

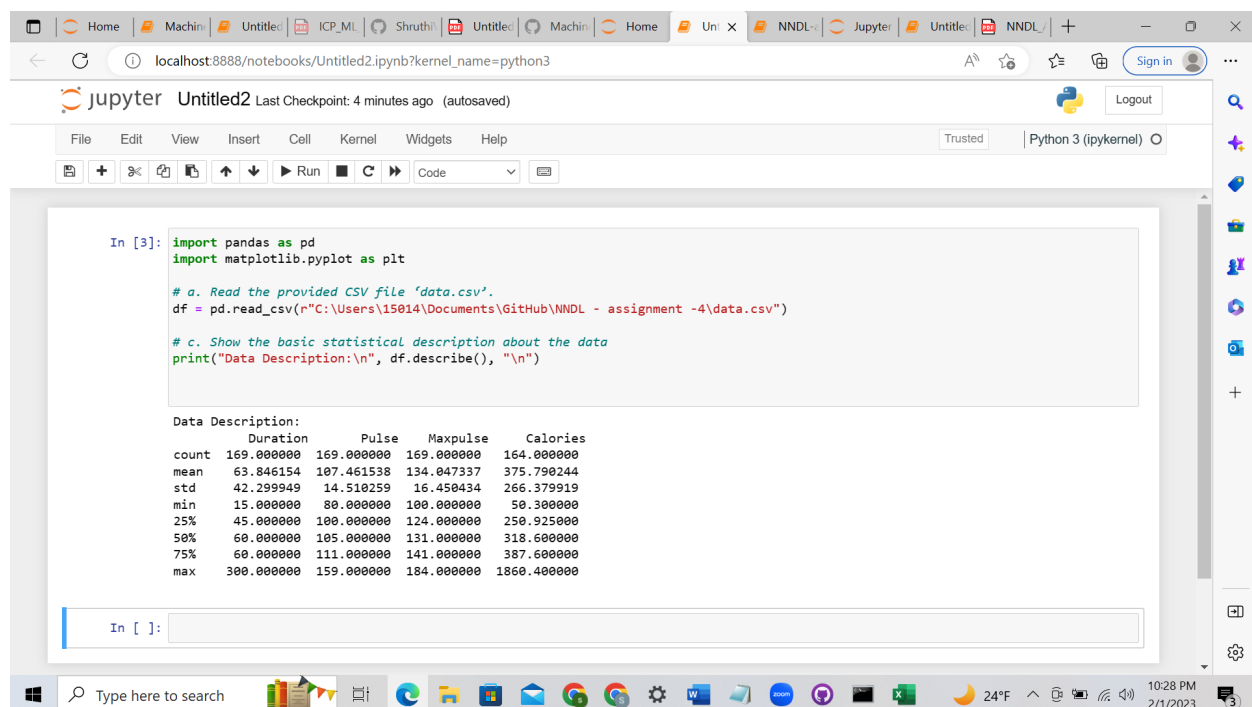
Spring 2023: CS5720 – NN &DL

In-Class Programming Assignment-2

1. Data Manipulation

- Read the provided CSV file 'data.csv'.
- <https://drive.google.com/drive/folders/1h8C3mLsso-R-sIOLsvoYwPLzy2fJ4IOF?usp=sharing>
- Show the basic statistical description about the data.
- Check if the data has null values.
- Replace the null values with the mean
- Select at least two columns and aggregate the data using: min, max, count, mean.
- Filter the dataframe to select the rows with calories values between 500 and 1000.
- Filter the dataframe to select the rows with calories values > 500 and pulse < 100.
- Create a new "df_modified" dataframe that contains all the columns from df except for "Maxpulse".
- Delete the "Maxpulse" column from the main df dataframe
- Convert the datatype of Calories column to int datatype.
- Using pandas create a scatter plot for the two columns (Duration and Calories).

Ans



```
In [3]: import pandas as pd
import matplotlib.pyplot as plt

# a. Read the provided CSV file 'data.csv'.
df = pd.read_csv(r"C:\Users\15014\Documents\GitHub\NNDL - assignment -4\data.csv")

# c. Show the basic statistical description about the data
print("Data Description:\n", df.describe(), "\n")
```

Data Description:

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

- Read the data from data.csv at the path C:\Users\15014\Documents\GitHub\NNDL - assignment -4\data.csv
- Using describe() method got the data description & printed the data description.

A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3'. The notebook title is 'Untitled2' with a note 'Last Checkpoint: 5 minutes ago (unsaved changes)'. The code cell 'In [4]:' contains two steps: first, checking for null values with 'df.isnull().sum()', and second, replacing null values with the mean using 'df.fillna(df.mean(), inplace=True)'. The output shows 'Missing Values:' for Duration, Pulse, and Maxpulse (all 0), and Calories (5). The second output shows the 'Data after Replacing Missing Values with Mean:' as a table with 169 rows and 4 columns: Duration, Pulse, Maxpulse, and Calories. The table lists values for rows 0 through 168.

```
In [4]: # d. Check if the data has null values
print("Missing Values:\n", df.isnull().sum(), "\n")

# i. Replace the null values with the mean
df.fillna(df.mean(), inplace=True)
print("Data after Replacing Missing Values with Mean:\n", df, "\n")
```

Missing Values:

Duration	0
Pulse	0
Maxpulse	0
Calories	5

dtype: int64

Data after Replacing Missing Values with Mean:

	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 x 4 columns]

d) Checked if data has null values and replaced the null values with the mean using .fillna() method

A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3'. The notebook title is 'Untitled2' with a note 'Last Checkpoint: 6 minutes ago (unsaved changes)'. The code cell 'In [7]:' shows data aggregation for 'Calories' and 'Pulse' using 'grouped = df[selected_cols].agg(['min', 'max', 'count', 'mean'])'. The output shows 'Aggregated Data:' as a table with min, max, count, and mean for both columns. The code cell 'In [6]:' shows filtering the dataframe to rows with 'Calories' values between 500 and 1000 using 'filtered_df = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]'. The output shows 'Data with Calorie values between 500 and 1000:'.

```
In [7]: # e. Select at Least two columns and aggregate the data using: min, max, count, mean
selected_cols = ['Calories', 'Pulse']
grouped = df[selected_cols].agg(['min', 'max', 'count', 'mean'])
print("Aggregated Data:\n", grouped, "\n")
```

Aggregated Data:

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

```
In [6]: # f. Filter the dataframe to select the rows with calories values between 500 and 1000
filtered_df = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]
print("Data with Calorie values between 500 and 1000:\n", filtered_df, "\n")
```

Data with Calorie values between 500 and 1000:

e) Selected the two columns and aggregated the data using min , max,count , mean using .agg() method

The screenshot shows a Jupyter Notebook interface with a code cell executed. The code filters a dataframe to select rows where 'Calories' is between 500 and 1000. The output displays a table of the filtered data.

```
In [6]: # f. Filter the dataframe to select the rows with calories values between 500 and 1000
filtered_df = df[(df['Calories'] >= 500) & (df['Calories'] <= 1000)]
print("Data with Calorie values between 500 and 1000:\n", filtered_df, "\n")
```

	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

f) Filtered the data frame to select the rows with calories values between 500 & 1000

The screenshot shows a Jupyter Notebook interface with a code cell executed. The code filters a dataframe to select rows where 'Calories' is greater than 500 and 'Pulse' is less than 100. The output displays a table of the filtered data.

```
In [10]: # g. Filter the dataframe to select the rows with calories values > 500 and pulse < 100
filtered_df = df[(df['Calories'] > 500) & (df['Pulse'] < 100)]
print("Data with Calorie values > 500 and Pulse < 100:\n", filtered_df, "\n")
```

	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

g) Filtered the data frame to select the rows with calories values > 500 & pulse < 100

The screenshot shows a Jupyter Notebook interface with a code cell executed. The code creates a new dataframe 'df_modified' by dropping the 'Maxpulse' column from the original dataframe 'df'. The output displays the first few rows of the modified dataframe, showing columns 'Duration', 'Pulse', and 'Calories'.

```
In [9]: # h. Create a new "df_modified" dataframe that contains all the columns from df except for "Maxpulse"
df_modified = df.drop(['Maxpulse'], axis=1)
print("Modified Dataframe without Maxpulse Column:\n", df_modified, "\n")
```

Modified Dataframe without Maxpulse Column:

	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]

'h) created the df_modified dataframe that contains all the columns from df except for "Maxpulse" using drop() function.

The screenshot shows a Jupyter Notebook interface with a code cell executed. The code deletes the 'Maxpulse' column from the main dataframe 'df' using the drop() function with inplace=True. The output displays the first few rows of the modified dataframe, showing columns 'Duration', 'Pulse', and 'Calories'.

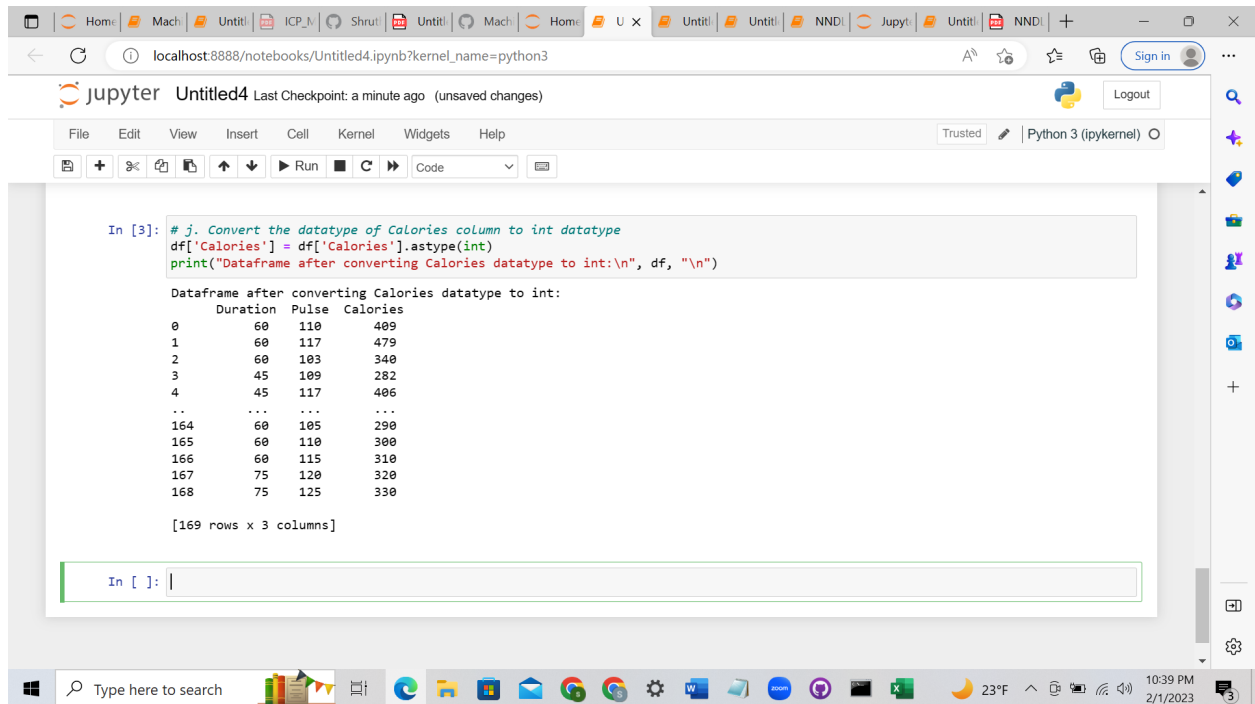
```
In [2]: # i. Delete the "Maxpulse" column from the main df dataframe
df.drop(['Maxpulse'], axis=1, inplace=True)
print("Dataframe after deleting Maxpulse Column:\n", df, "\n")
```

Dataframe after deleting Maxpulse Column:

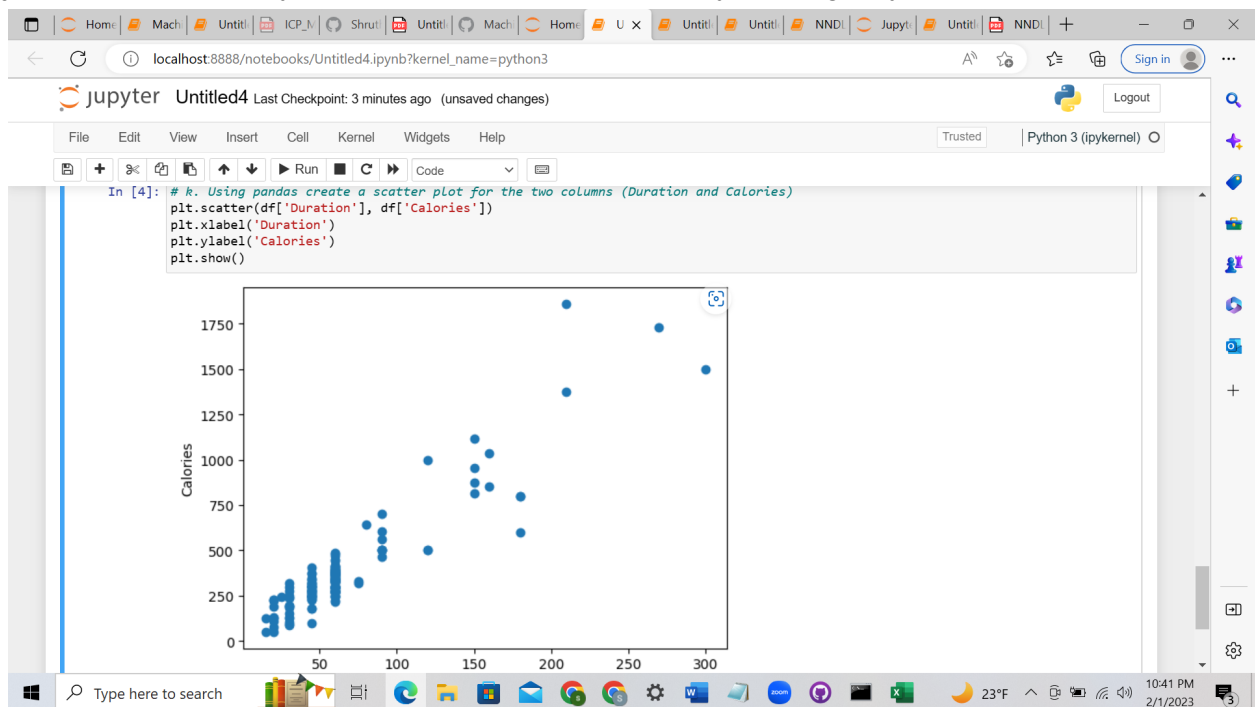
	Duration	Pulse	Calories
0	60	110	409.1
1	60	117	479.0
2	60	103	340.0
3	45	109	282.4
4	45	117	406.0
...
164	60	105	290.8
165	60	110	300.0
166	60	115	310.2
167	75	120	320.4
168	75	125	330.4

[169 rows x 3 columns]

i) Deleted the Maxpulse column from the main df dataframe and printed the output using drop() function.



j) converted the datatype of calories column to int datatype using astype() function

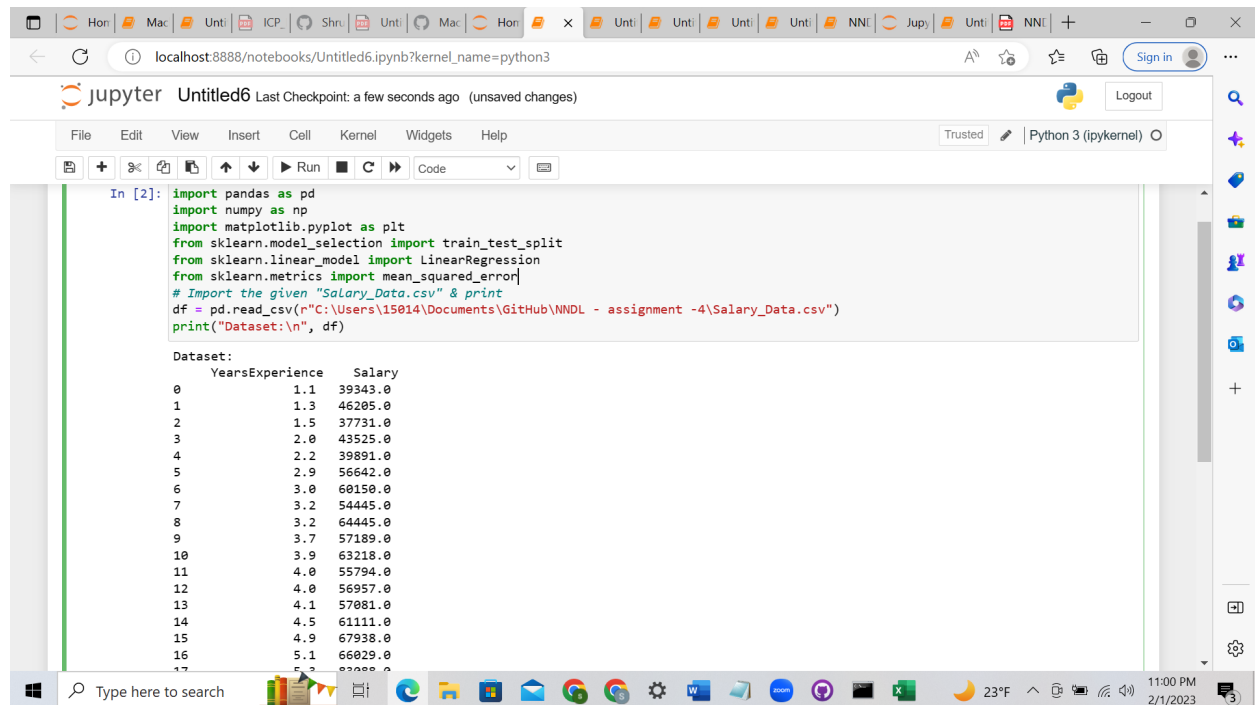


k) Using pandas created a scatter plot for the 2 columns Duration & calories using Scatter() function

2. Linear Regression

- Import the given "Salary_Data.csv"
- Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset.
- Train and predict the model.
- Calculate the mean_squared error
- Visualize both train and test data using scatter plot

Ans



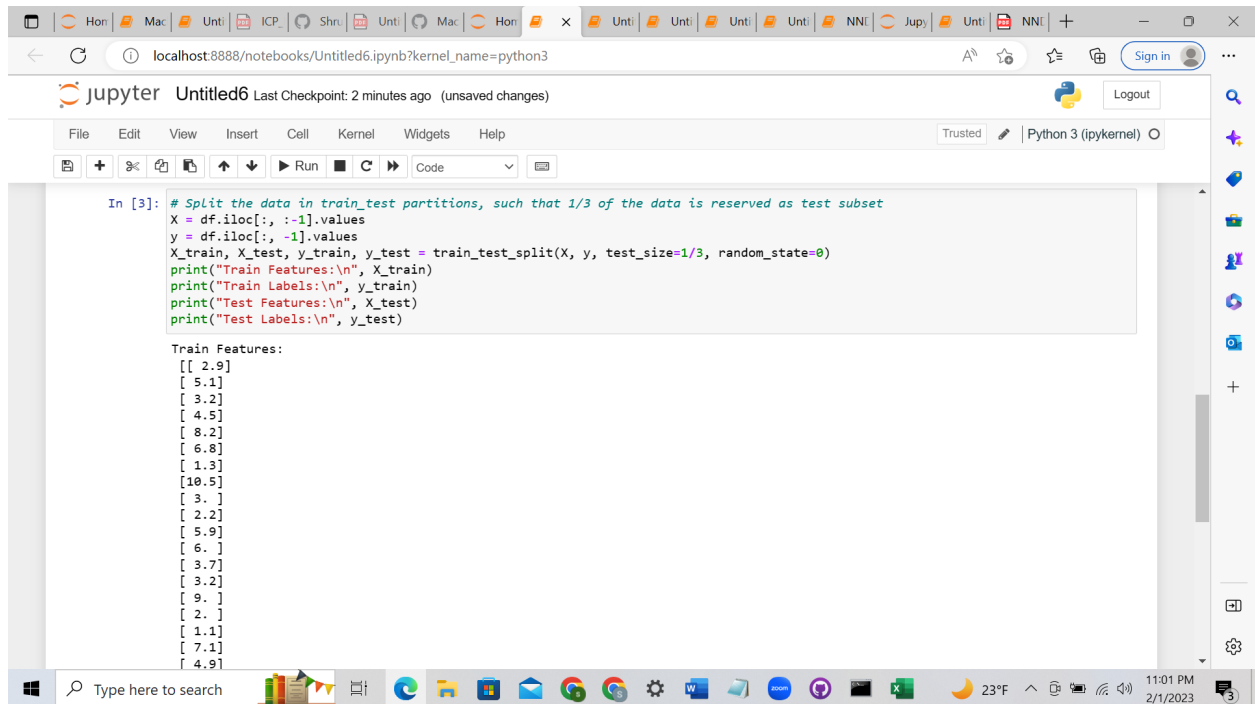
The screenshot shows a Jupyter Notebook window titled 'Untitled6' with a Python 3 kernel. The code in the first cell imports pandas, numpy, matplotlib, and sklearn. It then reads a CSV file named 'Salary_Data.csv' and prints the resulting DataFrame. The output shows a dataset with two columns: 'YearsExperience' and 'Salary'.

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# Import the given "Salary_Data.csv" & print
df = pd.read_csv(r"C:\Users\15014\Documents\GitHub\NNDL - assignment -4\Salary_Data.csv")
print("Dataset:\n", df)
```

Dataset:

	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	5.2	67000.0

- Imported the data set using pd.read_csv() function

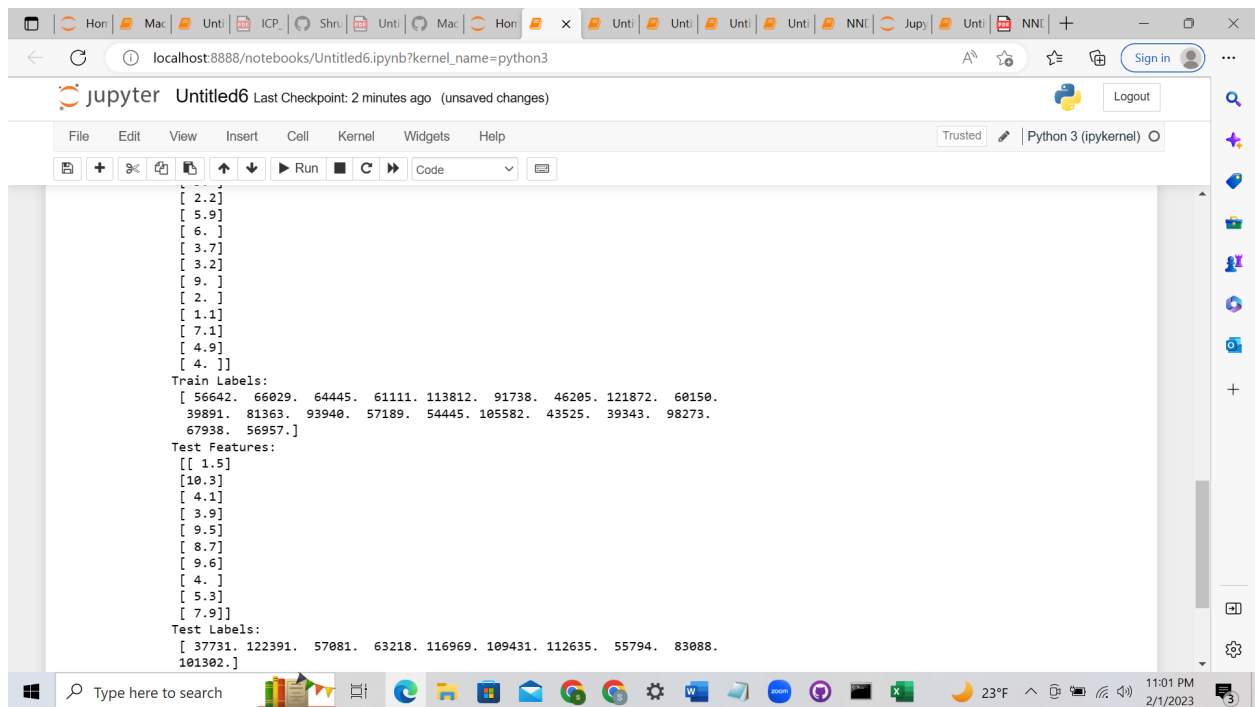


The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code:

```
In [3]: # Split the data in train_test partitions, such that 1/3 of the data is reserved as test subset
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3, random_state=0)
print("Train Features:\n", X_train)
print("Train Labels:\n", y_train)
print("Test Features:\n", X_test)
print("Test Labels:\n", y_test)
```

The output of the code cell shows the 'Train Features' as a 15x2 array of numerical values:

```
Train Features:
[[ 2.9]
 [ 5.1]
 [ 3.2]
 [ 4.5]
 [ 8.2]
 [ 6.8]
 [ 1.3]
 [10.5]
 [ 3. ]
 [ 2.2]
 [ 5.9]
 [ 6. ]
 [ 3.7]
 [ 3.2]
 [ 9. ]
 [ 2. ]
 [ 1.1]
 [ 7.1]
 [ 4.9]
```



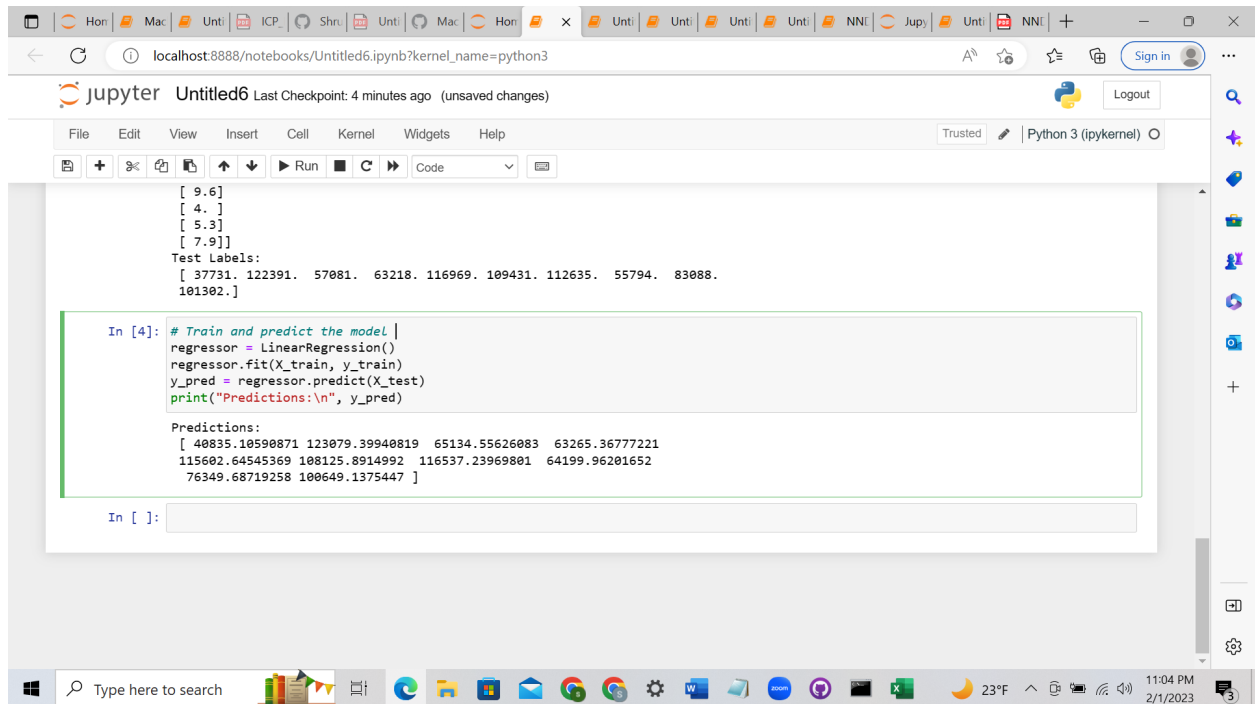
The screenshot shows the continuation of the Jupyter Notebook output. The 'Train Labels' are displayed as a 1D array of 15 numerical values:

```
Train Labels:
[ 56642.  66029.  64445.  61111. 113812.  91738.  46205. 121872.  60150.
 39891.  81363.  93940.  57189.  54445. 105582.  43525.  39343.  98273.
 67938.  56957.]
```

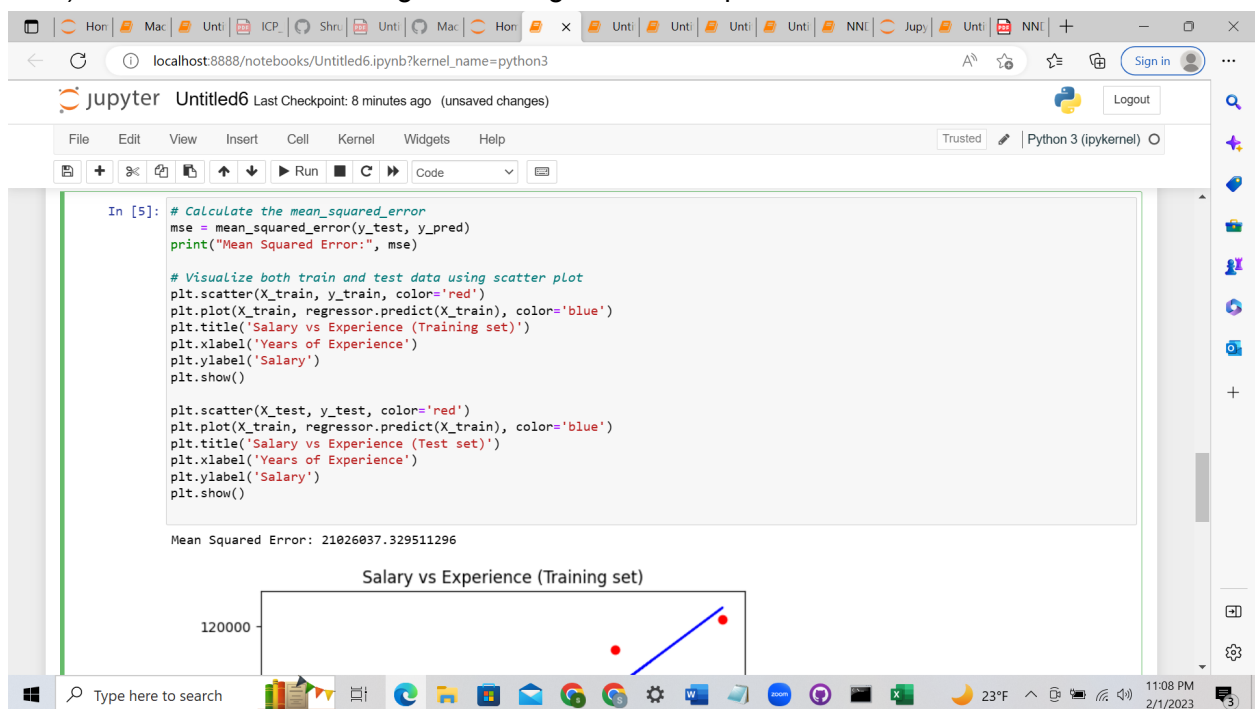
Following this, the 'Test Features' are shown as a 10x2 array of numerical values:

```
Test Features:
[[ 1.5]
 [10.3]
 [ 4.1]
 [ 3.9]
 [ 9.5]
 [ 8.7]
 [ 9.6]
 [ 4. ]
 [ 5.3]
 [ 7.9]]
```

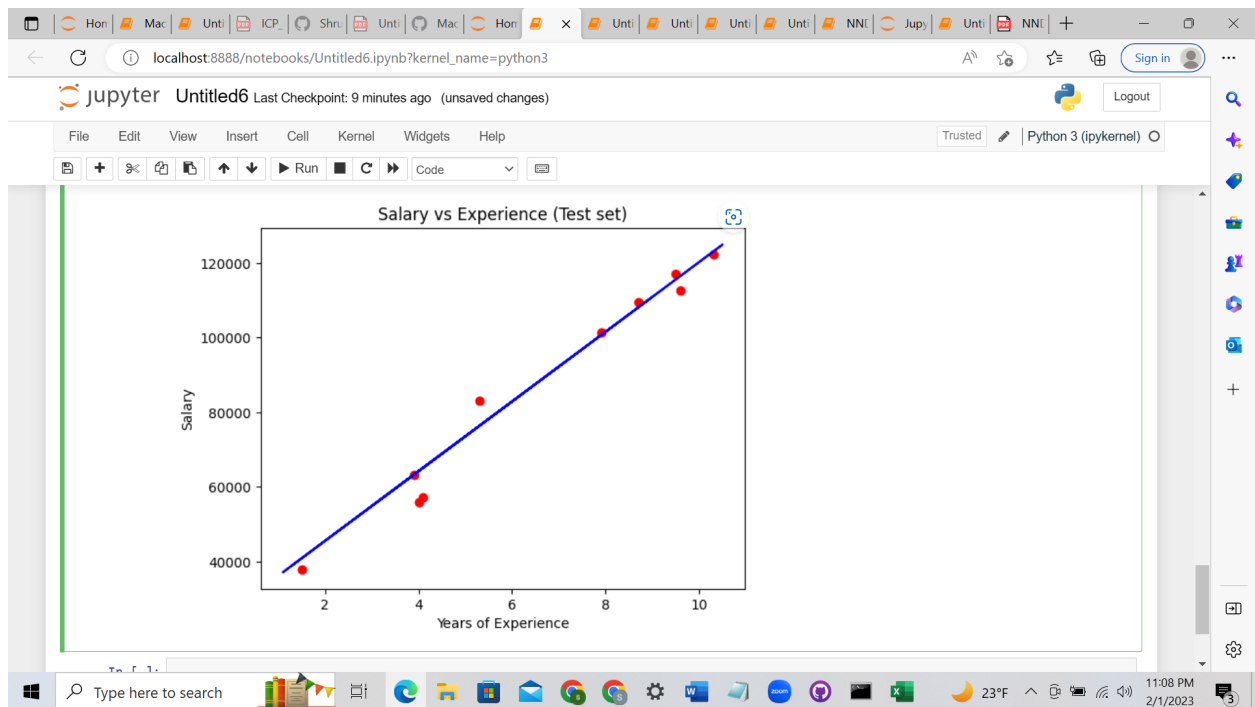
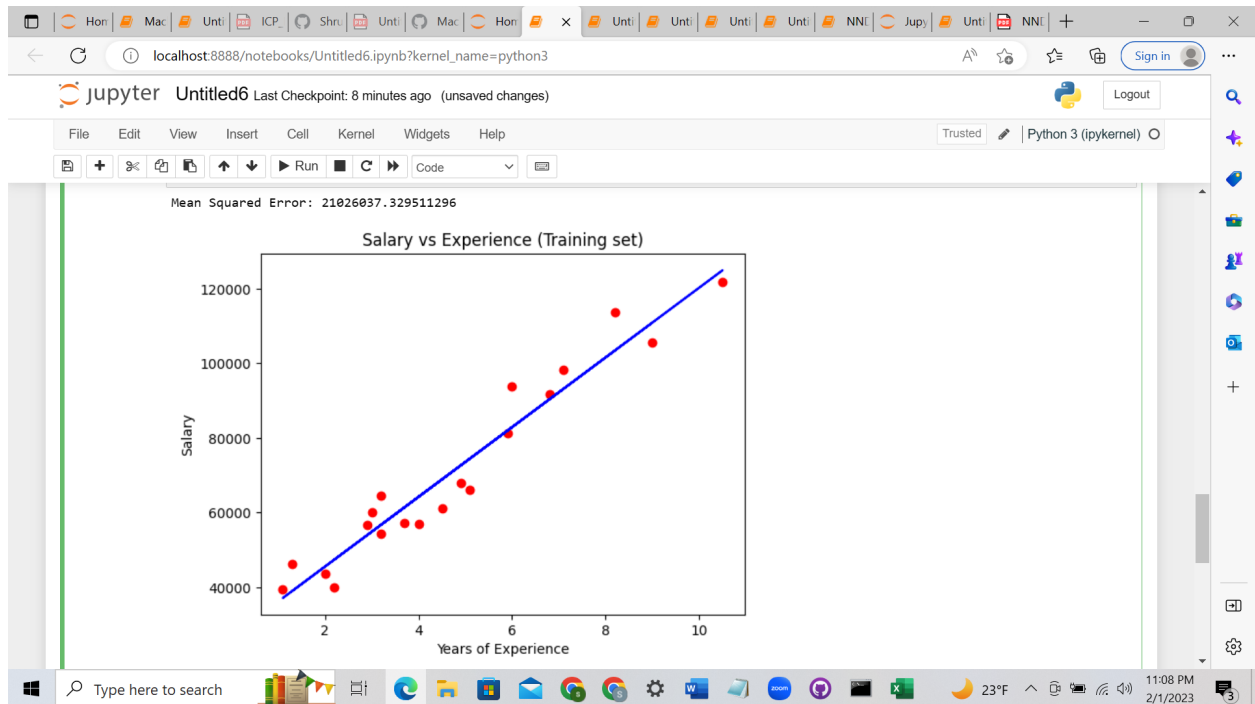
b) Splitted the data in to train_test partitions, such that 1/3 of the data is reserved as test subset using iloc() function



c) Trained the model using Linear Regression and predicted the model



d) Calculated the mean Squared Error using mean_squared_error() using function



e) Visualized the data set and train set using the scatter() function

Git hub repo : <https://github.com/ShruthiVallapReddy/NNDL-assignment---4.git>

