SHIV NADAR UNIVERSITY

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

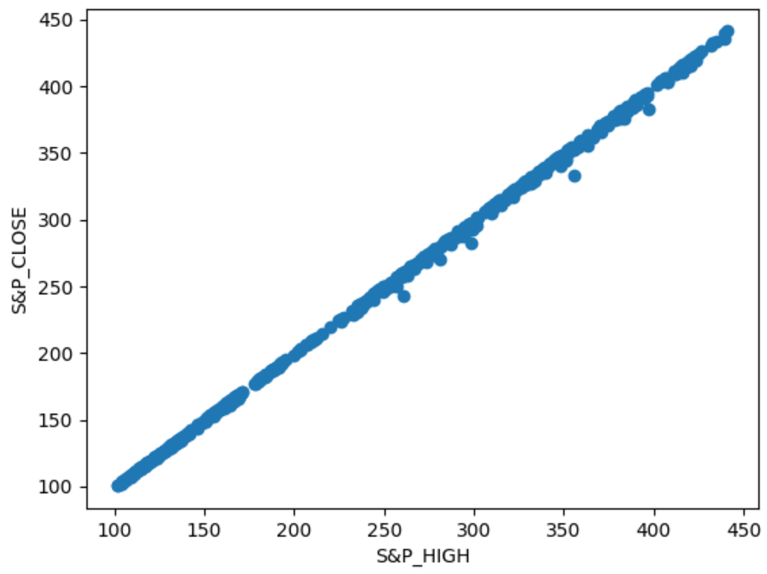
'Stock Price Forecasting' is an interesting field of study and a money oriented practical application of Machine Learning. A lot of research work is being carried out to predict the stock prices by financial institutes and is the topic of the hour. This trending topic intrigued us and we found the dataset 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).

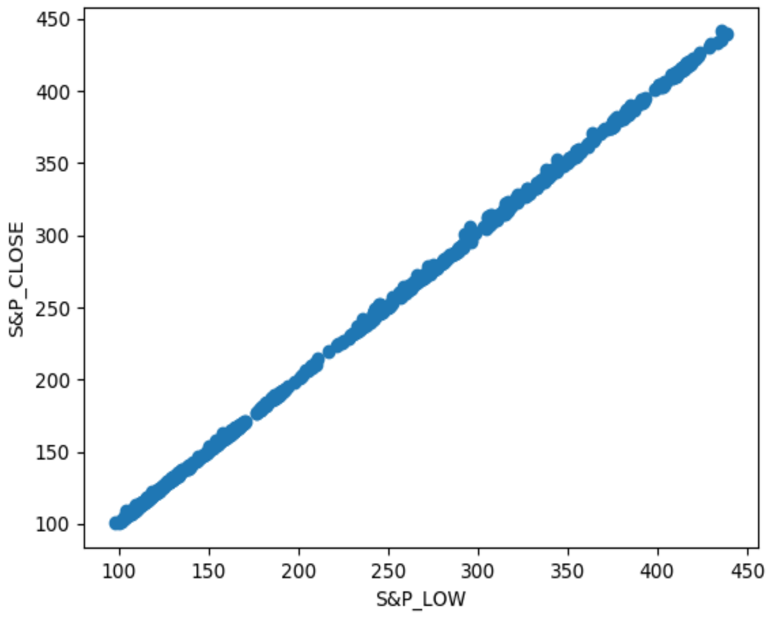
1. **S&P_HIGH**

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.



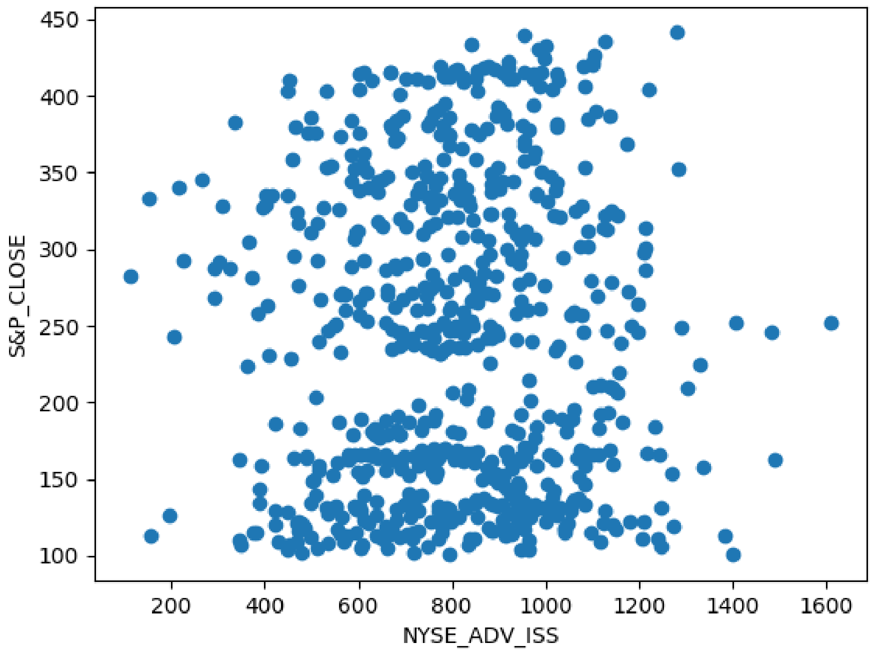
2. **S&P_LOW**

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.



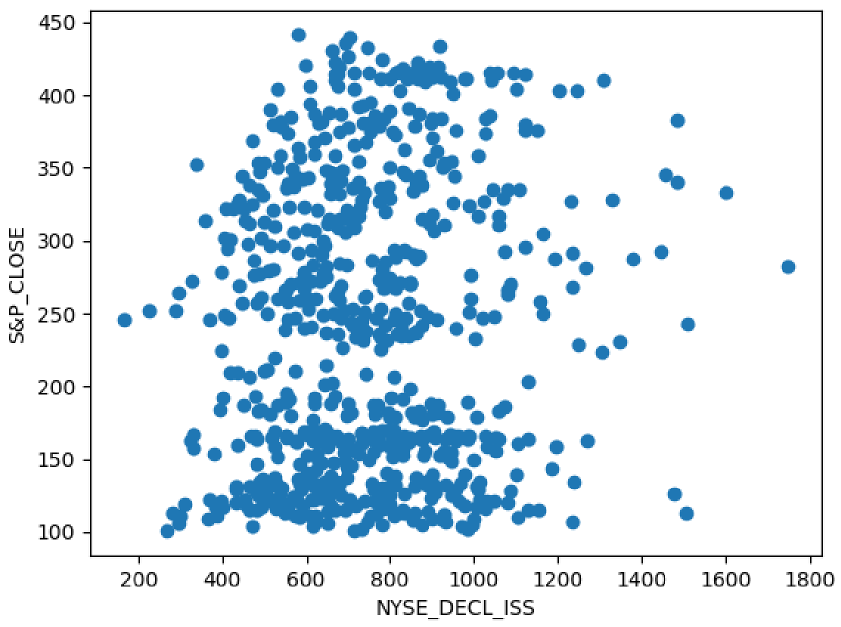
3. **NYSE_ADV_ISS**

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.



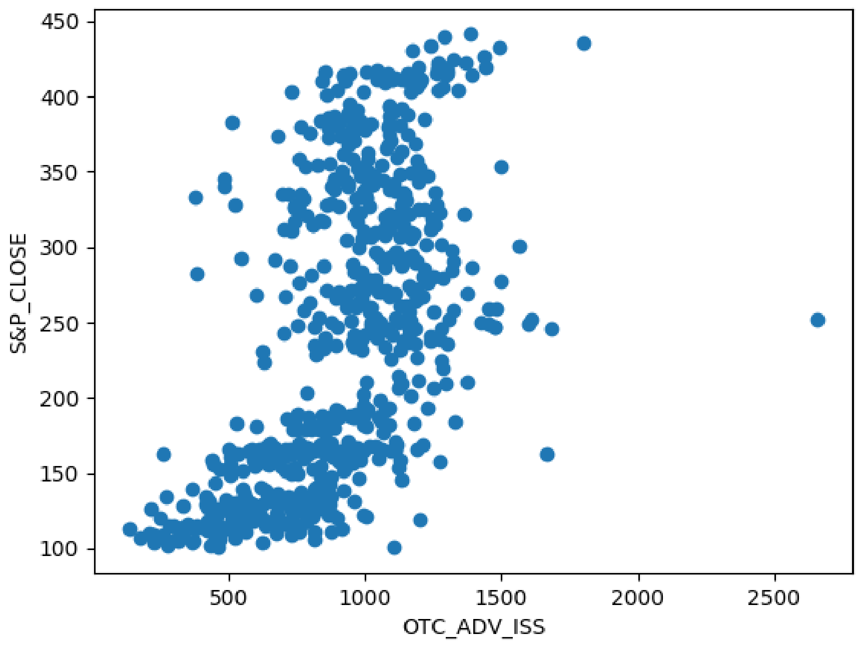
4. **NYSE_DECL_ISS**

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.



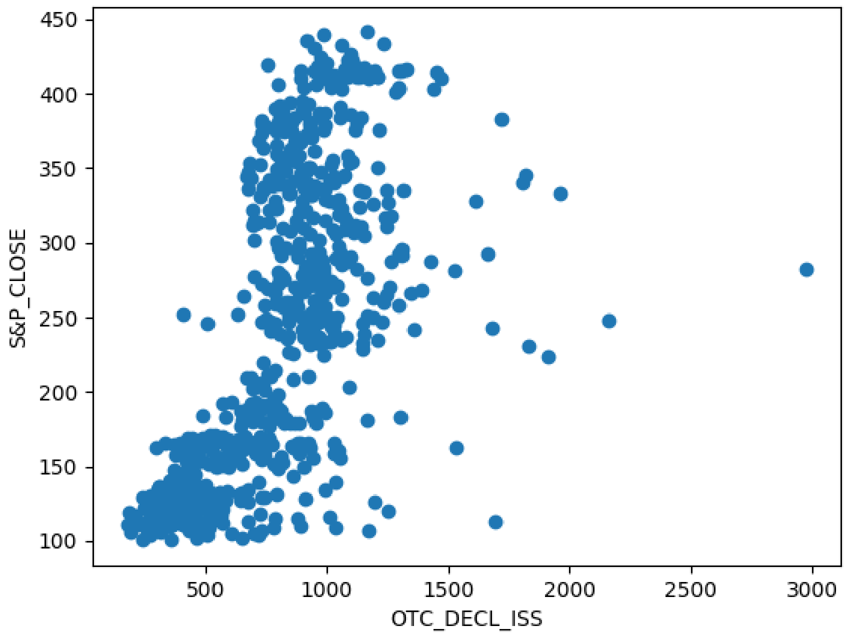
5. **OTC_ADV_ISS**

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



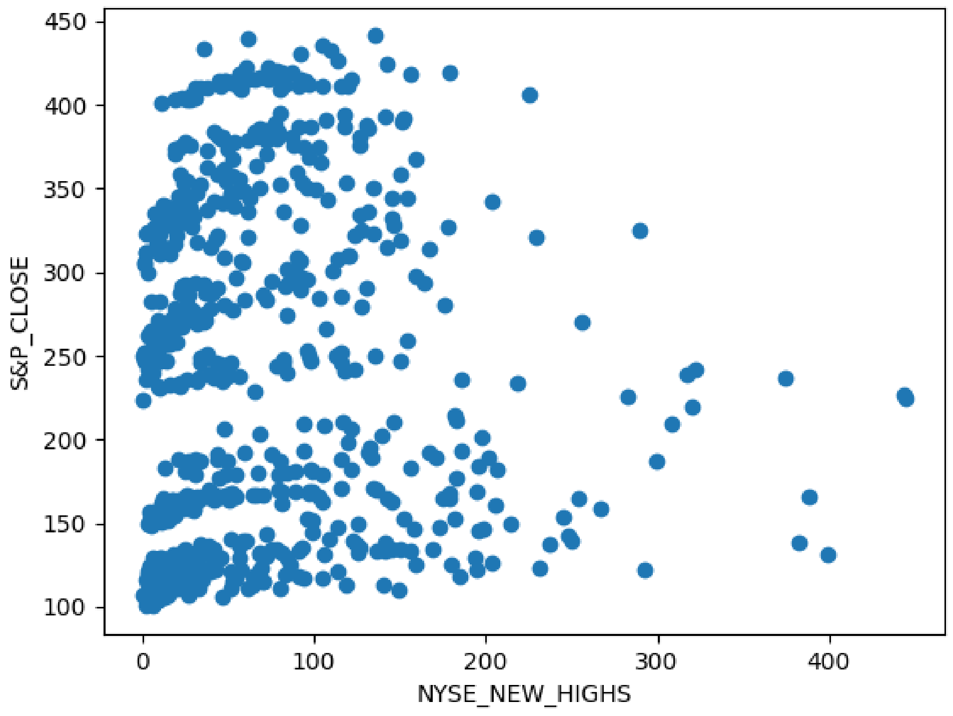
6. **OTC_DECL_ISS**

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



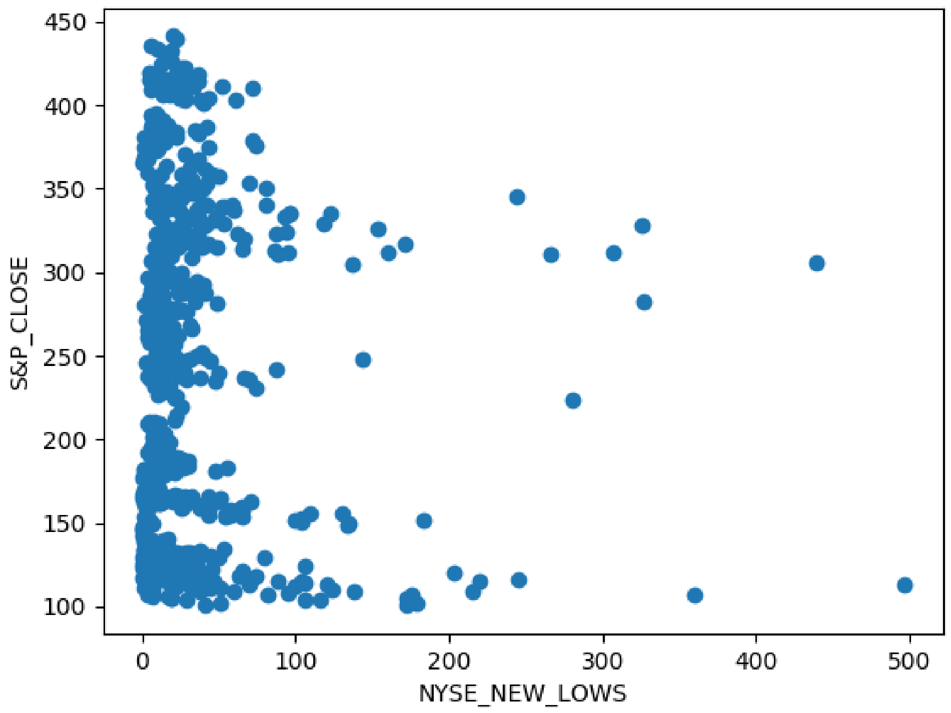
7. **NYSE_NEW_HIGHS**

Parameter tells number of best performing shares.



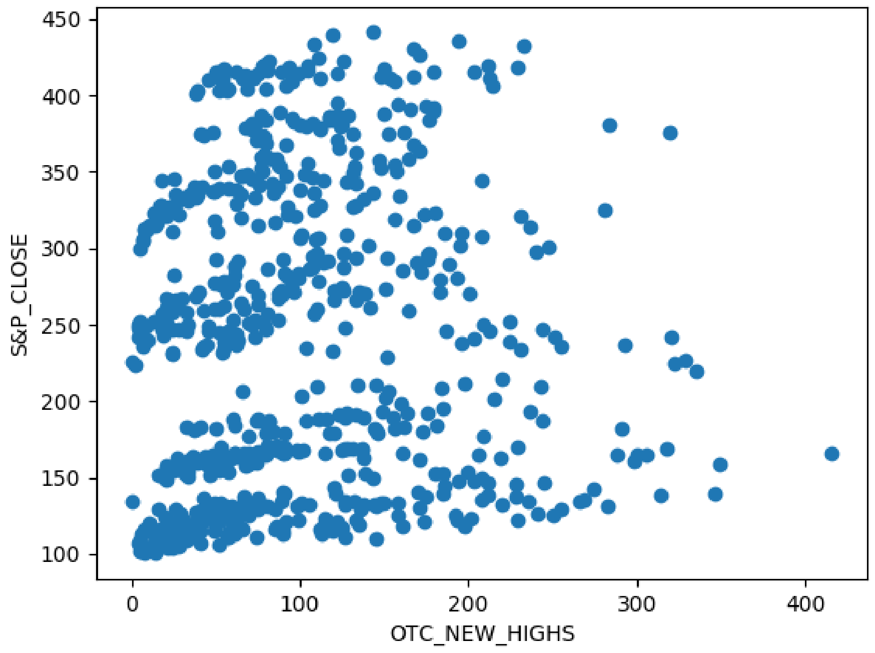
8. **NYSE_NEW_LOWS**

Parameter tells number of worst performing shares.



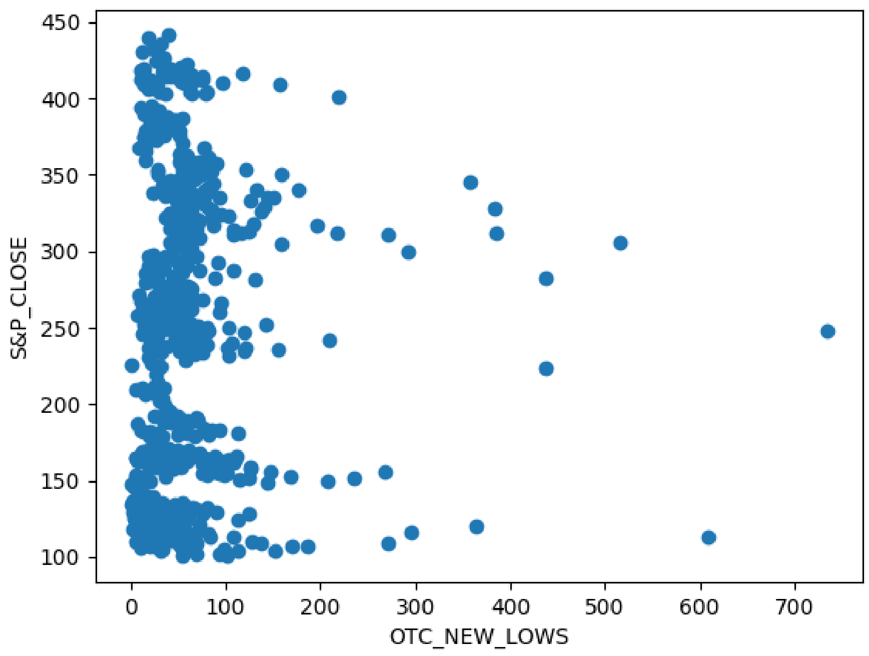
9. **OTC_NEW_HIGHS**

Parameter tells number of penny stocks which are best performing.



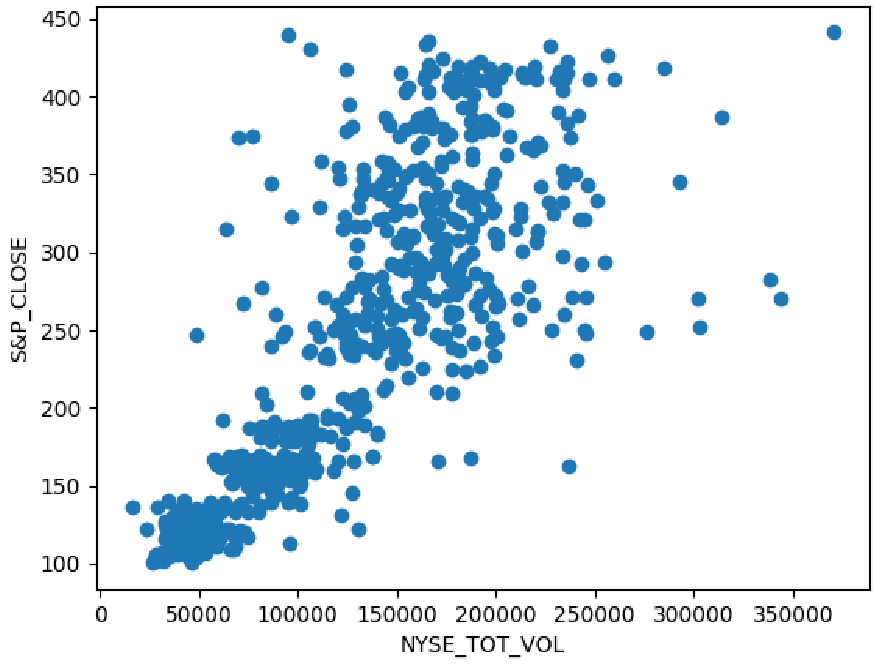
10. **OTC_NEW_LOWS**

Parameter tells number of penny stocks which are worst performing.



11. **NYSE_TOT_VOL**

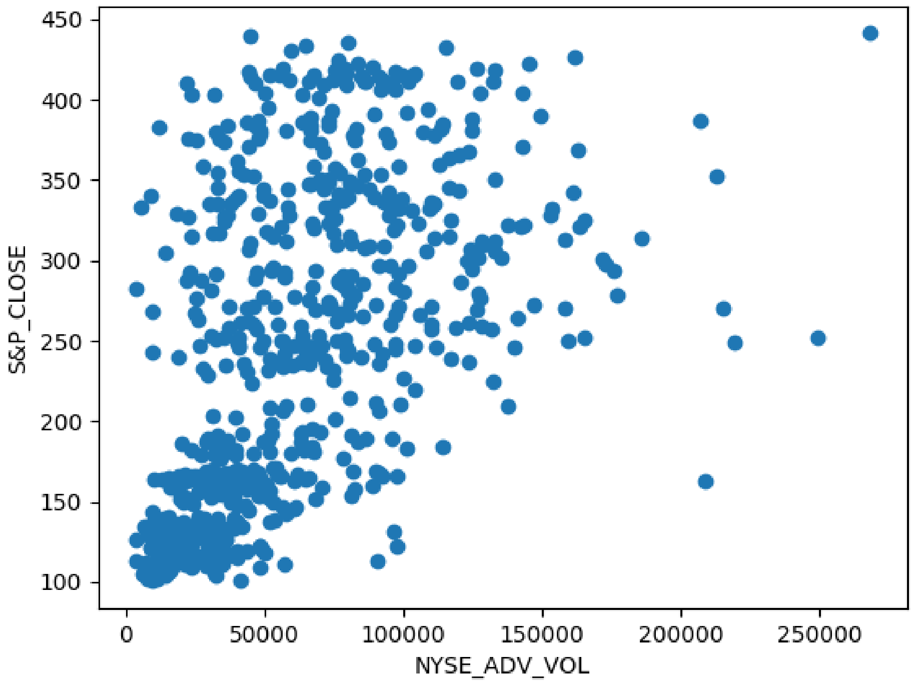
Total number of shares traded in the exchange.



12. **NYSE_ADV_VOL**

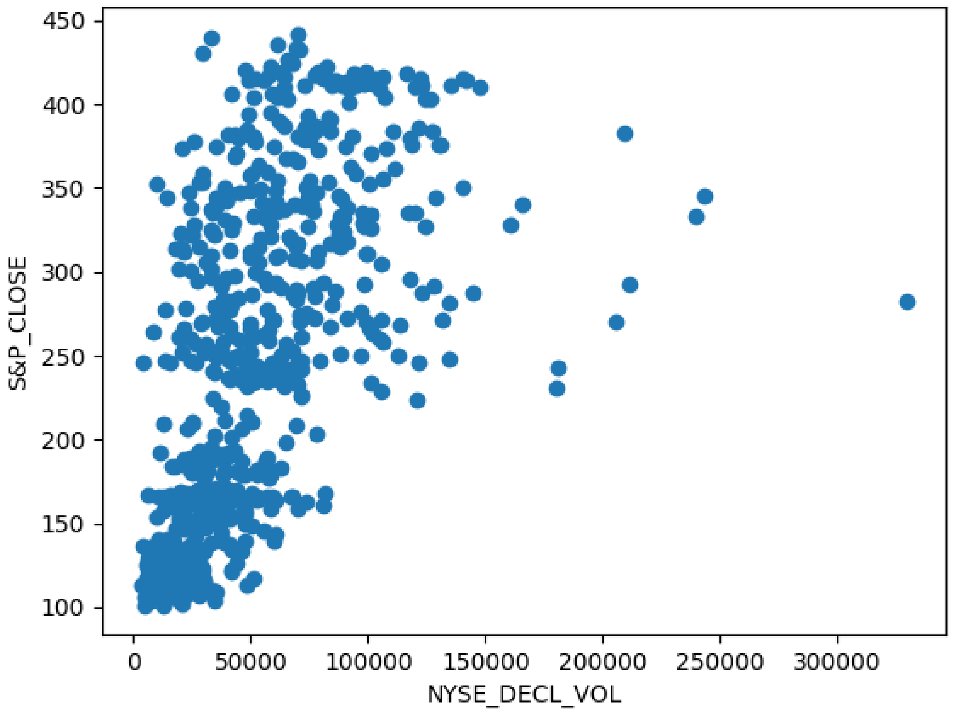
Total number of positive performing shares traded in the exchange.

13.



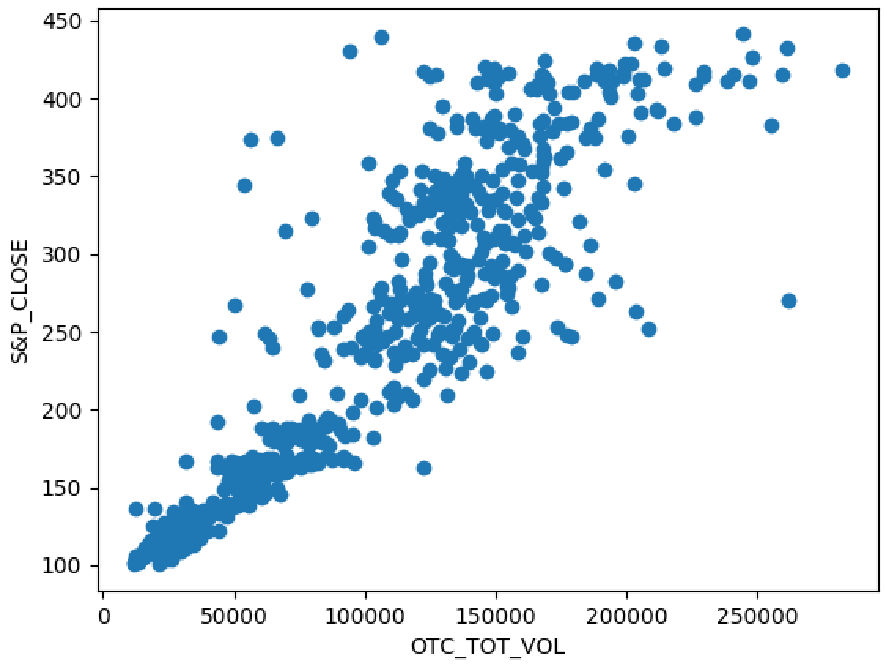
NYSE_DECL_VOL

Total number of negatively performing shares traded in the exchange.



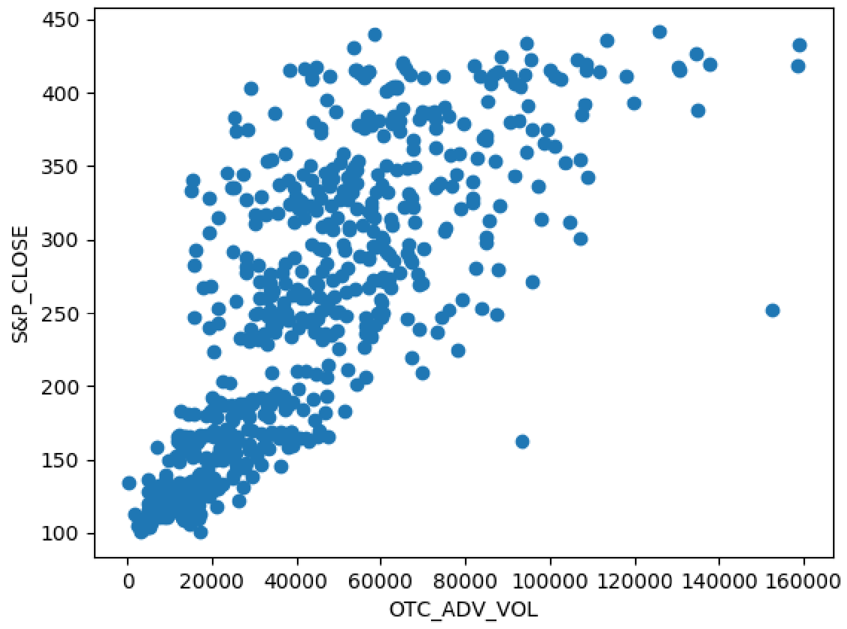
14. **OTC_TOT_VOL**

Total number of penny stock shares traded in the exchange.



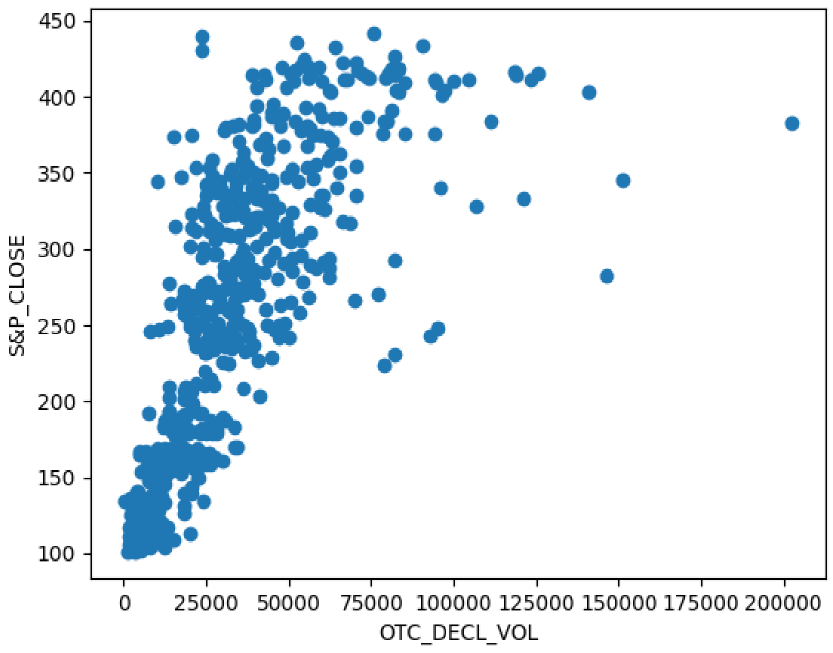
15. **OTC_ADV_VOL**

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



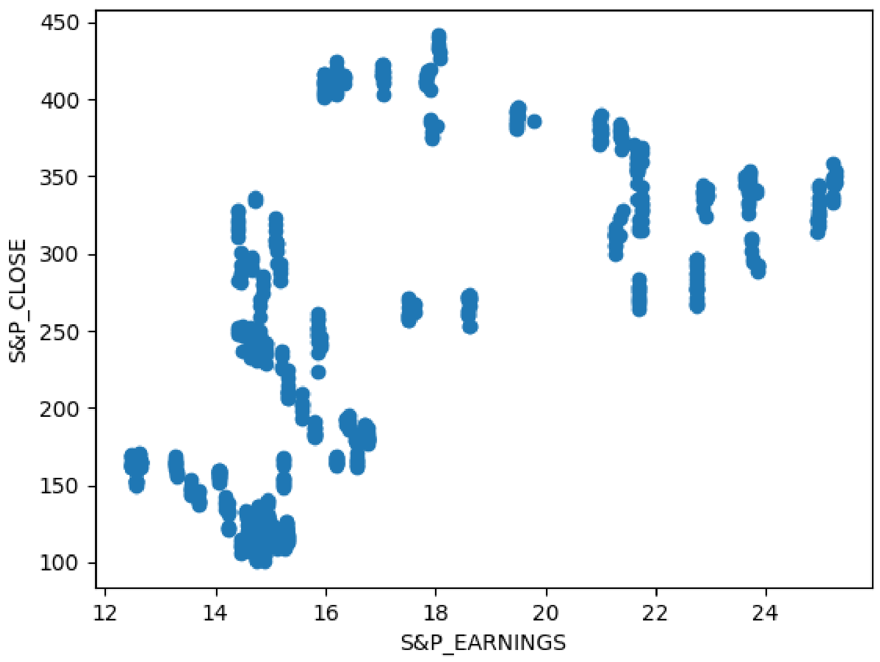
16. **OTC_DECL_VOL**

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



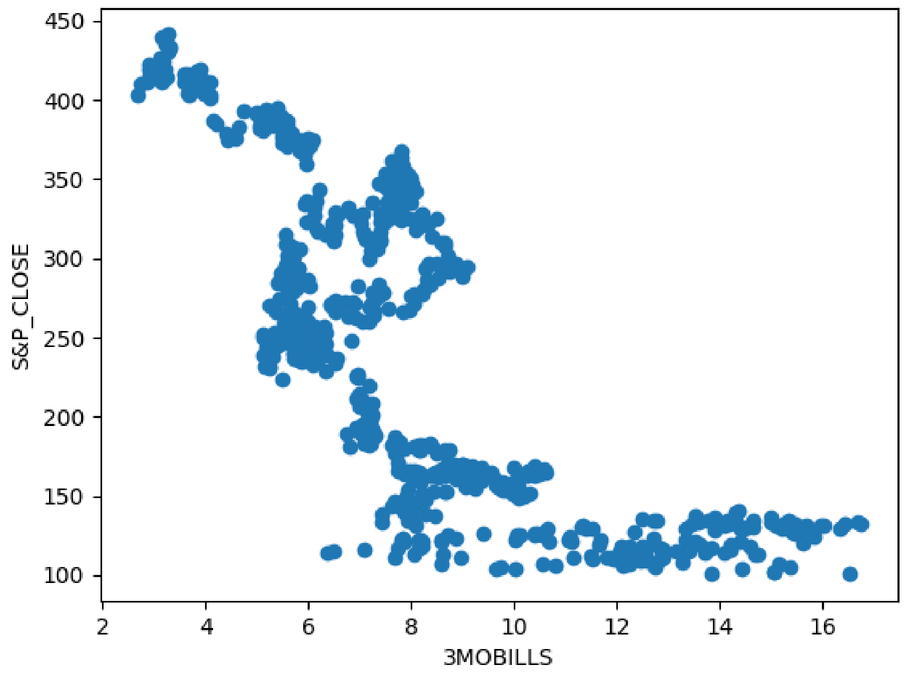
17. **S&P EARNINGS**

This feature hasn't been understood yet.



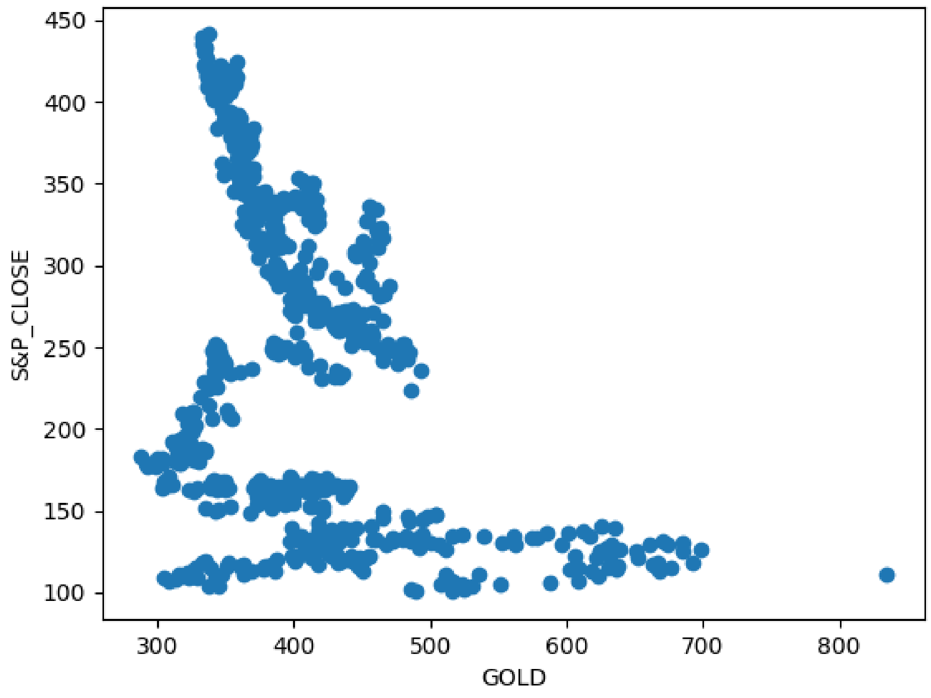
18. **3 MOBILLS**

This feature hasn't been understood yet.



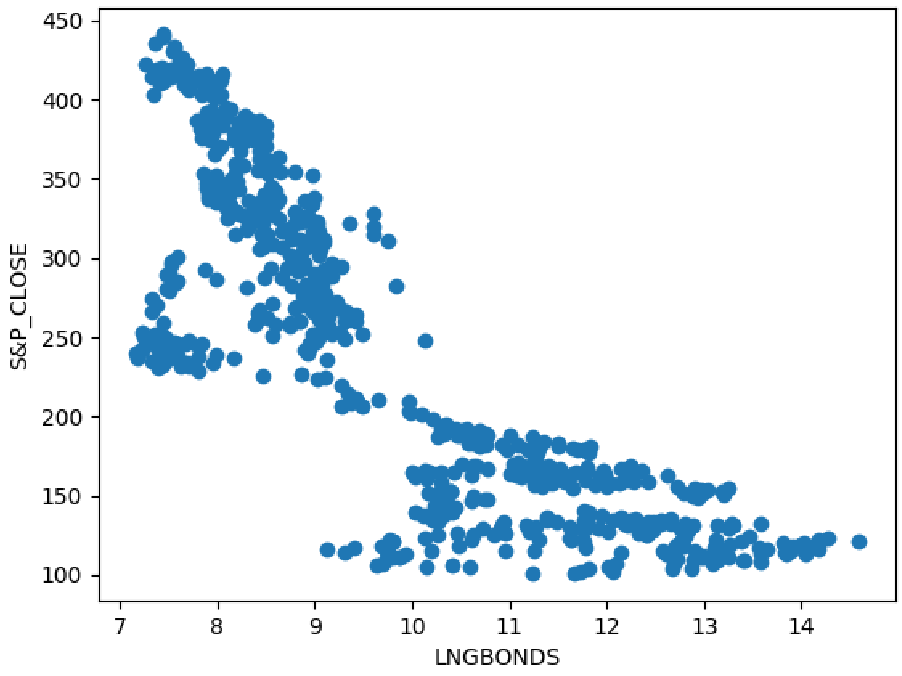
19. **LONGBONDS**

This feature hasn't been understood yet.

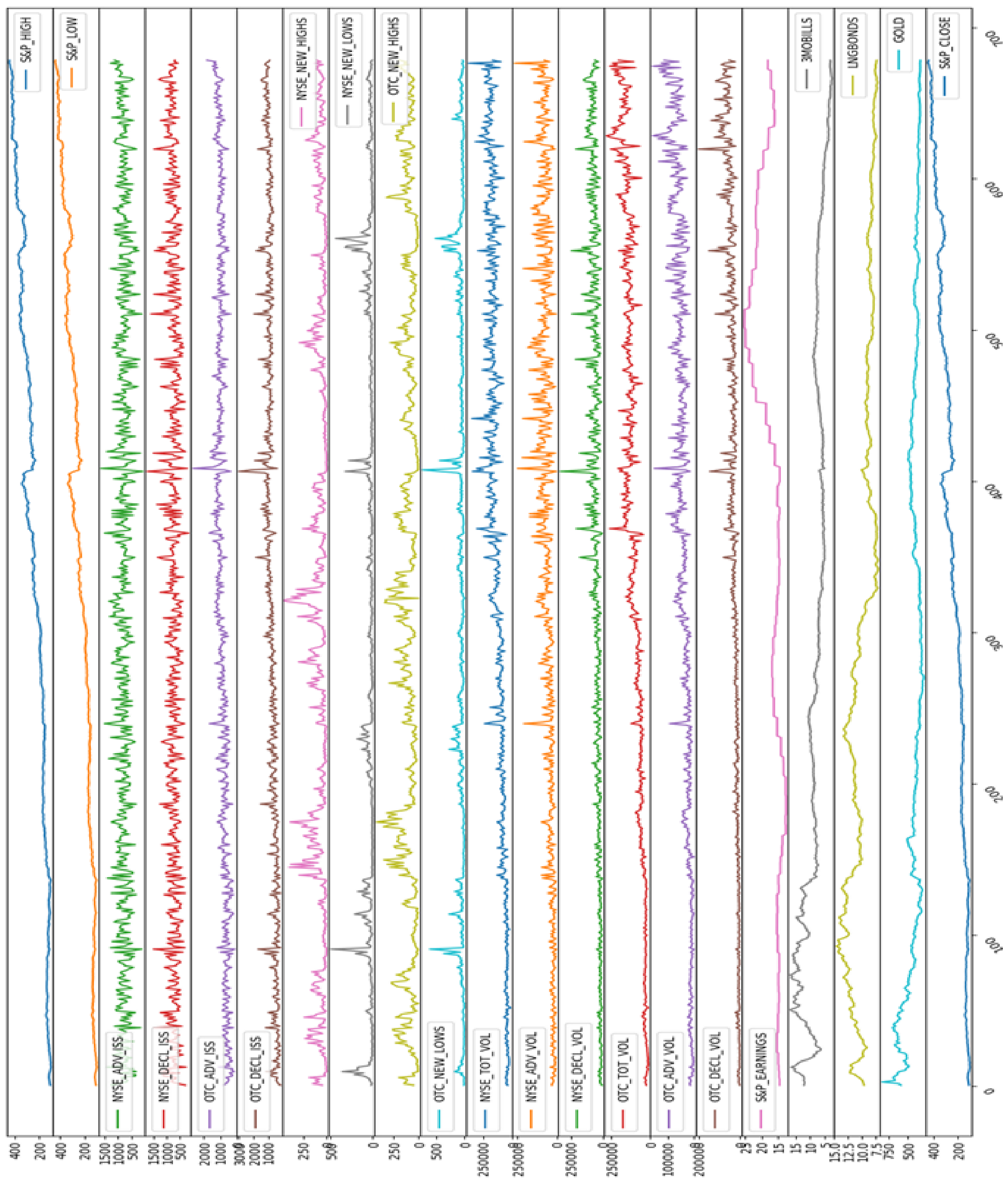


20. **GOLD**

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)

$Z = [(x-\mu)/\sigma]$

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.

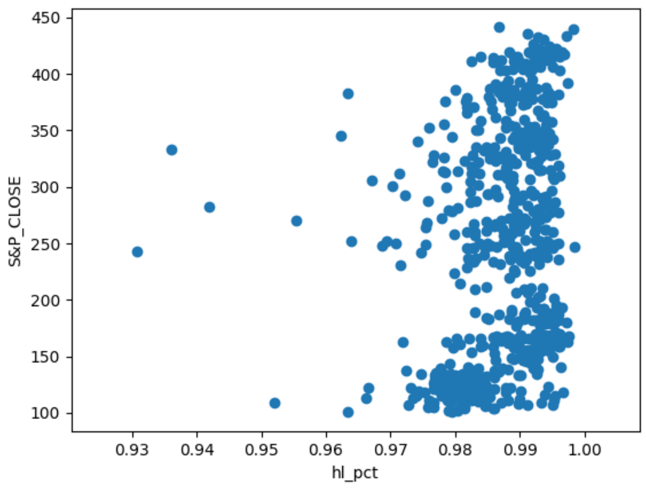
Merging

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**).

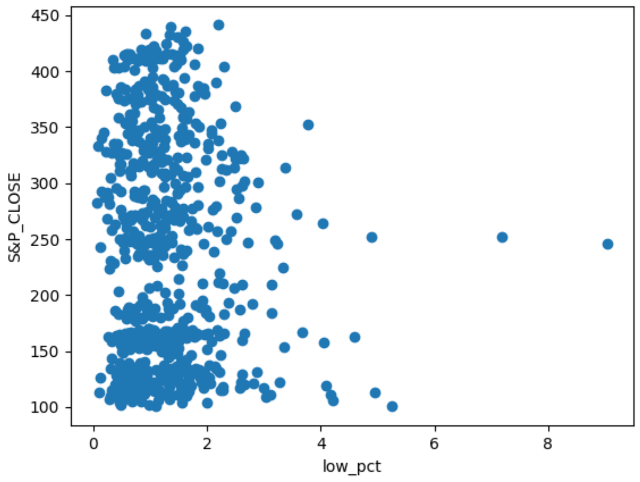


As we can see from the graph, correlation between **hl_pct** (**S&P_LOW÷ S&P_HIGH**) is approximately 0.20 which has actually ruined the both the features and incorporating this reduces our accuracy significantly.

(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

Z-Score 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)/σ).

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

Conclusion

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

No Domain Knowledge

Consider the situation where we do not have the idea of the features and the domain knowledge of the dataset. Different values of different features are provided to us but with no knowledge about the subject of the dataset.

In that regard, we find that **PCA** acts as a great tool in reducing the dimensionality in those datasets.

With Domain Knowledge (Based on this project)

Even after having the domain knowledge, feature reduction based on taking ratio of the values was not that beneficial.

But the domain knowledge played a significant role in removing the features while in feature reduction.

Currently, we are talking only in the terms of data exploration, Feature Scaling and Feature Reduction. In further reports the main focus would be on achieving greater accuracy by applying different algorithms on our dataset.

	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_NEW	NYSE_NEW	OTC_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 implemented 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;
    % Find StdDev
    feature_std = std(X_norm(:,feature_index));
    % (datatpoint - mean)/(stdDev)
    X_norm(:,feature_index) = X_norm(:,feature_index) / feature_std;

    sigma(feature_index) = feature_std;
    mu(feature_index) = feature_mean;
end

% 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;

    sigma(feature_index) = feature_std;
    mu(feature_index) = feature_mean;
end

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;
```

```
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);

%---------------------------------------------------------------------------
```

```
actualAccuracy =

   82.9055
```

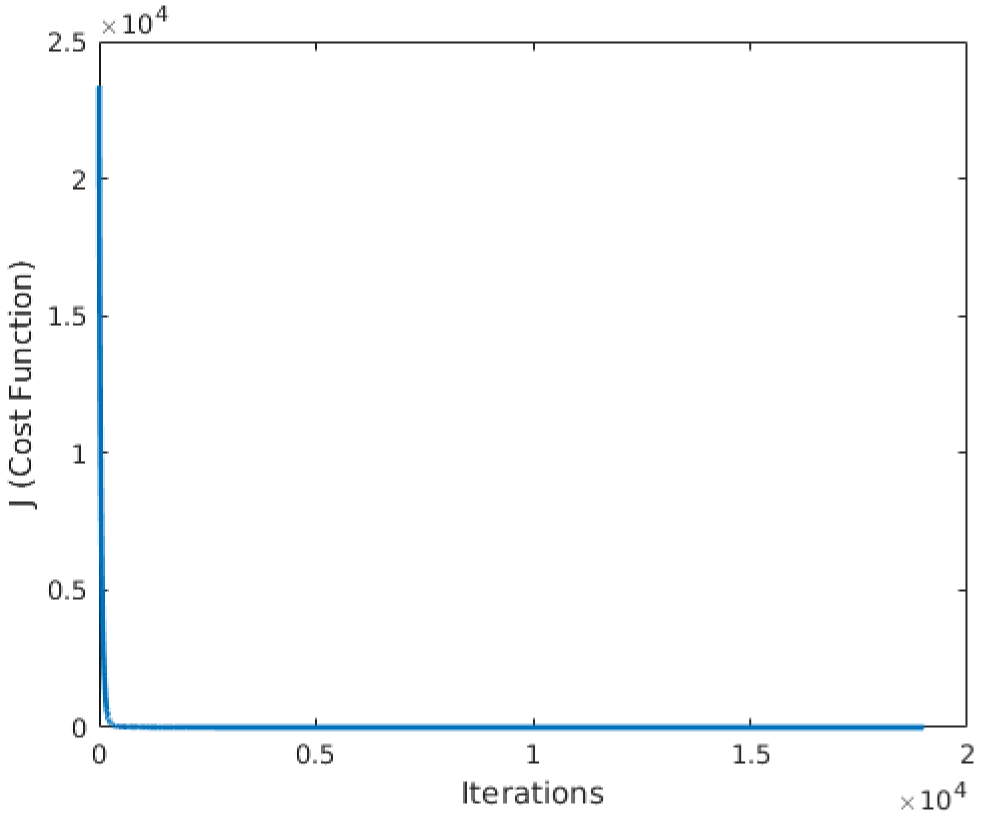
- Plot the results

```
%-----------------------------------------------------------------------------

% 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

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

```
%--------------------------------------------------------------------------
```

```
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;

    sigma(feature_index) = feature_std;
    mu(feature_index) = feature_mean;
end

% 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;

    sigma(feature_index) = feature_std;
    mu(feature_index) = feature_mean;
end

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;

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);
```

```
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