Machine Learning Project

for InfoTrie, Singapore

VISA Inc.

Stock Symbol (V) - NYSE

By Hayden Brown, 31 July 2016

Prelude:

I enjoyed doing this project and learnt a lot from it. During the project I focused on being efficient in the fastest time possible. However I do feel that this project does not reflect my style of Algorithmic Trading. Traditionally in the past, I have had two layers (layers as in different timeframes) for guidance of the trading strategy. One layer on the 'Day' timeframe and the other on the '4 Hour' timeframe. I could use one machine learning algorithm to cover both layers or I could use two different algorithms on each layer. To finalize the strategy, I then use technical indicators on very small timeframes for entry and exit of trades.

Utilities:

Research was done using R-3.3.1-win and RStudio 0.99.903 on Windows 7 OS with i7 Intel processor.

All the files below are to be saved in the **Libraries\Documents** directory of windows and in **C:\program files\R\R-3.3.1\bin**

Associated Files:

- **InfoTrie_snippets.R** – holds all the code snippets that were used to create the information in this report.
- **runinfotrie.R** – is a script that you can call in Command Prompt ('Dos') using the command
  R CMD BATCH runinfotrie.R runinfotrie.log
  First you need to place the runinfotrie.R script, runinfotrie.log file and the two CSV data files in the directory below:
  C:\program files\R\R-3.3.1\bin
  Then open the Command Prompt and change the directory address to point at the same location as the files. Next simply type in the command:
  R CMD BATCH runinfotrie.R runinfotrie.log
  And now view the '.log' file with Wordpad and read the processed output of the two '.csv' files. You should be able to see some of the information touched on throughout this report and them some more.
- **NS1-V_US.csv** – holds the dataset for the sentiment and news indications from Quandl.com
- **Yahoo-V_Visa_NYSE.csv** – holds the dataset for all the price information.
- **InfoTrie_Visa_ML.rds** – is the finalized machine learning code that can be called from within R to reanalyse new data.

Respect to the document:

This document and associated files must be kept in one complete package and under no circumstance be cut, edited or extracted from, without telling the author!

TASK:

To research for one stock and use the Day timeframe with machine learning to create forward indication of Buy/Sell signals. The input information will be sourced from Yahoo.com for OHLC prices, and Quandl.com/data/NS1 for Sentiment and News indication. The goal is to achieve >=70% accuracy.

Introduction:

Picking the stock of choice was a fast clumsy process. I essentially just used Yahoo stock screener and looked at stocks with > 1 Billion capital and average Beta. Then looked at the price graph for a nice wave pattern in an upward trend. The idea was to have a simple pattern for algorithms to learn.

If I was to do this process again, I would create a portfolio of stocks for which all will be quickly scanned for machine learning so only the best stocks showing the most predictability will be chosen.

In the end I settled on VISA Inc (V) as it had the nice price graph pattern described above. Also my thinking was Visa facilitates the global financial economy and an individual bank might collapse but Visa as financial infrastructure could possibly survive. Well maybe! . If blockcain technology is integrated in the future, I only see Visa using it as a complimentary technology. One last point for my choice was the possible advantage of collecting dividends from trading the stock, however this was not a focus point.

The Start! :

The data from the two CSV files were loaded and placed in the code as ‘dataset’ and ‘dataset2’. Then the data was pushed, pulled, chopped and punched until we had one dataset in the structure of:

Sentiment, Sentiment.High, Sentiment.Low, New.Volume, News.Buzz, OCV, Volatil, Price, Y

- OCV – (Open Price – Close Price) Volatility.
- Volatil – (High Price – Low Price) Volatility.
- Price – Adjusted Close Price; to remove splits of stock.
- Y – Prediction result for 1 Day forward; for training the algorithm.

All sections were ’numeric’ in nature except ‘Y’, it is a ‘factor’ for the algorithms to classify Buy or Sell. In the code Buy is shown by the value of ‘1’ and Sell by the value of ‘-1’. In the future this could easily be replaced with the words ‘Buy!’ and ‘Sell!’.

Lastly the dataset was split to a ratio of 80:20. A building dataset of 80% and a validation at the end, of an unseen dataset at 20%.

Data Peek (Descriptive statistics):

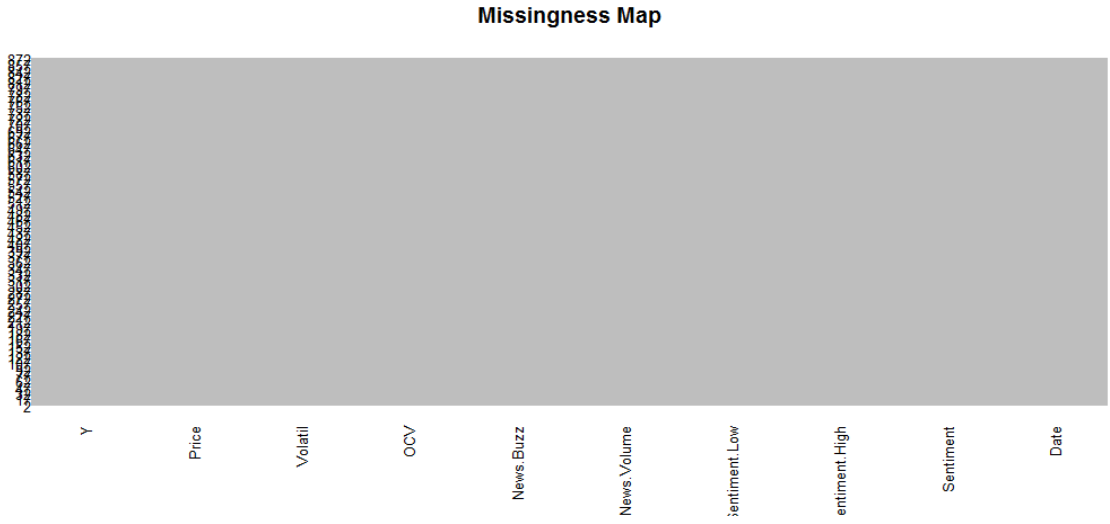
I then started my task with a peek at the data to get a feel for what I had to manage. Everything I covered was to allow me to get a feel for what was going to unfold. I covered top and bottom of data, dimensions of data, data types (as in integer, numeric, factor etc), type (class, as in Buy/Sell) distributions, data summary, standard deviation and last but not least, correlations of data. I will show the output in the 'InfoTrie_snippets' file, but in trying to keep this report short, I chose to skip to the next section. (The interesting stuff is at the end anyway!)

Class distributions is worth a mention as it is also confirmed in the next section when we visualize the data. As you can see below there are more Buy (1) signals then Sell (-1) signals. This is normal in an up trending market.

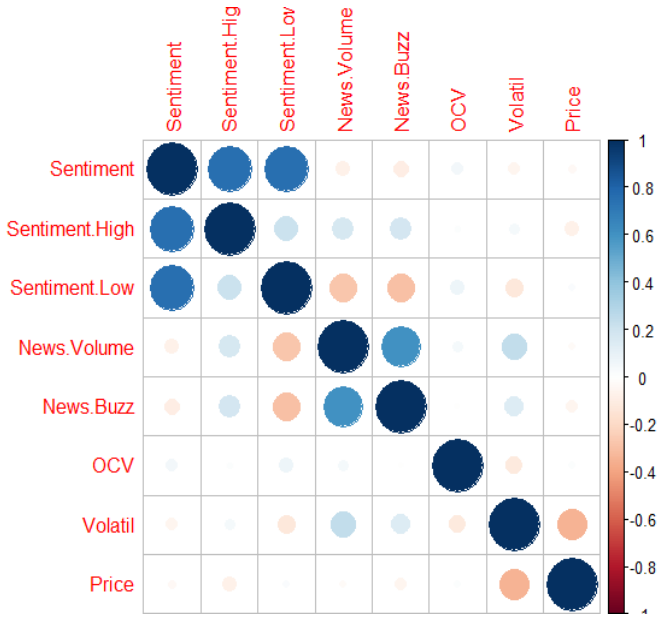
	freq	percentage
-1	328	46.9914
1	370	53.0086

Data visualizations:

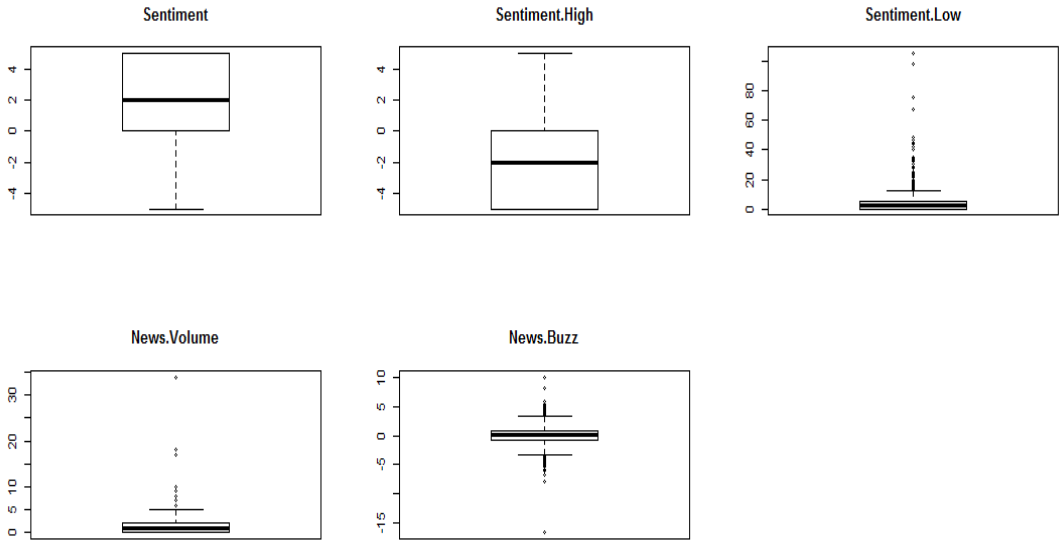
The first prudent thing to do in to look for missing data. It will only look for NA's (Not Available), not 0's that shouldn't be there or faulty data. From the graph below all data is present and there were no gaps. Gaps show up as a black line or rectangle block.



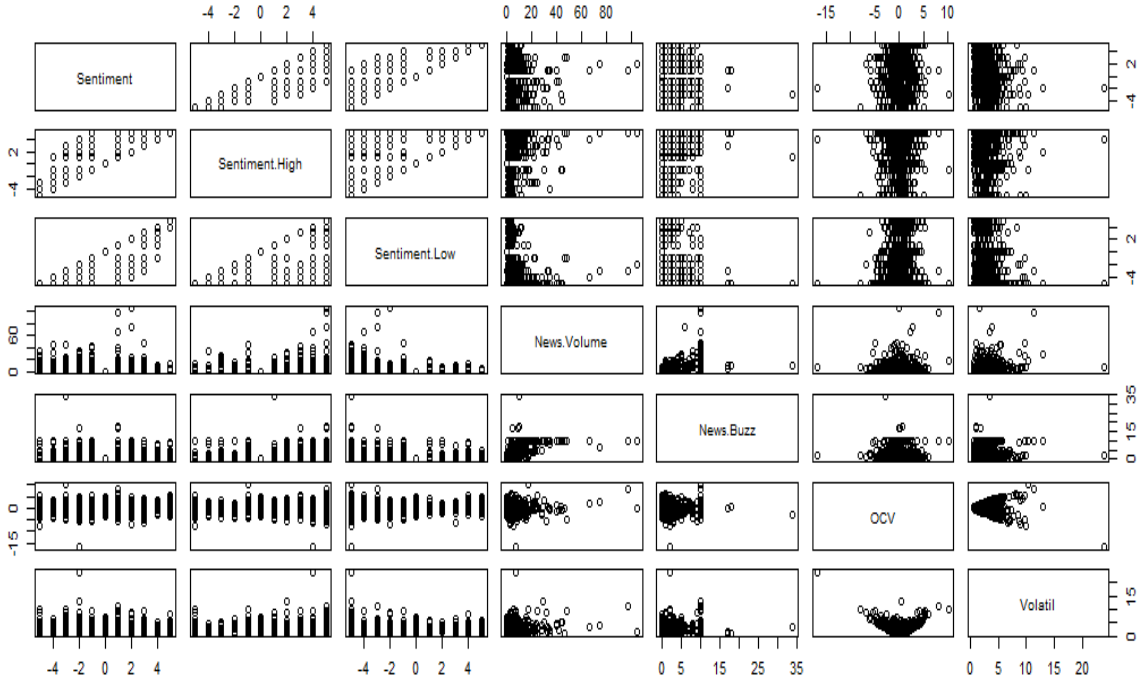
The last Statistical data I saw was correlations between the data. I thought this was interesting so a correlations plot would be my first visualization. The interesting thing in the graph below is a correlation between the Price (Adjusted Close Price) and the Volatil (High – Low Prices), even though the correlation is slightly negative. There is also a slight positive correlation between Volatil and News.Volume, a strong positive correlation between News.Buzz and News.Volume. And also a slight negative correlation between Sentiment.Low, News.Volume, News.Buzz, while the opposite slightly positive correlation between Sentiment.High, News.Volume, News.Buzz exists.



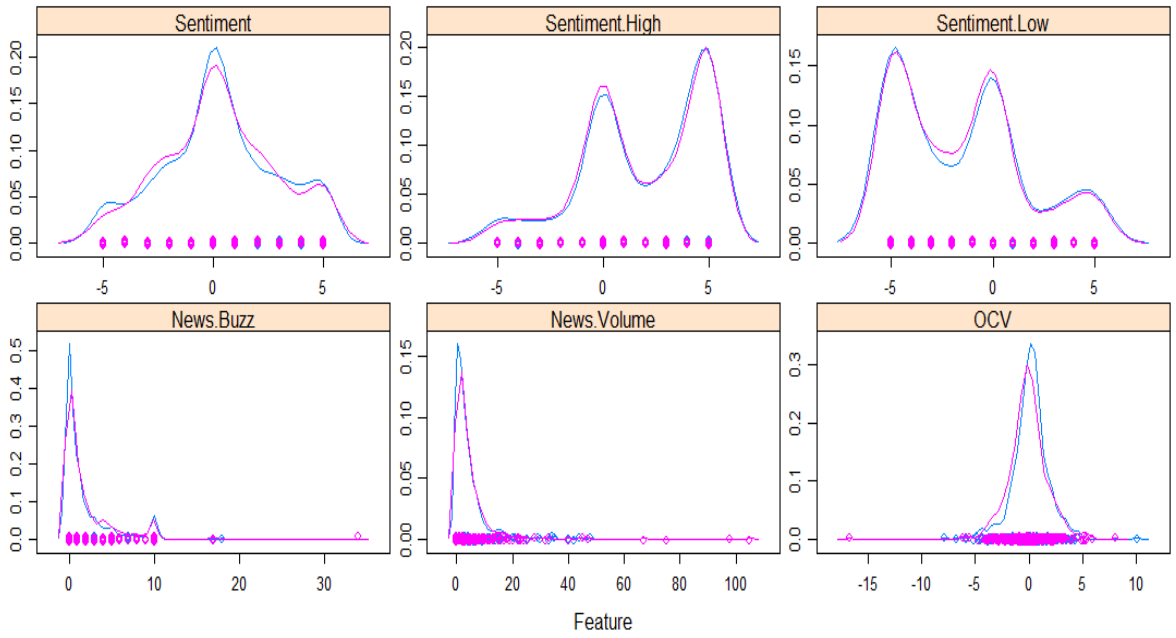
Next I did box and whisker plots on each indicator which showed some interesting views. Firstly 'Sentiment' is most positive and only briefly spikes negative as you can see on the graph below. Also 'News.Volume' has a base line near zero and spikes highly positive on occasion. Unlike 'News.Buzz' which has a base line near zero but can spike positively or negatively on occasion.



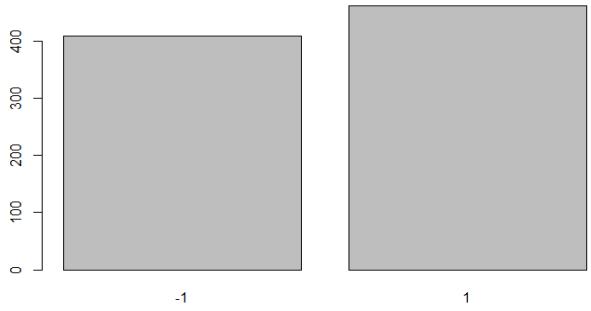
I was curious to get a feel for the nature of the data so the following graph is a scatterplot matrix of the indicators. A second scatterplot matrix separated by Buys and Sells was done, but did not reveal any new information so I left it out of this report. From the graph below you can see the nature of the indicators. The sentiment indicators give a more linear form of information (seen in the upper left of the graph), while the news indicators give more of a baseline with spikes (seen in the center of the graph), and finally the two volatility indicators (OCV, Volatil) give a more convex (Gaussian) type of nature (found in the lower right of the graph).



Density plots by type (class, as in Buy/Sell) can be useful to show where data moves. In the below graph the things worth noticing is that the 'Sentiment.High' and the 'Sentiment.Low' both have two peaks. The 'Sentiment.High' has a peak at the base (zero) and at 5 (positive territory), which shows were the information density is and makes sense that it is positive for high sentiment. The 'Sentiment.Low' has a peak at the base (zero) and at -5 (negative territory), which is also interesting as it shows the information density is opposite of 'Sentiment.High' and makes sense that it is negative for low sentiment. Finally when 'Sentiment.Low' and 'Sentiment.High' are combined you get the 'Sentiment' graph with most of the data is at the baseline (zero). Another interesting point is the 'OCV' (Open-Close Volatility) stays around the baseline (zero) and moves positively and negatively around that baseline giving a Gaussian shaped plot.



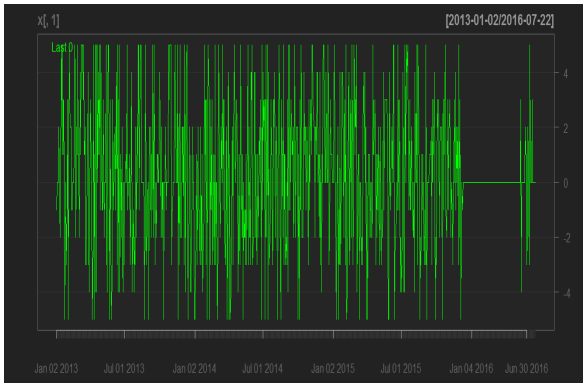
A simple comparison plot confirms what was said above, that there are more Buy (1) signals than Sell (-1) signals. But there are still a lot of signals from both to train the algorithm.

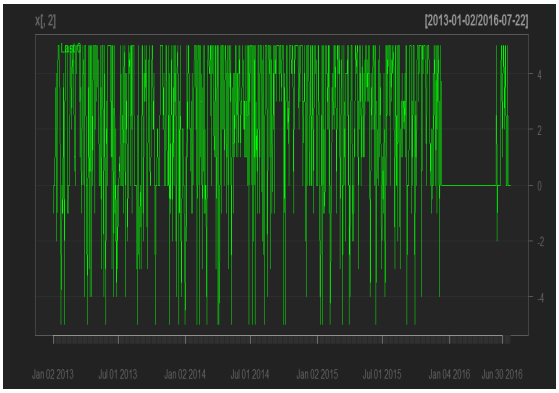


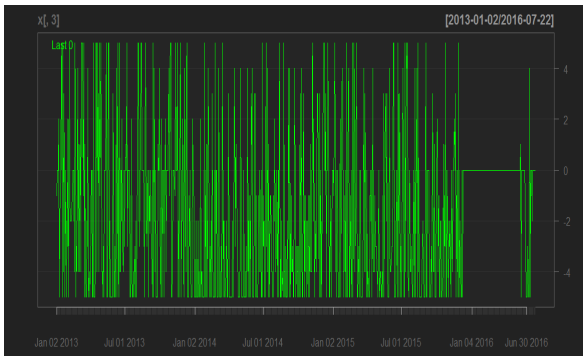
In the next step I used the quantmod package to chart the indicators data and the Price data. First chart is the price over time and we can see that the price sample for building looks good.

However when we chart the Sentiment and News indicators over time, we can see a funny pattern of flat data starting at the date of 4/12/2016 (right side of chart). This flat pattern is not found in the Volatility (OCV or Volatil) indicators so I don't think it has to do with the 20% unseen data. I would be best to cut this section of data out for building and training the algorithm, but I chose to leave it in and rush to finish the report.

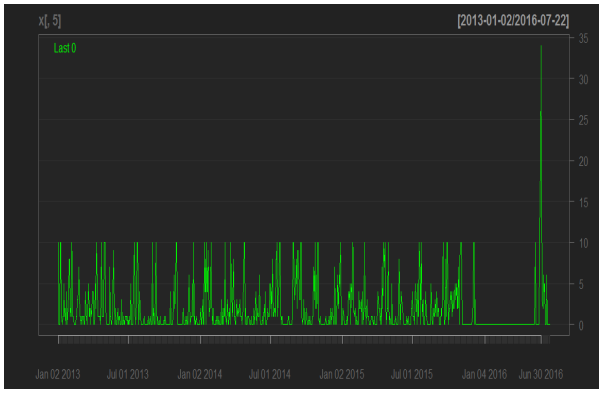
Sentiment:

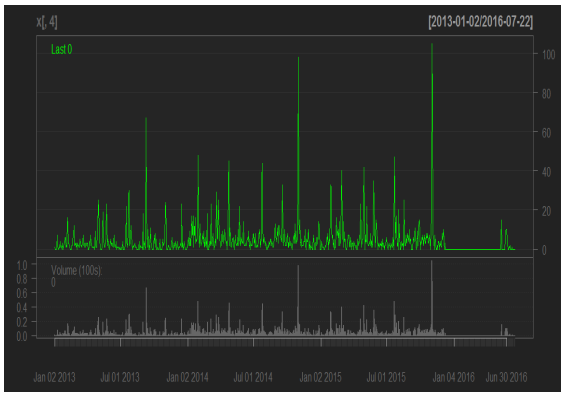




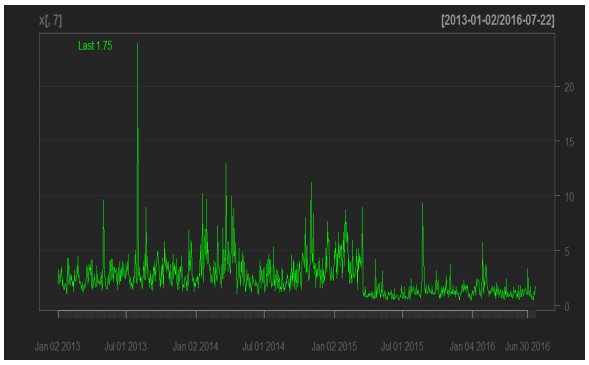


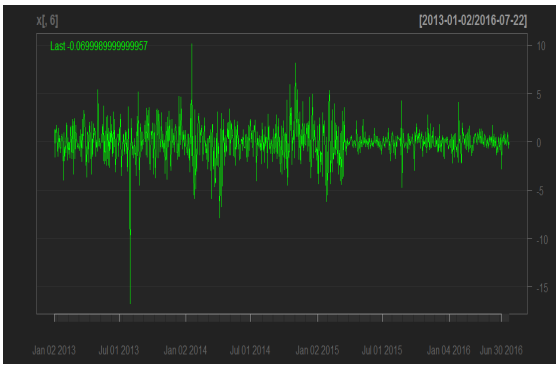
News:



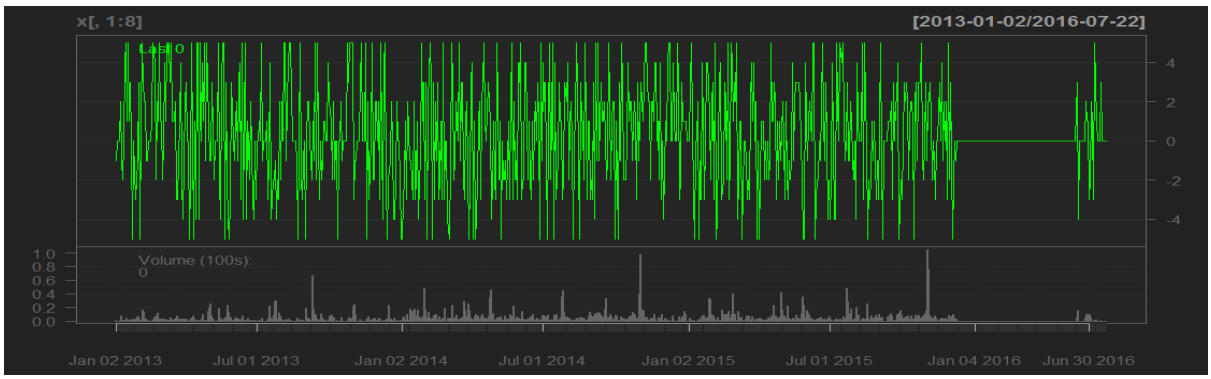


Volatility:





Its effect on the data is strong as I charted all indicators together over time:



Data Transforms:

Transforming data can also involve further cleaning of the data (like the flat spot in the data) and then re-splitting it again. I chose to continue on!. Most of the data transforms were prepared earlier, for example the Predictions (Y) were transformed to factors for binary buy/sell logic, providing more algorithm choice.

Advance transforms like "Independent Component Analysis Transform", "Principal Component Analysis Transform", "Yeo-Johnson Transform", "Box-Cox Transform" or Normalize/Standardize Data were not needed for this project.

Evaluate Algorithms:

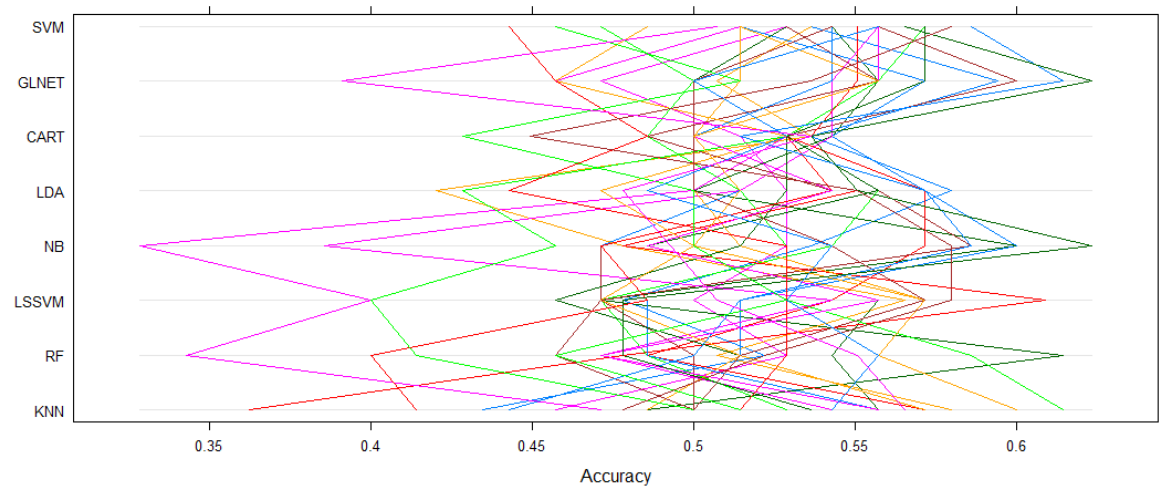
Initial testing of algorithms were performed through a CARET package test harness. A Repeated k-fold Cross Validation was chosen to be the test method as it is fast and can give a rough guide to performance.

A number of algorithms were tested and given the following short names; Classification and Regression Trees - CART, Linear Discriminant Analysis - LDA, Support Vector Machine - SVM, Least Squared Support Vector Machine - LSSVM, K Nearest Neighbour - KNN, Random Forest - RF, Regularized Regression - GLNET, Naive Bayes - NB. The results are shown below and the point of interest in the mean value when comparing algorithms, however the results can be different running the algorithm as a standalone rather than through a CARET harness.

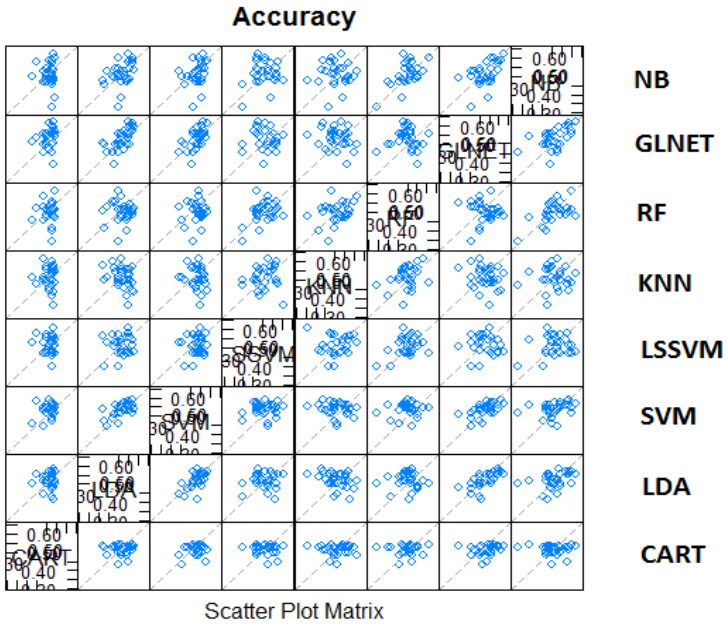
```
Accuracy
        Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
CART  0.4286  0.5000 0.5286 0.5153  0.5343 0.5429    0
LDA   0.4203  0.5000 0.5214 0.5196  0.5507 0.5797    0
SVM   0.4429  0.5143 0.5429 0.5339  0.5571 0.5857    0
LSSVM 0.4000  0.4731 0.5143 0.5120  0.5536 0.6087    0
KNN   0.3623  0.4857 0.5071 0.5124  0.5571 0.6143    0
RF    0.3429  0.4783 0.5108 0.5005  0.5269 0.6143    0
GLNET 0.3913  0.5000 0.5395 0.5301  0.5571 0.6232    0
NB    0.3286  0.4801 0.5143 0.5144  0.5429 0.6232    0
```

Now I will compare the results by visualizing the data. The first visualization will be a Parallel Plot. This graph shows each trial of each cross validation fold and how they behaved for each of the algorithms tested. I find it helpful in thinking about it, in the way it shows how different algorithms handle the data differently and how they could be combined in an ensemble prediction (e.g. stacking) at a later time. This works best if there are correlated movements in opposite directions. We can see one group of the same type of data movements with Dark Green spikes to the right with

GLNET, NB and RF. Then a second group with Red spikes to the right and left Dark Green positions with KNN, LSSVM and LDA. CART and SVM look like a third group with no clear data directions.



A Scatter Plot can be a good second view on the results as it help to see if algorithms are correlated or not. Each dot is a fold trial of data and each algorithm pair to the same dot. All the dots in a square of an algorithm pair, are all the fold trails of data. Correlation shows itself as a strong concentration of dots in the one spot or along the linear line in the square. Squares the show scattered dots are weakly correlated. From the graph there seems to be good scattering with LSSVM paired with NB, GLNET, RF and KNN. A second observation of good scattering is KNN paired with NB, GLNET and RF.



The following piece of data is not a plot but statistical table known as 'Statistical Significance'. The lower diagonal of the table shows p-values for the null hypothesis (if distributions are the same), basically the smaller is better. The upper diagonal of the table shows the estimated difference between the distributions. From the data table below we can note that KNN, LDA and CART show low values against SVM in the lower diagonal. In the upper diagonal it is worth noting that RF and GLNET shown higher difference paired with most other algorithms.

```
Accuracy
        CART   LDA         SVM         LSSVM       KNN         RF          GLNET       NB
CART            -0.0042857 -0.0186404  0.0032919   0.0029400   0.0147964  -0.0147964   0.0009248
LDA     1.0000             -0.0143547  0.0075776   0.0072257   0.0190821  -0.0105107   0.0052105
SVM     0.3379  0.8582                 0.0219324   0.0215804   0.0334369   0.0038440   0.0195652
LSSVM   1.0000  1.0000      1.0000                 -0.0003520   0.0115045  -0.0180883  -0.0023671
KNN     1.0000  1.0000      1.0000      1.0000                 0.0118565  -0.0177364  -0.0020152
RF      1.0000  1.0000      0.1798      1.0000      1.0000                 -0.0295928  -0.0138716
GLNET   1.0000  1.0000      1.0000      1.0000      1.0000      1.0000                 0.0157212
NB      1.0000  1.0000      1.0000      1.0000      1.0000      1.0000      1.0000
```

Other plots were done on the results but in keeping the report short I will leave them out. The code snippets can be found in the 'InfoTrie_snippets' file.

Compare Algorithms:

In this section of the report I took the time to run each algorithm as a standalone to see results outside of the CARET package of R. The two best results are below.

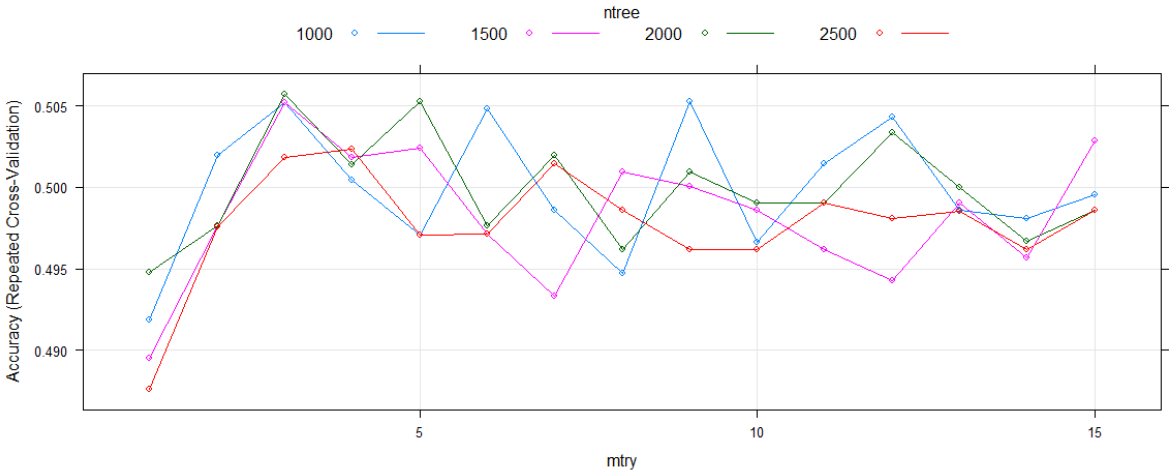
Random Forest shows 99.57% (0.9957 accuracy x 100), but I suspect overfitting;

```
               Accuracy : 0.9957
                 95% CI : (0.9875, 0.9991)
    No Information Rate : 0.5301
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9914
 Mcnemar's Test P-Value : 0.2482
```

K-Nearest Neighbour shows 74.21%;

```
               Accuracy : 0.7421
                 95% CI : (0.708, 0.7742)
    No Information Rate : 0.5301
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.482
 Mcnemar's Test P-Value : 0.8231
```

Since Random Forest has two variables, mtry and ntree, I quickly scanned a range to find the most optimized values which the results are show below. Best is mtry = 3 and ntree = 2000 which is the highest peak in the graph.



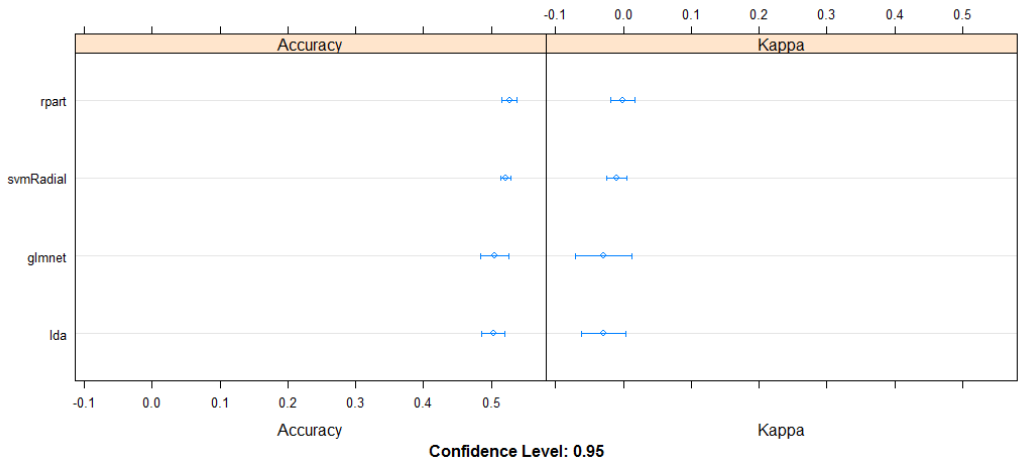
Ensemble (stacking algorithms):

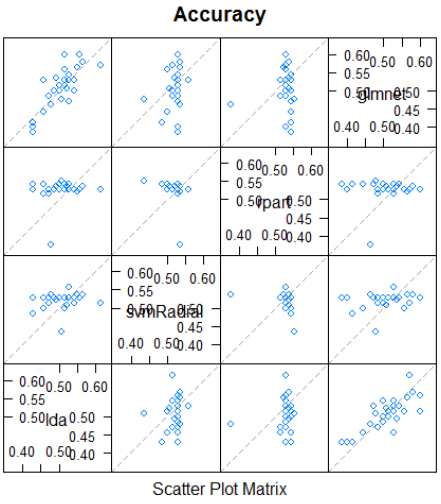
I decided to try to stack algorithms in the CARET package to try and do better than the Optimized Random Forest. I don't think it will work, but I took the best CARET results from before to see.

First are the results from the selection:

```
Accuracy
            Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
lda       0.4286  0.4731 0.5108 0.5029  0.5286 0.6143    0
svmRadial 0.4348  0.5161 0.5286 0.5215  0.5286 0.5571    0
rpart     0.3768  0.5286 0.5286 0.5272  0.5412 0.5507    0
glmnet    0.3857  0.4801 0.5108 0.5048  0.5343 0.6000    0
```

Which I plotted the result for quick visualization:





Now I place a Random Forest algorithm on top of the other algorithms to improve the results and I was able to achieve a 52.5%. Not really better than Random Forest standalone.

```
 mtry     Accuracy   Kappa
    2     0.5203737  0.03295115
    3     0.5251470  0.04318527
    4     0.5203775  0.03388664

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 3.
```

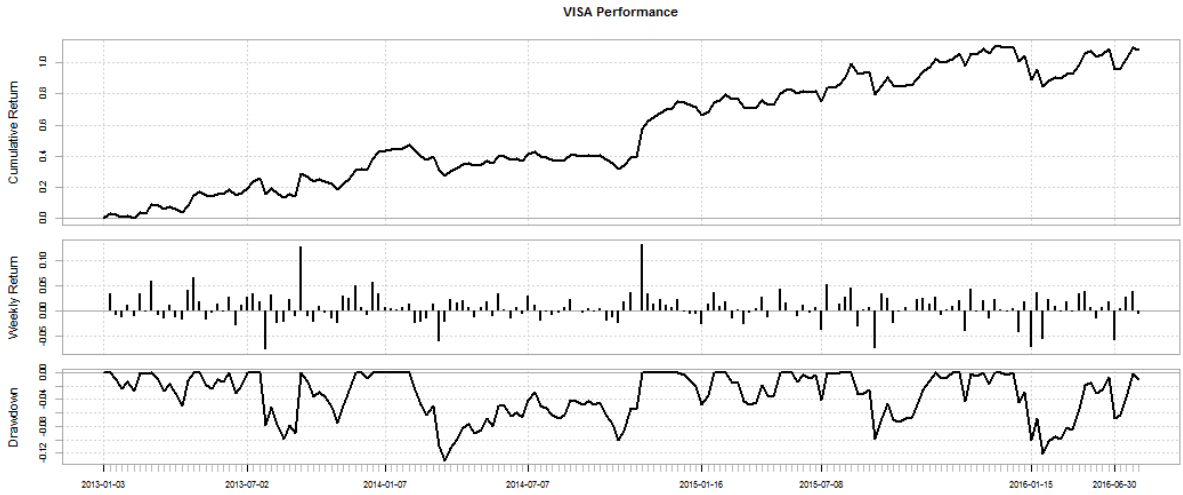
Finalize Model:

I choose the Random Forest standalone as the finalized model and save the model to file with the name “InfoTrie_Visa_ML.rds” so it can be recalled for use at a later date. In the meantime, I ran the model on the validation data, which is data that was held back from the dataset and the model has not yet seen it. This is the best way to get closer to real world results, but as the table below shows there was only a 54.6% accuracy of the algorithm on unseen data.

```
               Accuracy : 0.546
                 95% CI : (0.4689, 0.6215)
    No Information Rate : 0.5287
    P-Value [Acc > NIR] : 0.3526

                  Kappa : 0.0823
 Mcnemar's Test P-Value : 0.2606
```

This only show one side of the story as it is the accuracy of the algorithm to learn, so next I backtest the results from the unseen data in a simple method of using the price data as the index and plotting the results of 1 stock purchase at each signal, at market price with no leverage.

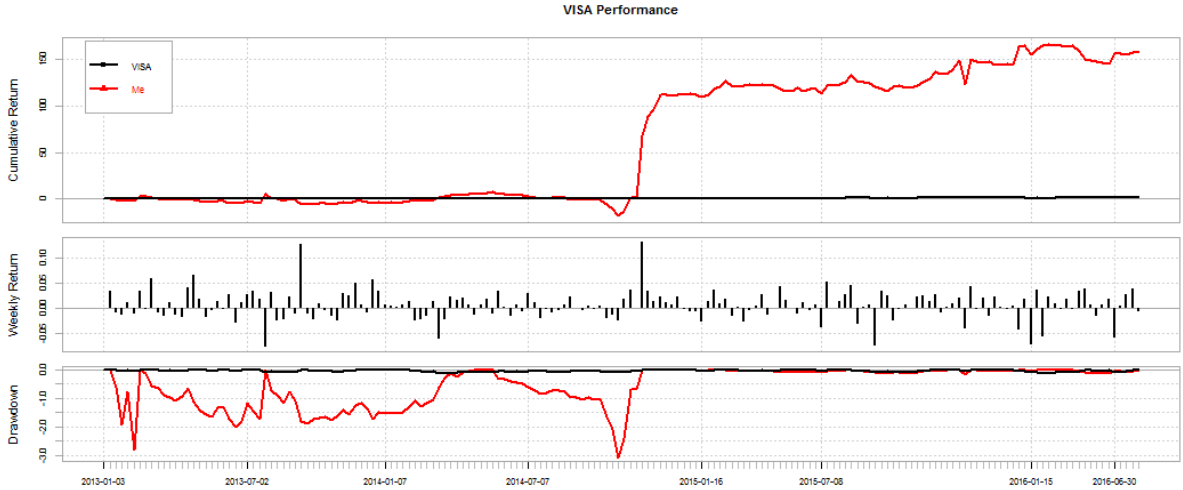
Index:



Results:

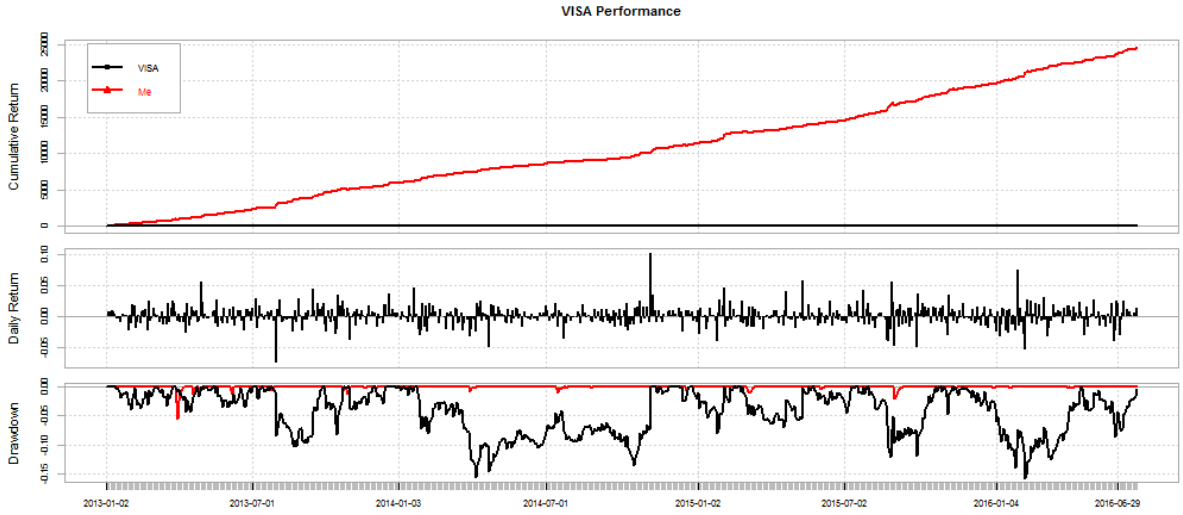
"Performance is measured as 1 Stock at market price with no leverage per trade!"

	Me	VISA
Cumulative Return	157.940	1.080
Annual Return	3.548	0.245
Annualized Sharpe Ratio	0.102	1.214
Win %	0.442	0.572
Annualized Volatility	34.892	0.202
Maximum Drawdown	-3.094	-0.132
Max Length Drawdown	35.000	39.000



Finally back testing the results of all seen data used to build the strategy:

	Me	VISA
Cumulative Return	24572.9963	1.111
Annual Return	37.4680	0.310
Annualized Sharpe Ratio	2.9375	1.283
Win %	0.9423	0.517
Annualized Volatility	12.7549	0.241
Maximum Drawdown	-0.0544	-0.158
Max Length Drawdown	64.0000	158.000



Conclusion:

Overall the results from the Random Forest algorithm were very average with only 54.6 % accuracy which did not reach my 70% target. However the algorithm still remained profitable in the back testing of the seen and unseen data. This is without any additional indicators (only price from yahoo.com and sentiment/news from quandl.com) or a learned trading strategy.

Things I would do for future reference, would be to scan a portfolio of stocks and only take the best ones that show the most predictability. On improving the system, I would focus on the work flow that I laid out at the start of this report and generate many strategies for a targeted stock or market. Then, I would forward test the strategies on a demo account for a period of time.

This will allow me to choose only the best (kind of like farming strategies). The strategies will already include two layers of indicators, one for guidance and the other for the entry/exit strategy. The guidance layer of indicators would then go through the same process laid out in this report to find the best type of algorithm. Once found, I prefer to code the machine learning algorithm, guidance indicators and entry/exit strategy indicators all into one code and add self-optimization as well.

Upon revising the workflow order I also have created a new workflow order, which is designed to cover all possible changes and leave no stone unturned. See below:

1) Process Data
2) Spot check in a harness all standard algorithms.
3) Check each data transform for all algorithms in the harness.
4) Optimize and tune each algorithm
5) Backtest each algorithm for profiting ability
6) Spot check in a harness all advance boosting and bagging algorithms
7) Optimize and tune each boosting and bagging algorithm
8) Backtest each boosting and bagging algorithm for profiting ability
9) Check stacking combinations of algorithms
10) Backtest each stacked algorithm for profiting ability
11) Choose the best algorithm for the data used.

Thank you. Contact at: https://angel.co/hayden-brown-1