

Project Report – Part 1

Machine Learning Implementation on ‘Stock Price Forecasting’

**Submitted to:**

Dr. Madan Gopal Professor Emeritus,

Dept. of Electrical Engineering Shiv Nadar University

Mr. Ashish Kushwaha Assistant Professor,

Dept. of Electrical Engineering Shiv Nadar University

**Submitted by:**

1. Archit Yadav,
2. Anoushkrit Goel,
3. Abhinav Garg

B-Tech III Year ECE Shiv Nadar University

Acknowledgement and Motivation

We would like to express our deepest appreciation to our Professor **Dr. Madan Gopal** who provided us the possibility to complete this report. A special gratitude we give to our Assistant Professor **Ashish Kushwaha**, whose contribution in stimulating suggestions and encouragement helped us to coordinate our project.

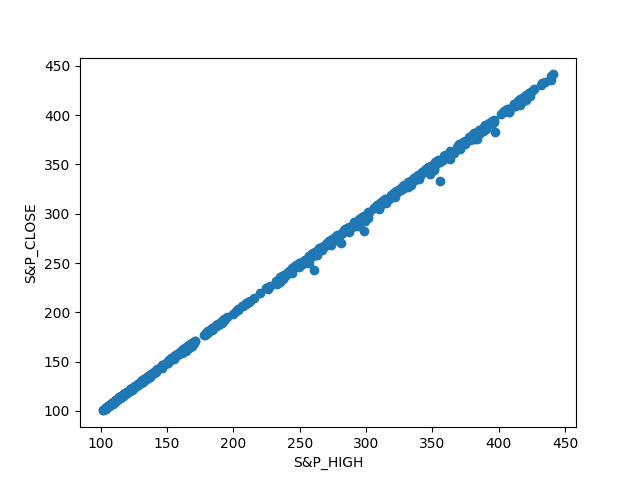
‘Stock Price Forecasting’ is an interesting field of study and a money oriented practical application of Machine Learning. A lot of research work is being carried out to predict the stock prices by financial institutes and is the topic of the hour. This trending topic intrigued us and we found the dataset to be challenging enough to be considered as our project.

Feature Analysis

There are 21 features in our dataset. The stocks market parameters listed in the dataset are listed from 1/4/1980 – 12/4/1992. Feature to be predicted is **S&P Close**. There are total 679 examples given in the dataset (roughly 4 example per month).

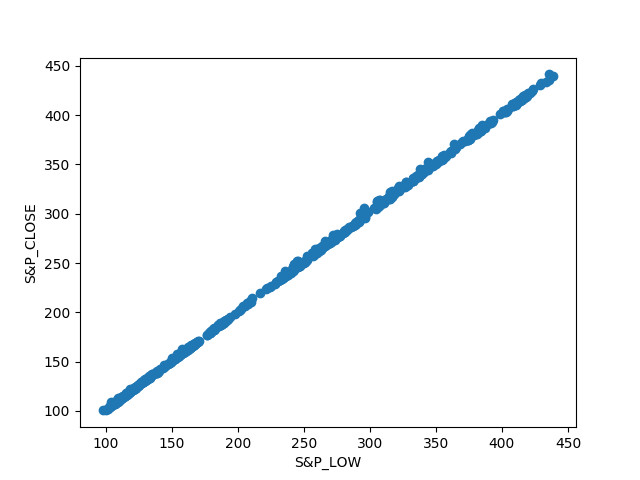
1. **S&P\_HIGH**

It’s a parameter of Standard & Poor listed 500 biggest companies of America. This parameter tells us the highest intra-day value achieved by these companies.



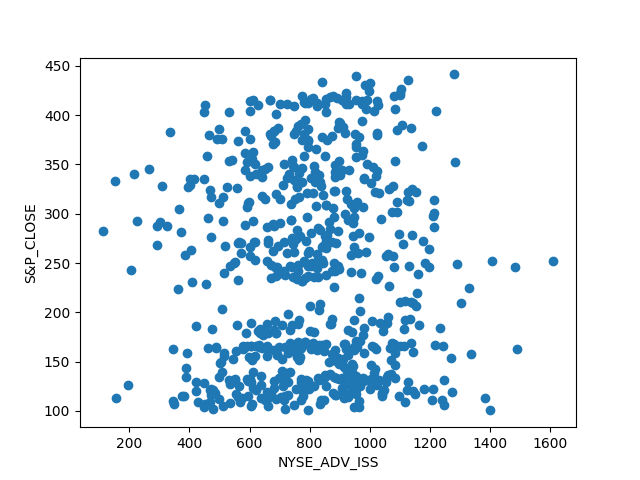
1. **S&P\_LOW**

It’s a parameter of Standard & Poor listed 500 biggest companies of America. This parameter tells us the lowest intra-day value achieved by these companies.



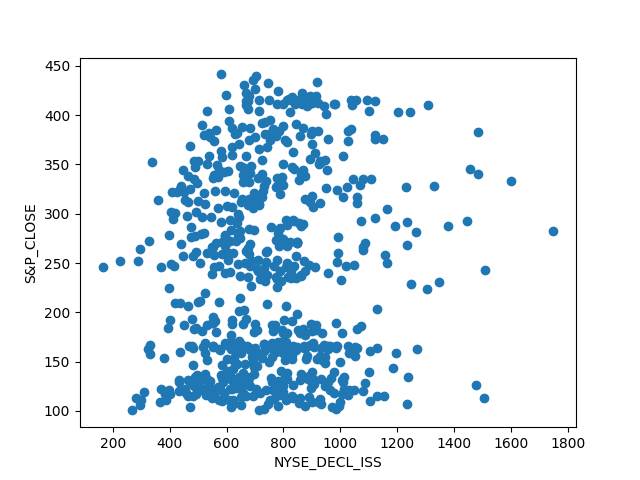
1. **NYSE\_ADV\_ISS**

This parameter indicates how many companies had an increase in their stock prices. This feature will eventually be removed as it has very low correlation value with the output label.



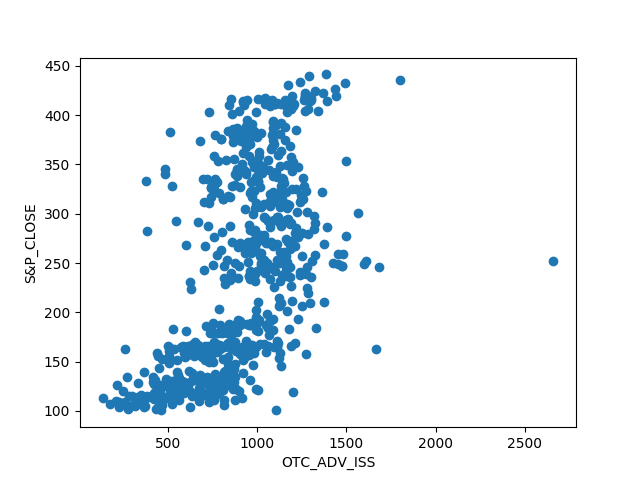
1. **NYSE\_DECL\_ISS**

This parameter indicates how many companies had a decrease in their stock prices. This feature will eventually be removed as it has very low correlation value with the output label.



1. **OTC\_ADV\_ISS**

Over-the-Counter (penny stocks) parameter indicates how many penny stocks had a surge in its value.

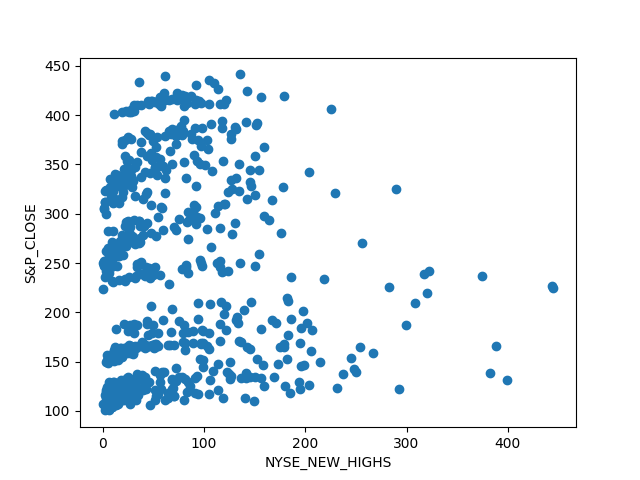


1. **OTC\_DECL\_ISS**

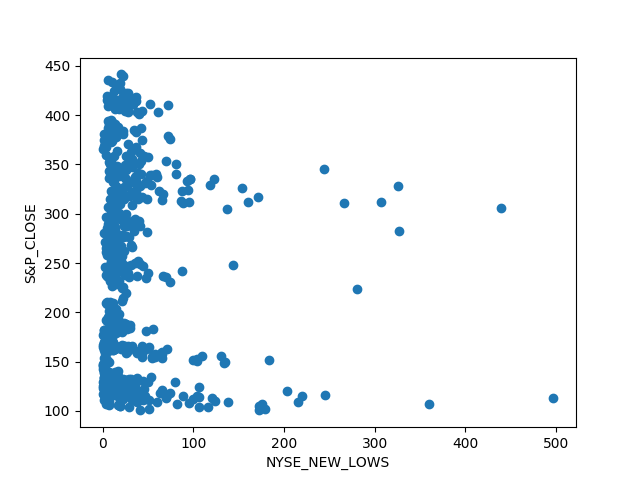
Over-the-Counter (penny stocks) parameter indicates how many penny stocks had a decrease in its value.



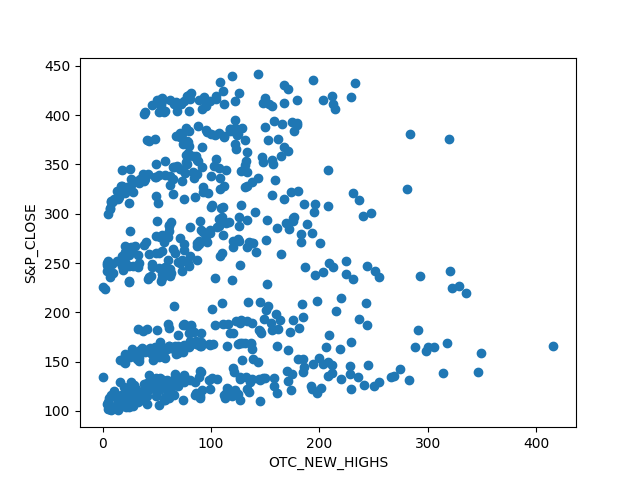
1. **NYSE\_NEW\_HIGHS**

Parameter tells number of best performing shares.

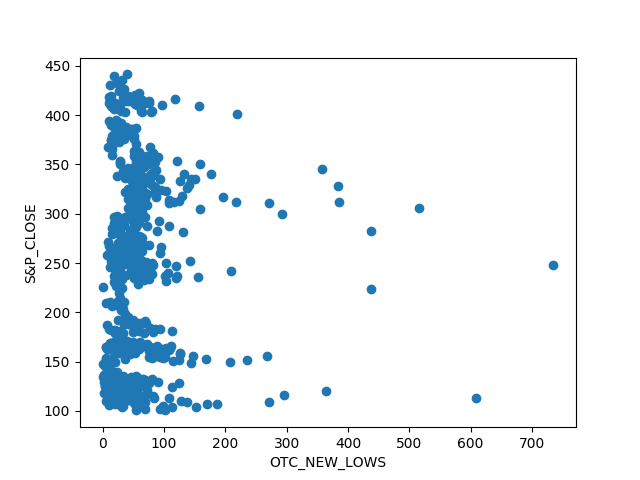
1. **NYSE\_NEW\_LOWS**

Parameter tells number of worst performing shares.

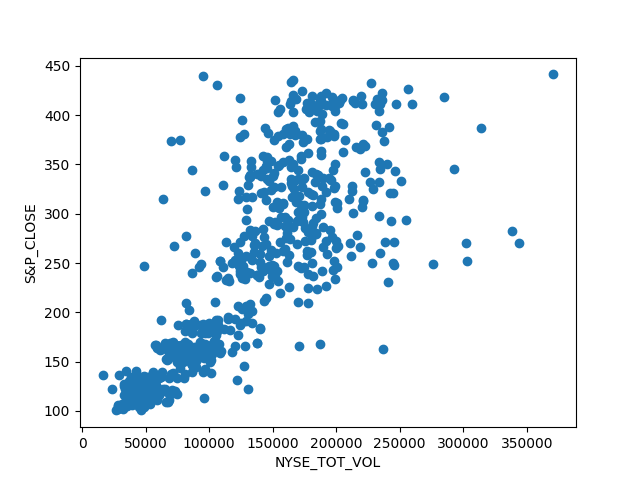
1. **OTC\_NEW\_HIGHS**

Parameter tells number of penny stocks which are best performing.

1. **OTC\_NEW\_LOWS**

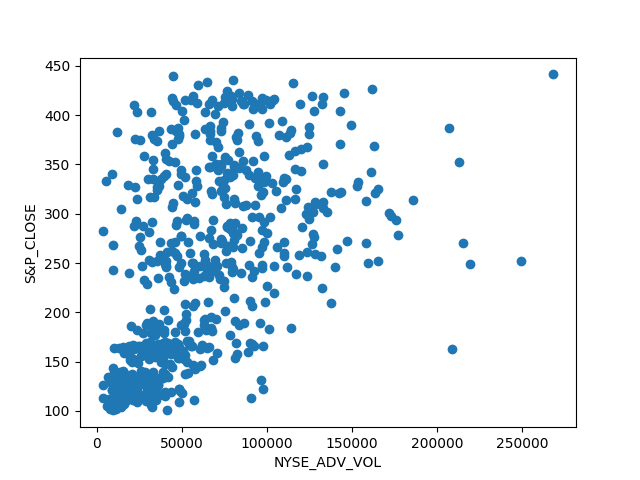
Parameter tells number of penny stocks which are worst performing.

1. **NYSE\_TOT\_VOL**

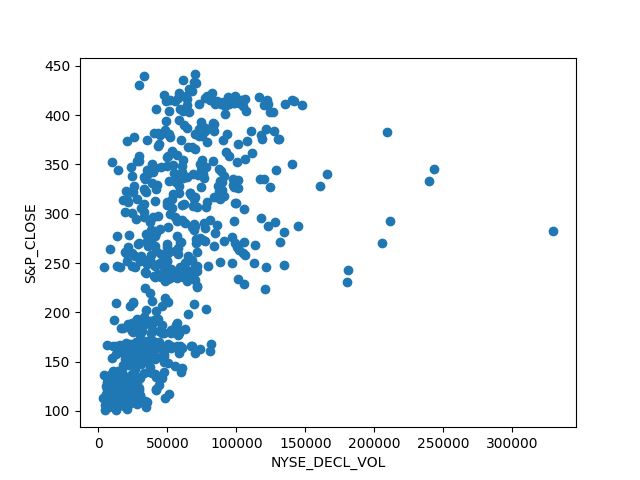
Total number of shares traded in the exchange.

1. **NYSE\_ADV\_VOL**

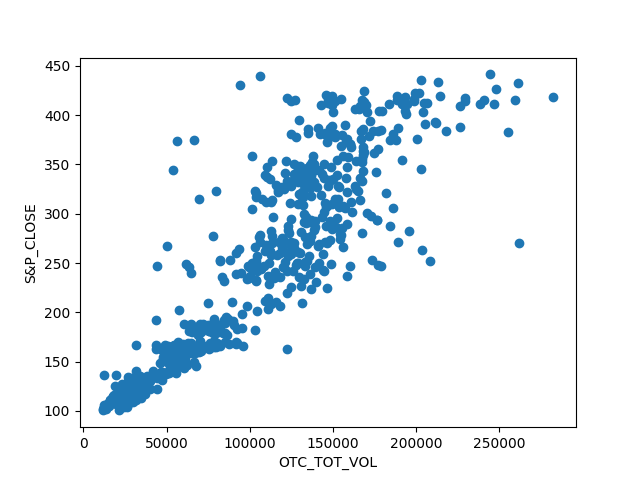
Total number of positive performing shares traded in the exchange.

1. 

**NYSE\_DECL\_VOL**

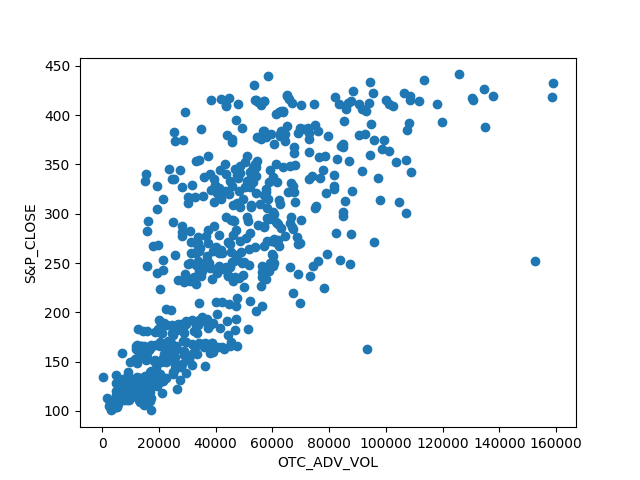
Total number of negatively performing shares traded in the exchange.

1. **OTC\_TOT\_VOL**

Total number of penny stock shares traded in the exchange.

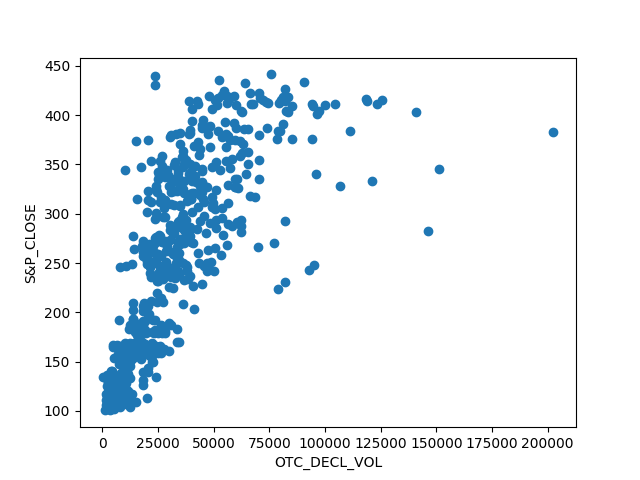
1. **OTC\_ADV\_VOL**

Total number of positive performing penny stock shares traded in the exchange.



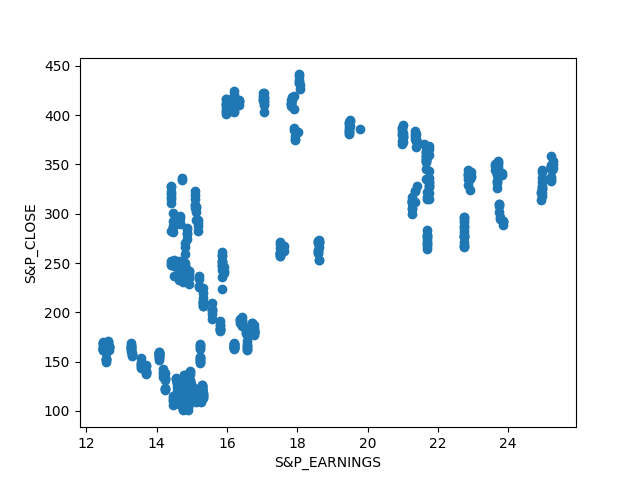
1. **OTC\_DECL\_VOL**

Total number of negatively performing penny stock shares traded in the exchange.

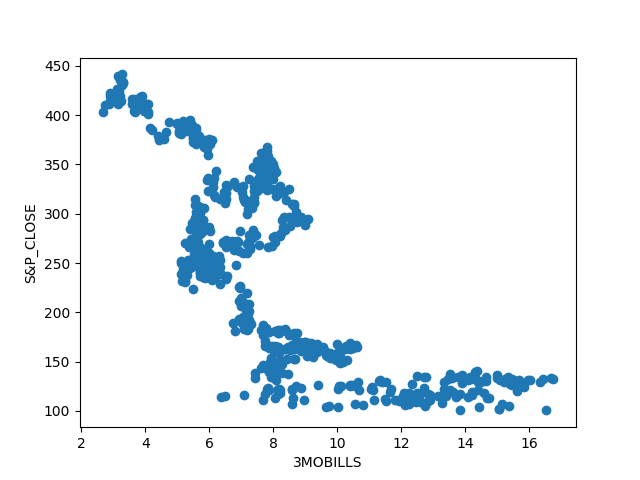


1. **S&P EARNINGS**

This feature hasn’t been understood yet.

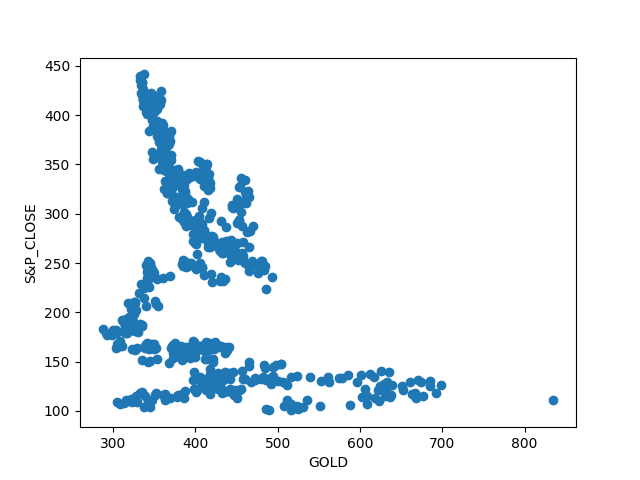


1. **3 MOBILLS**

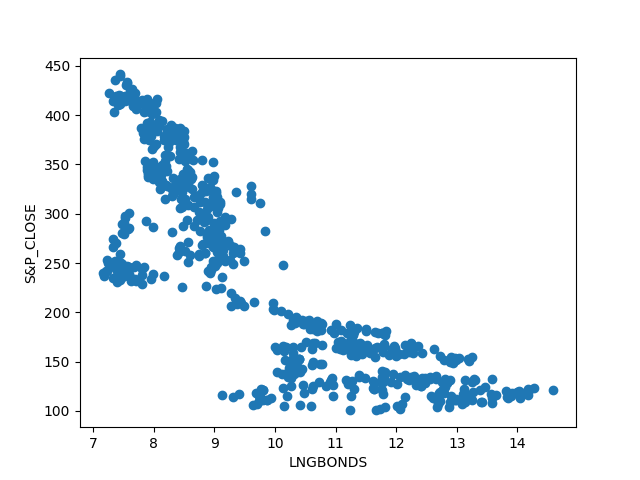
This feature hasn’t been understood yet.

1. **LONGBONDS**

This feature hasn’t been understood yet.

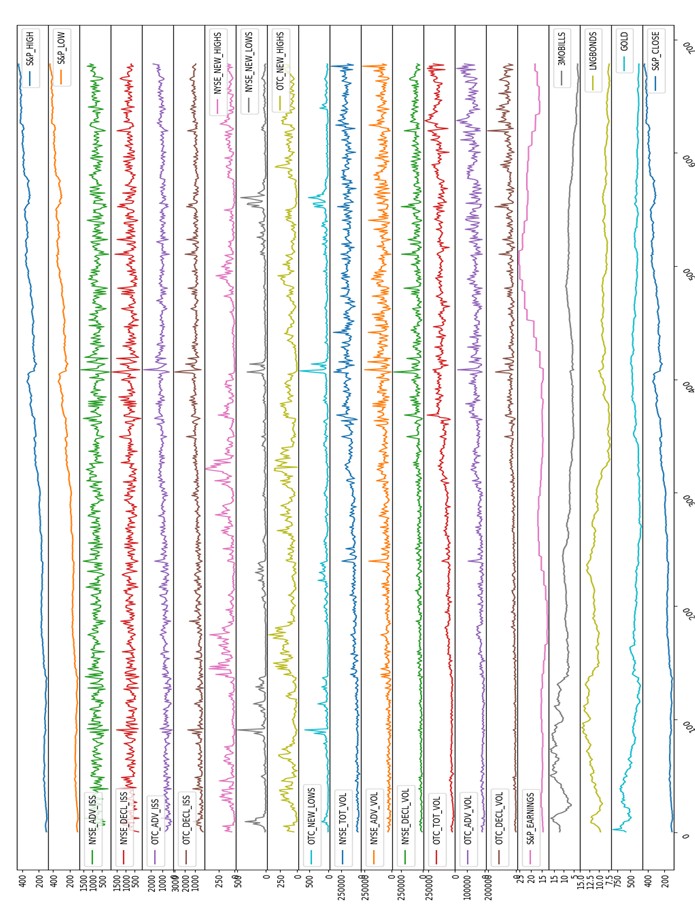


1. **GOLD**

How does S&P close affect the prices of gold

Correlation, Covariance and

Interdependencies



Graph for each feature

The above stated graph shows each and every feature without scaling and reduction. This is rough and unprocessed representation of the dataset provided to us. Through this we can easily visualize which parameters have high and low volatility (idea of standard deviation.

Observation

Observation from correlation matrix shown below was quite interesting as we were able to reduce the features with its help. We also tried to reduce some features by gaining some domain knowledge. The need for applying **Principal Component Analysis (PCA)** was eventually eliminated as the features were reduced through above steps.

Elimination

The features having correlation value with the output label (**S&P\_CLOSE)** less than 0.07 were removed straight away to reduce the features. The contribution of these features seemed not-worth mentioning as they had a very little effect on our variable to be predicted.

Elimination value (Correlation) ≤ 0.07

Hence, eliminating 3 features (**NYSE\_ADV\_ISS, NYSE\_DEC\_ISS, NYSE\_NEW\_LOWS**)

Removing Outliers

After removing the 3 features due to their less correlation with our output variable. The next step was to remove the outliers in the dataset. For this a new matrix was created and Z was calculated for each feature.

Z = (x - mean)/(standard deviation) Z = [(x-µ)/ơ]

Higher the value of Z more that particular point is away from the mean and hence higher the tendency of being an outlier. 5% of the 679 examples showing maximum value of Z were considered as an outlier and replaced with mean value of the feature on the original dataset.

Normalization

In order to have a uniform range of all the features, each point was divided by the maximum value of that feature. Normalization was done so that weights of

all the features remain constant and weights of particular features do not shoot up due to its high values.

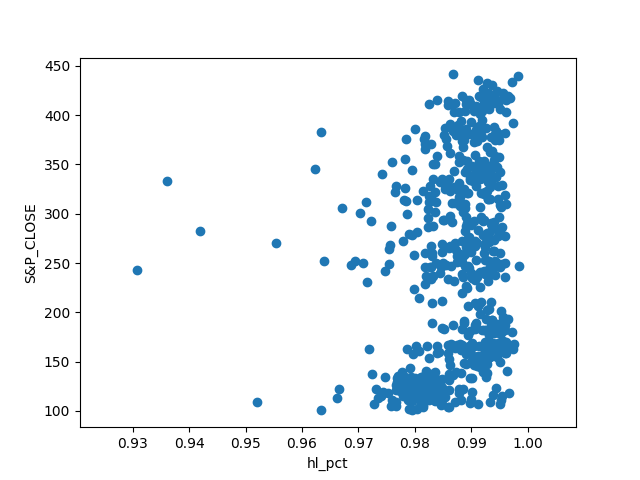
Merging

The features having correlation value with each other more than 0.80 were merged and their ratio was taken to describe the measure.

Merging Value (Correlation) ≥0.80

(S&P\_LOW÷ S&P\_HIGH).

Hence, merging 2 features (**S&P\_HIGH, S&P\_LOW**) by creating one feature describing their ratio (Value remains less than 1 because the ratio taken into consideration is (**S&P\_LOW**÷ **S&P\_HIGH**).



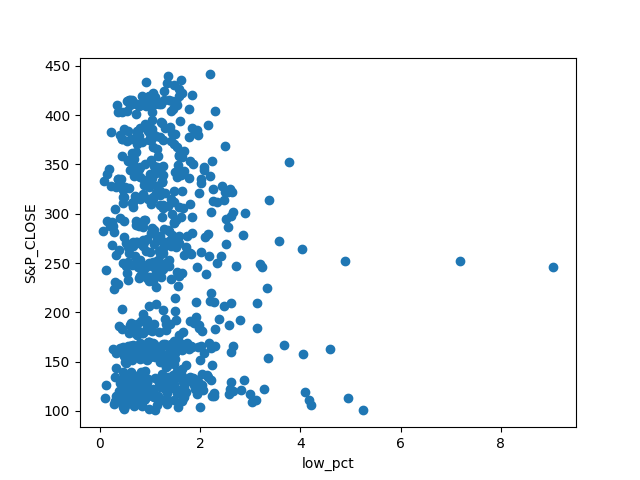
As we can see from the graph, correlation between **hl\_pct** (**S&P\_LOW**÷ **S&P\_HIGH**) is approximately 0.20 which has actually ruined the both the features and incorporating this reduces our accuracy significantly.

(NYSE\_ADV\_ISS / NYSE\_DECL\_ISS)

Now, merging these two features which showed a very low correlation with

**S&P\_CLOSE.**

Here, we tried not removing these features like we stated before whereas we incorporated these features and reduced (merged) them into one. This further gave us a variable **l\_pct (NYSE\_ADV\_ISS/NYSE\_DECL\_ISS)** which was even badly correlated. Hence, removing them seem to be the best option.



Even after applying the domain knowledge for the dataset, **PCA** may find to be useful for finding relation between two features which may not seem to be correlated to be each other with the help of domain knowledge.

//This remains a topic of research for the Project Report Part 2.

Algorithms (Applied)

Linear Regression Application

Before reducing the matrix

In the MATLAB script *“MultiVar\_LinearRegression.m”,* linear regression with multiple variables has been implemented for forecasting stock prices. Training been done on 80% of data while testing of the data is done on the rest 20%.

The accuracy of the results tends to be 82.9055%. In this script, all the features have been used and algorithm is applied on modified Z-score matrix.

**Learning rate (α) was chosen 0.01**. Earlier, the iterations were 2000 due to which cost function converged to 19.04. After increasing the iterations to 19000, the cost function reduced and converged to 0, which was the best case scenario considering other values of α.

**The value of cost function (J) started from 2.3\*10^4 all the way finally to 0.1959**

* 1. core Matrix

The same algorithm was applied in the script *“FeatureReduction\_DropCol.m”,* but this time 3 features were removed due to very less correlation value with label.

Features removed were **“NYSE\_ADV\_ISS”, “NYSE\_DECL\_ISS”** and **“NYSE\_NEW\_LOWS”.** Learning rate and iterations were kept constant and it was found that accuracy improved slightly (from 82.9055 to 82.9282%).

We further modified our dataset and this time we calculated and removed 5% outliers from each feature using the Z-Score matrix (Z score matrix is derived from input matrix using formula (x-µ)/ơ).

Normalisation of the data

Also we normalised data by dividing each and every value by the maximum value of the feature(column). We couldn’t calculate the accuracy of the same due some errors in the code.

// Errors in the code in the Normalisation of the data

S&P\_HIGH S&P\_LOW NYSE\_ADV NYSE\_DECL\_ISS OTC\_ADV\_ISS \

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S&P\_HIGH | 1 | 0.98921 | 0.027696 | 0.070367 | 0.441661 |
| S&P\_LOW | 0.98921 | 1 | 0.029383 | 0.068093 | 0.443318 |
| NYSE\_ADV\_ISS | 0.027696 | 0.029383 | 1 | -0.747096 | 0.418409 |
| NYSE\_DECL\_ISS | 0.070367 | 0.068093 | -0.7471 | 1 | -0.296923 |
| OTC\_ADV\_ISS | 0.441661 | 0.443318 | 0.418409 | -0.296923 | 1 |
| OTC\_DECL\_ISS | 0.512928 | 0.510169 | -0.31316 | 0.414572 | 0.13941 |
| NYSE\_NEW\_HIGHS | 0.114472 | 0.115436 | 0.264967 | -0.144023 | 0.243061 |
| NYSE\_NEW\_LOWS | 0.039511 | 0.037496 | -0.17377 | 0.133222 | -0.11693 |
| OTC\_NEW\_HIGHS | 0.117964 | 0.119606 | 0.192845 | -0.101059 | 0.2423 |
| OTC\_NEW\_LOWS | 0.083889 | 0.08203 | -0.20175 | 0.162851 | -0.075596 |
| NYSE\_TOT\_VOL | 0.654376 | 0.649211 | 0.082871 | 0.03254 | 0.491941 |
| NYSE\_ADV\_VOL | 0.448152 | 0.448401 | 0.443408 | -0.340354 | 0.659569 |
| NYSE\_DECL\_VOL | 0.509694 | 0.506232 | -0.3162 | 0.449111 | 0.164604 |
| OTC\_TOT\_VOL | 0.763859 | 0.76039 | 0.050106 | 0.068306 | 0.503244 |
| OTC\_ADV\_VOL | 0.658665 | 0.659158 | 0.241018 | -0.12313 | 0.672055 |
| OTC\_DECL\_VOL | 0.701928 | 0.698137 | -0.14879 | 0.270539 | 0.285644 |
| S&P\_EARNINGS | 0.398762 | 0.399117 | 0.003419 | -0.046785 | 0.204454 |
| 3MOBILLS | -0.595194 | -0.59508 | -0.04873 | -0.085397 | -0.440344 |
| LNGBONDS | -0.637751 | -0.63839 | -0.03346 | -0.071388 | -0.415445 |
| GOLD | -0.270311 | -0.27194 | -0.04457 | -0.02878 | -0.147413 |
| S&P\_CLOSE | 0.990155 | 0.991352 | 0.034361 | 0.063351 | 0.446691 |
| OTC\_DECL\_ISS NYSE\_NEWNYSE\_NEWOTC\_NEW\_HIGHS \ | | | | | |
| S&P\_HIGH | 0.512928 | 0.114472 | 0.039511 | 0.117964 | |
| S&P\_LOW | 0.510169 | 0.115436 | 0.037496 | 0.119606 | |
| NYSE\_ADV\_ISS | -0.313159 | 0.264967 | -0.17377 | 0.192845 | |
| NYSE\_DECL\_ISS | 0.414572 | -0.14402 | 0.133222 | -0.101059 | |
| OTC\_ADV\_ISS | 0.13941 | 0.243061 | -0.11693 | 0.2423 | |
| OTC\_DECL\_ISS | 1 | -0.08346 | 0.179113 | -0.053603 | |
| NYSE\_NEW\_HIGHS | -0.083459 | 1 | -0.40892 | 0.692775 | |
| NYSE\_NEW\_LOWS | 0.179113 | -0.40892 | 1 | -0.443418 | |
| OTC\_NEW\_HIGHS | -0.053603 | 0.692775 | -0.44342 | 1 | |
| OTC\_NEW\_LOWS | 0.267055 | -0.36164 | 0.558165 | -0.36112 | |
| NYSE\_TOT\_VOL | 0.484212 | 0.131598 | 0.022368 | 0.13369 | |
| NYSE\_ADV\_VOL | 0.157004 | 0.239072 | -0.07514 | 0.207599 | |
| NYSE\_DECL\_VOL | 0.65035 | 0.023611 | 0.068259 | 0.046604 | |
| OTC\_TOT\_VOL | 0.510016 | 0.149377 | 0.004689 | 0.175437 | |
| OTC\_ADV\_VOL | 0.301085 | 0.233477 | -0.07973 | 0.249181 | |
| OTC\_DECL\_VOL | 0.697915 | 0.05152 | 0.087368 | 0.069535 | |
| S&P\_EARNINGS | 0.238436 | -0.02502 | 0.074871 | -0.005268 | |
| 3MOBILLS | -0.472261 | -0.11598 | 0.069657 | -0.116251 | |
| LNGBONDS | -0.461451 | -0.16277 | 0.047874 | -0.138487 | |
| GOLD | -0.132365 | -0.12919 | -0.06624 | -0.083731 | |
| S&P\_CLOSE | 0.50671 | 0.116314 | 0.03758 | 0.11976 | |
| OTC\_NEW\_LOW NYSE\_ADV NYSE\_DECL\_VOL \ | | | | | |
| S&P\_HIGH | 0.083889 | 0.448152 | 0.509694 | | |
| S&P\_LOW | 0.08203 | 0.448401 | 0.506232 | | |
| NYSE\_ADV\_ISS | -0.201747 | 0.443408 | -0.3162 | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NYSE\_DECL\_ISS | 0.162851 | -0.34035 | 0.449111 |  |
| OTC\_ADV\_ISS | -0.075596 | 0.659569 | 0.164604 |  |
| OTC\_DECL\_ISS | 0.267055 | 0.157004 | 0.65035 |  |
| NYSE\_NEW\_HIGHS | -0.361639 | 0.239072 | 0.023611 |  |
| NYSE\_NEW\_LOWS | 0.558165 | -0.07514 | 0.068259 |  |
| OTC\_NEW\_HIGHS | -0.36112 | 0.207599 | 0.046604 |  |
| OTC\_NEW\_LOWS | 1 | -0.01961 | 0.152415 |  |
| NYSE\_TOT\_VOL | 0.08286 | 0.598977 | 0.549733 |  |
| NYSE\_ADV\_VOL | -0.019612 | 1 | 0.155866 |  |
| NYSE\_DECL\_VOL | 0.152415 | 0.155866 | 1 |  |
| OTC\_TOT\_VOL | 0.062016 | 0.518812 | 0.545424 |  |
| OTC\_ADV\_VOL | -0.040272 | 0.646756 | 0.351376 |  |
| OTC\_DECL\_VOL | 0.156646 | 0.322257 | 0.701488 |  |
| S&P\_EARNINGS | 0.084673 | 0.269203 | 0.252182 |  |
| 3MOBILLS | -0.004215 | -0.39358 | -0.45094 |  |
| LNGBONDS | -0.006069 | -0.38439 | -0.4248 |  |
| GOLD | -0.133859 | -0.1548 | -0.17104 |  |
| S&P\_CLOSE | 0.081887 | 0.454136 | 0.501967 |  |
| OTC\_TOT\_VOL | | OTC\_ADV\_ | OTC\_DECL | S&P\_EARNINGS \ |
| S&P\_HIGH | 0.763859 | 0.658665 | 0.701928 | 0.398762 |
| S&P\_LOW | 0.76039 | 0.659158 | 0.698137 | 0.399117 |
| NYSE\_ADV\_ISS | 0.050106 | 0.241018 | -0.14879 | 0.003419 |
| NYSE\_DECL\_ISS | 0.068306 | -0.12313 | 0.270539 | -0.046785 |
| OTC\_ADV\_ISS | 0.503244 | 0.672055 | 0.285644 | 0.204454 |
| OTC\_DECL\_ISS | 0.510016 | 0.301085 | 0.697915 | 0.238436 |
| NYSE\_NEW\_HIGHS | 0.149377 | 0.233477 | 0.05152 | -0.025016 |
| NYSE\_NEW\_LOWS | 0.004689 | -0.07973 | 0.087368 | 0.074871 |
| OTC\_NEW\_HIGHS | 0.175437 | 0.249181 | 0.069535 | -0.005268 |
| OTC\_NEW\_LOWS | 0.062016 | -0.04027 | 0.156646 | 0.084673 |
| NYSE\_TOT\_VOL | 0.77334 | 0.667572 | 0.661055 | 0.344188 |
| NYSE\_ADV\_VOL | 0.518812 | 0.646756 | 0.322257 | 0.269203 |
| NYSE\_DECL\_VOL | 0.545424 | 0.351376 | 0.701488 | 0.252182 |
| OTC\_TOT\_VOL | 1 | 0.744618 | 0.724416 | 0.34066 |
| OTC\_ADV\_VOL | 0.744618 | 1 | 0.503022 | 0.322319 |
| OTC\_DECL\_VOL | 0.724416 | 0.503022 | 1 | 0.333766 |
| S&P\_EARNINGS | 0.34066 | 0.322319 | 0.333766 | 1 |
| 3MOBILLS | -0.592407 | -0.53356 | -0.5624 | -0.185076 |
| LNGBONDS | -0.577489 | -0.54897 | -0.56023 | -0.26942 |
| GOLD | -0.201811 | -0.19117 | -0.21767 | -0.177139 |
| S&P\_CLOSE | 0.762227 | 0.662706 | 0.695342 | 0.398632 |
| 3MOBILLS | | LNGBONDS | GOLD | S&P\_CLOSE |
| S&P\_HIGH | -0.595194 | -0.63775 | -0.27031 | 0.990155 |
| S&P\_LOW | -0.595078 | -0.63839 | -0.27194 | 0.991352 |
| NYSE\_ADV\_ISS | -0.048728 | -0.03346 | -0.04457 | 0.034361 |
| NYSE\_DECL\_ISS | -0.085397 | -0.07139 | -0.02878 | 0.063351 |
| OTC\_ADV\_ISS | -0.440344 | -0.41545 | -0.14741 | 0.446691 |
| OTC\_DECL\_ISS | -0.472261 | -0.46145 | -0.13237 | 0.50671 |
| NYSE\_NEW\_HIGHS | -0.115983 | -0.16277 | -0.12919 | 0.116314 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NYSE\_NEW\_LOWS | 0.069657 | 0.047874 | -0.06624 | 0.03758 |
| OTC\_NEW\_HIGHS | -0.116251 | -0.13849 | -0.08373 | 0.11976 |
| OTC\_NEW\_LOWS | -0.004215 | -0.00607 | -0.13386 | 0.081887 |
| NYSE\_TOT\_VOL | -0.554455 | -0.52391 | -0.17266 | 0.652264 |
| NYSE\_ADV\_VOL | -0.393577 | -0.38439 | -0.1548 | 0.454136 |
| NYSE\_DECL\_VOL | -0.450936 | -0.4248 | -0.17104 | 0.501967 |
| OTC\_TOT\_VOL | -0.592407 | -0.57749 | -0.20181 | 0.762227 |
| OTC\_ADV\_VOL | -0.533563 | -0.54897 | -0.19117 | 0.662706 |
| OTC\_DECL\_VOL | -0.562402 | -0.56023 | -0.21767 | 0.695342 |
| S&P\_EARNINGS | -0.185076 | -0.26942 | -0.17714 | 0.398632 |
| 3MOBILLS | 1 | 0.669188 | 0.189963 | -0.594758 |
| LNGBONDS | 0.669188 | 1 | 0.146586 | -0.637985 |
| GOLD | 0.189963 | 0.146586 | 1 | -0.27114 |
| S&P\_CLOSE | -0.594758 | -0.63799 | -0.27114 | 1 |

**Multivariate Linear Regression**

**Objective**

In this MATLAB script, linear regression with multiple variabes has been coded and tried to be imple- mented for forecasting stock prices. Training has been done on 80% of data and prediction has been tried to be done on the rest 20% data. The accuracy of the results comes to be about 82.9055%. In this script, all features have been used and Z-score has been used on the input matrix.

clc clear close all

fileToRead = 'S&Pdata';

%Training would be done on 80% of data (1:550 out of 679) rangeTaken = 1:550;

% Import the complete spreadsheet file [xlsObjectComplete, xlsHeads] = xlsread(fileToRead);

% xlsHeads contains the headings in the form of a string xlsHeads = xlsHeads(2:22); % Remove the 'DATE' heading

% Filter just the S&P Close into a vector (Take the last col only) SP\_Close = xlsObjectComplete(rangeTaken, 22);

inputMatrix = xlsObjectComplete(rangeTaken, 2:21); thetaWeights = zeros(21, 1);

% Learning rate, or rate of descent alpha = 0.01;

iterations = 19000; % Obtained after a few hit and trials.

%--------------------------------------------------------------------------

* + - Normalization

%--------------------------------------------------------------------------

X = inputMatrix;

X\_norm = X;

mu = zeros(1, size(X, 2)); sigma = zeros(1, size(X, 2));

for feature\_index = 1:size(X,2)

% Find mean

feature\_mean = mean(X(:,feature\_index));

% (datatpoint - mean)

X\_norm(:,feature\_index) = X(:,feature\_index) - feature\_mean;

1

Multivariate Linear Regression

% Find StdDev

feature\_std = std(X\_norm(:,feature\_index));

% (datatpoint - mean)/(stdDev)

X\_norm(:,feature\_index) = X\_norm(:,feature\_index) / feature\_std;

end

sigma(feature\_index) = feature\_std; mu(feature\_index) = feature\_mean;

% Repeat the above code for getting complete normalized input matrix

X = xlsObjectComplete(:, 2:21); X\_norm\_Complete = X;

mu = zeros(1, size(X, 2)); sigma = zeros(1, size(X, 2));

for feature\_index = 1:size(X,2)

% Find mean

feature\_mean = mean(X(:,feature\_index));

% (datatpoint - mean)

X\_norm\_Complete(:,feature\_index) = X(:,feature\_index) - feature\_mean;

% Find StdDev

feature\_std = std(X\_norm\_Complete(:,feature\_index));

% (datatpoint - mean)/(stdDev) X\_norm\_Complete(:,feature\_index) =

X\_norm\_Complete(:,feature\_index) / feature\_std;

end

sigma(feature\_index) = feature\_std; mu(feature\_index) = feature\_mean;

X\_norm\_Complete = [ones(679, 1), X\_norm\_Complete];

% X\_norm\_Complete stores ALL the input variables of ALL features in a

% normalized manner, and a one vector is appended in the beginning. This

% will be exported to Excel/CSV later.

%--------------------------------------------------------------------------

* + - Gradient descent algo

%--------------------------------------------------------------------------

m = length(SP\_Close);

X\_norm = [ones(m, 1) X\_norm]; % The normalized X X = [ones(m, 1) inputMatrix];

y = SP\_Close;

2

Multivariate Linear Regression

for i = 1 : iterations

temp = thetaWeights - (alpha/m) \* X\_norm' \* (X\_norm \* thetaWeights

- y);

thetaWeights = temp;

jHistory(i) = (1 / (2\*m) ) \* sum(((X\_norm \* thetaWeights)-y).^2);

end

%--------------------------------------------------------------------------

* + - Cost function algo

%-------------------------------------------------------------------------- J = 0;

fprintf('Cost function:\n');

J = (1 / (2\*m) ) \* sum(((X\_norm \* thetaWeights)-y).^2)

%--------------------------------------------------------------------------

*Cost function:*

*J =*

*0.1959*

* + - Output variable (predicted)

%--------------------------------------------------------------------------

% Take the 20% (551:679) of all the output variables for testing SP\_Close\_ToBePredicted = xlsObjectComplete(551:679, 22);

% Preallocate yHat with zeros yHat = zeros(129, 1);

% Caculate the predicted output vaiable

% y = theta(0)\*x(0) + theta(1)\*x(1) + ... + theta(21)\*x(21);

% 21, because we've 21 features.

for i = 1 : 21

someTempVar = (thetaWeights(i) \* X\_norm\_Complete(551:679, i)); yHat = yHat + someTempVar;

end

difference = SP\_Close\_ToBePredicted - yHat;

accuracy = (difference./SP\_Close\_ToBePredicted)\*100; mean = mean(accuracy);

% Print the accuracy calculated actualAccuracy = 100-mean

% Export the normalized input matrix file to Excel/CSV

% xlswrite('matToExcel.xlsx', X\_norm\_Complete);

%--------------------------------------------------------------------------

3

Multivariate Linear Regression

*actualAccuracy = 82.9055*

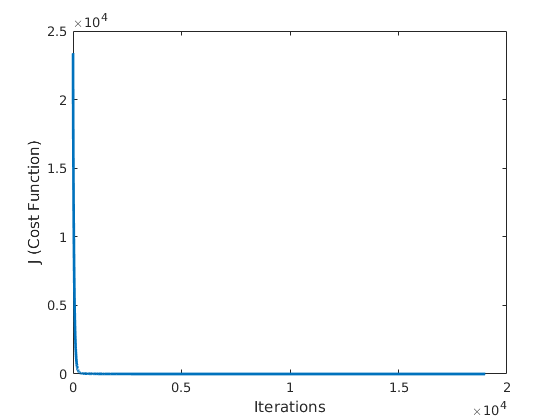
* + - Plot the results

%--------------------------------------------------------------------------

% Plotting the cost function

graph = plot(1:iterations, jHistory); set(graph,'LineWidth',2);

xlabel('Iterations'); ylabel('J (Cost Function)'); jHistory = jHistory';



*Published with MATLAB® R2017a*

4

## Multivariate Linear Regression with Reduced Features

#### Objective

In this MATLAB script, linear regression with multiple variabes has been code, built on the same script used earlier, but with 3 of the features reduced because correlation calculations. It gives a very slightly better accuracy (82.9282%) compared to the previous script.

clc clear close all

fileToRead = 'S&Pdata';

%Training would be done on 80% of data (1:550 out of 679) rangeTaken = 1:550;

% Import the complete spreadsheet file [xlsObjectComplete, xlsHeads] = xlsread(fileToRead);

% xlsHeads contains the headings in the form of a string vector

% Filter just the S&P Close into a vector SP\_Close = xlsObjectComplete(rangeTaken, 22);

xlsHeads = xlsHeads(2:21); % Remove the 'DATE' heading completeOP = xlsObjectComplete(:, 22);

xlsObjectComplete = xlsObjectComplete(:, 2:21); % Remove the date & OP coloumn

%--------------------------------------------------------------------------

* + - Feature reduction

xlsObjectComplete( :, [3, 4, 8] ) = []; % Remove 3, 4, 8 col

%--------------------------------------------------------------------------

inputMatrix = xlsObjectComplete(rangeTaken, 1:17); thetaWeights = zeros(17+1, 1);

% Learning rate, or rate of descent alpha = 0.01;

iterations = 19000; % Obtained after a few hit and trials.

%--------------------------------------------------------------------------

* + - Normalization

%-------------------------------------------------------------------------

1

Multivariate Linear Regres- sion with Reduced Features

X = inputMatrix;

X\_norm = X;

mu = zeros(1, size(X, 2)); sigma = zeros(1, size(X, 2));

for feature\_index = 1:size(X,2)

% Find mean

feature\_mean = mean(X(:,feature\_index));

% (datatpoint - mean)

X\_norm(:,feature\_index) = X(:,feature\_index) - feature\_mean;

% Find StdDev

feature\_std = std(X\_norm(:,feature\_index));

% (datatpoint - mean)/(stdDev)

X\_norm(:,feature\_index) = X\_norm(:,feature\_index) / feature\_std;

end

sigma(feature\_index) = feature\_std; mu(feature\_index) = feature\_mean;

% Repeat the above code for getting complete normalized input matrix

X = xlsObjectComplete(:, 1:17); X\_norm\_Complete = X;

mu = zeros(1, size(X, 2)); sigma = zeros(1, size(X, 2));

for feature\_index = 1:size(X,2)

% Find mean

feature\_mean = mean(X(:,feature\_index));

% (datatpoint - mean)

X\_norm\_Complete(:,feature\_index) = X(:,feature\_index) - feature\_mean;

% Find StdDev

feature\_std = std(X\_norm\_Complete(:,feature\_index));

% (datatpoint - mean)/(stdDev) X\_norm\_Complete(:,feature\_index) =

X\_norm\_Complete(:,feature\_index) / feature\_std;

end

sigma(feature\_index) = feature\_std; mu(feature\_index) = feature\_mean;

X\_norm\_Complete = [ones(679, 1), X\_norm\_Complete];

% X\_norm\_Complete stores ALL the input variables of ALL features in a

% normalized manner, and a one vector is appended in the beginning. This

% will be exported to Excel/CSV later.

%--------------------------------------------------------------------------

2

Multivariate Linear Regres- sion with Reduced Features

* + - Gradient descent algo

%--------------------------------------------------------------------------

m = length(SP\_Close);

X\_norm = [ones(m, 1) X\_norm]; % The normalized X X = [ones(m, 1) inputMatrix];

y = SP\_Close;

for i = 1 : iterations

temp = thetaWeights - (alpha/m) \* X\_norm' \* (X\_norm \* thetaWeights

- y);

thetaWeights = temp;

jHistory(i) = (1 / (2\*m) ) \* sum(((X\_norm \* thetaWeights)-y).^2);

end

%--------------------------------------------------------------------------

* + - Cost function algo

%-------------------------------------------------------------------------- J = 0;

fprintf('Cost function:\n');

J = (1 / (2\*m) ) \* sum(((X\_norm \* thetaWeights)-y).^2)

%--------------------------------------------------------------------------

*Cost function:*

*J =*

*0.1955*

* + - Output variable (predicted)

%--------------------------------------------------------------------------

% Take the 20% (551:679) of all the output variables for testing SP\_Close\_ToBePredicted = completeOP(551:679, 1);

% Preallocate yHat with zeros yHat = zeros(129, 1);

for i = 1 : 17

someTempVar = (thetaWeights(i) \* X\_norm\_Complete(551:679, i)); yHat = yHat + someTempVar;

end

difference = SP\_Close\_ToBePredicted - yHat;

accuracy = (difference./SP\_Close\_ToBePredicted)\*100; mean = mean(accuracy);

3

Multivariate Linear Regres- sion with Reduced Features

actualAccuracy = 100-mean

% Perform Outlier remover

outputX = outlierRemover(X\_norm\_Complete, xlsObjectComplete, 1, 1.75);

outputX = outlierRemover(X\_norm\_Complete, outputX, 2, 1.75);

outputX = outlierRemover(X\_norm\_Complete, outputX, 3, 1.45);

outputX = outlierRemover(X\_norm\_Complete, outputX, 4, 1.5);

outputX = outlierRemover(X\_norm\_Complete, outputX, 5, 1.9);

outputX = outlierRemover(X\_norm\_Complete, outputX, 6, 1.9);

outputX = outlierRemover(X\_norm\_Complete, outputX, 7, 1.4);

outputX = outlierRemover(X\_norm\_Complete, outputX, 8, 1.7);

outputX = outlierRemover(X\_norm\_Complete, outputX, 9, 1.8);

outputX = outlierRemover(X\_norm\_Complete, outputX, 10, 1.85);

outputX = outlierRemover(X\_norm\_Complete, outputX, 11, 1.7);

outputX = outlierRemover(X\_norm\_Complete, outputX, 12, 1.9);

outputX = outlierRemover(X\_norm\_Complete, outputX, 13, 1.95);

outputX = outlierRemover(X\_norm\_Complete, outputX, 14, 2);

outputX = outlierRemover(X\_norm\_Complete, outputX, 15, 2.2);

outputX = outlierRemover(X\_norm\_Complete, outputX, 16, 1.75);

outputX = outlierRemover(X\_norm\_Complete, outputX, 17, 2.5);

normalizedInputMatrix = maxNormalization(outputX);

% Export to CSV

% xlswrite('outlierOutput.xlsx', X\_Outlier);

*actualAccuracy = 82.9282*

*Published with MATLAB® R2017a*

4

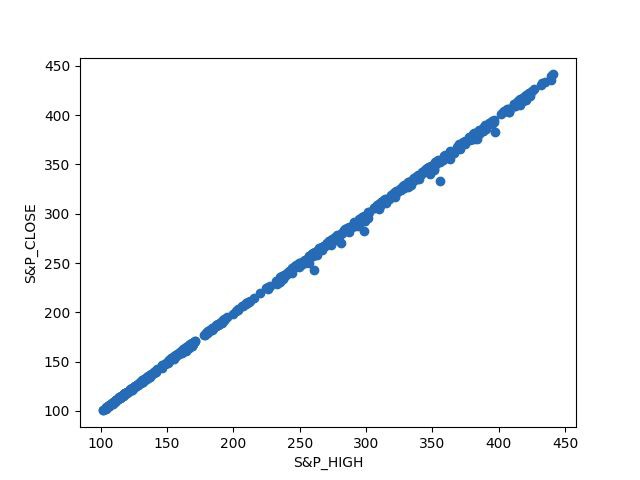
**PART - II**

###### Flaw in Normalization Method of Earlier Report

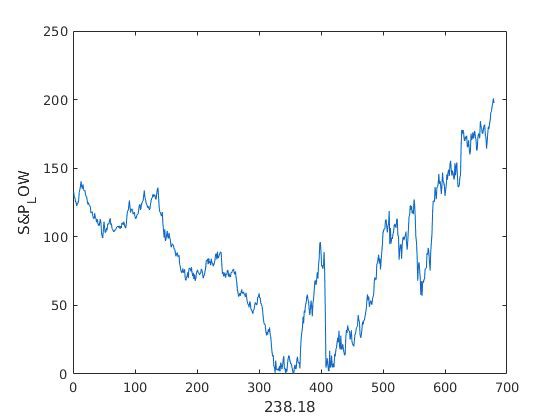
Earlier method opted for normalization was to calculate Z-score of every feature:

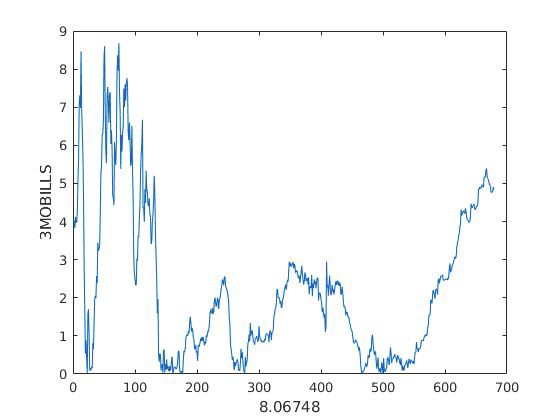
Z = (x-u)/σ

Where 𝛍 is the mean value of the feature and σ is the standard deviation. Higher the value of Z more is the tendency of being an outlier. This was the basis of normalisation of earlier report.

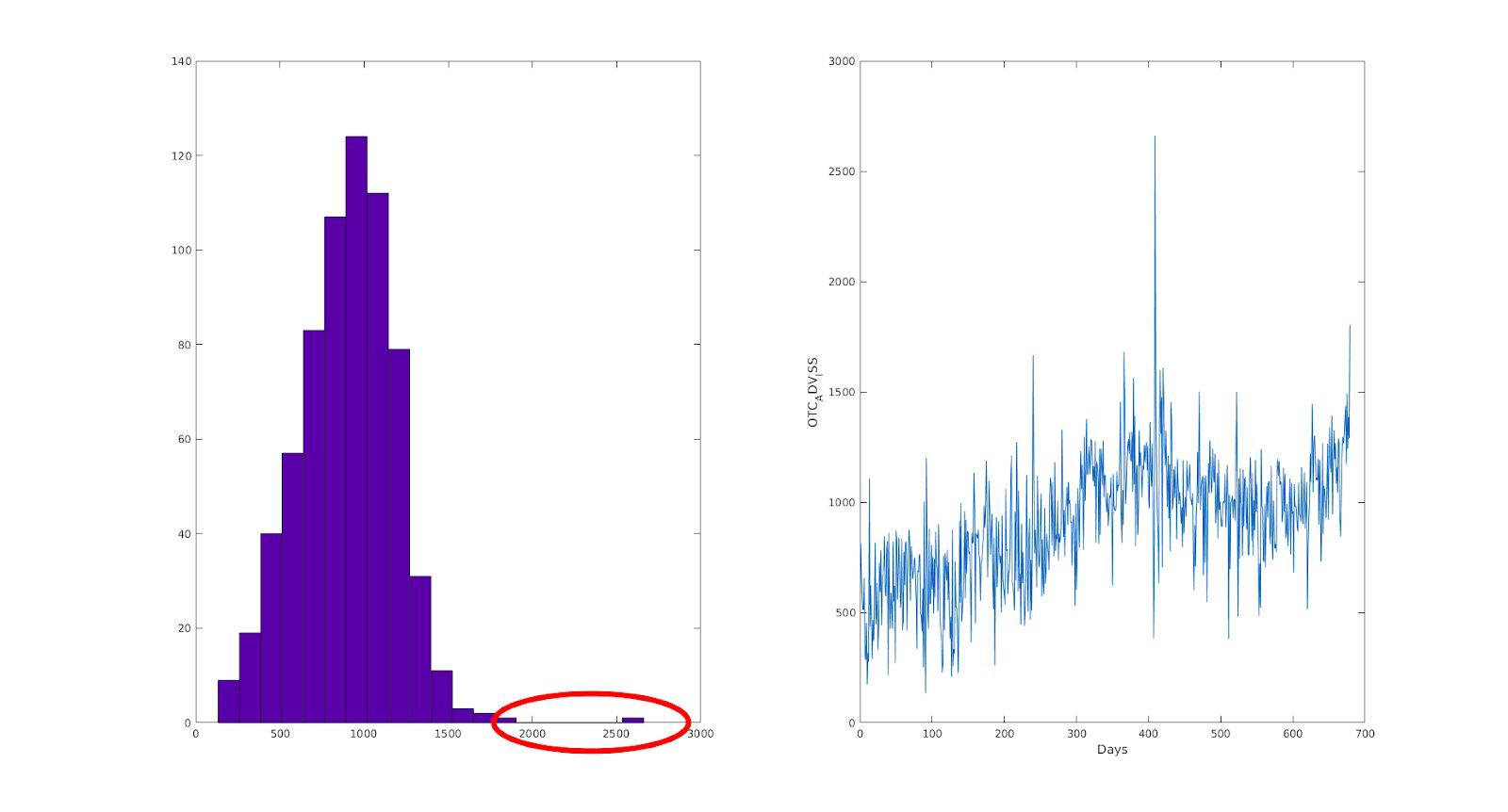
But we realized Since our dataset contains timestamp values (features dependant on previous time values) the initial values and the end values were treated as outliers.

As evident from the above diagram most of the end and starting date,valuable data is lost to the mean value. We further plotted the **| y - 𝛍 |** value of all the features and identified the timestamp features. The similarity between timestamp feature graph is that it results in the ‘V’ shape graph **| y - 𝛍 |**. The Normalisation failed in 10 features which are listed as (S&P Low, S&P High, Gold, 3 MOBILS, Long Bonds, S&P Earning, OTC\_ decline\_volume, OTC\_ total\_volume, OTC\_ advance\_volume, OTC\_ decline\_issues)



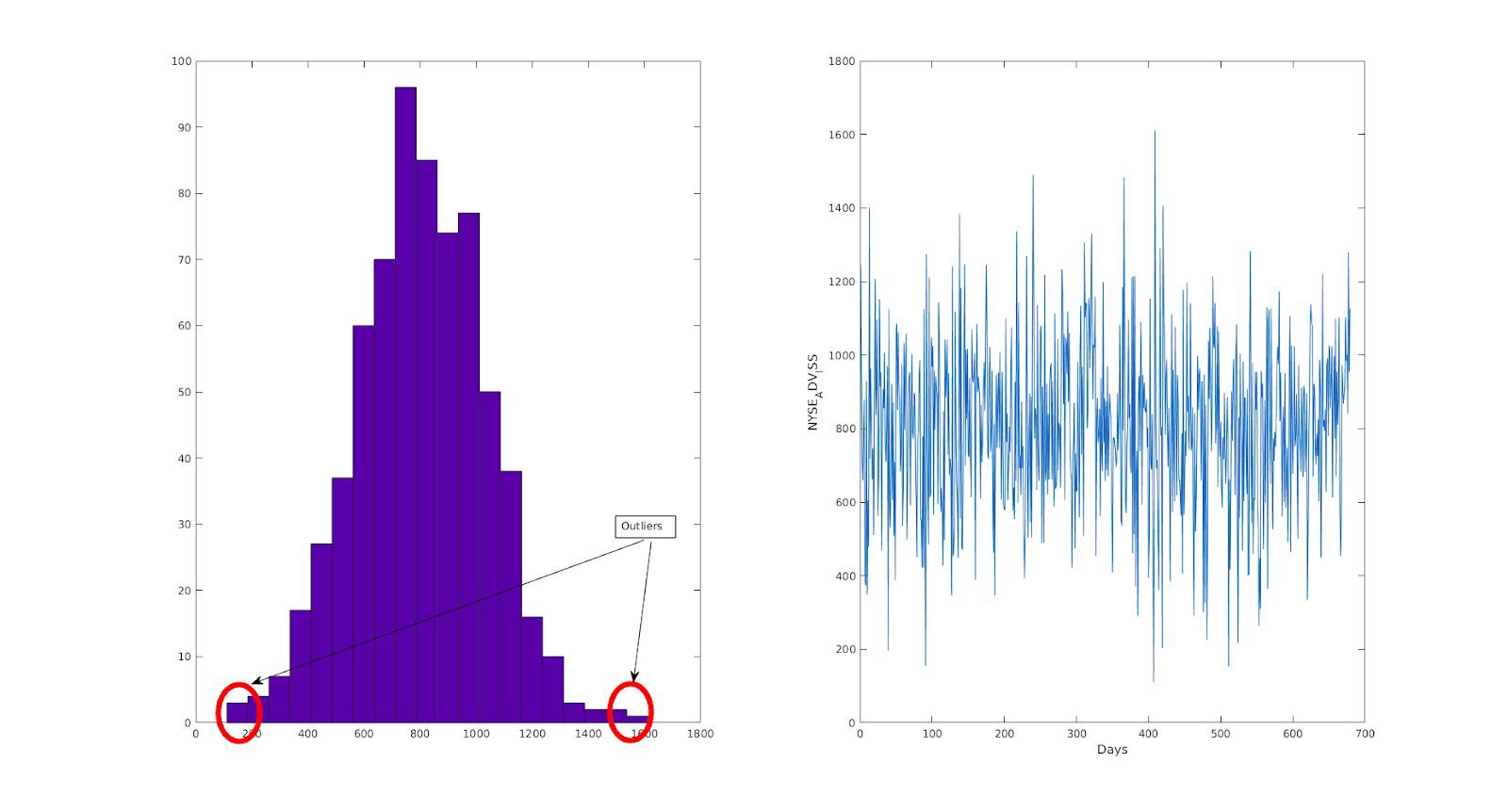


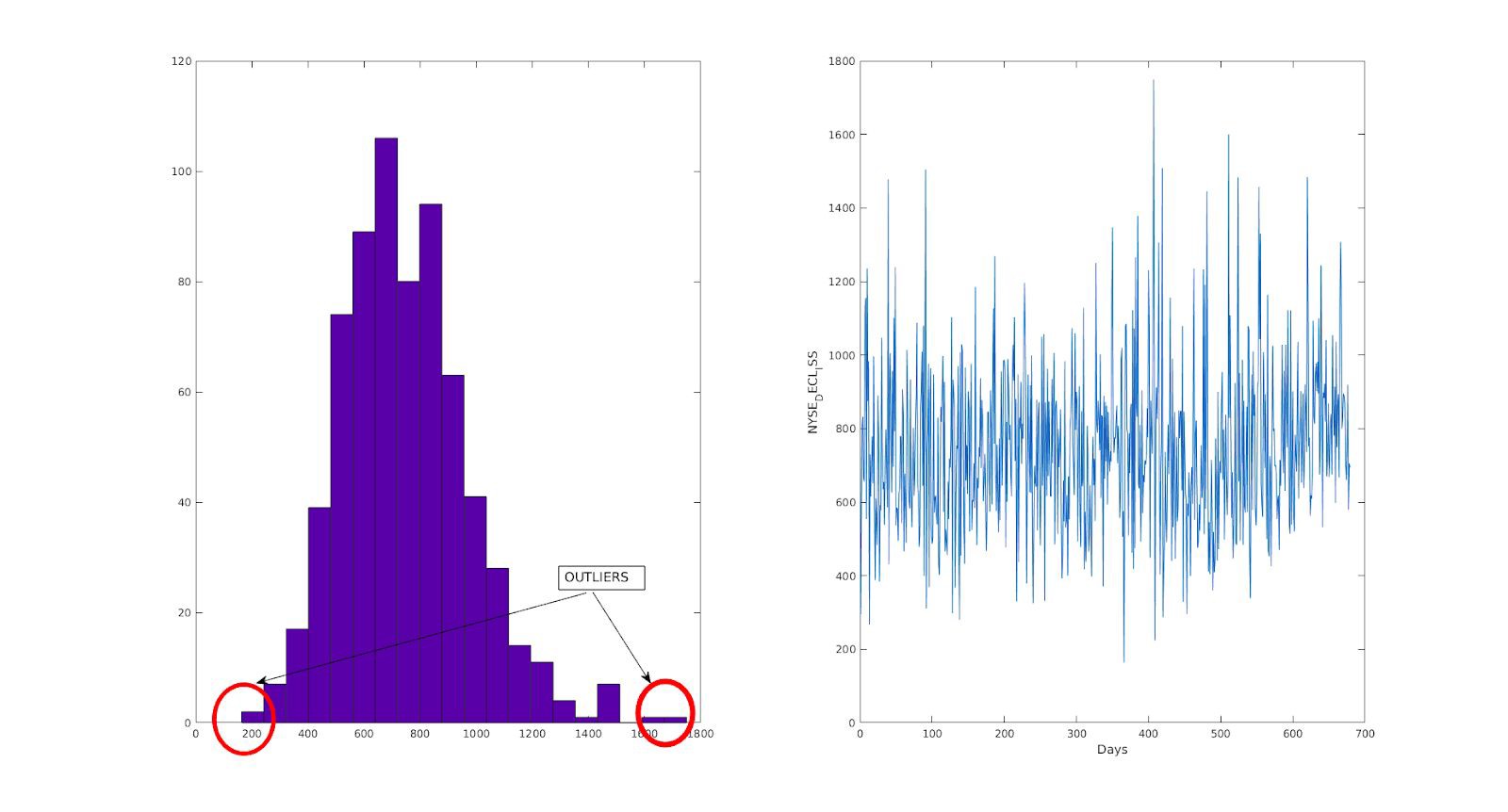
It is evident from the graph the **| y - 𝛍 |** value is maximum at the corners which results in losing out crucial data.

As as result the Z-score method to remove outliers were only implemented to non-timestamp features. From these Z-score values histogram diagrams were generated to see the frequency of each Z values in each interval. The threshold frequency of occurrence was considered to be 5.



Only the corner bars were considered as outliers cause they were attaining extreme values. Also it was made sure that important end values were not during the process.





**Flaw in Scaling Method of Earlier Report**

In the earlier report submitted the scaling process used to bring the features in the range of 0-1 was to divide the particular values of the feature with its maximum value. The flaw in this method is that the resultant values were not swinging from 0 to 1

instead from an arbitrary value to 1. As a result true scaling wasn’t achieved. The final method applied for bringing the swing from 0-1 is as follows.

(*Xj*

* *mean value of feature*) / (*maximum value of feature*)
* (*minimum value of feature*)

**Algorithms Applied Further**

Linear Regression

1. Mvregress was applied for the implementation of Linear Regression.
2. Before applying mvregress, the “cost function” coded by us was applied.

Linear Regression was applied on the following 4 matrices.

1. **Raw Data Matrix :** This is an untouched matrix provided to us
2. **Normalised\_Matrix :** In this matrix outliers are removed and all the features are scaled from 0 to 1
3. **Principal Component Analysis Matrix :** The PCA algorithm was applied 19 times on the Normaised\_Matrix.
4. **Reduced features matrix :** In this matrix based on the Domain knowledge and correlation matrix which features having minimum correlation value with respect output label was removed

For the project submission part 1 of midterm evaluation, we had used our own model of linear regression by creating our own code in MATLAB.Training had been done on 80% of data while testing of the data was done on the rest 20%. The accuracy of the results was 82.9055%.

Several attempts were made to improve our code, but that wasn’t possible and we simply couldn't improve our code.

So finally, we used an inbuilt function of MATLAB called mvregress(), which directly returns the weights after giving it our inputs and the output to be predicted.

Accuracy formula used by us in linear Regression is as follows

Accuracy = 1 - Σ(y - predicted output)/Σ(y - mean of y)

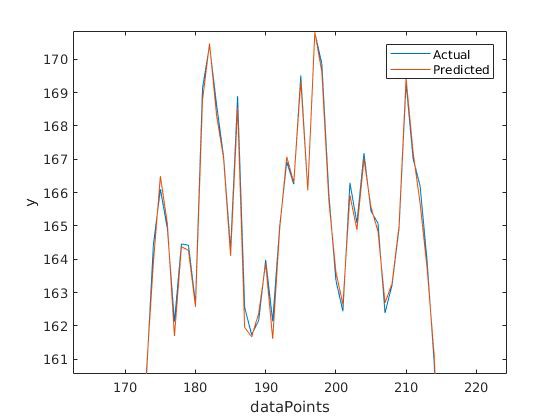
Training on raw data

At first, mvregress() was applied on raw data, and the results were a bit surprising.

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.100 | 0.4627 |



As can be seen from the graph, the actual values and the predicted ones were almost similar and were seeming to overlap. On a closer look though, we find:

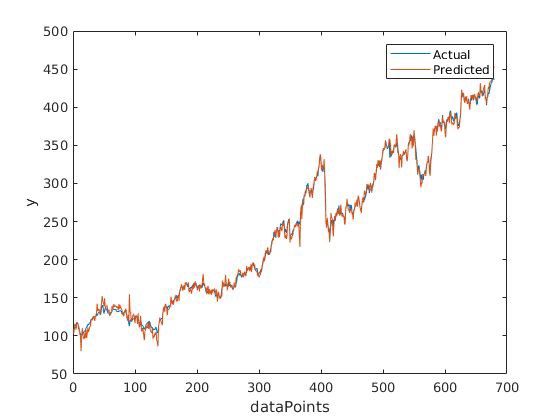


We find that there are differences, but very minute.

Training on normalized data

Training was now done on normalizzed data. This normalization was done on the basis of calculating the Z scores of some specific features and deciding upon a threshold. If there would be data points below the threshold values, those points would be replaced by the mean of the rest of the column's data.

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.9971 | 28.5097 |



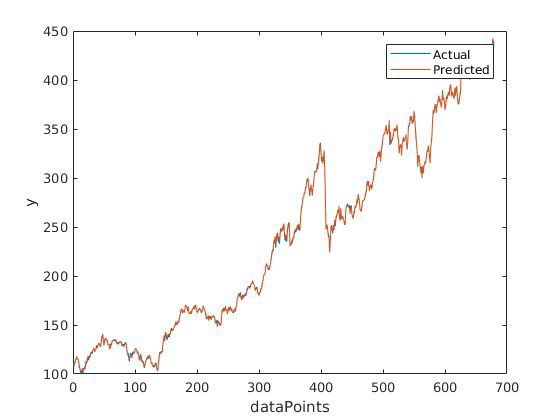
Both the accuracy kind of deteriorated, but it was better than the raw data’s case, because 100% accuracy is too good to be true, so in comparison to that, a 99% accuracy seems applicable.

Feature Reduction using PCA

Principal Component Analysis was tried to be incorporated in order to reduce some of the features of our dataset. MATLAB’s inbuilt function pca() was put to use for this by directly passing the input matrix to this function.

PCA - Level 1

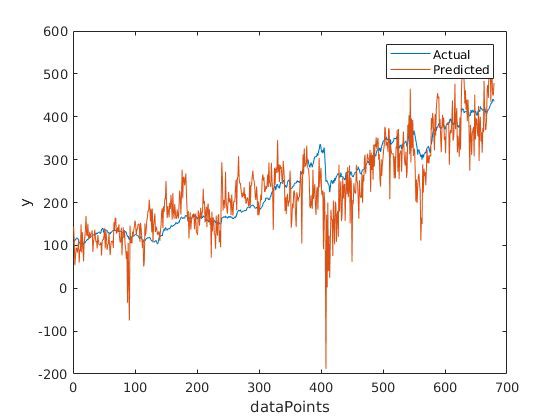
|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.9999 | 1.2822 |



The accuracy seems to have improved to 0.9999, and so does the MSE. MATLAB seems to decide feature reduction based on some inbuilt logic and do it on our dataset.

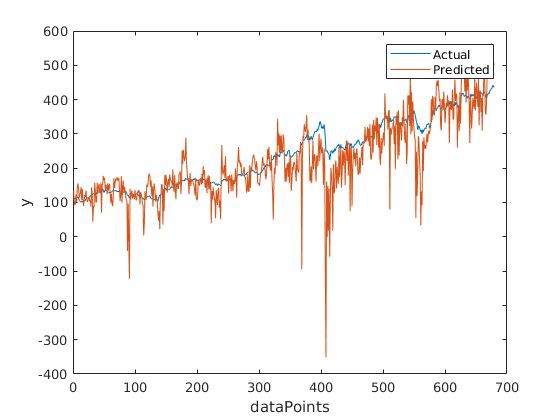
PCA - Level 2

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.6592 | 3363.1 |



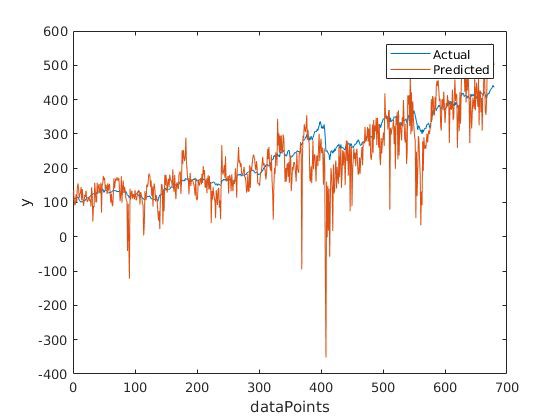
PCA - Level 3

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.5469 | 4471 |



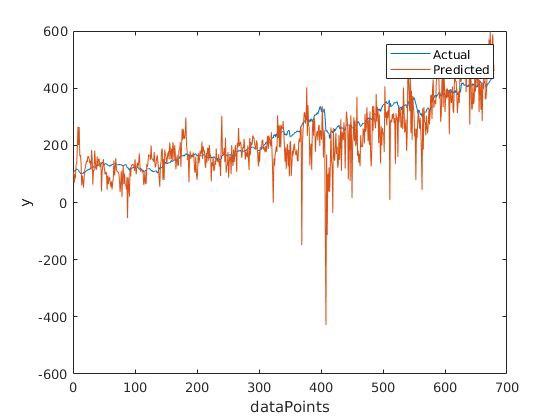
PCA - Level 4

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.5469 | 4471 |



PCA - Level 5

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.4144 | 5778.6 |



As we can see, the accuracy seems to be decreasing after apply PCA multiple times PCA only once was good enough to do some feature reduction.

,

Feature Reduction using correlation

Feature reduction was tried using the correlation coefficient matrix between each of the feature and output vector, which was made earlier in part 1 of the report. Since correlation coefficient gives values between 0 and 1, a threshold is now decided for this as follows - 10%, 15%, 30%, 50%, 55% and 60% of the correlation value. Values below the decided threshold value would be discarded (the whole feature would be discarded). Following were the results obtained:

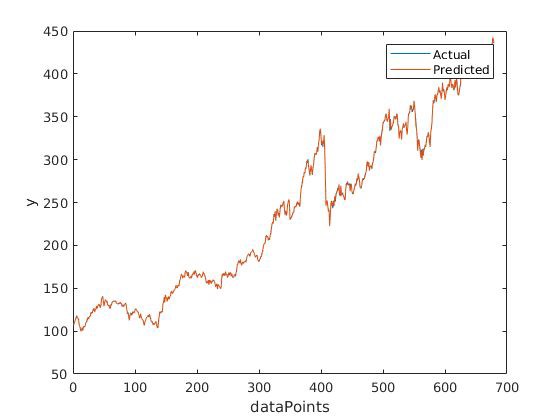
Threshold <10%

|  |  |
| --- | --- |
| Acuuracy | MSE |
| 0.9999 | 0.5127 |



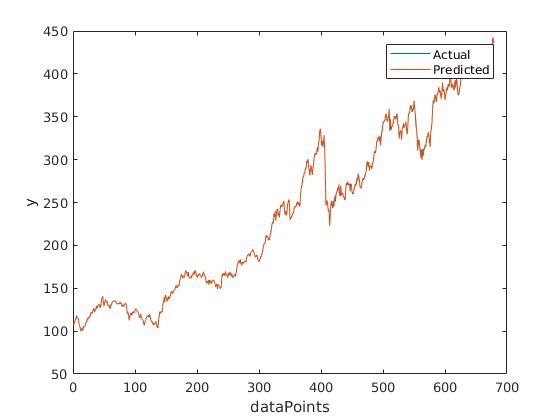
Threshold <15%

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.9999 | 0.5254 |



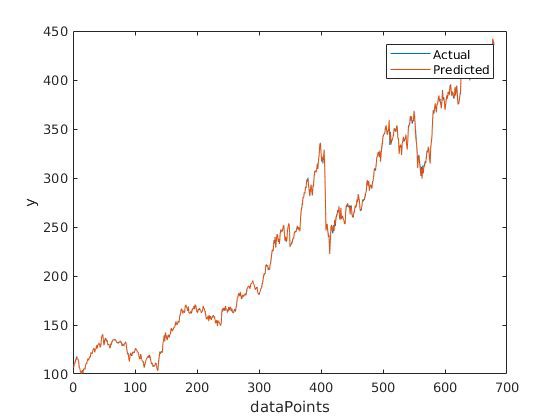
Threshold <30%

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.9999 | 0.5275 |



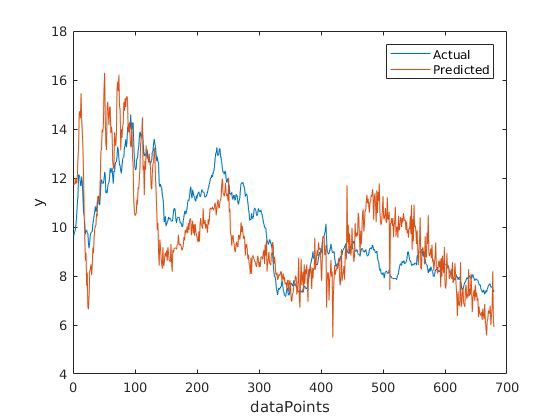
Threshold <50%

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.9999 | 0.5483 |



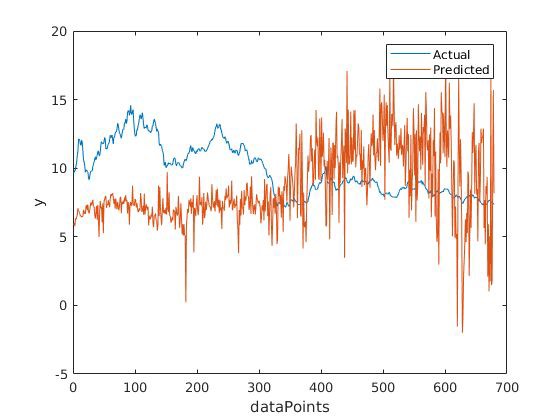
Threshold <55%

|  |  |
| --- | --- |
| Accuracy | MSE |
| 0.4043 | 2.0759 |



Threshold <60%

|  |  |
| --- | --- |
| Accuracy | MSE |
| -3.8393 | 16.8647 |



As can be seen from the above graphs, the MSE would always gradually increase, but the accuracy would remain constant and starts deteriorating after increasing the threshold of removal point to large threshold value.

The results can be summarized in the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Reduction scheme** | **Accuracy** | **MSE** | **Comments** |
|  |  |  |  |
| Raw data | 100.00% | 0.4627 | Too good to be true |
| Normalized data | 0.9971 | 28.5097 | Good accuracy, but MSE is large |
| PCA1 | 0.9999 | 1.2822 | Good accuracy with good MSE |
| PCA2 | 0.6592 | 3.36E+03 | Both deteriorating |
| PCA3 | 0.5469 | 4.47E+03 | Both deteriorating |

|  |  |  |  |
| --- | --- | --- | --- |
| PCA4 | 0.5469 | 4.47E+03 | Both deteriorating |
| PCA5 | 0.4144 | 5.78E+03 | Worst accuracy and MSE |
|  |  |  |  |
| (Threshold <10%) | 0.9999 | 0.5127 | [ NYSE\_ADV\_ISS, NYSE\_DECL\_ISS, NYSE\_NEW\_LOWS ,  OTC\_NEW\_LOW ] - **removed** |
| (Threshold <15%) | 0.9999 | 0.5254 | [OTC\_NEW\_HIGHS ,  NYSE\_NEW\_HIGHS, previous] -  **removed** |
| (Threshold <30%) | 0.9999 | 0.5275 | [ GOLD, previous ] - **removed** |
| (Threshold <50%) | 0.9999 | 0.5483 | [ S&P\_Earning, NYSE\_ADV\_VOL, OTC\_ADV\_ISS, previous ] - **removed** |
| (Threshold <55%) | 0.4043 | 2.0759 | [ NYSE\_DECL\_VOL,  OTC\_DECL\_ISS, previous ] -  **removed** |
| (Threshold <60%) | -3.8393 | 16.8647 | [ 3MOBILS, previous ] - **removed** |

**Support Vector Machines**

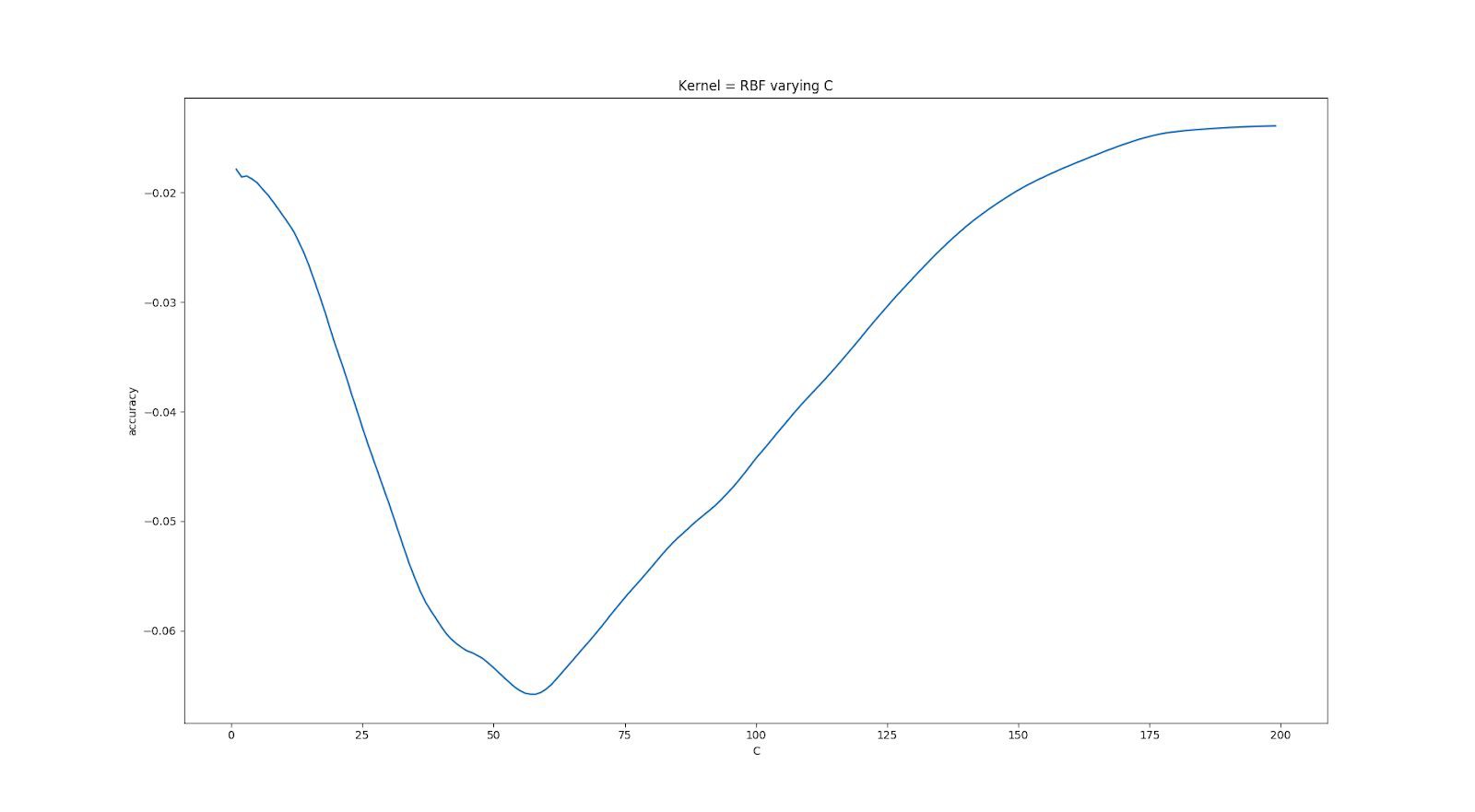
Support Vector Machine was applied on the following 4 matrices.

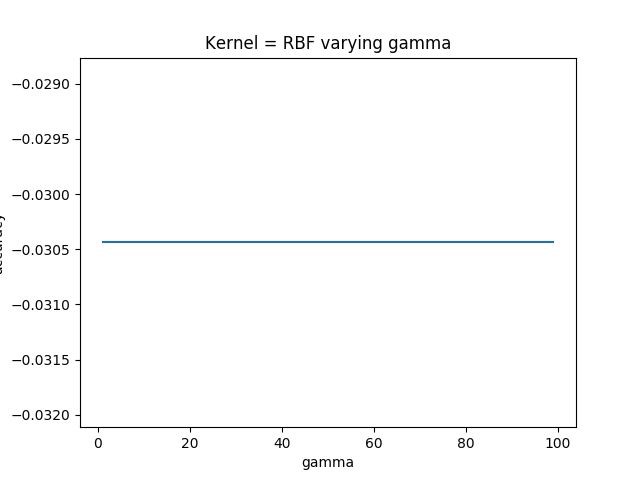
1. **Raw Data Matrix :** This is an untouched matrix provided to us
2. **Normalised\_Matrix :** In this matrix outliers are removed and all the features are scaled from 0 to 1
3. **Principal Component Analysis Matrix :** The PCA algorithm was applied 19 times on the Normaised\_Matrix.
4. **Reduced features matrix :** In this matrix based on the Domain knowledge and correlation matrix which features having minimum correlation value with respect output label was removed

Four kernels of SVM were applied on the matrices which are RBF, Linear, Poly , Sigmoid their best values of C and gamma were also founded by plotting various graphs. The SVM algorithm was applied on 80% of the data and 20% was used for testing.

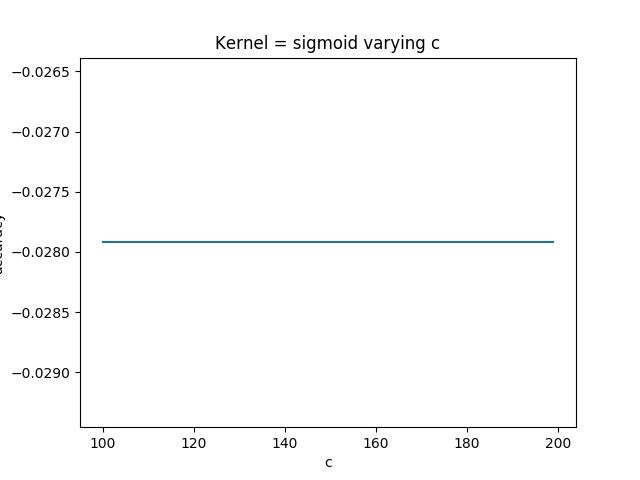
**Raw Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **RBF** | **Poly** | **Linear** | **Sigmoid** |
| **1%** | **Not Computable** | **Not Computable** | **-2%** |

SVM displayed very poor accuracy on RBF and Sigmoid, also upon applying Poly and Linear the program kept running and was forced to exit().



As it is evident from the above graph by varying gamma the accuracy of RBF kernel doesn’t change and stays constant.

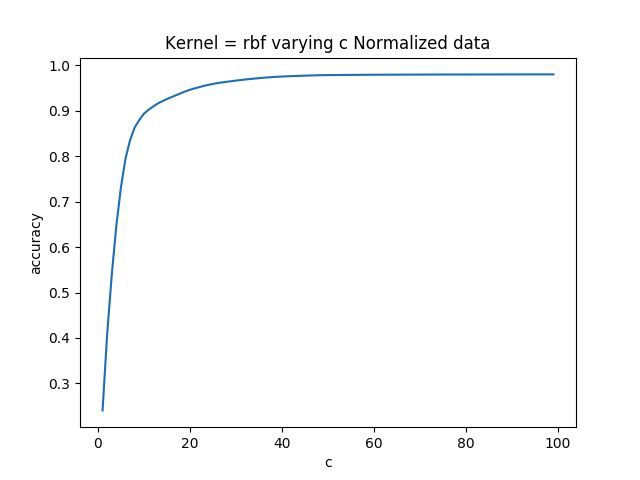


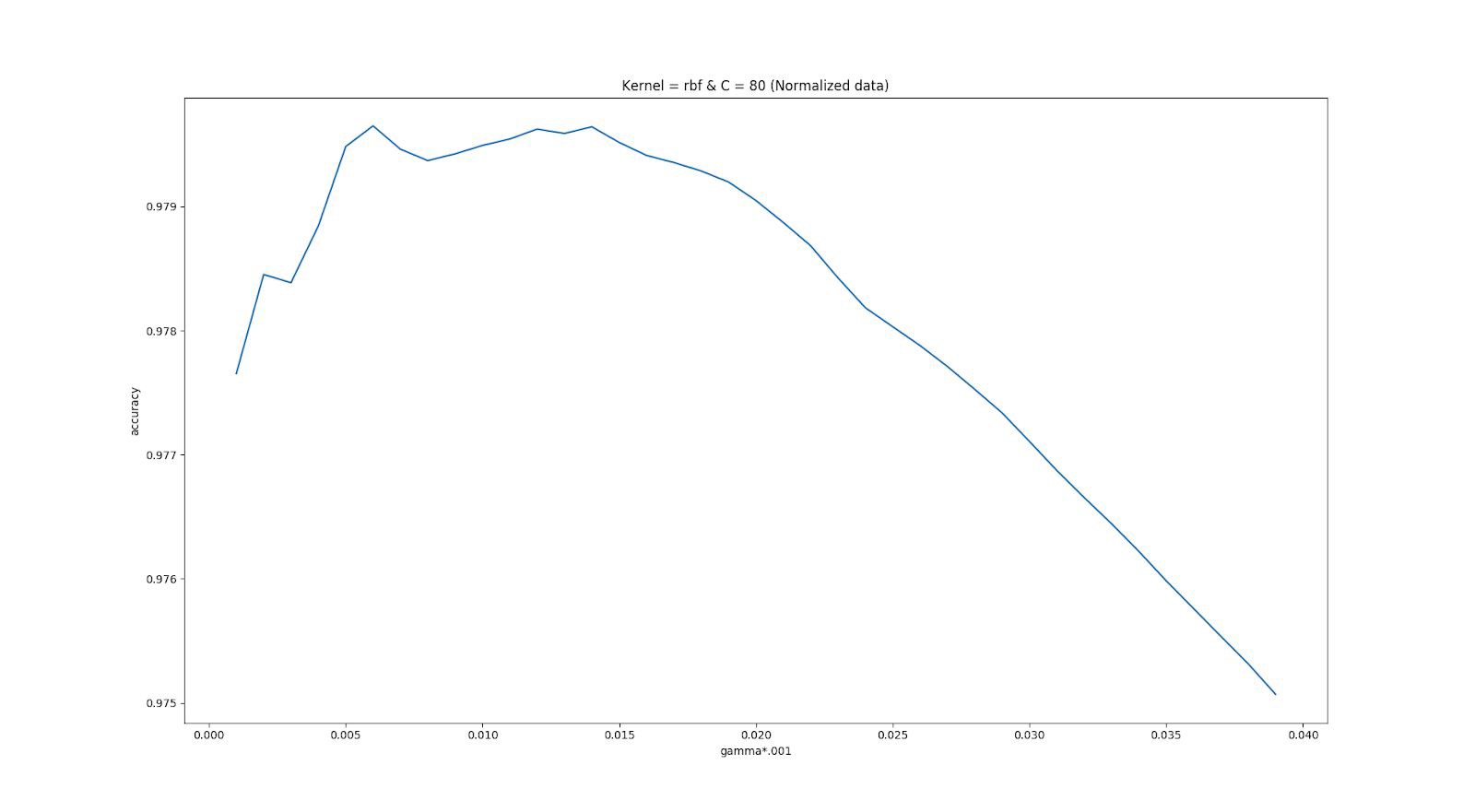
The value of sigmoid function remains constant even if the value of C is varied.

**Normalised Data**

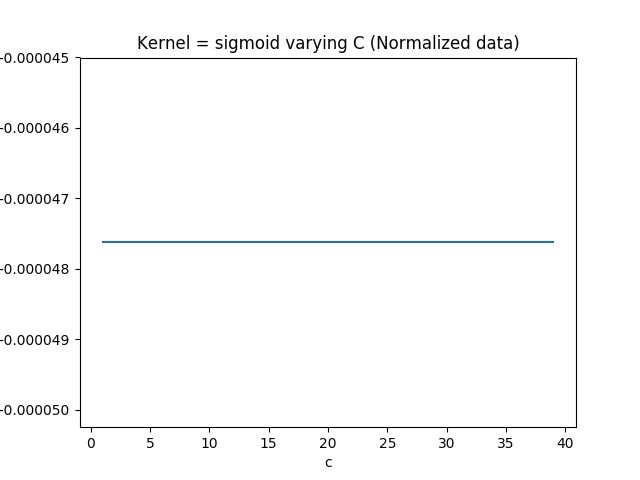
|  |  |  |  |
| --- | --- | --- | --- |
| **RBF** | **Poly** | **Linear** | **Sigmoid** |
| **98%** | **Not Computable** | **97.8%** | **-0.1%** |

This time SVM showed good results, RBF and Linear kernels were showing accuracy as high as 98%. The value of C and gamma were found out for RBF using the graphs, as it can be seen below the best accuracy is when C = 80 after that incrementing the value of C doesn’t have any effect on accuracy and it saturates





The above image clearly shows that at gamma = 0.006 maximum accuracy was achieved, and after that it started decreasing upon incrementing gamma.



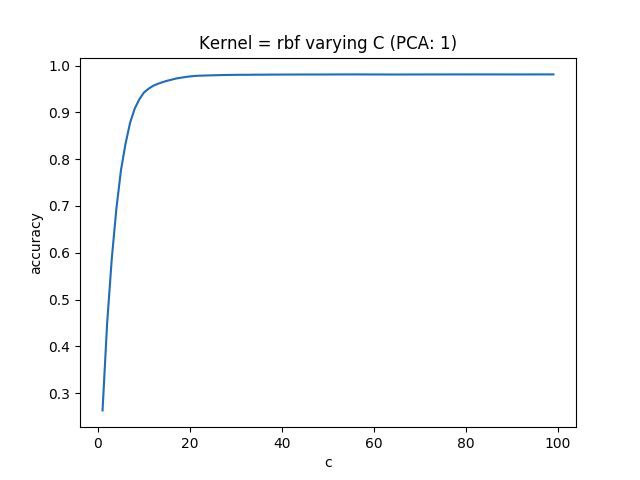
Sigmoid kernel can be seen as giving constant poor accuracy even when C was varied.

**Principal Component Analysis Matrix**

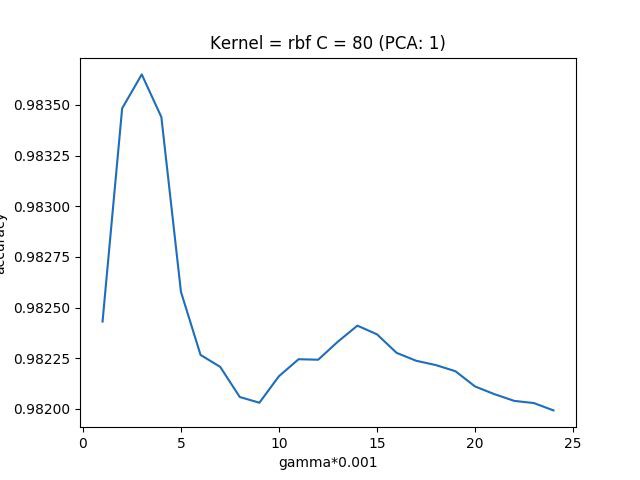
PCA algorithm was applied to normalised matrix as many as 19 times the results are listed below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **RBF** | **Poly** | **Linear** | **Sigmoid** |
| **PCA 1** | **98.36%** | **Not Computable** | **98.07%** | **-0.2%** |
| **PCA 2** | **97.2%** | **Not Computable** | **98.05%** | **-0.2%** |
| **PCA 3** | **97.2%** | **Not Computable** | **98.5%** | **-0.1%** |
| **PCA 4** | **97.4%** | **Not Computable** | **98%** | **-0.3%** |

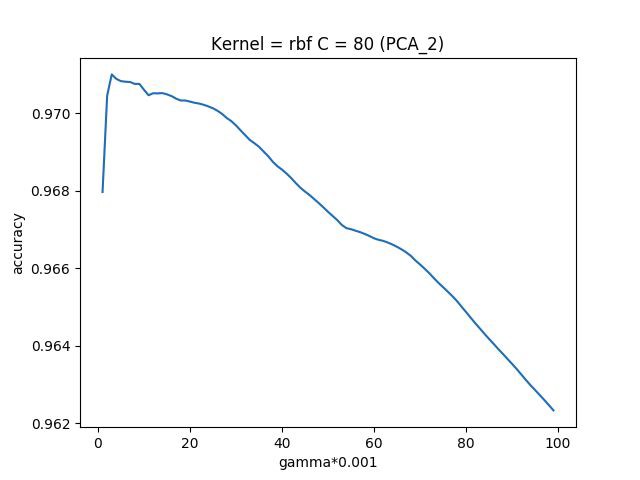
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PCA 5** | **-** | **Not Computable** | **98.57%** | **-0.2%** |
| **PCA 6** | **-** | **Not Computable** | **97.55%** | **-0.1%** |



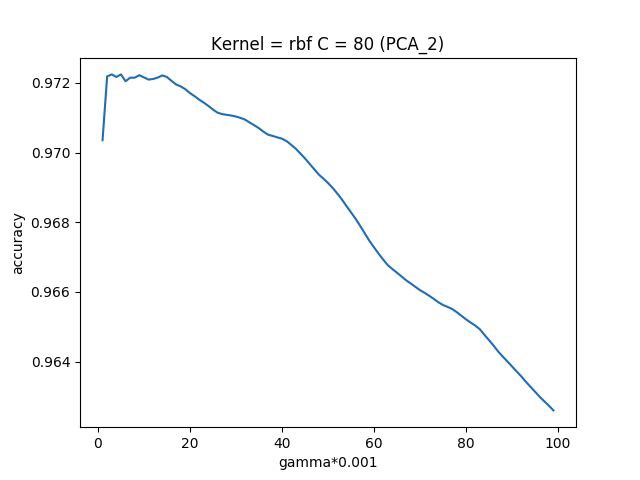
Sigmoid function was giving absurd values and Poly function always had to be forcefully terminated on the other hand RBF and Linear kernel were giving decent values. In the above diagram RBF was implemented after applying PCA algorithm on the normalised dataset only for one time. The best value of C that came out was 80. After keeping C constant at 80 gamma value was found out which was 98.4%, on further increasing the

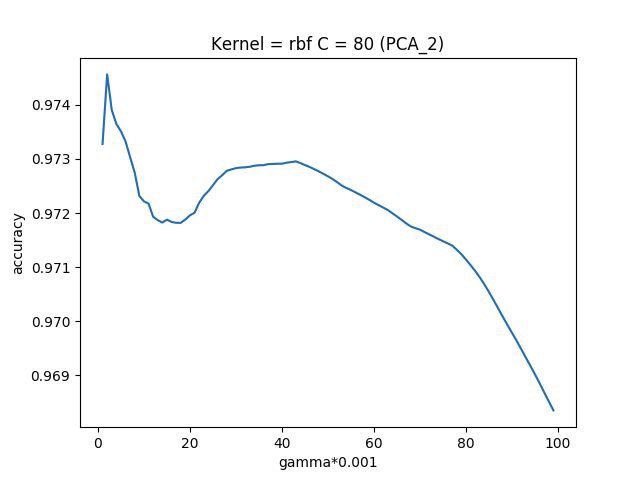
gamma the accuracy value deteriorated more it achieved maximum at 0.004.

In the below graph RBF was implemented after PCA function was called two time here we observed that value of C was similar to previous PCA 1 which is 80.



The best value of gamma also came out to be 0.008 which was same as the previous case of PCA\_2. Upon observing this, the value of C and Gamma was kept constant for all the PCA dataset at 80 and 0.004.





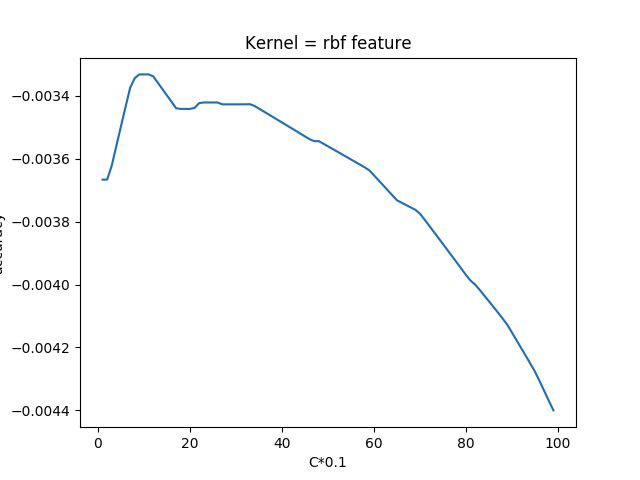
PCA 3 also gave similar results maximum value peaked at C = 80 and gamma = 0.004.

**Reduced Feature Dataset**

Upon Reducing the features with less than 10% correlation with output matrix, SVM algorithm no longer worked. Accuracy always remained a negative number and many kernels were not able to do computation on the reduced feature dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RBF | Poly | Linear | Sigmoid |
| Reduced Feature(<10% correlation) | **-0.3%** | **Not Computable** | **Not Computable** | **Not Computable** |

Since accuracy was effected so much it was decided not to apply SVM on further feature reduced matrix.



**Neural Networks**

* 1. There are 4 type of matrices to which the algorithm has been applied which are as follows -
     1. Raw\_Data\_Matrix
     2. Normalised\_Matrix
     3. Principal\_Component\_Analysis\_Matrix

**On Raw\_Data\_Matrix**

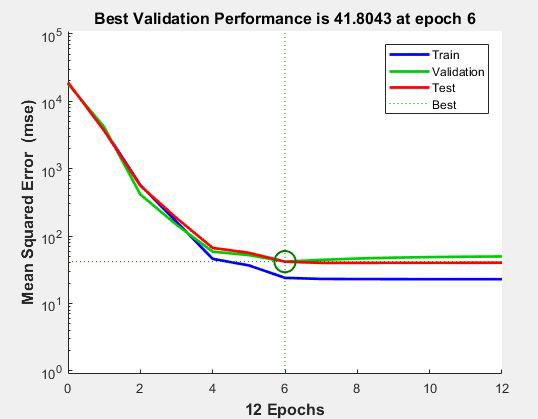
**Application of Nonlinear\_Autoregressive\_Response**

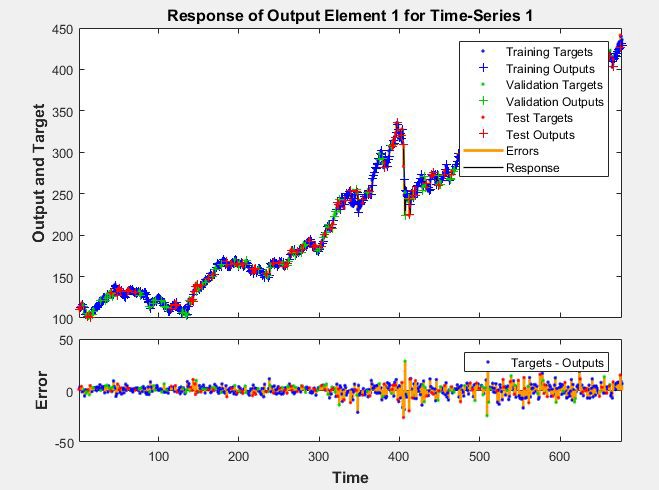
The application of NAR with the Target (S&P\_CLOSE) resulted in the following graph with MSE( Mean Square Error) increasing as we move forward in the timesteps.

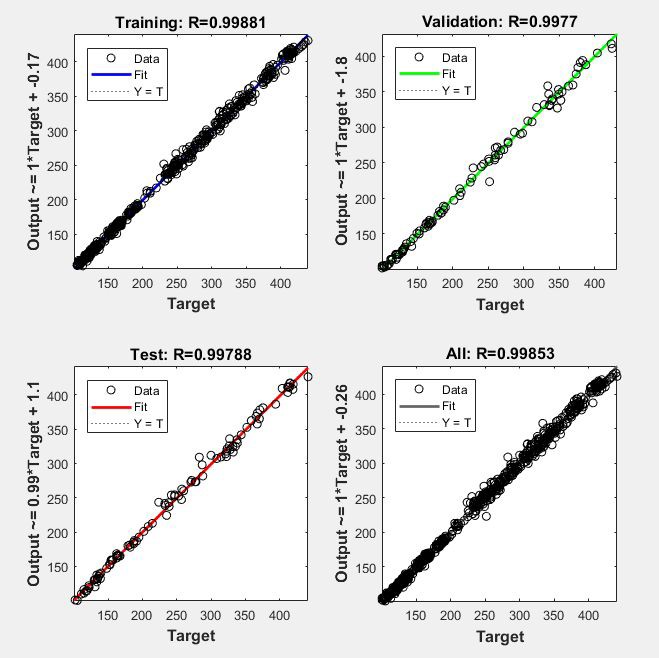
Here MSE has been taken as the Measure for performance. Training - 70%

Validation - 15%

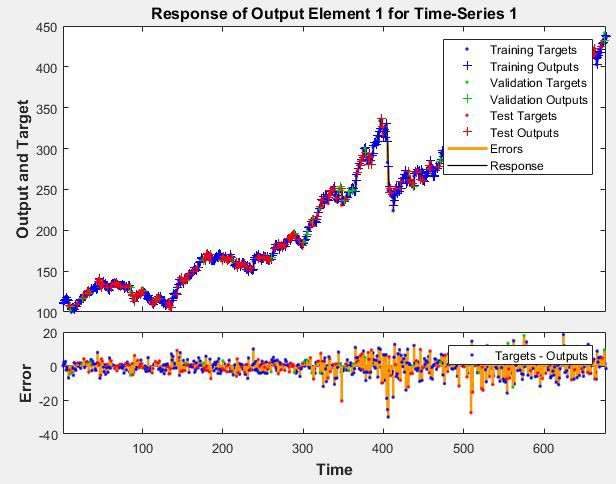
Testing - 15%

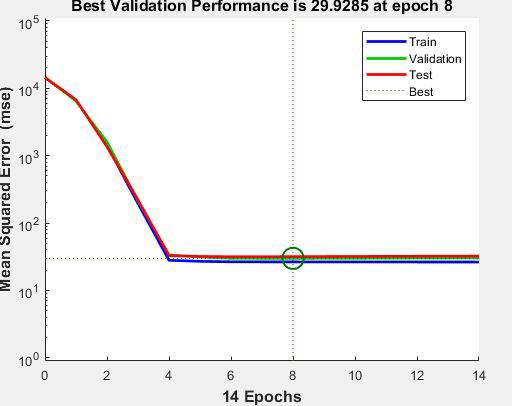


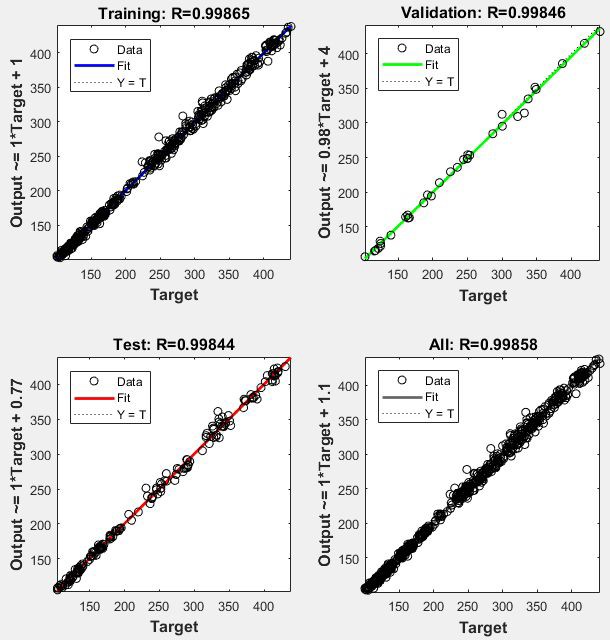




**Performance Plot**

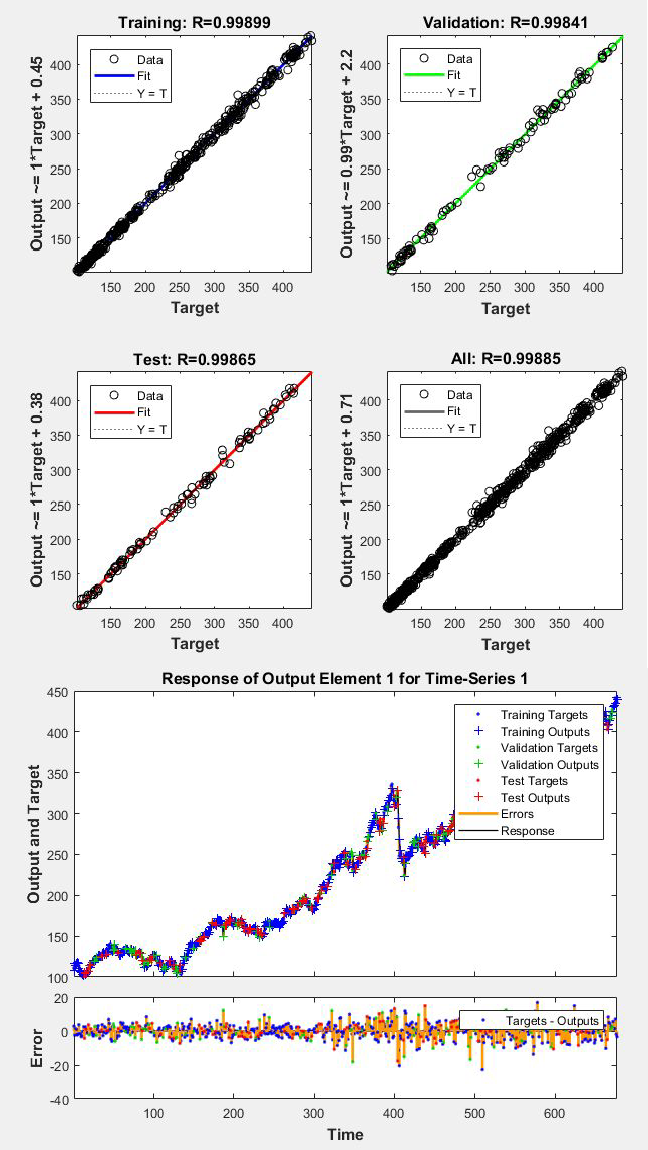
**Training - 70 % Testing - 5% Validation - 25%**





**Application of Nonlinear\_Autoregressive Exogenous Inputs (NARX)**

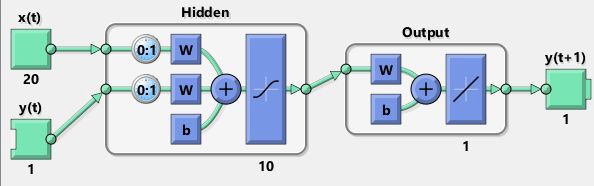


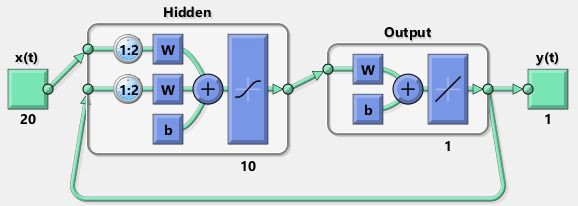
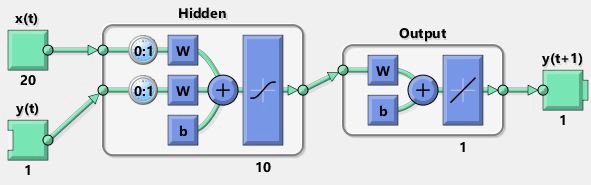
In this the Neural Network is trained at a configuration where

Training Data - 70%

Validation Data - 15%

Testing Data - 15%



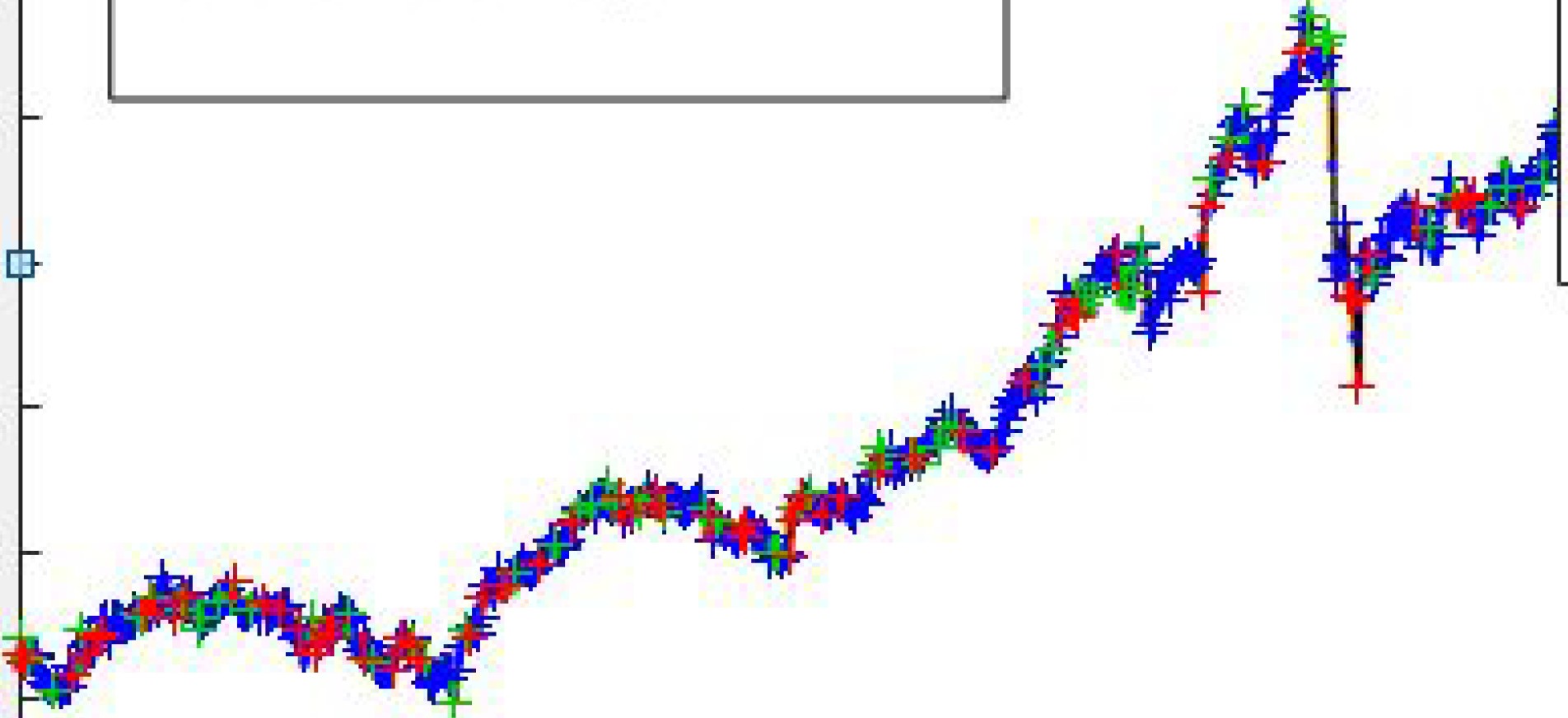


**On Normalised\_Data\_Matrix**

**Application of Nonlinear\_Autoregressive Exogenous Inputs (NARX)**

**Response of Output Element 1 for Time-Series 1**

450r,i..-- - - - - - - - - -,----{cl-- --,-- - - --,-- - - ----.-- ----EJ



Training Data - 70% Testing Data - 15 % Validation Data -15%

Training Targets

+ Training Outputs

Validation Targets

+ Validation Outputs

Test Targets

+ Test O utputs

* Errors
* - Response

400

,.,.. 350

**Cll**

.**C**..**l**

**ro** 300

**1-**

**"C**

.**[**.**i**.**;** 250

**:I**

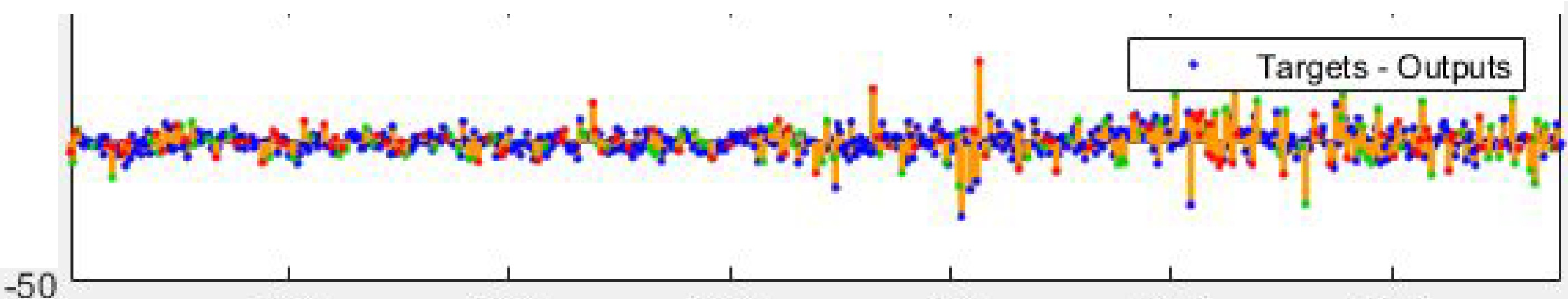
..**a**.**.** 200

**:I**

**O** 150

100

50 ~~·~~



50 - - - - - - - - - - - - - - - - - - - - - - - - - -

0

...

**0**

I:

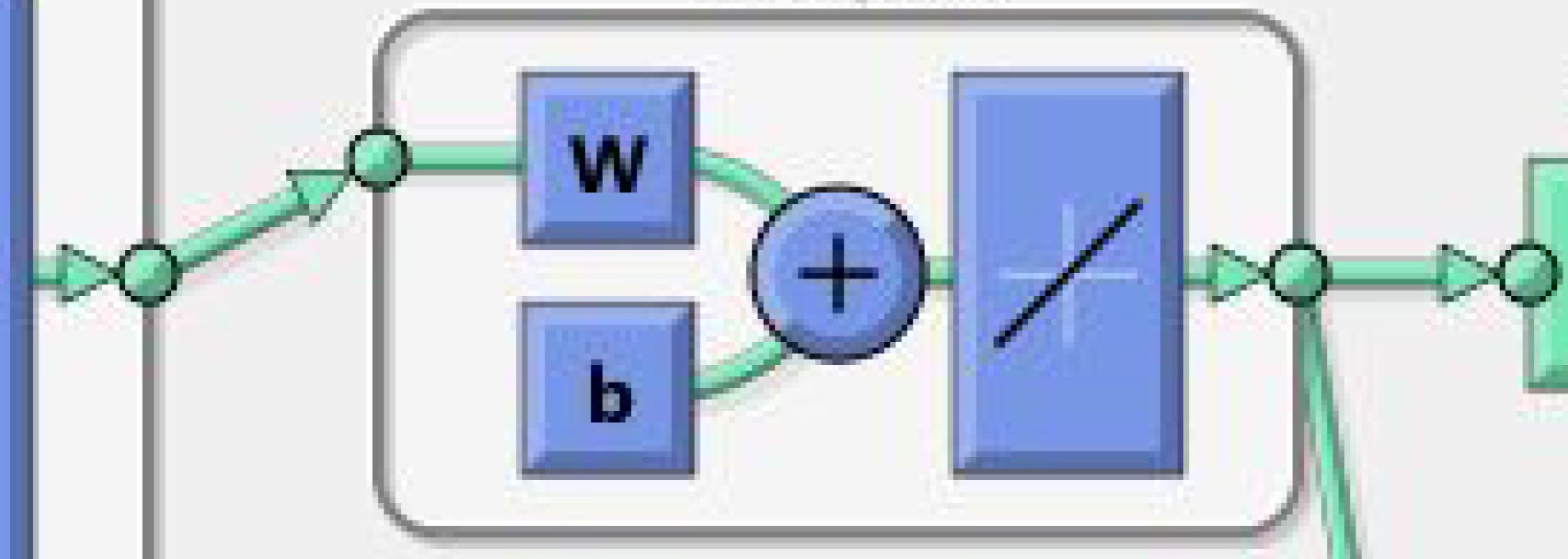
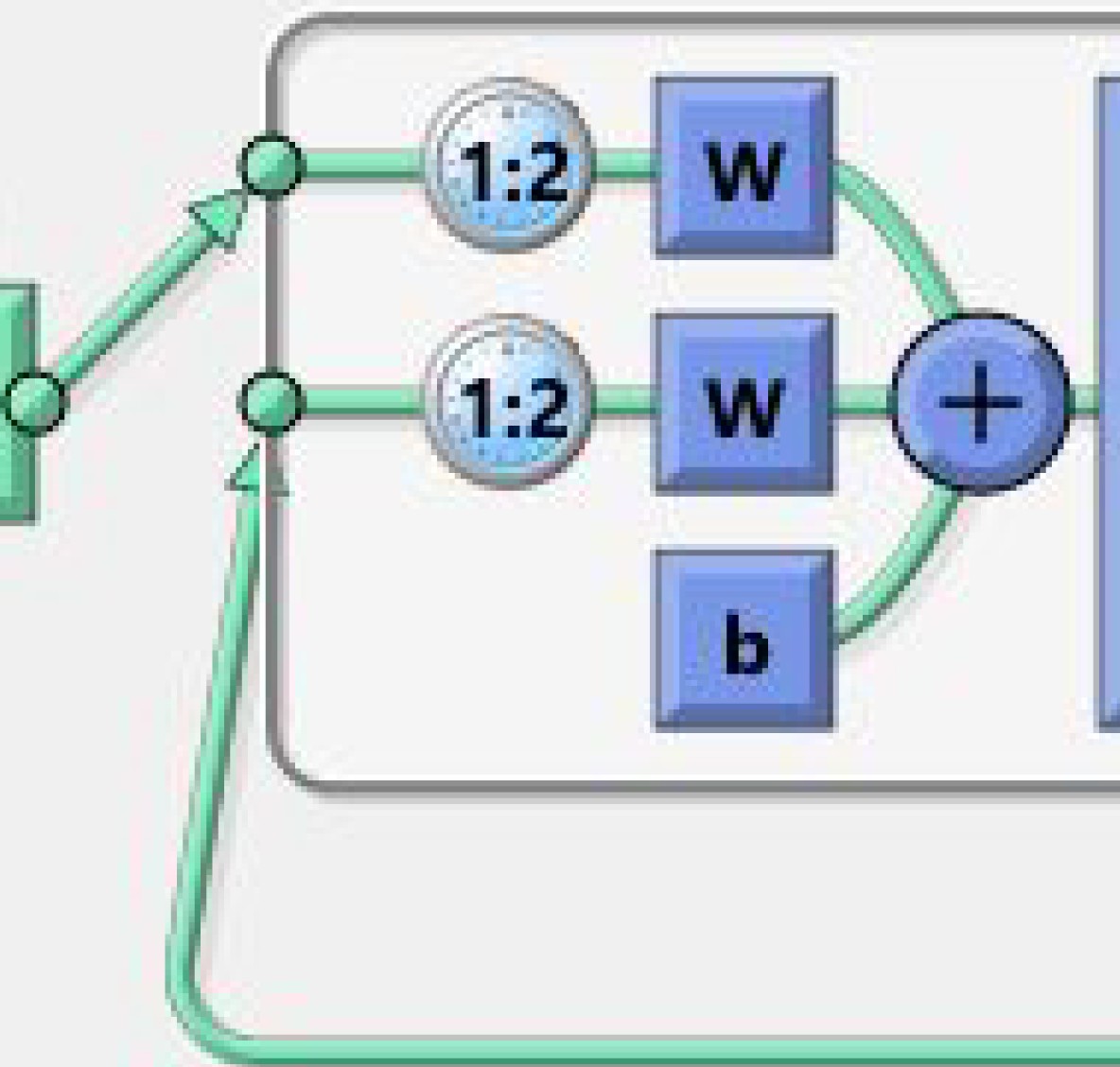
**w**

100 200 300 400 500 600

**Time**

**Hidden**

**x(t) y(t)**



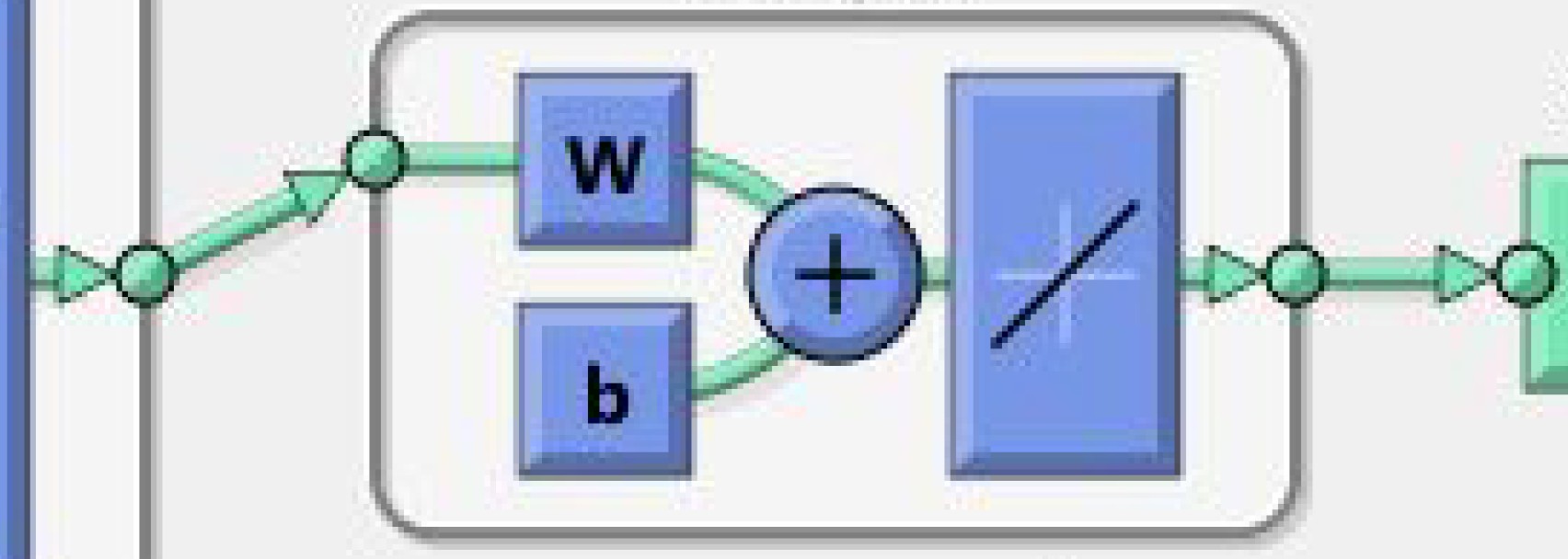
**Output**

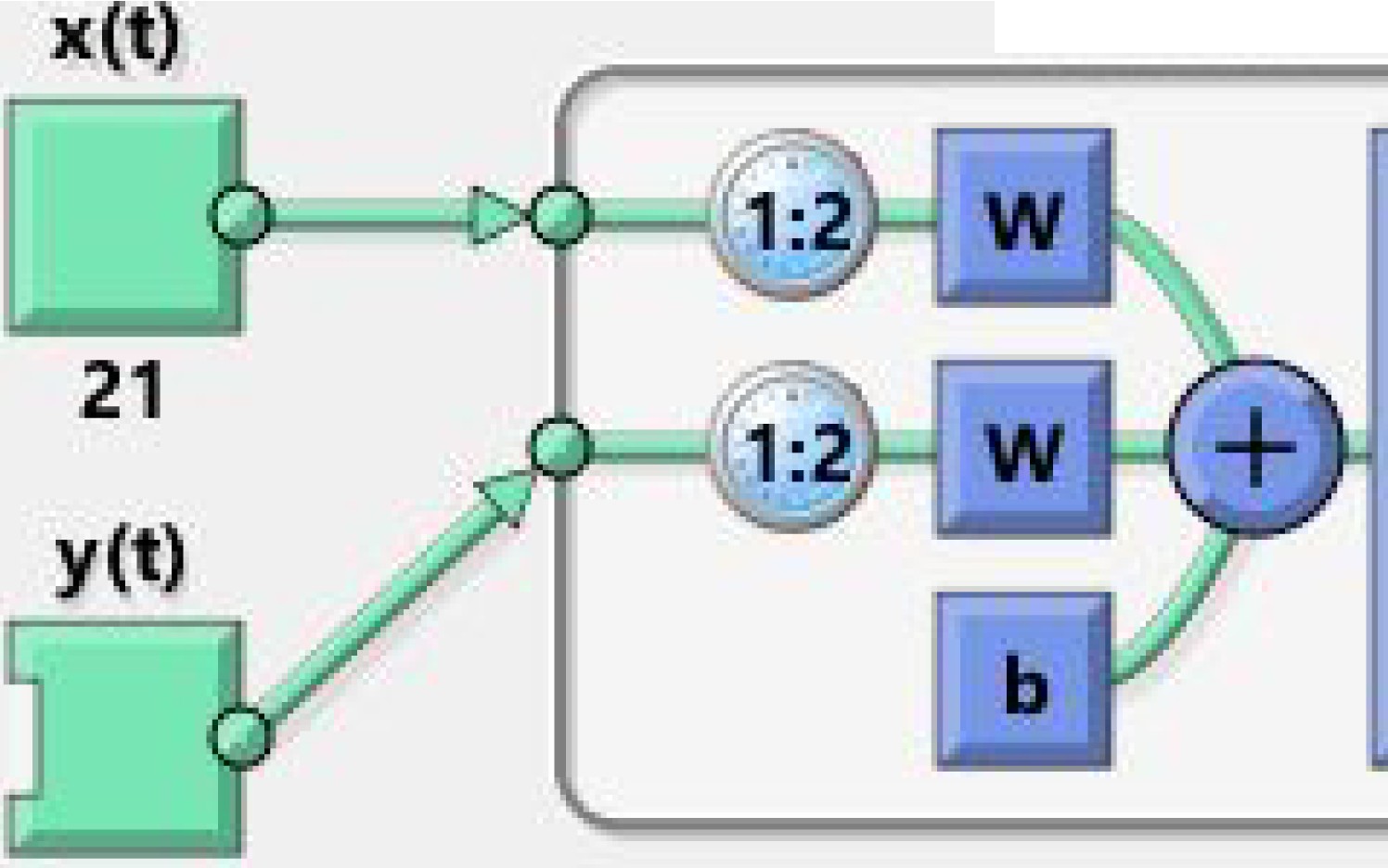
/

**1**

**10**

**21 1**

**Output**



**Hidden**

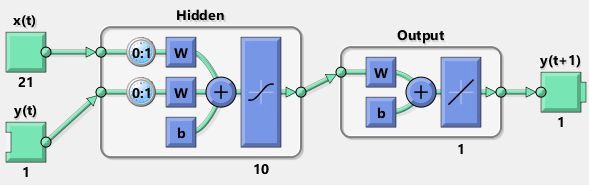
/

**y(t)**

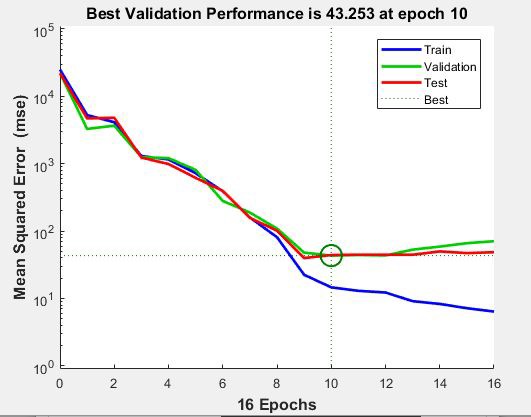
**1**

**1**

**1 10**



**On Principal Component Analysis Matrix**

**Application of Nonlinear\_Autoregressive Exogenous Inputs (NARX)**

**Response of Output Element 1 for Time-Series 1**

450 - - - - - - - - - - - - - - - - - - - - - - - - - - ---,



Training Targets

+ Training Outputs

Validation Targets

+ Validation Outputs

Test Targets

+ Test Outputs

* Errors
* - Response

400

..., 350

...

**Q)**

**Cl**

**(a** 300

**t-**

**"C**

..**1**.**5**. 250

**:I**

..**C** 200

**:I**

**O** 150

100

50 '-------"'-----'-----..

..

..

\_. \_ ,

0......

**W**

20 ,--------,,-----,----------------------,

100 200 300 400 500 600



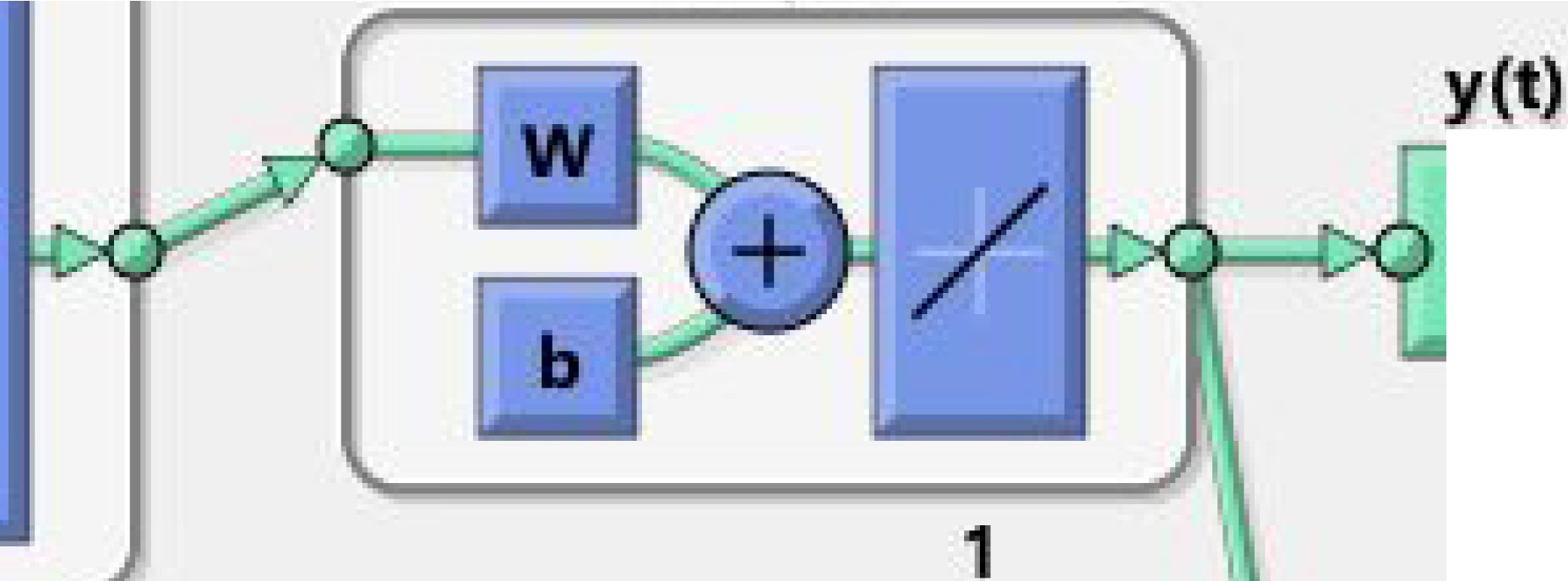
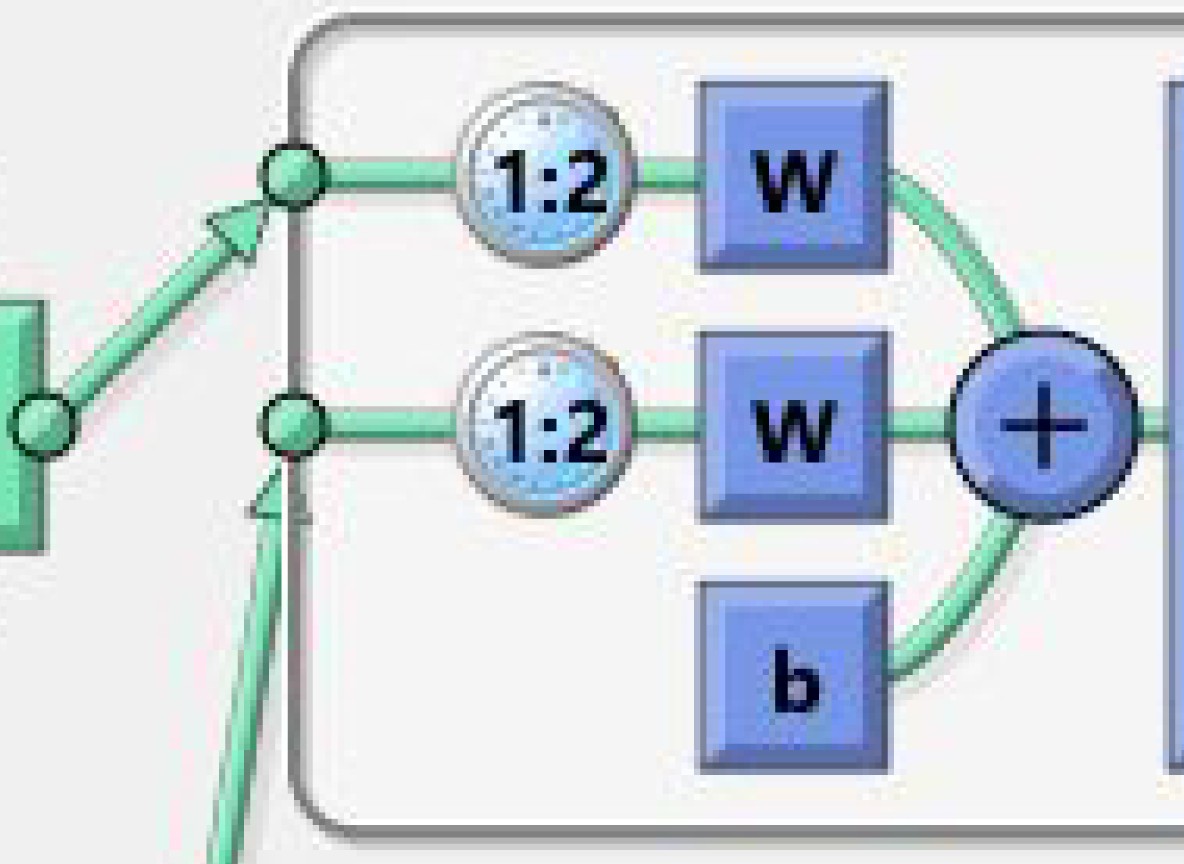
**o e;: P.'rrlllo H.f**

-20

**Time**

**Hidden**

**x(t)**



**Output**

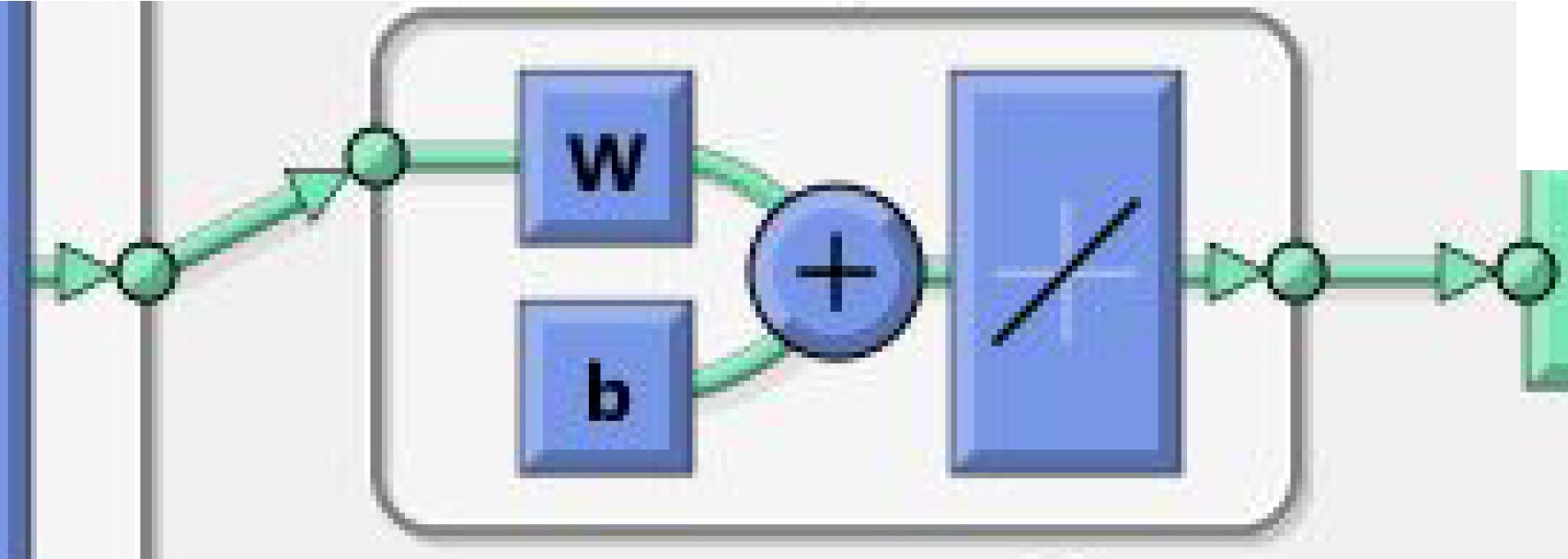
./

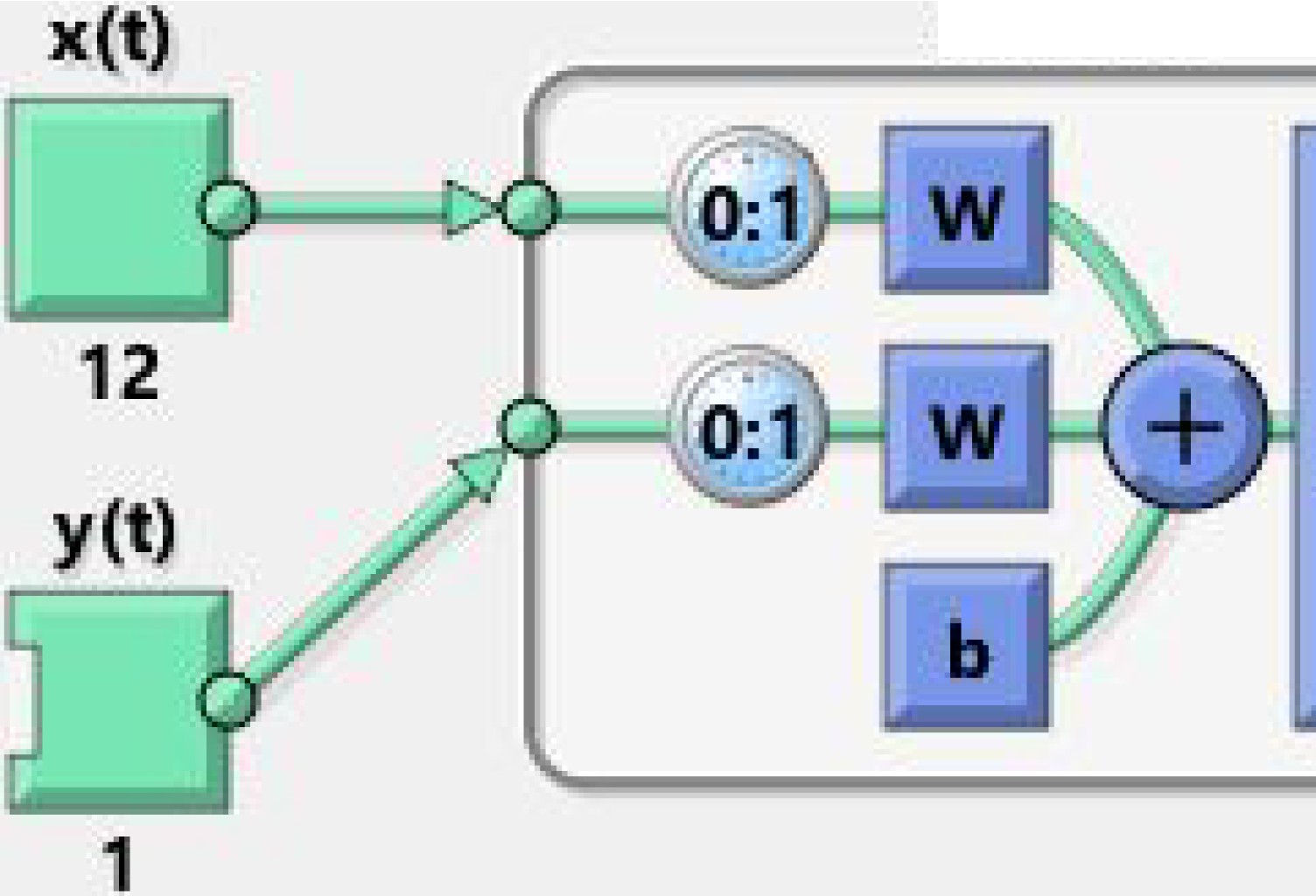
**1**

**10**

**12**

NARX !Neural !Netw ork - Pr edi ct One St ep Ahead (view) D X

**Output**



**Hidd,en**

./

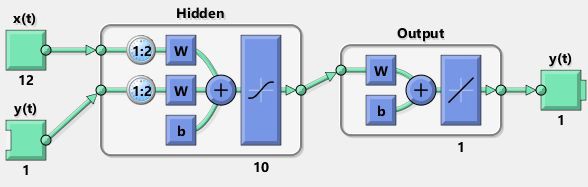
**10**

**1**

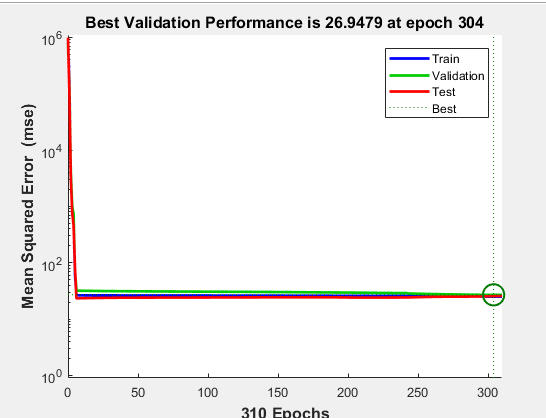
**y(t+1)**

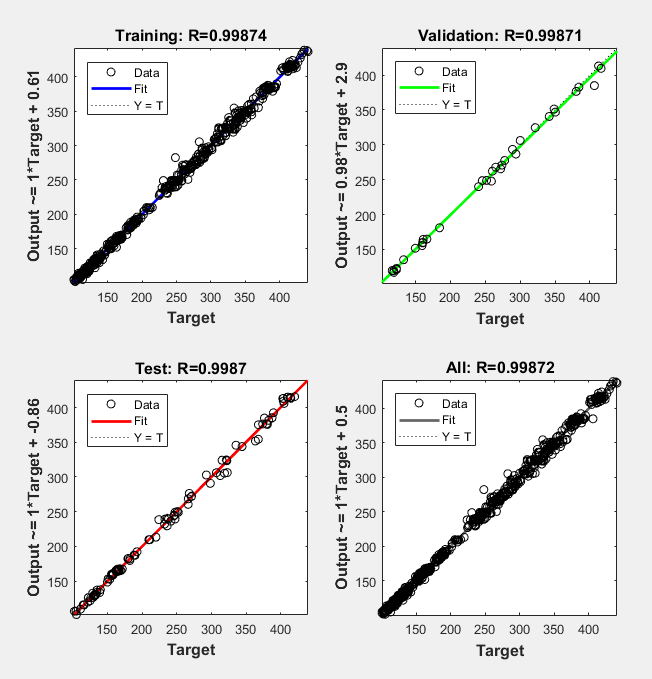


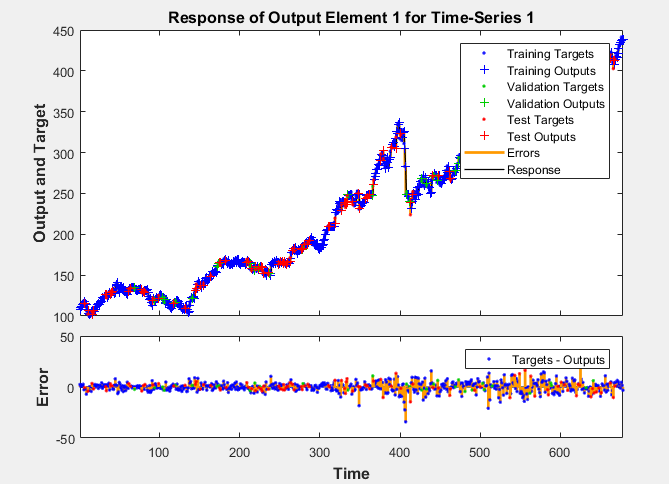
**1**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Raw Data** | **valPerfor mance** | **test performan ce** | **Close loop Performanc e** | **multiStepPerf ormance** | **stepAheadPerf ormance** |
| **NARX(Raw)** | 87.9897 | 2.70E+03 | 616.8959 | 29.56 | 435.8385 |
| **NARX(Z)** | 31.2154 | 37.578 | 5.16E+04 | 70.9248 | 25.223 |
| **NARX(PCA)** | 41.3106 | 63.615 | 3.64E+03 | 682.2212 | 26.2605 |







**Inference from Observation:**

It can be stated that no algorithm is better than another in absolute terms. Simple algorithm of linear regression was giving close to 99% accuracy when the raw data was given to it. On the same other hand when the same raw data was passed through SVM the prediction failed miserably.

Same was observed with Normalized data set, it failed miserably with linear regression but performed extremely well with SVM.

PCA matrix also failed with Linear Regression as it was returning high mean absolute error but when the same matrix was passed through SVM it gave close to 98% accuracy, the linear kernel of SVM reported 98.76% accuracy even when the PCA was applied 11 times. Feature Reduced Data set gave extremely low mean absolute error although as much as

10 features were removed which had correlation less the 50% but the same matrix performed horribly with SVM.

The current knowledge about Neural Networks is not enough and we cannot infer anything from the same. However, the Neural Networks were implemented.

**Predicted Values:**

The following are the predicted values of S&P close of next 7 weeks

**435.19**

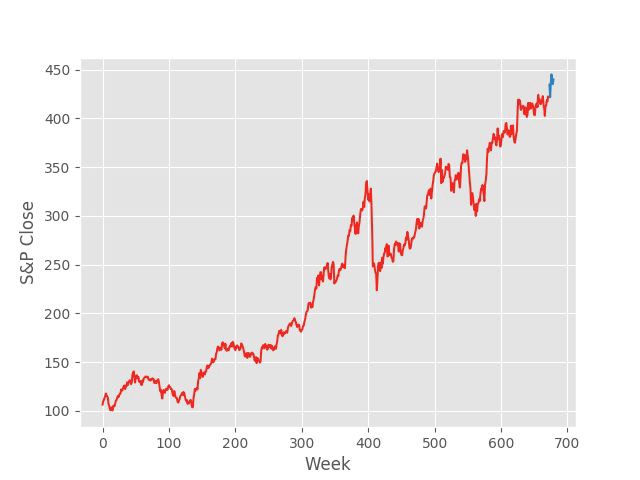
**424.39**

**432.75**

**447.23**

**441.35**

**437.38**

**440.86**

Conclusion

There are many things to conclude from this dataset and the algorithms applied by us.

Initially, the dataset was studied to acquire the necessary domain knowledge. After acquiring the domain knowledge, the dataset was optimized by the help of multiple ways (namely Normalization, Feature Reduction, Principal Component Analysis and matrix provided by Ashish Sir)

Further, Linear Regression (Cost Function and mvregress), Support Vector Machines and Neural Network was implemented on the dataset.

Different values of Accuracy for different