NEURAL NETWORKS ASSIGNMENT 4

NAME : AKHIL CHITTIPOTHULA

ST ID: 700758463

In [1]:
```
import numpy as np
import pandas as pd
```

In [20]:
```
# 1(a) Import the given "Data.csv"
dst_Data = pd.read_csv('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
```


```
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
```


In [21]:
```
#(c) Show the basic statistical description about the data.
dst_Data.head()
```

Out[21]:

	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

In [22]:
```
#(d)Check if the data has null values.
dst_Data.isnull().any()
```

```
dst_Data.isnull().any()
```

Out[22]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories     True
dtype: bool
```

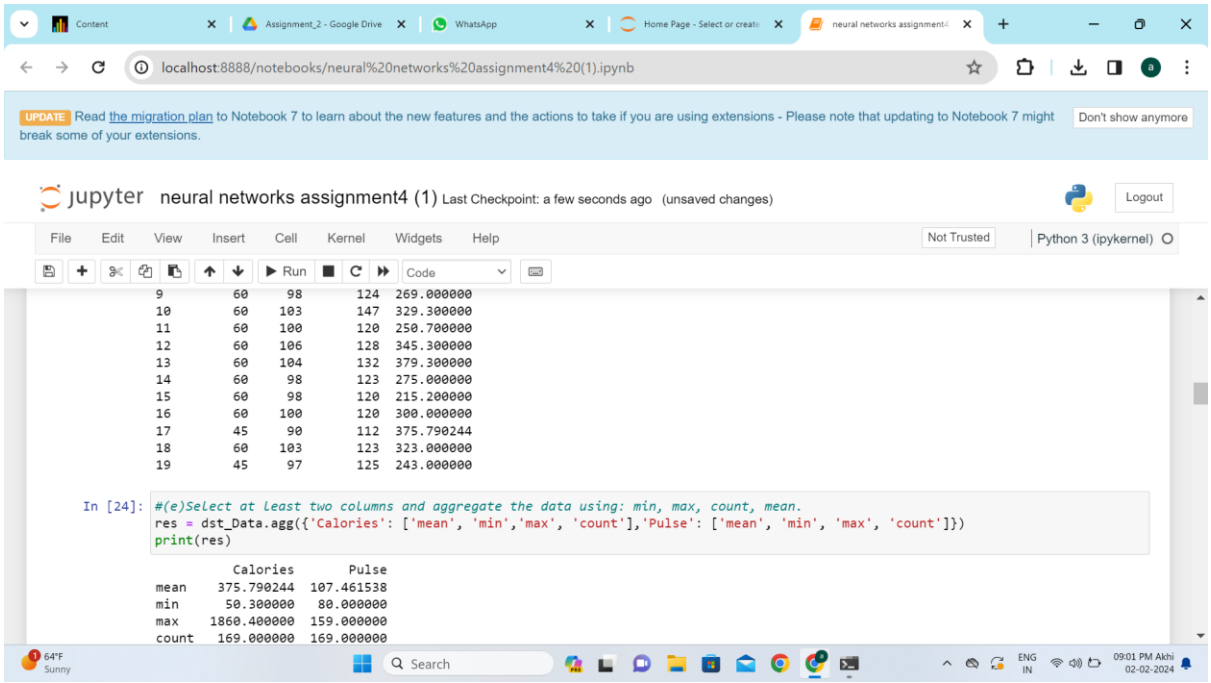
In [23]:
```
dst_Data.fillna(dst_Data.mean(), inplace=True)
dst_Data.isnull().any()
```

Out[23]:
```
Duration    False
Pulse       False
Maxpulse    False
Calories    False
dtype: bool
```

In [11]:
```
#d(i)Replace the null values with the mean
column_means = dst_Data.mean()
print(column_means)
dst Data = dst Data. fillna(column means)
```

```
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
5	60	102	127	300.000000
6	60	110	136	374.000000
7	45	104	134	253.300000
8	30	109	133	195.100000
9	60	98	124	269.000000
10	60	103	147	329.300000
11	60	100	120	250.700000
12	60	106	128	345.300000



In [25]:
```
#(f)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)
#(g)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
```

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

In [26]:
```
#(h)Create a new "df_modified" dataframe that contains all the columns from dst_data except for
#"Maxpulse".
df_modified = dst_Data.loc[:, dst_Data.columns != 'Maxpulse']
print(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 x 3 columns]
```

In [15]:
```
#(i). Delete the "Maxpulse" column from the main dst_data dataframe
dst_Data.drop('Maxpulse', inplace=True, axis=1)
print(dst_Data.dtypes)
```

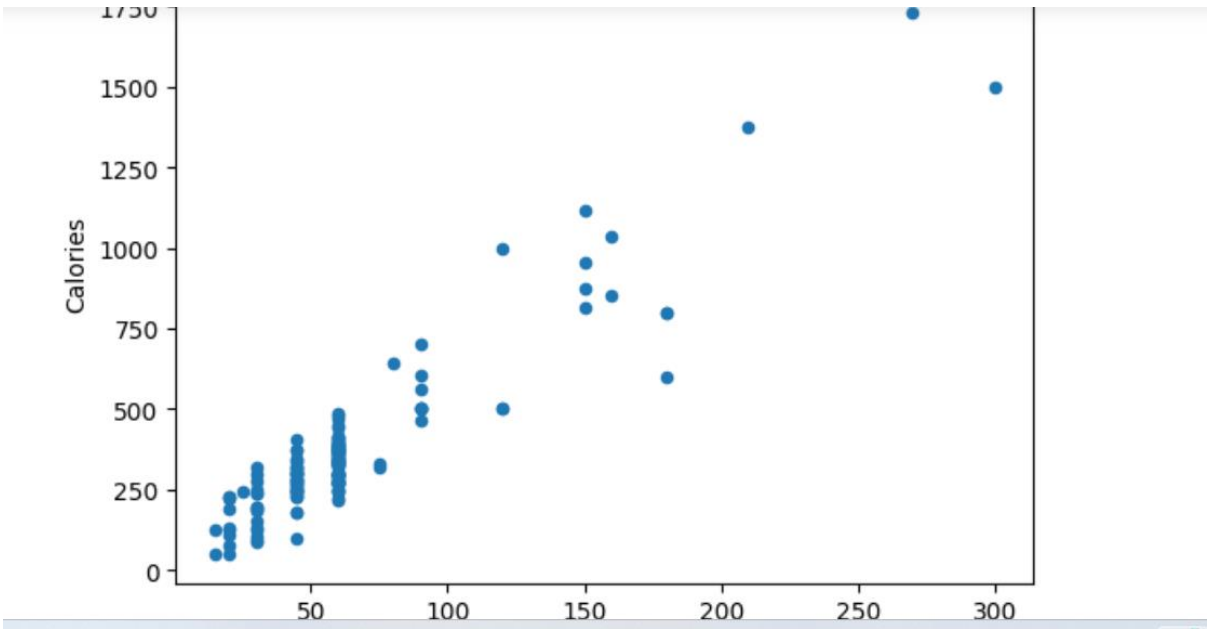
```
Duration      int64
Pulse         int64
Calories    float64
dtype: object
```

In [27]:
```
#(j). Convert the datatype of Calories column to int datatype
dst_Data["Calories"] = dst_Data["Calories"].astype(float).astype(int)
print(dst_Data.dtypes)
```

```
Duration    int64
Pulse       int64
Maxpulse    int64
Calories    int32
dtype: object
```

In [28]:
```
#(k)Using pandas create a scatter plot for the two columns (Duration and Calories).
as1 = dst_Data.plot.scatter(x='Duration',y='Calories')
print(as1)
```

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



2(a) Import the given “Salary_Data.csv”

dst_Sal = pd.read_csv('Salary_Data.csv') dst_Sal.info() dst_Sal.head()

In [34]:
```
A = dst_Sal.iloc[:, :-1].values   #excluding last column i.e., years of experience column
B = dst_Sal.iloc[:, 1].values     #only salary column
```

In [35]:
```
# (b) 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)
```

In [36]:
```
# (c) 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
```

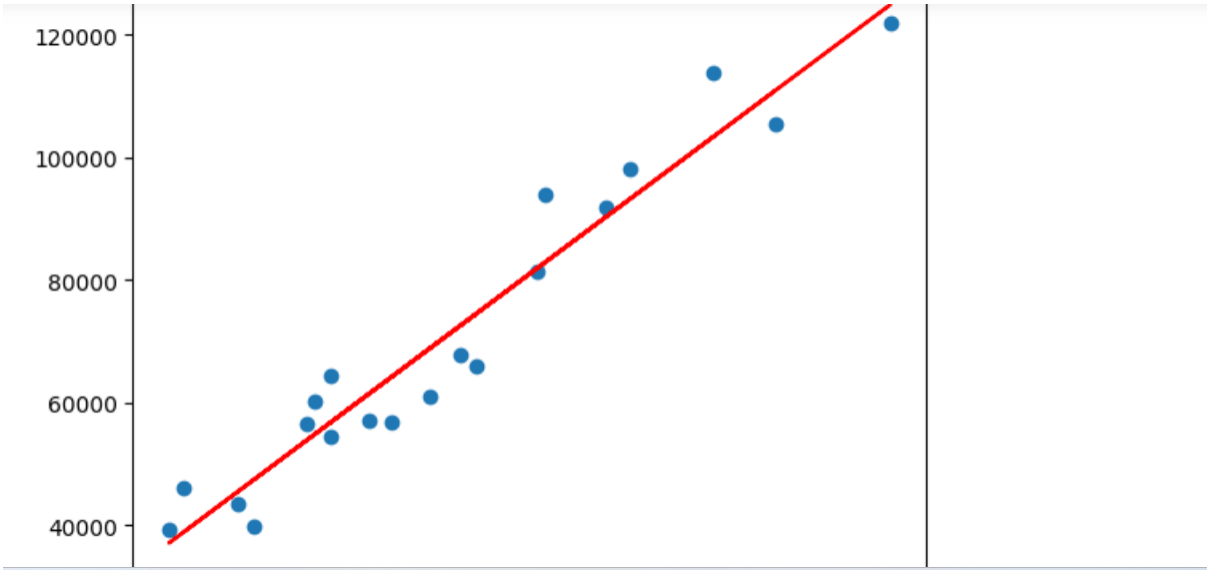
Out[36]:
```
array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

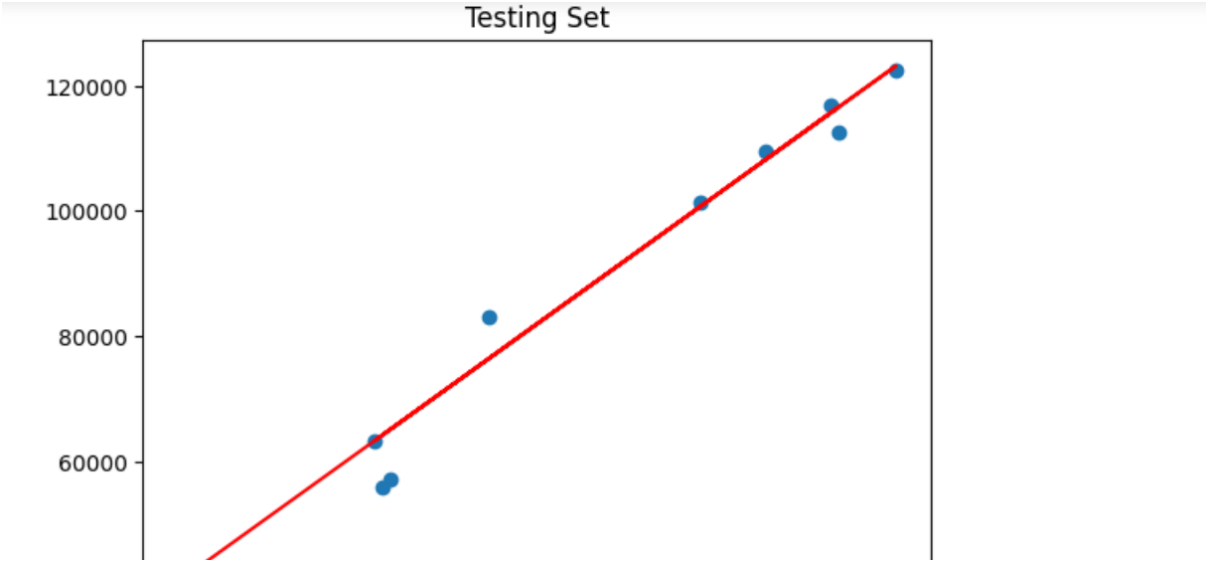
In [37]:
```
# (d) Calculate the mean_squared error
S_error = (B_Pred - B_test) ** 2
Sum_Serror = np.sum(S_error)
mean_squared_error = Sum_Serror / B_test.size
mean_squared_error
```

Out[37]:
```
21026037.329511296
```

In [38]:
```
# (e) 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()
```





VEDIO LINK:
https://drive.google.com/file/d/16pE3xz9KLMblVE_9vF_Tj3O9wCOUNbYl/view?usp=sharing