BITS F464 - MACHINE LEARNING COURSE PROJECT

Title: Using Machine Learning Algorithms for the prediction of thermodynamic properties for Industrial Applications

GROUP DETAILS

Group Number: 3

NAME	ID Number
Sai Ashwin Kumar C	2019A4PS0628H
S Danush	2019A1PS1460H
Vishal Kumar NK	2019A4PS0693H
Shivani Thirunagari	2019A4PS0754H
Hari Krishna Dhamodaran	2019A4PS1263H
R Sarvesh	2019A1PS1442H

GitHub Repository: https://github.com/sai-ashwin-2001/ml-proj-bitsf464

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

ज्ञानं परमं बलम्

Predicting saturated vapor pressure of LNG from density and temperature data with a view to improving tank pressure management.

Problem Definition:

Prediction and analysis of thermodynamic properties and its correlations using data-driven machine learning algorithms. Different algorithms are to be employed and results are compared for model selection.

Introduction:

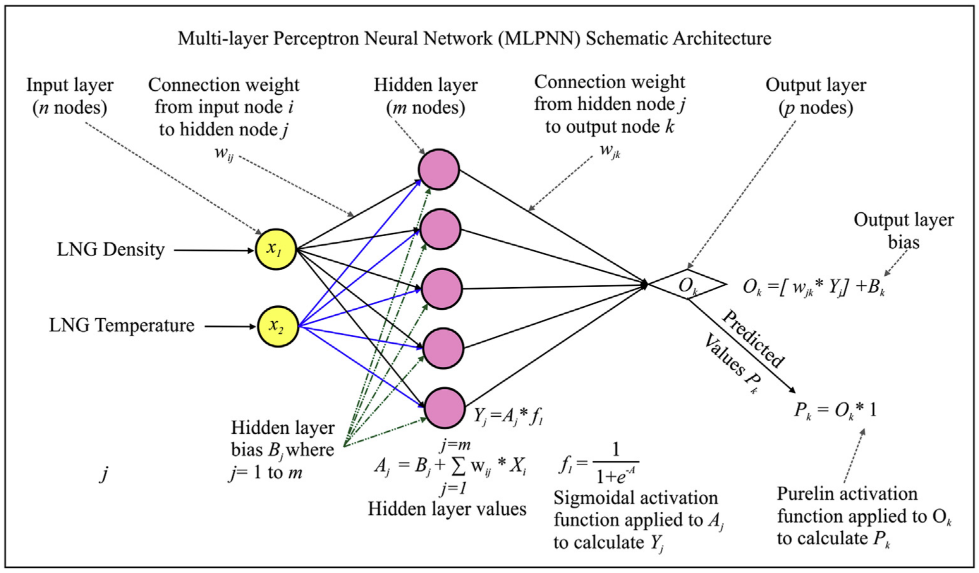
The aim of the study is to predict saturated vapor pressure (SVP) of LNG from various sources (different compositions) using temperature and pressure data for better tank pressure management. This is crucial as it is very difficult to use detailed thermodynamic relationships to determine saturated vapor pressure in dynamic conditions, especially due to various possible compositions. This temperature, pressure and SVP data was generated using Aspen HYSYS, a chemical process simulation software. The temperature range and the different compositions for different sources of LNG are taken in accordance with the reference paper. The framework that we've built predicts SVP in the tank, irrespective of the composition, and thus the application of the framework extends to cases with LNG from any source.

Link to the chosen reference paper - https://reader.elsevier.com/reader/sd/pii/S2405656120300596?token=2230630A7E03FA69B1213FE62304EA855DC85B47BB4681162150A8C71F8FDC53D9119731B30BFB904EE7590CD21D70CF&originRegion=eu-west-1&originCreation=20220422120910

1) ALGORITHMS EXPLAINED IN THE PAPER:

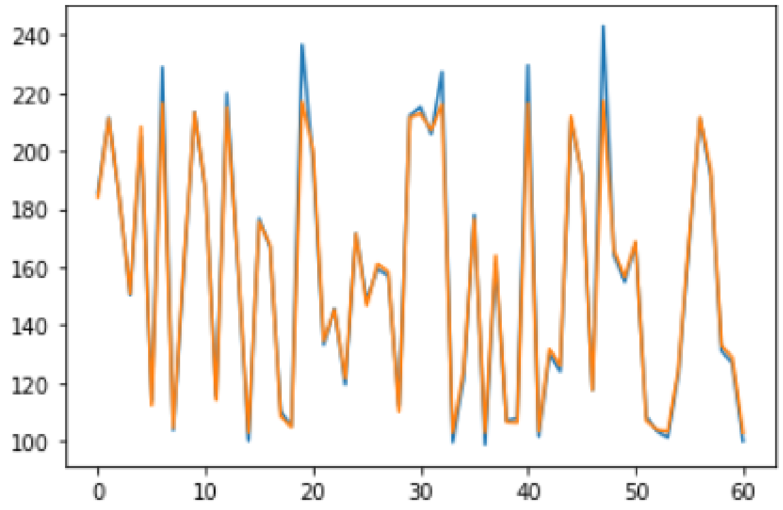
1.1) Multi Layer Perceptron Regression:

The first algorithm that was taken from the paper and implemented was the MLP-ANN algorithm(Multi Layer Perceptron Artificial Neural Network). Tensorflow was used for this purpose. The input layer consisted of LNG density and LNG temperature. For the hidden layer, 5 nodes were considered and a sigmoidal activation function was employed. For the output layer, a linear activation function was used. The model was made to go through as much as 500 iterations to minimise the error of training.


Multi-layer Perceptron Neural Network (MLPNN) Schematic Architecture
Input layer (n nodes)
Connection weight from input node i to hidden node j w_{ij}
Hidden layer (m nodes)
Connection weight from hidden node j to output node k w_{jk}
Output layer (p nodes)
Output layer bias
LNG Density
LNG Temperature
x_1
x_2
O_k
$O_k = [w_{jk} * Y_j] + B_k$
Predicted Values P_k
$P_k = O_k * 1$
$Y_j = A_j * f_l$
Hidden layer bias B_j where j= 1 to m
j
$A_j = B_j + \sum_{j=1}^{j=m} w_{ij} * X_i$
Hidden layer values
$f_l = \frac{1}{1+e^{-A}}$
Sigmoidal activation function applied to A_j to calculate Y_j
Purelin activation function applied to O_k to calculate P_k

The MSE came out to be `19.4131`. A plot of y_test and y_pred was plotted to compare the test results. The weights pertaining to the the different layers of the model were are also obtained.

```
R: 0.9976932533660026. R2: 0.9953918278.
```



Plot1

1.2)Transparent Open Box(TOB) Learning Network:

This algorithm was implemented in various steps:

STAGE-1:

First, all the elements were put into a 2D array involving N input variables plus 1 objective variable to predict for M data records. Then, the data was sorted in the ascending order. Statistical metrics like minimum and maximum were calculated for each of the variables in the dataset and the same was used for normalization was done for the M data records to get all the values in the range (-1,1). Then, the statistical metrics were calculated for the normalized data set. For each of the tuning records, Variable Square Error was calculated. Then, this was summed for each of the data records. Then, the top ten ranked records were selected on the criteria of having the lowest VSE. Here, rank no. 1 contributed the highest whereas rank no. 10 contributed the lowest to the data. Then, RMSE and R squared values were calculated.

STAGE-2:

Stage 2 was meant for tuning and optimization. The objective function was to minimize the RMSE as much as possible. The RMSE and R square values were calculated for different values of Q ranging from 2 to 10 to check for underfitting or overfitting and find an optimal value of Q. The optimal value of weight was also calculated.

Minimum rmse: 0.5432286706668947

Maximum r2: 0.9711484240308468

Optimal value of Q and weight: [9, 0.05263157894736842]

This optimal value of Q and weights was applied to the model by passing the testing data. We obtained the results very much comparable to the obtained tuning error and optimal r2 values.

```
RMSE - 0.9627833931117183, R2 - 0.5373370590392742
```

2) ALTERNATIVE METHODS OR APPROACHES TO SOLVING THE PROBLEM:

2.1) L2 Norm(Ridge Regression):

REGULARIZATION:

Regularization is a technique that helps to deal with the bias-variance problem of the model development. When small changes are made to the data, such as switching from training to testing data there can be huge difference in the estimates. Regularization can help smooth this problem substantially by adding a “lambda” parameter with the usual linear regression model.

Ridge regularization:

In the ridge regularization technique, the cost function gets modified by adding a square of the absolute value of the coefficients whole multiplied by lambda. Different values of lambda give us different models. The cost function **without regularization** is:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2$$

The **cost function with ridge regularization** is as follows:

$$\sum_{i=1}^{M} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{M} \left(y_i - \sum_{j=0}^{p} w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^{p} w_j^2$$

We have implemented the algorithm in a neat and simple way by taking the above equation in a vectorized form as follows:

$$\mathbf{J = ||Y - \Theta X||_2^2 + \lambda||\Theta||_2^2}$$

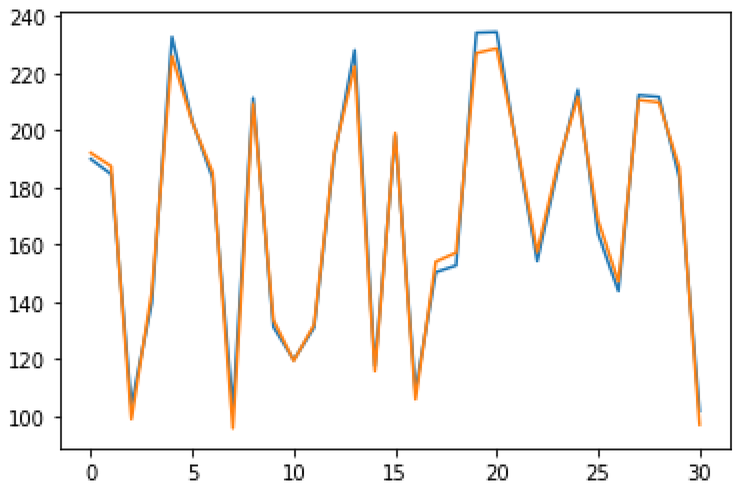
By minimizing the above cost function with respect to weight, we obtain the weights as:

$$\theta = (X^T X + \lambda I)^{-1} X^T Y$$

First, the optimal value for hyperparameter λ was found out using grid search. Least value of MSE obtained was the criteria used for optimal selection of λ, among an array of equally spaced values iterated .

We then train the model using the optimum λ value obtained through grid search and the training data on the above mentioned vectorized weights equation. The predictions of the dependent variable made by the trained model is obtained by a simple dot product between the weights and testing data of the independent variables. Then we compare the accuracy of the model by computing the metrics MSE, MAE and correlation coefficient between the testing data of the dependent variable and the predicted values from the model. The results for the implementation are as follows:

```
Mean Square Error:  13.773397029389496
Mean Absolute Error:  3.1622300463765924
Correlation Coefficient:  0.998097252172384
```



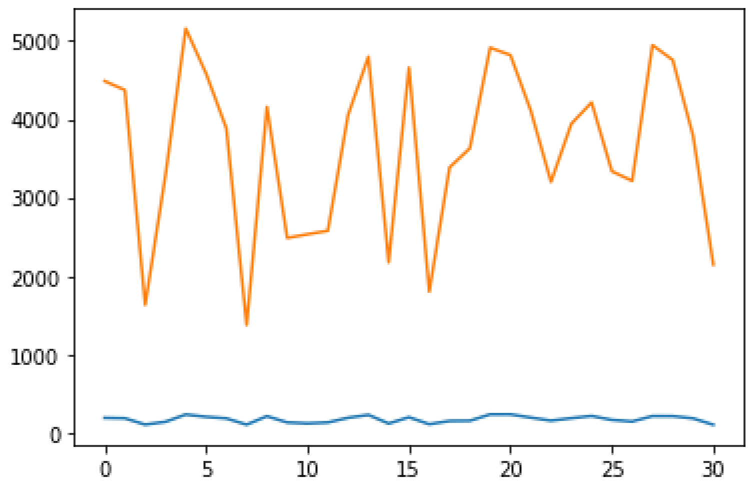
Plot3

While this method gives a very high correlation coefficient, this algorithm cannot be applied to very huge datasets due to increased computational complexity.

<u>2.1.1) BATCH GRADIENT DESCENT WITH L2 NORM:</u>

When the number of training examples is large, they are taken into batches and regression is performed. We have implemented the algorithm for the given data set. Again, an optimal hyperparameter was found out(α= 0.001).

```
Mean Square Error:  12984951.937469797
Mean Absolute Error:  3454.528335015921
Correlation Coefficient:  0.9817153088841384
```



Plot4

2.2) L1 Norm(Lasso/Least Angle Regression):

When the term added to the error function is lambda times the sum of absolute values of the coefficients, the regularization is known as lasso regularization. We use different values of lambda to get different models.

The difference between these two methods is that L1 regularization penalizes the sum of absolute values of the weights, whereas L2 regularization penalizes the sum of squares of the weights, which gives the latter the tendency to make the coefficients absolute zero.

The cost function using lasso regularization is:

$$\sum_{i=1}^{M}(y_i-\hat{y}_i)^2=\sum_{i=1}^{M}\left(y_i-\sum_{j=0}^{p}w_j\times x_{ij}\right)^2+\lambda\sum_{j=0}^{p}|w_j|$$

Again we have used the vectorized form in our models for representing the cost function and evaluating the weights.

$$\mathbf{J = ||Y - \Theta X||_2^2 + \lambda||\Theta||_1}$$

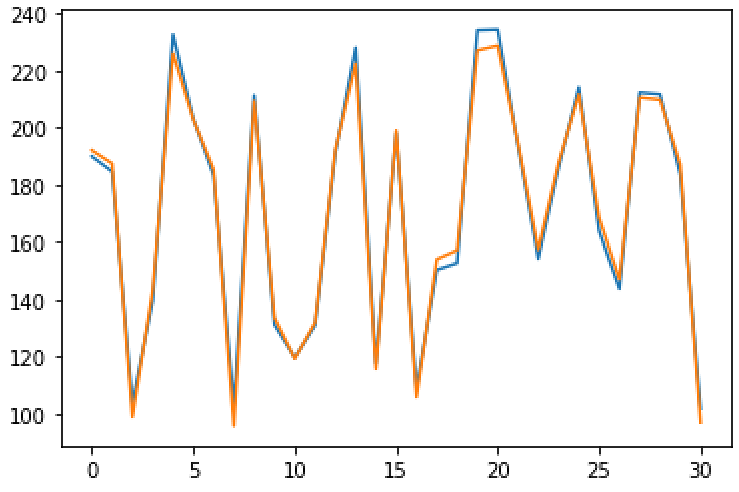
Minimising the cost function w.r.t weight and equating to zero, we obtain the expression for the weight vector as:

$$\mathbf{\Theta = argmin_{MSE}(((X^TX)^{-1}(X^TY - \lambda)),((X^TX)^{-1}(X^TY + \lambda)))}$$

For a particular model, there are two options, either λ to be added or subtracted from the hypothesis (i.e. X^TY term). The weights are computed by choosing the option that returns the least training/validation error.

Again as we did for L2 regularization, we have found the optimal hyperparameter λ for this case too by using grid search. The optimal λ is chosen by iterating through an array of values with the criteria of minimum validation MSE returned by the model.

The optimum λ value and training data are passed into the model to obtain the weights (using the vectorized form), which are then multiplied with the instances of the test vector to provide the model predictions. Then we have compared the predictions and actual values using the metrics MSE, MAE, and CC. The results of the implementation are provided below:

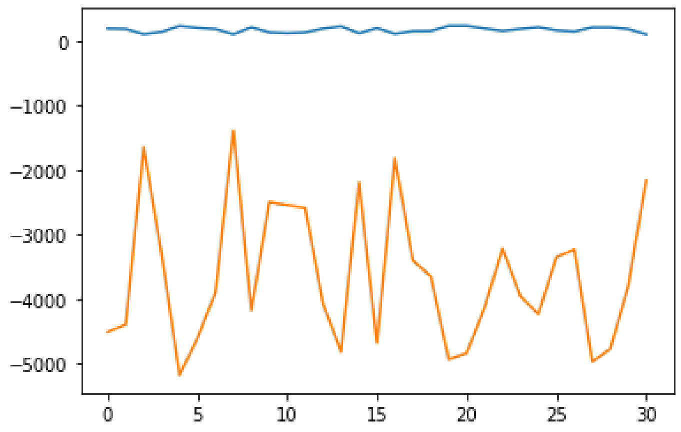


Plot5

```
Mean Square Error:  13.687635091836444
Mean Absolute Error:  3.1559323861202326
Correlation Coefficient:  0.9981066047524438
```

2.2.1) BATCH GRADIENT DESCENT WITH L1 NORM:

Again, Batch Gradient descent was performed with L1 norm this time. Optimal value of hyperparameter came out to be 0.001 and a correlation coefficient of 0.03 was obtained.



Plot6

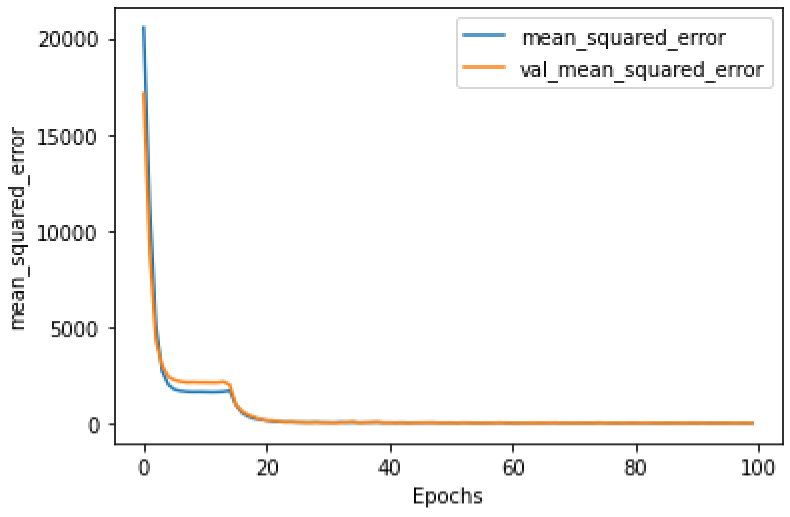
```
Mean Square Error:  15801321.079228736
Mean Absolute Error:  3817.2116426908256
Correlation Coefficient:  0.01839374045136269
```

The error in the Batch Gradient Algorithm is high and this algorithm was not successful in minimising the errors, as BGD is usually used for large training data where it is broken down into batches and modeled, but in our case we have a small training dataset. Hence, the results are not satisfactory.

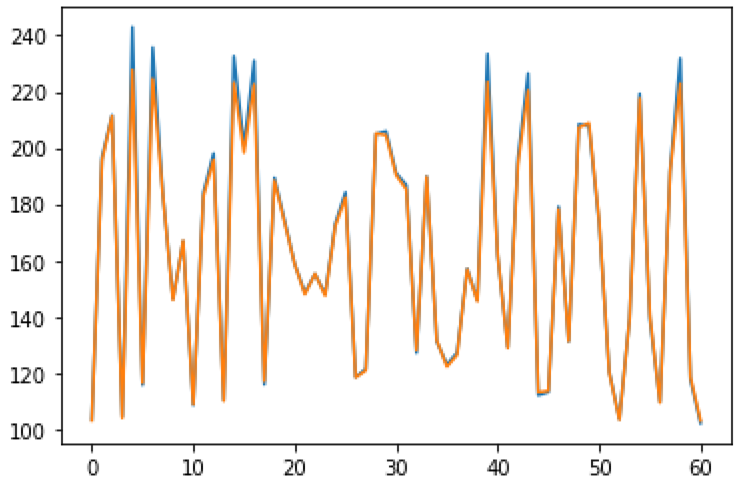
3) DEEP NEURAL NETWORK MODEL:

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. We have employed a DNN to the dataset using tensorflow and keras. First, the train-test split was done on the data and and then applied fit_transform with standard scaler so as to remove the mean and scales each feature/variable to unit variance, before applying the model. For the model, three hidden layers with 200,500,300 nodes respectively were considered along with a learning rate of 0.02. We have also added a 'Dropout layer' with every subsequent hidden layer, which randomly sets input units to 0 with a frequency of rate at each step during training time, considering 20% as the dropout rate. Dropout layers are important in training DNNs because they prevent overfitting on the training data.

The Deep Neural Network model achieves better results than what the single-layer MLP Regressor could do in 500 epochs, within 100 epochs. After running through 100 epochs, a graph of "epochs" vs "Mean Squared Error" was plotted. A very high correlation coefficient of `0.999068` was observed. loss: `1.81686`. Mean squared error: `12.89248`



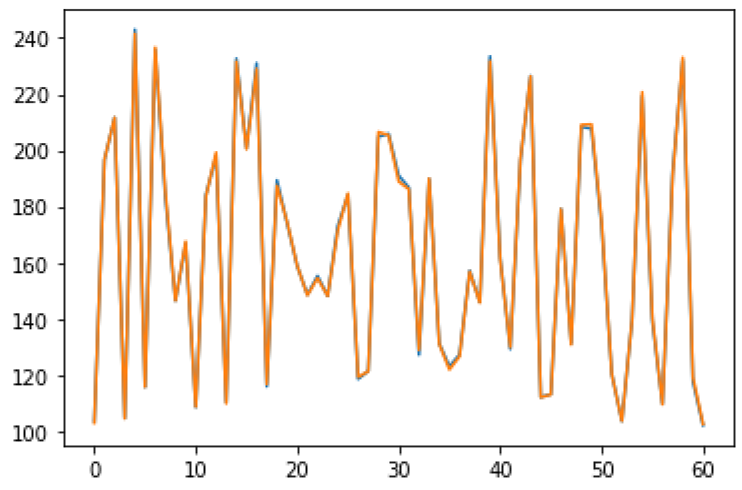
Plot7



A graph of y_test and y_pred was plotted to compare the predictions.

2.4) XG Boost Model:

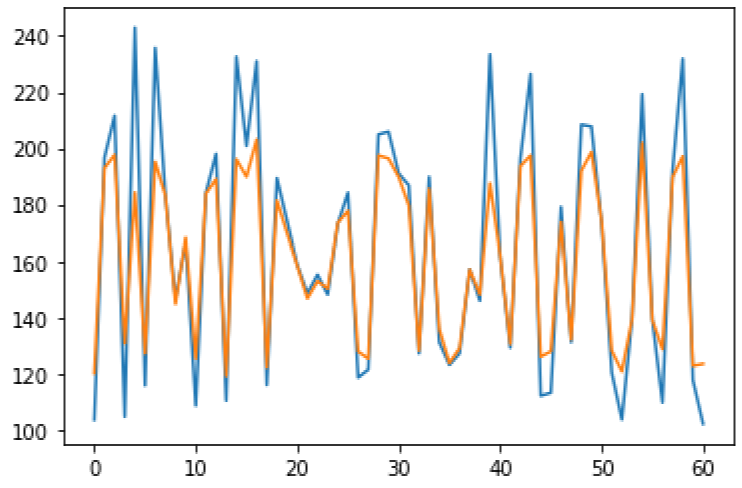
XG Boost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. It is mainly used in prediction type problems, thus we have implemented the same for our model. We fit the model after scaling the training data and calculated rmse which turned out to be 0.9399. An enormous correlation coefficient of 0.99987 was obtained. A graph of y_test and y_pred was plotted to compare the predictions.



Plot8

<u>2.5) SVM Regressor Model:</u>

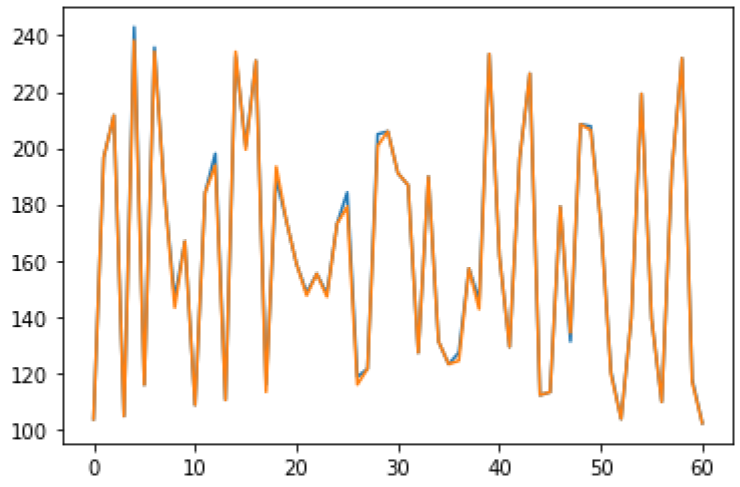
In the SVR or Support Vector Regression Model, all the points that are close to the decision boundary are considered. A line is fit and we obtain the required metrics. We used sklearn to implement this algorithm. Correlation coefficient was calculated and it turned out to be 0.9824 which is pretty high. The value of RMSE came out to be `16.253`. A graph of y_test and y_pred was plotted to compare the predictions.



Plot9

2.6) Random Forest Regression

Random Forest algorithm is a combination of many decision trees. In the case of a regression problem, the final output is the mean of all the outputs from all the decision trees. We have used sklearn to implement this algorithm. Cross- validation is done to obtain optimal number of decision trees or number of estimators and then chosen to be 100 here. The line is fit and the accuracy metrics are calculated.
Correlation coefficient: 0.9998
RMSE: 1.1484



Plot 10

2.7) K Nearest Neighbors Regression

K nearest neighbors is a simple algorithm that stores all available cases and predict the numerical target based on a similarity measure. We have implemented this algorithm using sklearn. After cross validation, the value of K is set as 2. Line was fit, and accuracy metrics calculated.
Correlation coefficient: 0.9994
RMSE: 2.16715

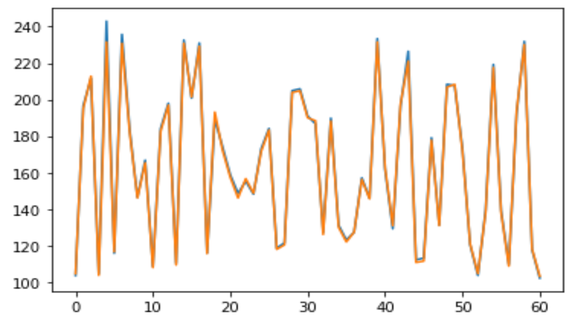


TABLE OF COMPARISON:

S. No.	ALGORITHM	MEASURE USED
1.1	Multi-Layer Perceptron(MLP) Regression	a) RMSE: 4.406027497429569 b) MAE: 2.263472557067871 c) R: 0.9976932533660026 d) R2: 0.9953918278
1.2	Transparent Open Box(TOB) Learning Network	a) RMSE: 0.9627833931117183 b) R2: 0.5373370590392742
2.1	L2 Norm(Ridge Regression)	a) MSE: 13.773397029389496 b) MAE: 3.1622300463765924 c) R: 0.998097252172384
2.1.1	Batch Gradient Descent with L2 Norm	a) MSE: 12984951.937469797 b) MAE: 3454.528335015921 c) R: 0.9817153088841384
2.2	L1 Norm(Lasso Regression)	a) MSE: 13.687635091836444 b) MAE: 3.1559323861202326 c) R: 0.9981066047524438
2.2.1	Batch Gradient Descent with L1 Norm	a) MSE: 15801321.079228736 b) MAE: 3817.2116426908256 c) R: 0.01839374045136269
2.3	Deep Neural Network Model	a) R : 0.9990686343119035 b) MSE: 12.892489433288574 c) MAE: 1.8168684244155884
2.4	XG-Boost Model	a) R : 0.9998760182166879 b) RMSE : 0.9399646417901627
2.5	SVM Regressor Model	a) R : 0.9824549360347229 b) RMSE : 16.252848067389255
2.6	Random Forest Regressor Model	a) R : 0.9998139258537272 b) RMSE : 1.1484409800599138
2.7	K-Nearest Neighbours Regressor Model	a) R : 0.9994981656858772 b) RMSE : 2.167152243435395

Legend:

1) RMSE: Root Mean Squared Error
2) MSE: Mean Squared Error
3) MAE: Mean Absolute Error

4) R: Correlation Coefficient
5) R2: Coefficient of Determination

CONCLUSIONS:

We have successfully implemented the algorithms mentioned in the paper under consideration. For the MLP-ANN, a very high correlation coefficient of `R: 0.9976932533660026` was obtained upon implementation. The value of coefficient of determination, R2 was `R2: 0.9953918278,` the value mentioned in the paper is 0.9899 which is quite close. The RMSE we got was `4.406027497429569` and the original value mentioned in the paper is 4.306. Also, for the TOB Model, the results obtained in the paper are: RMSE: 3.376 kPaA and R2: 0.9939. Upon implementation of the algorithm, the values we got are `R2: 0.9627833931117183, RMSE: 0.5373370590392742,` which are quite comparable.

Apart from this, we implemented other algorithms to see if we could achieve a better result than mentioned in the paper. The highest R value was obtained in the XG-Boost algorithm(`R : 0.9998760182166879),` making it the best among the other algorithms. The other algorithms also fared well, each one of them giving a correlation coefficient of around 0.98. The RMSE for XG-Boost is the least among the other algorithms.

Contributions:

NAME	ID Number	CONTRIBUTION
Sai Ashwin Kumar C	2019A4PS0628H	● L1 Regression ● TOB Learning Network ● Deep Neural Network ● Report - L2 norm, L1 norm regression, DNN
Vishal Kumar NK	2019A4PS0693H	● L2 regression ● TOB Learning Network(few steps) ● Report- L1, L2, MLP, TOB, XG-Boost, SVM, Compiling Results, Conclusions
Hari Krishna D	2019A4PS1263H	● Deep Neural Network

		Model ● XG Boost Model ● SVM Regressor Model
Shivani Thirunagari	2019A4PS0754H	● Multi Layer Perceptron Regression ● Report- L1, L2, MLP, TOB, XG Boost, SVM, Compiling Results, Conclusions
Danush S	2019A1PS1460H	● Problem Definition, Introduction ● Choosing the paper and getting the dataset ● Random Forest Regressor Algorithm- Implementation and Report
Sarvesh R	2019A1PS1442H	● K-Nearest Neighbours Regression - Implementation and Report