Deep Learning for Momentum Stock Trading

Machine Learning Engineer Nanodegree

Fredrik Hellander

March 31st, 2017

Definition

Project Overview

In the domain of finance and trading data is available in abundance and quantitative and statistical analysis has been employed successfully for a long time. The applications are countless but the overall goal often remains the same; predict future market moves by analyzing historical data. Machine learning would seem to be an ideal fit for an industry which revolves around data but the evidence is conflicting, many attempts have been made to apply machine learning algorithms for predicting stock-market prices with limited success [1].

The rather pessimistic results from academic investigations and literature reviews is perhaps not surprising; stock markets are noisy and stochastic in nature and finding relevant patterns is not easy. Furthermore, they are also extremely competitive and short term speculative trading is in some sense a zero-sum game, to find successful trading algorithm one has to outsmart many other strategies and there is evidence that analyzing time-series of price data is insufficient [2].

Many algorithm and trading strategies trade 'momentum'; they try to find stocks in a positive or negative trend and bet that this development will continue for a period of time. The hypothesis in this project is that using a combination of historic pricing data with online metadata regarding the interest of a particular stock can be used to implement a successful momentum trading strategy.

Financial data is available from providers such as Yahoo Finance [3], Google Finance [4] and Nasdaq [5] and it will be complemented with metadata regarding the volume of Google Searches made for certain keywords, available from Google Trends [6]. The goal is to train a neural network for a regression task and predict future stock movements based on historical data.

Problem Statement

The future return of a stock is predicted by a regression model using a neural network trained on historical price and google trends data. The validity and usefulness of the predictions is evaluated by implementing the model in a trading strategy.

The task is undertaken in the following steps:

1. Acquire and preprocess Google Trends data
2. Acquire and preprocess historical stock data
3. Investigate patterns and correlations in data
4. Train a small neural network for regression task of predicting future stock movements
5. Evaluate model performances by implementing it as a trading strategy and back-testing and benchmark against index

> The ultimate evaluation and measure of usefulness of the model is if it can generate and excessive return on capital when compared to a relevant benchmark. In other words, if the model is back-tested on 10 stocks and trading for one year will the return on capital be better than simply buying and holding an equally weighted basket of these 10 stocks during the same period.

Metrics

Several different metrics is used at various stage of the project, according to the project plan above.

At step 3 correlations in the data will be investigated by measuring the linear dependence between variables. Pearson correlation coefficient for two variables *X* and *Y* is calculated as:

$$\rho = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Where $cov(X,Y)$ is the covariance between *X* and *Y* and σ the standard deviation of the respective variable.

In step 4 the model is trained using the mean squared error (MSE) as the loss function, a common loss function for regression tasks.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

The models ability to predict future returns is evaluated by calculating the R^2-score, defined below:

$$R^2 = 1 - \sum_{i=1}^{n} \left(\widehat{Y}_i - Y_i\right)^2 / \sum_{i=1}^{n} (\bar{Y} - Y_i)^2$$

Where $\widehat{Y}_i$ is the model estimate and Y_i the actual value for day i, and $\bar{Y}$ the mean of the observed data. The R^2 -score is a measure that indicated the proportion of variance in the dependent variable (future stock return) that is predicted by the model.

In step 4 of the project plan and back-testing the model as a trading strategy the performance of the trading strategy is evaluated by calculating the alpha value, α, the difference in return of the model, R_m, against the benchmark return, R_b.

$$\alpha = R_m - R_b$$

The sharp ratios of the benchmark and the model is also compared where the sharp ratio is a measure of risk adjusted return defined as annualized return, R, normalized by annualized standard deviation, σ.

$$S = \frac{R}{\sigma}$$

Annualized returns, alpha value and sharp ratio are the most common metrics for evaluating hedge funds and other exotics investment strategies.

Analysis

Data Exploration

In this project two data sources have been utilized; metadata about the number of google searches made for certain keywords from Google Trends [6] and financial stock data from Yahoo Finance [3].

Google Trends data is available on a dedicated google site for visualization and exploration and also through the Pytrends TrendReq API [7]. It provides the user with time series of relative google search volumes for keywords, topics and in regions. Daily search volumes are available for periods of up to 260 days, and longer time series are available as average weekly or monthly search volumes, for each time period requested the search volume is normalized in a range of 0 to 100.

Below in figure 1 is an graph of the relative search volumes in Sweden for the last 5 years and for the keywords 'Starbreeze' and 'Precise Biometrics', two publicly traded companies on the Stockholm stock exchange.

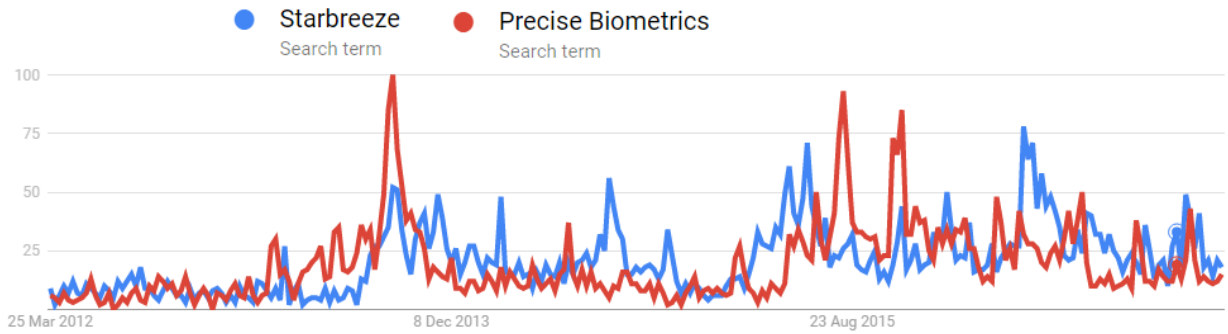


Figure 1: Illustration of Google Trends time histories for relative weekly search volumes (relative scale 0-100)

Even on a weekly average basis, as is presented here, the reported search volumes are volatile and noisy. Careful data processing is needed to extract useful insight. Below in Table 1 are descriptive statistics presented for the relative daily search volumes of the same keywords for the last 3.9 years.

	Starbreeze	**Precise Biometrics**
Count	1424	1424
Mean	21.0	20.1
Std	12.3	12.1
Min Value	0.0	0.0
1st Quartile	13.5	12.6
2nd Quartile	18.5	17.7
3Rd Quartile	25.0	24.5
Max Value	99.3	99.9

Table 1: Descriptive statistics for relative daily search volumes

The rationale behind looking at the volume of google searches is simple; humans tend to seek information as a precursor to acting and in this day and age a google is the first source to turn to. Even accessing a website that has been visited in the past can be more convenient by googling a keyword. Thus, the first step towards buying stocks in a certain company could be to google the company name. By this logic there should be a positive Pearson correlation coefficient between future stock price and google search volumes, a high search volume for a stock name should indicate a growing base of potential buyers and a shift in the supply-demand relationship.

However, there are also contradictory patterns than can arise; a large change in the price of stock can generate media interest and drive google search traffic. Hence, google searches will be lagging stock price movements. Negative news for a company will also cause an increase in google search activity and a drop in share price, a pattern leading to a negative Pearson correlation coefficient.

There are two problems in collecting the google trends data. First, the largest range of daily data available is 260 days and longer time-series needs to be stitched together. Secondly, historic data (older than 7-days) is calculated from a random sample of google searches and for keywords with low search volume this sample seems insufficient, these issued are discussed further in Methodology and Data Preprocessing.

Historical stock prices are available for free at Yahoo! Finance [3] and can be accessed through a Python API. Daily information regarding opening price (OPEN), closing price (CLOSE), intraday high (HIGH), intraday low (LOW), number of shares traded (VOLUME) and stock price adjusted for stock splits and dividends (ADJUSTED CLOSE). From these basic quantities, many indicators can be calculated, such as simple moving averages, volatility, MACD, RSI and other technical indicators used to forecast stock prices.

For this application the intention is to identify stocks in upwards or downward trends, therefore the main price signals used will be technical indicators of volume and adjusted closing price.

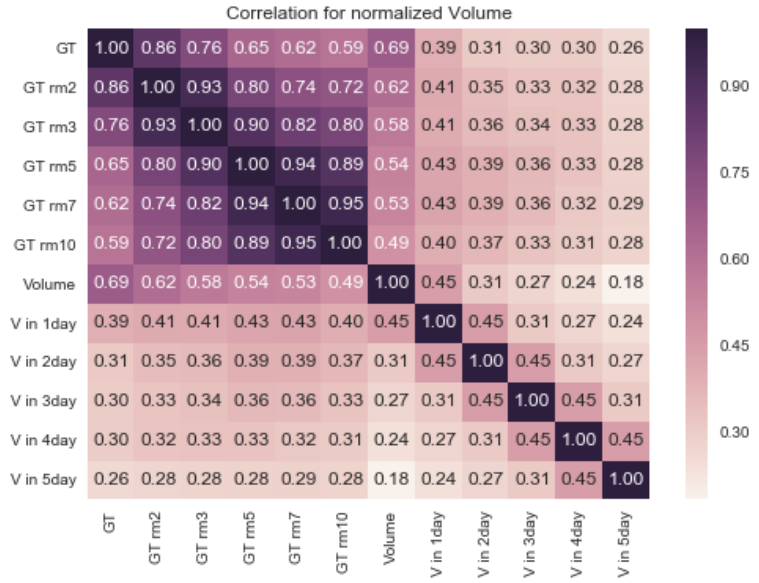
Exploratory Visualization

As previously discussed there are rivaling patterns in the relationship between the number of google searches for a stock and its share price, and a simple linear relationship is probably not possible to establish and predict future share prices.

Google Trends is intended to measure relative interest of a topic and in this case for a stock. What should be a strong correlation is then the volume of traded stocks to Google Trends. If this correlation cannot be found it can be safe to say that the particular keyword and Google Trends data holds little value for modelling a stock.

In this section the results from data exploration for the stock ‘Starbreeze’ is presented, individual variation in correlation between variables is large from one stock to another but the general conclusions drawn are representative of the sample of stocks investigated.

Figure 2: Pearson correlation between normalized trading volume of 'Starbreeze' and normalized search volumes from Google Trends for the last 4 years



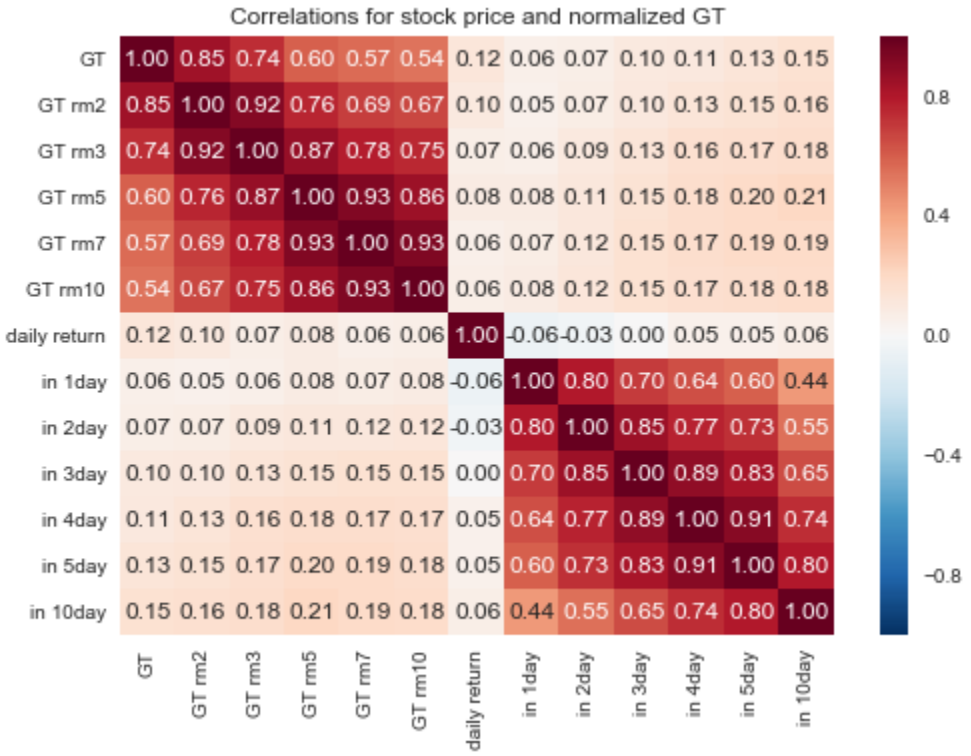
In figure 2 above a heatmap of the Pearsons correlation between Google Trends data and the volume of stocks traded for 'Starbreeze' is plotted. 'GT' is shorthand for Google Trends, and 'rmX' stands for rolling mean over X days, the correlation to the volume of stocks traded at the same day and 1-5 days into the future is shown.

Both the stock volume and Google Trends data have been normalized against a rolling mean to improve the correlation. Since trading and interest in shares fluctuate and change over time patterns are more easily discovered in relation to some moving average. The normalized volume is calculated as the difference between volume and the rolling 4-month average, divided by the 4-month average. Thus, a value of 1.0 means that today's volume of shares traded is twice as large as the rolling 4-month average. In the same fashion the Google Trends data is normalized against its 3-month rolling mean for the strongest correlation.

As can be seen in figure 2 there is a strong linear correlation between the normalized volume and the normalized Google Trend data. There is a 69% linear correlation between the variables during the same day and this correlation extends into the future; there is 43% correlation between the 5-day moving average of normalized Google Trend and the normalized volume of shares traded the next day, and 28% to the normalized volume of shares traded in 5 days' time. Hence, there is evidence to support that Google Trends data can predict future volume of trading for the stock 'Starbreeze'.

Since the ultimate goal is to predict price moves in the stock it is also relevant to check for correlations between Google Trends data and stock price changes.

Figure 3: Pearson correlations between percentage price change of the stock 'Starbreeze' and normalized search volumes from Google Trends for the last 4 years



As can be expected, Figure 3 show less correlation between normalized Google Trends data and price changes in the 'Starbreeze' stock. There is a positive linear correlation between the stock price change in 5 days and the normalized rolling 5 day mean of about 21% but many other stocks investigated exhibit smaller and even negative correlations.

The relationship between price change in a stock and Google Trends data is not as straightforward as with the Volume of the stock traded. Further data and nonlinear combination is needed to establish a pattern.

Algorithms and Techniques

Historically, neural networks have not been used extensively for creating stock trading algorithms. Neural networks have been shown to be versatile and flexible in learning complex human like tasks, like recognize objects in a picture. However, in terms of stock trading it is not certain if there actually are relevant patterns to discover; a complex and flexible model could be very prone to overfitting and not generalize well.

By this argument simpler models such as linear regressions, SVM or decision trees is perhaps a better fit. The main reason for using neural networks in this project is that they do not suffer from the curse of dimensionally in the same fashion as other techniques. The hypothesis is that individual stocks exhibit individual trading patterns and that the best choice of variables will vary. Instead of having to go through time consuming manual process of selecting the best variables we can input a large selection of indicators, moving averages and technical trading signals and optimize the neural network towards selecting the most relevant variables and ignore others.

An input vector X consisting of historic price data and complied historical metadata is fed to the network. The network consists of one or more hidden layers, the output layer is a single node summing all signals output by the previous layer and denoted Yˆ, the network estimate of the next day's price movement. In training the network output Yˆ is compared to the actual price change Y and the root-means-squared-error is calculated as the error and backpropagated to make the network learn.

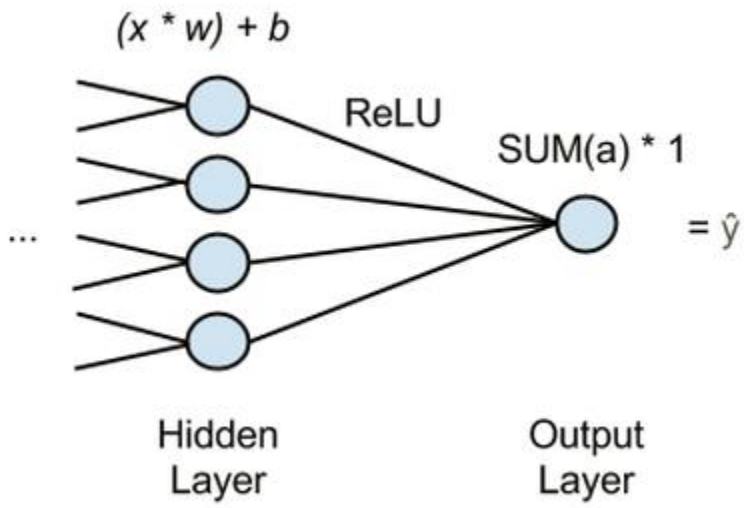


Figure 4: Illustration of neural network with regression output

A neural network model is built for several stocks and the trading algorithm will simply buy or sell the stocks with the largest predicted price moves.

Benchmark

The trading model is back-tested against unseen data and its returns will be compared to an equally weighted basket of the same stocks it is trading during the same period. In this way, the

trading model is compared to a passive 'buy and hold' strategy. The benchmark comparison will thus reveal if it any advantage is gained from trading by the model over simply owning the same stocks.

Methodology

Data Preprocessing

In the project the two data sources used are Google Trends and Yahoo Finance, which require different preprocessing.

Google Trends Data

The hypothesis for this project was that meta data for the number of google searches for a stock name should be a leading indicator into the interest of that stock. For this theory to hold it is pertinent that many different individuals purchase the stock in small volumes; if a few large investors or institutional buyers dominate the trade Google Trends data cannot be expected to give insight into the action of these few 'key' individuals.

The target stocks for this trading model is thus companies with a small market capitalization where the stock trading is dominated by private individuals and retail investors. However, getting Google Trends data for these stocks is problematic. Google does not actually return the search volumes in numbers but the relative interest during a time period, calculated from a random sample of google searches. For less popular search terms this random sample seems to be too small to use for statistical inference. For example, the search term 'Football' is popular and Google Trends will return excellent statistics over its historical popularity whereas searching for 'Curling' will show different levels of interest for similar time periods – the result for this less popular search term is dependent on which random sample that is drawn.

There seems to be a catch 22 in using Google Trends data for trading stocks; for large companies dominated by institutional investors google searches is not expected to be significant for the trading, smaller companies do not have enough search volumes for Google Trends to be reliable.

In an attempt to solve this problem an algorithm is used to merge many Google Trends results. For each specific time period and date range a new random sample of search history is returned in Google Trends. In 'data-collection/gtrend_sampler.py' a sliding window technique is used to sample each day 250 times but for a different time range, yielding a unique random sample, and

then averaging the results to generate a more reliable result. Finally, the results are normalized against a 90-day moving average for input into the neural networks model.

Yahoo Finance Data

In this application 'Adj Close', the closing share price of a stock and 'Volume' the number of shares traded is being utilized. Both the closing price and the number of shares traded are normalized against their own 90-day moving average, which allows comparison against other stocks and generates a better signal value I modelling. The normalization is calculated as

$$\bar{X}_i = \frac{X_i - (\sum_{i-n}^{n} X_i)/n}{(\sum_{i-n}^{n} X_i)/n}$$

Where $\bar{X}_i$ is the normalized value of the variable X_i, representing either Volume, Google Trends, or Adjusted Closing Price.

From the financial data a number of different technical indicators have been calculated but are not used in the final model, and the definitions are omitted.

From the financial data and adjusted closing price the percentage price gain five days into the future is calculated for each day. The goal of the regression model is to learn to predict this value.

Implementation

The implementation in this project is done in Python heavily relying on the Pandas [8], Numpy [9] and Tensorflow [10] modules. Furthermore, the data acquisition is carried out using the Pytrends [7] and the Pandas Datareader [8] package.

Implementation is done in four main python scripts:

Data-collection/gtrend_sampler.py; using the PyTrends module data from google trends is repeatedly access to average many random samples. Google limits the number of data requests per day and account to 1,600 and therefore the longest time series than can be concatenated, using a maximum sampling frequency is 4 years. Unfortunately, this leaves the data acquisition a manual process where data only can be obtained from one keyword each day. The output is saved as pickled numpy array.

Data-processing/compile_data.py; this python script reads and compiles data from Google Trends and Yahoo Finance, normalize inputs, and calculates moving averages and technical

indicators. The dataset is then split into a training, validation and testing set as input to the neural network model. In in this scrip the decision is take which inputs to use and which stocks to trade, the final model is trained and tested on 10 different stocks.

Train_NN.py; contains the definition of the neural network and trains a model on the inputs compiled by *compile_data.py*. The model is a small fully connected network with three layers, rectified linear unit as activation functions and dropout for each layer. The network is trained using stochastic gradient descent using an adagrad optimizer to reduce the mean squared error between the regression output and the percentage gain or loss of the stock five days into the future.

The neural network is flexible enough to accept a large number of variables and can be trained on different type of stock data but is otherwise kept small to reduce the problem of overfitting. Each layer contains 240 coefficients and dropout between each layer prevents overfitting. In training and validation the R-squared value of the regression output is calculated and evaluated whereas the testing data is saved for back testing and evaluating the model return.

The neural network takes a vector of inputs and outputs an indicated percentage change in the stock price for the next five days. No recurrence or memory has been implemented, but instead the network sees historically price and data by being given a range of moving averages.

The model performance is tested in the scrip *model_backtest.py* in this script the models trained from *NN_train.py* is loaded and used as a prediction for the most recent 100 days of trading. The implemented trading strategy is then simply to buy a stock the model predicts will increases over the next five days, if the model predicts a decrease in price the stock is sold short. The cumulative return and volatility is calculated and compared to the benchmark.

Refinement

Initially data from Google Trends caused a lot of confusion; before the process by which random samples are drawn was discovered no trends or patterns could be identified. The data returned was simply random noise, the implemented sampling process significantly improved the data quality but the ideally the actual search volumes would be available and not estimated by a proxy index.

In the model development predictions for several different timespans were tested. The time range of one trading week (5 days) proved to be the most successful; one a daily basis the

stocks movement are too stochastic to predict and learn any patterns. Five days seems to be the ideal timespan for identifying a trend.

In terms of training the neural network significant dropout is key to having predictive power to unseen data. The initial results were obtained without dropout and resulting in very high R^2-scores of around 0.5 for training but negative scores for validation. The complexity of a neural network gives a high risk of overfitting for stock trading. After iterating with dropout and smaller networks an R^2-score for both training and validation is around 0.1 for most stocks.

Results

Model Evaluation and Validation

The final mode has the following architecture:

3-layer Neural Network

- Input data, size: [batch, num_variables]
- 1st Layer : fully connected, size: [num_variables, 16]
- Relu
- Dropout
- 2nd Layer: fully connected, size: [16, 16]
- Relu
- Dropout
- 3rd Layer: fully connected [16, 16]
- Relu
- Dropout
- Regression Output layer, [16 1]

The model architecture was chosen after an optimization process; the size of the network was originally wider and deeper but very prone to overfitting. The width and depth of the network was successively decreased and the dropout increased until the best average validation score was obtained for all stocks.

The following graphs show the trading results for a few stocks.

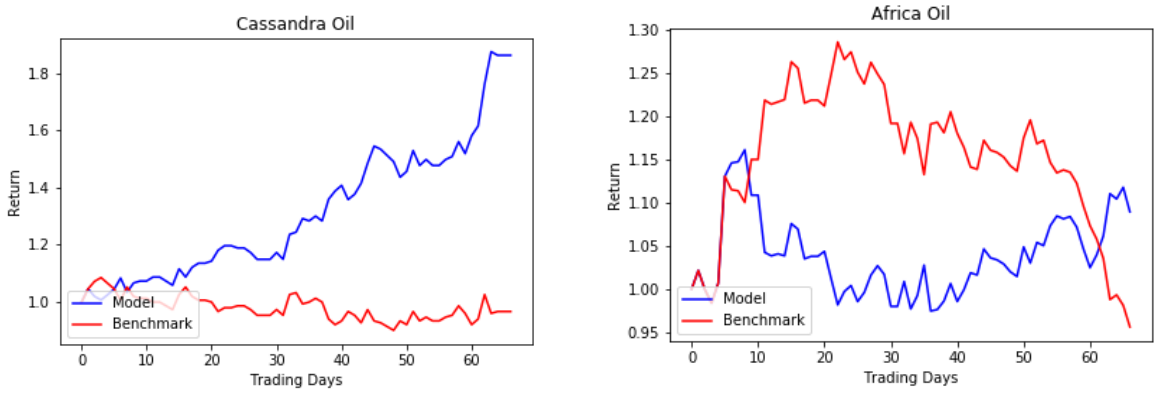


Figure 5: Trading result for stock Africa Oil and Cassandra Oil

The model is very successful in trading the 'Cassandra Oil' stock, with stock trading it is sometimes hard to distinguish luck from a successful strategy but a momentum trading strategy should perform well for stocks in clear trading patterns and during these days 'Cassandra Oil' is continuously falling, the overall return of the model is 80% whereas the traded stock is down 10%. For 'Africa Oil' the model underperforms the stock initially where rapid changes in price occur and no clear pattern exits, once the stock gradually starts declining, and a falling trend is established the model outperforms the stock, the model return for the period is 10% whereas the traded stock is down 5%.

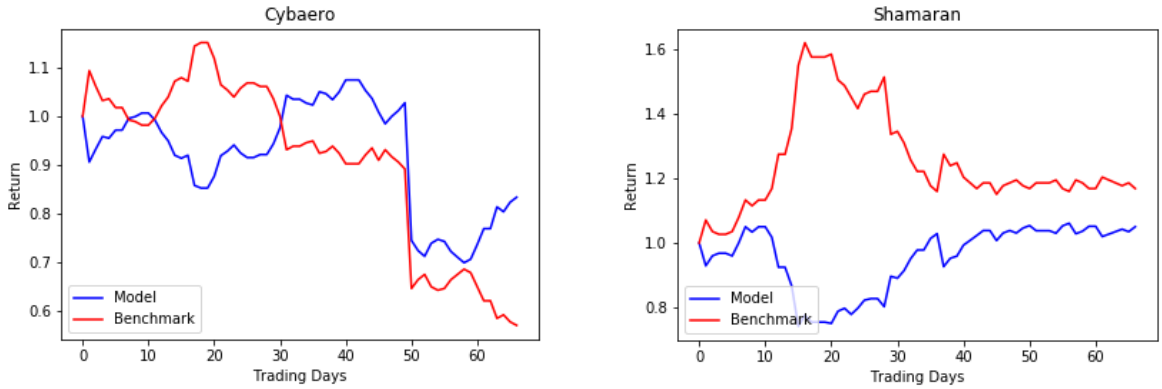


Figure 6: Trading results for stock Cybaero and Shamaran

For the stock 'Cybaero' the initial trend is unclear and the model underperforms the stock, at day 50 there is a sharp drop in the share price, driven by a press release from the company which the model natural fails to anticipate and there is a corresponding drop in return for both the stock and the model. After the sharp drop in share price a negative trend is established with the model correctly predicts and the model return ends the trading period with a positive alpha

of 20%. For the stock ‘Shamaran’ the initial sharp increase in stock price is not correctly traded, perhaps the price movements are too fast for the model to identify this upward trend. The stock peaks around day 20 and then gradually declines, the model trades this range better and recover some of the loss but overall during the period underperforms the stock by about 20%.

Justification

In order to get a more comprehensive view of the model performance the model return for all 10 stocks considered is plotted against an equally weighted benchmark of the same stocks.

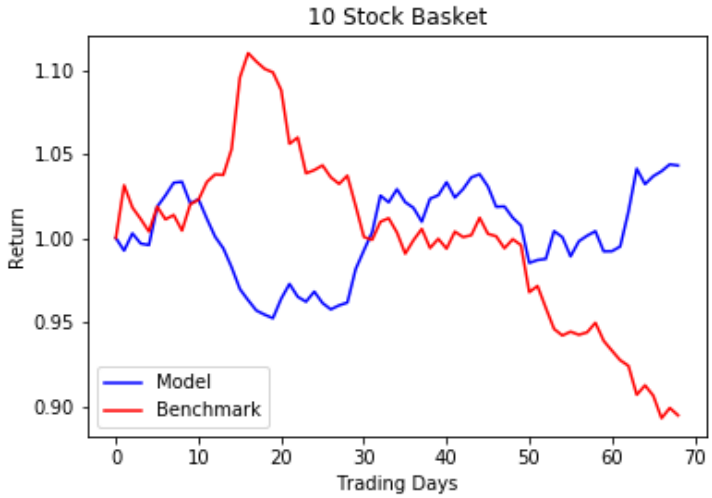


Figure 7: Comparison of model return against equally weighted benchmark of 10 stocks

In figure 6 above the return of the trading model can be seen to underperform the basket benchmark initially, for rapid price changes in stock price as occurs initially the model actually predicts declines in stock prices and the return is negative. After the peak in stock prices around day 20 a gradual decline in stock price occurs. The model is more successful in correctly identifying this trend and the model starts to outperform the benchmark.

The absolute return for the model over the period is about 5% whereas the benchmark declines with 10%, the alpha value of the model is calculated to exactly 15.34% and significantly outperforms the benchmark during the period.

The annualized return of the model is 16.9% and the annualized alpha value is 52.8%, accounting for volatility and calculating the risk weighted measure by the sharp ratio gives an annual sharp ratio of 0.41 for the model and -0.37 for the benchmark. As a further point of reference the sharp ratio for the 30 most traded stocks on the Stockholm Stock Exchange is 0.64 over the last 3 years.

Conclusion

Reflection

In this project a stock trading algorithm has been developed for small- and micro-cap stocks on the Stockholm Stock Exchange. The model utilizes data from Google Trends and time series of stock price data for Yahoo Finance to identify stocks in a rising or declining trading pattern, a momentum strategy.

For the time period tested and the stocks traded the model significantly outperforms the benchmark it is compared against. Using the model generates excessive return without any additional volatility or risk exposure and thus seems to be the better option than just buying and holding the same stocks.

A neural network is used to identify trading patterns for each individual stock, the neural network is prone to overfitting but has the benefit of accepting a large number of input variables and selecting the most important features for each stock. The model architecture thus does not need to be changed if different stocks are to be traded and additional variables and price signals are to be explored.

However, it is not possible to make any definitive judgement regarding the future performance of the model. Due to an unexpected difficulty in collecting data from Google Trends the number of stocks the model is tested on and the period is has been back tested for is very limited.

Improvement

As discussed earlier the problem with Google Trends data is that it is deduced from a random sample of all search volume history, for less popular search terms this data becomes notoriously unreliable. In order to rectify the problem as much data as possible is collected by sampling and averaged, however, given the limits in the number of data requests allowed information for only one stock and a 4-year period can be compiled for each day.

A significant improvement to this project and allowing for much more rigorous training and testing would be to use the raw data on search volumes directly, instead of estimations. This data can perhaps be purchased but is not available freely at this moment. There is indeed evidence to the predictive power of Google Trends in stock trading and with access to this data directly it would be very interesting to build an improved model and perform a more thorough evaluation.

References

[1]
Fernando Fernández-Rodríguez, Christian González-Martel and Simón Sosvilla-Rivero. On the profitability of Technical Trading Rules based on Artificial Neural Networks: Evidence from the Madrid Stock Market. Elsevier 2000

[2]
Evan Hurwitz, Tshilidzi Marwala,State of the Art Review for Applying Computational Intelligence and Machine Learning Techniques to Portfolio Optimisation. arXiv:0910.2276

[3]
https://www.finance.yahoo.com

[4]
https://www.google.com/finance

[5]
https://data.nasdaq.com/

[6]
https://trends.google.com/trends/

[7]
https://github.com/GeneralMills/pytrends

[8]
http://pandas.pydata.org/

[9]
http://www.numpy.org/

[10]
http://www.tensorflow.org/