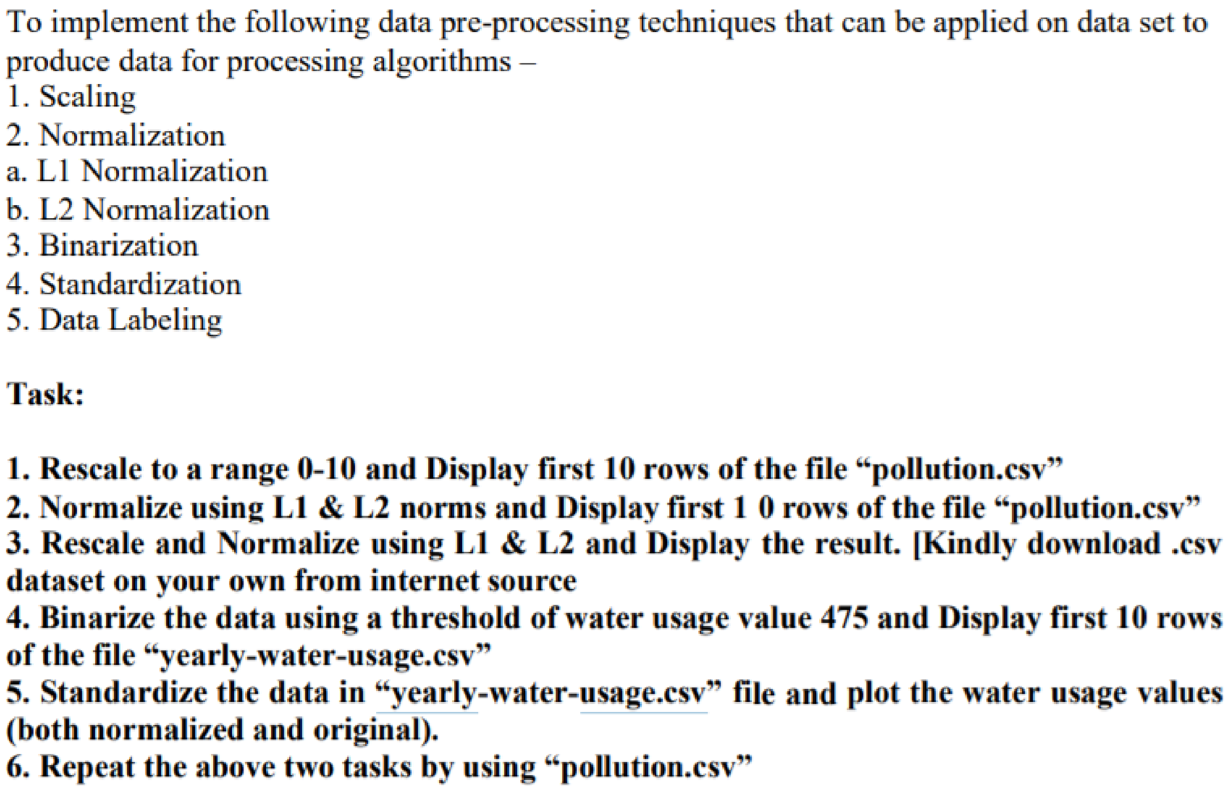
Name: **Swarup Tripathy** Assignment Number: **1**
Course: **ECE3502: IoT Domain Analyst** Date: February 9, 2022

Reg No: 19BEE0167

Problem 1



To implement the following data pre-processing techniques that can be applied on data set to produce data for processing algorithms –
1. Scaling
2. Normalization
a. L1 Normalization
b. L2 Normalization
3. Binarization
4. Standardization
5. Data Labeling

Task:

1. Rescale to a range 0-10 and Display first 10 rows of the file "pollution.csv"
2. Normalize using L1 & L2 norms and Display first 1 0 rows of the file "pollution.csv"
3. Rescale and Normalize using L1 & L2 and Display the result. [Kindly download .csv dataset on your own from internet source
4. Binarize the data using a threshold of water usage value 475 and Display first 10 rows of the file "yearly-water-usage.csv"
5. Standardize the data in "yearly-water-usage.csv" file and plot the water usage values (both normalized and original).
6. Repeat the above two tasks by using "pollution.csv"

1 Python Code

```
# SCALING
from pandas import read_csv
from numpy import set_printoptions
from sklearn import preprocessing

# Read the CSV file and prepare the array
names = ['No','year','month','day','hour','pm2.5','DEWP','TEMP','PRES','Iws','Is','Ir']
dataframep = read_csv("pollution.csv", names=names)
array = dataframep.values

# Using MinMaxScaler class to rescale the data in the range of 0 and 1.
data_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
```

```
# ( X - min(X) )/( max(X) - min(X) )
data_rescaled = data_scaler.fit_transform(array)

# Setting the precision to 2
set_printoptions(precision=2)

# Showing the first 10 rows in the output
print ("\nScaled data:\n", data_rescaled[0:10])

print(dataframep.head(10))
```

1.1 Code output

```
    Scaled data:
 [[0.   0.   0.   0.03 0.   0.32 0.34 0.36 0.47 0.   0.   0.  ]
 [0.   0.   0.   0.03 0.04 0.37 0.38 0.36 0.47 0.01 0.   0.  ]
 [0.   0.   0.   0.03 0.09 0.39 0.5  0.33 0.5  0.01 0.   0.  ]
 [0.   0.   0.   0.03 0.13 0.45 0.62 0.33 0.53 0.01 0.04 0.  ]
 [0.01 0.   0.   0.03 0.17 0.34 0.62 0.33 0.53 0.02 0.07 0.  ]
 [0.01 0.   0.   0.03 0.22 0.27 0.62 0.31 0.53 0.02 0.11 0.  ]
 [0.01 0.   0.   0.03 0.26 0.26 0.62 0.31 0.55 0.03 0.15 0.  ]
 [0.01 0.   0.   0.03 0.3  0.31 0.62 0.33 0.58 0.03 0.   0.  ]
 [0.01 0.   0.   0.03 0.35 0.3  0.59 0.31 0.58 0.04 0.   0.  ]
 [0.01 0.   0.   0.03 0.39 0.33 0.62 0.33 0.61 0.04 0.   0.  ]]

    No  year  month  day  hour  pm2.5  DEWP  TEMP  PRES    Iws  Is  Ir
0    1  2010      1    2     0    129   -16    -4  1020   1.79   0   0
1    2  2010      1    2     1    148   -15    -4  1020   2.68   0   0
2    3  2010      1    2     2    159   -11    -5  1021   3.57   0   0
3    4  2010      1    2     3    181    -7    -5  1022   5.36   1   0
4    5  2010      1    2     4    138    -7    -5  1022   6.25   2   0
5    6  2010      1    2     5    109    -7    -6  1022   7.14   3   0
6    7  2010      1    2     6    105    -7    -6  1023   8.93   4   0
7    8  2010      1    2     7    124    -7    -5  1024  10.72   0   0
8    9  2010      1    2     8    120    -8    -6  1024  12.51   0   0
9   10  2010      1    2     9    132    -7    -5  1025  14.30   0   0
```

2 Python Code

```
# The code for L1 Normalization is

'''
It may be defined as the normalization technique that modifies the dataset values in a way
that in each row the sum of the absolute values will always be up to 1.
It is also called Least Absolute Deviations.
'''

from pandas import read_csv
import numpy as np
from numpy import set_printoptions
from sklearn.preprocessing import Normalizer

# Using Normalizer class with L1 to normalize the data.
Data_normalizer1 = Normalizer(norm=’l1’).fit(array)
Data_normalizer2 = Normalizer(norm=’l2’).fit(array)
Data_normalized1 = Data_normalizer1.transform(array)
Data_normalized2 = Data_normalizer2.transform(array)

# Setting the precision to 2
set_printoptions(precision=2)

# Showing the first 10 rows in the output
print ("\nNormalized data:\n", Data_normalized1 [0:10])
print ("\nNormalized data:\n", Data_normalized2 [0:10])
```

2.1 Code output

```
Normalized l1 data:
 [[ 3.14e-04  6.31e-01  3.14e-04  6.28e-04  0.00e+00  4.05e-02 -5.02e-03
  -1.26e-03  3.20e-01  5.62e-04  0.00e+00  0.00e+00]
 [ 6.24e-04  6.27e-01  3.12e-04  6.24e-04  3.12e-04  4.62e-02 -4.68e-03
  -1.25e-03  3.18e-01  8.36e-04  0.00e+00  0.00e+00]
 [ 9.32e-04  6.25e-01  3.11e-04  6.22e-04  6.22e-04  4.94e-02 -3.42e-03
  -1.55e-03  3.17e-01  1.11e-03  0.00e+00  0.00e+00]
 [ 1.23e-03  6.20e-01  3.09e-04  6.17e-04  9.26e-04  5.58e-02 -2.16e-03
  -1.54e-03  3.15e-01  1.65e-03  3.09e-04  0.00e+00]
 [ 1.56e-03  6.28e-01  3.12e-04  6.25e-04  1.25e-03  4.31e-02 -2.19e-03
  -1.56e-03  3.19e-01  1.95e-03  6.25e-04  0.00e+00]
 [ 1.89e-03  6.32e-01  3.15e-04  6.29e-04  1.57e-03  3.43e-02 -2.20e-03
  -1.89e-03  3.22e-01  2.25e-03  9.44e-04  0.00e+00]
 [ 2.20e-03  6.32e-01  3.14e-04  6.29e-04  1.89e-03  3.30e-02 -2.20e-03
  -1.89e-03  3.22e-01  2.81e-03  1.26e-03  0.00e+00]
 [ 2.50e-03  6.28e-01  3.13e-04  6.25e-04  2.19e-03  3.88e-02 -2.19e-03
  -1.56e-03  3.20e-01  3.35e-03  0.00e+00  0.00e+00]
 [ 2.81e-03  6.28e-01  3.12e-04  6.25e-04  2.50e-03  3.75e-02 -2.50e-03
  -1.87e-03  3.20e-01  3.91e-03  0.00e+00  0.00e+00]
 [ 3.11e-03  6.25e-01  3.11e-04  6.22e-04  2.80e-03  4.11e-02 -2.18e-03
  -1.56e-03  3.19e-01  4.45e-03  0.00e+00  0.00e+00]]

  Normalized l2 data:
 [[ 4.43e-04  8.90e-01  4.43e-04  8.86e-04  0.00e+00  5.71e-02 -7.09e-03
  -1.77e-03  4.52e-01  7.93e-04  0.00e+00  0.00e+00]
 [ 8.85e-04  8.90e-01  4.43e-04  8.85e-04  4.43e-04  6.55e-02 -6.64e-03
  -1.77e-03  4.52e-01  1.19e-03  0.00e+00  0.00e+00]
 [ 1.33e-03  8.89e-01  4.42e-04  8.85e-04  8.85e-04  7.04e-02 -4.87e-03
  -2.21e-03  4.52e-01  1.58e-03  0.00e+00  0.00e+00]
 [ 1.77e-03  8.89e-01  4.42e-04  8.84e-04  1.33e-03  8.00e-02 -3.09e-03
  -2.21e-03  4.52e-01  2.37e-03  4.42e-04  0.00e+00]
 [ 2.21e-03  8.90e-01  4.43e-04  8.85e-04  1.77e-03  6.11e-02 -3.10e-03
  -2.21e-03  4.52e-01  2.77e-03  8.85e-04  0.00e+00]
 [ 2.66e-03  8.90e-01  4.43e-04  8.86e-04  2.21e-03  4.83e-02 -3.10e-03
  -2.66e-03  4.53e-01  3.16e-03  1.33e-03  0.00e+00]
 [ 3.10e-03  8.90e-01  4.43e-04  8.86e-04  2.66e-03  4.65e-02 -3.10e-03
  -2.66e-03  4.53e-01  3.96e-03  1.77e-03  0.00e+00]
 [ 3.54e-03  8.90e-01  4.43e-04  8.85e-04  3.10e-03  5.49e-02 -3.10e-03
  -2.21e-03  4.53e-01  4.74e-03  0.00e+00  0.00e+00]
 [ 3.98e-03  8.90e-01  4.43e-04  8.85e-04  3.54e-03  5.31e-02 -3.54e-03
  -2.66e-03  4.53e-01  5.54e-03  0.00e+00  0.00e+00]
 [ 4.42e-03  8.89e-01  4.42e-04  8.85e-04  3.98e-03  5.84e-02 -3.10e-03
  -2.21e-03  4.53e-01  6.33e-03  0.00e+00  0.00e+00]]
```

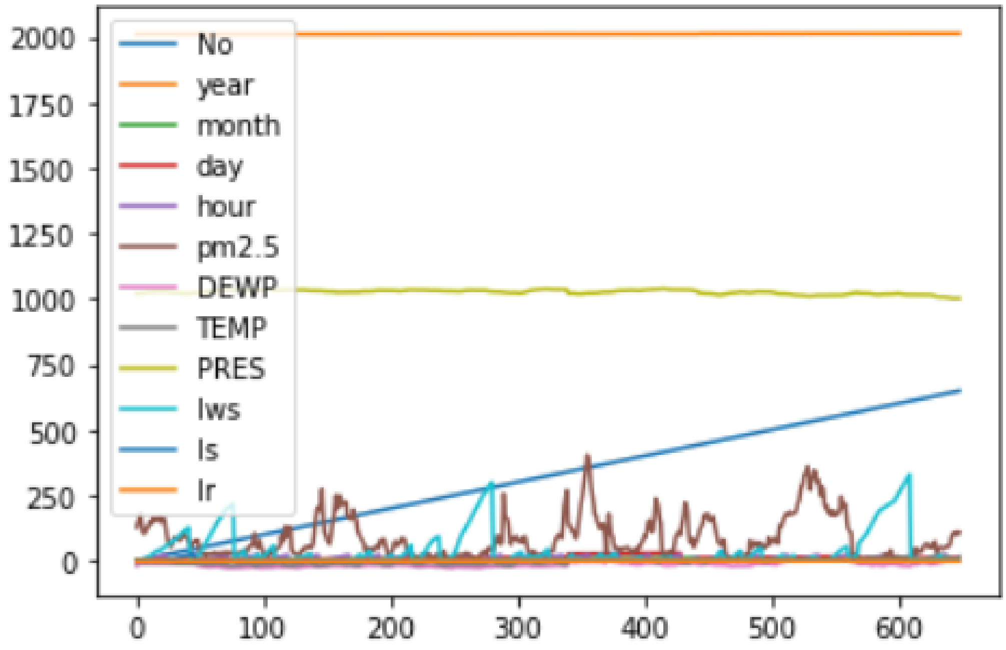
3 Python Code

```
    # BINARIZATION

'''
the process of dividing data into two groups and assigning one out. of two values to all
the members of the same group.
This is usually accomplished by defining a threshold t and assigning the value 0 to all
the data points below the threshold and 1 to those above it.
'''

from sklearn.preprocessing import Binarizer

# Using Binarize class to convert the data into binary values.
binarizer = Binarizer(threshold=0.5).fit(array)
Data_binarized = binarizer.transform(array)

# Showing the first 3 rows in the output
print ("\nBinary data:\n", Data_binarized [0:10])

dataframep = read_csv("pollution.csv", names=names)
dataframep.plot()
```

3.1 Code output

```
Binary data:
 [[1. 1. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0.]
 [1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0.]]
```

```
<AxesSubplot:>
```



4 Python Code for Yearly Water Usage

```
dataframew = read_csv("yearly-water-usage.csv", names=names)
print(dataframew.head(10))


names = ['Year','Water']
dataframew = read_csv("yearly-water-usage.csv", names=names)
array = dataframew.values

# Using Binarize class to convert the data into binary values.
binarizer = Binarizer(threshold=0.5).fit(array)
Data_binarized = binarizer.transform(array)

# Showing the first 3 rows in the output
print ("\nBinary data:\n", Data_binarized [0:10])
```

4.1 Code output

```
   Year  Water
0  1885    356
1  1886    386
2  1887    397
3  1888    397
4  1889    413
5  1890    458
6  1891    485
7  1892    344
8  1893    390
```

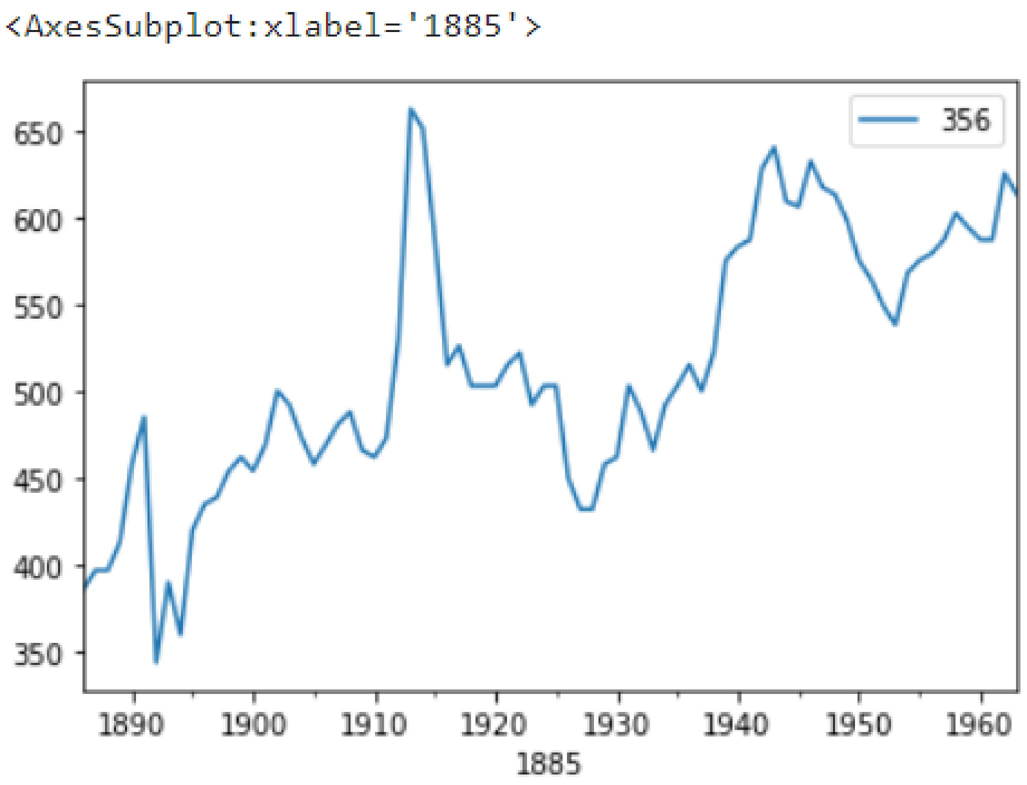
```
9  1894    360

Binary data:
 [[1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]]
```

5 Python Code

```
    from sklearn.preprocessing import Normalizer

# Using Normalizer class with L1 to normalize the data.
Data_normalizer = Normalizer(norm='l1').fit(array)
Data_normalized = Data_normalizer.transform(array)

# Setting the precision to 2
set_printoptions(precision=2)

# Showing the first 10 rows in the output
print ("\nNormalized data:\n", Data_normalized [0:10])

dataframew = read_csv("yearly-water-usage.csv", index_col=0, parse_dates=True)
dataframew.plot()
```

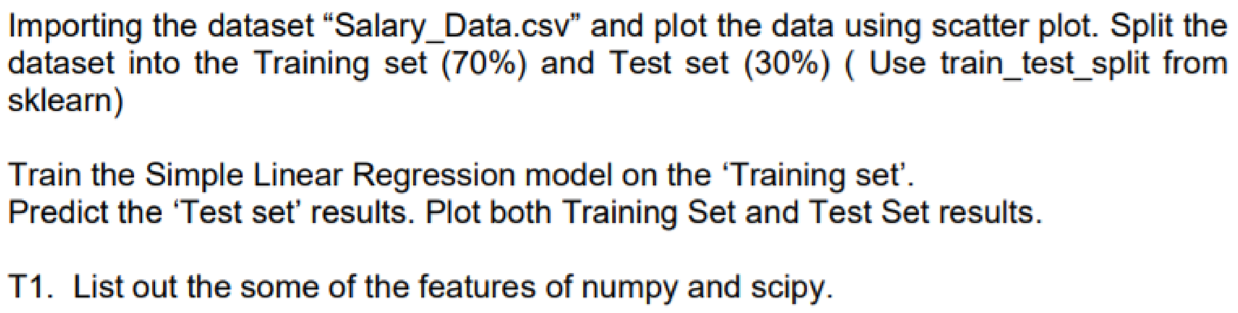
5.1 Code Output

```
Normalized data:
 [[0.84 0.16]
 [0.83 0.17]
 [0.83 0.17]
 [0.83 0.17]
 [0.82 0.18]
 [0.8  0.2 ]
 [0.8  0.2 ]
 [0.85 0.15]
 [0.83 0.17]
 [0.84 0.16]]
```



```
<AxesSubplot:xlabel='1885'>
```

Problem 2



Importing the dataset “Salary_Data.csv” and plot the data using scatter plot. Split the dataset into the Training set (70%) and Test set (30%) (Use train_test_split from sklearn)

Train the Simple Linear Regression model on the ‘Training set’.
Predict the ‘Test set’ results. Plot both Training Set and Test Set results.

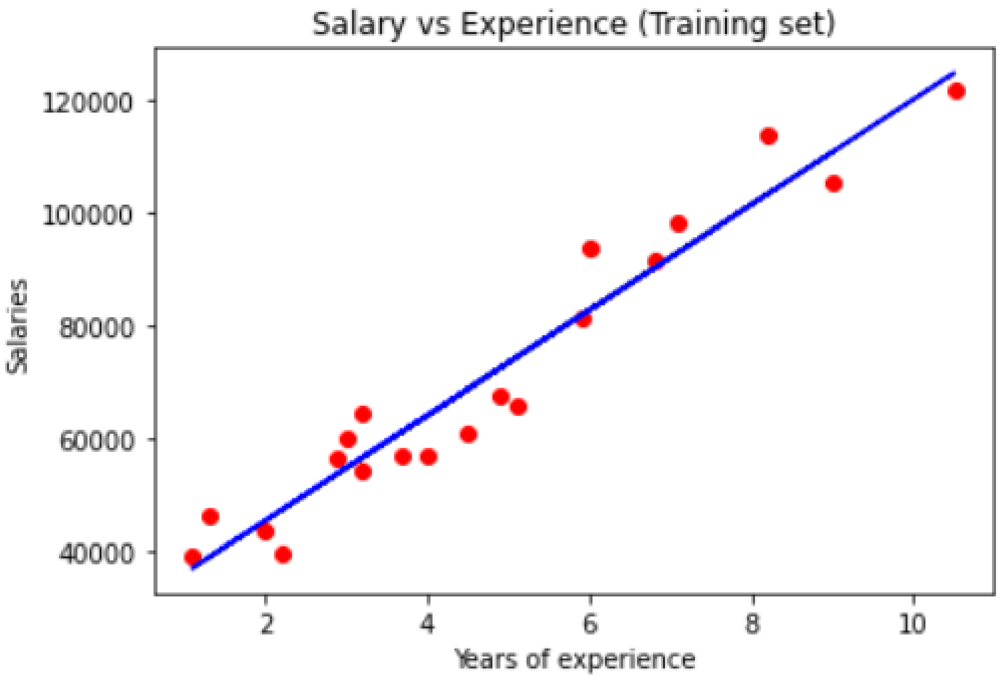
T1. List out the some of the features of numpy and scipy.

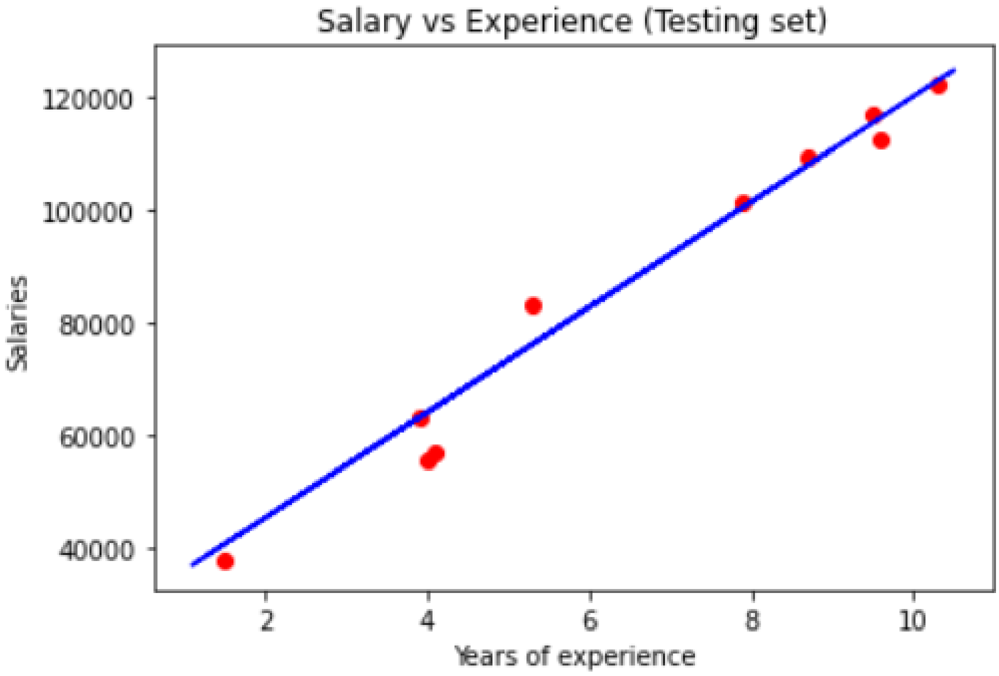
6 Python Code

```
    import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

datasetSalary = pd.read_csv('Salary_Data.csv')
datasetSalary.head()
```

```
# data preprocessing
X = datasetSalary.iloc[:, :-1].values  #independent variable array
y = datasetSalary.iloc[:,1].values  #dependent variable vector

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=1/3,random_state=0)

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train,y_train) #actually produces the linear eqn for the data

#plot for the TRAIN

plt.scatter(X_train, y_train, color='red') # plotting the observation line

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

plt.title("Salary vs Experience (Training set)") # stating the title of the graph

plt.xlabel("Years of experience") # adding the name of x-axis
plt.ylabel("Salaries") # adding the name of y-axis
plt.show() # specifies end of graph

#plot for the TEST

plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue') # plotting the regression line

plt.title("Salary vs Experience (Testing set)")

plt.xlabel("Years of experience")
plt.ylabel("Salaries")
plt.show()
```

6.1 Output





7 Conclusion and Features of Numpy and Scipy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on arrays.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

SciPy is a free and open-source Python library used for scientific computing and technical computing. It is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. As mentioned earlier, SciPy builds on NumPy and therefore if you import SciPy, there is no need to import NumPy.

We were able to perform the assigned tasks and in result we learned to play with numpy and arrays and further plotting on using Matplotlib.