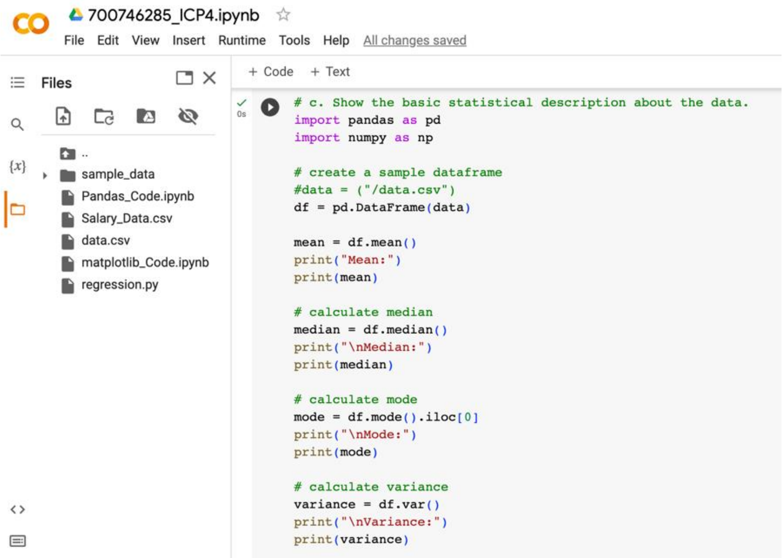
a. Read the provided CSV file 'data.csv'.
b. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing



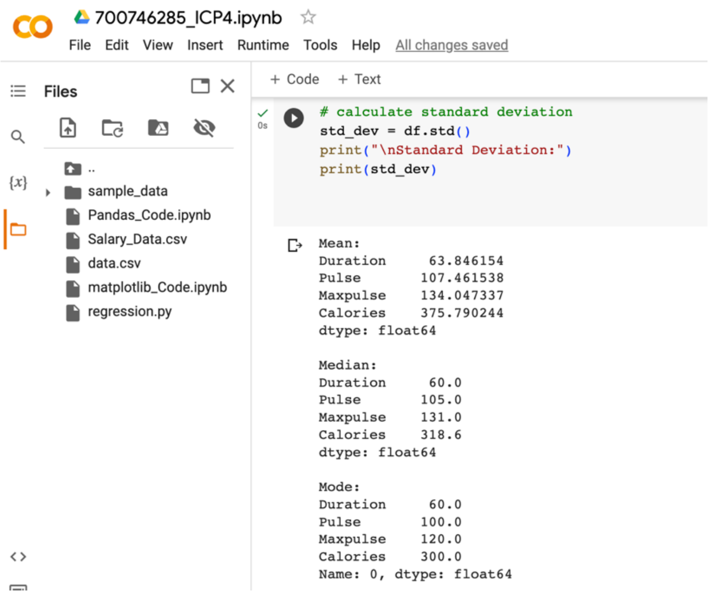
```
#1. Data Manipulation
# a. Read the provided CSV file 'data.csv'.
# b. https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing
import pandas as pd
data = pd.read_csv("/content/data.csv")
data
```

```
     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
...       ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
169 rows × 4 columns
```

c. Show the basic statistical description about the data.

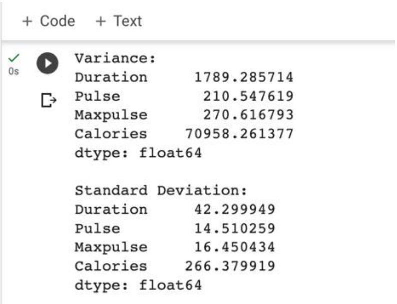


```
# c. Show the basic statistical description about the data.
import pandas as pd
import numpy as np

# create a sample dataframe
#data = ("/data.csv")
df = pd.DataFrame(data)

mean = df.mean()
print("Mean:")
print(mean)

# calculate median
median = df.median()
print("\nMedian:")
print(median)

# calculate mode
mode = df.mode().iloc[0]
print("\nMode:")
print(mode)

# calculate variance
variance = df.var()
print("\nVariance:")
print(variance)
```



```
# calculate standard deviation
std_dev = df.std()
print("\nStandard Deviation:")
print(std_dev)
```

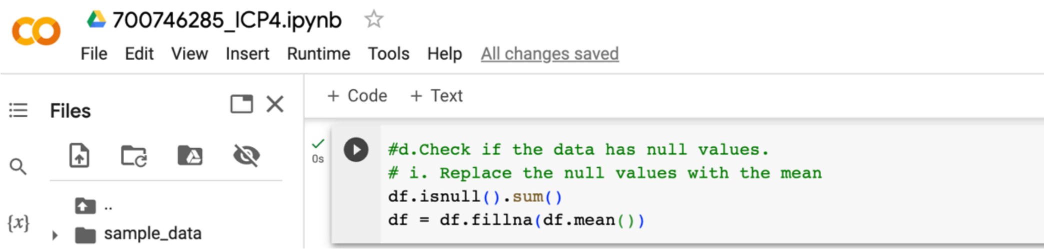
```
Mean:
Duration      63.846154
Pulse        107.461538
Maxpulse     134.047337
Calories     375.790244
dtype: float64

Median:
Duration      60.0
Pulse        105.0
Maxpulse     131.0
Calories     318.6
dtype: float64

Mode:
Duration      60.0
Pulse        100.0
Maxpulse     120.0
Calories     300.0
Name: 0, dtype: float64
```



```
Variance:
Duration      1789.285714
Pulse          210.547619
Maxpulse       270.616793
Calories     70958.261377
dtype: float64

Standard Deviation:
Duration      42.299949
Pulse         14.510259
Maxpulse      16.450434
Calories     266.379919
dtype: float64
```

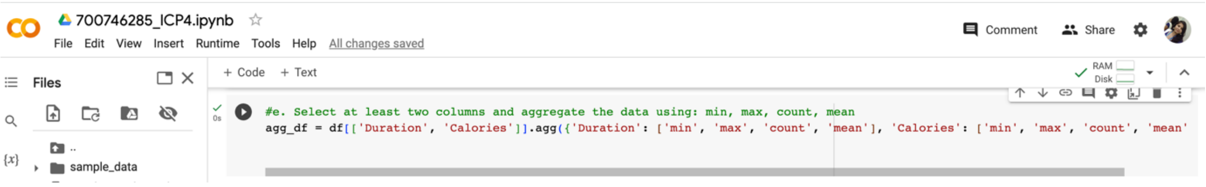
d. Check if the data has null values.

   i. Replace the null values with the mean



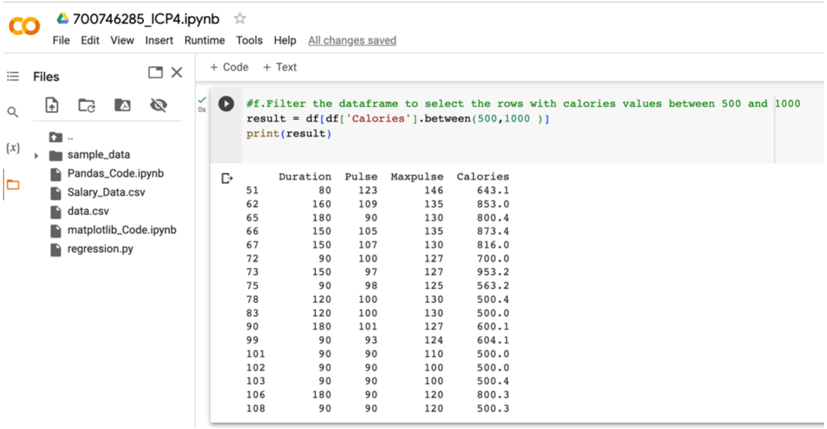
```
#d.Check if the data has null values.
# i. Replace the null values with the mean
df.isnull().sum()
df = df.fillna(df.mean())
```

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



```
#e. Select at least two columns and aggregate the data using: min, max, count, mean
agg_df = df[['Duration', 'Calories']].agg({'Duration': ['min', 'max', 'count', 'mean'], 'Calories': ['min', 'max', 'count', 'mean'
```

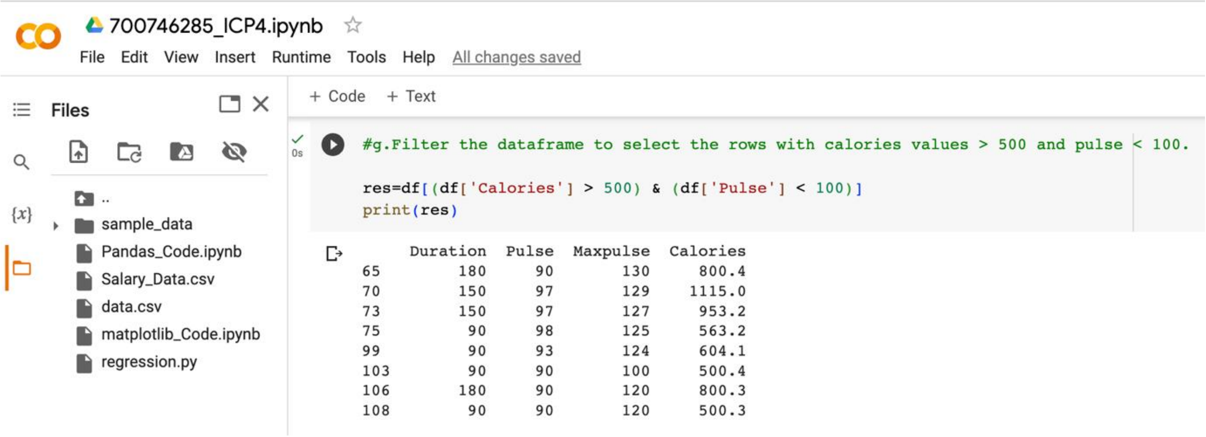
f. Filter the dataframe to select the rows with calories values between 500 and1000.



```
#f.Filter the dataframe to select the rows with calories values between 500 and 1000
result = df[df['Calories'].between(500,1000 )]
print(result)
```

```
     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
83        120    100       130     500.0
90        180    101       127     600.1
99         90     93       124     604.1
101        90     90       110     500.0
102        90     90       100     500.0
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

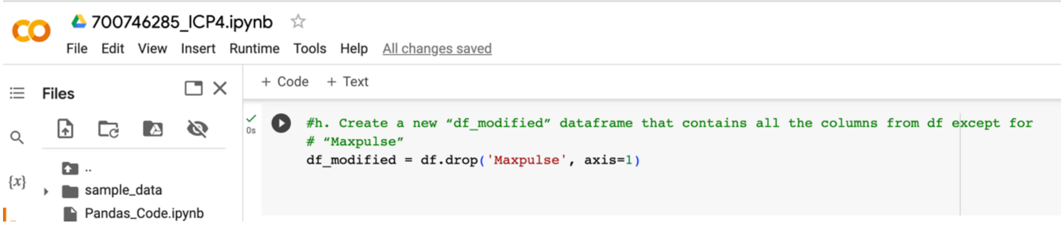
g. Filter the dataframe to select the rows with calories values > 500 and pulse <100.



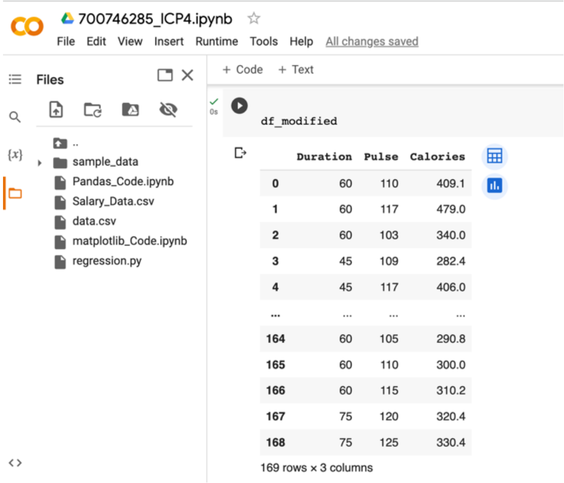
```
#g.Filter the dataframe to select the rows with calories values > 500 and pulse < 100.

res=df[(df['Calories'] > 500) & (df['Pulse'] < 100)]
print(res)
```

```
     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
99         90     93       124     604.1
103        90     90       100     500.4
106       180     90       120     800.3
108        90     90       120     500.3
```

h. Create a new “df_modified” dataframe that contains all the columns from df except for “Maxpulse”.



```
#h. Create a new “df_modified” dataframe that contains all the columns from df except for
# “Maxpulse”
df_modified = df.drop('Maxpulse', axis=1)
```

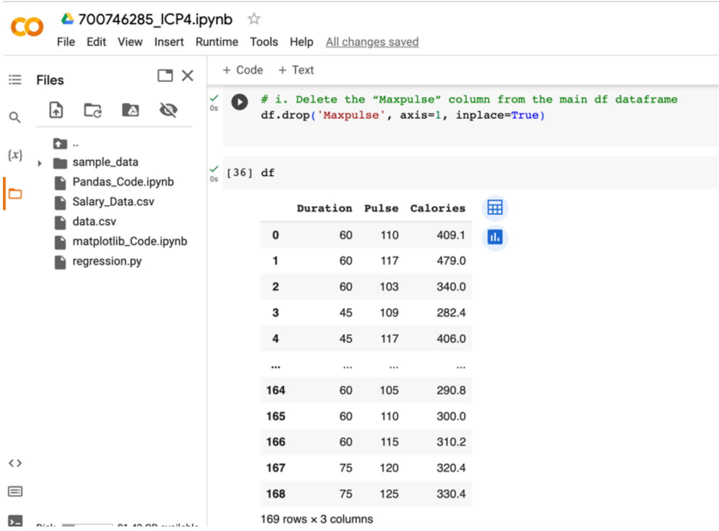


```
df_modified
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0
...	...	...	...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

169 rows × 3 columns

i. Delete the “Maxpulse” column from the main df dataframe



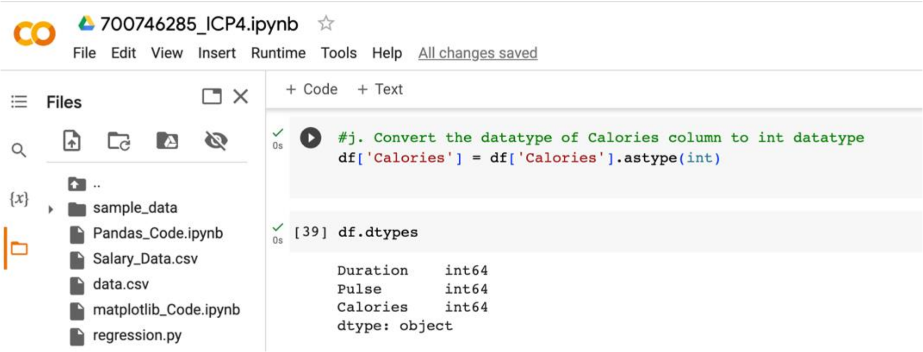
```
# i. Delete the "Maxpulse" column from the main df dataframe
df.drop('Maxpulse', axis=1, inplace=True)
```

```
[36] df
```

	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0
...	...	...	...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

169 rows × 3 columns

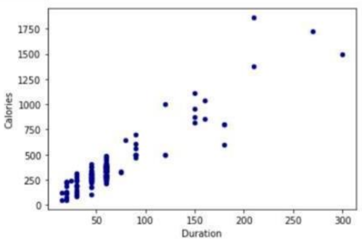
j. Convert the datatype of Calories column to int datatype.

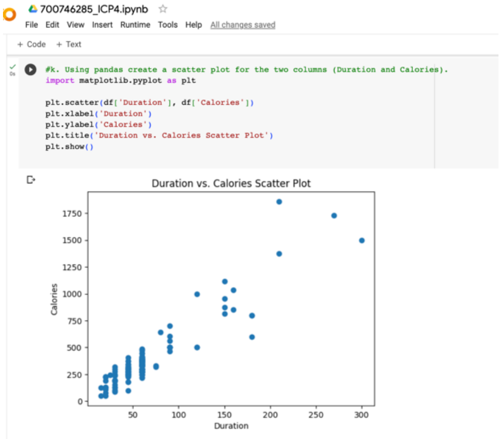


```
#j. Convert the datatype of Calories column to int datatype
df['Calories'] = df['Calories'].astype(int)
```

```
[39] df.dtypes

Duration    int64
Pulse       int64
Calories    int64
dtype: object
```

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

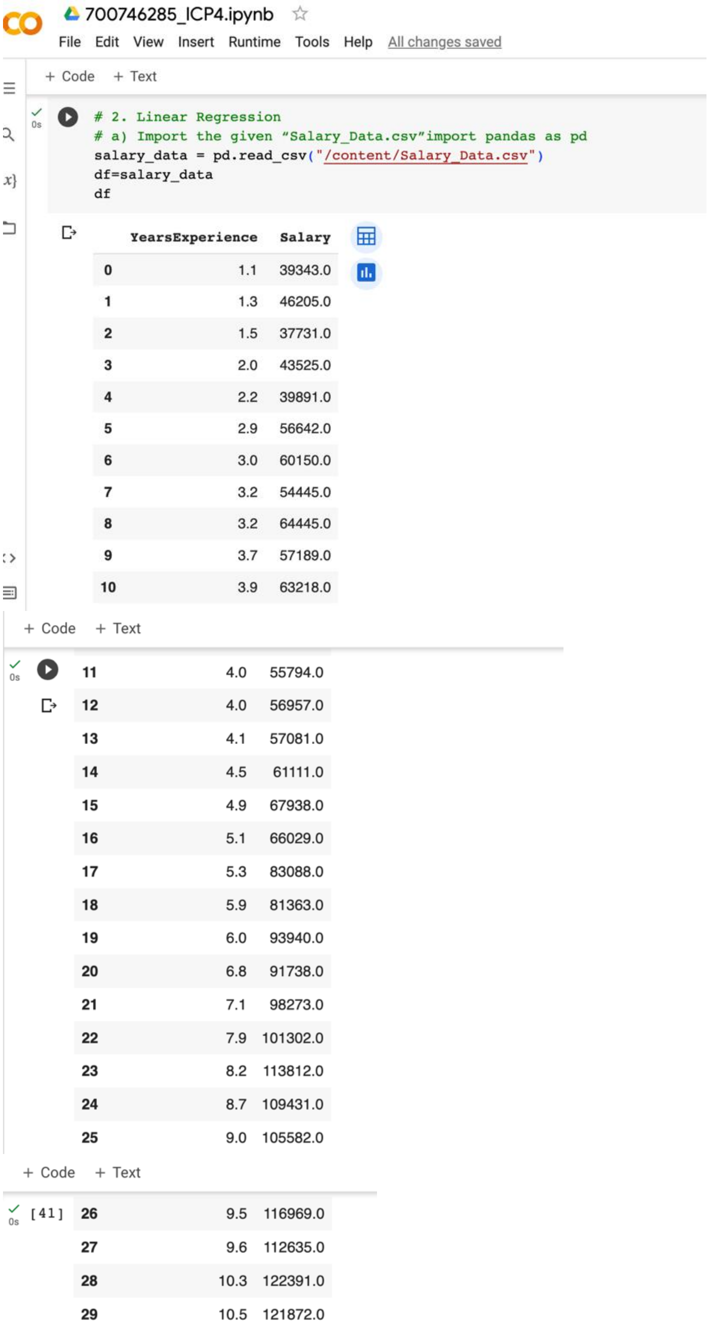



700746285_ICP4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Duration vs. Calories Scatter Plot
1750
1500
1250
1000
750
500
250
0
Calories
50
100
150
200
250
300
Duration

```
#k. Using pandas create a scatter plot for the two columns (Duration and Calories).
import matplotlib.pyplot as plt

plt.scatter(df['Duration'], df['Calories'])
plt.xlabel('Duration')
plt.ylabel('Calories')
plt.title('Duration vs. Calories Scatter Plot')
plt.show()
```

2. Linear Regression

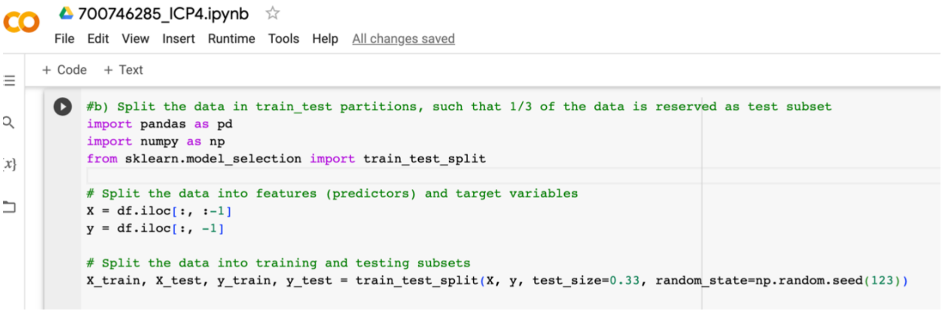
a) Import the given “Salary_Data.csv”



```
# 2. Linear Regression
# a) Import the given “Salary_Data.csv”import pandas as pd
salary_data = pd.read_csv("/content/Salary_Data.csv")
df=salary_data
df
```

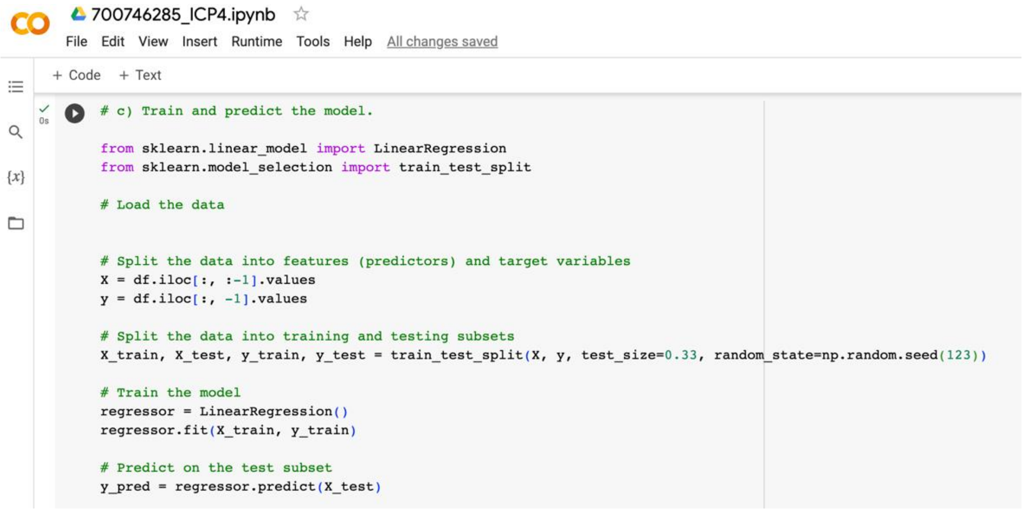
	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

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



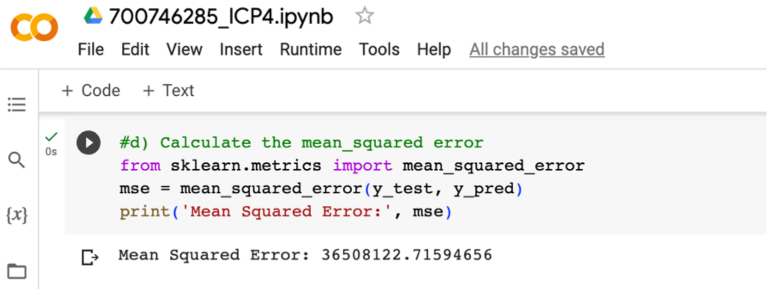
```
#b) Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

# Split the data into features (predictors) and target variables
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Split the data into training and testing subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=np.random.seed(123))
```

c) Train and predict the model.



```
# c) Train and predict the model.

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# Load the data


# Split the data into features (predictors) and target variables
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values

# Split the data into training and testing subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=np.random.seed(123))

# Train the model
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predict on the test subset
y_pred = regressor.predict(X_test)
```

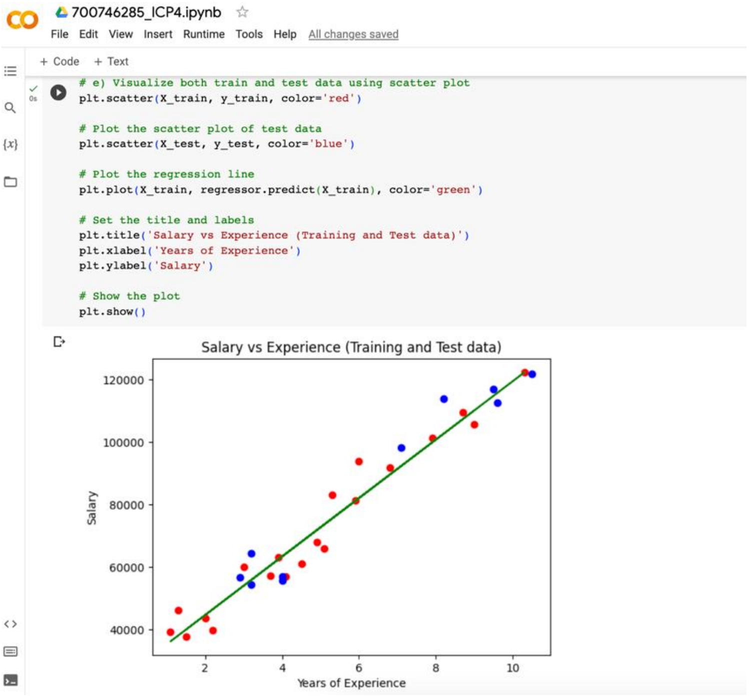
d) Calculate the mean_squared error



```
#d) Calculate the mean_squared error
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error:', mse)
```

```
Mean Squared Error: 36508122.71594656
```

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



700746285_ICP4.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
0s
Salary vs Experience (Training and Test data)
Salary
120000
100000
80000
60000
40000
Years of Experience
2
4
6
8
10

```
# e) Visualize both train and test data using scatter plot
plt.scatter(X_train, y_train, color='red')

# Plot the scatter plot of test data
plt.scatter(X_test, y_test, color='blue')

# Plot the regression line
plt.plot(X_train, regressor.predict(X_train), color='green')

# Set the title and labels
plt.title('Salary vs Experience (Training and Test data)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')

# Show the plot
plt.show()
```