A CONSTITUENT SENTIMENT APPROACH TO STOCK MARKET TREND PREDICTION

by Frédéric Maréchal

A thesis submitted in partial fulfilment of the requirements for a MSc Data Science
at Goldsmith's College (University London)
Computing Department

September 2017

Thesis Supervisor: Dr Daniel Stamate

Preface

I would like to thank my partner and my son, Lucille Boquet and Maximilien Maréchal, who have supported me relentlessly and patiently throughout this last year. My parents, Claude and Nicole Maréchal, who have given me the strength to succeed through hardship. My supervisor, Dr Daniel Stamate, who has guided my footsteps throughout this process.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Pseudocode

Abstract

Numerous studies, involving machine learning models, have demonstrated that sentiment expressed on diverse media have predictive power on the stock market trend. However, they employ machine learning methodologies that are not suited for time series. In this thesis, we explore the prediction power of the XLE index constituents' sentiment using a robust approach. We start by implementing a custom feature selection which feeds into numerous machine learning models. Then, a proxy for the XLE index trend prediction accuracy rate is generated from the sum of the weighted index constituents' accuracy rate, produced by the 'best' models. Scenarios involving the addition of the sentiment or sentiment momentum variables show that sentiment does not add prediction power to the trend prediction. Consequently, we studied the prediction power of sentiment and sentiment momentum on the XLE index and its constituents' volatility. In this case, there is a clear reduction in root mean square error (RMSE), indicating that sentiment has predictive power on volatility.

Chapter 1 Introduction

Stock market trend prediction is at the centre of investment strategies. The fluctuation of assets and their predictability has been studied by many researchers over several decades. Supporters of the Efficient Market Hypothesis (EMH) and the random walk theory consider that it is impossible to predict future trends. However, some studies have proven the existence of volatility clustering and regime change under certain conditions that made the prediction of market return possible, or at least partially.

The main tools for trend prediction have been so far, the use of technical analysis (i.e. market data derived measures) and fundamental analysis (i.e. accounting and economic aggregates) to help build a view on assets' future trend.

The development of statistics for time series and the emergence of fast and cheap processors have enabled numerous researchers to explore trend forecasting from different angles. Traditionally, pure statistical variable autoregressive models have been implemented to verify a model capacity to predict assets' future trend. More recently, studies have focused their attention on machine learning models coupled with technical and fundamental data.

This research extends on current machine learning approaches. It aims at establishing whether sentiment has prediction power on the XLE index (US energy index) trend. First, the research considers the effect of the sentiment at the index level. Second, it analyses the effect of sentiment on the predictability of each constituent's trend. It then studies whether the reconstruction of the index prediction from its constituents' sentiment improves the trend predictability. This study also examines the impact of sentiment on the volatility predictability at the index and constituents level.

In nutshell, the sentiment trend prediction analysis employs supervised classification models and compare their performance with and without the sentiment variable. As for the volatility prediction, supervised regression models are used.

The innovation of this research, relating to the trend prediction, lies with the number of technical indicators under analysis, the "2-way" feature selection process (a Wrapper and a Filter method), as well as the study of an entire index constituents' sentiment impact.

Each asset (i.e. index and stock) is assigned more than 50 technical explanatory variables. The "2-way" feature selection reduces the space of explanatory variables to the minimum set that maximises the test accuracy rate for each machine learning models, on a 20 years long training data set. The machine learning models' basket is composed of 8 algorithms; such as discriminant models, random forest, recurrent neural networks. etc. Once, the best model, coupled with its feature selection and hyper-parameters sets, has been found for each stock, a base scenario is then built. The base scenario is generated for both i) the XLE index and ii) the XLE index's constituents.

Numerous scenarios are run with the addition of the sentiment and sentiment momentum to examine whether the addition of these variables add prediction power to the base scenario.

The second experiment, involving the volatility as a response variable, mimics largely the previous framework. However, it is restricted to recurrent neural network models.

The remainder of this thesis is organised as follows. Chapter 2 critically examines the current researches existing in this field. It also details the machine learning models that are used for these experiments. Chapter 3 reviews in details the scope of the experiment. It describes the software infrastructure developed to enable easy re-use and extendibility. It also details the data collection process, the pre-processing, the feature selection and the performance measure generation under a sliding time window framework. Chapter 4

reveals the findings of both experiments on both i) the trend and ii) volatility prediction improvement, when sentiment is involved. Chapter 5 concludes and proposes further axes of research.

1.1 Definition of Terms

The study of index and stock market trend prediction requires the clarification of some problem domain terms.

Business terms:

- Technical Analysis: a trading tool employed to forecast future asset moves by analysing statistics gathered from trading activity (Technical Analysis, 2017).
- Technical Indicator: a statistical or otherwise fabricated metric, usually using price move (Technical Indicator, 2017).
- Sentiment Analysis: it is the process of opinion mining to gauge positive or negative 'feeling' the market has relating to a stock performance.
- XLE Index: it is an exchange traded fund based on the energy sector industry. It contains 36 New York listed energy sector stocks (e.g. Exxon Mobil Corp, Chevron Corp & Co, Schlumberger Ltd).
- Index constituents: the list of all stocks that are contained in the index.
- Asset: in the context of this thesis it means either a stock or an index.

Software development terms:

- API: the application program interface (API) defines how software components should interact.
- ETL: it is a short name for an 'Extract, Transform, Load' IT system
- OHLC: it means the average of the Open/High/Low/Close price at *t*.
- HLC: it means the average of the High/Low/Close price at *t*.
- NSE: it stands for the New York Stock Exchange.

1.2 Aims and Objectives of the Research Project

The corner stone of this project is to establish whether sentiment and/or sentiment momentum have predictive power on the XLE index trend prediction. Two cases are studied:

- Case 1 – the sentiment is applied at the index level. As the index sentiment is not provided by *Quandl*, a proxy is generated from the weighted constituents' sentiment.
- Case 2 – the sentiment impact is analysed for each individual constituent. Then, the sum of the weighted stocks' prediction accuracy rates, is generated.

In both case, the sentiment prediction performance is compared to the original index prediction performance, i.e. where sentiment is excluded.

The second axis of research focuses on the impact of sentiment on i) the index volatility prediction and ii) the individual stock's volatility prediction.

To achieve this aim, an *Extract Transform* and *Load* (ETL), supporting a machine learning pluggable framework, has been developed.

1.3 Contribution to Knowledge

With the advent of new technologies and cheap memory/hardware, the last decade has seen the birth of automatic and systematic trading strategies managed by 'intelligent' computers, without the need for human intervention.
The automatic trading decision criteria could be as simple as: i) a hard–coded set of pre–defined rules based on technical and/or fundament signals or ii) the use of *Artificial Intelligence* (AI) to self–learn rules (e.g. Q–learning algorithm) to take buy/sell positions on the stock market. This study complements and enhance the current approaches as it provides a methodology to integrate *Quandl* sentiment into the decision–making process. However, it is not limited to testing the addition of sentiment, it could be used for testing the prediction power of any other variables.
This methodology could be used for index arbitrage or portfolio allocation strategies as part of either i) a fully automated trading tool or ii) an advising tool for enhancing human–based decision–making process.
Although the infrastructure has primarily been used for forecasting the market trend, it could be extended to other response variables.
Furthermore, this experiment required the engineering of a consistent and extendible piece of software that gaps missing functionality in the Caret library suite. Indeed, Caret only supports the generation of time windows, it does not natively support time series training/optimising and testing. This proposed solution could serve as a potential solution for a future integration into the Caret package or other related libraries.

Chapter 2 The Literature Review

2.1 The Market Efficiency

The ability to predict future stock market moves has always been the investors' grail. Currently, there is no consensus among scholars on the ability to predict these moves. The Efficient Market Hypothesis (EMH), introduced by Fama (1965), is broken down in three forms of varying strength:

- The *weak* form argues that the current market price reflects all current information available to the investor. There are no serial price dependencies, therefore future prices cannot be predicted by historical prices. Consequently, technical analysis cannot help with producing consistent excess returns.
- The *semi–strong* EMH is a superset of the *weak* EMH with the added concept of instantaneous reactions of market prices to new public information.
- The *strong* EMH goes one step further by asserting that non–public information (e.g. "insider" information) is also reflected instantaneous in the stock market prices.

From a EMH view point, the market trend prediction is impossible. The same conclusion is reached by the supporters of the Random Walk. The later approach the problem from a different angle. Bachelier (1900) and Malkiel (1973) proved that market prices followed a Random Walk, which means the future market price is equally likely to move up or down.

Nevertheless, volatility clustering and sharp regime change and market anomalies (Keim, 2006) can be witnessed in practice. Ang and Timmerman (2011) associated these changes to transitory market "jumps" which affect the stocks' trend and volatility. Such events could be due to crises such as Black Friday in 1929, the oil crisis in 1973 or the 2008–2009 subprime crisis. Regime changes have long been studied in the land of Exchange Rate, Interest Rates, Asset Allocation and Equity Return. Research studies found that aggregate stock market returns were predictable. However, the quality of the predictability power varies considerable over time. Henkel, Martin and Nardari (2011) determined that the predictability level was also dependent on the business cycle. According to the author, the predictability power was the strongest during market downturn.

Niederhoffer et al. (1966) proved that Foreign Exchange price predictability came from the traditional market dynamics, where a price *'bounces'* between a bid–ask bracket. This interval defines the highest and lowest price individuals are ready to buy (or sell) the asset at a given point in time.

Rechenthin (2014) devised a simple experiment to predict price using probability. In the EMH framework, market independence implies that the probability of the current price momentum to move up, when the previous price momentum was down, should bear the same probability than the current price momentum probability going up, ignoring past prices. In other words:

$$P(\Delta p_t = \text{up} \mid \Delta p_{t-1} = \text{down}) = P(\Delta p_t = \text{up}) \quad (1.0)$$

where: $\Delta p_t = p_t - p_{t-1}$

p_t: price at period t (i.e. current price)

p_{t-1}: price at period t–1

This experimentation demonstrated that the probability of an upward price movement was approximately 50%. Whereas, the conditional probability that consisted of predicting an upward price movement at date t, given the price momentum at t–1, generated a probability approximating 42.2%. This small–scale experiment showed that the price trend at *t*, is not entirely independent of the price *t*–1 trend.

In conclusion, the literature implies that correctly predicting stock price trend is feasible, at least partially (i.e. not for all time windows) and that EMH, and by extension stock price independence, do not hold true.

2.2 The Stock Market Prediction Methodologies

The main two axes employed to attempt to discover market price trends have historically been: i) fundamental economic analysis and ii) technical analysis. The former concerns the analysis of companies accounting balances and economic ecosystem that may affect it (i.e. macro–/micro–economic events). The later involves analysing historical prices, volatilities and volumes to predict future price moves, Suresh (2013).
In recent years, another direction has been taken by researchers. The deployment of machine learning algorithms coupled with sentiment analysis have been studied to try and predict the stock market volatilities and trends. The remainder of this section detailed the different approach available in the literature and their potential limitations.

2.2.1 Machine Learning Algorithms and Technical Indicators

Vaiz and Ramaswami (2016) investigated the prediction power of technical analysis indicators, on their own, to estimate the future price of a stock traded on an exchange. For this, the daily OHLC price and volume data for the six highest market capitalisation companies of the NSE were gathered for a period spanning from January 2012 to December 2015. Twenty–two technical indicators (e.g. RSI, EMA, MACD, etc.), and three supervised classification tree models; namely the Iterative Dichotomizer (ID3) classification, the regression trees (CART) and the C5.0, were selected to perform test predictions. The close price and the technical indicators were used respectively as the response and explanatory variables. The author concluded that, in the best–case scenario, these models achieved on average was 85% of the accuracy in predicting the market trend. He also claimed an accuracy rate improvement compared to similar studies by Parikh and Shah (2015), Senyurt and Subasi A. (n.d.) and Vaiz and Ramaswami (2016). These experiments used Decision Tree model, Random Forest, and a Naïve Bayesian classifier with respectively 80.08%, 78.8% and 73.8% of test accuracy.

2.2.2 Machine Learning Algorithms and Fundamental Indicators

In their research for establishing the predicting power of fundament variables in forecasting the market movement, Joshi et al (2013) proved that only fundamental explanatory variable was capable of a prediction accuracy of 61.86%, with a Random Forest. When technical and fundamental variables were mixed, the best model, a decision tree, returned 80.08% performance. Imandoust and Bolandraftar (2014) reached the same conclusions in their independent research. These results comforted the idea that a non–negligible proportion of the trend was explained by other factors.

2.2.3 Machine Learning Algorithms and Sentiment

2.2.3.1 Explanatory Variables – Only Sentiment

The most noticeable feature that has lately attracted researchers' attention is the study of the impact of sentiment analysis. Meesad and Li (2014) presented a methodology involving the steps shown in Figure 1.

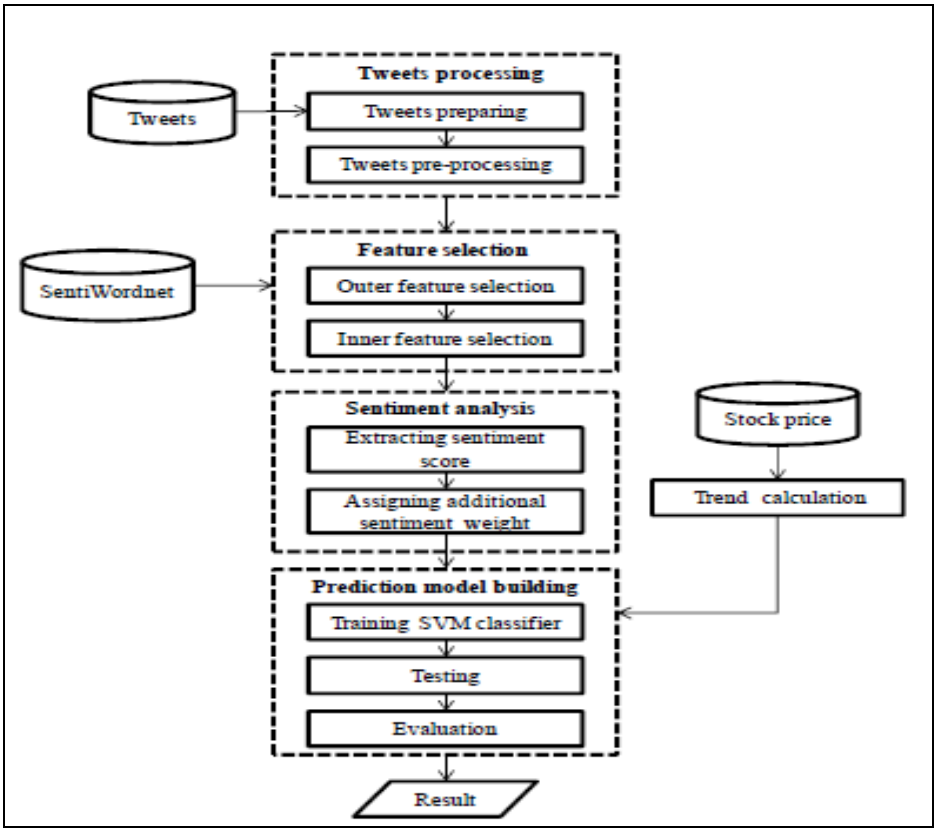


Figure 1 – A Twitter's sentiment analysis infrastructure

The steps are detailed below:

- Tweets processing: this phase generated Bag–of–words, which were extracted from 4,622 tweets, covering an 11 months' period. The Bag–of–words only contained words that might bear sentiment information. The punctuation, symbols, urls, stop words (function words such as 'a', 'and', 'of', etc.), and words with similar meaning/root (stemming) were removed. Each tweet became a feature (or vector) of word.

- Hybrid feature selection: this step combines an outer an inner cross–validation feature reduction steps.

- Corpus sentiment extract: this layer propagated each feature word into the *SentiWordNet* API, an opinion lexicon library. It returned a *Positive/Neutral/Negative* probability for the given word. A feature sentiment; $sent_{(ti)}$ was calculated by averaging the difference between positive scores ($score_{(pos)}$) and negative scores ($score_{(pos)}$), over the number of words (n) in a feature. The formula is given below:

$$senti_{(ti)} = (\sum score_{(pos)} - \sum score_{(neg)}) / n \quad (2.0)$$

- A new weight W_{ij}, derived from (2.1), was built and served as attribute in the feature selection:

$$Wij = \begin{cases} vij + senti(ti), & senti(ti) > 0 \\ -1 * vij + senti(ti), & senti(ti) < 0 \end{cases} \quad (2.1)$$

$v_{ij:}$ represents the weight of the token t_j in d_i as defined in the Term Frequency–Inverse Document Frequency (TF–IDF) methodology, defined as:

$$vij = tf(di, tj) * idf(tj) \quad (2.2)$$

Where $tf(d_{i,}t_j)$: the frequency of the token t_j in the document d_i
And $\sum Nt_{ij}$: the total number of all token present in d_i
And $Df(t_j)$: the total number of documents containing token t_j
And D : total number of documents

$$where\ tf(di, tj) = \frac{Ntij}{\sum Ntik} \quad (2.3)$$

$$and\ idf(tj) = \log(\frac{|D|}{\mathrm{df(tj)}}) \quad (2.4)$$

A document d_i was represented by a word vector:

$$\boldsymbol{vi} = [vi1, vi2, , \dots, vin]$$

This was this weight added as part of the feature selection to predict the trend.

- The trend was defined as follows:

$$Trend = \begin{cases} up, & price\ today - price\ yesterday\ > 0 \\ down, & price\ today - price\ yesterday < 0 \end{cases} \quad (2.5)$$

The model consisted of a response variable; the trend generated in (2.5) as well as the list of attributes, generated in (2.1). Then, the sentiment attributes were feature selected and transformed, as per (2.2). The paper claimed that fitting Support Vector Machine (SVM) algorithm, against these attributes, combined with a Leave–One–Out (LOO) cross–validation method yielded 93.4% prediction test performance.

2.2.3.1 Explanatory Variables – Technical Indicators and Sentiment

A logical extension of both above methodologies was developed by Halgamuge (2007). The research presented a model that used both news releases and technical indicators as predicators to enhance the predictability of the daily stock price trends. The experiment consisted of i) building seven technical indicators and ii) tokenising news articles (both company and market specific) to serve as attributes in a SVM model. The predicted response variable was the daily stock price of BHP Billiton Ltd (BHP.AX). The general approach is summarised in Figure 2, borrowed from Halgamuge (2007), below:

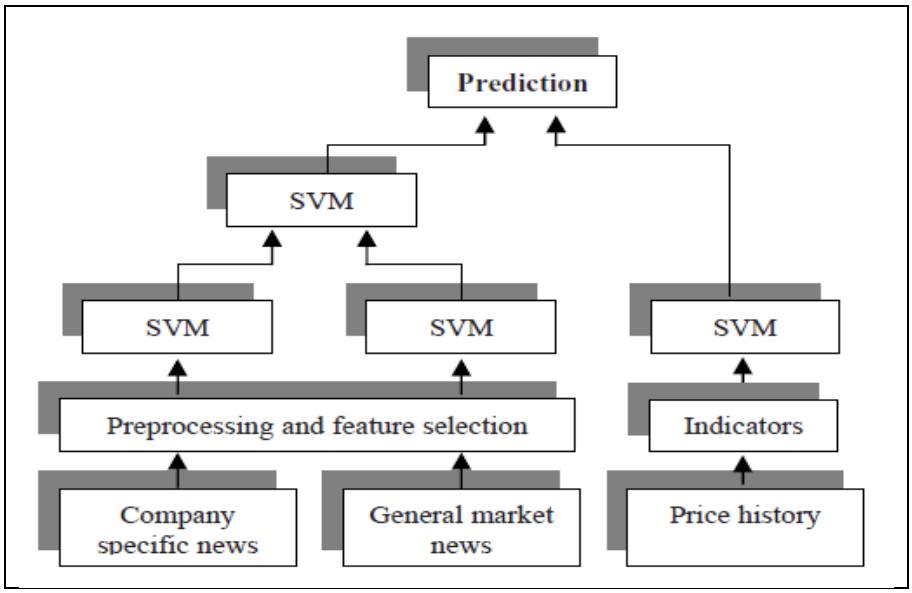


Figure 2 – Technical and sentiment analysis prediction workflow

The selected technical analysis indices were constructed from the OHLC stock prices. The sentiment extraction methodology was very close to the one described previously, namely i) each piece of news was vectorised, ii) each element (i.e. word) in the vector was tested to establish whether it belonged to the vocabulary, iii) stop words were removed, iv) the remaining word list was fed into the *WordNet* API. The later replaced words with a higher concept. At the end, the concepts were weighted using TF–IDF mechanism. The two groups of news were trained and classified using a SVM model. Then, the result was loaded into another SVM model to generate a price trend joint prediction. The model fit was based on a training and validation set. There was no cross–validation. The SVM prediction test results for this experiment were:

- 58.8% for a model only containing technical indicators.
- 62.5% for a model only using the company news.
- 50.0% for a model only using the market news.
- 64.77% for a model using both the company and market news.
- 70.1% for a model using the price, the company and market news.

The study then reviewed the impact of both technical analysis and sentiment in the case of simulated trading strategy. According to the author a two–month trading simulation generated $511 (with technical and sentiment), versus $284 and $275 respectively when the technical analysis and sentiment were used independently.

2.2.4 The Vector Autoregressive Model (VAR) Statistical Model and Sentiment

2.2.4.1 The Pure Statistical Approach

The literature analysed in the previous section focused on machine learning algorithm to establish the predictive power of sentiment on the stock market trend. However, they fail short of analysing the statistical significance and the stability of the prediction accuracy. The research carried out by Gilbert and Karahalios (2010) offers a step in this direction. The study was carried out on a dataset of 20 million posts from the LiveJournal website. The authors proved, using a Granger–causal framework and a Monte Carlo simulation, that *Negative* sentiment tended to influence negatively the S&P500. For this, they used a specialised Live–Journal corpus and classified articles' sentences to distinguish between *anxious, worried, nervous and fearful* vs *not anxious* sentiment. This was an improvement on the methodologies deployed by Halgamuge (2007) and Meesad and Li (2014), (which used non–domain specific corpora).

Gilbert and Karahalios (2010) used two 10–fold cross–validated classifiers: i) a boosted decision tree (C1) and ii) a bagged Complement Naïve Bayes (C2) to classify words in the *anxious, worried, nervous and fearful* sentiment vs *not anxious* category. The authors defined the Anxiety index as:

$$A_t = \log(C_{t+1}) - \log(C_t), \text{ with } C_t = \max(C1_t, C2_t) \quad (2.6)$$

The daily close prices (SP_t) were downloaded from the *Yahoo!Finance* website. From the price data, the authors derived a several variables:

The daily stock price log–return as R_t, as $R_t = \log(SP_{t+1}) - \log(SP_t)$ (2.7)

The stock market 'acceleration' metric as $M_t = R_{t+1} - R_t$ (2.8)

The proxy for volatility as $VOL_t = (R_{t+1} * R_{t+1}) - (R_t * R_t)$ (2.9)

$$VLM_t = \log(Volume_t / Volume_{t-1}) \quad (2.10)$$

The researchers then statistically tested the relationship between the Anxiety Index and the stock market trend. A linear Granger causality (Granger, 1969) was used to check whether the sentiment information described in (2.6), was useful in forecasting the stock trend. The methodology involves comparing the variance explained by two linear models. The first model (M1) only used price at *t*, lagged prices or derived price measures, the second model (M2) built on the first one by adding the Anxiety Index predicator (A_t).

$$\boldsymbol{M1}:\ M_t = \alpha + \sum_{i=1}^{3} \beta_i\, M_{t-i} + \sum_{i=1}^{3} \Upsilon_i\, VOL_{t-i} + \sum_{i=1}^{3} \delta_i\, VML_{t-i} + \varepsilon_{1t} \qquad (2.11)$$

$$\boldsymbol{M2}:\ M_t = \alpha + \sum_{i=1}^{3} \beta_i\, M_{t-i} + \sum_{i=1}^{3} \Upsilon_i\, VOL_{t-i} + \sum_{i=1}^{3} \delta_i\, VML_{t-i} + \sum_{i=1}^{3} \lambda_i\, A_{t-i} + \varepsilon_{2t} \qquad (2.12)$$

The result of this analysis showed that M2 performed significantly better than M1. The Durbin–Watson test also demonstrated that there was no serious autocorrelation issue. It also revealed that high anxiety level impact negatively on the market. However, the authors recognised that the model exhibited heteroscedasticity. Furthermore, the residuals did not follow a normal distribution. Consequently, there was a risk that the true distribution could deviate enough from the theoretical F–Statistic, inducing a bias in the p–values.
Therefore, a Monte Carlo simulations framework was deployed to confirm/inform these values. Formally, a new C_t was drawn from the Anxiety Index distribution, and the above experiment was run one million times. Each time, the F–Statistic was recorded when it was larger than the F–Statistic obtained initially. The sum of all these F–Statistic was averaged over the number of simulations. They provided the experiment p–value that confirmed the statistical significance of the initial results. The final main finding came from reversing the Granger Causality (M2 –> M1). The test showed that the recent past had no predictive power on the Anxiety Index. Therefore, it seemed to indicate that the source of fluctuation of the Anxiety Index did not come from the stock market price fluctuations.

2.2.4.2 A Mixed Statistical and Machine Learning Approach

Although the above methodology demonstrated a more robust statistical approach, there were some limitations in the tools employed to establish the impact of *Negative* sentiment on the stock trend. The linear Granger causality test assumed that the two models under analysis were linear. Moreover, the authors recognised that it was sensitive to non–stationary time series, where the mean, variance and autocorrelation varied with time. Furthermore, the residuals were not normally–distributed. Gilbert and Karahalios (2010) also observed that there was a significant statistical difference between the Monte Carlo simulation empirical F–Statistic and the theoretical F–Statistic.
In their paper, Olaniyan et al (2015) critically investigated the suitability of using a Monte Carlo simulation as a validation tool to offset the shortcomings of the linear Granger Causality test. Using a Monte Carlo inverse transform and a bootstrap sampling method, the authors proved that empirical and expected F–Statistic were still significantly apart.
The researchers also conducted a non–parametric statistical test, developed by Baeck and Brock (1992), on the residuals of the VAR models M1 and M2. This test concluded that the original findings, relative to the predictive power of the Anxiety Index on the stock market trend, were biased by the presence of heteroscedasticity in the residuals.
Olaniyan, Stamate and Logofatu (2015) inferred that, contrary to the results obtained by Gilbert and Karahalios (2010), the Anxiety Index did not possess any significant predictive information on the stock market.
In the light of the above conclusions, Olaniyan R. et al (2015) re–oriented the previous experiment. First, they introduced a new set of attributes: lagged volatilities and *Positive/Negative* sentiment variables. The volatilities were generated via an exponential GARCH (1,1) process (a.k.a. EGARCH). Second, the authors abandoned the Anxiety Index proposed by Gilbert and Karahalios (2010). Instead, they replaced it by the Downside Hedge Twitter Sentiment indicator[1]. Third, the researchers used i) a non–parametric and nonlinear

[1] Available at http://www.downsidehedge.com/twitter–stock–market–sentiment–download/

approach and ii) a hybrid GARCH and artificial neural networks to test the prediction power of the *Positive* and *Negative* sentiments. The new experimental settings are detailed in the next paragraph.

Olaniyan R. et al (2015) formed a new VAR model, similar to (2.11) and (2.12), with the following attributes:

- a measure representing the sentiment 'momentum' between two time–periods (At), as shown in (2.14)
- a dummy variable (D_t) to capture the *Positive* (P_t) and *Negative* (N_t) sentiment impact on the market volatility, as shown in (2.15),
- a volatility time series denoted as Q_t (2.17)

$A_t = S_t+1 – S_t$, with S_t being the sentiment level at date t). (2.13)

$$D_t = \begin{cases} D_t = 1, where\ A_t - A_{t-1} > 0 \\ D_t = 0, where\ A_t - A_{t-1} \leq 0 \end{cases} \quad (2.14)$$

$P_t = A^2_t * D_t$ (2.15)

$N_t = A^2_t – (1– D_t)$ (2.16)

$Q_t = \ln(\sigma^2_t)$ (2.17)

$V_t = \log(Volume_t/Volume_{t-1})$ (2.18)

$$\boldsymbol{M1}: \quad M_t = \alpha + \sum_{i=1}^{3} \beta_i M_{t-i} + \sum_{i=1}^{3} \Upsilon_i V_{t-i} + \sum_{i=1}^{3} \delta_i Q_{t-i} + \varepsilon_{1t} \quad (2.19)$$

$$\boldsymbol{M2}: \quad M_t = \alpha + \sum_{i=1}^{3} \beta_i M_{t-i} + \sum_{i=1}^{3} \Upsilon_i V_{t-i} + \sum_{i=1}^{3} \delta_i Q_{t-i} + \sum_{i=1}^{3} \lambda_i A_{t-i} + \varepsilon_{2t} \quad (2.20)$$

Traditionally volatility models were built using a GARCH methodology, which used the residuals from a linear model as input to produce the volatility time series. However, this approach did not capture the impact of market asymmetry on volatility. Consequently, the authors used the Exponential GARGH (a.k.a. EGARCH) in an attempt to overcome this limitation. The Ljung–Box test, that tests for temporal autocorrelation in the errors of a regression model, showed that *Positive* sentiments reduced volatility (as expected), however *Negative* sentiments did not seem to have a significant impact on volatility. This was an unexpected result. Running a linear Granger causality test showed that M2 would outperform M1.

The authors demonstrated that the residuals:

i) presented autocorrelation
ii) did not follow a normal distribution
iii) were heteroscedastic in variance.

Although a Monte Carlo simulation seemed to confirm the Granger causality results, once again, the theoretical F–Statistic was significantly different from the experimental one. The same conclusion was reached when testing against the bootstrap sampling Monte Carlo simulation. Even worst, running the non–linear Granger test, proposed by Baek and Brock (1992), showed that sentiments had no significant impact on predicting the stock market returns.

As a final experiment, Olaniyan et al (2015) explored the predictive power of sentiment on volatility Q_t. They used an EGARCH lagged volatilities Q_{t-1} and Q_{t-3}, coupled with the *Positive* and *Negative* sentiment P_{t-1}, P_{t-2} and N_{t-1}, N_{t-2} as attributes variable into a feed–forward neural network (NN), an Elman recursive NN and an Elman recursive NN. The authors reached the conclusion that:

i) past volatility was the main contributor to predicting future volatility,
ii) positive sentiment was negatively correlated with the future volatility, and
iii) negative sentiment appeared to have the least influence on the stock volatility.

2.2.5 Current Approaches Limitations

2.2.5.1 The Bag–of–Words Approach

Although the accuracy results produced by Meesad and Li (2014) and Halgamuge (2007) looks very promising, they are some concerns with this overall approach:

- The utilisation of a Bag–of–words to express the sentiment of a tweet is limited by two main factors. As highlighted by Grishman (2014), some words or combination of words are ambiguous in their opinion. For example, 'low quality' represents a *Negative* opinion, however 'low price' corresponds to a *Positive* opinion. Furthermore, when opinion is expressed over a several stocks in the same tweet, the Bag–of–words cannot distinguish references to any particular stock.
- Lexicons such as *SentiWordNet* are limited when it comes to providing a sentiment likelihood related to financial specialist vocabulary. For example, the terms rally or uptick, expressing an upward price move, should represent a *Positive* sentiment. Independent Tests carried out on *SentiWordNet* API showed that the above words return neither a *Positive* nor a *Negative* sentiment (c.f. Appendix 1). Consequently, relying on non–domain specialised lexicon has the power to skew the overall predicted sentiment present in a document.
- The dataset under analysis is quite small; less than 5,000 tweets over an 11 months' period for Meesad and Li (2014) and a one–year period for Halgamuge (2007). This does not seem representative enough to ensure the model stability (i.e. low variance) over time.

Schumaker and Chen (2009) used a similar approach; mixing Bag–of–words from news articles and stock prices (at the time of the news release) as inputs into an SVM. The market direction test accuracy results approximated 57%, which is well below the 70%–80% accuracy rate presented by Meesad and Li (2014) and Halgamuge (2007). This demonstrates that they can be wide variation in accuracy results partly caused by the nature of the sentiment gathering process.

2.2.5.2 The Attribute Selection

All the machine learning approaches presented in the literature review only consider the impact of the sentiment, on the trend prediction, for small set of handpicked technical indicators. In the VAR statistical model, only the raw price and volume are being used. It would be interesting to investigate the sentiment prediction power on a more complete list of technical indicators. The main reason is that technical indicators seem so far to have the most significant impact. Therefore, selecting a technical indicator set that is too restrictive could generate over optimistic results when it comes to the sentiment true predictive power.

2.2.5.3 The Attribute Correlation and Multicollinearity

None of the papers in the literature review considers the problem of correlation of explanatory variable vs the response variable, or the issue of potential multicollinearity between explanatory variables. The presence of both these effects could have a potential to make the accuracy results unstable.

2.2.5.4 The Validation Set Method

James et Al (2015) state that the validation set approach is not an appropriate method in case of financial time series. Figure 3, borrowed from Rechenthin (2014), shows that there are moments of high and low predictability in the stock market. If such moments were contained under the same unique test set, then the predictability power would be reduced, as these moments would "average out".

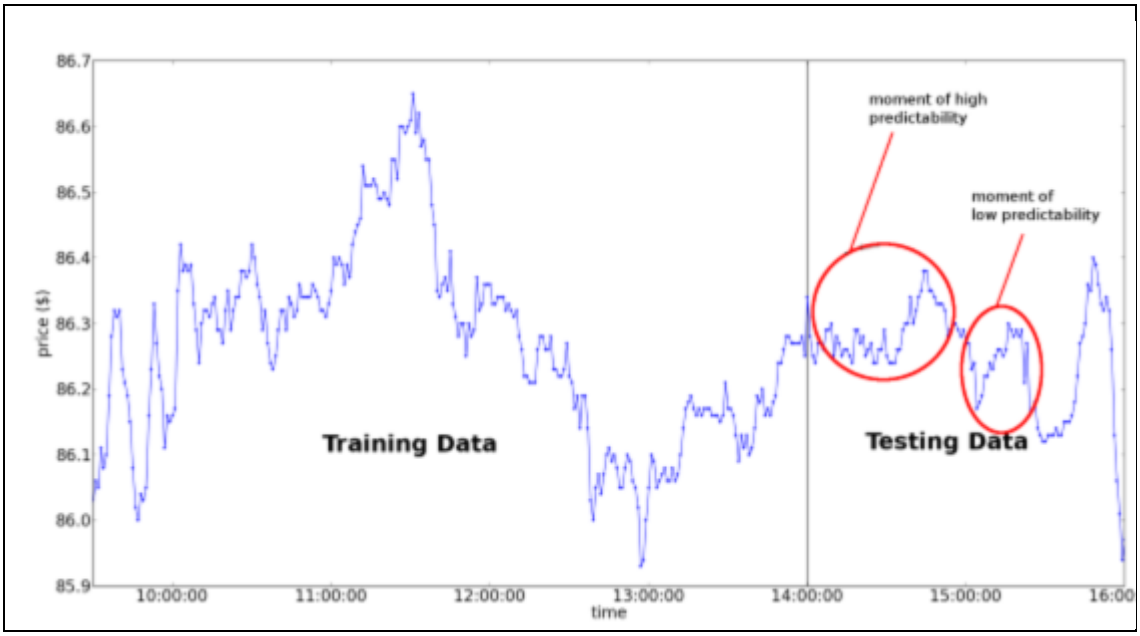


Figure 3 – Hold out period averaging predictability

2.2.5.4.1 The Cross–Validation Approach

Cross–validation, and more generally resampling methods are an essential part of the model performance assessment on test data (James et Al, 2015). Cross–validation is particularly useful when the test dataset is relatively small, as it computes an average of all test errors performed by the cross–validation process (see below for more details).

The test error is a generic term that can be applied to the validation set, in the context of model optimisation or a test set, when the model accuracy performance needs to be computed.

Cross–validation has a clear advantage over a validation set approach. In the later, the test set is selected randomly and it is used as the only test set. With cross–validation there is no ambiguity on which optimised model generate the smallest validation classification error. As a corollary, it reduces the variance relating to the model performance on test data.

James et Al (2015) and Kuhn (2013) describe the three main methodologies to perform cross–validation:

- The Leave–One–Out Cross–Validation (LOOCV) – A schematic representation is shown in Figure 4 below, borrowed from James et Al (2015). The whole data set (represented by the black rectangle) comprises *n* records. The original dataset is replicated *n* times (vertical split). At each split, the next record is held as the test set (pink segment). The rest corresponds to the training set (blue segment). The test error is obtained by generating the error rate for each split. At the end of the process, the average error is calculated over the *n* splits.

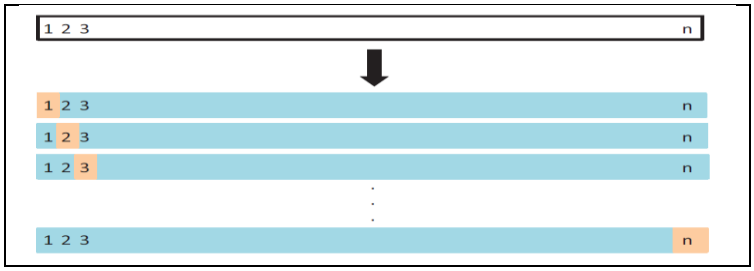


Figure 4 – The LOOCV procedure

- The k–Fold Cross–Validation – It is similar to the LOOCV but the number of splits is usually constrained to 5 or 10 and the test set represents a proportion of the training set. This is shown in Figure 5, borrowed from James et Al (2015). As before, at the end of the process, the average error is calculated over the *n* splits.

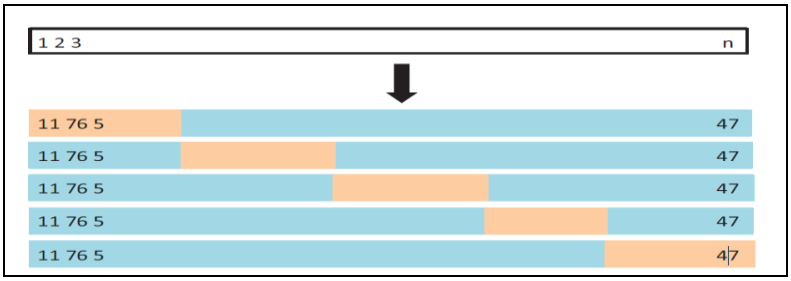


Figure 5 – The k–Fold CV procedure

- The Bootstrap – It is very similar to the k–Fold with replacement, which means some samples could be represented more than once, and others not at all. The later are called the *out–of–bag* samples.

All these methods are a definite improvement on the validation sample, as they reduce the variance (at the cost of increasing the bias). However, they are intrinsically incompatible with time–series prediction. Indeed, the path dependency nature of time–series forbid the leaking of future prices from the validation/test sets into the training set. Furthermore, financial assets price move, denoted $\Delta S_t = S_{t+\Delta t} - S_t$, are assumed to follow a *geometric Brownian motion* which takes three time–dependent parameters into consideration: i) the stock price S_t, ii) the mean rate of return of stock μ and iii) and the stock volatility σ (Schoutens, 2003). It is expressed as follows:

$\Delta S_t = S_t(\mu\Delta t + \sigma \Delta W_t)$, where: (2.21)
$S_t \mu\Delta t$ is the expected increase in S, and
$S_t \sigma \Delta W_t$ represents the random part of the return.
ΔW_t corresponds to the normally distributed noise term driving the stock price directions.

Instead, a sliding time windows should be the preferred solution for such experiment. The implementation of this method is described in detail in the 'Methodology' chapter, section 'Sliding Time Windows'.

2.3 Machine Learning Supervised Classification Algorithms

The aim of this section is to review the theoretical foundations of the machine learning supervised classification algorithms that have been selected for this experiment. This is not a fully exhaustive review of all existing approaches.

2.3.1 Linear Discriminant Analysis (LDA)

2.3.1.1 Model Assumptions

The model assumes that each attribute should follow a normal (Gaussian) distribution. When it is not, the case, then the LDA model can still be used. However, the user should be aware that the model results may be unstable. One way to resolve this issue, is to apply a Box–Cox transform on the non–normally distributed attribute prior to running the model (Kuhn and Johnson, 2013). This transformation should only be applied to the training data (i.e. neither the validation or test data).

2.3.1.2 Model Description

The LDA model was selected instead of the Logistic Regression as there are both very similar approaches. Both use the Bayes' theorem (directly for the Logistic Regression and indirectly for the LDA model). Furthermore, according to James et Al (2015). the Logistic Regression parameters estimate are unstable where classes are well–separated.
As show in (2.22), the Bayes' Theorem states that:

$$\Pr(Y = k|X = x) = \Pr(Y = k|X) = P_k(X) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{k} \pi_l f_l(x)} \quad (2.22)$$

π_k can be estimated by computing the fraction of the training observations that belongs to the k^{th} class. Depending on the value of p, i.e. when p= 1 or p> 1, $f_k(x)$ has a different form. The equation (2.23) shows the case for the univariate Gaussian density function. Whereas (2.24)

displays the case for a multivariate Gaussian density function. In the below equations, p denotes the number of explanatory variables. Furthermore, μ_k and σ_k represent respectively the mean and standard deviation of the k[th] class. In the multivariate case, μ and σ represent the mean and standard deviation of X, where $X = (x1,x2, ...,xn)$. In other words, X is a vector of predictors. This model assumes that "each individual predictor follows a one–dimensional normal distribution, with some correlation between each pair of predictors" James et Al (2015). Therefore, $X \sim N(\mu, \sum)$, where $E(X) = \mu$ is the mean of X and $Cov(X) = \sum$ is the covariance matrix of X.

$$When\ p = 1: \qquad (2.23)$$
$$f_k(x) = \frac{1}{sqrt(2\pi\ \sigma_k\ \sigma_k)} \exp\left(\frac{1}{-2\ \sigma_k\ \sigma_k}\ (x - \mu_k)^2\right)$$

$$When\ p > 1: \qquad (2.24)$$
$$f_k(x) = \frac{1}{(\pi)^{p/2}\ (|\sum|)^{1/2}} \exp\left(\frac{1}{-2}\ (x - \mu)^T\ (|\sum|)^{-1}\ (x - \mu)\right)$$

π_k and $f_k(x)$ are fed back into the Bayes' Theorem to perform the class prediction. This can be represented graphically as in Figure 6, borrowed from James et Al (2015).

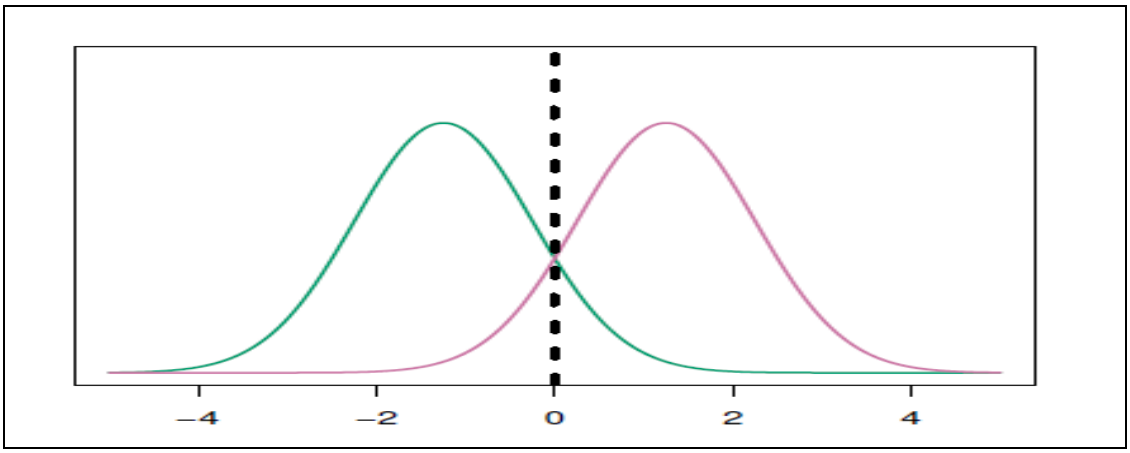


Figure 6 – LDA decision boundary (dotted line)

2.3.1.3 Strengths and Weaknesses

There is no parameter tuning, which reduces the compute power requirement and makes it a faster model to run. However, it has the potential to overfit more the data, compared to algorithms supporting regularisation (e.g. Support Vector Machine, Random Forest).

2.3.2 Quadratic Discriminant Analysis (QDA)

2.3.2.1 Model Assumptions

Like the LDA, the QDA assumes that each attribute class should follow a normal (Gaussian) distribution. For furthermore details, please refer to section 2.3.1 above.

2.3.2.2 Model Description

Like the LDA, QDA estimates the parameters feeding into the Bayes' Theorem parameters. However, the QDA assumes an individual covariance matrix per class (James et al, 2015). Consequently, an observation from the k^{th} class follows a normal distribution of mean μ_k and covariance matrix of $\sum_k$, i.e. $X \sim N(\mu_k, \sum_k)$.

2.3.2.3 Strengths and Weaknesses

Essentially the same as the LDA model, with the added benefits that the QDA has the potential to improve the bias compared to the LDA model (at the cost of a higher variance). This is produced by the estimation of separate covariance matrix for each class. Therefore, the QDA model is recommended when there is a large number of training observations. In this case, it will tend to have a lower bias, and the variance becomes less of an issue. When the number of training sample is small, then LDA might be preferred.

2.3.3 Penalized Discriminant Analysis (PDA)

2.3.3.1 Model Assumptions

Like the LDA, the QDA assumes that each attribute class should follow a normal (Gaussian) distribution. For furthermore details, please refer to section 2.3.1 above.

2.3.3.2 Model Description

Hastie et al (1995) described the PDA as being an extension to the LDA model, where the covariance matrix ($\sum$) is regularised as follows: $\sum + \lambda\Omega$, where Ω is a "roughness"–type penalty matrix. The rest of the LDA algorithm is untouched.

2.3.3.3 Strengths and Weaknesses

The PDA is particularly useful for overcoming the problem of high–dimensional correlated predictors.

2.3.4 Support Vector Machine (SVM)

2.3.4.1 Model Assumptions

The *SVM* model does not make assumption on the shape of the explanatory data, contrary to a *Regression* model for example, where the data is assumed to be normally distributed. However, it is recommended that each variable is scaled, i.e. having a standard deviation of one. Scaling removes the dominating effect of variable with larger values compared with variable with smaller values.

2.3.4.2 Review

As we are classifying two classes that are linearly separable, only the SVM with the linear *Kernel* was selected. The other *Kernel*s were not considered. As explained by James et Al (2015), the linear SMV is akin to a linear regression. Both try to find a hyperplane that separates two classes; one belonging to the positive class (i.e. the *Up* direction) and one belonging to the negative class (i.e. the *Down* direction). However, contrary to the Linear or Logistic Regression models, the aim is not to find the hyperplane that perfectly classifies all

the training data. On the contrary, the *SVM* model chooses the hyperplane that generates the largest gap (i.e. margin) between the two classes. This is illustrated in Figure 7, borrowed and modified from James et Al (2015). The hyperplane is delimited by the blue and pink margins. The hyperplane is represented by the linear equation of type $f(X) = \beta_0 + \beta_1 x_{1 +\ldots+} \beta_p x_p$, where $\beta 0$ is the intercept and the $\beta 1$, …., βp are the normal vectors that point in a direction orthogonal to the separating hyperplane line (c.f. Figure 7). The margin, called M, results from a constraint optimisation problem where each record *i*, in the training data, should satisfy the equation 2.25. The variable *C* represents the *cost*, i.e. a regularisation hyper–parameter that need to be optimised during the training/validation phase.

$$y_i (\beta 0 + \beta 1 x_{i1 + \ldots. +} \beta 1 x_{ip}) \geq M (1-\varepsilon_i), \quad (2.25)$$
$$\text{where } \varepsilon_i \geq 0 \text{ and } \sum_{i=1}^{n} \varepsilon i \leq C$$

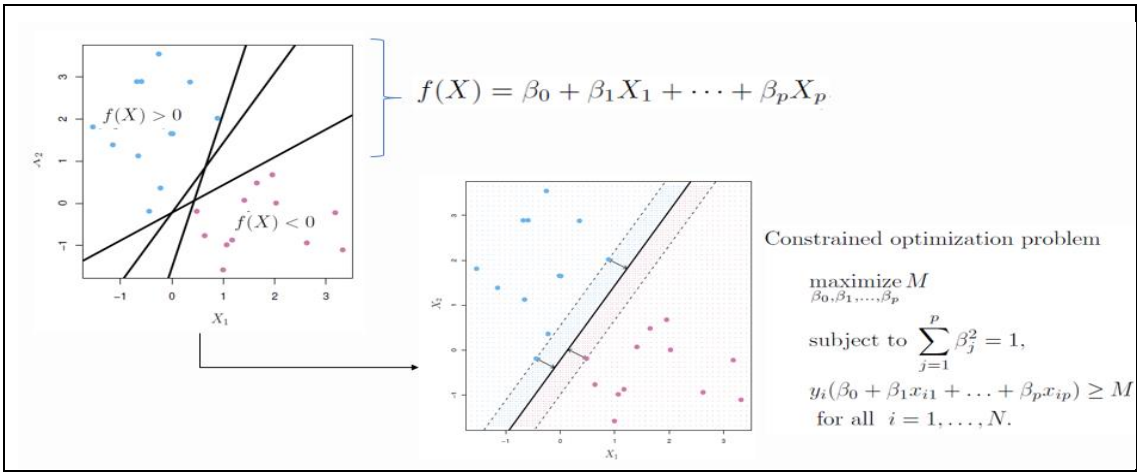


Figure 7 – The SVM linear model

2.3.4.3 Strengths and Weakness

The regularisation parameter, i.e. finding the optimal *cost* for the training data, enables the reduction of overfitting. However, as for most parameter based machine learning algorithms, there is a price to pay in term of potential misclassification of new training data. This is known as the Bias and Variance trade–off.

Furthermore, when the Kernel is linear, as this is the case in this experiment, the SVM does not suffer from increase memory and slowness as imposed with the polynomial Kernel for example.

2.3.5 Bagging and Random Forest (RF)

2.3.5.1 Model Assumptions

These models do not make any assumption about the shape of the data, contrary to Regression, SVM or Neural Network models.

2.3.5.2 Model Description

The building block of a RF model is a Tree model. The first section details the Tree model, then it reviews the Bagging model as a precursor for the RF model description.

2.3.5.2.1 The Tree Model

A indicated by James et al (2015), a tree is composed of a root node, that branches into nodes, which themselves branch into other nodes. A node that does not contain another node is called a leaf. The branch split is based on rules starting at the beginning of the tree. Tree algorithms may vary slightly. However, the process of building a classification Tree follows these main steps:

1. Assuming $X_1, X_2, \ldots, X_p$ represent the predictor space. The aim is to divide the predictor space into distinct (non–overlapping) regions $R_1, R_2, \ldots, R_p$.
   For classification Tree models, usually either i) the classification error rate (2.26), ii) the Gini index (2.27) or iii) the cross–entropy index (2.28) is calculated alongside a recursive binary split algorithm to identify the regions that minimise the classification error.

$$E = 1 - \max_k \hat{p}_{ik} \quad (2.26)$$

$$G = \sum_{k=1}^{n} \hat{p}_{ik} (1 - \hat{p}_{ik}) \quad (2.27)$$

$$D = -\sum_{k=1}^{n} \hat{p}_{ik} (\log \hat{p}_{ik}) \quad (2.28)$$

   where $\hat{p}_{ik}$ is the proportion of observations in region *i* that belongs to class *k*

2. A new observation is assigned to one of the region $R_1, R_2, \ldots, R_p$ based its attributes values $X_1, X_2, \ldots, X_p$
3. The class associated with a region R_j is defined by the most dominant observation class (i.e. voting system).
4. Tree pruning (i.e. cutting–off some of the leaf nodes) can be used to reduce overfitting.

2.3.5.2.2 The Bagging Model

Bagging is a procedure that reduces the variance of any statistical methods (James et al, 2015). In this case the classification Tree model, the aim is to generate B different bootstrapped training datasets. Bootstrapping is a resampling method that creates a number B of datasets containing the same schema of the original dataset. However, the rows are randomly selected (with replacement) from the original datasets. Once, the bootstrap is complete, then the classification Tree model is trained on each of the b^{th} bootstrap datasets. The classification Tree model prediction noted $\hat{f}_{bag}(x)$ corresponds to the most commonly occurring class (i.e. majority vote).

2.3.5.2.1 The Random Forest Model

Random Forests only differ from Bagging models by decorrelating the Tree models (James et al, 2015). The aim is to reduce the variance of the average Tree models. Bagging is a special case of Random Forest, where the number of variables randomly sampled (as candidates for each Tree model split) is equal to the number of attributes in the model, denoted m. In the case of the Random Forest, the splits are usually set to $\sqrt{m}$ or $\frac{m}{2}$.

2.3.5.3 Strengths and Weakness

Tree models can handle numerical and categorical variable. Tree models, coupled with their graphical presentation, are more closely aligned to problem domain representation. Consequently, they provide an intuitive way of interpreting results, James et al (2015).

2.3.6 Multilayer Perceptron with Weighted Decay (MLP)

2.3.6.1 Model Assumptions

The model requires the explanatory variables to be scaled and centred prior to be fed into any neural networks models as shown in (2.29).

$$z_i = (x_i - \min(x)) / \max(x) - \min(x), \quad (2.29)$$

This is to ensure the data is normalised in a range between 0 and 1.

2.3.6.2 Review

As explained by Haykin (1999) and Maciel and Ballini (n.d.), a fully connected Multilayer Perceptron (MLP) consists of a two layers network representation, as shown in Figure 8 borrowed and modified from Nikolaev (n.d.).

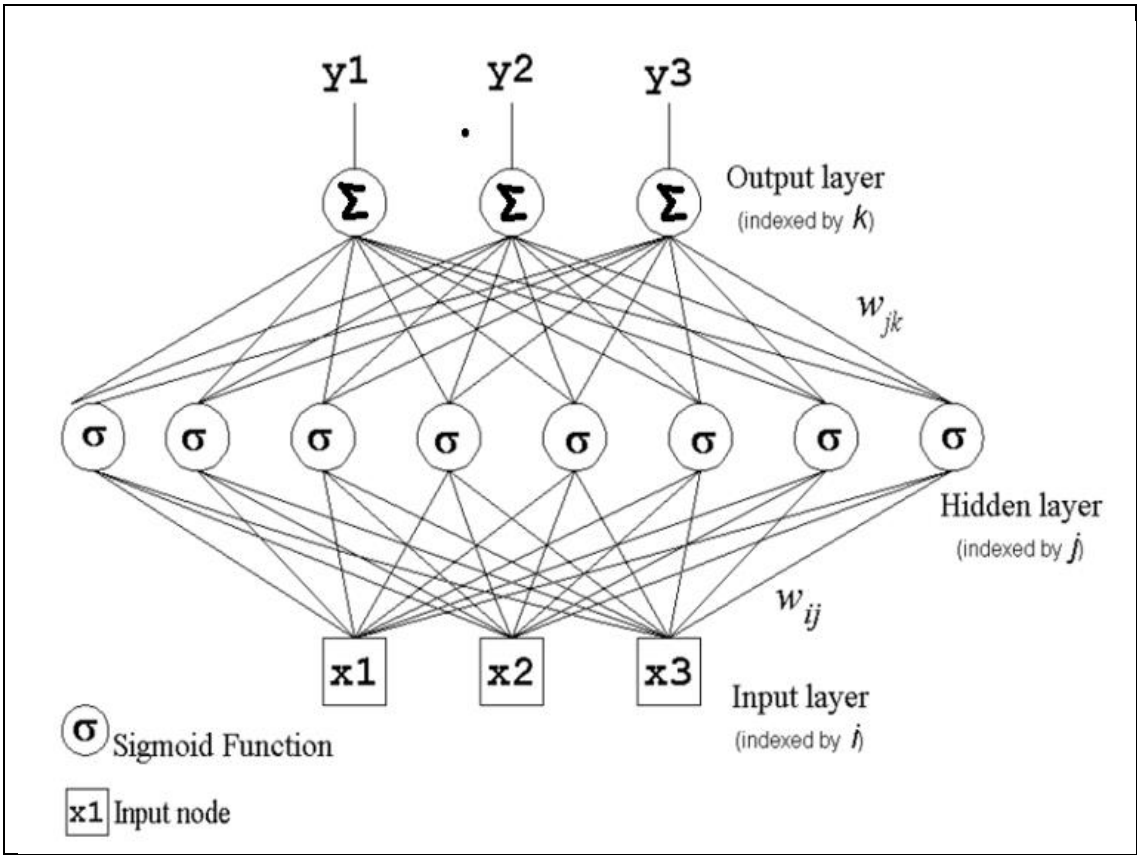


Figure 8 – Multilayer perceptron (MLP)

It starts from the input data (x_1,….., x_n) , i.e. there is one input per explanatory variable, which feed forward into the first layer, a.k.a. the *Hidden* layer. The *Hidden* layer consists of a set of nodes which individually computes the *Hidden* layer output by summing the product of each input by their respective weight, as shown in (2.30), and transforming the result via a function before passing it to the second layer. a.k.a. the *Output* layer. The function transformation usually takes the form of a sigmoid (or logistic function), as shown in (2.31), or a hyperbolic tangent function, a.k.a. tanh, as shown in (2.32).

$$s_j = \Sigma_{i=0}^{d} w_{ij} x_i \quad (2.30)$$

where w_i represents the weight for each input x_i feeding into the *Hidden* layer.

$$o_j = \sigma(s_j) = 1 / (1 + e^{-s_j}) \text{ for the sigmoid case.} \quad (2.31)$$

$$o_j = \tanh(s_j) = (e^{-s_j} - e^{-s_j}) / (e^{-s_j} + e^{-s_j}) \text{ for the tanh case.} \quad (2.32)$$

The output of each *Hidden* layer serves as input into the *Output* layer, by summing the product of the individual *Hidden* layer weights by their respective output, as show in (2.33), where

$$o_k = s_k \text{ , where } s_k = \Sigma_{i=0}^{d} w_{ik} o_j \quad (2.33)$$

where w_{ik} represents the weight for each *Hidden* layer o_j feeding into the *Output* layer.

Neural Networks "self–learn their weights optimisation". The "self–learn" underlying principle lies in the Backpropagation training algorithm. It was discovered independently by (Parker and LeCun cited in Haykin 1999). The algorithm is detailed in Algorithm 1, inspired from Nikolaev (n.d.) and it assumes no bias. The Learning Rate is critical for discovering the true global minimum of the error distance. The Decay penalises the weight change. It is helpful in speeding convergence, avoiding local minima and helps reduce overfitting.

$\{(\mathbf{x}_e, y_e)\}_{e=1}^{N}$, initial weights w_i set to small random values
Learning rate $0 \leq \eta \leq 1$
Decay $0 \leq \lambda \leq 1$
repeat
  for *training example* $(\mathbf{x}, y)$ **do**
    calculate the outputs at the hidden and output layer:
    $o_j = s(s_j) = 1 / (1 + e^{-s_j})$
    $o_k = s_k$
    compute the error derivatives b_k at the nodes k in the output layer:
    $b_k = o_k (1 - o_k) [y_k - o_k]$
    compute the changes for weights $j \rightarrow k$ on connections to nodes in the output layer:
    $\Delta w_{jk} = \eta b_k o_j - \lambda \eta w_{jk}$
    compute the error derivatives b_j for the hidden nodes j with the formula:
    $b_j = o_j (1 - o_j) [\sum_k b_k w_{jk}]$
    compute the changes for the weights $i \rightarrow j$ on connections to nodes in the hidden layer:
    $\Delta w_{ij} = \eta b_j o_i - \lambda \eta w_{ij}$
    update the weights by the computed changes:
    $w_{jk} = w_{jk} + \Delta w_{jk}$
    $w_{ij} = w_{ij} + \Delta w_{ij}$
until *termination condition is met*

Algorithm 1 – MLP *Backpropagation* algorithm (with no bias and no momemtum)

2.3.6.3 Strengths and Weaknesses

On the positive side, MLP models are highly tolerant to noisy data. On the negative side, MLP models are computationally expensive. The time to reach completion is dependent on the number of iterations as well as the number of nodes in the *Hidden* layer. Furthermore, there is no guarantee it will find the global optimum given the number of iterations. Finally, there are 'Black Box' algorithm, therefore they are not biased towards interpretation. However, in case of prediction, this is not an issue.

2.3.1 Elman/Jordan Recursive Neural Network (Elman RNN)

2.3.1.1 Model Assumptions

The assumptions are identical as the MLP model.

2.3.1.2 Model Description

As discussed by Lewis (2017), Elman and Jordan RNNs are specialised in learning from time–varying patterns. They are an extension on the MLP network, in that they contain an extra layer named the recurrent layer, as show in Figure 9 (borrowed and modified from Wang & Al (2016)). The difference between an Elman and a Jordan RNN lies in the fact that the recurrent layers respectively feed from a) the hidden layer or b) the output layer.

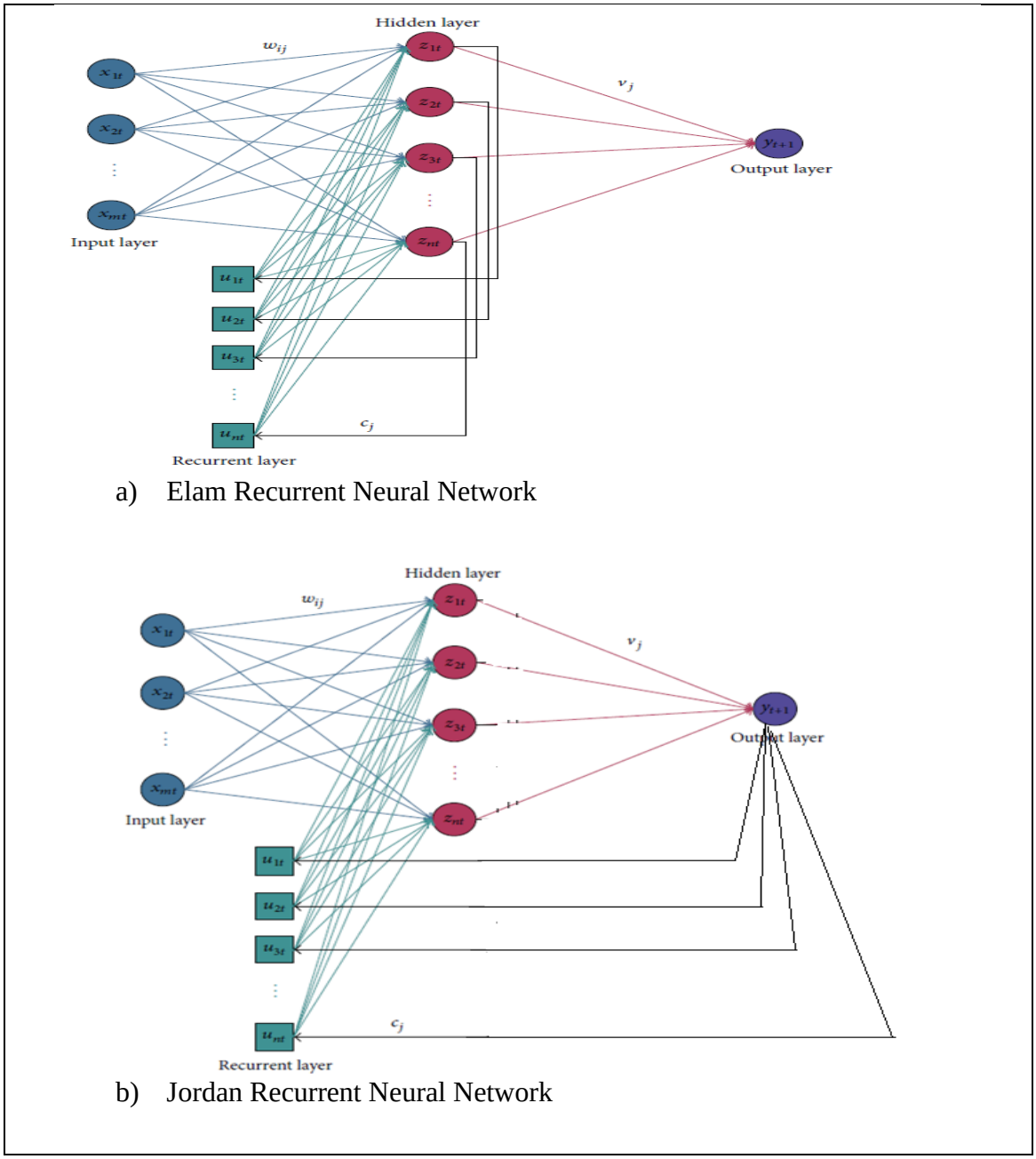


Figure 9 – Simple recurrent neural network

In this framework, the number of recurrent neurons equals to the number of hidden neurons. The recurrent unit provides a 'short–term memory' to the network. Hidden layer activation and/or output values are stored from the previous step t–1 and are consumed by the network at time t. The green lines show the recurrent fully connected layer with the *Hidden* layer. The black lines show the feedback from the hidden neurons to their respective recursive units. The *backpropagation* through time algorithm is detailed in Algorithm 2, inspired from Wang & Al (2016), Abdulkarim (2016) and Haykin (1999).

2.3.1.1 Strengths and Weaknesses

According to Wang & Al (2016), Elman and Jordan RNN strengths spans from supporting time series, having non–linear prediction capabilities coupled with a faster convergence, compared to MLP models.

xit (i = 1, 2, . . . , m) represents the set of input vector of neurons at time t,
yt+1 denotes the output of the network at time t + 1
zjt (j =1, 2, . . . , n) represents the output of hidden layer neurons at time t,
ujt (j = 1, 2, . . . , n) corresponds the recurrent layer neurons.
wij denotes the weight connecting the node i in the input layer neurons to the node j in the hidden layer.
cj represents the weights that connect the node j in the hidden layer neurons to the node in the recurrent layer
Vj represents the weights that connect the node j in the hidden layer neurons to the output

Repeat
for *training example (***x**, *y)* **do**
Calculate the output at the Hidden Layer $y_{t+1}(k)$

$$\text{net}_{jt}(k) = \sum_{i=1}^{n} w_{ij}x_{it}(k-1) + \sum_{j=1}^{m} c_j u_{jt}(k)$$

If Elman
$u_{jt}(k) = z_{jt}(k-1)$, where i = 1, 2, . . . , n and j = 1, 2, . . . , m.
Elseif Jordan
$u_{jt}(k) = y_t(k-1)$, where i = 1, 2, . . . , n
End if

$$z_{jt}(k) = f_H(\text{net}_{jt}(k)) = f_H\left(\sum_{i=1}^{n} w_{ij}x_{it}(k) + \sum_{j=1}^{m} c_j u_{jt}(k)\right), \text{ where } f_H = 1/1(1+e^{-x})$$

$$y_{t+1}(k) = f_T\left(\sum_{j=1}^{m} V_j z_{jt}(k)\right)$$

Generate the Error

$E(t_n) = 0.5 \times \sum (d_{tn} - y_t)^2$, where tn is the time of the sample n (n = 1, …N) and d_{tn} is the actual value

Calculate the *Backpropagation through time*

$\Delta w_{ij} = -\eta\, \partial E(t_n) / \partial w_{ij}$

$\Delta c_j = -\eta\, \partial E(t_n) / \partial c_j$

$\Delta V_j = -\eta\, \partial E(t_n) / \partial V_j$

$w_{ij}^{k+1} = w_{ij}^{k} + \Delta w_{ij}^{k}$

$c_j^{k+1} = c_j^{k} + \Delta c_j^{k}$

$v_j^{k+1} = v_j^{k} + \Delta v_j^{k}$

$$y_{t+1} = f_T\left(\sum_{j=1}^{m} V_j f_H\left(\sum_{i=1}^{n} w_{ij}xit + \sum_{j=1}^{m} c_j u_{jt}\right)\right)$$

until *termination condition is met*

Algorithm 2 – Simple RRN *Backpropagation* through time

Chapter 3 The Methodology

3.1 The Scope of the Experiment

The aim of this proposal is to establish whether the XLE index constituents' sentiments can improve the index trend prediction. The XLE stock constituent list is available in Appendix 2. The first step in the process consists of downloading historical price/volume information from *Yahoo!Finance,* as well as the sentiment data from *Quandl*. From the price information, the daily return is produced. This serves as the base for the generation of the categorical response variable, which takes two labels *Up* and *Down* (c.f. section 'The Response Variable'). The price/volume information is used to generate the technical indicators explanatory variables (c.f. section 'The Technical Indicators'). The sentiment information (S_t) is declined in lagged sentiment (S_{t-1}) as well as a sentiment momentum ($S_{t-}–S_{t-1}$) and its lags. The machine learning algorithms described above, i.e. the Support Vector Machine, the Random Forest, the Neural Networks and the Discriminant Analysis models support two–class classification predictions. They have therefore been selected as the supervised classification predictive models. Once the response and explanatory variables are built, the experiment's workflow is broken down in the following steps:

Step 1 – As detailed in the section 'Feature Selection and Extraction', this phase generates 16 (=15+1) feature selection lists. Each model is run against each feature selection list and the test accuracy rate is recorded at each time.

Step 2 – For each stock/index, the model (and its feature selection list) that produces the highest test accuracy rate is retained as part of the *Base* scenario.

Step 3 – As the sentiment is not available for the XLE index, a proxy for the index sentiment is constructed from the index constituent sentiments, named SIS_t[2]. The proxy index sentiment is simply the sum of the product of the sentiment score (SS_i) by the weight (W_i) of each stock, for each date t, as show in (3.1).

$$SIS_t = \sum_{i=1}^{n} (SS_i * W_i)_t \text{ , where n is the number of stocks} \qquad (3.1)$$

Step 4 – Numerous scenarios have been created involving the sentiment (S_t), the sentiment momentum ($S_{t-}–S_{t-1}$), and their respective lags. Each scenario produces a scenario accuracy rate, for each stock, named *SAi*.

Step 5 – A proxy index weighted accuracy rate (PIWA) is generated for the constituents, both for the base line and the scenarios. The proxy index weighted accuracy rate, defined in (3.2), is sum of the product of the stock weights (W_i) by the stock accurate rate (SA_i).

$$PIWA = \sum_{i=1}^{n} (SA_i * W_i), \text{ where n is the number of stocks} \qquad (3.2)$$

Step 6 – The index test accuracy rate, named IAR_{sent}, generated from technical indicators variables and the proxy sentiment (SIS_t), is compared to PIWA of step 5. If the experiment shows that the PIWA is greater than the IAR_{sent}, this means the XLE constituents' sentiment has extra predictive power.

[2] The proxy index sentiment data is available in the folder: ..\data\XLE\processed\sentiment.index

3.2 The Experiment Innovation

This experiment is innovative at two levels. First, it implements a 2–ways feature selection process, in order to optimise the stock base-line prediction test accuracy. In a nutshell, the best test accuracy for a given stock is produce by retaining the highest test accuracy rate across all models running vs two different feature lists (c.f. 'Feature Selection and Extraction').
Second, although the literature lists several studies relative to the technical indices predictive power of relatively small portfolio of stock, no study has been carried out on the impact of sentiment at an index level, containing more than 30 stocks. The remainder of this section details i) the data gathering process, ii) the pre–processing, the iii) 2–ways feature selection process and iv) the proposed trend prediction comparison.

3.3 The Experiment Environment

The code of this experiment was written in i) Python (3.4.x), used for web data scrapping, an ii) R (3.3.x) for the data exploration and machine learning. The Python code was developed in the *Jupyter Notebook* IDE and ran on a windows machine. The R code was developed under the RStudio IDE. It was then run on the Goldsmiths' Data Science dedicated Unix cluster servers (c.f. Appendices 3 and 4).

3.4 The Software High Level Design

To run this experiment, an infrastructure was built. It is composed of four main modules, namely i) the Data Source Service, ii) the Extract Transform Load (ETL) Service, iii) the Machine Learning Service and iv) the Information Management Service, as show in Figure 10. The infrastructure was engineered as a Python (green boxes) and R (blue boxes) solution which can on a PC or on a server (for further information please refer to Appendices 3 and 4). As it shown in Figure 10, the infrastructure offers plugins services and modules that can be added at the different layers of the pipeline. This provides extensibility and separation of concern between the components of the solution. A description of each service is provided in the next sections.

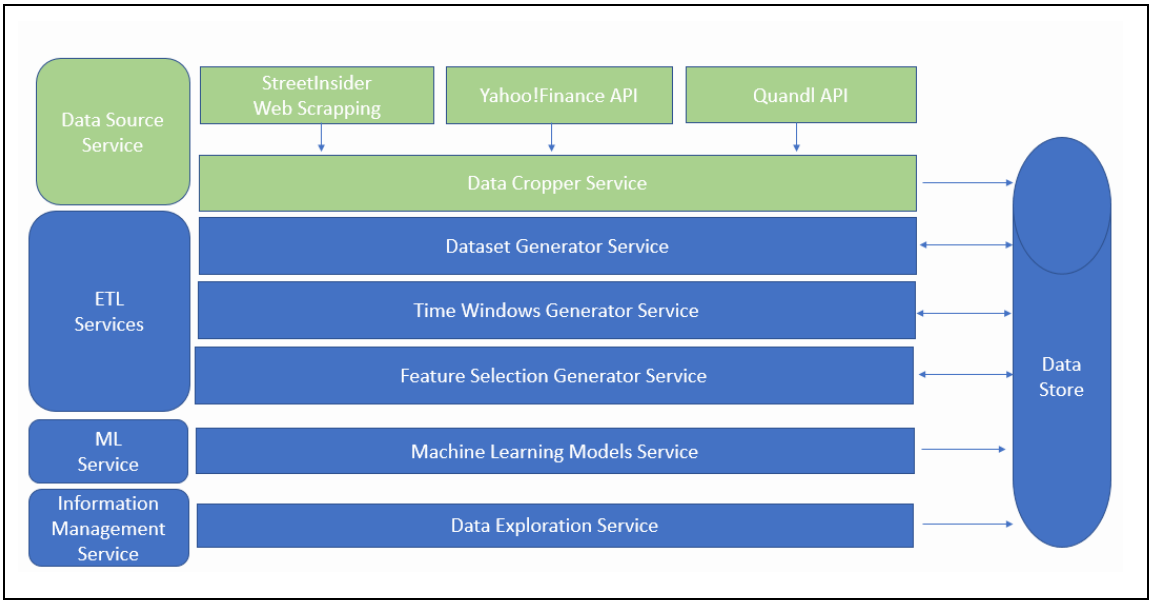


Figure 10 – The pipeline architecture

3.4.1 The Data Source Service

The *Data Cropper Service* connects i) to the *Yahoo!Finance API* (for historical price/volume information) and ii) to the *Quandl API (*for sentiment data*)* to download the raw data. This data is collected into *CSV* files, that are stored in a dedicated directory on a file system. The code is available in Appendix 5.

3.4.2 The Dataset Generator Service

It loads the raw market and sentiment data from the Data Store and generates one unique dataset per stock/index containing the technical and sentiment indicators. On completion, each stock/index dataset is stored back into the Data Store. Please refer to 'The Technical Indicators' section for further information on the indicators generation. Pseudocode 1 describes the high-level logic. The R code is available in Appendix 6.

```
Asset is a generic term that defines a stock constituent of the XLE or the XLE index itself.
repeat
  foreach asset do
    foreach date do

     #Step 1- Generate the explanatory variable
      data = Load_Asset_Market_Data(asset)                  # (contain the historical/volume information)
      Generate_Technical_Indicators (data, asset,date)      # (e.g. SMA 20 days , SMI, RSI 10days, etc.)
      Generate_Price_Lags(data, asset, date)                # (e.g. P_t-1, P_t-2, P_t-3, etc)
      Load_Sentiment_Data(data, asset)                      # (contain the historical sentiment scores)
      Generate_Sentiment_Lags (data, asset, date)           # (e.g. S_t-1, S_t-2, S_t-3, etc)
      Generate_Sentiment_Momentum (data, asset, date)       # (e.g. S_t-S_t-1, S_t-1-S_t-2, S_t-2-S_t-3, etc)
      Generate_Sentiment_Momentum (data, asset, date)

     #Step 2- Generate the response variable
      Generate_Daily_Log_Returns(data, asset, date)
      Generate_Trend_Direction (data, asset, date)

     #Step 3- Store the data to the DataStore
     Save (data)

    until there is no more date available
  until there is no more asset available
```

Pseudocode 1 – The dataset generator logic

3.4.3 The Time Window Generator Service

It generates a time sliding window matrix for each stock/index, and it stores it into the *Data Store*. As shown in Table 1a below, a 100 days' time-window is created (c.f. column *time_window_index* from 1 to 100)[3]. Each column represents the start/end indices of the training/validation and test data sets. In this example, the first time–window row shows that:

- The training set starts at index 8596 and ends at 8815. This represents 220 records.
- The validation set starts at index 8881 and ends at 8816. This represents 66 records.
- The test set starts at 8886 and ends at 8882. This represents 5 records.

test_end_index	test_start_index	validation_end_index	validation_start_index	training_end_index	training_start_index
8886	8882	8881	8816	8815	8596
8890	8886	8885	8820	8819	8600
8894	8890	8889	8824	8823	8604
8898	8894	8893	8828	8827	8608

Table 1a – A Time Window Snapshot

The pseudo code for the generation of the time-window is provided below. The R code is available in Appendix 7.

```
Asset is a generic term that defines a stock constituent of the XLE or the XLE index itself.
test_set_bucket = 10
validation_set_bucket = 66
training_set_bucket = 220
number_sliding_windows = 100
slid_win_idx = 0
data = new DataFrame()                    # it is the data frame containing the time slices
repeat
  foreach asset do
    while (slid_win_idx < number_sliding_windows) {
      test_end_index = max_index     #max_index is highest index available in the file
      test_start_index = test_end_index - test_set_bucket + 1
      validation_end_index = test_start_index - 1
      validation_start_index = validation_end_index - validation_set_bucket + 1
      training_end_index = validation_start_index - 1
      training_start_index = training_end_index - training_set_bucket + 1
      #Add the start/end indices for the training/validation and test datasets
      add_row(data, test_end_index, test_start_index, validation_end_index, validation_start_index,
              training_end_index, training_start_index)
    }

    #Store the data to the DataStore
    Save (data)

  until there is no more asset available
```

Pseudocode 2 – The time window generator logic

[3] The time window data for each asset is available in the folder ..\data\XLE\processed\time.windows\100.

3.4.4 The Feature Selection Service

This service enables a two-way feature selection based on two different competing methodologies, a Wrapper and a Filter feature selection. The full detail of the methodology, the rational behind this idea, and logic is available in section 'Feature Selection and Extraction'. The R code is available in Appendix 8 and Appendix 8bis.

3.4.4.1 The Machine Learning Models Service

The prediction step is supported by the *Machine Learning Model Service*. It encapsulates a suite of machine learning algorithms specialised in supervised classification tasks, such as Random Forest, Support Vector Machine, Neural Networks, etc. The below boxes contain the pseudocode, which loosely follow the R/Caret implementation logic. The full R models' implementation is available in Appendix 9.

```
training_case = true      #this is settable by the user
training_data             #this is the training_data used for the prediction
the_method = pda          #it could be set to pda/lda or qda model

If training_case == true
   the_lambda =c(0,0.001,0.001,0.005,0.1,0.5,1)
Else
    #the_lambda originates from the optimisation phase on the validation set
End if

training_data = box_cox(training_data)    #the training data is transformed so that the shape follows a
                                          #normal distribution

#Generate the best fit based on the training data, the method (e.g. pda) and a list of lambdas.
fit = generate_fit( training_data,
                    method = the_method,
                    grid = the_lambda )

#Generate a prediction based on the fit against the test data
pred = predict(fit, test_data)

#Generate the confusion table
confusion_martix = generate_confusion_table(pred)
```

Pseudocode 3 – The PDA/LDA/QDA prediction generation logic

```
training_case = true      #this is settable by the user
training_data             #this is the training_data used for the prediction
the_method = parRf        #this is the parallel Random Forest model
the_ntree = 500           #this is the number of trees in the forest

If training_case == true
   p = get_attribute_list(training_data)   #gets the number of attributes (i.e. explanatory variables)
   the_mtry=c(p, sqrt(p), p/2)             #when mtry= p this is the bagging case
Else
    # the_mtry originates from the optimisation phase on the validation set
End if

#Generate the best fit based on the training data, the method (e.g. pda) and a list of lambdas.
fit = generate_fit( training_data,
                    method = the_method,
                    grid = the_mtry )

#Generate a prediction based on the fit against the test data
pred = predict(fit, test_data)

#Generate the confusion table
confusion_martix = generate_confusion_table(pred)
```

Pseudocode 4 – The random forest prediction generation logic

```
training_case = true      #this is settable by the user
training_data             #this is the training_data used for the prediction
the_method = svm          #this is the SVM model
the_kernel = linear       #this is SVM kernel

If training_case == true
   the_cost_list = c(0.001 , 0.01 , 0.1, 1 ,100 )  #this is the list of cost to define the width of the margin
Else
    # the_cost_list originates from the optimisation phase on the validation set
End if

training_data = center(training_data)    #center the data
training_data = scale(training_data)     #scale the data

#Generate the best fit based on the training data, the method (e.g. pda) and a list of lambdas.
fit = generate_fit( training_data,
                    method = the_method,
                    grid = the_cost,
                    kernel = the_kernel)

#Generate a prediction based on the fit against the test data
pred = predict(fit, test_data)

#Generate the confusion table
confusion_martix = generate_confusion_table(pred)
```

Pseudocode 5 – The SVM prediction generation logic

```
training_case = true            #this is settable by the user
training_data                   #this is the training_data used for the prediction
the_method = nnnet              #this is the mlp model
the_max_iteration = 1000        #set the maximum number of iterations

If training_case == true
    the_size = seq(1:15)            #the hidden layer size
    the_decay=c(0,0.001,0.1,1)      #the weight decay list
Else
    # the_size and the_decay originate from the optimisation phase on the validation set
End if

training_data = center(training_data)     #center the data
training_data = scale(training_data)      #scale the data

#Generate the best fit based on the training data, the method (e.g. pda) and a list of lambdas.
fit = generate_fit( training_data,
                    method = the_method,
                    grid = (the_size,the_decay),
                    max_iter = the_max_iteration)

#Generate a prediction based on the fit against the test data
pred = predict(fit, test_data)

#Generate the confusion table
confusion_martix = generate_confusion_table(pred)

#Save
Save(confusion_martix)
```

Pseudocode 6 – The MLP with weight decay prediction generation logic

```
training_case = true            #this is settable by the user
training_data                   #this is the training_data used for the prediction
the_method = jordan             #it could be set to the Jordan or elman model

If training_case == true
   the_size = c(5,7,10,15,20)                       #the hidden layer size
   the_iterations = c(100,500,1000,1500,2000)       #the number of iterations
   the_decay=c(0,0.001,0.01,0.1,1)                  #the weight decay list
Else
    # the_size, the_decay and the the_iterations originate
    # from the optimisation phase on the validation set
End if

training_data = center(training_data)    #center the data
training_data = scale(training_data)     #scale the data

#Generate the best fit based on the training data, the method (e.g. pda) and a list of lambdas.
fit = generate_fit(  training_data,
                     method = the_method,
                     grid = (the_size,the_decay),
                     max_iterations = the_iterations)

#Generate a prediction based on the fit against the test data
pred = predict(fit, test_data)

#Transform the prediction (a positive or negative number between -1 and 1) into a Up or Down class
pred = transform_pred_into_class(pred)

#Generate the confusion table
confusion_martix = generate_confusion_table(pred)

#Save
Save(confusion_martix
```

Pseudocode 7 – The recurrent neural network prediction generation logic

3.4.1 The Data Exploration Service

The last component, named the *Data Exploration Service* implements numerous functions supporting data discovery and analysis (e.g. analysis of missing values, statistical tests, etc.). As this is code not centric to the problem under analysis, the pseudo code is no shown. However, R code in Appendix 10.

3.5 The Data Collection

3.5.1 Raw Data

The raw data[4] was downloaded from specialist sites, via a custom–made Python program (c.f. Appendix 5) through their respective API. The data relates to two types of explanatory variables:

- The market data:
  The XLE index and its constituent historical price and volume were downloaded from the *Yahoo!Finance* API. It contained the following fields:
  - Date: the date of the record.
  - *Open*: the index/stock price at the market opening time.
  - *High*: The highest price for a given period.
  - *Low*: The lowest price for a given period.
  - *Adjusted Close*: the daily closing price amendment (i.e. adjustment). This is the close stock price including any distributions and corporate actions that occurred at any time prior to the next day's opening.
  - *Close*: the close price for a given period. It does not contain any adjustments.
  - *Volume*: the amount of stock exchanged for a given period.

  The market data is available from the date the stock was launched, a.k.a. *Initial Public Offering* (IPO) date to 14–Dec–2016.

- The sentiment data:
  This data was downloaded from the *Quandl* API. As sentiment data is concept that emerged recently, only the five years of data is available. It contains the following information:
  - Article Sentiment: it is the average sentiment of all the articles (from last 24 hours) related to the asset. The value varies between –1 and 1. As stated in the Accern (2009), the higher the score, the more positive is the outlook.
  - Impact Score: it establishes "whether an event has a chance of affecting the stock price of that company by more than 1% (close price – open price / open price) at the end of the trading day" (Accern, 2009).

At this point, it is worth discussing the advantage of using *Quandl* as a choice for a sentiment data source, versus creating a sentiment parser from scratch, as prescribed in the literature. As mention in Accern (2009), *Quandl* collect content over 20 million news and blog sources real–time. They retain the relevant articles and extrapolate the sentiment. Although the sentiment generation is a black box, the Accern documentation indicates that the articles' sentiment is generated from a three steps process:

- First, deep learning is applied to the article to generate the sentiment.
- Second, a bag–of–words strategy is used to weight article keywords positively or negatively.
- Third, n–grams are deployed to identify sequences of text relating to specific sentiment. In this case, context is taken into consideration. The Accern engine can distinguish between sentences such as "Apple gets into a major lawsuit" and "Apple won a lawsuit". As a result, it provides a different sentiment.

[4] The raw data is available in the folder ..\data\XLE\raw

At the end of the process, a linear weighted average is applied over the three above steps to generate a sentiment between –1 and +1.
The Accern engine implementation is therefore superior i) in terms of volume of articles under analysis and ii) the sentiment algorithm generation seem more mature than the ones proposed in the literature review.

3.5.2 Data Pre–processing

This section reviews all steps involved in the production of the data used in this experiment[5].

3.5.2.1 The Response Variable

Predicting the index/stock market trend means predicting the daily index/stock return (a.k.a. return). As the return is assumed to be log–normally distributed (Schoutens, 2003), the return at date t, R_t is defined as follows: $R_t = \log(Close_{t+1}/ Close_t)$. This study focuses on predicting two categorical market moves: *Up* and *Down*, not the return value, for a date t. Therefore, the supervised classification task requires the response variable to be dummified as defined in (3.3). The reason why zero returns are bucketed in the *Down* trend category is explained in section 'Class Rebalancing'.

$$\begin{cases} Return_t > 0 \text{ then } Direction\ is\ set\ to\ Up \\ Return_t \leq 0 \text{ then } Direction\ is\ set\ to\ Down \end{cases} \quad (3.3)$$

3.5.2.2 The Explanatory Variables

The explanatory variables consist of three category types: i) the price/volume information and the price lags, ii) the technical indicators (index derived from the historical prices and volume), and iii) the sentiment data and its lags. All explanatory variables are of numeric and continuous types. The remainder of this section details the data transformation to generate the predictor variables.

3.5.2.2.1 The Price and Volume

As the close price and volume are known to be important market signals for investors, the close price at date t, $Close_t$ and its five days lags, namely $Close_{t-1}$, $Close_{t-2,\ldots}$, $Close_{t-3}$ as well as the volume at date t, $Volume_t$ are added to attribute list.

3.5.2.2.2 The Technical Indicators

The goal of technical analysis is to identify patterns in stock market time series by extracting noisy data. As the aim of this experiment is to predict the trend, a targeted subset of available technical indicators described by Murphy (1986) has been selected. The details of their usage and implementation is available in the remainder of this section. The below definition and formulas have been leveraged from Murphy (1986), Rechenthin (2014) and the Indicator Reference (2016). As a general note, the values of *n* listed below, i.e. the number of lagged days from the current date t, are the values used in this experiment.

[5] The processed data is available in ..\data\XLE\processed\final

Momentum Indicators

- Rate Of Change (ROC) – It is calculated as the percentage change of the current closing price at *t* from the close price at *t–n*, where *n*= 1,2,5,10,20

$$ROC_{(n)t} = (Close_{t} - Close_{t-n}) / Close_{t-n} \quad (3.4)$$

- Momentum– It shows the difference between the current closing price at *t* and the close price at *t–n*, where *n*= 5,10,20.

$$Momemtum_{(n)t} = Close_{t} - Close_{t-n} \quad (3.5)$$

- Momentum Absolute– It displays the absolute difference between the current closing price at *t* and the close price at *t–n*, where *n*= 5,10,20.

$$Momemtum_{(n)t} = |Close_{t} - Close_{t-n}| \quad (3.6)$$

- Williams %R (WR) – It is a momentum indicator that measures overbought asset, when the level is between 0 and –20. On the contrary, an oversold asset is when the level is between –80 and –100. An overbought (oversold) corresponds a potential buy (sell) signal.

$$HighestHigh(n)t = \text{highest high covering t – n instances} \quad (3.7)$$

$$LowestLow(n)t = \text{lowest low covering t – n instances} \quad (3.8)$$

$$WR(n)t = -100 \times \frac{(HighestHigh(n)t - Closet)}{(HighestHigh(n)t - LowestLow(n)t)} \quad (3.9)$$

where n= 5,10,20

- Relative Strength Index (RSI) – It intends to establish the strength or weakness of a stock/index by calculating the ratio of the average gain over previous periods *n*. Then it divides it by the average loss over the previous *n* periods. Here *n* is set to 5,10,14,20.
  RSI is bounded between 0 and 100, with values greater than 70 indicating an overbought stock/index and values below 30 indicating an oversold stock/index.

$$RS_{(n)t} = \frac{(\text{average gain over t – n periods})}{(\text{average loss over t – n periods})} \quad (3.10)$$

$$RSI_{(n)t} = 100 - (100 / (1 + RS_{(n)t})) \quad (3.11)$$

- Stochastic Momentum Oscillator (SMO) – It indicates the position of each day's close relative to the high/low range over the previous *n* instances, where *n*=14.

$$\text{HighestHigh}_{(n)t} = \text{highest high covering t – n instances} \quad (3.12)$$

$$\text{LowestLow}_{(n)t} = \text{lowest low covering t – n instances} \quad (3.13)$$

$$\%K = K_{(n)t} = 100 \times \frac{(\text{Closet} - \text{LowestLow(n)t})}{(\text{HighestHigh(n)t} - \text{LowestLow(n)t})} \quad (3.14)$$

$$\%D = \text{Stoch}_{(n)t} = \left(\frac{1}{n}\right) * \sum_{t=1}^{n} K_{(n)t-1} \quad (3.15)$$

And with:
Fast Stochastic refers to both %K and %D where %K is unsmoothed
Slow Stochastic refers to both %K and %D where %K is smoothed
Raw %K represents the un-smoothed %K
Fast %K represents un-smoothed %K
Slow %K is the smoothed %K
Fast %D corresponds to the moving average of an un-smoothed %K
Slow %D corresponds to the moving average of a smoothed %K

- Stochastic Momentum Indicator (SMI) – It is very close in concept to the SMO. The SMI shows the position of each day's close relative to the midpoint of the high/low range over the previous *n* instances, where *n*=13.

$$\text{HighestHigh}_{(n)t} = \text{highest high covering t – n instances} \quad (3.16)$$

$$\text{LowestLow}_{(n)t} = \text{lowest low covering t – n instances} \quad (3.17)$$

$$cm_{(n)t} = \text{Close}_t - ((\text{HighestHigh}_n - \text{LowestLow}_n)/2) \quad (3.18)$$

$$hl_{(n)t} = \text{HighestHigh}_n - \text{LowestLow}_n \quad (3.19)$$

$$cm_{(n)t} = \text{EMA}(\text{EMA}((cm_n)) \quad (3.20)$$

$$hl_{(n)t} = \text{EMA}(\text{EMA}((hl_n)) \quad (3.21)$$

$$SMI_{(n)t} = 100 \times (cm_{(n)t} / (hl_{(n)t}/2)) \quad (3.22)$$

$$SMI_{signal} = \text{EMA(SMI)} \quad (3.23)$$

- Chande Momentum Oscillator (CMO) – It is an attempt to signal overbought, greater than 50, and oversold, less than –50, stock/index positions.

$Up_{(n)t} = \sum_{i=1}^{n}(Close_t - Close_{t-i})$, on the "up" intervals (3.24)

$Down_{(n)t} = \sum_{i=1}^{n}(Close_{t-i} - Close_t)$, on the "down" intervals (3.25)

$CMO_{(n)t} = 100 \times (Up_{(n)t} - Down_{(n)t}) / (Up_{(n)t} + Down_{(n)t})$ (3.26)

Where n = 5,10 and 20

Trend Indicators

- Simple Moving Average (SMA) – Each output value is the average of the previous *n* day values. In this experiment n is set to 20,50,100,200. As each value in the period carries equal weight, it makes it less responsive to recent changes.

$$SMA_{(n)t} = \frac{\sum_1^n Close_t}{n} \quad (3.27)$$

- Exponential Moving Average (EMA) – It is a cumulative calculation, including all data (even values outside of the period). More recent values have a greater contribution to the average, past values have a diminishing contribution.

$EMA_{(n)t}$: {$Close_t - EMA_{(n)t-1}$} x multiplier + $EMA_{(n)t-1}$ (3.28)

Where:
  - multiplier is (2 / (n + 1)), and
  - *n is* the number of lag days for the moving average (20,50,100,200)

- Moving Average Convergence/Divergence (MACD) – It represents the difference between a short–term moving average and a longer–term moving average. A signal line, the exponential moving average of the MACD indicates when to buy or sell. When the MACD crosses below the signal, a buy signal is produced and reverse. The $DiffMACDSignal_{(n1,\ n2,\ n3)t}$ can also be used: a positive (negative) value indicate signals a potential buy (sell).

$$ShortEMA_{(n1)t} = EMA \text{ of size } n1 \quad (3.29)$$

$$ShortEMA_{(n1)t} = EMA \text{ of size } n1 \quad (3.30)$$

$$LongEMA_{(n2)t} = EMA \text{ of size } n2 \quad (3.31)$$

$$MACD_{(n1,\ n2)t} = ShortEMA_{(n1)t} - LongEMA_{(n2)t} \quad (3.32)$$

$$SignalLine_{(n1,\ n2,\ n3)t} = EMA(MACD_{(n1,\ n2)\ t}) \text{ of size } n3 \quad (3.33)$$

$$DiffMACDSignal_{(n1,\ n2,\ n3)t} = MACD_{(n1,\ n2)t} - SignalLine_{(n1,\ n2,\ n3)t} \quad (3.34)$$

Where n1, n2 and n3 are respectively set to 12, 26 and 9

- Parabolic Stop And Reverse (SAR) – It indicates that the stock/index should be sold when the price crosses the SAR.

$$\text{If long and high} > xp \text{ then } xp = high \text{ and } af = af + step \quad (3.35)$$

$$\text{If short and low} < xp \text{ then } xp = low \text{ and } af = af + step \quad (3.36)$$

$$SAR_t = (xp - SAR_{t-1}) \times af + SAR_{t-1} \quad (3.37)$$

Where xp and af (defaulted to 0.02) are respectively the extreme point and the accelerated factor.

Volatility

- Average True Range (ATR) – It measures volatility. A high (low) ATR indicates a high (low) volatility.

$$TrueHigh_t = \text{Highest of } high_0 \text{ or } close_{t-1} \quad (3.38)$$

$$TrueLow_t = \text{Lowest of } low_0 \text{ or } close_{t-1} \quad (3.39)$$

$$TR_t = TrueHigh_t - TrueLow_t \quad (3.40)$$

$$ATR_{(n)t} = (ATR_{t-1} \times (n-1) + TR_t) / n \quad (3.41)$$

Where n = 14

- Volatility – The is the close 10–days annualised volatility (where the number of days per year is to 260).

$$\sigma_{(n)t} = \sqrt{\frac{N}{n-2} \sum_{t=1}^{n-1} (r_t - avg(r))^2} \quad (3.42)$$

$$r_i = \log(\frac{Ct}{Ct-1}) \quad (3.43)$$

$$avg(r) = \frac{\sum_{t=1}^{n-1} r_t}{(n-1)} \quad (3.44)$$

Volume

- Money Flow Index (MFI) – It generates a buying and selling signal using the HLC previous price as well as the volume information. The value generated by $MoneyFlowIndex_{(n)t}$ is bounded between 0 and 100. 80 indicating an overbought signal and 20 indicating an oversold signal.

$$TypicalAvgPrice_{(n)t} = \left(\frac{1}{n}\right) \sum_{i=1}^{n} \frac{high_{t-i} + low_{t-i} + close_{t-i}}{3} \quad (3.45)$$

$$TypicalAvgVolume_{(n)t} = \left(\frac{1}{n}\right) \sum_{i=1}^{n} volume_{t-i} \quad (3.46)$$

$$MoneyFlow_{(n)t} = TypicalAvgPrice_{(n)t} \times TypicalAvgVolume_{(n)t} \quad (3.47)$$

$$PositiveMoneyFlow_{(n)t} = \sum_{i=1}^{n} (when\ MoneyFlow_{(n)t-i} \geq MoneyFlow_{(n)t-i-1}) \quad (3.48)$$

$$NegativeMoneyFlow_{(n)t} = \sum_{i=1}^{n} (when\ MoneyFlow_{(n)t-i} < MoneyFlow_{(n)t-i-1}) \quad (3.49)$$

$$MoneyRatio_{(n)t} = \frac{PositiveMoneyFlow(n)t}{NegativeMoneyFlow(n)t} \quad (3.50)$$

$$MoneyFlowIndex_{(n)t} = 100 - \frac{100}{(1 + MoneyRatio_{(n)t})} \quad (3.51)$$

where n = 14

Others

- *Bollinger Band (BB)* – This indicator contains a middle, upper and lower band. It indicates overbought and oversold conditions. When the price is close to the upper or lower band, it signals an imminent change in direction.

$$\text{MiddleBand}_{(n)t} = \left(\frac{1}{n}\right) \sum_{i=1}^{n} \frac{high_{t-i} + low_{t-i} + close_{t-i}}{3} \quad (3.52)$$

$$\sigma_{(n)t} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \frac{high_{t-i} + low_{t-i} + close_{t-i}}{3} - \text{MiddleBand}_{(n)t}} \quad (3.53)$$

$$\text{UpperBand}_{(d,n)t} = \text{MiddleBand}_{(n)t} + (sd \times \sigma_{(n)t}) \quad (3.54)$$

$$\text{LowerBand}_{(d,n)t} = \text{MiddleBand}_{(n)t} - (sd \times \sigma_{(n)t}) \quad (3.55)$$

$$\text{BBand}_{(n)t} = \begin{cases} close_t > \text{UpperBand}_{(d,n)t} : \text{"overbought"} \\ close_t < \text{UpperBand}_{(d,n)t} : \text{"oversold"} \\ else : \text{"do nothing"} \end{cases} \quad (3.56)$$

Where n = 20, sd =2

3.5.2.2.3 The Sentiment Indicator

For this experiment, the following sentiment variables are added:

- The sentiment at date t (S_t) and its three days lags, namely S_{t-1}, S_{t-2}, S_{t-3}
- The sentiment momentum (SM_t), and the sentiment momentum three days lags, namely SM_{t-1}, SM_{t-2}, SM_{t-3}. The formula the sentiment momentum is shown below:

$$SM_{(n)t} = S_t - S_{t-n} \quad (3.57)$$

Where n =1,2,3

3.5.2.2.4 The EGARCH Volatility generation

The exponential (GARCH) model is one of the numerous generalised autoregressive conditional heteroskedasticity (GARCH) models that can be used to forecast the volatility Brownlees et Al (2012). The GARCH(1,1) models (Engle,1982 and Bollerslev,1986 cited in Brownlees et Al, 2012, p.7) follow the process described below:

$$h_{t+1} = w + \alpha r^2 + \beta h_{t,} \quad (3.58)$$

where $(\alpha+\beta < 1)$ and

h_{t+1} represents the volatility forecast.

Equation (3.58) implies that the process is mean reversing. This is imposed by $(\alpha+\beta < 1)$. It also indicates that the amplitude of past returns influences past volatilities.

The exponential EGARCH (Nelson, 1991 cited in Brownlees et Al, 2012, p.7) models the log variance as follows:

$$\ln(h_{t+1}) = w + \alpha(|\varepsilon_t| - E(|\varepsilon_t|)) + y\,\varepsilon_t + \beta h_{t,} \quad (3.59)$$

where $\varepsilon_t = r_t / \sqrt{h_t}$

3.6 The Experimental Framework

3.6.1 The Choice of the XLE index

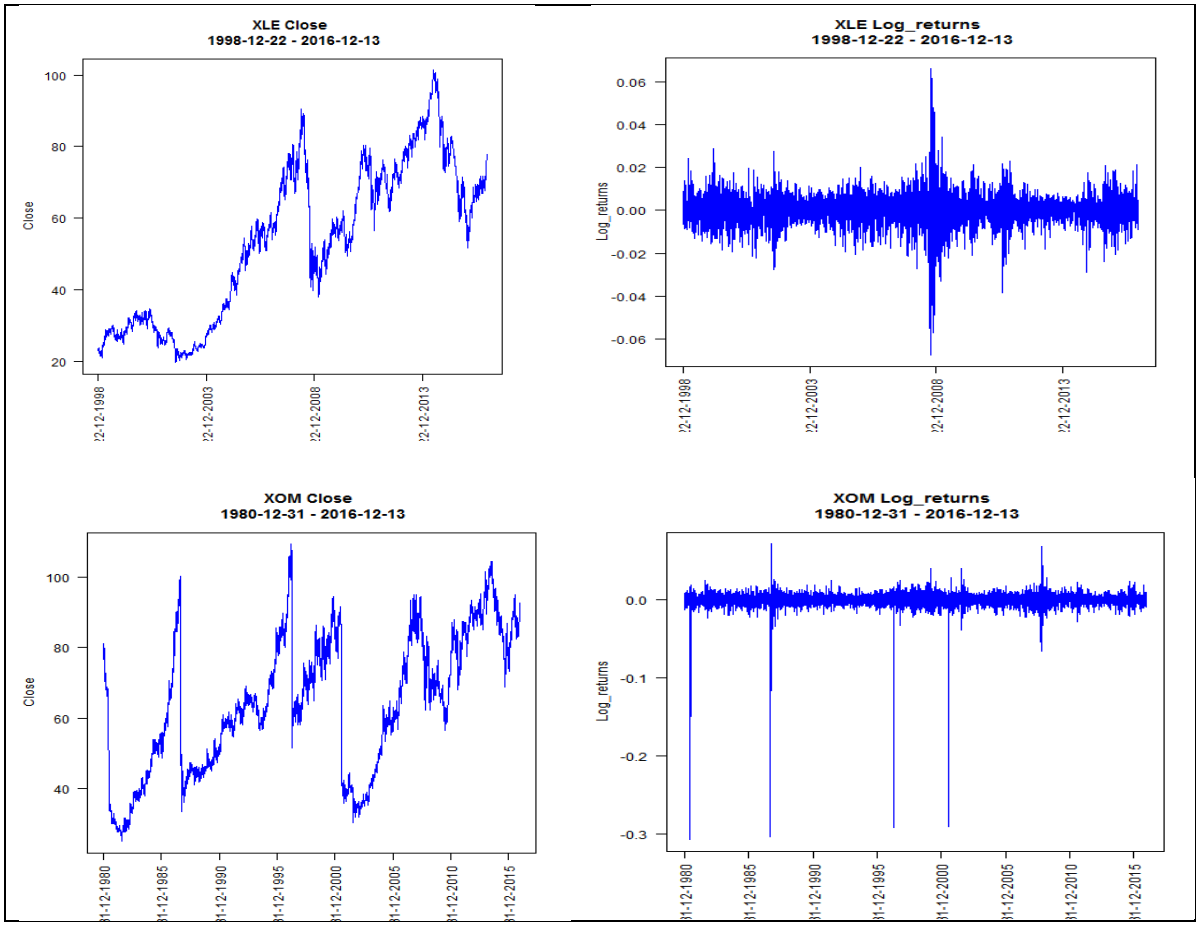
Initially, the XLF index (financial sector index) and its constituents were selected to conduct this experiment. However, it became apparent during the initial phase of analysis, that data was missing. This had the potential to skew the analysis results. For example, two of the index's constituents SPGI (S&P Global Inc) and WLTW (Willis Towers Watson PLC) did not contain any sentiment data. Furthermore, BRK-B (Berkshire Hathaway B), the first largest weight, representing 10% of the index weight was missing 85% of the sentiment data. The same comment applied to BAC (Bank of America Corp), the 4th largest weight, representing 8% of the index. In this case, 37% of the sentiment data was missing.
The XLE index looked like a better fit as no constituent were missing sentiment data. In addition, the first 2 constituents, which represent 17%, 15% of the index were only missing 6%, 2% of the sentiment data.

3.6.2 Exploratory Data Analysis

The predictor variables present in each of the XLE index and its constituents datasets are continuous variable. The technical indicators all derive from the price and volume and the sentiment is represented by a numerical value that fluctuates between 1.0 and –1.0. The response variable, i.e. the returns, is also a continuous variable, that has been dummified to a *Up* and *Down* category, depending on the return level (c.f. section 'The Response Variable' for further details). This was necessary as we are performing a supervised classification task. The remainder of this section described the time series data distribution and behaviours.

3.6.2.1 Time Series Visualisation

The below plots show the price fluctuation and log returns movement over time for the XLE index and its three largest constituents, namely Exxon Mobil Corp (XOM), Chevron Corp (CVX) and Schlumberger Ltd (SLB). The full set of graphs are available in Appendix 11. Although there is a general upward price trend between 1980 and 2012 observed across these graphs. It is difficult to spot similarity in trend patterns between these assets.
Large spike can be observed which corresponds to extreme events that are usually clustered, hence the notion of volatility clustering.
The index and constituents log returns seem to vary around their individual mean as provided in Table 1 below. This corresponds to the concept of mean reversion, i.e. the tendency of the log returns to converge back to its mean.



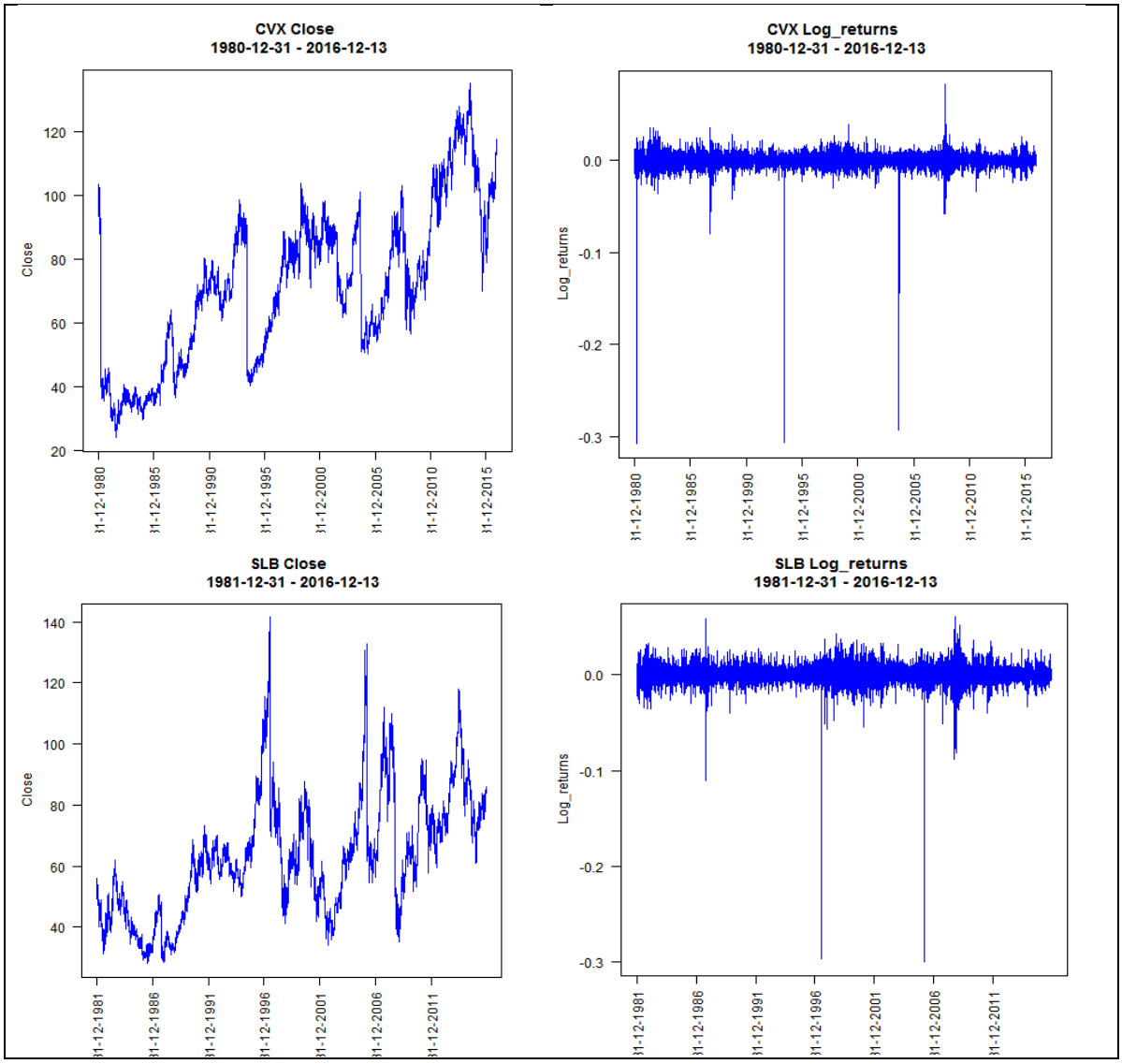


Figure 11 – XLE and the main six constituents close price and return time series

3.6.2.2 Response Variable Descriptive Statistics

Table 2 displays some basic statistical properties of the index and its constituents over their individual time period (usually greater than 20 years). All assets bar RRC and SE show a negative skew, which means most values are concentrated on the right of the mean, with extreme values to the left (and reverse). This is a sign of asymmetry and deviation from the normal distribution. The kurtosis relates to the flattening or "peakedness" of the distribution. A large majority of the assets show a kurtosis greater than 3, i.e. their distribution is leptokurtic. It is sharper than the normal distribution, with values clustered around the mean and showing the existence of thick tails. This means a higher probability of extreme values than in a normal distribution. This is opposite for PSX and PXD that show a kurtosis inferior to 3. The statistical theory states that normally distributed assets should have a skewness of value 0 and a kurtosis of value 3.

Name	Minimum	Maximum	Mean	Median	Variance	Skewness	Kurtosis
XLE	–0.067748	0.066231	0.000114	0.000274	0.000058	–0.399458	8.570529
APA	–0.311733	0.083923	0.000058	0	0.000129	–2.646812	70.680149
APC	–0.308981	0.092018	0.000068	0	0.000133	–5.019408	131.064398
BHI	–0.197411	0.108233	0.000072	0	0.00013	–0.69397	17.957388
CHK	–0.317946	0.128543	–0.000039	0	0.000346	–1.74969	30.186695
COG	–0.312524	0.100846	0.000018	0	0.000164	–5.959679	138.061088
COP	–0.518234	0.068355	0.000012	0	0.000116	–15.372576	681.099037
CVX	–0.30762	0.082262	0.000029	0	0.000081	–12.906504	435.676841
CXO	–0.110763	0.116642	0.000436	0.000472	0.000189	–0.176954	8.099715
DVN	–0.298315	0.272385	0.000078	0	0.000145	–0.448106	86.624168
EOG	–0.297494	0.080649	0.000105	0	0.000146	–6.352048	157.61129
EQT	–0.303948	0.092543	0.000049	0	0.000092	–9.442066	265.974991
FTI	–0.286031	0.078953	0.000052	0.000211	0.000162	–5.8476	129.124458
HAL	–0.478203	0.094652	–0.000024	0	0.000158	–9.2363	289.321101
HES	–0.477643	0.06706	0.000051	0	0.000119	–10.341109	436.176472
HP	–0.310735	0.102103	0.000028	0	0.000143	–3.919504	101.064355
KMI	–0.05883	0.062846	–0.000118	–0.000121	0.000068	0.16143	7.401637
MPC	–0.291451	0.047901	0.000083	0.000668	0.000166	–8.558746	189.515821
MRO	–0.307658	0.091163	–0.000024	0	0.000102	–4.131857	112.86563
MUR	–0.298237	0.069003	0.000014	0	0.000102	–6.150313	182.801565
NBL	–0.301103	0.09283	0.000043	0	0.000131	–3.972026	109.852524
NFX	–0.30103	0.072362	0.000069	0	0.000166	–4.525666	102.245647
NOV	–0.360152	0.094903	0.000058	0.000125	0.000234	–4.097776	90.681988
OKE	–0.305516	0.114073	0.000023	0	0.000103	–8.750222	259.656545
OXY	–0.305361	0.072281	0.000053	0	0.000083	–4.558422	151.723059
PSX	–0.035722	0.036723	0.000344	0.000425	0.000059	–0.075077	1.800237
PXD	–0.109024	0.07545	0.000145	0	0.000149	–0.295732	5.080561
RIG	–0.29611	0.078709	–0.000031	0	0.00017	–1.959343	46.93421
RRC	–0.163372	0.185961	0.000154	0	0.000208	0.159311	11.060387
SE	–0.050443	0.072812	0.000061	0.000135	0.000061	0.302737	9.635573
SLB	–0.300122	0.060375	0.000021	0	0.00011	–5.463674	151.965826
SWN	–0.476128	0.135658	–0.000028	0	0.000217	–8.064212	220.642454
TSO	–0.285521	0.125347	0.000092	0	0.000197	–1.11808	28.713586
VLO	–0.310644	0.080159	0.00005	0	0.000158	–3.724985	90.818794
WMB	–0.409454	0.303217	0.000003	0	0.000209	–4.416387	154.745735
XEC	–0.077567	0.088044	0.00026	0.000449	0.000117	–0.175281	5.220591
XOM	–0.30744	0.071552	0.000014	0	0.000077	–16.93736	576.978204

Table 1 – Descriptive statistics

3.6.2.3 Explanatory Variables Normal Distribution Test

Some of the linear machine learning models (such as the Linear Regression, Linear Discriminant Analysis, Quadradic Discriminant Analysis, etc.) assume that the explanatory variable follows a normal distribution. When this is not the case, and the training data is untouched, this could lead to instability in the model prediction accuracy. In this experiment, the lack of normality is remediated by a Box–Cox transformation. The later transforms skewed data into a Gaussian normal distribution (Box & Cox,1964, cited in Kuhn & Johnson, 2013, p.32). In order to establish the presence of a divergence from the normal distribution, the Kolmogorov–Smirnov (*ks*) test (Frank and Massey, 1951) is run on all the explanatory variables and their expected label, i.e. the dummified response variable output *Up* or *Down*.

The Kolmogorov–Smirnov test *Null* hypothesis (H0) indicates the data distribution seems to follow a normal distribution. H0 is rejected when the Kolmogorov Smirnov test returns a p–value less than 0.05. Table 2 shows that the count of attributes that do not follow a normal distribution dominate each asset. The *ks* test full results are available in the Appendix 12. For most of the assets more than 80% of the attributes do not follow a normal distribution. Therefore, it the Box-Cox transform is applied accordingly.
Figure 12 shows QQplots for both cases where the *Null* hypothesis is rejected and not rejected. It shows the shape of the attribute distribution against a normally distributed QQ plot.

Code/H0 Test	Count	Code/H0 Test	Count	Code/H0 Test	Count
NOV		**FTI**		**PSX**	
H0 is not rejected	10	H0 is not rejected	8	H0 is not rejected	34
H0 is rejected	98	H0 is rejected	100	H0 is rejected	74
APA		**HAL**		**PXD**	
H0 is not rejected	6	H0 is not rejected	2	H0 is not rejected	11
H0 is rejected	102	H0 is rejected	106	H0 is rejected	97
APC		**HES**		**RIG**	
H0 is not rejected	12	H0 is rejected	108	H0 is not rejected	14
H0 is rejected	96	**HP**		H0 is rejected	94
BHI		H0 is not rejected	11	**RRC**	
H0 is not rejected	10	H0 is rejected	97	H0 is not rejected	11
H0 is rejected	98	**KMI**		H0 is rejected	97
CHK		H0 is not rejected	22	**SE**	
H0 is not rejected	10	H0 is rejected	86	H0 is not rejected	18
H0 is rejected	98	**MPC**		H0 is rejected	90
COG		H0 is not rejected	18	**SLB**	
H0 is not rejected	4	H0 is rejected	90	H0 is not rejected	8
H0 is rejected	104	**MRO**		H0 is rejected	100
COP		H0 is not rejected	8	**SWN**	
H0 is not rejected	6	H0 is rejected	100	H0 is not rejected	10
H0 is rejected	102	**MUR**		H0 is rejected	98
CVX		H0 is not rejected	2	**TSO**	
H0 is not rejected	6	H0 is rejected	106	H0 is not rejected	6
H0 is rejected	102	**NBL**		H0 is rejected	102

CXO	
H0 is not rejected	16
H0 is rejected	92
DVN	
H0 is not rejected	11
H0 is rejected	97
EOG	
H0 is not rejected	11
H0 is rejected	97
EQT	
H0 is not rejected	3
H0 is rejected	105
H0 is not rejected	5
H0 is rejected	103
NFX	
H0 is not rejected	8
H0 is rejected	100
OKE	
H0 is not rejected	5
H0 is rejected	103
OXY	
H0 is not rejected	5
H0 is rejected	103
VLO	
H0 is not rejected	6
H0 is rejected	102
WMB	
H0 is rejected	108
XEC	
H0 is not rejected	11
H0 is rejected	97
XLE	
H0 is not rejected	11
H0 is rejected	97
XOM	
H0 is not rejected	2
H0 is rejected	104

Table 2 – Count *Null* hypothesis (H0) rejections per asset

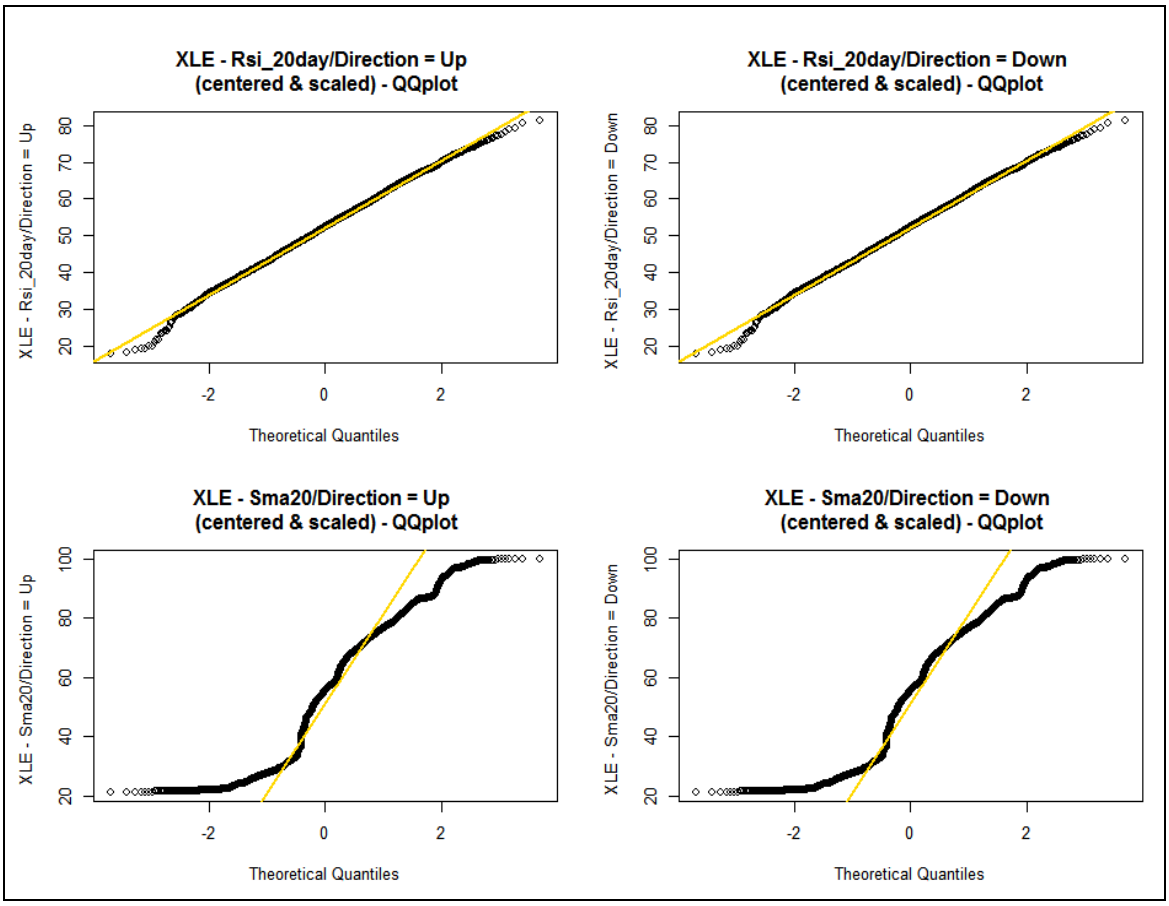


Figure 12 – QQplots: H0 is rejected (top two) / H0 is not rejected (bottom two)

3.6.2.4 Correlation and Multicollinearity

Most of the explanatory variables are price dependent. It is therefore important to check for the presence of correlation i) between each explanatory variable and the response variable, and ii) the correlation within the explanatory variables.
Figure 13 shows the degree of correlation between variable on the training set for the XLE index. The redder (bluer) the dot, the closer the correlation is to -1 (+1). It is clear that numerous variables show a large degree of correlation (e.g. SMA20 vs SAR has a correlation close to +1). This issue needs to be address during the feature selection phase (c.f. section 'Feature Selection and Extraction'). The correlation for the three main index constituents is shown in the Appendix 13. The same conclusion is reached for the stocks.

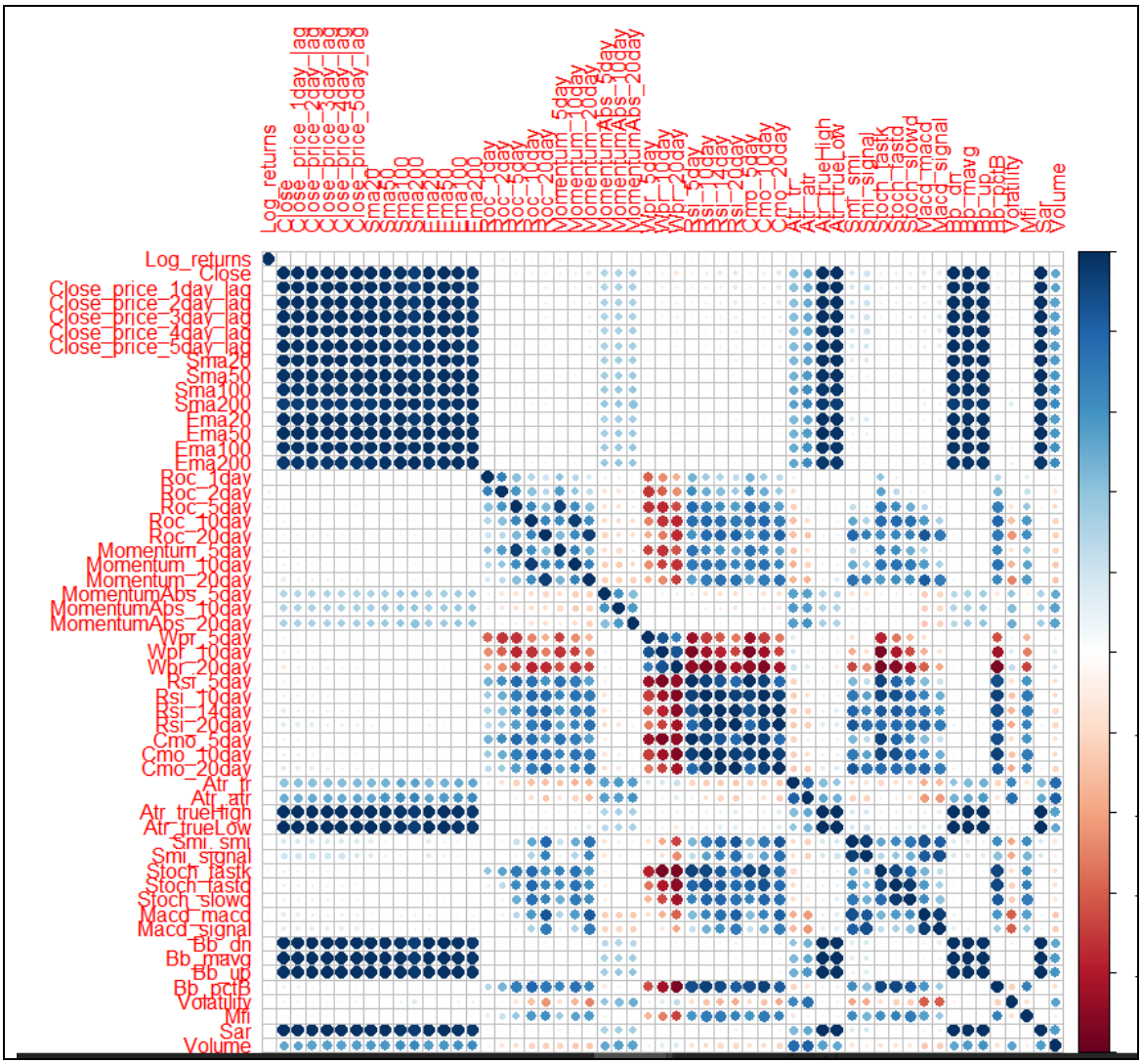


Figure 13 – XLE correlation matrix

3.6.3 Missing Data

3.6.3.1 Missing Market Data

The initial raw market data provides the historical daily price and volume for each stocks and index under analysis (from Monday to Friday). There is no business day where market data is missing. However, the data pre–processing step generates missing data. For example, the Moving Average indictors need numerous historical dates to produce a value for a given date. Consequently, lagging indicators cannot produce data for the initial number of days corresponding to their day range lag (e.g. the first 10 days for a 10-days lag). Data imputation, using median or K–Nearest Neighbours (KNN) methods was discarded as this would have generated new patterns in the time series data. Consequently, all missing data generated by lagged indicators were deleted.

3.6.3.2 Missing Sentiment Data

Sentiment data is only available for the past 5 years at most in *Quandl*. The stock sentiment population shows that sentiment is not available on all dates. Initially, K–Nearest Neighbours (KNN) was implemented with k set to 5. Some of the stocks, such as APA (Apache Corp), were missing sentiment data on numerous successive dates range from 2 to 10. This effect made the use of the KNN difficult. Instead imputation using the median is used. This imputation only applies to the training set. No imputation is generated for the validation or test sets.

3.6.4 Pre–processing

Data pre–processing is applied at two different places:

- The generic step – a first pre–processing step is performed on the training data prior to the feature selection. This involves removing i) near zero variance columns, ii) linearly dependent columns, and iii) deleting attributes showing a correlation within themselves greater than 95%. This pre–processing is applied to each stock/index, from the first record set row to the end of the first–time window training date.

- The specific step – the second pre–processing step is only applied to the time–windows training data set (i.e. neither the validation, nor the test sets). This is performed prior to training a machine learning model. This pre–processing is tailored–made to fit each algorithm data requirement, or data distribution assumptions. For example, in the case of linear models; such as the Logistic Regression or the Quadratic Discriminant Analysis, a Box–Cox transformed is applied to the data prior to fitting the models. This is required as these models assume attributes are normally distributed. Transforms, such as missing data imputation, scaling, centering, etc. are also implemented at this level.

The generic pre–processing step provides a systematic, consistent and efficient way for pre–processing the data across all models. The specific pre–processing steps offers a more granular data pre–processing step, that can be costumed to each model.

3.6.1 Class Rebalancing

An initial analysis showed that the returns produced three types of outcome: a positive, a zero outcome and a negative. Therefore, the response variable classification naturally mapped to a three levels category: *Up/Neutral* and *Down*. Table 3 shows the median and standard deviation of the 'return = 0' are respectively 5.63% and 3.27%, with a minimum and maximum frequency representation of respectively 0.29% and 12.96%. These values were generated from the list of the XLE constituents. By the same token, negative and positive returns show respectively a frequency median of 47.63% (standard deviation = 2.31%) and 46.41% (standard deviation = 1.40%). The full detail is available in Appendix 14. It is a clear indication of a class imbalance.

Usually class imbalance is resolved using pre–processing methods such as SMOTE, or an *epsilon*. SMOTE, short for *Proxy Minority Over-sampling Technique* (Chawla et al, 2002), aims at rebalancing imbalanced datasets. SMOTE under-samples and/or over-samples classes to rebalance the record set, and therefore improve the prediction performance. As for the *epsilon,* it is a 'fudge' factor that forces some records to change category to achieve a better class balance. In the same vein, some Machine Learning models, such as the SVM model, support class weights to improve detection of classes of interest.

However, as the underlying data is a time series, resampling methods such as SMOTE should not be used as they would generate 'fake' pattern in the data. On the other side, initial tests showed that an epsilon of 0.025 (25bp) was needed to rebalance classes, with approximately a 33% frequency appearance for each category. This type of level is unacceptable for trading. This is especially true in the case of low value/high volume trading, where profit and loss is made with margin varying between 1bp and 5bp.

Due to magnitude of zero return class imbalance, it was decided to merge the zero returns into the negative return bucket. This was chosen as the negative return bucket is slightly under represented compared to the positive one. Although, it would be interesting to merge the zero returns into the positive returns, but it was out scoped for the purpose of this experiment.

Table 4 shows the impact on the negative returns frequencies, following the merge of zero returns. Consequently, this experiment limits itself to a supervised two levels classification problem. The response variable is set to *Up* only for positive returns, and *Down* for the rest. This produces a slight class imbalance, skewed towards the negative returns, but this is not a major issue. Therefore, there is no attempt in this experiment to reduce the class imbalance in such case.

Frequency	returns = 0	returns > 0	returns < 0
Mean	5.34%	47.88%	46.78%
Median	5.96%	47.63%	46.41%
Std Dev	3.27%	2.31%	1.40%
Min	0.29%	42.79%	44.17%
Max	12.96%	53.37%	50.65%

Table 3 – Descriptive statistics for a three classes response variable

Frequency	returns > 0	returns <= 0
Mean	47.88%	52.12%
Median	47.63%	52.37%
Std Dev	2.31%	4.67%
Min	42.79%	44.46%
Max	53.37%	63.61%

Table 4 – Descriptive statistics for an aggregated two classes response variable

3.6.2 Feature Selection and Extraction

3.6.2.1 High Level Description

Each index constituent contains 54 technical indicators attributes generated in the pre–processing phase (c.f. section 'Data Pre–processing'). As mentioned by (Kuhn & Johnson, 2013) the presence of non–informative attributes can add extra variance to the model prediction and therefore affect negatively the overall accuracy of the model. Furthermore, it is difficult to assess upfront the degree of impact of non–informative variables, as it depends on the selected models. Some models, such as tree based models contains a layer of feature selection embedded into the model fitting. Whereas other models, such as linear regression or neural networks models are the most affected by the presence of redundant attributes. It therefore evident that feature selection is necessary to ensure maximum prediction performance.

There are two main method types of feature reduction; i) the Wrapper methods and ii) the Filter methods. Wrapper methods consist of a series of steps which aim at adding/removing the number attributes in order to maximise the model's accuracy score. The Filter methods, on the other hand, grade the suitability of an attribute list based on factors that do not consider the overall effectiveness of the model.

Both these methods present advantages and drawbacks. The Filter methods do not require model parameter tuning, as the model effectiveness is not taken into consideration. Consequently, it is faster to run and it consumes less computing resources than a Wrapper method. However, as the predictors are evaluated individually, multi–collinearity (i.e. a high–level of collinearity) between predictors could emerge from the selection process. The advantage of Filter methods is that they are less prone to over–fit the training data.

This experiment aims at establishing the extra prediction power sentiment adds to the prediction power of technical indicators. To be robust and correct, this experience requires to test each model against its optimal feature selection list. The aim is to obtain the 'best' base line accuracy rate for each model under analysis.

To satisfy the robustness requirement, a feature selection workflow shown in Figure 14 is run against each model and asset. This serves for the generation of the 'best' base line prediction test accuracy.

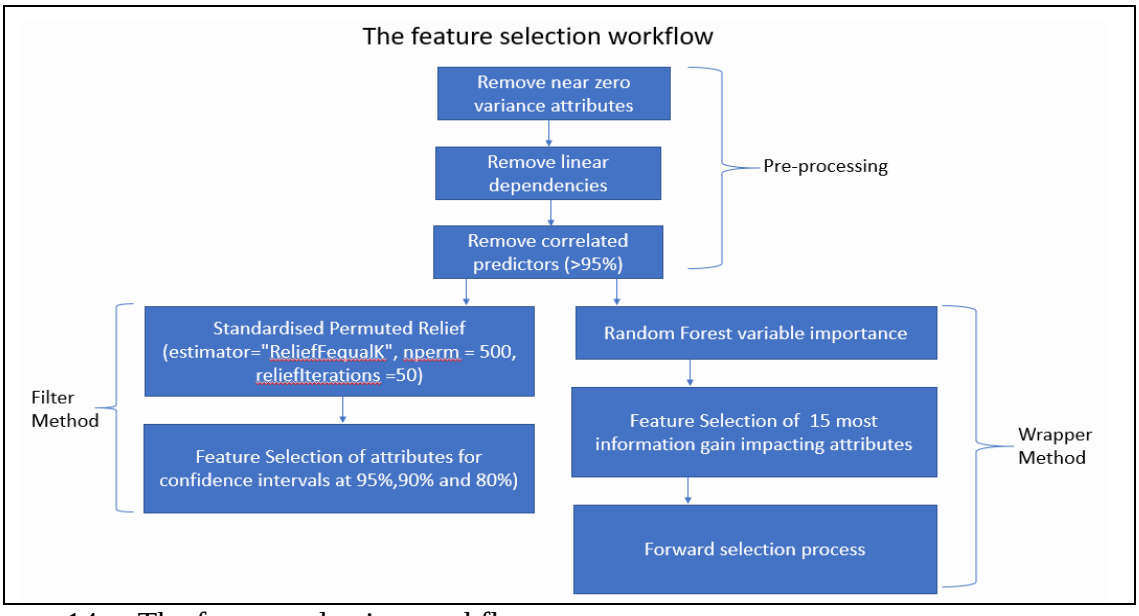


Figure 14 – The feature selection workflow

The pre–processing step consists in i) removing zero variance attributes (i.e. predictors that do not vary), ii) eliminating predictors that are linearly dependent on others and iii) pruning highly correlated variable. The purpose of this step is to minimise the variance so that the prediction does not become sensitive in the model. On completion of this step, i) the Relief Filter method and ii) the Random Forest Variable Importance algorithm followed by a Forward selection process is run in parallel, across all the assets. The aim is to establish which selection feature maximised the models' accuracy. This process is described in Algorithm 3.

```
Select a feature selection type (i.e. ReliefF or Random Forest Variable Importance)
repeat
  for each asset do
    data = load_data(asset)            #Load the underlying pre–processed data relating to
                                       #historical price/volume and technical indicators
                                       #(i.e. sentiment data is not present)
    tw = load_tw(asset)                #Load the time window frame for the asset
    reduce_frame(data, tw)             #Reduce the data date range
                                       #from inception to to the end of the first training set
    remove_zero_variance(data)         #Remove zero variance attributes.
    remove_linear_dep(data)            #Remove linear dependencies.
    Remove_corr(data)                  #Remove correlated attributes.
  end

  if feature selection is ReliefF then
    Call the permuted_relief(data) function with 500 permutations and 50 iterations.
  else
    Call the random_forest_mean_decrease_gini(data) funtion
  end
until there is no more asset
```

Algorithm 3 – The Feature Selection Workflow Algorithm

The next sections review in–depth the implementation of the variable selection process:

i) using a Filter method, implemented with the Relief model, and
ii) using a Wrapper method implemented by a Random Forest Variable Importance algorithm, followed by a Forward selection process.

3.6.1 The General Methodology

To avoid overfitting, the variable selection process exclusively runs on training data, that is never use during the validation or testing phases. Consequently, the variable selection dataset ranges from the stock inception date to the end date of the first training time window, as show on Figure 15.

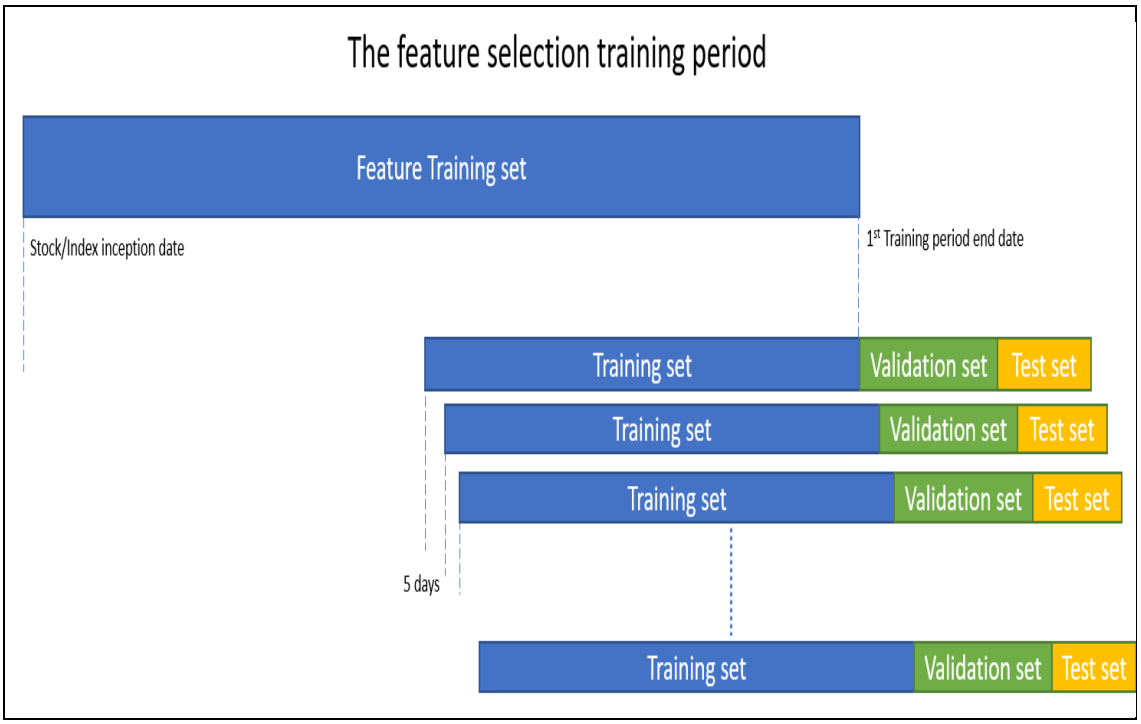


Figure 15 – Feature selection training period

There are many methodologies available for performing feature selection. Only two of them were selected out of the space of possible methodologies, due to the time/process consuming nature of feature selection. Some of the methodologies were discarded as they applied to unsupervised classification task, such as the *Principal Component Analysis* (*PCA*). The focus was therefore centred around feature selection methodologies for supervised classification task where the response variable is categorical and the predicators are numeric. These methods and their implementation are described in detail in the next sections.

3.6.2 The Filter Method

Two famous possible algorithms are available to for this type of filtering. One approach is to use a *Receiver Operating Characteristic* (*ROC*) curve. The aim is to retain predictor that 'best' separate the response variable classes. Predictors with an *Area Under the Curve (AUC)* of 0.5 are considered as irrelevant, whereas the ones leaning towards 1.0 are considered as perfectly separable predictors.
The second approach, the one retained for this study, is named the *Relief* model (Kira & Rendell, 1992 and Kononenko, 1994, cited in Kuhn & Johnson, 2013, p.472–474). This model can recognise nonlinear relationship between the response and the predictor variables.
It measures the predictors variables importance in isolation to other predictors. The algorithm randomly selects the sample set from the training data.
For each iteration, the algorithm takes the feature vector *x* from one instance and calculate its distance, *diff (x,y),* to the feature vector *y* closest to *x*, from each class. Kuhn and Johnson (2013) provide the formula for this distance calculation. This is shown in (3.60).

$$\text{Diff}(x,y) = (x-y)/C \tag{3.60}$$

Where

- C is the scaling constant between 0 and 1 for each predictor, and
- y is associated to the closest same–class instance (nearHit) or closest different–class instance (nearMiss) value.

The overall predictor score (S_i) is the sum of the differences, as shown in (3.61). In other words, predictors that show clear separation between classes should have close-by hits and distant misses. The higher the overall score, the more important is the predictor.

$$S_i = S_i - (x-\text{nearHit})^2 - (x-\text{nearMiss})^2 \tag{3.61}$$

Once all predictors are assigned a score, the next step is to select the most impacting predictors. For this, the permutation test (Good, 2000, cited in Kuhn & Johnson, 2013, p.476) is used to assess the predictor scores. The first step consists in randomly shuffling the class labels and regenerating the Relief overall scores, each time. In this implementation, the number of permutation is set to 500 and the number of iteration is set to 50. A histogram of the randomised score is generated and compared to the original, non randomised, observed score. In order to select feature given a confidence interval (c.i.), the permutation distribution is centered and scaled to produce a normal distribution of zero mean, and one standard deviation. In this study, initial experimentation showed that a unique confidence interval cut off point of 95% (1.96) did not always produce a feature list for some of the stocks. This happens when all the predictors score outside of the [–1.96,1.96] range. It was therefore decided to use three confidence intervals 95% (1.96), 90% (1.645) and 80% (1.28). When the list of selected features contains some predictors, then the process stops. Otherwise, the next range is tested. The next range is set to [–1.1645,1. 1645] (i.e. 90% confidence interval). As previously, if any predictors are retained, then they are considered as the final feature selection. Otherwise, the next and last interval range is tested [–1.28,1.28] (i.e. 80% confidence interval) is tested. This produces the final set of features selection.

By allowing for some flexibility around the feature selection, it enables to the 'best' feature selection for most of the stocks, i.e. the one with the highest confidence interval (95%), whilst ensuring a feature selection for lowest but still acceptable confidence intervals (the lowest one being 80%). The logic implementation pseudo code is available in Algorithm 4. The full R code is available in Appendix 8bis for more details. All results are available in Appendix 15.

```
#feature_selection_list represents the list of selected features using the permutated Relief method
repeat
  for each asset do
    feature_selection_list = permutated relief (asset, 1.96)
    if (feature_selection_list is empty) then
      feature_selection_list = permutated_ relief (asset, 1.645)
    else
      feature_selection_list = permutated_ relief (asset, 1.28)
    endif
until there is no more asset
```

Algorithm 4 – The permutated Relief feature selection.

Table 5 shows the feature selection for three assets at different levels of confidence intervals.

	perm.standardized
Momentum_10day	2.074645265
MomentumAbs_10day	1.793285857
Roc_10day	1.448368777
Smi_smi	1.398109204
MomentumAbs_5day	1.084293572
Wpr_5day	1.056260694
Smi_signal	0.929895318
Momentum_5day	0.860893427
Roc_20day	0.842843114
Bb_pctB	0.667610504
Stoch_slowd	0.362669476
Stoch_fastk	0.278135083
Roc_5day	0.105564989
Cmo_5day	-0.062617825
Roc_1day	-0.124363492
Wpr_20day	-0.198234982
Macd_macd	-0.458359156
Mfi	-0.48361069
Wpr_10day	-0.601287442
Momentum_20day	-0.906115256
Atr_atr	-0.942968961
Atr_tr	-1.005896359
MomentumAbs_20day	-1.03254572
Close_price_4day_lag	-1.281339064
Macd_signal	-1.432551332
Volatility	-1.692683785
Cmo_10day	-2.226379389
Roc_2day	-2.534294844
Cmo_20day	-2.545368963
Volume	-2.823705385

a) XLE (|level|>1.96 – 95% c.i.)

	perm.standardized
Roc_1day	2.617898748
Atr_tr	1.647807099
Roc_2day	1.25933535
Sma100	0.986747302
Wpr_10day	0.829138729
Ema200	0.654902329
Wpr_5day	0.481006206
Stoch_fastk	0.354355741
Momentum_20day	0.340718026
Macd_macd	0.324458955
MomentumAbs_20day	0.269400051
Sma20	0.055915316
Smi_signal	0.043905975
Mfi	-0.049841018
Wpr_20day	-0.140351561
Smi_smi	-0.142370002
Macd_signal	-0.156939618
Stoch_slowd	-0.266055368
Momentum_5day	-0.27819864
MomentumAbs_5day	-0.290589495
Cmo_5day	-0.600854757
Bb_pctB	-0.661857476
MomentumAbs_10day	-0.683655288
Atr_atr	-0.87816916
Cmo_20day	-1.03969306
Volume	-1.04042354
Momentum_10day	-1.354010881
Volatility	-1.638848995

b) KMI (|level|>1.645 – 90% c.i.)

	perm.standardized
Roc_20day	1.604437113
Wpr_20day	1.554924154
Cmo_10day	1.270080985
Roc_5day	1.119483781
Cmo_5day	1.087991962
Bb_pctB	1.032308008
Momentum_10day	1.008765804
Wpr_10day	0.904513715
MomentumAbs_10day	0.529141177
Volatility	0.440550265
Macd_signal	0.427807479
Smi_signal	0.307154621
Momentum_20day	0.219783662
Atr_tr	0.173885894
Mfi	0.035131085
Stoch_fastk	0.019966381
Wpr_5day	-0.0384412
Sma200	-0.069096232
Momentum_5day	-0.125518065
Roc_1day	-0.137689472
Cmo_20day	-0.195720512
Roc_10day	-0.228013808
Roc_2day	-0.303251854
Stoch_slowd	-0.343510548
Sma20	-0.539738255
MomentumAbs_5day	-0.628299126
Atr_atr	-0.631340653
Volume	-0.859099049
MomentumAbs_20day	-1.078179694

c)TSO (|level|>1.28 – 80% c.i.)

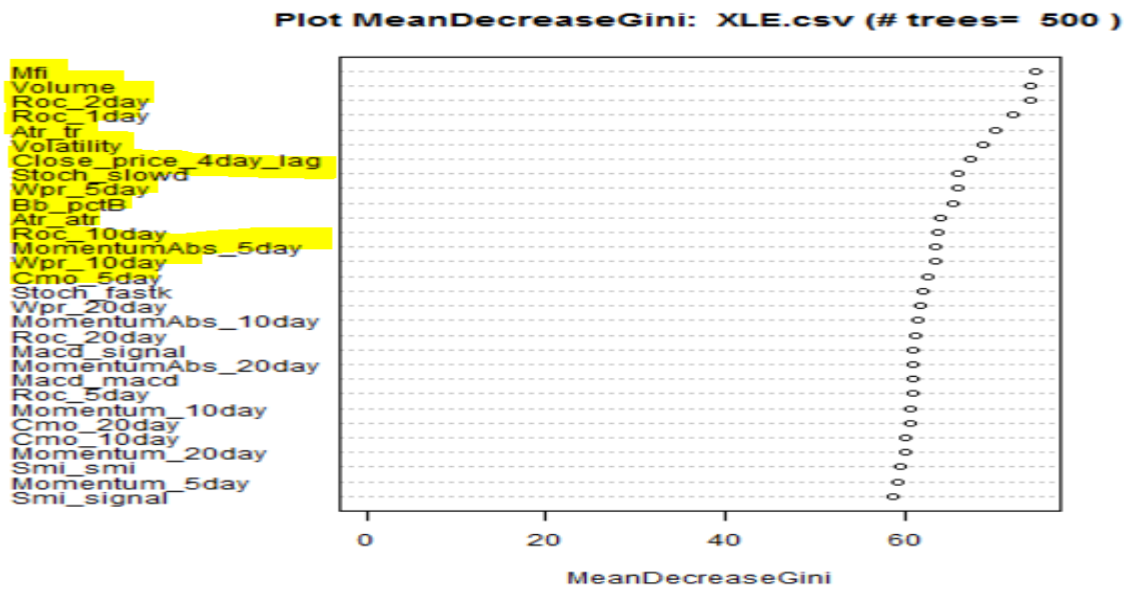
Table 5 – Feature selection for XLE, KMI and TSO at different c.i. levels.

3.6.1 The Wrapper Method

3.6.1.1 Detail of the implementation

Wrapper methods, contrary to the Filtering methods, evaluate the performance of the feature selection for each a set of machine learning algorithm. The main advantage over the Filtering is that predictors are evaluated together, instead of individually. However, this has an impact on the computing resources and time to generate the 'best' list of feature selected attributes. Furthermore, there is a risk of overfitting. In this experiment, the Mean Decrease Gini Index was selected as the scoring mechanism to discover the most important features. The Mean Decrease Gini Index measures the variable importance based on the Gini impurity index, not the model performance. It is then necessary to apply a complementary selection process to assess which predictor combinations provides the best accuracy measure. In this, study a Forward selection is used with an arbitrary cut–off point at 15. This means only the first 15 most important features are selected at all time. For each selected asset, 15 dictionaries, mapping the response variable versus the list of selected attributes, a.k.a. formula, are generated and run against each model. Each formula is built recursively by iteratively adding the next most important attribute to the following formula (c.f. Table 6). The best feature selection corresponded to the formula providing the highest accuracy/kappa for a given underlying and model (c.f. Algorithm 5). All results are available in Appendix 15.

First, the list of most important variables is generated for an asset (e.g. XLE), as shown in the graph below.



Then, the formulae are generated as follows:

formula1 = Mfi
formula2 = Volume + Roc_2day
…
formula15 = Mfi+ Volume + …. + Cmo_5day

With *Mfi* being the most important attribute, *Volume* the next most important, etc.

Table 6 – Example of formulae generation based the Wrapper method.

```
Load the list of features selected scored attributes for each underlying.
 repeat
    for each asset do
     formulas_generator (...,15)  #Call the formula generator function with a cut–off set at 15.
                                  #It generates 15 formulae by iteratively adding an extra attribute to
                                  #the following formula
                                  #i.e. the first formula starts with the most important attribute the last
                                  #one contains the entire 15 attributes
     for machine learning algorithm in the algorithm list do
       for underlying dataset in the underlying list do
         ....
         generate_model_performance_measures(..) # Generate the average validation/test  performance
                                                 # measures for each model and feature selection
         ....
        End
        ...
        generate_model_performance_measures(..)  #Store the formula that generates the highest test
                                                 #accuracy.
        ...
      end
    end
 until there is no more underlying
```

Algorithm 5 – The Wrapper method selection process

3.6.1.1 Rational for choosing the Forward Selection

The Forward selection and Backward selection processes mirror of each other. As explained above the Forward selection adds recursively attributes to the list of features selection until the maximum accuracy is reached, or the maximum number of attributes cut-off point is reached. The Backward selection does the opposite, it starts with the full list of attributes (up the defined cut-off) and remove recursively one attribute at a time until the maximum accuracy is reached.

Initial test for this experiment showed that out of the 50+ variables, most of the time, a small number of attribute (up to 5) maximised the accuracy. It is therefore more efficient to start from the beginning of the list and step forward in the process, hence the choice of the Forward selection process. This comment is only pertinent to this experiment, in the case of another index, the Backward selection process might be the better choice.

3.6.2 Sliding Time Windows

As mentioned earlier in the report, the unique test set or cross–validation methods for model optimisation and test accuracy generation are not adequate for financial time series prediction. Instead a sliding time window approach should be implemented. Each time slice contains a training, validation and test set. The overall test accuracy is produced by averaging the time windows' test accuracies over all periods. The Figure 16, borrowed from Torgot (2011) illustrates two variations of the same concept: part a) shows that each time slice is contiguous, part b) each time window is randomly selected (without replacement).

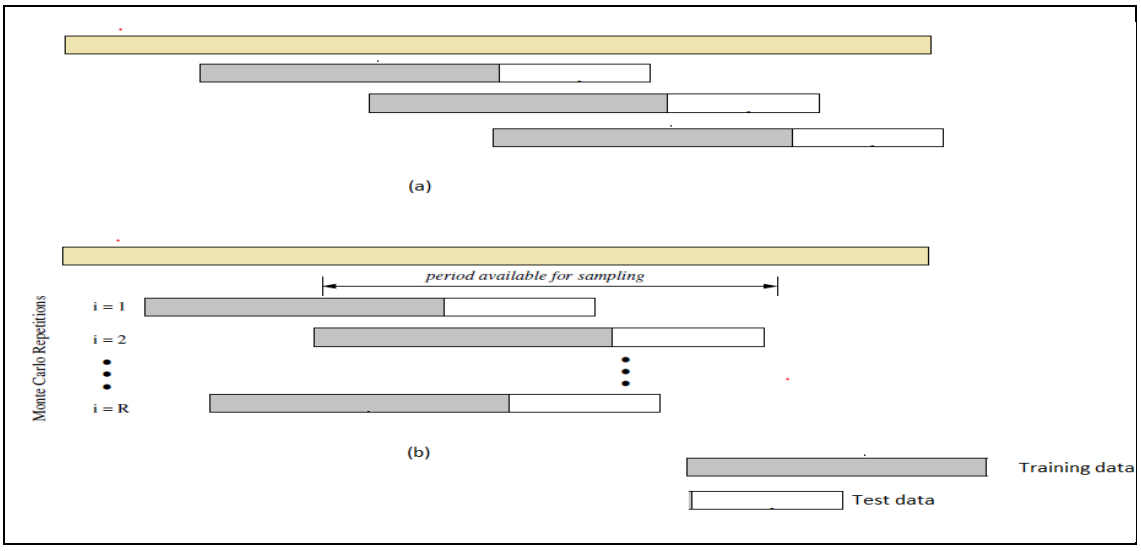


Figure 16 – Sliding time windows a) the vanilla case b) the Monte Carlo case.

Figure 17 clarifies the execution of this procedure as implemented in this experiment. A time window is composed of a contiguous 100 time-slices of equal lengths. Each time slide is divided in three parts:

i) a training set containing 220 records, which corresponds to approximately 1 year of business day prices,
ii) a validation set holding the next 66 dates (30% of the training data), and
iii) a test set representing 5 business days of data.

Figure 17a) below corresponds to the model training and optimisation. It is performed on the training, with a list of pre-defined hyper-parameters, against a validation set for each time slice. The hyper-parameter set producing the maximum validation accuracy rate, for a given time slice is named the 'best' hyper parameter set. It is then used later down in the process for the test accurate rate generation (see next paragraph for more details). At this point, the validation performance is averaged across all time slices.
So far, test sets are ignored on purpose. It is important to note that the next time-slice date corresponds to the last test date of the previous test period, plus one day. This is to avoid test data "leaking" into the training data.

Figure 17b) is concerned with the generation of the average test accuracy rate across the 100 time-slices. This time, the training set is made of the concatenation of the training and validation set ranges. For any given time-slice, the model is run with this new concatenated training set and the 'best' optimised hyper–parameter set, against the 5-days test set. This process is repeated 100 times and the averages test classification accuracy is produced.

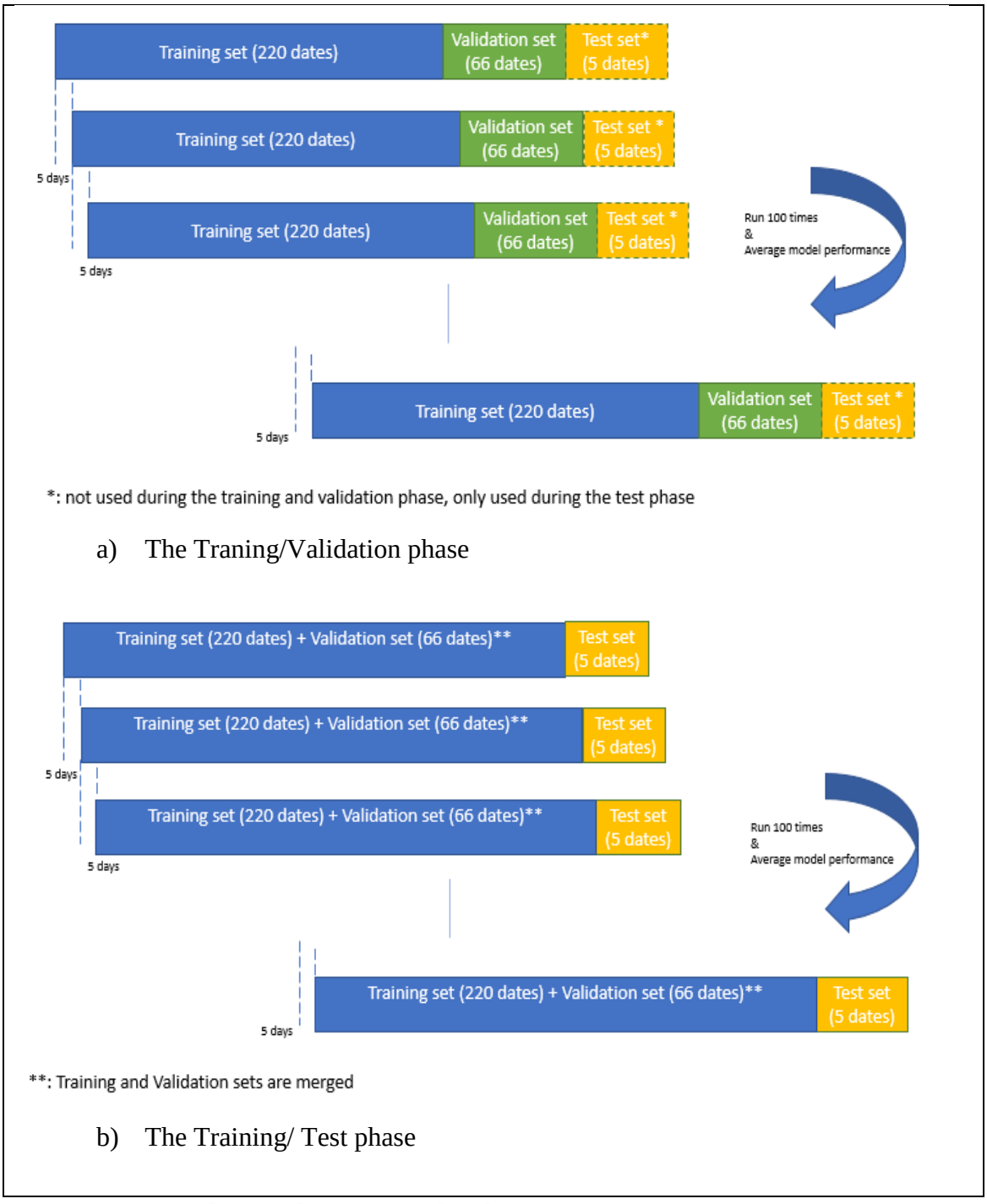


Figure 17 – Time slice implementation details

3.6.3 Anatomy of the Results Generation

3.6.3.1 The Framework

The previous sections, especially the i) 'Feature Selection and Extraction' and ii) the 'Sliding time Window' are the foundation for the generation of the base line test accuracy for each asset, against the market data (i.e. not taking the sentiment into account). Indeed, each asset and its list of Machine Learning models are run against all the feature selections generated by both the Wrapper and Filter methods. In the Wrapper case, there is a list of 15 sets of feature selected attributes (the first set contains only one variable), and the last set contains the 15 most important variables. The asset vs its models are run for each feature selection set until the maximum accurate rate is obtained. The feature selection set corresponding to this optimum is tagged as being the 'best' feature selection list for the Wrapper.
For the Filter case, each asset vs its models are run against a unique list of selected features.
Therefore, at the end of the process, only the best of Wrapper or Filter vs an asset and a model is retained, i.e. the one that produces the test highest accuracy rate. This logic is presented in (3.62)

$$A_{ij} = \text{Max}\ (\text{Max Wrapper Test Accuracy Rate}_{ij}, \text{Filter Test Accuracy Rate}_{ij}) \quad (3.62)$$

Where:

- Max Wrapper Accuracy Rate_{ij} =
  Max (Wrapper_1 Test Accuracy Rate_i ,…, Wrapper_k Test Accuracy Rate_i)$_j$
- i represents the i^{th} asset
- k represents the k^{th} Feature Selection list
- j represents the j^{th} Machine Learning model

3.6.3.2 The Model List

This paragraph lists the models and the hyper parameters used for this experiment. As explained in previous sections, each model (and its hyper-parameters) are run against each stock/index.

Model Name	***Hyper-parameters***
Logistic Discriminant Analysis (LDA)	None
Quadratic Discriminant Analysis (QDA)	None
Penalised Discriminant Analysis (PDA)	None
Support Vector Machine (SVM)	Kernel = Linear Cost = (0.001, 0.01, 0.1, 1 ,100)
Random Forest	Tree number = 500 Random Forest splits = $\sqrt{p}$ or $\frac{mp}{2}$. Bagging splits = p *'p'* represents the number of attributes in the model.
Multilayer Perceptron with Weighted Decay (MLP)	Hidden Layer size = [1,2,…,15] The weight decay = (0,0.001,0.1,1) Number of iterations = 1000
Elman and Jordan Recursive Neural Networks (RNN)	Hidden Layer size = (5,7,10,15,20) The weight decay = (0,0.001,0.01,0.1,1) Number of iterations = (100,500,1000,1500,2000)

Table 7 – Model list and their hyper-parameters

3.6.3.3 The Output Description

3.6.3.3.1 The trend prediction ouput

As each model is initially run against all stocks/index, their time window and for both feature selection methods[6]. It produces numerous outputs that are used to produce the validation and test performance measures. The section provides a high-level description of the output for i) one stock, namely CHK (Chesapeake Energy Corp), ii) one Machine Learning model, namely the Random Forest, using the Filter method only. The same logic is applied to all other stocks/index, models and feature selection methods.

- The training and validation phase. The confusion matrix for each time slide, i.e. 100 in this case, is generated from the parameters set that produce the highest validation accuracy rate. The first item represents the short name of the stock (highlighted in yellow). The second item, highlighted in green, represents the hyper-parameter list. In this case, there are 500 trees and a split set at 4. The next item, highlighted in grey, represents the 'formula id', representing the response variable name and the feature selection list. The last item is the time slice, ranging from 1 to 100.

  CHK_500_4_1960_1.csv
  …..
  CHK_500_4_1960_100.csv

  At the end, a final confusion matrix, named final_CHK_1960.csv, is created from the sum of all confusion matrices. From this confusion matrix, the performance measures are generated.

- Test phase. On completion of the generation of each training/validation confusion matrices, the test set is generated from the parameter list provided previously. The same file outputs are generated (but in another directory).

3.6.3.3.2 The volatility prediction output

In this case, the output of the Jordan and Elman RNNs model, for a given asset, is stored in two files; one containing the training and the other containing the test results. Each file contains the performance result for a time slice. This is represented by a row entry stored between the index 2 and 101 in the file. The first should be ignored. The last row corresponds to the average of each performance measure, for a given time window[7].

Training phase file name:

DVN.csvelman_final_train_DVN.csv_m_1_vol, m_1_volume, m_2_volume, m_3_volume
Test phase file name:
DVN.csvelman_final_test_DVN.csv_m_1_vol, m_1_volume, m_2_volume, m_3_volume

The above yellow highlighted section corresponds to the asset name. The green section relates to the name of the model. The grey and blue sections correspond respectively to the phases (training or testing) and the list of attributes employed for the scenario run.

[6] The results are available in \data\XLE\processed\models*model_name*\100\confusion.matrix. The *model_name* should be replaced by the model in question, e.g. elman.
[7] The results are available in ..\data\XLE\processed\models\elman\100\mse and ..\data\XLE\processed\models\jordan\100\mse.

3.1 The performance measures

Two types of performance measures were selected. This is because the trend prediction relates to a supervised classification problem, while the volatility prediction relates to a supervised regression problem.

3.1.1 The trend prediction performance measures

This paragraph reviews the performance measures that were retained for this experiment. They are derived from the confusion matrix shown in Table 8.

Total Population	**Prediction Up**	**Prediction Down**	**Total**
Expected Up	True Positive (TP)	False Negative (FN)	g1 = TP+FN
Expected Down	False Positive (FP)	True Negative (TN)	g2 = FP + TN
Total	f1 = TP+FP	f2 = FN+TN	

Table 8 – The confusion matrix

- Accuracy Rate. It is a measure of the statistical bias. Accuracy is between 0 (maximum bias) and 1 (no bias). It is calculated as (TP+TN)/(TP+FN+FP+TN). The corollary measure is the error rate, i.e. 1.0 – Accuracy Rate. This provides a measure of the overall quality of the prediction (either going Up or Down).

- Sensitivity. It measures the proportion of positives that are correctly identified (i.e. how good is a test at detecting the positives). It is calculated as TP/(TP+FN). This measure is useful for investors that do not have a position in the asset under analysis. This can help them to make an investment decision based prediction for an upward move.

- Specificity. It measures the proportion of negatives that are correctly identified. It is calculated as TP/(TP+FN). This measure is useful for investors and need to know whether the market is predicted to go down.

- Kappa (vs AUC). The Area Under the Curve (AUC) of a Receiver Operator Characteristic (ROC) is defined as the sensitivity plotted against (1-Specificity). The balance between the specificity and sensitivity is an indication of model performance. The AUC ranges between 0 and 1 (perfect classification). It measures the area under the ROC. However, this measure was not retained in this analysis, as it contains some intrinsic limitations (Wiersma et al, 2011). For example, the error components are not equally weighted and it does not provide information about the errors spatial distribution. Instead the author proposes to use the Kappa measure in its place (Cohen,1960). The Kappa evaluates the agreement between classes correcting the random chance agreement. The kappa is a measure between 0 and 1. Table 9 shows the interpretation for different range of Kappa (Altman,1991). It should be noted that is grid is only a guidance.

Kappa	Agreement
<0.20	Poor
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Good
0.81-1.00	Very Good

Table 9 – Interpretation of the Kappas

3.1.2 The volatility prediction performance measures

The following metrics were selected; the mean square error (MSE) and the root mean square error (RMSE), for the supervised regression models.

- The MSE measures the average of the square of the errors of an estimator. It is calculated as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y})^2 \tag{3.63}$$

- The RMSE measures the sample standard deviation of the difference between the actual and the predicted values. It is defined as:

$$\text{RMSE} = \sqrt{MSE} \tag{3.64}$$

Chapter 4 The Results

4.1 The Trend Prediction

4.1.1 The Base Scenario

Table 10 displays the validation/test performance measures for the base scenario, i.e. the one only relating to predicting the trend using the technical indicators only[8].
Each row in the table represents an asset (a stock or the XLE index). The first seven columns indicate respectively the code (e.g. XLE), long name (e.g. Energy Select Sector), asset type (e.g. index), the feature type (e.g. gini), the formula used to generate the model fitting (e.g. Direction~Volume+Roc_2days), the scenario (e.g. original) as well as the model name (e.g. Elman). The following 5 columns display the accuracy rate, kappa, sensitivity and specificity for the test results. The next 5 columns show the same information for the validation results. The last column presents the weighted test accuracy rate for each asset.
The index test accuracy rate is highlighted in orange in the table's top left hand corner.
The sum of the constituents' weighted accuracy rates is highlighted in orange in the table's top right hand corner. These are the main two measures used for comparison in the rest of this section.

Observation 1 – The XLE index prediction test accuracy rate is 54%, with a kappa of 7%. The sensitivity is 60% and the specificity is at 48%. The validation results are slightly better with an accuracy at 60%, a kappa at 21% and sensitivity and specificity at respectively 67% and 54%.

Observation 2 – The stocks test prediction accuracy median and mode are very close at approximately 54%, and range between the 52% and 54%. The median of the Kappa is at 7%, with a maximum kappa at 16%. Only 8 out of the 36 stocks shows kappa greater than 10% (e.g. APA 16%, HP 13%, etc.). The sensitivity and specificity show a median of 56% and 52% respectively. The validation performance measures show similar results, although slightly better the board.

Observation 3 – The proxy test accuracy rate, obtained by summing the product of each stock weight by its test accuracy rate, is 54%.

Conclusion – The base scenario shows that there is a large part of random behaviour in the stock market move. This is shown by many stock prediction showing close to zero kappa. However, it is also interesting to note that some of the stocks show 'acceptable' kappa given the problem domain, at round 10%.

[8] The base and scenario underlying data is available in the folder: ..\results

Code	Long Name	Asset Typ	Feature Selection Type	Formula Id	Formula Description	Scenario	Model	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weight	Weighted Test Accuracy Rate
XLE	Energy Select Sector	index	permutate	XLE_1960	Direction~Momentum_10day+Cmo_10day+Roc_2day+Cmo_20day+Volume	original	elman	test	0.54	0.07	0.60	0.48	validation	0.60	0.21	0.67	0.54		0.54
APA	Apache Corp	stock	permutate	APA_1280	Direction~Roc_2day+Wpr_5day+Mfi+Roc_5day	original	elman	test	0.58	0.16	0.53	0.63	validation	0.62	0.23	0.57	0.66	1.91%	0.01
APC	Anadarko Petroleum Co	stock	gini	APC_15	Direction~Volume+Roc_2day	original	lda	test	0.53	0.07	0.40	0.67	validation	0.53	0.06	0.55	0.51	2.98%	0.02
BHI	Baker Hughes Inc	stock	permutate	BHI_1960	Direction~Momentum_10day+MomentumAbs_10day	original	lda	test	0.54	0.07	0.61	0.46	validation	0.53	0.06	0.64	0.40	2.43%	0.01
CHK	Chesapeake Energy Cor	stock	permutate	CHK_1960	Direction~Roc_2day+MomentumAbs_10day+Sma100+Volatility	original	svmLinear	test	0.55	0.01	1.00	0.01	validation	0.54	0.07	0.97	0.02	0.47%	0.00
COG	Cabot Oil & Gas A	stock	permutate	COG_1960	Direction~Cmo_20day+Atr_tr	original	svmLinear	test	0.55	0.05	0.86	0.18	validation	0.49	-0.02	0.80	0.17	1.52%	0.01
COP	ConocoPhillips	stock	gini	COP_15	Direction~Volume+Mfi	original	rf_500	test	0.52	0.04	0.57	0.47	validation	0.51	0.02	0.53	0.48	3.12%	0.02
CVX	Chevron Corp	stock	gini	CVX_15	Direction~Volume+Roc_1day	original	svmLinear	test	0.54	0.03	0.94	0.09	validation	0.54	0.09	0.94	0.09	14.81%	0.08
CXO	Concho Resources Inc	stock	gini	CXO_15	Direction~Atr_tr+Roc_2day	original	mlp_weight_decay	test	0.54	0.07	0.49	0.58	validation	0.52	0.04	0.51	0.53	1.30%	0.01
DVN	Devon Energy Corp	stock	permutate	DVN_1960	Direction~Roc_1day+Roc_2day+Cmo_5day	original	jordan	test	0.54	0.08	0.45	0.63	validation	0.61	0.22	0.57	0.65	1.88%	0.01
EOG	EOG Resources	stock	gini	EOG_14	Direction~Volume+Mfi+Roc_1day	original	elman	test	0.54	0.07	0.55	0.52	validation	0.61	0.23	0.62	0.60	4.64%	0.02
EQT	EQT Corporation	stock	gini	EQT_15	Direction~Volume+Volatility	original	lda	test	0.52	0.05	0.40	0.65	validation	0.53	0.06	0.42	0.65	0.79%	0.00
FTI	FMC Technologies Inc	stock	permutate	FTI_1960	Direction~Stoch_fastk	original	lda	test	0.53	0.03	0.86	0.17	validation	0.52	0.04	0.91	0.09	0.94%	0.01
HAL	Halliburton Co	stock	permutate	HAL_1280	Direction~Mfi+Sma200+MomentumAbs_5day+Momentum_5day+Macd_signa	original	qda	test	0.54	0.07	0.76	0.31	validation	0.51	0.01	0.52	0.49	3.66%	0.02
HES	Hess Corp	stock	permutate	HES_1960	Direction~Roc_2day+Roc_1day	original	elman	test	0.54	0.07	0.63	0.44	validation	0.61	0.22	0.62	0.59	1.40%	0.01
HP	Helmerich & Payne Inc	stock	gini	HP_15	Direction~Volume+Mfi	original	lda	test	0.56	0.13	0.45	0.69	validation	0.51	0.01	0.42	0.60	0.58%	0.00
KMI	Kinder Morgan Inc	stock	permutate	KMI_1645	Direction~Roc_1day+Atr_tr	original	jordan	test	0.56	0.13	0.62	0.51	validation	0.62	0.24	0.67	0.57	2.65%	0.01
MPC	Marathon Petroleum Co	stock	gini	MPC_15	Direction~Volume+Roc_2day	original	lda	test	0.55	0.02	0.03	0.99	validation	0.53	0.07	0.03	0.97	1.70%	0.01
MRO	Marathon Oil Corp	stock	permutate	MRO_1960	Direction~Atr_tr+Smi_signal	original	svmLinear	test	0.54	0.00	1.00	0.00	validation	0.54	0.08	0.94	0.06	1.20%	0.01
MUR	Murphy Oil Corp	stock	gini	MUR_15	Direction~Volume+Roc_1day	original	mlp_weight_decay	test	0.54	0.07	0.63	0.44	validation	0.52	0.04	0.62	0.41	0.48%	0.00
NBL	Noble Energy Inc	stock	permutate	NBL_1645	Direction~Cmo_10day+Atr_atr	original	jordan	test	0.54	0.07	0.63	0.45	validation	0.58	0.17	0.59	0.58	1.58%	0.01
NFX	Newfield Exploration Co	stock	gini	NFX_14	Direction~Volume+Roc_1day+Atr_atr	original	jordan	test	0.54	0.07	0.43	0.64	validation	0.61	0.22	0.48	0.74	0.60%	0.00
NOV	National Oilwell Varco In	stock	gini	NOV_15	Direction~Roc_1day+Mfi	original	svmLinear	test	0.54	0.00	0.93	0.07	validation	0.54	0.09	0.93	0.07	1.25%	0.01
OKE	ONEOK Inc	stock	gini	OKE_14	Direction~Volume+Roc_1day+Mfi	original	elman	test	0.53	0.06	0.47	0.59	validation	0.60	0.20	0.53	0.66	0.80%	0.00
OXY	Occidental Petroleum	stock	gini	OXY_15	Direction~Volume+Ema20	original	mlp_weight_decay	test	0.54	0.07	0.56	0.51	validation	0.50	0.00	0.50	0.50	3.14%	0.02
PSX	Phillips 66	stock	permutate	PSX_1645	Direction~Cmo_5day+MomentumAbs_10day	original	elman	test	0.53	0.05	0.53	0.52	validation	0.58	0.17	0.51	0.66	2.55%	0.01
PXD	Pioneer Natural Resourc	stock	gini	PXD_14	Direction~Mfi+Roc_1day+Volume	original	elman	test	0.58	0.15	0.59	0.56	validation	0.64	0.29	0.71	0.58	4.78%	0.03
RIG	Transocean Ltd	stock	gini	RIG_15	Direction~Volume+Mfi	original	qda	test	0.56	0.11	0.59	0.52	validation	0.52	0.05	0.58	0.46	0.37%	0.00
RRC	Range Resources Corp	stock	gini	RRC_15	Direction~Volume+Atr_atr	original	svmLinear	test	0.56	0.00	1.00	0.00	validation	0.56	0.12	1.00	0.00	0.68%	0.00
SE	Spectra Energy Corp	stock	gini	SE_14	Direction~Volatility+Roc_1day+Atr_tr	original	jordan	test	0.54	0.09	0.55	0.54	validation	0.61	0.22	0.59	0.62	2.53%	0.01
SLB	Schlumberger Ltd	stock	gini	SLB_15	Direction~Volume+Mfi	original	mlp_weight_decay	test	0.55	0.07	0.80	0.27	validation	0.52	0.05	0.77	0.25	8.19%	0.05
SWN	Southwestern Energy Co	stock	gini	SWN_15	Direction~Volume+Mfi	original	svmLinear	test	0.55	0.00	1.00	0.00	validation	0.55	0.10	1.00	0.00	0.46%	0.00
TSO	Tesoro Corp	stock	permutate	TSO_1280	Direction~Roc_20day+Wpr_20day	original	elman	test	0.54	0.09	0.54	0.55	validation	0.62	0.24	0.60	0.64	2.22%	0.01
VLO	Valero Energy Corp	stock	gini	VLO_15	Direction~Volume+Mfi	original	rf_500	test	0.56	0.12	0.55	0.57	validation	0.52	0.03	0.49	0.54	2.84%	0.02
WMB	The Williams Companies	stock	permutate	WMB_128	Direction~Smi_signal+Stoch_slowd	original	elman	test	0.56	0.12	0.53	0.60	validation	0.63	0.26	0.60	0.66	1.87%	0.01
XEC	Cimarex Energy Co	stock	permutate	XEC_1960	Direction~Bb_up+MomentumAbs_5day	original	pda	test	0.56	0.11	0.45	0.66	validation	0.55	0.10	0.57	0.53	0.86%	0.00
XOM	Exxon Mobil Corp	stock	gini	XOM_15	Direction~Roc_1day+Volume	original	pda	test	0.53	0.07	0.38	0.70	validation	0.51	0.02	0.37	0.65	16.80%	0.09
							Median		0.54	0.07	0.56	0.52		0.54	0.08	0.59	0.54		
							Mode		0.54	0.00	1.00	0.00		0.54	0.09	1.00	0.00		
							Min		0.52	0.00	0.03	0.00		0.49	-0.02	0.03	0.00		
							Max		0.58	0.16	1.00	0.99		0.64	0.29	1.00	0.97		

Table 10 – The trend base scenario results

4.1.2 The Sentiment Scenario

Tables 11/12/13 and 14 show the validation/test performance measures for the four sentiment scenarios. Scenario 1,2,3 and 4 represent respectively the base scenario with the addition of the sentiment variable (S_t), sentiment variable (S_{t-1}), sentiment variable (S_{t-2}) and sentiment variable (S_{t-3}). In these scenarios, only one sentiment variable is added at a time. The last column of each scenario results' table displays the gain/loss of test prediction accuracy rate between a sentiment scenario and the base scenario. A gain is shown in green.

Observation 1 – Across all scenarios, it is apparent that the test accuracy rate is only slightly improved for a few assets (e.g. EQT in scenario 1 with a 0.02% improvement). Most of the assets seem to either i) not be impacted or ii) negatively impacted by the sentiment variable. However, it should be noted that all these results are very close to zero and therefore they are not statistically significant.

Observation 2 – The results show test accuracy rates for the index at 53%, 40%, 49% and 52% for each of the scenarios. The weighted constituents test accuracy rates are set to 53% across all scenarios.

Conclusion – The index and constituent's index trend prediction test accuracy rates are not improved by the addition of sentiment.

Scenario	Code	Type	Accurate Rate	Kappa	Sensitivity	Specificity	Type	Accurate Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment - Base)
sentiment	XLE	test	**0.53**	0.05	0.59	0.47	validation	0.53	0.05	0.59	0.47		
												0.53	
sentiment	APA	test	0.51	0.01	0.50	0.51	validation	0.62	0.24	0.64	0.60	0.01	-0.07
sentiment	APC	test	0.53	0.07	0.40	0.67	validation	0.53	0.06	0.52	0.54	0.02	0.00
sentiment	BHI	test	0.53	0.05	0.56	0.49	validation	0.53	0.06	0.56	0.49	0.01	-0.01
sentiment	CHK	test	0.55	0.00	1.00	0.00	validation	0.54	0.08	0.97	0.02	0.00	0.00
sentiment	COG	test	0.55	0.04	0.89	0.14	validation	0.55	0.10	0.89	0.14	0.01	0.00
sentiment	COP	test	0.51	0.01	0.56	0.44	validation	0.52	0.03	0.54	0.49	0.02	-0.01
sentiment	CVX	test	0.54	0.03	0.94	0.09	validation	0.54	0.09	0.94	0.09	0.08	0.00
sentiment	CXO	test	0.51	0.01	0.50	0.51	validation	0.53	0.06	0.54	0.52	0.01	-0.03
sentiment	DVN	test	0.53	0.06	0.51	0.55	validation	0.61	0.22	0.60	0.62	0.01	-0.01
sentiment	EOG	test	0.55	0.09	0.55	0.54	validation	0.55	0.09	0.55	0.54	0.03	0.01
sentiment	EQT	test	0.54	0.09	0.37	0.72	validation	0.54	0.07	0.37	0.72	0.00	0.02
sentiment	FTI	test	0.53	0.03	0.81	0.21	validation	0.53	0.06	0.81	0.21	0.00	0.00
sentiment	HAL	test	0.54	0.05	0.77	0.28	validation	0.51	0.03	0.51	0.52	0.02	-0.01
sentiment	HES	test	0.45	-0.11	0.48	0.41	validation	0.45	-0.11	0.48	0.41	0.01	-0.09
sentiment	HP	test	0.57	0.14	0.47	0.67	validation	0.51	0.02	0.45	0.57	0.00	0.00
sentiment	KMI	test	0.53	0.05	0.57	0.49	validation	0.53	0.05	0.57	0.49	0.01	-0.04
sentiment	MPC	test	0.54	0.00	0.00	1.00	validation	0.54	0.08	0.00	1.00	0.01	-0.01
sentiment	MRO	test	0.54	0.00	0.99	0.01	validation	0.54	0.09	0.99	0.01	0.01	0.00
sentiment	MUR	test	0.49	-0.03	0.62	0.36	validation	0.49	-0.02	0.62	0.36	0.00	-0.05
sentiment	NBL	test	0.51	0.02	0.58	0.44	validation	0.51	0.02	0.58	0.44	0.01	-0.03
sentiment	NFX	test	0.49	-0.02	0.43	0.55	validation	0.49	-0.02	0.43	0.55	0.00	-0.05
sentiment	NOV	test	0.54	0.00	0.89	0.10	validation	0.54	0.08	0.89	0.10	0.01	0.00
sentiment	OKE	test	0.50	-0.01	0.49	0.51	validation	0.50	0.00	0.49	0.51	0.00	-0.04
sentiment	OXY	test	0.52	0.03	0.55	0.48	validation	0.52	0.04	0.55	0.48	0.02	-0.02
sentiment	PSX	test	0.48	-0.04	0.54	0.42	validation	0.48	-0.04	0.54	0.42	0.01	-0.05
sentiment	PXD	test	0.55	0.11	0.56	0.55	validation	0.55	0.11	0.56	0.55	0.03	-0.02
sentiment	RIG	test	0.55	0.10	0.57	0.53	validation	0.55	0.10	0.57	0.53	0.00	-0.01
sentiment	RRC	test	0.56	0.00	1.00	0.00	validation	0.56	0.12	1.00	0.00	0.00	0.00
sentiment	SE	test	0.53	0.06	0.61	0.45	validation	0.53	0.06	0.61	0.45	0.01	-0.01
sentiment	SLB	test	0.49	-0.04	0.67	0.29	validation	0.49	-0.01	0.67	0.29	0.04	-0.06
sentiment	SWN	test	0.55	0.00	1.00	0.00	validation	0.55	0.10	1.00	0.00	0.00	0.00
sentiment	TSO	test	0.50	0.00	0.57	0.43	validation	0.50	0.01	0.57	0.43	0.01	-0.04
sentiment	VLO	test	0.57	0.12	0.51	0.62	validation	0.57	0.13	0.51	0.62	0.02	0.00
sentiment	WMB	test	0.54	0.09	0.50	0.59	validation	0.54	0.09	0.50	0.59	0.01	-0.02
sentiment	XEC	test	0.56	0.12	0.45	0.67	validation	0.56	0.12	0.45	0.67	0.00	0.00
sentiment	XOM	test	0.53	0.07	0.38	0.70	validation	0.53	0.07	0.38	0.70	0.09	0.00

Table 11 – S_t scenario (1) results

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment - Base)
sentiment_m1	XLE	test	**0.50**	0.00	0.50	0.50	validation	0.50	0.00	0.50	0.50		
												0.53	
sentiment_m1	APA	test	0.51	0.03	0.51	0.52	validation	0.61	0.22	0.60	0.62	0.01	-0.06
sentiment_m1	APC	test	0.53	0.07	0.40	0.67	validation	0.53	0.06	0.54	0.52	0.02	0.00
sentiment_m1	BHI	test	0.53	0.05	0.57	0.48	validation	0.53	0.05	0.57	0.48	0.01	-0.01
sentiment_m1	CHK	test	0.55	0.01	1.00	0.01	validation	0.54	0.08	0.99	0.01	0.00	0.00
sentiment_m1	COG	test	0.55	0.04	0.88	0.16	validation	0.55	0.10	0.88	0.16	0.01	0.00
sentiment_m1	COP	test	0.51	0.01	0.54	0.47	validation	0.50	0.01	0.49	0.52	0.02	-0.01
sentiment_m1	CVX	test	0.54	0.03	0.95	0.08	validation	0.54	0.09	0.95	0.08	0.08	0.00
sentiment_m1	CXO	test	0.50	0.00	0.47	0.53	validation	0.51	0.03	0.50	0.53	0.01	-0.04
sentiment_m1	DVN	test	0.51	0.03	0.48	0.54	validation	0.61	0.22	0.59	0.63	0.01	-0.03
sentiment_m1	EOG	test	0.50	0.00	0.50	0.50	validation	0.50	0.00	0.50	0.50	0.02	-0.04
sentiment_m1	EQT	test	0.52	0.06	0.38	0.68	validation	0.52	0.04	0.38	0.68	0.00	0.00
sentiment_m1	FTI	test	0.51	-0.01	0.81	0.19	validation	0.51	0.02	0.81	0.19	0.00	-0.02
sentiment_m1	HAL	test	0.54	0.07	0.75	0.31	validation	0.51	0.02	0.53	0.49	0.02	0.00
sentiment_m1	HES	test	0.49	-0.03	0.52	0.45	validation	0.49	-0.03	0.52	0.45	0.01	-0.05
sentiment_m1	HP	test	0.57	0.15	0.40	0.75	validation	0.51	0.01	0.42	0.60	0.00	0.01
sentiment_m1	KMI	test	0.52	0.04	0.54	0.50	validation	0.52	0.04	0.54	0.50	0.01	-0.04
sentiment_m1	MPC	test	0.54	0.00	0.00	1.00	validation	0.54	0.08	0.00	1.00	0.01	-0.01
sentiment_m1	MRO	test	0.54	0.00	1.00	0.00	validation	0.54	0.09	1.00	0.00	0.01	0.00
sentiment_m1	MUR	test	0.53	0.05	0.61	0.44	validation	0.53	0.06	0.61	0.44	0.00	-0.01
sentiment_m1	NBL	test	0.49	-0.02	0.49	0.49	validation	0.49	-0.02	0.49	0.49	0.01	-0.04
sentiment_m1	NFX	test	0.48	-0.05	0.46	0.50	validation	0.48	-0.05	0.46	0.50	0.00	-0.06
sentiment_m1	NOV	test	0.53	-0.03	0.89	0.08	validation	0.53	0.06	0.89	0.08	0.01	-0.01
sentiment_m1	OKE	test	0.51	0.01	0.47	0.54	validation	0.51	0.02	0.47	0.54	0.00	-0.03
sentiment_m1	OXY	test	0.54	0.09	0.58	0.51	validation	0.54	0.09	0.58	0.51	0.02	0.01
sentiment_m1	PSX	test	0.51	0.02	0.53	0.49	validation	0.51	0.02	0.53	0.49	0.01	-0.02
sentiment_m1	PXD	test	0.54	0.08	0.51	0.56	validation	0.54	0.08	0.51	0.56	0.03	-0.04
sentiment_m1	RIG	test	0.56	0.11	0.59	0.52	validation	0.56	0.12	0.59	0.52	0.00	0.00
sentiment_m1	RRC	test	0.56	0.00	1.00	0.00	validation	0.56	0.11	1.00	0.00	0.00	0.00
sentiment_m1	SE	test	0.53	0.06	0.62	0.44	validation	0.53	0.05	0.62	0.44	0.01	-0.02
sentiment_m1	SLB	test	0.49	-0.06	0.77	0.17	validation	0.49	-0.01	0.77	0.17	0.04	-0.06
sentiment_m1	SWN	test	0.55	0.00	0.98	0.02	validation	0.55	0.10	0.98	0.02	0.00	0.00
sentiment_m1	TSO	test	0.54	0.08	0.63	0.45	validation	0.54	0.08	0.63	0.45	0.01	0.00
sentiment_m1	VLO	test	0.54	0.06	0.45	0.61	validation	0.54	0.07	0.45	0.61	0.02	-0.03
sentiment_m1	WMB	test	0.56	0.11	0.53	0.59	validation	0.56	0.11	0.53	0.59	0.01	-0.01
sentiment_m1	XEC	test	0.56	0.11	0.45	0.66	validation	0.56	0.12	0.45	0.66	0.00	0.00
sentiment_m1	XOM	test	0.54	0.09	0.32	0.78	validation	0.54	0.09	0.32	0.78	0.09	0.01

Table 12 – S_{t-1} scenario (2) results

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment - Base)
sentiment_m2	XLE	test	**0.49**	-0.02	0.46	0.52	validation	0.49	-0.02	0.46	0.52		
												0.53	
sentiment_m2	APA	test	0.53	0.05	0.57	0.48	validation	0.61	0.22	0.61	0.61	0.01	-0.05
sentiment_m2	APC	test	0.53	0.07	0.43	0.64	validation	0.53	0.06	0.61	0.45	0.02	0.00
sentiment_m2	BHI	test	0.51	0.01	0.55	0.45	validation	0.51	0.01	0.55	0.45	0.01	-0.03
sentiment_m2	CHK	test	0.55	0.01	1.00	0.01	validation	0.54	0.08	0.97	0.02	0.00	0.00
sentiment_m2	COG	test	0.54	0.02	0.86	0.16	validation	0.54	0.08	0.86	0.16	0.01	-0.01
sentiment_m2	COP	test	0.52	0.04	0.57	0.47	validation	0.52	0.04	0.54	0.49	0.02	0.00
sentiment_m2	CVX	test	0.54	0.02	0.93	0.09	validation	0.54	0.08	0.93	0.09	0.08	-0.01
sentiment_m2	CXO	test	0.50	0.00	0.46	0.53	validation	0.52	0.04	0.44	0.59	0.01	-0.04
sentiment_m2	DVN	test	0.51	0.01	0.42	0.59	validation	0.61	0.21	0.58	0.63	0.01	-0.04
sentiment_m2	EOG	test	0.56	0.12	0.51	0.61	validation	0.56	0.12	0.51	0.61	0.03	0.02
sentiment_m2	EQT	test	0.51	0.04	0.39	0.65	validation	0.51	0.02	0.39	0.65	0.00	-0.01
sentiment_m2	FTI	test	0.54	0.05	0.86	0.18	validation	0.54	0.08	0.86	0.18	0.01	0.01
sentiment_m2	HAL	test	0.55	0.07	0.77	0.30	validation	0.51	0.01	0.53	0.48	0.02	0.00
sentiment_m2	HES	test	0.50	0.00	0.51	0.49	validation	0.50	0.00	0.51	0.49	0.01	-0.04
sentiment_m2	HP	test	0.57	0.14	0.40	0.74	validation	0.51	0.01	0.42	0.60	0.00	0.00
sentiment_m2	KMI	test	0.53	0.06	0.61	0.45	validation	0.53	0.06	0.61	0.45	0.01	-0.03
sentiment_m2	MPC	test	0.55	0.02	0.04	0.98	validation	0.55	0.09	0.04	0.98	0.01	0.00
sentiment_m2	MRO	test	0.54	0.00	0.94	0.06	validation	0.54	0.08	0.94	0.06	0.01	0.00
sentiment_m2	MUR	test	0.52	0.04	0.59	0.45	validation	0.52	0.05	0.59	0.45	0.00	-0.02
sentiment_m2	NBL	test	0.53	0.05	0.59	0.46	validation	0.53	0.05	0.59	0.46	0.01	-0.01
sentiment_m2	NFX	test	0.54	0.08	0.44	0.64	validation	0.54	0.08	0.44	0.64	0.00	0.00
sentiment_m2	NOV	test	0.54	-0.01	0.91	0.08	validation	0.54	0.08	0.91	0.08	0.01	0.00
sentiment_m2	OKE	test	0.51	0.01	0.42	0.58	validation	0.51	0.02	0.42	0.58	0.00	-0.03
sentiment_m2	OXY	test	0.51	0.02	0.46	0.56	validation	0.51	0.02	0.46	0.56	0.02	-0.03
sentiment_m2	PSX	test	0.49	-0.01	0.56	0.43	validation	0.49	-0.02	0.56	0.43	0.01	-0.03
sentiment_m2	PXD	test	0.53	0.06	0.53	0.53	validation	0.53	0.06	0.53	0.53	0.03	-0.05
sentiment_m2	RIG	test	0.55	0.10	0.61	0.49	validation	0.55	0.11	0.61	0.49	0.00	0.00
sentiment_m2	RRC	test	0.56	0.02	0.99	0.02	validation	0.56	0.13	0.99	0.02	0.00	0.01
sentiment_m2	SE	test	0.54	0.07	0.60	0.47	validation	0.54	0.07	0.60	0.47	0.01	-0.01
sentiment_m2	SLB	test	0.50	-0.02	0.63	0.35	validation	0.50	0.00	0.63	0.35	0.04	-0.05
sentiment_m2	SWN	test	0.55	0.01	1.00	0.01	validation	0.55	0.11	1.00	0.01	0.00	0.00
sentiment_m2	TSO	test	0.51	0.02	0.58	0.43	validation	0.51	0.02	0.58	0.43	0.01	-0.03
sentiment_m2	VLO	test	0.54	0.08	0.54	0.54	validation	0.54	0.08	0.54	0.54	0.02	-0.02
sentiment_m2	WMB	test	0.59	0.17	0.60	0.58	validation	0.59	0.17	0.60	0.58	0.01	0.02
sentiment_m2	XEC	test	0.56	0.11	0.45	0.66	validation	0.56	0.12	0.45	0.66	0.00	0.00
sentiment_m2	XOM	test	0.54	0.09	0.32	0.77	validation	0.54	0.08	0.32	0.77	0.09	0.01

Table 13 – S_{t-2} scenario (3) results

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment - Base)
sentiment_m3	XLE	test	**0.52**	0.04	0.62	0.41	validation	0.52	0.04	0.62	0.41		
												0.53	
sentiment_m3	APA	test	0.51	0.03	0.51	0.52	validation	0.62	0.23	0.63	0.60	0.01	-0.06
sentiment_m3	APC	test	0.53	0.07	0.39	0.68	validation	0.52	0.05	0.48	0.58	0.02	0.00
sentiment_m3	BHI	test	0.54	0.07	0.61	0.46	validation	0.54	0.08	0.61	0.46	0.01	0.00
sentiment_m3	CHK	test	0.55	0.00	1.00	0.00	validation	0.54	0.08	0.98	0.02	0.00	0.00
sentiment_m3	COG	test	0.55	0.04	0.89	0.15	validation	0.55	0.10	0.89	0.15	0.01	0.00
sentiment_m3	COP	test	0.49	-0.04	0.56	0.41	validation	0.50	-0.01	0.53	0.45	0.02	-0.03
sentiment_m3	CVX	test	0.54	0.03	0.94	0.09	validation	0.54	0.09	0.94	0.09	0.08	0.00
sentiment_m3	CXO	test	0.55	0.09	0.56	0.53	validation	0.54	0.08	0.54	0.54	0.01	0.01
sentiment_m3	DVN	test	0.50	-0.01	0.44	0.55	validation	0.61	0.21	0.57	0.64	0.01	-0.05
sentiment_m3	EOG	test	0.52	0.03	0.49	0.54	validation	0.52	0.03	0.49	0.54	0.02	-0.02
sentiment_m3	EQT	test	0.52	0.05	0.42	0.63	validation	0.52	0.04	0.42	0.63	0.00	0.00
sentiment_m3	FTI	test	0.53	0.03	0.82	0.21	validation	0.53	0.06	0.82	0.21	0.00	0.00
sentiment_m3	HAL	test	0.54	0.06	0.74	0.31	validation	0.51	0.02	0.53	0.49	0.02	-0.01
sentiment_m3	HES	test	0.51	0.02	0.54	0.48	validation	0.51	0.02	0.54	0.48	0.01	-0.03
sentiment_m3	HP	test	0.56	0.13	0.45	0.69	validation	0.51	0.01	0.42	0.60	0.00	0.00
sentiment_m3	KMI	test	0.57	0.13	0.59	0.54	validation	0.57	0.13	0.59	0.54	0.01	0.00
sentiment_m3	MPC	test	0.55	0.02	0.02	1.00	validation	0.55	0.10	0.02	1.00	0.01	0.00
sentiment_m3	MRO	test	0.54	0.01	0.96	0.05	validation	0.54	0.09	0.96	0.05	0.01	0.00
sentiment_m3	MUR	test	0.51	0.00	0.61	0.39	validation	0.51	0.01	0.61	0.39	0.00	-0.03
sentiment_m3	NBL	test	0.50	0.00	0.56	0.44	validation	0.50	0.00	0.56	0.44	0.01	-0.04
sentiment_m3	NFX	test	0.51	0.01	0.41	0.61	validation	0.51	0.01	0.41	0.61	0.00	-0.03
sentiment_m3	NOV	test	0.55	0.01	0.91	0.10	validation	0.55	0.09	0.91	0.10	0.01	0.00
sentiment_m3	OKE	test	0.50	-0.01	0.48	0.52	validation	0.50	0.00	0.48	0.52	0.00	-0.04
sentiment_m3	OXY	test	0.48	-0.04	0.57	0.39	validation	0.48	-0.03	0.57	0.39	0.02	-0.05
sentiment_m3	PSX	test	0.51	0.02	0.53	0.49	validation	0.51	0.02	0.53	0.49	0.01	-0.02
sentiment_m3	PXD	test	0.53	0.06	0.50	0.56	validation	0.53	0.06	0.50	0.56	0.03	-0.04
sentiment_m3	RIG	test	0.56	0.11	0.58	0.53	validation	0.56	0.11	0.58	0.53	0.00	0.00
sentiment_m3	RRC	test	0.55	-0.01	0.99	0.00	validation	0.55	0.11	0.99	0.00	0.00	0.00
sentiment_m3	SE	test	0.53	0.06	0.50	0.56	validation	0.53	0.06	0.50	0.56	0.01	-0.01
sentiment_m3	SLB	test	0.50	-0.01	0.61	0.39	validation	0.50	0.01	0.61	0.39	0.04	-0.05
sentiment_m3	SWN	test	0.54	-0.02	0.98	0.00	validation	0.54	0.08	0.98	0.00	0.00	-0.01
sentiment_m3	TSO	test	0.49	-0.01	0.52	0.46	validation	0.49	-0.01	0.52	0.46	0.01	-0.05
sentiment_m3	VLO	test	0.54	0.08	0.49	0.58	validation	0.54	0.08	0.49	0.58	0.02	-0.02
sentiment_m3	WMB	test	0.53	0.06	0.50	0.56	validation	0.53	0.06	0.50	0.56	0.01	-0.03
sentiment_m3	XEC	test	0.56	0.11	0.45	0.66	validation	0.56	0.12	0.45	0.66	0.00	0.00
sentiment_m3	XOM	test	0.53	0.06	0.42	0.64	validation	0.53	0.06	0.42	0.64	0.09	-0.01

Table 14 – S_{t-3} scenario (4) results

4.1.3 The Sentiment Momentum Scenario

Tables 15 to 18 show the validation/test performance measures for the four sentiment momentum scenarios. In scenario 1, the sentiment momentum, i.e. $SM_t = (S_t - S_{t-1})$ is added as an extra variable to all the models. Scenario 6 augments scenario 5 by adding the t-1 lag version of scenario momentum, i.e. $SM_{t-1.}$ Scenario 7 and 8 continue in the same fashion by adding to the previous scenario an extra lagged sentiment momentum, i.e. respectively SM_{t-2} and $SM_{t-3.}$

Observation – The results show test accuracy rate for the index at 49%, 45%, 48% and 48% for each of the scenarios. The weighted constituents test accuracy rates show the following levels: 53%,52%,52% and 52%.

Conclusion – The constituents' sentiment momentum has a slightly better prediction power than the index's sentiment moment. However, the prediction power remains below the 54% level shown in the base case. Once again, the sentiment momentum does not seem to improve the index or constituent's index trend prediction.

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment Momentum - Base)
sentiment_momentum	XLE	test	**0.49**	-0.02	0.53	0.45	validation	0.49	-0.02	0.53	0.45		-0.05
												0.53	
sentiment_momentum	APA	test	0.48	-0.04	0.45	0.52	validation	0.61	0.23	0.62	0.61	0.01	-0.10
sentiment_momentum	APC	test	0.52	0.05	0.37	0.68	validation	0.53	0.06	0.55	0.51	0.02	-0.01
sentiment_momentum	BHI	test	0.54	0.08	0.59	0.49	validation	0.54	0.09	0.59	0.49	0.01	0.01
sentiment_momentum	CHK	test	0.55	0.00	0.99	0.01	validation	0.54	0.08	0.98	0.01	0.00	0.00
sentiment_momentum	COG	test	0.55	0.04	0.85	0.19	validation	0.55	0.10	0.85	0.19	0.01	0.00
sentiment_momentum	COP	test	0.51	0.02	0.57	0.44	validation	0.51	0.02	0.55	0.47	0.02	-0.01
sentiment_momentum	CVX	test	0.54	0.03	0.95	0.08	validation	0.54	0.09	0.95	0.08	0.08	0.00
sentiment_momentum	CXO	test	0.49	-0.01	0.42	0.56	validation	0.50	0.00	0.48	0.52	0.01	-0.04
sentiment_momentum	DVN	test	0.50	-0.01	0.44	0.55	validation	0.61	0.21	0.57	0.64	0.01	-0.05
sentiment_momentum	EOG	test	0.53	0.05	0.50	0.55	validation	0.53	0.05	0.50	0.55	0.02	-0.01
sentiment_momentum	EQT	test	0.52	0.05	0.42	0.64	validation	0.52	0.04	0.42	0.64	0.00	0.00
sentiment_momentum	FTI	test	0.53	0.02	0.85	0.17	validation	0.53	0.06	0.85	0.17	0.00	0.00
sentiment_momentum	HAL	test	0.55	0.07	0.77	0.31	validation	0.50	0.01	0.51	0.49	0.02	0.00
sentiment_momentum	HES	test	0.48	-0.04	0.53	0.42	validation	0.48	-0.04	0.53	0.42	0.01	-0.06
sentiment_momentum	HP	test	0.56	0.13	0.45	0.69	validation	0.51	0.01	0.42	0.60	0.00	0.00
sentiment_momentum	KMI	test	0.56	0.12	0.63	0.50	validation	0.56	0.12	0.63	0.50	0.01	0.00
sentiment_momentum	MPC	test	0.54	0.00	0.00	1.00	validation	0.54	0.08	0.00	1.00	0.01	-0.01
sentiment_momentum	MRO	test	0.53	-0.03	0.96	0.01	validation	0.53	0.06	0.96	0.01	0.01	-0.02
sentiment_momentum	MUR	test	0.50	-0.01	0.60	0.39	validation	0.50	0.00	0.60	0.39	0.00	-0.04
sentiment_momentum	NBL	test	0.51	0.01	0.58	0.43	validation	0.51	0.01	0.58	0.43	0.01	-0.03
sentiment_momentum	NFX	test	0.54	0.07	0.48	0.59	validation	0.54	0.07	0.48	0.59	0.00	0.00
sentiment_momentum	NOV	test	0.54	-0.01	0.91	0.08	validation	0.54	0.08	0.91	0.08	0.01	-0.01
sentiment_momentum	OKE	test	0.55	0.09	0.47	0.62	validation	0.55	0.09	0.47	0.62	0.00	0.01
sentiment_momentum	OXY	test	0.53	0.07	0.56	0.51	validation	0.53	0.07	0.56	0.51	0.02	0.00
sentiment_momentum	PSX	test	0.49	-0.03	0.48	0.49	validation	0.49	-0.03	0.48	0.49	0.01	-0.04
sentiment_momentum	PXD	test	0.54	0.08	0.52	0.56	validation	0.54	0.08	0.52	0.56	0.03	-0.03
sentiment_momentum	RIG	test	0.56	0.11	0.58	0.53	validation	0.56	0.11	0.58	0.53	0.00	0.00
sentiment_momentum	RRC	test	0.56	0.00	1.00	0.00	validation	0.56	0.12	1.00	0.00	0.00	0.00
sentiment_momentum	SE	test	0.54	0.08	0.62	0.46	validation	0.54	0.08	0.62	0.46	0.01	-0.01
sentiment_momentum	SLB	test	0.47	-0.09	0.61	0.30	validation	0.47	-0.07	0.61	0.30	0.04	-0.09
sentiment_momentum	SWN	test	0.55	0.00	1.00	0.00	validation	0.55	0.10	1.00	0.00	0.00	0.00
sentiment_momentum	TSO	test	0.53	0.07	0.55	0.51	validation	0.53	0.07	0.55	0.51	0.01	-0.01
sentiment_momentum	VLO	test	0.53	0.05	0.49	0.56	validation	0.53	0.06	0.49	0.56	0.02	-0.03
sentiment_momentum	WMB	test	0.59	0.17	0.68	0.49	validation	0.59	0.18	0.68	0.49	0.01	0.03
sentiment_momentum	XEC	test	0.56	0.11	0.45	0.66	validation	0.56	0.12	0.45	0.66	0.00	0.00
sentiment_momentum	XOM	test	0.53	0.07	0.38	0.70	validation	0.53	0.07	0.38	0.70	0.09	0.00

Table 15 – SM_t scenario (5) results

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	AccuracyRat	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment Momentum mm1 - Base)
sentiment_momentum_mm1	XLE	test	**0.45**	-0.09	0.42	0.49	validation	0.45	-0.09	0.42	0.49		
												0.52	
sentiment_momentum_mm1	APA	test	0.52	0.04	0.53	0.52	validation	0.61	0.23	0.65	0.58	0.01	-0.05
sentiment_momentum_mm1	APC	test	0.52	0.06	0.37	0.69	validation	0.53	0.05	0.54	0.52	0.02	-0.01
sentiment_momentum_mm1	BHI	test	0.52	0.04	0.57	0.47	validation	0.52	0.04	0.57	0.47	0.01	-0.01
sentiment_momentum_mm1	CHK	test	0.55	0.01	1.00	0.01	validation	0.54	0.08	0.97	0.02	0.00	0.00
sentiment_momentum_mm1	COG	test	0.55	0.05	0.86	0.19	validation	0.55	0.10	0.86	0.19	0.01	0.00
sentiment_momentum_mm1	COP	test	0.52	0.02	0.63	0.38	validation	0.52	0.03	0.59	0.43	0.02	0.00
sentiment_momentum_mm1	CVX	test	0.53	0.01	0.93	0.09	validation	0.53	0.07	0.93	0.09	0.08	-0.01
sentiment_momentum_mm1	CXO	test	0.52	0.04	0.50	0.54	validation	0.50	-0.01	0.47	0.52	0.01	-0.01
sentiment_momentum_mm1	DVN	test	0.52	0.04	0.48	0.56	validation	0.60	0.20	0.60	0.61	0.01	-0.02
sentiment_momentum_mm1	EOG	test	0.51	0.03	0.53	0.49	validation	0.51	0.03	0.53	0.49	0.02	-0.02
sentiment_momentum_mm1	EQT	test	0.52	0.05	0.42	0.64	validation	0.52	0.04	0.42	0.64	0.00	0.00
sentiment_momentum_mm1	FTI	test	0.53	0.03	0.80	0.23	validation	0.53	0.06	0.80	0.23	0.00	0.00
sentiment_momentum_mm1	HAL	test	0.53	0.05	0.74	0.31	validation	0.50	0.00	0.54	0.46	0.02	-0.01
sentiment_momentum_mm1	HES	test	0.55	0.10	0.58	0.53	validation	0.55	0.10	0.58	0.53	0.01	0.02
sentiment_momentum_mm1	HP	test	0.56	0.13	0.43	0.70	validation	0.51	0.01	0.42	0.60	0.00	0.00
sentiment_momentum_mm1	KMI	test	0.53	0.06	0.61	0.45	validation	0.53	0.06	0.61	0.45	0.01	-0.03
sentiment_momentum_mm1	MPC	test	0.55	0.02	0.03	0.99	validation	0.55	0.10	0.03	0.99	0.01	0.00
sentiment_momentum_mm1	MRO	test	0.54	-0.01	0.96	0.03	validation	0.54	0.07	0.96	0.03	0.01	-0.01
sentiment_momentum_mm1	MUR	test	0.52	0.04	0.61	0.43	validation	0.52	0.05	0.61	0.43	0.00	-0.02
sentiment_momentum_mm1	NBL	test	0.49	-0.02	0.44	0.54	validation	0.49	-0.02	0.44	0.54	0.01	-0.05
sentiment_momentum_mm1	NFX	test	0.51	0.03	0.47	0.56	validation	0.51	0.03	0.47	0.56	0.00	-0.02
sentiment_momentum_mm1	NOV	test	0.54	-0.01	0.90	0.08	validation	0.54	0.07	0.90	0.08	0.01	-0.01
sentiment_momentum_mm1	OKE	test	0.54	0.07	0.46	0.61	validation	0.54	0.08	0.46	0.61	0.00	0.00
sentiment_momentum_mm1	OXY	test	0.46	-0.09	0.49	0.41	validation	0.46	-0.09	0.49	0.41	0.01	-0.08
sentiment_momentum_mm1	PSX	test	0.50	0.00	0.48	0.52	validation	0.50	0.00	0.48	0.52	0.01	-0.03
sentiment_momentum_mm1	PXD	test	0.54	0.09	0.57	0.52	validation	0.54	0.09	0.57	0.52	0.03	-0.03
sentiment_momentum_mm1	RIG	test	0.56	0.11	0.59	0.52	validation	0.56	0.12	0.59	0.52	0.00	0.00
sentiment_momentum_mm1	RRC	test	0.56	0.01	1.00	0.00	validation	0.56	0.12	1.00	0.00	0.00	0.00
sentiment_momentum_mm1	SE	test	0.50	-0.01	0.46	0.53	validation	0.50	-0.01	0.46	0.53	0.01	-0.05
sentiment_momentum_mm1	SLB	test	0.48	-0.07	0.63	0.31	validation	0.48	-0.04	0.63	0.31	0.04	-0.07
sentiment_momentum_mm1	SWN	test	0.55	0.00	1.00	0.00	validation	0.55	0.10	1.00	0.00	0.00	0.00
sentiment_momentum_mm1	TSO	test	0.54	0.07	0.63	0.43	validation	0.54	0.07	0.63	0.43	0.01	-0.01
sentiment_momentum_mm1	VLO	test	0.53	0.04	0.45	0.59	validation	0.53	0.05	0.45	0.59	0.01	-0.04
sentiment_momentum_mm1	WMB	test	0.56	0.12	0.58	0.54	validation	0.56	0.12	0.58	0.54	0.01	0.00
sentiment_momentum_mm1	XEC	test	0.56	0.11	0.44	0.67	validation	0.56	0.12	0.44	0.67	0.00	0.00
sentiment_momentum_mm1	XOM	test	0.53	0.07	0.38	0.70	validation	0.53	0.07	0.38	0.70	0.09	0.00

Table 16 – $SM_t + SM_{t-1}$ scenario (6) results

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment Momentum mm1m2 - Base)
sentiment_momentum_mm1m2	XLE	test	**0.48**	-0.04	0.50	0.46	validation	0.48	-0.04	0.50	0.46		
												0.52	
sentiment_momentum_mm1m2	APA	test	0.54	0.08	0.52	0.56	validation	0.62	0.23	0.61	0.62	0.01	-0.04
sentiment_momentum_mm1m2	APC	test	0.53	0.08	0.43	0.65	validation	0.53	0.06	0.59	0.47	0.02	0.00
sentiment_momentum_mm1m2	BHI	test	0.52	0.04	0.55	0.49	validation	0.52	0.04	0.55	0.49	0.01	-0.01
sentiment_momentum_mm1m2	CHK	test	0.55	0.00	0.98	0.02	validation	0.54	0.08	0.97	0.02	0.00	-0.01
sentiment_momentum_mm1m2	COG	test	0.55	0.03	0.88	0.14	validation	0.55	0.09	0.88	0.14	0.01	-0.01
sentiment_momentum_mm1m2	COP	test	0.48	-0.04	0.53	0.43	validation	0.52	0.05	0.57	0.46	0.02	-0.04
sentiment_momentum_mm1m2	CVX	test	0.55	0.04	0.96	0.08	validation	0.55	0.10	0.96	0.08	0.08	0.01
sentiment_momentum_mm1m2	CXO	test	0.50	0.01	0.43	0.58	validation	0.50	0.01	0.48	0.53	0.01	-0.03
sentiment_momentum_mm1m2	DVN	test	0.53	0.06	0.48	0.58	validation	0.60	0.20	0.55	0.65	0.01	-0.01
sentiment_momentum_mm1m2	EOG	test	0.49	-0.02	0.47	0.51	validation	0.49	-0.02	0.47	0.51	0.02	-0.05
sentiment_momentum_mm1m2	EQT	test	0.53	0.07	0.39	0.68	validation	0.53	0.05	0.39	0.68	0.00	0.01
sentiment_momentum_mm1m2	FTI	test	0.54	0.06	0.83	0.23	validation	0.54	0.08	0.83	0.23	0.01	0.01
sentiment_momentum_mm1m2	HAL	test	0.53	0.04	0.75	0.29	validation	0.51	0.01	0.55	0.46	0.02	-0.02
sentiment_momentum_mm1m2	HES	test	0.47	-0.06	0.52	0.42	validation	0.47	-0.07	0.52	0.42	0.01	-0.07
sentiment_momentum_mm1m2	HP	test	0.57	0.15	0.41	0.74	validation	0.51	0.02	0.44	0.58	0.00	0.01
sentiment_momentum_mm1m2	KMI	test	0.53	0.06	0.58	0.49	validation	0.53	0.06	0.58	0.49	0.01	-0.03
sentiment_momentum_mm1m2	MPC	test	0.54	0.00	0.00	1.00	validation	0.54	0.08	0.00	1.00	0.01	-0.01
sentiment_momentum_mm1m2	MRO	test	0.54	0.01	0.92	0.10	validation	0.54	0.08	0.92	0.10	0.01	0.00
sentiment_momentum_mm1m2	MUR	test	0.52	0.04	0.64	0.40	validation	0.52	0.05	0.64	0.40	0.00	-0.02
sentiment_momentum_mm1m2	NBL	test	0.48	-0.04	0.49	0.47	validation	0.48	-0.04	0.49	0.47	0.01	-0.05
sentiment_momentum_mm1m2	NFX	test	0.53	0.06	0.50	0.55	validation	0.53	0.05	0.50	0.55	0.00	-0.01
sentiment_momentum_mm1m2	NOV	test	0.53	-0.02	0.84	0.15	validation	0.53	0.06	0.84	0.15	0.01	-0.02
sentiment_momentum_mm1m2	OKE	test	0.52	0.03	0.41	0.62	validation	0.52	0.04	0.41	0.62	0.00	-0.01
sentiment_momentum_mm1m2	OXY	test	0.48	-0.04	0.46	0.50	validation	0.48	-0.04	0.46	0.50	0.02	-0.06
sentiment_momentum_mm1m2	PSX	test	0.53	0.05	0.51	0.54	validation	0.53	0.06	0.51	0.54	0.01	0.00
sentiment_momentum_mm1m2	PXD	test	0.51	0.03	0.55	0.48	validation	0.51	0.03	0.55	0.48	0.02	-0.06
sentiment_momentum_mm1m2	RIG	test	0.55	0.10	0.58	0.52	validation	0.55	0.10	0.58	0.52	0.00	-0.01
sentiment_momentum_mm1m2	RRC	test	0.56	0.00	0.99	0.00	validation	0.56	0.11	0.99	0.00	0.00	0.00
sentiment_momentum_mm1m2	SE	test	0.54	0.08	0.51	0.57	validation	0.54	0.08	0.51	0.57	0.01	0.00
sentiment_momentum_mm1m2	SLB	test	0.46	-0.09	0.53	0.38	validation	0.46	-0.08	0.53	0.38	0.04	-0.09
sentiment_momentum_mm1m2	SWN	test	0.54	-0.01	0.90	0.09	validation	0.54	0.08	0.90	0.09	0.00	-0.01
sentiment_momentum_mm1m2	TSO	test	0.54	0.08	0.61	0.46	validation	0.54	0.08	0.61	0.46	0.01	0.00
sentiment_momentum_mm1m2	VLO	test	0.52	0.02	0.45	0.58	validation	0.52	0.03	0.45	0.58	0.01	-0.05
sentiment_momentum_mm1m2	WMB	test	0.56	0.12	0.59	0.53	validation	0.56	0.13	0.59	0.53	0.01	0.00
sentiment_momentum_mm1m2	XEC	test	0.56	0.11	0.44	0.67	validation	0.56	0.12	0.44	0.67	0.00	0.00
sentiment_momentum_mm1m2	XOM	test	0.54	0.08	0.37	0.71	validation	0.54	0.08	0.37	0.71	0.09	0.01

Table 17 – $SM_t + SM_{t-1} + SM_{t-2}$ scenario (7) results

Scenario	Code	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Type	Accuracy Rate	Kappa	Sensitivity	Specificity	Weighted Test Accuracy Rate	Delta Base Test Accuracy Rate (Sentiment Momentum mm1m2m3 - Base)
sentiment_momentum_mm1m2m3	XLE	test	**0.48**	-0.04	0.50	0.46	validation	0.48	-0.04	0.50	0.46		
												0.52	
sentiment_momentum_mm1m2m3	APA	test	0.54	0.08	0.52	0.56	validation	0.62	0.23	0.61	0.62	0.01	-0.04
sentiment_momentum_mm1m2m3	APC	test	0.53	0.08	0.43	0.65	validation	0.53	0.06	0.59	0.47	0.02	0.00
sentiment_momentum_mm1m2m3	BHI	test	0.52	0.04	0.55	0.49	validation	0.52	0.04	0.55	0.49	0.01	-0.01
sentiment_momentum_mm1m2m3	CHK	test	0.55	0.00	0.98	0.02	validation	0.54	0.08	0.97	0.02	0.00	-0.01
sentiment_momentum_mm1m2m3	COG	test	0.55	0.03	0.88	0.14	validation	0.55	0.09	0.88	0.14	0.01	-0.01
sentiment_momentum_mm1m2m3	COP	test	0.48	-0.04	0.53	0.43	validation	0.52	0.05	0.57	0.46	0.02	-0.04
sentiment_momentum_mm1m2m3	CVX	test	0.55	0.04	0.96	0.08	validation	0.55	0.10	0.96	0.08	0.08	0.01
sentiment_momentum_mm1m2m3	CXO	test	0.50	0.01	0.43	0.58	validation	0.50	0.01	0.48	0.53	0.01	-0.03
sentiment_momentum_mm1m2m3	DVN	test	0.53	0.06	0.48	0.58	validation	0.60	0.20	0.55	0.65	0.01	-0.01
sentiment_momentum_mm1m2m3	EOG	test	0.49	-0.02	0.47	0.51	validation	0.49	-0.02	0.47	0.51	0.02	-0.05
sentiment_momentum_mm1m2m3	EQT	test	0.53	0.07	0.39	0.68	validation	0.53	0.05	0.39	0.68	0.00	0.01
sentiment_momentum_mm1m2m3	FTI	test	0.54	0.06	0.83	0.23	validation	0.54	0.08	0.83	0.23	0.01	0.01
sentiment_momentum_mm1m2m3	HAL	test	0.53	0.04	0.75	0.29	validation	0.51	0.01	0.55	0.46	0.02	-0.02
sentiment_momentum_mm1m2m3	HES	test	0.47	-0.06	0.52	0.42	validation	0.47	-0.07	0.52	0.42	0.01	-0.07
sentiment_momentum_mm1m2m3	HP	test	0.57	0.15	0.41	0.74	validation	0.51	0.02	0.44	0.58	0.00	0.01
sentiment_momentum_mm1m2m3	KMI	test	0.53	0.06	0.58	0.49	validation	0.53	0.06	0.58	0.49	0.01	-0.03
sentiment_momentum_mm1m2m3	MPC	test	0.54	0.00	0.00	1.00	validation	0.54	0.08	0.00	1.00	0.01	-0.01
sentiment_momentum_mm1m2m3	MRO	test	0.54	0.01	0.92	0.10	validation	0.54	0.08	0.92	0.10	0.01	0.00
sentiment_momentum_mm1m2m3	MUR	test	0.52	0.04	0.64	0.40	validation	0.52	0.05	0.64	0.40	0.00	-0.02
sentiment_momentum_mm1m2m3	NBL	test	0.48	-0.04	0.49	0.47	validation	0.48	-0.04	0.49	0.47	0.01	-0.05
sentiment_momentum_mm1m2m3	NFX	test	0.53	0.06	0.50	0.55	validation	0.53	0.05	0.50	0.55	0.00	-0.01
sentiment_momentum_mm1m2m3	NOV	test	0.53	-0.02	0.84	0.15	validation	0.53	0.06	0.84	0.15	0.01	-0.02
sentiment_momentum_mm1m2m3	OKE	test	0.52	0.03	0.41	0.62	validation	0.52	0.04	0.41	0.62	0.00	-0.01
sentiment_momentum_mm1m2m3	OXY	test	0.48	-0.04	0.46	0.50	validation	0.48	-0.04	0.46	0.50	0.02	-0.06
sentiment_momentum_mm1m2m3	PSX	test	0.53	0.05	0.51	0.54	validation	0.53	0.06	0.51	0.54	0.01	0.00
sentiment_momentum_mm1m2m3	PXD	test	0.51	0.03	0.55	0.48	validation	0.51	0.03	0.55	0.48	0.02	-0.06
sentiment_momentum_mm1m2m3	RIG	test	0.55	0.10	0.58	0.52	validation	0.55	0.10	0.58	0.52	0.00	-0.01
sentiment_momentum_mm1m2m3	RRC	test	0.56	0.00	0.99	0.00	validation	0.56	0.11	0.99	0.00	0.00	0.00
sentiment_momentum_mm1m2m3	SE	test	0.54	0.08	0.51	0.57	validation	0.54	0.08	0.51	0.57	0.01	0.00
sentiment_momentum_mm1m2m3	SLB	test	0.46	-0.09	0.53	0.38	validation	0.46	-0.08	0.53	0.38	0.04	-0.09
sentiment_momentum_mm1m2m3	SWN	test	0.54	-0.01	0.90	0.09	validation	0.54	0.08	0.90	0.09	0.00	-0.01
sentiment_momentum_mm1m2m3	TSO	test	0.54	0.08	0.61	0.46	validation	0.54	0.08	0.61	0.46	0.01	0.00
sentiment_momentum_mm1m2m3	VLO	test	0.52	0.02	0.45	0.58	validation	0.52	0.03	0.45	0.58	0.01	-0.05
sentiment_momentum_mm1m2m3	WMB	test	0.56	0.12	0.59	0.53	validation	0.56	0.13	0.59	0.53	0.01	0.00
sentiment_momentum_mm1m2m3	XEC	test	0.56	0.11	0.44	0.67	validation	0.56	0.12	0.44	0.67	0.00	0.00
sentiment_momentum_mm1m2m3	XOM	test	0.54	0.08	0.37	0.71	validation	0.54	0.08	0.37	0.71	0.09	0.01

Table 18 – $SM_t + SM_{t-1} + SM_{t-2} + SM_{t-3}$ scenario (8) results

4.1 Volatility Prediction

As shown in the previous section, it appears that sentiment does not add prediction power to the trend forecasting. This section investigates the prediction power of sentiment on volatility forecasting.

As stated by Brownlees et Al (2012), standard tools for forecasting volatility are the GARCH models. In their study Olaniyan et Al (2015) established that the exponential (GARCH) is "efficient and reliable in predicting the stock market volatility". Furthermore, the study conducted by Sabri (2008) showed that the volume has both an impact on stock price and volatility. Therefore, the models listed in this experiment use the lag versions of the EGARCH volatility and the volume, for the period t-1, t-2 and t-3.

Since the volatility (the response variable) is unobserved; a reasonable proxy for the volatility is calculated from the square root of the return (Triacca, 2007 cited Giles, 2007 p.3). The pseudocode is presented below and the full R implementation is available in Appendix 16.

```
model = "Jordan"                                                    #the model can be set to Jordan
                                                                    #or Elman
number_sliding_windows = 100

repeat
  foreach asset do
  validation_rmse_acc = 0.0, test_rmse_acc = 0.0
   validation_rmse_avg = 0.0, test_rmse_avg = 0.0

  data_return = generate_return(data)

  data_vol = data_return ^2                                         #the proxy volatility

  data_egarch_vol = generate_egarch_vol(data_return)
  data_egarch_vol_lag1 = generate_egarch_vol_lag(data_egarch_vol,1)  #the egarch vol_t-1
  data_egarch_vol_lag2 = generate_egarch_vol_lag (data_egarch_vol,2)  #the egarch vol_t-2
  data_egarch_vol_lag3 = generate_egarch_vol_lag (data_egarch_vol,3)  #the egarch vol_t-3

  data_volume_lag1 = generate_volume_lag(data_volume,1)             #the volume_t-1
  data_volume_lag2 = generate_volume_lag(data_volume,2)             #the volume_t-2
  data_volume_lag3 = generate_volume_lag(data_volume,3)             #the volume_t-3

  while (slid_win_idx < number_sliding_windows) {
       #generate the training, validation and test data
       #impute any missing value (e.g. using the median)
       training_data = generate_training_data(slid_win_idx)
       validation_data = generate_validation_data(slid_win_idx)

       #use a min-max normalisation
       training_data = normalise(training_data)
       validation_data = normalise(validation _data)
       test_data = normalise(test _data)

       #define hyper-parameters
       the_size = c(5,7,10,15,20)
       the_iterations = c(100,500,1000,1500,2000)
       the_decay=c(0,0.001,0.01,0.1,1)
```

```
        #generate the optimised validation RMSE, and
        #get the optimised size,  iterations and decay (i.e. out best_size, etc.)
         validation_rmse = genereate_rmse( training_data, validation_data,
                                                 the_size, the iterations, the_decay,
                                                  out best_size, out best_iterations, out best_decay)

         #generate the training and test data -> the training data contains the training and validation data
         #impute any missing value (e.g. using the median)
        training_data = generate_training_data(slid_win_idx)
        test_data = generate_test_data(slid_win_idx)

        #generate the test rmse based on the 'best' hyper-parameter set
        test_rmse = genereate_rmse( training_data, test_data,
                                          best_size, best_iterations, best_decay)

         #accumulate the rmse across the time windows (i.e. 100 iterations)
         validation_rmse_acc = validation_rmse_acc + validation_rmse
         test_rmse_acc = test_rmse_acc + test_rmse
}
#generate the average of the rmse across the time windows (i.e. 100 iterations)
validation_rmse_avg = validation_rmse_acc /   number_sliding_windows
test_rmse_avg = test_rmse_acc /   number_sliding_windows

#save to file
save(asset, validation_rmse_avg)
save(asset, test_rmse_avg)
```

Pseudocode 8 – The volatility prediction with recurrent neural network models

4.1.1 The Experiment Set up

This model only contains six explanatory variables. Therefore, the feature section process is greatly simplified. A correlation matrix is used to establish the variables that have the most impact.

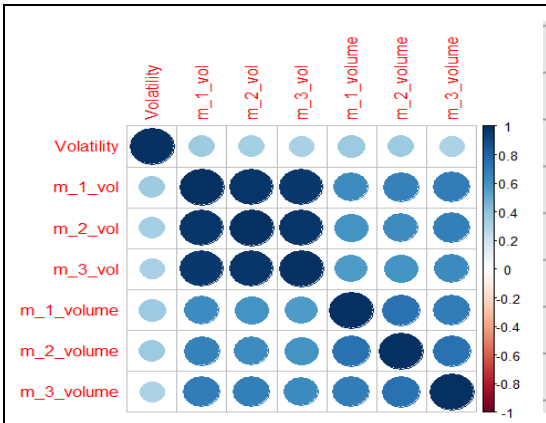
As for the previous experiment, each stock/index under analysis is studied for a list of models (i.e. the Jordan and Elman RNNs) coupled with their own feature selection set. The hyper-parameters list used for both models are the same as the ones in the previous experiment (c.f. section 'The Model List').

In this case the root mean square error (RMSE) is used as the performance measure. The lower the RMSE, the better the model prediction power[9].

[9] The base and scenario underlying data is available in ..\results\res_vol.csv

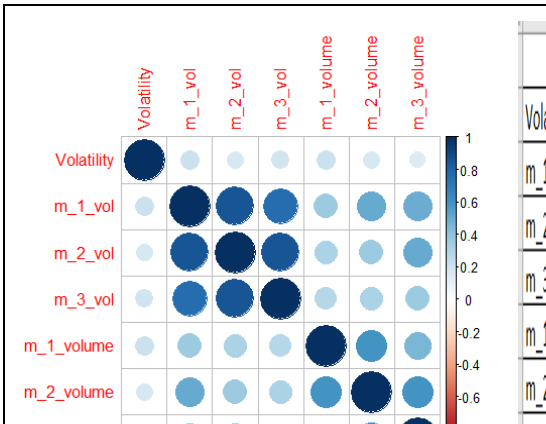
4.1.1 The Feature Selection Step

Tables 19 and 20 are representative of two groups of stock/index correlation behaviours[10]. The first group, represented by the XLE index, shows a high degree of correlation between the volatility $GARCH_t$ and $GARCH_{t-1}$ (98%). The same comment can be made about the volatility $GARCH_t$ and $GARCH_{t-2}$ (96%). In this case, only the volatility $GARCH_t$ is selected, the two others are dropped.
The second group, represented by the EOG stock, still shows a high degree of correlation between the volatility and its respective lags at respectively 85% and 76%. However, the volatility $GARCH_{t-1}$ and $GARCH_{t-2}$ are retained, as they are below the 95% correlation cut off.
The volume and its lags, of both groups, do not show high level of multi-collinearity. Therefore, the lagged volume variables are all selected.



	Volatility	m_1_vol	m_2_vol	m_3_vol	m_1_volume	m_2_volume	m_3_volume
Volatility	1	0.353617017	0.339390564	0.328819821	0.366397913	0.360268119	0.318523434
m_1_vol	0.353617017	1	0.979103619	0.960275004	0.626320887	0.67835041	0.690601998
m_2_vol	0.339390564	0.979103619	1	0.979130552	0.591858941	0.629242883	0.68095514
m_3_vol	0.328819821	0.960275004	0.979130552	1	0.563607148	0.594493951	0.631695995
m_1_volume	0.366397913	0.626320887	0.591858941	0.563607148	1	0.745764937	0.690806311
m_2_volume	0.360268119	0.67835041	0.629242883	0.594493951	0.745764937	1	0.747502083
m_3_volume	0.318523434	0.690601998	0.68095514	0.631695995	0.690806311	0.747502083	1

Table 19 – XLE index correlation matrices (Volatility is the response variable)



	Volatility	m_1_vol	m_2_vol	m_3_vol	m_1_volume	m_2_volume	m_3_volume
Volatility	1	0.21069936	0.175737897	0.194396723	0.213229488	0.173320002	0.149140047
m_1_vol	0.21069936	1	0.854695616	0.760846524	0.360336453	0.506467883	0.492784735
m_2_vol	0.175737897	0.854695616	1	0.854968245	0.310389696	0.361455288	0.507494222
m_3_vol	0.194396723	0.760846524	0.854968245	1	0.286483087	0.31230023	0.363451034
m_1_volume	0.213229488	0.360336453	0.310389696	0.286483087	1	0.592051387	0.451258618
m_2_volume	0.173320002	0.506467883	0.361455288	0.31230023	0.592051387	1	0.592286083
m_3_volume	0.149140047	0.492784735	0.507494222	0.363451034	0.451258618	0.592286083	1

Table 20 – EOG index correlation matrices (Volatility is the response variable)

[10] The correlation of each asset is available in folder: ..\data\XLE\processed\feature.selection\correlation

4.1.2 The results

4.1.2.1 The Base Scenario

Table 20 shows the base results for each of the asset. It lists the following columns:

- The code and long name of the asset (e.g. EOG / EOG Resources)
- The asset type (e.g. stock or index)
- The feature selection type (i.e. correlation base)
- The scenario name. In this case original means the base scenario not containing sentiment information.
- The model name (i.e. Jordan or Elman RNNs)
- The first result type. It is set to validation, meaning the following MSE and RMSE columns refer to the validation MSE and RMSE.
- The second result type. It is set to test, meaning the following MSE and RMSE columns refer to the test RMSE and RMSE.

code	long_name	asset_type	feature_selection_type	scenario	model_na	type	mse	rmse	type	mse	rmse
EOG	EOG Resources	stock	corr	original	jordan	validation	8.26499E-08	0.000287489	test	6.08375E-06	0.002466525
HAL	Halliburton Co	stock	corr	original	jordan	validation	8.44295E-08	0.000290568	test	4.82982E-07	0.000694969
NBL	Noble Energy Inc	stock	corr	original	jordan	validation	8.54172E-08	0.000292262	test	3.30916E-06	0.00181911
OXY	Occidental Petroleum	stock	corr	original	jordan	validation	8.10904E-08	0.000284764	test	2.28934E-06	0.001513056
APA	Apache Corp	stock	corr	original	elman	validation	8.55383E-08	0.000292469	test	1.50849E-05	0.00388393
APC	Anadarko Petroleum Corp	stock	corr	original	elman	validation	8.77932E-08	0.000296299	test	1.00128E-05	0.003164297
BHI	Baker Hughes Inc	stock	corr	original	elman	validation	8.59994E-08	0.000293257	test	5.77237E-06	0.002402576
CHK	Chesapeake Energy Corp	stock	corr	original	elman	validation	9.58993E-08	0.000309676	test	0.000613228	0.024763445
COG	Cabot Oil & Gas A	stock	corr	original	elman	validation	1.15199E-07	0.000339409	test	1.31169E-06	0.00114529
COP	ConocoPhillips	stock	corr	original	elman	validation	9.61667E-08	0.000310108	test	5.6532E-06	0.002377647
CVX	Chevron Corp	stock	corr	original	elman	validation	8.77383E-08	0.000296207	test	6.30649E-06	0.002511272
CXO	Concho Resources Inc	stock	corr	original	elman	validation	9.65087E-08	0.000310659	test	1.34246E-05	0.003663957
DVN	Devon Energy Corp	stock	corr	original	elman	validation	1.07276E-07	0.00032753	test	2.9615E-05	0.00544197
EQT	EQT Corporation	stock	corr	original	elman	validation	9.98086E-08	0.000315925	test	1.39219E-05	0.003731212
FTI	FMC Technologies Inc	stock	corr	original	elman	validation	9.06837E-08	0.000301137	test	1.01027E-05	0.00317847
HES	Hess Corp	stock	corr	original	elman	validation	1.06456E-07	0.000326277	test	1.58233E-05	0.003977849
HP	Helmerich & Payne Inc	stock	corr	original	elman	validation	8.61391E-08	0.000293495	test	1.30465E-05	0.003611996
KMI	Kinder Morgan Inc	stock	corr	original	elman	validation	9.58391E-08	0.000309579	test	1.82461E-05	0.004271542
MPC	Marathon Petroleum Corp.	stock	corr	original	elman	validation	9.66574E-08	0.000310898	test	6.6297E-06	0.002574819
MRO	Marathon Oil Corp	stock	corr	original	elman	validation	1.11455E-07	0.000333848	test	2.81736E-05	0.005307879
MUR	Murphy Oil Corp	stock	corr	original	elman	validation	8.65427E-08	0.000294181	test	0.00010169	0.010084131
NFX	Newfield Exploration Co	stock	corr	original	elman	validation	9.72088E-08	0.000311783	test	1.26437E-06	0.001124441
NOV	National Oilwell Varco Inc	stock	corr	original	elman	validation	9.60874E-08	0.00030998	test	1.26414E-05	0.003555478
OKE	ONEOK Inc	stock	corr	original	elman	validation	9.32134E-08	0.000305309	test	1.76268E-06	0.00132766
PSX	Phillips 66	stock	corr	original	elman	validation	8.62256E-08	0.000293642	test	4.0845E-06	0.002021015
PXD	Pioneer Natural Resources	stock	corr	original	elman	validation	9.82152E-08	0.000313393	test	7.16031E-06	0.002675875
RIG	Transocean Ltd	stock	corr	original	elman	validation	8.59549E-08	0.000293181	test	9.52608E-05	0.009760162
RRC	Range Resources Corp	stock	corr	original	elman	validation	8.66535E-08	0.00029437	test	1.63858E-05	0.004047932
SE	Spectra Energy Corp	stock	corr	original	elman	validation	9.07882E-08	0.000301311	test	5.00992E-06	0.002238286
SLB	Schlumberger Ltd	stock	corr	original	elman	validation	8.3935E-08	0.000289715	test	7.16236E-06	0.002676259
SWN	Southwestern Energy Co	stock	corr	original	elman	validation	8.22333E-08	0.000286764	test	0.000146202	0.012091419
TSO	Tesoro Corp	stock	corr	original	elman	validation	8.56028E-08	0.00029258	test	6.10452E-06	0.002470733
VLO	Valero Energy Corp	stock	corr	original	elman	validation	9.31148E-08	0.000305147	test	5.28989E-06	0.002299977
WMB	The Williams Companies Inc	stock	corr	original	elman	validation	9.77473E-08	0.000312646	test	1.32526E-05	0.003640411
XEC	Cimarex Energy Co	stock	corr	original	elman	validation	8.81159E-08	0.000296843	test	7.6083E-06	0.002758315
XLE	Energy Select Sector	index	corr	original	elman	validation	8.58322E-08	0.000292971	test	4.05461E-06	0.002013606
XOM	Exxon Mobil Corp	stock	corr	original	elman	validation	8.71524E-08	0.000295216	test	5.16943E-06	0.002273639

Table 20 – The volatility base scenario results

4.1.2.2 The Sentiment Scenario

Table 21 shows the MSE/RMSE validation and test results when the sentiment S_{t-1} is added to the equation. The column named 'delta S1-Base' corresponds to the difference in RMSEs between the base results RMSE and the sentiment scenario RMSE. Negative figures are highlighted in green. They show a reduction in RMSE due to the addition of the sentiment variable.

In this case, there is a reduction in test RMSE for 24 stocks out of the 36. These 24 stocks represent 87.87% of total index weight. The sum of the constituents' weighted RMSE is 0.002900439.

There is also a reduction in test RMSE for XLE index. The RMSE moves from 0.002013606 in the base scenario to 0.000646459 in the S_{t-1} scenario.

code	model_name	scenario	type	mse	rmse	type	mse	rmse	delta S1 - Base	weights	weighted rmse
EOG	jordan	sentiment_m1	validation	1.22428E-06	0.001106473	test	3.69082E-06	0.001921151	-0.000545374	4.64%	8.91414E-05
HAL	jordan	sentiment_m1	validation	5.98251E-07	0.000773467	test	3.04904E-07	0.000552181	-0.000142788	3.66%	2.02098E-05
NBL	jordan	sentiment_m1	validation	2.86618E-06	0.00169298	test	1.88432E-06	0.001372707	-0.000446403	1.58%	2.16888E-05
OXY	jordan	sentiment_m1	validation	4.59378E-07	0.000677774	test	1.72062E-06	0.001311725	-0.000201332	3.14%	4.11882E-05
APA	elman	sentiment_m1	validation	5.40497E-06	0.002324859	test	2.0349E-05	0.004510981	0.000627052	1.91%	8.61597E-05
APC	elman	sentiment_m1	validation	6.73796E-06	0.002595758	test	2.57376E-06	0.001604293	-0.001560004	2.98%	4.78079E-05
BHI	elman	sentiment_m1	validation	1.60363E-06	0.001266343	test	1.66168E-06	0.001289063	-0.001113514	2.43%	3.13242E-05
CHK	elman	sentiment_m1	validation	0.000457501	0.021389264	test	0.000733947	0.027091447	0.002328003	0.47%	0.00012733
COG	elman	sentiment_m1	validation	8.49214E-06	0.002914128	test	1.72927E-05	0.00415845	0.00301316	1.52%	6.32084E-05
COP	elman	sentiment_m1	validation	3.38591E-06	0.001840085	test	5.30372E-06	0.002302981	-7.46657E-05	3.12%	7.1853E-05
CVX	elman	sentiment_m1	validation	4.28324E-07	0.000654464	test	6.02729E-07	0.000776356	-0.001734916	14.81%	0.000114978
CXO	elman	sentiment_m1	validation	6.12021E-06	0.002473906	test	1.44432E-06	0.001201798	-0.002462158	1.30%	1.56234E-05
DVN	elman	sentiment_m1	validation	1.2753E-05	0.003571139	test	2.92193E-05	0.005405485	-3.64848E-05	1.88%	0.000101623
EQT	elman	sentiment_m1	validation	7.84349E-06	0.002800624	test	5.06125E-06	0.002249722	-0.00148149	0.79%	1.77728E-05
FTI	elman	sentiment_m1	validation	9.86438E-07	0.000993196	test	8.30923E-06	0.002882574	-0.000295896	0.94%	2.70962E-05
HES	elman	sentiment_m1	validation	9.19988E-06	0.00303313	test	1.88909E-06	0.001374441	-0.002603408	1.40%	1.92422E-05
HP	elman	sentiment_m1	validation	2.34418E-06	0.001531072	test	1.41077E-06	0.001187758	-0.002424238	0.58%	6.889E-06
KMI	elman	sentiment_m1	validation	1.44327E-05	0.003799035	test	1.20192E-05	0.003466878	-0.000804663	2.65%	9.18723E-05
MPC	elman	sentiment_m1	validation	6.4679E-06	0.002543206	test	4.9705E-05	0.007050176	0.004475356	1.70%	0.000119853
MRO	elman	sentiment_m1	validation	2.00918E-05	0.00448239	test	2.4106E-05	0.004909789	-0.000398091	1.20%	5.89175E-05
MUR	elman	sentiment_m1	validation	7.29311E-06	0.002700575	test	0.000169687	0.013026387	0.002942257	0.48%	6.25267E-05
NFX	elman	sentiment_m1	validation	9.02727E-06	0.003004541	test	3.10356E-05	0.005570959	0.004446519	0.60%	3.34258E-05
NOV	elman	sentiment_m1	validation	1.77189E-06	0.001331125	test	2.00076E-05	0.004472984	0.000917506	1.25%	5.59123E-05
OKE	elman	sentiment_m1	validation	1.8283E-05	0.004275858	test	3.20359E-05	0.005660024	0.004332364	0.80%	4.52802E-05
PSX	elman	sentiment_m1	validation	5.39742E-07	0.000734672	test	1.5293E-07	0.000391063	-0.001629952	2.55%	9.9721E-06
PXD	elman	sentiment_m1	validation	2.62789E-06	0.001621078	test	1.20019E-06	0.001095532	-0.001580343	4.78%	5.23664E-05
RIG	elman	sentiment_m1	validation	3.20358E-06	0.001789855	test	0.000148241	0.012175436	0.002415273	0.37%	4.50491E-05
RRC	elman	sentiment_m1	validation	1.85391E-05	0.00430571	test	1.73119E-05	0.004160751	0.000112819	0.68%	2.82931E-05
SE	elman	sentiment_m1	validation	1.79938E-06	0.00134141	test	2.43E-07	0.000492951	-0.001745335	2.53%	1.24717E-05
SLB	elman	sentiment_m1	validation	4.22883E-07	0.000650295	test	6.57553E-07	0.000810897	-0.001865363	8.19%	6.64124E-05
SWN	elman	sentiment_m1	validation	5.28955E-05	0.007272931	test	0.000185669	0.013626054	0.001534635	0.46%	6.26798E-05
TSO	elman	sentiment_m1	validation	5.51196E-06	0.002347757	test	1.16883E-06	0.001081126	-0.001389606	2.22%	2.4001E-05
VLO	elman	sentiment_m1	validation	2.50129E-06	0.001581547	test	3.19487E-07	0.000565232	-0.001734745	2.84%	1.60526E-05
WMB	elman	sentiment_m1	validation	0.000632759	0.025154703	test	0.003792647	0.061584471	0.05794406	1.87%	0.00115163
XEC	elman	sentiment_m1	validation	3.88223E-06	0.001970337	test	1.11761E-06	0.001057173	-0.001701143	0.86%	9.09169E-06
XLE	elman	sentiment_m1	validation	3.49233E-07	0.00059096	test	4.1791E-07	0.000646459	-0.001367147		0
XOM	elman	sentiment_m1	validation	2.86713E-07	0.000535456	test	9.39536E-08	0.000306519	-0.00196712	16.80%	5.14951E-05
										total	**0.002900439**

Table 21 –S_{t-1} scenario (9) results

4.1.2.3 The Sentiment Momentum Scenario

Table 22 shows the MSE/RMSE validation and test results when the sentiment momentums SM_{t-1} and SM_{t-2} are added to the equation. The column named 'delta SMM1M2-Base' corresponds to the difference in RMSEs between the base results RMSE and the sentiment momentum scenario RMSE. Negative figures are highlighted in green. They show a reduction in RMSE due to the addition of the sentiment momentum variables.

In this case, there is a reduction in test RMSE for 19 stocks out of the 36. These 19 stocks represent 65.42% of total index weight. The sum of the constituents' weighted RMSE is 0.003191146.

There is also a reduction in test RMSE for XLE index. The RMSE moves from 0.002013606 in the base scenario to 0.00069079 in the $SM_{t-1} + SM_{t-2}$ scenario.

code	model_name	scenario	type	mse	rmse	type	mse	rmse	delta SMM1M2 - Base	weights	weighted rmse
EOG	jordan	sentiment_momentum_mm1m2	validation	1.26247E-06	0.001123596	test	3.88143E-06	0.001970135	-0.00049639	4.64%	9.14142E-05
HAL	jordan	sentiment_momentum_mm1m2	validation	6.16404E-07	0.000785114	test	2.48947E-07	0.000498946	-0.000196023	3.66%	1.82614E-05
NBL	jordan	sentiment_momentum_mm1m2	validation	2.99022E-06	0.001729226	test	1.4098E-06	0.001187348	-0.000631762	1.58%	1.87601E-05
OXY	jordan	sentiment_momentum_mm1m2	validation	4.42033E-07	0.000664855	test	1.73887E-06	0.001318663	-0.000194393	3.14%	4.1406E-05
APA	elman	sentiment_momentum_mm1m2	validation	5.38433E-06	0.002320416	test	1.13616E-05	0.00337069	-0.000513239	1.91%	6.43802E-05
APC	elman	sentiment_momentum_mm1m2	validation	8.08767E-06	0.002843883	test	6.09037E-06	0.002467868	-0.000696429	2.98%	7.35425E-05
BHI	elman	sentiment_momentum_mm1m2	validation	6.18371E-06	0.002486706	test	9.95769E-06	0.00315558	0.000753004	2.43%	7.66806E-05
CHK	elman	sentiment_momentum_mm1m2	validation	0.000411966	0.020296941	test	0.000547635	0.02340161	-0.001361834	0.47%	0.000109988
COG	elman	sentiment_momentum_mm1m2	validation	1.77473E-05	0.004212751	test	2.63232E-05	0.005130615	0.003985325	1.52%	7.79854E-05
COP	elman	sentiment_momentum_mm1m2	validation	3.79659E-06	0.001948484	test	3.61728E-06	0.001901915	-0.000475732	3.12%	5.93397E-05
CVX	elman	sentiment_momentum_mm1m2	validation	2.87533E-06	0.00169568	test	4.8817E-06	0.002209457	-0.000301816	14.81%	0.000327221
CXO	elman	sentiment_momentum_mm1m2	validation	6.31272E-06	0.002512512	test	1.22911E-05	0.003505862	-0.000158095	1.30%	4.55762E-05
DVN	elman	sentiment_momentum_mm1m2	validation	1.34492E-05	0.003667317	test	4.3922E-05	0.00662737	0.0011854	1.88%	0.000124595
EQT	elman	sentiment_momentum_mm1m2	validation	1.24705E-05	0.003531365	test	1.31329E-05	0.003623933	-0.000107279	0.79%	2.86291E-05
FTI	elman	sentiment_momentum_mm1m2	validation	3.68056E-06	0.001918478	test	7.06667E-06	0.002658321	-0.000520148	0.94%	2.49882E-05
HES	elman	sentiment_momentum_mm1m2	validation	9.80629E-06	0.0031315	test	2.65977E-05	0.005157299	0.00117945	1.40%	7.22022E-05
HP	elman	sentiment_momentum_mm1m2	validation	2.98943E-06	0.001728996	test	5.43633E-06	0.002331594	-0.001280401	0.58%	1.35232E-05
KMI	elman	sentiment_momentum_mm1m2	validation	1.49277E-05	0.003863636	test	1.77158E-05	0.004209018	-6.25239E-05	2.65%	0.000111539
MPC	elman	sentiment_momentum_mm1m2	validation	5.11662E-06	0.002261995	test	1.13276E-05	0.003365655	0.000790835	1.70%	5.72161E-05
MRO	elman	sentiment_momentum_mm1m2	validation	1.69012E-05	0.004111112	test	0.000403828	0.020095481	0.014787602	1.20%	0.000241146
MUR	elman	sentiment_momentum_mm1m2	validation	9.04276E-06	0.003007118	test	0.000158962	0.012608016	0.002523885	0.48%	6.05185E-05
NFX	elman	sentiment_momentum_mm1m2	validation	3.19018E-05	0.005648167	test	4.9306E-05	0.007021826	0.005897386	0.60%	4.2131E-05
NOV	elman	sentiment_momentum_mm1m2	validation	3.43824E-06	0.001854249	test	2.31884E-05	0.004815432	0.001259954	1.25%	6.01929E-05
OKE	elman	sentiment_momentum_mm1m2	validation	1.90792E-05	0.004367976	test	1.09973E-05	0.003316214	0.001988554	0.80%	2.65297E-05
PSX	elman	sentiment_momentum_mm1m2	validation	2.9163E-06	0.001707718	test	5.36014E-06	0.002315198	0.000294183	2.55%	5.90375E-05
PXD	elman	sentiment_momentum_mm1m2	validation	3.44838E-06	0.001856981	test	1.45162E-05	0.003810008	0.001134133	4.78%	0.000182118
RIG	elman	sentiment_momentum_mm1m2	validation	4.62669E-06	0.002150975	test	0.000103641	0.010180426	0.000420264	0.37%	3.76676E-05
RRC	elman	sentiment_momentum_mm1m2	validation	1.87136E-05	0.004325922	test	8.03385E-06	0.002834404	-0.001213528	0.68%	1.9274E-05
SE	elman	sentiment_momentum_mm1m2	validation	2.86831E-06	0.00169361	test	2.50624E-06	0.00158311	-0.000655176	2.53%	4.00527E-05
SLB	elman	sentiment_momentum_mm1m2	validation	6.43362E-06	0.002536457	test	8.2859E-06	0.002878524	0.000202265	8.19%	0.000235751
SWN	elman	sentiment_momentum_mm1m2	validation	5.86402E-05	0.007657689	test	0.000171642	0.013101225	0.001009806	0.46%	6.02656E-05
TSO	elman	sentiment_momentum_mm1m2	validation	7.49381E-06	0.002737483	test	7.90498E-06	0.002811579	0.000340847	2.22%	6.24171E-05
VLO	elman	sentiment_momentum_mm1m2	validation	2.47586E-06	0.001573486	test	3.37726E-06	0.001837732	-0.000462245	2.84%	5.21916E-05
WMB	elman	sentiment_momentum_mm1m2	validation	0.000663628	0.025760971	test	9.93145E-05	0.009965668	0.006325257	1.87%	0.000186358
XEC	elman	sentiment_momentum_mm1m2	validation	5.10921E-06	0.002260357	test	8.52769E-06	0.00292022	0.00161905	0.86%	2.51139E-05
XLE	elman	sentiment_momentum_mm1m2	validation	5.47508E-07	0.000739938	test	4.77191E-07	0.00069079	-0.001322816		0
XOM	elman	sentiment_momentum_mm1m2	validation	2.62116E-06	0.001619	test	4.67214E-06	0.002161513	-0.000112126	16.80%	0.000363134
										total	**0.003191146**

Table 22 –$SM_{t-1} + SM_{t-2}$ scenario (10) results

4.1.2.4 Conclusion

These results show that sentiment has a positive impact on the index volatility prediction. The sentiment variable (S_{t-1}) seems to have a greater prediction power impact than the sentiment momentums ($SM_{t-1} + SM_{t-2}$), at both at the index and the constituents' level.

The proxy for the index volatility was not re-generated from the constituent (c.f. recommendations and potential extensions of the next section). However, 87% of index weight show an improvement of volatility prediction, after the addition of sentiment (S_{t-1}). This result suggests that the proxy index volatility prediction, generated from the constituents, should be improved when sentiment is present.

The next step would be to compare the volatility prediction improvement when the sentiment is applied at the index level, and when it is applied at the individual constituents' level.

Chapter 5 Conclusion

This thesis proposes a new approach to establish the prediction power of sentiment when it comes to predict an index trend. This experiment involves the XLE index and its 36 constituents.

First, the experiment devises a "2-way" feature selection (a Wrapper and a Filter method). Each of the selected features set is fitted against a basket of 8 machine learning models (such as discriminant models, random forest, recurrent neural networks. etc.). This mechanism provides the benefit of optimising the test accuracy rate for each asset's trend prediction. The best model is retained for each asset. This forms the base scenario (where the sentiment variable is excluded).

The next step consists in regenerating these test accuracy rates for all the assets, with extra variables relating to the sentiment, the sentiment momentum and their lags.

The results show that the market trend prediction of the index cannot be improved by the addition of the sentiments or sentiment momentums variables at the index level. The same result is obtained when the proxy index trend prediction is regenerated from the constituents. Although some of the stocks did show small improvements in trend prediction, between 1% and 3% depending the scenario, these results were not statistically significant.

At first sight, these results looked disappointing, compared to some of the current literature review's results. Especially the studies produced by Meesad and Li (2014) and Halgamuge (2007). However, it is important to underline that our research is more robust and complete than the ones previously mentioned.

First, our study uses sentiment build from an independent and complex engine generator. The literature only proposes a limited set of data sources coupled with a Bag-Of-Words approach, which has known context capability limitations. Second, in the literature, the list of selected technical indicators is only restricted to a few. In this experiment, the base scenario variable contains a list of more than 50 technical indicators. Third, the mentioned pieces of literature implement machine learning tools that are not adapted for time series; such as the use of cross-validation or a unique test set. In this thesis, a sliding time windows has been implemented over a period of more than 2 years. Therefore, our study proposes a more systematic and rigorous approach towards trend prediction. It produces results that are in line with the conclusions inferred from a robust statistical research performed by Olaniyan R. et al (2015), i.e. sentiment or sentiment momentum has no predictive power on trend.

A second set of experiments was put in place to establish the predictive power of sentiment on volatility forecast. The explanatory variables were composed of i) lagged EGARCH volatility variables, and ii) lagged volume variables. This time, only the Jordan and Elman RNNs were considered. In this instance, the results were positive. First, the addition of sentiment at the index level shows a significant reduction of the volatility prediction RMSE. This means the proxy index sentiment has a positive impact on the volatility prediction. Second, it also shows that the positive effect of sentiment was present at the individual stock level.

The next section relates to the recommendations and potential extensions that could be added to this study:

- The second experiment did not go to the extent of comparing the sum of the index's constituents' volatility prediction vs the index volatility prediction, when sentiment is present. It should be noted that the sum of the constituents' weighted volatility does not generated a proxy for the index volatility. Instead the following formula needs to be implemented:

$$\sigma_w^2 = w^T S\, w = [w_1,\ldots,w_2] \begin{bmatrix} \sigma_{11} \ldots \ldots \sigma_{1N} \\ \sigma_{21} \ldots \ldots \sigma_{2N} \\ \ldots \ldots \ldots \ldots \ldots \\ \sigma_{1N} \ldots \ldots \sigma_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ \ldots \\ \ldots \\ w_N \end{bmatrix}$$

  where $w_1,\ldots,w_2$ are the weights and $\sigma_{11} \ldots . \sigma_{1N}$ are the volatilities.

- Although the second experiment considered the EGARCH, other ARCH processes are available in the literature, such as: the generalized autoregressive conditional heteroskedasticity(GARCH), the Threshold ARCH (TARCH), the asymmetric power ARCH (APARCH) or the nonlinear GARCH (Brownlees, 2012). It could be interesting to investigate the impact of sentiment with these different settings.

- In their research, Wang et Al (2016) concluded that the stochastic Backpropagation through time improves the accuracy of financial time series forecasting. An improvement of our proposed framework could involve the replacement of the current Elman and Jordan RNNs with their stochastic version. This aim would be to establish whether the stochastic Backpropagation through time can improve volatility prediction when the sentiment is added.

- It would be beneficial to perform the same market trend prediction test on the S&P500 index and its constituents to establish whether other stocks' trend prediction are more sensitive to the sentiment data than the ones in this experiment.

- *Quandl* does not provide sentiment for indices, only for stocks. The current approach consisted in building a proxy index sentiment from the stock list. This experiment assumes the index constituents and weights are constant over all time periods. A better approach would be to build a time varying proxy index sentiment based on the historical daily index's constituents, and their respective weights. However, this has a cost, as this historical information is not freely available on the web.

- It would be useful to build an engine that generates sentiment scores and compare the results with *Quandl* to establish whether the *Quandl* sentiment score quality can be improved. At the very least, it would be useful to build sentiment where Quandl have not published them.

References

Abdulkarim S.A. 2016. “Time Series Prediction with Simple Recurrent Neural Networks”. *Bayero Journal of Pure and Applied Sciences.* 9(1): pp.19–24.

Accern. “Alpha One Guide Book – The most comprehensive trading analytics data set, Actionable Trading Analytics”. http://joshua.mcelfre.sh/AlphaOne_UserGuide.pdf (accessed February 1, 2009).

Accern. “Alphaone News Sentiment”. https://www.quantopian.com/data/accern/alphaone. (Accessed 12 Jun 2017).

Altman, D.G. 1991. *Practical Statistics for Medical Research.* Chapman & Hall. London.

Ang A., Timmerman A. (2011), “Regime Changes and Financial Markets”. *Netspar Discussion Papers*. DP–06/20011–068. pp1–19.

Brownlees C., Engle R., and Kelly B. (2012), “A practical guide to volatility forecasting through calm and storm”, *The Journal of Risk*. Volume 14/Number 2. pp3-22

Bachelier L. 1900, “The Theory of Speculation”. http://www.radio.goldseek.com/bachelier–thesis–theory–of–speculation–en.pdf. (Accessed 12 Jul 2017).

Baek E., and Brock W. 1992. “A general test for nonlinear Granger causality: bivariate model”. Working paper. Iowa State University.

Chawla NV., Bowyer KW, Hall LO., Kegelmeyer WP.2002. “SMOTE: Proxy Minority Over-Sampling Technique”. *Journal of Artificial Intelligence Research*. pp321-357.

Cohen J. (1960). “A Coefficient of Agreement for Nominal Scales”. *Educational and Psychological Measurement*. pp.20-37.

Fama E. 1964. “The behavior of stock market prices”. *Journal of Business*. Volume 38, Issue 1. p34–105.

Frank J. Massey Jr. 1951. “The Kolmogorov–Smirnov Test for Goodness of Fit.” *Journal of the American Statistical Association*. Vol. 46, No. 253. pp. 68– 78

Gilbert E., and Karahalios K 2010. “Widespread worry and the stock market”. *In Proceedings of the 4th International Conference on Weblogs and Social Media*.pp58–65.

Giles D.E. 2007. “Some Properties of Absolute Returns as a Proxy for Volatility”, *Econometrics Working Paper EWP0706,* University of Victoria, ISSN 1485-6441, p3.

Granger C.W.J. 1969. *“*Investigating Causal Relations by Econometric Models and Cross–spectral Methods”. *Econometrica*. Vol. 37, No. 3. pp. 424–438.

Grishman 2014. “Natural Language Processing”. http://www.cs.nyu.edu/courses/spring14/CSCI–GA.2590–001/Lecture2.html *(Accessed: April 2014).*

Halgamuge S.K. 2007. “Combining News and Technical Indicators in Daily Stock Price Trends Prediction”. *Conference: Advances in Neural Networks – ISNN 2007*. 4th International Symposium on Neural Networks. ISNN 2007 Proceedings. Part III.

Hastie T., Bujas A., and Tibshirani R. 1995. *“*Penalized Discriminant Analysis”, *The Annals of Statisitcs*. Vol. 23. No 1. pp.73–192.

Haykin S. 1999, *Neural Networks – A Comprehensive Foundation*. Second Edition. Printice Hall International Inc. New Jersey. pp156–247.

Henkel S.J., Martin J.S., and Nardari F. 2011. “Time–varying Short–Horizon Predictability”. *Journal of Financial Economics*, 99, pp.560–580.

Imandoust S.B., and Bolandraftar M. 2014. “Forecasting the direction of stock market index movement using three data mining techniques: the case of Tehran Stock Exchange”. *Int. Journal of Engineering Research and Applications*. ISSN: 2248–9622, Vol. 4, Issue 6(Version 2). pp.106–117.

Indicator Reference. 2016. http://www.fmlabs.com/reference/default.htm, (Accessed 27 March 2017).

James G., Witten D., and Hastie T., Tibshirani R. 2015, “An Introduction to Statistical Learning with R”. *Springer*. USA. pp.175–197, pp.340–350.

Joshi K., Bharathi H.N., and Rao J. 2013., “Stock Trend Prediction Using News Sentiment Analysis”, https://arxiv.org/ftp/arxiv/papers/1607/1607.01958.pdf (Accessed: 13–Apr–2017).

Keim D. 2006. “Financial Market Anomalie”. *http://finance.wharton.upenn.edu/~keim/research/NewPalgraveAnomalies(May302006).pdf* .(Accessed: 14–Apr–2017).

Kuhn M., Johnson K. 2013. “Applied Predictive Modelling*”. Springer*. USA. pp.69–73.

Lewis N.D. 2017, *Neural Networks for Time Series Forecasting with R: An* Intuitive Step *by* Step. Blueprint for Beginners. Kindle Edition. pp.141–179.

Niederhoffer V., and Osborne M. F. M. 1966. “Market making and reversal on the stock exchange”. *Journal of the American Statistical Association*. 61(316). pp.897– 916.

Nikolaev N. (n.d.), “Multilayer Perceptrons (Continuation)”. *http://homepages.gold.ac.uk/nikolaev/311bpr.htm.*(Accessed: 04–April–2017).

Maciel L.S., and Ballini R. n.d. “Design a neural network for time series financial Forecasting: accuracy and robustness analysis”. https://www.academia.edu/4472250/DESIGN_A_NEURAL_NETWORK_FOR_TIME_SERIES_FINANCIAL_FORECASTING_ACCURACY_AND_ROBUSTNESS_ANALISYS *(*Accessed: 13–Jul–2017*).*

Malkiel B. G. 1973. “A Random Walk Down Wall Street”. *https://www.academia.edu/10850809/A_Random_Walk_Down_Wall_Street_The_Time–Tested_Strategy_for_Successful_Investing.*(Accessed: 13–Apr–2017)

Meesad P., and Li J. 2014. “Stock trend prediction relying on text mining and sentiment analysis with tweets*”. In: 2014 Fourth World Congress on Information and Communication Technologies (WICT)*. pp. 257–262.

Murphy J. 1986. *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications,* New York Institute of Finance, pp.225–262.

Olaniyan R., Stamate D., Lahcen O, and Logofatu D. 2015. “Sentiment and stock market volatility predictive modelling – A hybrid approach”, In: Eric Gaussier; Longbing Cao; Patrick Gallinari; James Kwok; Gabriela Pasi and Osmar Zaiane, eds. *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on.* Paris: IEEE, pp. 1–10.

Olaniyan R., Stamate D., and Logofatu D. 2015. *“*Social web–based anxiety index’s predictive information on S&P 500 revisited”, *Proceedings of the 3rd Intl. Symposium on Statistical Learning and Data Sciences*.

Parikh V. and Shah P. 2015. “Stock Prediction and Automated Trading System*”, Computer Science & Electronics Journals*. Vol– 6, Issue–1 Sep. pp.104–111.

Rechenthin M.D. 2014. “Machine–learning classification techniques for the analysis and prediction of high–frequency stock direction”. http://ir.uiowa.edu/cgi/viewcontent.cgi?article=5248&context=etd. (Accessed: 14–Apr–2017).

Sabri N.R. 2008. “The impact of trading volume on stock price volatility in the Arab economy”, *Journal of Derivatives & Hedge Funds*. Volume 14, Issue 3–4. pp 285–298

Schoutens L. 2003. *Levy Processes in Finance,* West Sussex, England, p3, pp.27–28.

Schumaker R. P. and Chen H. 2009. “Textual analysis of stock market prediction using breaking financial news:The AZFinText system”, *ACMTrans.Inform.Syst.27,* 2, Article12.

Senyurt G., and Subasi A. (n.d.), “Stock market movement direction prediction using tree algorithms” *http://eprints.ibu.edu.ba/1187/1/41.%20Stock%20market%20movement%20direction%20prediction%20using%20tree%20algorithms.pdf,* (Accessed: 14–Apr–2017).

Suresh A.S. 2013. “A Study on Fundamental and Technical Analysis”, *International Journal of Marketing, Financial Services & Management Research*, Vol.2, No. 5.

Torgot L. 2011. *Data Mining with R: Learning with Case Studies*. Chapman & Hall/CRC. Florida. chp3. pp95–163.

Vaiz J.S., and Ramaswami M. 2016. “A Study on Technical Indicators in Stock Price Movement Prediction Using Decision Tree Algorithms”, *American Journal of Engineering Research – (AJER)*. Volume 5, Issue 12. pp–207–212.

Wang J., Wang J., Fang W., and Niu H. 2016, “Financial Time Series Prediction Using Elman Recurrent Random Neural Networks”, *Computational Intelligence and Neuroscience.* Article ID 4742515.

Wiersma Y, Huettmann F., Drew A.C. 2011. “Predictive Species and Habitat Modeling in Landscape Ecology”. https://books.google.co.uk/books?id=1V5gupaI5_IC&pg=PA147&lpg=PA147&dq=can+the+auc+and+kappa+disagree?&source=bl&ots=Pav9xYH5pT&sig=9cCUcQ5LLCAiOSkTqiw5BiUQ2qI&hl=en&sa=X&ved=0ahUKEwiPq67CrtHVAhXDCMAKHQ7MDt0Q6AEIKDAA#v=onepage&q=can%20the%20auc%20and%20kappa%20disagree%3F&f=false (Accessed 12 Aug 2017).

Appendices

Appendix 1 – *SentiWordNet* API limitations

The following python code shows that not some words, such as uptick or rally, return neither a *Positive* nor a *Negative* sentiment.

In [23]:
```
from nltk.corpus import sentiwordnet as swn

print ("***** bad ****")
senti = swn.senti_synsets("bad")
senti_synsets = list(senti)
senti_synset = senti_synsets[0]
print ("pos: " + str(senti_synset.pos_score()))
print ("neg: " + str(senti_synset.neg_score()))

print ("***** uptick ****")
senti = swn.senti_synsets("uptick")
senti_synsets = list(senti)
senti_synset = senti_synsets[0]
print ("pos: " + str(senti_synset.pos_score()))
print ("neg: " + str(senti_synset.neg_score()))

print ("***** rally *****")
senti = swn.senti_synsets("rally")
senti_synsets = list(senti)
senti_synset = senti_synsets[0]
print ("pos: " + str(senti_synset.pos_score()))
print ("neg: " + str(senti_synset.neg_score()))
```

```
***** bad ****
pos: 0.0
neg: 0.875
***** uptick ****
pos: 0.0
neg: 0.0
***** rally ****
pos: 0.0
neg: 0.0
```

Appendix 2 – XLE Basket Weights

The below table shows the list of the XLE stock constituents and their weights as of 14–Dec–2016.

Symbol	Company Name	Index Weight
XOM	Exxon Mobil Corp	16.80%
CVX	Chevron Corp	14.81%
SLB	Schlumberger Ltd	8.19%
PXD	Pioneer Natural Resources	4.78%
EOG	EOG Resources	4.64%
HAL	Halliburton Co	3.66%
OXY	Occidental Petroleum	3.14%
COP	ConocoPhillips	3.12%
APC	Anadarko Petroleum Corp	2.98%
VLO	Valero Energy Corp	2.84%
KMI	Kinder Morgan Inc	2.65%
PSX	Phillips 66	2.55%
SE	Spectra Energy Corp	2.53%
BHI	Baker Hughes Inc	2.43%
TSO	Tesoro Corp	2.22%
APA	Apache Corp	1.91%
DVN	Devon Energy Corp	1.88%
WMB	The Williams Companies Inc	1.87%

Symbol	Company Name	Index Weight
MPC	Marathon Petroleum Corp.	1.70%
NBL	Noble Energy Inc	1.58%
COG	Cabot Oil & Gas A	1.52%
HES	Hess Corp	1.40%
CXO	Concho Resources Inc	1.30%
NOV	National Oilwell Varco Inc	1.25%
MRO	Marathon Oil Corp	1.20%
FTI	FMC Technologies Inc	0.94%
XEC	Cimarex Energy Co	0.86%
OKE	ONEOK Inc	0.80%
EQT	EQT Corporation	0.79%
RRC	Range Resources Corp	0.68%
NFX	Newfield Exploration Co	0.60%
HP	Helmerich & Payne Inc	0.58%
MUR	Murphy Oil Corp	0.48%
CHK	Chesapeake Energy Corp	0.47%
SWN	Southwestern Energy Co	0.46%
RIG	Transocean Ltd	0.37%

Appendix 3 – The software Requirement

Software Dependencies	R – Version 3.3.3 & R Studio – Version 0.99.903 Python – 3.4.1
Hardware Dependencies	Hardware Windows 10 64bits platform, supported by an Intel Core i7–67000HQ processor / RAM: DDR4 8GB. As well as the Goldsmiths' clustered server.

Appendix 4 – How to Run the Code

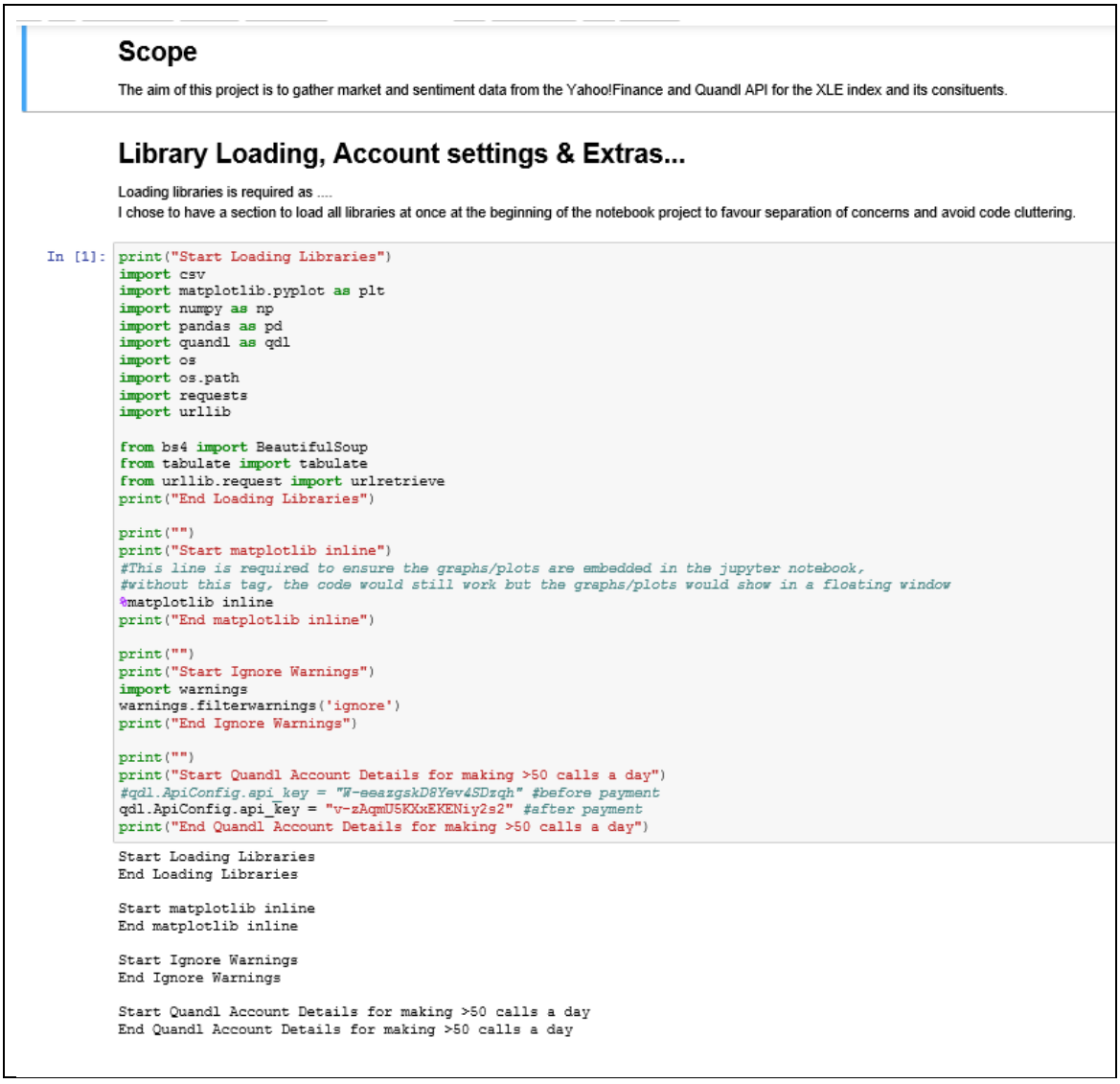
All services can be run independently. However, each service needs the data produced by the predecessor to complete successfully. The full code is available in the below appendices.

Data Cropper Service	../data.source.loaders/StockPriceAndSentimentCropper.ipynb
DataSet Generator Service	../asset.dataset.generator.batch.svc
Time Windows Generator Service	../ time.windows.generator.batch.svc
Feature Selection Generator Service	../feature.selection.generator.batch.svc
Machine Learning Models Service	Run all 'rmd' files present in the following folder to generate the base results for all assets/all models based on a 'permutated relief' feature selection model: ..\pc.run.original Run all 'rmd' files present in the following folder to generate the base results for all assets/all models based on a 'gini index' feature selection model: ..\server.run.original Run all 'rmd' files present in the following folders to generate the sentiment and sentiment mometum results for all assets/all models based on either a 'gini index' or 'relief' feature selection model (depending on set-up): ..\pc.run.sentiment ..\pc.run.sentiment.m1 ..\pc.run.sentiment.m2 ..\pc.run.sentiment.m3 .. \pc.run.sentiment.momentum.rmd .. \pc.run.sentiment.momentum.mm1 .. \pc.run.sentiment.momentum.mm1m2 ..\pc.run.sentiment.momentum.mm1m2m3
Dataset Explorer Service	../data.explorator.batch.svc

Appendix 5 – Python Data Source Service Code

The code is available under ../data.source.loaders/StockPriceAndSentimentCropper.ipynb. It is also pasted in this appendix for convenience.

It runs under Jupyter Notebook with a Python 3.5.2 kernel, as follows:
Start a windows command line:
>> cd ../data.source.loaders/
>> jupyter notebook "StockPriceAndSentimentCropper.ipynb"



Scope

The aim of this project is to gather market and sentiment data from the Yahoo!Finance and Quandl API for the XLE index and its consituents.

Library Loading, Account settings & Extras...

Loading libraries is required as ....
I chose to have a section to load all libraries at once at the beginning of the notebook project to favour separation of concerns and avoid code cluttering.

In [1]:
```
print("Start Loading Libraries")
import csv
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import quandl as qdl
import os
import os.path
import requests
import urllib

from bs4 import BeautifulSoup
from tabulate import tabulate
from urllib.request import urlretrieve
print("End Loading Libraries")

print("")
print("Start matplotlib inline")
#This line is required to ensure the graphs/plots are embedded in the jupyter notebook,
#without this tag, the code would still work but the graphs/plots would show in a floating window
%matplotlib inline
print("End matplotlib inline")

print("")
print("Start Ignore Warnings")
import warnings
warnings.filterwarnings('ignore')
print("End Ignore Warnings")

print("")
print("Start Quandl Account Details for making >50 calls a day")
#qdl.ApiConfig.api_key = "W-eeazgskD8Yev4SDzqh" #before payment
qdl.ApiConfig.api_key = "v-zAqmU5KXxEKENiy2s2" #after payment
print("End Quandl Account Details for making >50 calls a day")
```

```
Start Loading Libraries
End Loading Libraries

Start matplotlib inline
End matplotlib inline

Start Ignore Warnings
End Ignore Warnings

Start Quandl Account Details for making >50 calls a day
End Quandl Account Details for making >50 calls a day
```

Common Functions

In [2]:
```
print ("Start Common Shared functions")
def mkdir(directory_full_path):
    if not os.path.exists(directory_full_path):
        os.makedirs(directory_full_path)

def write_to_csv(file_full_path, rows, read_write_option='w', mutiple_row=True):
    with open(file_full_path, read_write_option, newline='') as mycsvfile:
        thedatawriter = csv.writer(mycsvfile)
        if (mutiple_row):
            thedatawriter.writerows(rows)
        else:
            thedatawriter.writerow(rows)

def delete_file(file_name_full_path):
    if (os.path.exists(file_name_full_path)):
        os.remove(file_name_full_path)
print ("End Common Shared functions")
```

```
Start Common Shared functions
End Common Shared functions
```

Stock Prices Cropper Functions

In [3]:
```
print ("Start Stock Prices Cropper Functions")
#Functions...
base_url = "http://ichart.finance.yahoo.com/table.csv?s="
input_path = output_path = "C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/MktSentAnalysis/Data"

def make_url(ticker_symbol):
    return base_url + ticker_symbol

def make_output_filename(ticker_symbol, directory="Unknown"):
    return output_path + "/" + directory + "/", output_path + "/" + directory + "/" + ticker_symbol + ".csv"

def make_input_filename(file_name):
    return input_path + "/" + file_name + ".csv"

def pull_historical_data(ticker_symbol, directory="Unknown"):
    try:
        directory_full_path, file_full_path = make_output_filename(ticker_symbol, directory)
        ticker_url = make_url(ticker_symbol)
        mkdir(directory_full_path)
        urlretrieve(ticker_url, file_full_path)
    except Exception as e: # catch *all* exceptions
        directory_full_path, file_full_path = make_output_filename(ticker_symbol, directory)
        outfile = open(file_full_path, "w")
        print(e)
        outfile.write(str(e))
        outfile.close()

print ("End Stock Prices Cropper Functions")
```

```
Start Stock Prices Cropper Functions
End Stock Prices Cropper Functions
```

Run Stock Prices Cropper...

The data is sourced for free from Yahoo, via the Yahoo API.

In [5]:
```
#get the Data1 from beg of time....
print ("Start Cropping...")
file_name = make_input_filename("XLE_Stocks")
df = pd.read_csv(file_name)  # Read the file
#for each symbol in the input file...
for index, row in df.iterrows():
    ticker = row['Symbol']
    print ("Running " + ticker + "...")
    #get the data from yahoo and store into CSV
    pull_historical_data (ticker, "Prices/XLE_Stocks")
print ("End Cropping...")

print ("Start Cropping Index...")
pull_historical_data ('XLE', "Prices/XLE_Indices")
print ("Start Cropping Index...")
```

```
Start Cropping Index...
Start Cropping Index...
```

Run Mkt Sentiment Data Cropper...

The data is sourced from the Quandl databases (for a fee)... For full details on the dB access, Python API and other pieces of information relating to Quandl, please visit https://www.quandl.com/

In [ ]:
```
#API details  => https://www.quandl.com/tools/python

def get_quandl_sentiment_to_csv(input_file_name, prefix, ouput_dir):
    file_name = make_input_filename(input_file_name)
    df = pd.read_csv(file_name)  # Read the file
    #for each symbol in the input file...
    for index, row in df.iterrows():
        ticker = row['Symbol']
        print(ticker)
        full_ticker_name = prefix+"/"+ticker
        try:
            df = qdl.get(full_ticker_name)
            df.to_csv(ouput_dir+ticker+'.csv')
        except:
            print ('Could not load: ' + full_ticker_name)
            pass

print ("Start Sentiment Cropping...")
get_quandl_sentiment_to_csv("US_Stocks","AOS", "Data/Sentiments/US_Stocks/")
print ("End Sentiment Cropping...")
print ("Start Rating Cropping...")
get_quandl_sentiment_to_csv("US_Stocks","ZRH", "Data/Ratings/US_Stocks/")
print ("End Rating Cropping...")
```

Code - ../data.source.loaders/StockPriceAndSentimentCropper.ipynb

Appendix 6 – The DataSet Generator Code

This code corresponds to the generation of explanatory and response variable dataset that is used by the machine learning models. The code can be found in the following places:

Code Location	Description
../asset.dataset.generator.batch.svc.rmd	This is the entry point that delegates into the asset.dataset.generator.r for each asset.
../data.generator.functions/asset.dataset.generator.r	It is a wrapper component that contains logic to glue technical and sentiment data produced by explanatory.variables.r and response.variables.r
../data.generator.functions/explanatory.variables.r	It is responsible for generating the list of explanatory variables (i.e. lag prices and technical indicators)
../data.generator.functions/response.variables.r	It is responsible for generating the response variable (i.e. the direction)

````
---
#Markdown doc generation: http://rmarkdown.rstudio.com/word_document_format.html
#Markdown cheat sheet:
#http://nestacms.com/docs/creating-content/markdown-cheat-sheet
#http://rmarkdown.rstudio.com/authoring_basics.html
title: "Asset DataSet Geneator Service"
author: Frederic Marechal
date: April-Sep, 2017
output: word_document
---
################################################################################
################################################################################
#######             Clean-Up
################################################################################
################################################################################
```{r cleanup}
#Clear console
cat("\014")
```
################################################################################
################################################################################
#######             Directory Config
################################################################################
################################################################################
```{r working_directory_config}
is_server_run = FALSE
if(is_server_run){
  setwd('/host/dsm1/fmare001/stats/svm/deliverables')
}else{
  setwd('C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/Thesis/Finance/z_Code')
}
#index_name = 'XLF'
index_name = 'XLE'
input_dir = paste(getwd(), "/data/", index_name,sep="")

index_constituents_df = read.csv(paste(input_dir, "/index_constituents.csv",sep=""),header=TRUE)
input_dir = paste(getwd(), "/data/",index_name,"/raw/market.data/prices/",sep="")
input_dir_sentiment = paste(getwd(), "/data/",index_name,"/raw/sentiment.data/",sep="")
input_dir_fundamental = paste(getwd(), "/data/",index_name,"/raw/fundamental.data/",sep="")
output_dir = paste(getwd(), "/data/",index_name,"/processed/",sep="")
#epsilon = 0.0025
epsilon = 0
```

################################################################################
################################################################################
#######             Load Dependencies
````

```
################################################################################
################################################################################
```{r load_depedencies}
source("utilities.r")
source("data.generator.functions/asset.dataset.generator.r")
source("data.generator.functions/explanatory.variables.r")
source("data.generator.functions/response.variables.r")
source("prepocessing.functions.r")

library(foreach) #to allow parallel processing
# need to install doMC via install.packages("doMC", repos="http://R-Forge.R-project.org")
library(doMC) #install and register a parallel backend, see https://www.r-bloggers.com/parallel-r-loops-for-windows-and-linux/
registerDoMC(6) # number of parallel processes
```

##
##
#######   Directory Clean-up
##
##
```{r load_directory_cleanup}
delete_all_files_in_directory(paste(output_dir,"final/", sep=""))
delete_all_files_in_directory(paste(output_dir,"first_cut/", sep=""))
```

##
##
#######   DataSet Generator As Batch
##
##
```{r dataset_generator}
#%do%: non parallel
#%dopar%: parallel version
#Run the stocks...
foreach(i=1:dim(index_constituents_df)[1]) %dopar% {
  stock_name = as.character(index_constituents_df[1][[1]][[i]])
  print(stock_name)
  if (length(stock_name) >0){
    dataset_generator(input_dir,input_dir_sentiment, input_dir_fundamental, output_dir, stock_name, epsilon)
  }
}
#Run the index...
dataset_generator(input_dir,input_dir_sentiment, input_dir_fundamental, output_dir, index_name, epsilon, TRUE)
```
```

Code - ../asset.dataset.generator.batch.svc.rmd

```
#author: Frederic Marechal
#date: April-Sep, 2017
dataset_generator = function (input_dir, input_dir_sentiment, input_dir_fundamental, output_dir, stock_name, epsilon, is_index=FALSE){
  ################################################################################
  ################################################################################
  #######              DataSet Generator
  ################################################################################
  ################################################################################
  print_section_info(paste("Start analysis for Stock:", stock_name, ""))
  file_path = paste(paste(input_dir,stock_name, sep="")[[1]], ".csv", sep="")[[1]]
  if (!file.exists(file_path)){
    #stock mkt price file is missing... prompt the user...
    cat("WARNING ---> FILE: ", file_path, " MKT DATA COULD NOT BE LOADED!\n",sep="")
    save_to_csv(NULL, output_dir,paste("/error/market.data/",stock_name, ".csv",sep = ""))
  } else {
    #Step1 - Get the stock market price data
    df = load_from_csv(input_dir, stock_name )
```

```
  df = generate_index(df)
  df = generate_date(df)
  #Explanatatory
  df = generate_technical_indicators(df)
  df = generate_lagged_prices(df)
  #Step 2 - Get the sentiment data & add to the final dataframe
  sentiment_file_path = paste(paste(input_dir_sentiment,stock_name, sep="")[[1]], ".csv", sep="")[[1]]
  if (!file.exists(sentiment_file_path)){
   #stock sentiment file is missing... prompt the user...
   cat("WARNING ---> FILE: ", file_path, " SENTIMENT DATA COULD NOT BE LOADED!\n",sep="")
   save_to_csv(NULL, output_dir,paste("/error/sentiment.data/",stock_name, ".csv",sep = ""))
  } else {
   #Get the stock market sentiment data
   sentiment_df = load_from_csv(input_dir_sentiment, stock_name )
   df = generate_sentiment(df, sentiment_df)
   df = generate_lagged_sentiments(df)
   df = generate_sentiment_momentum(df)
   df = generate_sentiment_momentum_pos_neg(df)
   df = generate_sentiment_pos_neg(df)
  }
  if (is_index == FALSE)
  {
   #Step 3 - Get the fundamental data & add to the final dataframe
   fundamental_file_path = paste(paste(input_dir_fundamental,stock_name, sep="")[[1]], ".csv", sep="")[[1]]
   if (!file.exists(fundamental_file_path)){
    #stock sentiment file is missing... prompt the user...
    cat("WARNING ---> FILE: ", file_path, " FUNDAMENTAL DATA COULD NOT BE LOADED!\n",sep="")
    save_to_csv(NULL, output_dir,paste("/error/fundamental.data/",stock_name, ".csv",sep = ""))
   } else {
    #Get the stock market sentiment data
    fundamental_df = load_from_csv(input_dir_fundamental, stock_name )
    df = generate_fundamental(df, fundamental_df)
   }
  }
  #Step 4 - Generate Response variable
  df = generate_log_returns_and_direction(df, epsilon)
  #Step 5 - Last actions...
  #Remove first 200days => remove NA for tech indicators and
  #Remove last date, has it has a NA for log_returns log (Pt+1/Pt) => Pt+1 is not available
  df_final = df[(201:dim(df)[1]-1),]

  #Remove mising Values
  #res = missing_data_removal(stock_name, df, output_dir)
  #df = res$dataset
  #Reduce the datasest cols
  #Select the list of attributes and explained variable used for the mdel building

  if (is_index == FALSE)
  {
   #The stock case....
   if("Article_sentiment" %in% colnames(df)){
    df_final =  subset(df_final, select=c(
                Log_returns,
                Direction,
                Direction_Flag,
                Close,
Close_price_1day_lag,Close_price_2day_lag,Close_price_3day_lag,Close_price_4day_lag,Close_price_5day_lag,
                Sma20,Sma50,Sma100,Sma200,
                Ema20,Ema50,Ema100,Ema200,
                Roc_1day,Roc_2day,Roc_5day, Roc_10day, Roc_20day,
                Momentum_5day,Momentum_10day,Momentum_20day,
                MomentumAbs_5day, MomentumAbs_10day,MomentumAbs_20day,
                Wpr_5day, Wpr_10day,Wpr_20day,
                Rsi_5day,Rsi_10day,Rsi_14day,Rsi_20day,
                Cmo_5day,Cmo_10day,Cmo_20day,
                Atr_tr,Atr_atr, Atr_trueHigh, Atr_trueLow,
                Smi_smi,Smi_signal,
```

```
                    Stoch_fastk,Stoch_fastd,Stoch_slowd,
                    Macd_macd,Macd_signal,
                    Bb_dn,Bb_mavg, Bb_up, Bb_pctB,
                    Volatility,
                    Mfi,
                    Sar,
                    Volume,
                    Price_earning_ratio,
                    Article_sentiment,
                    Article_sentiment,
                    Article_sentiment_1day_lag,
                    Article_sentiment_2day_lag,
                    Article_sentiment_3day_lag,
                    Article_sentiment_4day_lag,
                    Article_sentiment_momentum,
                    Article_sentiment_momentum_minus_1_day,
                    Article_sentiment_momentum_minus_2_day,
                    Article_sentiment_momentum_minus_3_day,
                    Article_sentiment_momentum_pos,
                    Article_sentiment_momentum_neg,
                    Article_sentiment_momentum_minus_1_day_pos,
                    Article_sentiment_momentum_minus_1_day_neg,
                    Article_sentiment_momentum_minus_2_day_pos,
                    Article_sentiment_momentum_minus_2_day_neg,
                    Article_sentiment_momentum_minus_3_day_pos,
                    Article_sentiment_momentum_minus_3_day_neg,
                    Article_sentiment_pos,
                    Article_sentiment_neg,
                    Article_sentiment_minus_1_day_pos,
                    Article_sentiment_minus_1_day_neg,
                    Article_sentiment_minus_2_day_pos,
                    Article_sentiment_minus_2_day_neg,
                    Article_sentiment_minus_3_day_pos,
                    Article_sentiment_minus_3_day_neg
                    ))}
    else{
      df_final =  subset(df_final, select=c(
              Log_returns,
              Direction,
              Direction_Flag,
              Close,
Close_price_1day_lag,Close_price_2day_lag,Close_price_3day_lag,Close_price_4day_lag,Close_price_5day_lag,
              Sma20,Sma50,Sma100,Sma200,
              Ema20,Ema50,Ema100,Ema200,
              Roc_1day,Roc_2day,Roc_5day, Roc_10day, Roc_20day,
              Momentum_5day,Momentum_10day,Momentum_20day,
              MomentumAbs_5day, MomentumAbs_10day,MomentumAbs_20day,
              Wpr_5day, Wpr_10day,Wpr_20day,
              Rsi_5day,Rsi_10day,Rsi_14day,Rsi_20day,
              Cmo_5day,Cmo_10day,Cmo_20day,
              Atr_tr,Atr_atr, Atr_trueHigh, Atr_trueLow,
              Smi_smi,Smi_signal,
              Stoch_fastk,Stoch_fastd,Stoch_slowd,
              Macd_macd,Macd_signal,
              Bb_dn,Bb_mavg, Bb_up, Bb_pctB,
              Volatility,
              Mfi,
              Sar,
              Volume,
              Price_earning_ratio
      ))}
  }else{
    #The index case....
    df_final =  subset(df_final, select=c(
            Log_returns,
            Direction,
            Direction_Flag,
```

```
                Close,
Close_price_1day_lag,Close_price_2day_lag,Close_price_3day_lag,Close_price_4day_lag,Close_price_5day_lag,
                Sma20,Sma50,Sma100,Sma200,
                Ema20,Ema50,Ema100,Ema200,
                Roc_1day,Roc_2day,Roc_5day, Roc_10day, Roc_20day,
                Momentum_5day,Momentum_10day,Momentum_20day,
                MomentumAbs_5day, MomentumAbs_10day,MomentumAbs_20day,
                Wpr_5day, Wpr_10day,Wpr_20day,
                Rsi_5day,Rsi_10day,Rsi_14day,Rsi_20day,
                Cmo_5day,Cmo_10day,Cmo_20day,
                Atr_tr,Atr_atr, Atr_trueHigh, Atr_trueLow,
                Smi_smi,Smi_signal,
                Stoch_fastk,Stoch_fastd,Stoch_slowd,
                Macd_macd,Macd_signal,
                Bb_dn,Bb_mavg, Bb_up, Bb_pctB,
                Volatility,
                Mfi,
                Sar,
                Volume,
                #no price earning ratio...
                Article_sentiment,
                Article_sentiment,
                Article_sentiment_1day_lag,
                Article_sentiment_2day_lag,
                Article_sentiment_3day_lag,
                Article_sentiment_4day_lag,
                Article_sentiment_momentum,
                Article_sentiment_momentum_minus_1_day,
                Article_sentiment_momentum_minus_2_day,
                Article_sentiment_momentum_minus_3_day,
                Article_sentiment_momentum_pos,
                Article_sentiment_momentum_neg,
                Article_sentiment_momentum_minus_1_day_pos,
                Article_sentiment_momentum_minus_1_day_neg,
                Article_sentiment_momentum_minus_2_day_pos,
                Article_sentiment_momentum_minus_2_day_neg,
                Article_sentiment_momentum_minus_3_day_pos,
                Article_sentiment_momentum_minus_3_day_neg,
                Article_sentiment_pos,
                Article_sentiment_neg,
                Article_sentiment_minus_1_day_pos,
                Article_sentiment_minus_1_day_neg,
                Article_sentiment_minus_2_day_pos,
                Article_sentiment_minus_2_day_neg,
                Article_sentiment_minus_3_day_pos,
                Article_sentiment_minus_3_day_neg
    ))
   }

   #Save
   cat (paste("\nSave processed dataset:", stock_name,"\n", sep=""))
   save_to_csv(df, output_dir,paste("/first_cut/",stock_name, ".csv",sep = ""))
   save_to_csv(df_final, output_dir,paste("/final/",stock_name, ".csv",sep = ""))
   #save_to_csv(data.frame(res$percentage_remove), output_dir,paste("/missing.removed/",stock_name, ".csv",sep =
""))
  }
 }
```

Code - ../ data.generator.functions / asset.dataset.generator.r

```
#author: Frederic Marechal
#date: April-Sep, 2017
################################################################################
################################################################################
#######              Clean-Up
################################################################################
################################################################################
cat("\014")
################################################################################
################################################################################
#######              Load Dependencies
################################################################################
################################################################################
library(quantmod)
################################################################################
################################################################################
#######              Explanatory Variable Generation Functions
################################################################################
################################################################################

#Add the Close price day lag to the analysis (  #Ensure the dataset is ordered in data asc order)
add_close_price_day_lag = function (stock_df, nb_days){
  res = generate_day_lag("Close","Close_price", stock_df, nb_days)
  return (res)
}
#Add the sentiment lag to the analysis (assumes the dataset is ordered in data asc order)
add_sentiment_day_lag = function (stock_df, nb_days){
  res = generate_day_lag("Article_sentiment","Article_sentiment", stock_df, nb_days)
  return (res)
}
generate_day_lag = function (source_col_name, target_col_name, stock_df, nb_days){
  #Generate the column dynalically based on the nb_days
  target_col_name = paste(target_col_name,"_", nb_days, "day_lag", sep="")
  for (k in 1:nrow(stock_df)) {
    if (k>nb_days){
      stock_df[k,target_col_name] = stock_df[k-nb_days,source_col_name]
    }
    else{
      stock_df[k,target_col_name] = NA
    }
  }
  return (stock_df)
}

generate_technical_indicators = function (df){
  #Sort by increasing dates (i.e. decreasing index)
  df = df[ order(df$Index , decreasing = TRUE ),]
  #Add a the indicators column (the average value of the High,Low and Close)
  df["Hlc"] = (df["High"] + df["Low"] + df["Close"])/3
  df["Hl"] = (df["High"] + df["Low"]) /2
  df["Sma20"] = SMA(df$Close, n=20)
  df["Sma50"] = SMA(df$Close, n=50)
  df["Sma100"] = SMA(df$Close, n=100)
  df["Sma200"] = SMA(df$Close, n=200)
  df["Ema20"] = EMA(df$Close, n=20)
  df["Ema50"] = EMA(df$Close, n=50)
  df["Ema100"] = EMA(df$Close, n=100)
  df["Ema200"] = EMA(df$Close, n=200)
  df["Roc_1day"] = ROC(df$Close, n=1, type = c("discrete"))
  df["Roc_2day"] = ROC(df$Close, n=2, type = c("discrete"))
  df["Roc_5day"] = ROC(df$Close, n=5, type = c("discrete"))
  df["Roc_10day"] = ROC(df$Close, n=10, type = c("discrete"))
  df["Roc_20day"] = ROC(df$Close, n=20, type = c("discrete"))
  df["Momentum_5day"] = momentum(df$Close, n=5)
  df["Momentum_10day"] = momentum(df$Close, n=10)
```

```
  df["Momentum_20day"] = momentum(df$Close, n=20)
  df["MomentumAbs_5day"] = abs(df["Momentum_5day"])
  df["MomentumAbs_10day"] = abs(df["Momentum_10day"])
  df["MomentumAbs_20day"] = abs(df["Momentum_20day"])
  df["Wpr_5day"] = WPR(df[,c("High","Low","Close")], n=5)
  df["Wpr_10day"] = WPR(df[,c("High","Low","Close")], n=10)
  df["Wpr_20day"] = WPR(df[,c("High","Low","Close")], n=20)
  df["Rsi_5day"] = RSI(df$Close, n=5)
  df["Rsi_10day"] = RSI(df$Close, n=10)
  df["Rsi_14day"] = RSI(df$Close, n=14)
  df["Rsi_20day"] = RSI(df$Close, n=20)
  df["Cmo_5day"] = RSI(df$Close, n=5)
  df["Cmo_10day"] = RSI(df$Close, n=10)
  df["Cmo_20day"] = RSI(df$Close, n=20)
  df["Atr_tr"] = ATR(HLC(df[,c("High","Low","Close")]))[][,1]
  df["Atr_atr"] = ATR(HLC(df[,c("High","Low","Close")]))[][,2]
  df["Atr_trueHigh"] = ATR(HLC(df[,c("High","Low","Close")]))[][,3]
  df["Atr_trueLow"] = ATR(HLC(df[,c("High","Low","Close")]))[][,4]
  df["Smi_smi"] = SMI(HLC(df[,c("High","Low","Close")]))[][,1]
  df["Smi_signal"] = SMI(HLC(df[,c("High","Low","Close")]))[][,2]
  df["Stoch_fastk"] = stoch(HLC(df[,c("High","Low","Close")]))[][,1]
  df["Stoch_fastd"] = stoch(HLC(df[,c("High","Low","Close")]))[][,2]
  df["Stoch_slowd"] = stoch(HLC(df[,c("High","Low","Close")]))[][,3]
  df["Macd_macd"] = MACD((df[,c("Close")]))[][,1]
  df["Macd_signal"] = MACD((df[,c("Close")]))[][,2]
  df["Bb_dn"] = BBands(HLC(df[,c("High","Low","Close")]))[][,1]
  df["Bb_mavg"] = BBands(HLC(df[,c("High","Low","Close")]))[][,2]
  df["Bb_up"] = BBands(HLC(df[,c("High","Low","Close")]))[][,3]
  df["Bb_pctB"] = BBands(HLC(df[,c("High","Low","Close")]))[][,4]
  df["Mfi"] = MFI(HLC(df[,c("High","Low","Close")]), df[,c("Volume")])
  df["Sar"] = SAR(df[,c("High","Close")])
  df["Volatility"] = volatility(OHLC(df[,c("Open","High","Low","Close")]), calc = "garman")
  return (df)
}

generate_lagged_prices =function(df){
  df = add_close_price_day_lag(df, 1)
  df = add_close_price_day_lag(df, 2)
  df = add_close_price_day_lag(df, 3)
  df = add_close_price_day_lag(df, 4)
  df = add_close_price_day_lag(df, 5)
  df = add_close_price_day_lag(df, 10)
  df = add_close_price_day_lag(df, 20)
  return (df)
}
generate_sentiment =function(df, sentiment_df){
  default_value = NA
  df_rows = dim(df)[1]
  #Add sentiment col headers df
  df["Article_sentiment"] = default_value
  df["Impact_score"] = default_value
  #Add an extra DateAsDate column in the sentiment dataframe
  sentiment_df = generate_date(sentiment_df)
  #Match the sentiment date to the Market data date and upgrade the market data data frame (df)
  for (i in 1:df_rows){
    sent_index = which(sentiment_df$DateAsDate == df$DateAsDate[i])
    if (length(sent_index) >0){
      df[i,]$Article_sentiment = sentiment_df$Article.Sentiment[sent_index]
      df[i,]$Impact_score = sentiment_df$Impact.Score[sent_index]
    }
  }
  return(df)
}

generate_lagged_sentiments = function(df){
  df = add_sentiment_day_lag(df, 1)
  df = add_sentiment_day_lag(df, 2)
```

```
  df = add_sentiment_day_lag(df, 3)
  df = add_sentiment_day_lag(df, 4)
  return (df)
}

generate_sentiment_momentum = function(df){
  df["Article_sentiment_momentum"] = df["Article_sentiment"] - df["Article_sentiment_1day_lag"]
  df["Article_sentiment_momentum_minus_1_day"] = df["Article_sentiment_1day_lag"] -
df["Article_sentiment_2day_lag"]
  df["Article_sentiment_momentum_minus_2_day"] = df["Article_sentiment_2day_lag"] -
df["Article_sentiment_3day_lag"]
  df["Article_sentiment_momentum_minus_3_day"] = df["Article_sentiment_3day_lag"] -
df["Article_sentiment_4day_lag"]
  return (df)
}

generate_sentiment_momentum_pos_neg = function(df){
  sentiment_momentum = df["Article_sentiment_momentum"]
  sentiment_momentum_square = sentiment_momentum^2
  df["Article_sentiment_momentum_pos"] = sentiment_momentum_square  * (as.numeric(sentiment_momentum>0))
  df["Article_sentiment_momentum_neg"] = sentiment_momentum_square  * (as.numeric(sentiment_momentum<0))

  sentiment_momentum = df["Article_sentiment_momentum_minus_1_day"]
  sentiment_momentum_square = sentiment_momentum^2
  df["Article_sentiment_momentum_minus_1_day_pos"] = sentiment_momentum_square  *
(as.numeric(sentiment_momentum>0))
  df["Article_sentiment_momentum_minus_1_day_neg"] = sentiment_momentum_square  *
(as.numeric(sentiment_momentum<0))

  sentiment_momentum = df["Article_sentiment_momentum_minus_2_day"]
  sentiment_momentum_square = sentiment_momentum^2
  df["Article_sentiment_momentum_minus_2_day_pos"] = sentiment_momentum_square  *
(as.numeric(sentiment_momentum>0))
  df["Article_sentiment_momentum_minus_2_day_neg"] = sentiment_momentum_square  *
(as.numeric(sentiment_momentum<0))

  sentiment_momentum = df["Article_sentiment_momentum_minus_3_day"]
  sentiment_momentum_square = sentiment_momentum^3
  df["Article_sentiment_momentum_minus_3_day_pos"] = sentiment_momentum_square  *
(as.numeric(sentiment_momentum>0))
  df["Article_sentiment_momentum_minus_3_day_neg"] = sentiment_momentum_square  *
(as.numeric(sentiment_momentum<0))
  return (df)
}

generate_sentiment_pos_neg = function(df){
  sentiment = df["Article_sentiment"]
  sentiment_square = sentiment^2
  df["Article_sentiment_pos"] = sentiment_square  * (as.numeric(sentiment>0))
  df["Article_sentiment_neg"] = sentiment_square  * (as.numeric(sentiment<0))

  sentiment = df["Article_sentiment_1day_lag"]
  sentiment_square = sentiment^2
  df["Article_sentiment_minus_1_day_pos"] = sentiment_square  * (as.numeric(sentiment>0))
  df["Article_sentiment_minus_1_day_neg"] = sentiment_square  * (as.numeric(sentiment<0))

  sentiment = df["Article_sentiment_2day_lag"]
  sentiment_square = sentiment^2
  df["Article_sentiment_minus_2_day_pos"] = sentiment_square  * (as.numeric(sentiment>0))
  df["Article_sentiment_minus_2_day_neg"] = sentiment_square  * (as.numeric(sentiment<0))

  sentiment = df["Article_sentiment_3day_lag"]
  sentiment_square = sentiment^3
  df["Article_sentiment_minus_3_day_pos"] = sentiment_square  * (as.numeric(sentiment>0))
  df["Article_sentiment_minus_3_day_neg"] = sentiment_square  * (as.numeric(sentiment<0))
  return (df)
}
```

```
generate_fundamental = function(df, fundamental_df){
 #Convert the string date to a date format and store in a new column
 fundamental_df$DateAsDate = as.Date(as.character(fundamental_df$Date), "%m/%d/%y")
 #Convert EPS string to numeric and store in a new column
 fundamental_df$Eps = as.numeric(substring(fundamental_df$EPS, 2))
 default_value = NA
 df_rows = dim(df)[1]
 #Add fundamental col headers df
 df["Eps"] = default_value
 #Match the fundamental date to the Market data date and upgrade the market data data frame (df)
 for (i in 1:df_rows){
  fundamental_index = which(fundamental_df$DateAsDate == df$DateAsDate[i])
  if (length(fundamental_index) >0){
   df[i,]$Eps = fundamental_df$Eps[fundamental_index]
  }
 }
 #Patch Eps
 df["Eps_patch"] = default_value
 df["Price_earning_ratio"] = default_value
 current_eps = default_value
 for (i in 1:df_rows){
  df[i,]$Eps_patch = current_eps
  if (is.na(df[i,]$Eps) == FALSE){
   current_eps = df[i,]$Eps
  }
  df[i,]$Eps_patch = current_eps
  df[i,]$Price_earning_ratio = df[i,]$Close/current_eps
 }
 return(df)
}
```

Code - ../ data.generator.functions / explanatory.variables.r

```
#author: Frederic Marechal
#date: April-Sep, 2017

#Clear console
cat("\014")
#This function the log_returns. It indicates whether the market when up, down or sideways (neutral) from one day to the
next.
#Please ensure the dataset is ordered in data descending order, as the calculations assume the ordering for the calculation
generate_log_returns_and_direction = function (stock_df,epsilon){
 stock_df["Direction"] = NA
 stock_df["Log_returns"] = NA
 for (k in 1:nrow(stock_df)) {
  #Generate the log return value
  stock_df[k,"Log_returns"] =  log10(stock_df$Close[k+1] / stock_df$Close[k])
   #Indicate whether it is going up or down
  if (k < nrow(stock_df)) {
   if (!is.na( stock_df[k,"Log_returns"])){
    #This is the default values of the 'Direction' and 'Direction_Flag'
    stock_df[k,"Direction"] = 'NA'
    stock_df[k,"Direction_Flag"] = -1
    #Do the Epsilon check
    if (stock_df[k,"Log_returns"] > epsilon){
     stock_df[k,"Direction"] = 'Up'
     stock_df[k,"Direction_Flag"] = 1
    }
    else{
     stock_df[k,"Direction"] = 'Down'
     stock_df[k,"Direction_Flag"] = 0
    }
   }
  }
 }
 return(stock_df)
}
```

Code - ../ data.generator.functions / response.variables.r

Appendix 7 – The Time Window Generator Logic

This code corresponds to the time window generation used by the machine learning models. The code can be found in the following places:

Code Location	Description
../ time.windows.generator.batch.svc.rmd	This is the entry point that delegates into the time.windows.generator.r for each asset.
../ time.windows.generator.r	It is responsible for generating the time window training/validation/test slices for each time slice for a given asset

````
---
#Markdown doc generation: http://rmarkdown.rstudio.com/word_document_format.html
#Markdown cheat sheet:
#http://nestacms.com/docs/creating-content/markdown-cheat-sheet
#http://rmarkdown.rstudio.com/authoring_basics.html
title: "Asset DataSet Geneator Service"
author: Frederic Marechal
date: April-Sep, 2017
output: word_document
---
################################################################################
################################################################################
#######              Clean-Up
################################################################################
################################################################################
```{r cleanup}
#Clear console
cat("\014")
```



################################################################################
################################################################################
#######              Directory Config
################################################################################
################################################################################
```{r working_directory_config}
is_server_run = FALSE
if(is_server_run){
  setwd('/host/dsm1/fmare001/stats/svm/deliverables')
}else{
  setwd('C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/Thesis/Finance/z_Code')
}
index_name = 'XLE'
input_dir = paste(getwd(), "/data/", index_name, "/processed/final" ,sep="")
output_dir = paste(getwd(), "/data/", index_name, "/processed/time.windows",sep="")
output_dir_error = paste(getwd(), "/data/", index_name, "/processed/error/time.windows",sep="")
```
################################################################################
################################################################################
#######              Load Dependencies
################################################################################
################################################################################
```{r load_depedencies}
source("utilities.r")
source("time.windows.generator.r")
library(foreach) #to allow parallel processing
# need to install doMC via install.packages("doMC", repos="http://R-Forge.R-project.org")
library(doMC) #install and register a parallel backend, see https://www.r-bloggers.com/parallel-r-loops-for-windows-and-linux/
````
````

```
registerDoMC(6) # number of parallel processes
```
```
################################################################################
################################################################################
#######               Directory Clean-up
################################################################################
################################################################################
```{r load_directory_cleanup}
delete_all_files_in_directory(paste(output_dir,"/100",sep=""))
delete_all_files_in_directory(paste(output_dir,"/180",sep=""))
delete_all_files_in_directory(paste(output_dir,"/200",sep=""))
delete_all_files_in_directory(paste(output_dir,"/300",sep=""))
delete_all_files_in_directory(paste(output_dir,"/400",sep=""))
delete_all_files_in_directory(paste(output_dir,"/500",sep=""))
delete_all_files_in_directory(paste(output_dir,"/1000",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/100",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/180",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/200",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/300",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/400",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/500",sep=""))
delete_all_files_in_directory(paste(output_dir_error,"/1000",sep=""))
```
```
################################################################################
################################################################################
#######               DataSet Generator As Batch
################################################################################
################################################################################
```{r dataset_generator}
validation_set_bucket =  66 #number of validation days
training_set_bucket = 220  #number of training days
test_set_bucket = 5 #number of test days
number_sliding_windows = 100 #number of time sliding windows
files = list.files(input_dir)
#%do%: non parallel
#%dopar%: parallel version
#Run the stocks...
foreach(i=1:length(files)) %dopar% {
  file = files[i]
  df = load_from_csv(paste(input_dir, "/",sep=""), file,TRUE)
  time_window_generator(df,file, number_sliding_windows, training_set_bucket,validation_set_bucket, test_set_bucket,
output_dir, output_dir_error )
}
#Run the index...
df = load_from_csv(paste(input_dir, "/",sep=""), index_name,FALSE)
time_window_generator(df,index_name, number_sliding_windows, training_set_bucket,validation_set_bucket,
test_set_bucket, output_dir, output_dir_error )
```
```

Code - ../ time.windows.generator.batch.svc

```
#author: Frederic Marechal
#date: April-Sep, 2017
#Add a new row in the time window dataframe
add_row_to_time_windows = function(time_window_df,
                                    test_end_index, test_start_index,
                                    validation_end_index,validation_start_index,
                                    training_end_index, training_start_index){
  #Create a new row
  newrow = c(test_end_index, test_start_index,
              validation_end_index, validation_start_index,
              training_end_index, training_start_index)
  #Add the new row to the dataset
  time_window_df[nrow(time_window_df)+1,] <- newrow
  return (time_window_df)
}
#Generate the time window list
```
```

```
time_window_generator = function(df, file, number_sliding_windows, training_set_bucket,validation_set_bucket,
test_set_bucket, output_dir, output_dir_error){
  #create the time window schema
  time_window_df = data.frame(  test_end_index = 0,
                                test_start_index = 0,
                                validation_end_index = 0,
                                validation_start_index = 0,
                                training_end_index = 0,
                                training_start_index = 0,
                                stringsAsFactors=FALSE)
  #Step 1 - build a list of traning/validation/test sets start/end index
  #We start from the nearest date and we walk backards...
  slid_win_idx = 0
  test_end_index = dim(df)[1]
  while (slid_win_idx < number_sliding_windows) {
    test_start_index = test_end_index - test_set_bucket + 1
    validation_end_index = test_start_index - 1
    validation_start_index = validation_end_index - validation_set_bucket + 1
    training_end_index = validation_start_index - 1
    training_start_index = training_end_index - training_set_bucket + 1
    #cannot continue as the training index would be 0 or negative
    if (training_start_index <1) {break}
    #add the time window to the list
    time_window_df = add_row_to_time_windows(time_window_df,
                                      test_end_index, test_start_index,
                                      validation_end_index, validation_start_index,
                                      training_end_index, training_start_index)
    #now reset the 'test_end_index'
    if (test_set_bucket > 1){
      test_end_index = test_end_index - test_set_bucket +1
    }else{
      test_end_index = test_end_index - test_set_bucket
    }

    if (slid_win_idx > test_end_index ) {
      print("ERROR - The sliding window cannot be greater than the end test index...")
      break;
    }
    slid_win_idx = slid_win_idx + 1
  }
  #Step 2 - Remove first row as this is necessary (all columns are set to 0)
  time_window_df = time_window_df[-1,]
  #Step 3 - Now reverse the time series in the list to ensure we keep the the time windows in the time order.
  time_window_df = time_window_df[rev(rownames(time_window_df)),]
  #Save error
  output_error_full_path = paste(output_dir_error, "/", number_sliding_windows, "/",sep = "")
  if (slid_win_idx< number_sliding_windows){
    cat (paste("\nSave error time window :", file, " to: ", output_error_full_path, "\n", sep=""))
    error_df=data.frame(actual_sliding_windows = slid_win_idx,stringsAsFactors=FALSE)
    save_to_csv(error_df, output_error_full_path, file)
  }
  #Save output
  output_full_path = paste(output_dir, "/", number_sliding_windows, "/",sep = "")
  cat (paste("\nSave time window :", file, " to: ", output_full_path, "\n", sep=""))
  save_to_csv(time_window_df, output_full_path, file)
  #cat (paste("\nSave time window ", file, " count: ", slid_win_idx, "\n", sep=""))
  #save_to_csv(data.frame(slid_win_idx),  paste(output_full_path, ".counts/",number_sliding_windows, "/",sep = "") ,
file)
 }
```

Code - ../ time.windows.generator.r

Appendix 8 –The Feature Selection Service

This code corresponds to the time window generation used by the machine learning models. The code can be found in the following places:

Code Location	Description
../feature.selection.generator.batch.svc.rmd	First it removes the near zero variance, linear dependent attributes as well as the correlated predictors. Then it deleguates into feature.selection.generator.r. to be run in two modes: i) permutated relief or random forest mean decrease gini
../ feature.selection.generator.r	It is responsible for producing the permutated relief feature selection, random forest mean decrease gini results and graph

```
---
#Markdown doc generation: http://rmarkdown.rstudio.com/word_document_format.html
#Markdown cheat sheet:
#http://nestacms.com/docs/creating-content/markdown-cheat-sheet
#http://rmarkdown.rstudio.com/authoring_basics.html
title: "Asset DataSet Geneator Service"
author: Frederic Marechal
date: April-Sep, 2017
output: word_document
---
################################################################################
################################################################################
#######              Clean-Up
################################################################################
################################################################################
```{r cleanup}
#Clear console
cat("\014")
```
##
##
#######   Directory Config
##
##
```{r working_directory_config}
is_server_run = FALSE
if(is_server_run){
  setwd('/host/dsm1/fmare001/stats/svm/deliverables')
}else{
  setwd('C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/Thesis/Finance/z_Code')
}
index_name = 'XLE'
feature_selection = 'permuted_relief'
#feature_selection = 'random_forest_mean_decrease_gini'
number_sliding_windows = 100
#number_sliding_windows = 200
n_trees = 500
ignore_sentiment_attributes = TRUE #TRUE: ignore sentiment attributes / FALSE: take them into consideration
ignore_fundamental_attributes = TRUE #TRUE: ignore fundamental attributes / FALSE: take them into consideration

time_windows_input_dir = paste(getwd(), "/data/", index_name,
"/processed/time.windows/",number_sliding_windows,sep="")
data_input_dir = paste(getwd(), "/data/", index_name, "/processed/final/", sep="")
output_dir = paste(getwd(), "/data/", index_name, "/processed/feature.selection/",sep="")
graphs_output_dir = paste(getwd(), "/graphs/", index_name,"/feature.selection/",sep="")

if (feature_selection == 'permuted_relief'){
  output_dir = paste(output_dir,feature_selection,"/",number_sliding_windows,"/",sep="")
  graphs_output_dir = paste(graphs_output_dir,feature_selection,"/",number_sliding_windows,"/",sep="")
} else if (feature_selection == 'random_forest_mean_decrease_gini'){
  output_dir = paste(output_dir,"/mean.decrease.gini/",number_sliding_windows,"/",sep="")
  graphs_output_dir = paste(graphs_output_dir,"/mean.decrease.gini/",number_sliding_windows,"/",sep="")
```
```

```
} else {
   print (paste("ERROR ----> ", feature_selection, " not supported", sep=""))
}
```
```
################################################################################
################################################################################
#######              Load Dependencies
################################################################################
################################################################################
```{r load_depedencies}
require(randomForest)
require(CORElearn)
require(AppliedPredictiveModeling)
require(caret)
source("utilities.r")
source("feature.selection.generator.r")
library(foreach) #to allow parallel processing
# need to install doMC via install.packages("doMC", repos="http://R-Forge.R-project.org")
library(doMC) #install and register a parallel backend, see https://www.r-bloggers.com/parallel-r-loops-for-windows-and-linux/
registerDoMC(6) # number of parallel processes
```
##
##
#######   Directory Clean-up
##
##
```{r load_directory_cleanup}
delete_all_files_in_directory(output_dir)
delete_all_files_in_directory(graphs_output_dir)
```

##
##
#######   Pre processing Functions
##
##
```{r pre_processing}
#The function removes the zero variance attributes
remove_near_zero_variance_columns = function(data) {
  print ("Remove_near_zero_variance_columns --> Enter")
  colnames = names(data)
  nzv = nearZeroVar(data)
  nzv = colnames(data[nzv])
  print (paste("Zero variance cols --> ", nzv, sep=""))
  res = data
  if (length(nzv) > 0){
    res = data[, colnames[!colnames %in% nzv]]}
  print ("Remove_near_zero_variance_columns --> Exit")
  return(res)
}
#The function findLinearCombos uses the QR decomposition of a matrix
#to enumerate sets of linear combinations (if they exist).
#For each linear combination, it will incrementally remove columns
#from the matrix and test to see if the dependencies have been resolved.
remove_linear_dependencies = function(data) {
  print ("Remove_linear_dependencies --> Enter")
  comboInfo = findLinearCombos(data.matrix(data))
  removeCount = 0
  # The dataset minus the identified col(s)
  if (is.null(comboInfo$remove) == FALSE) {
    res = data[, -comboInfo$remove]
    removeCount =length(comboInfo$remove)
  }
  print(paste("Number of removed linearly dependent col(s): ", removeCount, sep=""))
  print ("Remove_linear_dependencies --> Exit")
  return (data)
```
```

```
}
# Identify and remove Correlated Predictors -> The collinearity issue
# Only to be applied to numerica columns
remove_correlated_predictors = function(data, correlationCutOff= 0.95){
  print ("Remove_correlated_predictors --> Enter")
  # Get the correlation
  corr = cor(data)
  # This function searches through a correlation matrix and returns a vector of integers corresponding
  # to columns to remove to reduce pair-wise correlations.
  cols_index_to_delete = findCorrelation(corr, cutoff = correlationCutOff)
  removeCount = 0
  count_col_to_delete = 0
  if (length(cols_index_to_delete) > 0){
    res = data[, -cols_index_to_delete]
    removeCount = length(cols_index_to_delete)
    print(paste("Number of removed correlated predictors: ", removeCount, sep=""))
    print ("Remove_correlated_predictors --> Exit")
    return(res)
  }
  print ("Remove_correlated_predictors --> Exit")
  return (data)
}
```
```
################################################################################
################################################################################
#######              Feature Selection Generator (Batch)
################################################################################
################################################################################
```{r dataset_generator}
time_windows_files = list.files(time_windows_input_dir)
#%do%: non parallel
#%dopar%: parallel version
foreach(i=1:length(time_windows_files)) %dopar% {
  file_name = time_windows_files[i]
  cat (paste("\nTry feature selection for: ", file_name,  "\n", sep=""))
  time_windows_df = load_from_csv(paste(time_windows_input_dir, "/",sep=""), file_name,TRUE)
  data_df = load_from_csv(paste(data_input_dir, "/",sep=""), file_name,TRUE)

  #If TRUE, means ignore any sentiment attributes
  if (ignore_sentiment_attributes==TRUE){
    data_df = data_df[!grepl("Article_sentiment",names(data_df))]
    #data_df = subset( data_df, select = -Article_sentiment ) #data_df[-c("Article_sentiment"),]
  }

  #If TRUE, means ignore any fundamental attributes
  if (ignore_fundamental_attributes==TRUE){
    data_df = data_df[!grepl("Price_earning_ratio",names(data_df))]
  }
  #reduce the datarange
  range = 1:time_windows_df[1,"training_end_index"] #case for inception to end of first trainingset
  data_df = data_df[range,]

  #Step 1- Pre-process data
  log_returns_col = get_col_index(data_df, "Log_returns")
  Log_returns = data_df$Log_returns
  direction_col = get_col_index(data_df, "Direction")
  Direction = data_df$Direction
  direction_flag_col = get_col_index(data_df, "Direction_Flag")
  Direction_Flag = data_df$Direction_Flag
  #remove near variance cols
  pre_process_data_df = data_df[,-c(log_returns_col, direction_col,direction_flag_col)]
  pre_process_data_df = remove_near_zero_variance_columns (pre_process_data_df)
  #remove multicollinear columns
  pre_process_data_df = remove_linear_dependencies(pre_process_data_df)
  #remove correlated columns
  pre_process_data_df = remove_correlated_predictors(pre_process_data_df)
   #re-attached the respones variables
```
```

```
  data_df = cbind(pre_process_data_df, Log_returns)
  data_df = cbind(data_df, Direction)
  data_df = cbind(data_df, Direction_Flag)
  #Step 1- Generate the res
  print (feature_selection)
  if (feature_selection == 'permuted_relief'){
      res = permuted_relief(file_name,data_df,the_estimator="ReliefFequalK", the_nperm = 500, the_reliefIterations =50,
output_dir, graphs_output_dir)
  } else if (feature_selection == 'random_forest_mean_decrease_gini'){
      res = random_forest_mean_decrease_gini(file_name, data_df,n_trees,output_dir, graphs_output_dir)
    #Error during wrapup: task 1 failed - "only 0's may be mixed with negative subscripts" => do not understand where
this comes from but seems ok
  } else {
      print (paste("ERROR ----> ", feature_selection, " not supported", sep=""))
  }
}
```

Code - ../ feature.selection.generator.batch.svc

```
#author: Frederic Marechal
#date: April-Sep, 2017
permuted_relief = function(file_name,data_df,the_estimator="ReliefFequalK", the_nperm = 500, the_reliefIterations
=50, output_dir, graphs_output_dir){
  cat (paste("\nEnter permuted relief: ", file_name, " to: ", output_dir, "\n", sep=""))
  set.seed(1)
  perm <- permuteRelief(x = data_df[,c(-get_col_index(data_df,"Log_returns"),-get_col_index(data_df,"Direction"),-
get_col_index(data_df,"Direction_Flag"))],
                        y = data_df$Direction,
                        nperm = the_nperm,
                        estimator = the_estimator,
                        ReliefIterations = the_reliefIterations)

  perm_standardized_df = data.frame(sort(perm$standardized,decreasing = TRUE))
  perm_standardized_df = rename_col(perm_standardized_df,
'sort.perm.standardized..decreasing...TRUE.','perm.standardized')
  #Save
  cat (paste("\nSave permuted relief: ", file_name, " to: ", output_dir, "\n", sep=""))
  save_to_csv(data.frame(perm_standardized_df), output_dir, file_name, TRUE)
  #Generate graph
  permuted_relief_graph(file_name, perm, graphs_output_dir)
  return (perm)
}
permuted_relief_graph = function(file_name,perm, output_dir){
  png(filename=paste(output_dir,file_name,".png",sep=""))
  #save plot
  histogram(~ value|Predictor,data = perm$permutations)
  cat (paste("\nSave histogram permuted relief :", file_name, " to: ", output_dir, "\n", sep=""))
  dev.off()
}
random_forest_mean_decrease_gini = function(file_name,data_df,n_trees, output_dir, graph_output_dir){
  cat (paste("\nEnter mean decrease gini: ", file_name, " to: ", output_dir, "\n", sep=""))
  set.seed(1)
  #Feature select against all variables bar (Direction_Flag) as it is 100% correlated to the Direction explanatory variable
  fit = randomForest( factor(Direction)~.-Direction_Flag-Log_returns,  #encode the vector as a factor
                      data=data_df,
                      ntree = n_trees,
                      #mtry=6, defaulted to SQRT(p) for a classification tree
                      importance=TRUE)
  #Generate the importance data and order by the column "MeanDecreaseGini" (desc)
  importance_res = importance(fit)
  importance_res = importance_res[order(importance_res,"MeanDecreaseGini"] , decreasing = TRUE ),]
  #Save
  cat (paste("\nSave mean decrease gini: ", file_name, " to: ", output_dir, "\n", sep=""))
  save_to_csv(data.frame(importance_res), output_dir, file_name, TRUE)
  #Generate graph
  mean_decrease_gini_graph(file_name, fit, graph_output_dir)
```

```
  return (importance_res)
 }
 mean_decrease_gini_graph = function(file_name,fit, output_dir){
  png(filename=paste(output_dir,file_name,".png",sep=""))
  cat (paste("\nSave plot mean decrease gini :", file_name, " to: ", output_dir, "\n", sep=""))
  #save plot
  varImpPlot(fit,type=2, main = paste("Plot MeanDecreaseGini: ", file_name, "(# trees= ",n_trees ,")" ))
  dev.off()
 }
 formulas_generator = function(input_dir, file_name, underlying_name, response_variable_name, cut_off){
  feature_selection_df = load_from_csv(paste(input_dir, "/",sep=""), file_name,TRUE)
  #Get the list of formulas to test
  formulas = generate_formulas(response_variable_name, feature_selection_df, cut_off)[c(-1,-2,-3)]
  #Save formula dictionary
  file_name = paste("/",underlying_name,"_formula_dict.csv",sep="")
  cat(paste("Save ", file_name, sep=""))
  save_to_csv(formulas, input_dir,file_name)
  return (formulas)
 }
```

Code - ../ feature.selection.generator.r

Appendix 8bis – The Batch Run Service

This is the central service for generating performance measures from a model. First, the user needs to configure the service with the following information: i) define an index (e.g. *index_name = 'XLE'*), ii) set the number of time windows (e.g. *number_sliding_windows =100*), iii) choose the model name (e.g. *model_short_name = "elman"*), iv) select a scenario (e.g. *scenario = "original"*) and finally vi) choose *Feature Selection* algorithm so that the relevant feature selection set(s) can be loaded (e.g. formula_type = "RandomForest"). In a case of a random forest 15 *Feature Selection* sets are generated. The *Feature Selection* set generated the best accuracy result is retained. For the permutated *Relief* only one set is available. The final *Feature Selection* is based on confidence interval.

The below code example present the content of the file '..\pc.run.original run.models.batch.svc_batch1.elman.r'. All the cases are listed in the table below. The code is the same across all files. The only difference lies in the parameters sets.

Machine Learning Models Service	Run all 'rmd' files present in the following folder to generate the base results for all assets/all models based on a 'permutated relief' feature selection model: ..\pc.run.original Run all 'rmd' files present in the following folder to generate the base results for all assets/all models based on a 'gini index' feature selection model: ..\server.run.original Run all 'rmd' files present in the following folders to generate the sentiment and sentiment mometum results for all assets/all models based on either a 'gini index' or 'relief' feature selection model (depending on set-up): ..\pc.run.sentiment ..\pc.run.sentiment.m1 ..\pc.run.sentiment.m2 ..\pc.run.sentiment.m3 ..\pc.run.sentiment.momentum.rmd ..\pc.run.sentiment.momentum.mm1 ..\pc.run.sentiment.momentum.mm1m2 ..\pc.run.sentiment.momentum.mm1m2m3

```
is_server_run = FALSE
if(is_server_run){
  setwd('/host/dsm1/fmare001/Thesis/Finance/z_Code')
}else{
  setwd('C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/Thesis/Finance/z_Code')
}

index_name = 'XLE'
number_sliding_windows = 100

#model_name = "Linear Descriminent Analysis (LDA)"
#model_short_name = "lda"

#model_name = "Quadratic Descriminent Analysis (QDA)"
#model_short_name = "qda"

#model_name = "Penalised Descriminent Analysis (QDA)"
#model_short_name = "pda"

#model_name = "Boosting Model"
#model_short_name = "boosting"

#model_name = "Suppport Vector Machine (Linear)"
#model_short_name = "svmLinear"

#model_name = "Suppport Vector Machine (Poly)"
#model_short_name = "svmPoly"

#model_name = "Random Forest (RF ntree = 500)"
#model_short_name = "rf_500"

#model_name = "Random Forest (RF ntree = 1000)"
#model_short_name = "rf_1000"

#model_name = "Random Forest (RF ntree = 1500)"
#model_short_name = "rf_1500"

#model_name = "Mlp Weigth Decay"
#model_short_name = "mlp_weight_decay"

model_name = "Elman Recurrent Neural Net"
model_short_name = "elman"

#model_name = "Jordan Recurrent Neural Net"
#model_short_name = "jordan"

scenario = "original"
#scenario = "sentiment"
#scenario = "sentiment_momentum"
if (scenario == "sentiment"){
  formula_append="Article_sentiment"
  #formula_append="Article_sentiment_2day_lag+Article_sentiment_3day_lag+Article_sentiment_4day_lag"
} else if (scenario == "sentiment_momentum"){
  formula_append="Article_sentiment_momentum"
}

formula_type = "PermutedRelief"
#formula_type = "RandomForest"
if (formula_type == "RandomForest"){
  feature_selection_sub_dir = 'mean.decrease.gini/'
}else if (formula_type == "PermutedRelief"){
  feature_selection_sub_dir = 'permuted_relief/'
}else{
  print(paste("run.models.batch.svc --> formula_type not supported...", formula_type, Sep=""))
}
```

```
time_windows_input_dir = paste(getwd(), "/data/", index_name,
"/processed/time.windows/",number_sliding_windows,sep="")
feature_selection_input_dir = paste(getwd(), "/data/", index_name,
"/processed/feature.selection/",feature_selection_sub_dir,number_sliding_windows,"/",sep="")
data_input_dir = paste(getwd(), "/data/", index_name, "/processed/final/", sep="")
confusion_matrix_output_dir = paste(getwd(), "/data/", index_name,"/processed/models/",sep="")
confusion_matrix_output_err_dir = paste(getwd(), "/data/", index_name, "/processed/models/",
model_short_name,"/",number_sliding_windows,"/confusion.matrix.error/",sep="")

#This is to prevent lib conflicts
if (model_short_name == "elman" || model_short_name == "jordan"){
  require(zoo)
  require(RSNNS)
}
require(caret)
require(psych)
require(epiR)
require(randomForest)
require(MASS)
require(gbm)
require (e1071)
require(nnet)
source("utilities.r")
source("statisitical.tests.r")
source("feature.selection.generator.r")
source("models/model_utils.r")
source("models/model_executor.r")
source("models/lda.r")
source("models/qda.r")
source("models/rf.r")
source("models/mlp_weight_decay.r")
source("models/elman_jordan.r")
source("models/pda.r")
source("models/svm.r")
#source("models/boosting.r")
#source("models/lm.r")
library(foreach) #to allow parallel processing
# need to install doMC via install.packages("doMC", repos="http://R-Forge.R-project.org")
library(doMC) #install and register a parallel backend, see https://www.r-bloggers.com/parallel-r-loops-for-windows-
and-linux/
registerDoMC(1) # number of parallel processes


time_windows_files = list.files(time_windows_input_dir)
feature_selection_files = list.files(feature_selection_input_dir)

#%do%: non parallel
#%dopar%: parallel version
seq_feat_select_files = seq(length(feature_selection_files))
for (k in seq_feat_select_files)  {
    feature_selection_file_name = feature_selection_files[k]
    #if (k>9 && k<20){
        if (feature_selection_file_name == "XLE.csv"){
      #Get the list of impacting variables
      underlying_name = remove_file_extension(feature_selection_file_name)
      #Generate Formula dico
      print(paste("Formula Type:", formula_type))
      if (formula_type == "RandomForest"){
        print("Auto Generate Formula...")
        formulas = formulas_generator(feature_selection_input_dir, feature_selection_file_name, underlying_name,
"Direction", 15)
        if (model_short_name == "elman" || model_short_name == "jordan"){
          formulas = formulas[1:(length(formulas)-1)]
        }
      }else if (formula_type == "PermutedRelief"){
        if (!(model_short_name == "elman" && model_short_name == "jordan")){
          #1.96 => 95% (2.5% on each side)
```

```
        formulas = generate_formulas_permuted_relief(feature_selection_input_dir, underlying_name, "Direction", 1.96)
        #no attributes have been found for the previous confident interval
        #try 1.645 => 90% (5% on each side)
        if (grepl("~", formulas$form) == FALSE ){
           formulas = generate_formulas_permuted_relief(feature_selection_input_dir, underlying_name, "Direction",
                       1.645)
        }
        #no attributes have been found for the previous confident interval
        #try 1.28 => 80% (10% on each side)
        if (grepl("~", formulas$form) == FALSE){
           formulas = generate_formulas_permuted_relief(feature_selection_input_dir, underlying_name, "Direction",
                       1.28)
        }
        else {
          print("special case...")
          #1.96 => 95% (2.5% on each side)
          formulas = generate_formulas_permuted_relief(feature_selection_input_dir, underlying_name, "Direction",
                      1.96)
          #no attributes have been found for the previous confident interval
          #try 1.645 => 90% (5% on each side)
          if (grepl("~", formulas$form) == FALSE || length(all.vars(as.formula(formulas$form))) < 3 ){
             formulas = generate_formulas_permuted_relief(feature_selection_input_dir, underlying_name, "Direction",
                         1.645)
          }
          #no attributes have been found for the previous confident interval
          #try 1.28 => 80% (10% on each side)
          if (grepl("~", formulas$form) == FALSE || length(all.vars(as.formula(formulas$form))) < 3){
             formulas = generate_formulas_permuted_relief(feature_selection_input_dir, underlying_name, "Direction",
                         1.28)
          }
        }
       }
     }
     #Get the data
     data_df = load_from_csv(paste(data_input_dir, "/",sep=""), feature_selection_file_name,TRUE)
     #Generate the sum of confusion matrices for each fornula
     seq_formulas = rev(seq(length(formulas)))
     curr_avg_accuracy = 0.0
     #For each formula
     for (p in seq_formulas){
        if (formula_type == "RandomForest"){
          formula_as_srtring = formulas[[p]][[2]]
          formula_idx = formulas[[p]][[1]]
        }else if (formula_type == "PermutedRelief"){
           formula_as_srtring = formulas$form[p]
           formula_idx = formulas$idx[p]
        }
        #formula_as_srtring = paste(formula_as_srtring,'+Price_earning_ratio',sep="")

        #save formula info for an underlying and a formula id
        formula_df =
data.frame(underlying_name=underlying_name,formula_type=formula_type,formula_idx=as.numeric(formulas[[p]][[1]
]),
                          formula_as_srtring=formula_as_srtring, stringsAsFactors=FALSE)
        formula_file_name_root = paste(underlying_name, "_", formula_idx,sep="")
        formula_file_name = paste("Formula_",formula_file_name_root,".csv",sep="")
        cat (paste("\nSave model formula for: ", formula_file_name, "\n", sep=""))
        save_to_csv(formula_df, confusion_matrix_output_dir,formula_file_name)

        time_windows_df = load_from_csv(paste(time_windows_input_dir, "/",sep=""),
                                  feature_selection_file_name,TRUE)
        if (is.na(formula_as_srtring)==FALSE){
          #formula_as_srtring = paste (formula_as_srtring, "+", "Price_earning_ratio", sep="")
          if (scenario != "original"){
            formula_as_srtring = paste (formula_as_srtring, "+", formula_append, sep="")
          }else{
            #ensure no sentiment cols => when sentiment is not required => small optmisation
```

```
 if (underlying_name != "XLE"){
   data_df = data_df[ , -grep("Article_sentiment", names(data_df))]
  }
}

formula = as.formula(formula_as_srtring)
time_windows_seq = seq(dim(time_windows_df)[1])
for (tw_index in time_windows_seq){
 #if ( tw_index <21) {
   #To improve on this.....
   data_df = load_from_csv(paste(data_input_dir, "/",sep=""), feature_selection_file_name,TRUE)
   if (scenario != "original"){
    formula_as_srtring = paste (formula_as_srtring, "+", formula_append, sep="")
   }else{
    #ensure no sentiment cols => when sentiment is not required => small optmisation
    if (underlying_name != "XLE"){
     data_df = data_df[ , -grep("Article_sentiment", names(data_df))]}
    }

   #Step 1- Training/Optimising the model
   #Get the Training and Validation data for the given time window
   training_range =
   time_windows_df[tw_index,"training_start_index"]:time_windows_df[tw_index,"training_end_index"]
   training_data = data_df[training_range,]
   validation_range =
   time_windows_df[tw_index,"validation_start_index"]:time_windows_df[tw_index,"validation_end_index"]
   validation_data = data_df[validation_range,]
   #impute missing data
   training_data = pre_process(df=training_data,method = "medianImpute")
   validation_data = pre_process(df=validation_data,method = "medianImpute")
   #store the confusion matrix file

   optimised_parameters = run_model_factory( model_name, model_short_name,
                                "train",scenario,underlying_name,
                                formula_idx, formula,
                                NULL,
                                number_sliding_windows,tw_index,
                                training_data,validation_data, confusion_matrix_output_dir)
   #Step 2- Test the model
   training_range =
  time_windows_df[tw_index,"training_start_index"]:time_windows_df[tw_index,"validation_end_index"]
   training_data = data_df[training_range,]
   test_range = time_windows_df[tw_index,"test_start_index"]:time_windows_df[tw_index,"test_end_index"]
   test_data = data_df[test_range,]
   #impute missing data
   training_data = pre_process(df=training_data,method = "medianImpute")
   test_data = pre_process(df=test_data,method = "medianImpute")

   #store the confusion matrix file
   optimised_parameters = run_model_factory( model_name, model_short_name,
                                "test",scenario,underlying_name,
                                formula_idx, formula,
                                optimised_parameters,
                                number_sliding_windows,tw_index,
                                training_data,test_data, confusion_matrix_output_dir)

   #}
}

source("models/model_utils.r")
#Generate the overall training confusion matrix and performance measures for a given underlying and formula

generate_model_performance_measures(model_short_name,"train",scenario,underlying_name,number_sliding_window
s, formula_idx, confusion_matrix_output_dir)
```

```
        #Generate the overall test confusion matrix and performance measures for a given underlying and formula
        accuracy_rate_avg_unrounded =
generate_model_performance_measures(model_short_name,"test",scenario,underlying_name,number_sliding_windows,
formula_idx, confusion_matrix_output_dir)

        #Quite when the accuracy is going down
        if ( accuracy_rate_avg_unrounded <= curr_avg_accuracy){
          print("Best formula has been found...")
          break;
        }
        curr_avg_accuracy = accuracy_rate_avg_unrounded
      }
    }
  }
}
```
Code - ..\pc.run.originalrun.models.batch.svc_batch1.elman.r

Appendix 9 – The Machine Learning Models

These following boxes list the Machine Learning models' implementation.

QDA Model

```
#author: Frederic Marechal
#date: April-Sep, 2017
#the test_data_param could be a traning/validation or test dataset
run_model_qda = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")
  cat (paste("\nRunning: ", model_name, " formula idx: ",formula_idx, " formula: ", formula_as_srtring ," for time
window: ", tw_index ,"\n", sep=""))
   #caret implementation
  set.seed(1)
  preprocess_training_data = preProcess(training_data, method=c("BoxCox"))
  preprocess_training_data =predict( preprocess_training_data , training_data )
   set.seed(1)
  qda.fit = NULL
  qda.fit = tryCatch( train ( formula,
                        preprocess_training_data,
                        method = "qda",
                        metric="kappa"),
                  error=function(e) next)
   if (is.null(qda.fit) == FALSE){
   #Generate the prediction
   qda.pred = predict(qda.fit, test_data)
   #Generate the confusion matrix
   qda.confusion_table =  table(qda.pred, test_data$Direction)
   #Save confusion matrix
   file_name_root = paste(underlying_name, "_", formula_idx,sep="")
   file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
   cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
   save_to_csv(qda.confusion_table, output_dir,file_name)
  }
}
```

Code - ./models/qda.r

LDA Model

```
#author: Frederic Marechal
#date: April-Sep, 2017
#the test_data_param could be a traning/validation or test dataset
run_model_lda = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")
  cat (paste("\nRunning: ", model_name, " formula idx: ",formula_idx, " formula: ", formula_as_srtring ," for time
window: ", tw_index ,"\n", sep=""))
  #caret implementation
  set.seed(1)
  preprocess_training_data = preProcess(training_data, method=c("BoxCox"))
  preprocess_training_data =predict( preprocess_training_data , training_data )
   set.seed(1)
  lda.fit = NULL
  lda.fit = tryCatch( train ( formula,
                        preprocess_training_data,
                        method = "lda",
                         metric="kappa"),
                  error=function(e) next)
  if (is.null(lda.fit) == FALSE){
    #Generate the prediction
    lda.pred = predict(lda.fit, test_data)
```

```
    #confusionMatrix(lda.pred, test_data$Direction)
    #Generate the confusion matrix
    lda.confusion_table =  table(lda.pred, test_data$Direction)
    #Patch the confusion matrix to ensure a 3*3 schema is always present
    #lda.confusion_table = patch_confusion_table(lda.confusion_table)
     #Save confusion matrix
    file_name_root = paste(underlying_name, "_", formula_idx,sep="")
    file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
    cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
    save_to_csv(lda.confusion_table, output_dir,file_name)
   }
}
```

Code - .. ./models/lda.r

PDA Model

```
#author: Frederic Marechal
#date: April-Sep, 2017
#the test_data_param could be a traning/validation or test dataset
run_model_pda = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")
  cat (paste("\nRunning: ", model_name, " formula idx: ",formula_idx, " formula: ", formula_as_srtring ," for time
window: ", tw_index ,"\n", sep=""))
  print (optimised_parameters)
  if (is.null(optimised_parameters)){
   print("using training parameters....")
   the_lambda =c(0,0.001,0.001,0.005,0.1,0.5,1)
  }else {
   print("using optmised parameters....")
   the_lambda  = c(optimised_parameters$hyper_parameters$lambda )
  }
  #caret implementation
  set.seed(1)
  preprocess_training_data = preProcess(training_data, method=c("BoxCox"))
  preprocess_training_data =predict( preprocess_training_data , training_data )
   set.seed(1)
  pda.fit = NULL
  pda.fit = tryCatch(  train ( formula,
                               preprocess_training_data,
                               method = "pda",
                               metric="kappa",
                               tuneGrid=expand.grid(.lambda=the_lambda)),
                       error=function(e) next)
    if (is.null(pda.fit) == FALSE){
   #Find the best size and decay
   the_lambda = optimised_parameters$hyper_parameters$lambda = as.numeric(pda.fit$bestTune$lambda)
    #Generate the prediction
   pda.pred = predict(pda.fit, test_data)
   #Generate the confusion matrix
   pda.confusion_table =  table(pda.pred, test_data$Direction)
   #Save confusion matrix
   file_name_root = paste(underlying_name, "_",the_lambda,"_", formula_idx,sep="")
   file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
   cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
   save_to_csv(pda.confusion_table, output_dir,file_name)
  }
   return (optimised_parameters)
}
```

Code - .. ./models/pda.r

SVM Model

```
#author: Frederic Marechal
#date: April-Sep, 2017
run_model_svm_linear = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  run_model_svm ("svmLinear", model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
}
run_model_svm_poly = function( model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  run_model_svm ("svmPoly", model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
}
run_model_svm = function(svm_model, model_name, model_short_name, type,scenario,underlying_name,
formula_idx, formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
 output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")
    cat (paste("\nRunning svm: ", svm_model,"model", model_name, " formula idx: ",formula_idx, " formula: ",
formula_as_srtring ," for time window: ", tw_index ,"\n", sep=""))
    print (optimised_parameters)
    formula = get_formula_with_factor(formula)
    #The model above model will be run each cost in the below cost list:
 if (is.null(optimised_parameters)){
   if (svm_model == "svmLinear"){
     print("using training parameters....")
     cost_list = c(0.001 , 0.01 , 0.1, 1 ,100 )
     degree_list = c(1) #default values not in use
     scale_list = c(1) #default values not in use
     gamma_list = c(1) #default values not in use
    }
    else if (svm_model == "svmPoly"){
     #cost_list = c(0.001 , 0.01 , 0.1, 1 ,100 )
     #gamma_list = c(0.01,0.05,0.5,1,100)
     #degree_list = c(2,3)
     cost_list = c(0.001 , 0.01 , 0.1, 1 ,100 )
     gamma_list = c(0.01,0.05,0.5,1,100)
     scale_list = c(1)
     degree_list = c(2,3)
    }
 } else {
     print("using optmised parameters for cost and gamma....")
     cost_list = c(optimised_parameters$hyper_parameters$cost)
     gamma_list = c(optimised_parameters$hyper_parameters$gamma)
     scale_list = c(optimised_parameters$hyper_parameters$scale)
     degree_list = c(optimised_parameters$hyper_parameters$degree)
 }
 if (svm_model == "svmLinear"){
   the_tuneGrid=expand.grid(.C=cost_list)
 }else if (svm_model == "svmPoly"){
   the_tuneGrid=expand.grid(.C=cost_list,.gamma = gamma_list, .scale = scale_list,  .degree = degree_list )
 }
 #https://topepo.github.io/caret/train-models-by-tag.html#Support_Vector_Machines
 #https://stats.stackexchange.com/questions/82162/cohens-kappa-in-plain-english
 #http://web2.cs.columbia.edu/~julia/courses/CS6998/Interrater_agreement.Kappa_statistic.pdf
 #http://www.personality-project.org/r/html/kappa.html
 #http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/
 set.seed(1)
 svm.fit = NULL
 svm.fit =  tryCatch(  train ( formula,
                               data=training_data,
                               method = svm_model,
                               trace = FALSE,
                               preProc = c("center","scale"),
                               metric = "Kappa",
                               probability = TRUE, #indicates the model should allow for probability predictions
```

```
                        tuneGrid=the_tuneGrid),
                    error=function(e) next)
  if (is.null(svm.fit) == FALSE){
    #Find the best size and decay
    if (svm_model == "svmLinear"){
      the_cost = optimised_parameters$hyper_parameters$cost = as.numeric(svm.fit$bestTune$C)
      the_gamma = optimised_parameters$hyper_parameters$gamma = 1
      the_scale = optimised_parameters$hyper_parameters$scale = 1
      the_degree = optimised_parameters$hyper_parameters$degree = 1
    }else if (svm_model == "svmPoly"){
      the_cost = optimised_parameters$hyper_parameters$cost = as.numeric(svm.fit$bestTune$C)
      the_gamma = optimised_parameters$hyper_parameters$gamma = as.numeric(svm.fit$bestTune$gamma)
      the_scale = optimised_parameters$hyper_parameters$gamma = as.numeric(svm.fit$bestTune$scale)
      the_degree = optimised_parameters$hyper_parameters$degree = as.numeric(svm.fit$bestTune$degree)
    }
    #Generate the prediction
    svm.pred = predict(svm.fit, test_data)
    #Generate the confusion matrix
    svm.confusion_table =  table(svm.pred, test_data$Direction)
    #Save confusion matrix
    file_name_root = paste(underlying_name, "_", the_cost, "_", the_gamma, "_", the_scale, "_", the_degree, "_",
formula_idx,sep="")
    file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
    cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
    save_to_csv(svm.confusion_table, output_dir,file_name)
  }
    return (optimised_parameters)
}
```

Code - .. ./models/svm.r

Random Forest

```
#author: Frederic Marechal
#date: April-Sep, 2017
run_model_rf_ntree_1000 = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  run_model_rf(1000,model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
}

run_model_rf_ntree_500 = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  run_model_rf(500,model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
}
run_model_rf_ntree_1500 = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
    print (output_dir)
    run_model_rf(1500,model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
}
#the test_data_param could be a traning/validation or test dataset
run_model_rf = function(the_ntree, model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
    output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")
    cat (paste("\nRunning rf: ", the_ntree, "model",  model_name, " formula idx: ",formula_idx, " formula: ",
formula_as_srtring ," for time window: ", tw_index ,"\n", sep=""))
   if (is.null(optimised_parameters)){
     #count the number of '+' that occur in the formula and add one to get the number of attributes
     p = sapply(regmatches(as.character(formula)[3], gregexpr("\\+", as.character(formula)[3])), length)+1
     #try=p is the bagging version
     tunegrid = expand.grid(mtry=c(p, ceiling(sqrt(p)), ceiling(p/2)))
   }else{
     tunegrid = expand.grid(mtry=c(optimised_parameters$mtry))
```

```
  }
  set.seed(1)
  rf.fit = NULL
  rf.fit = tryCatch( train ( formula,
                        data = training_data,
                        method = "parRF",
                        metric = "Kappa",
                        ntree = the_ntree,
                        tuneGrid=tunegrid,
                        importance = TRUE),
                error=function(e) next)
   if (is.null(rf.fit) == FALSE){
    #Find the best mtry
    the_mtry= optimised_parameters$mtry = as.numeric(rf.fit$bestTune$mtry)
    print (paste("best mtry:", the_mtry, sep=""))
    #Step 1 - Generate the prediction
    #Generate the prediction
    rf.pred = predict(rf.fit, test_data)
    #Generate the confusion matrix
    rf.confusion_table =  table(rf.pred, test_data$Direction)
     #Save confusion matrix
    file_name_root = paste(underlying_name, "_",the_ntree,"_",the_mtry,"_", formula_idx,sep="")
    file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
    cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
    save_to_csv(rf.confusion_table, output_dir,file_name)
   }
  return (optimised_parameters)
 }
```

Code - .. ./models/rf_500.r

Multi Layer Perceptron Weighted Decay

```
#author: Frederic Marechal
#date: April-Sep, 2017
#the test_data_param could be a traning/validation or test dataset
run_model_mlp_weight_decay = function(model_name, model_short_name, type,scenario,underlying_name,
formula_idx, formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")
  cat (paste("\nRunning: ", model_name, " formula idx: ",formula_idx, " formula: ", formula_as_srtring ," for time
window: ", tw_index ,"\n", sep=""))
  print (optimised_parameters)

  if (is.null(optimised_parameters)){
    print("using training parameters....")
    the_size = seq(1:15)
    the_decay=c(0,0.001,0.1,1)
  }else{
    print("using optmised parameters....")
    the_size = c(optimised_parameters$hyper_parameters$size)
    the_decay = c(optimised_parameters$hyper_parameters$decay)
  }
    #generate columns lists
  cols = c('Log_returns')
  cols_no_return = c()
  for (attribute in all.vars(formula))
  {
    if (attribute != "Direction"){
      cols[length(cols)+1] = attribute
    }
  }
   set.seed(1)
  mlp.fit = NULL
  mlp.fit = tryCatch( train( formula,
                        data = training_data,
                        method='nnet',
```

```
                        trace = FALSE,
                        preProcess = c('center', 'scale'),
                        metric = "Kappa",
                        maxit = 1000,
                        tuneGrid=expand.grid(.size=the_size,.decay=the_decay)),
                    error=function(e) next)
    if (is.null(mlp.fit) == FALSE){
    #Find the best size and decay
    the_size = optimised_parameters$hyper_parameters$size = as.numeric(mlp.fit$bestTune$size)
    the_decay = optimised_parameters$hyper_parameters$decay = as.numeric(mlp.fit$bestTune$decay)
    #Generate the prediction
    mlp.pred = predict(mlp.fit, test_data)
    #Generate the confusion matrix
    mlp.confusion_table =  table(mlp.pred, test_data$Direction)
     #Save confusion matrix
    file_name_root = paste(underlying_name, "_",the_size,"_",the_decay,"_", formula_idx,sep="")
    file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
    cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
    save_to_csv(mlp.confusion_table, output_dir,file_name)
   }
    return (optimised_parameters)
 }
```

Code - .. ./models/mlp_weight_decay.r

Elman/Jordan Recursive Neural Networks

```
#author: Frederic Marechal
#date: April-Sep, 2017
run_model_elman = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  run_rnn("elman",model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
 }
run_model_jordan = function(model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  run_rnn("jordan",model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir)
 }
run_rnn = function(rnn_model, model_name, model_short_name, type,scenario,underlying_name, formula_idx,
formula,optimised_parameters, number_sliding_windows,tw_index, training_data,test_data, output_dir){
  output_dir = paste(output_dir,model_short_name, "/",number_sliding_windows,"/confusion.matrix/",
type,"/",scenario,"/",sep="")

  cat (paste("\nRunning rnn: ", rnn_model, "model", model_name, " formula idx: ",formula_idx, " formula: ",
formula_as_srtring ," for time window: ", tw_index ,"\n", sep=""))
  print (optimised_parameters)
  if (is.null(optimised_parameters)){
    print("using training parameters....")
    the_size = c(5,7,10,15,20)
    the_iterations = c(100,500,1000,1500,2000)
    the_decay=c(0,0.001,0.01,0.1,1)
  }else{
    print("using optmised parameters....")
    the_size = c(optimised_parameters$hyper_parameters$size)
    the_iterations = c(optimised_parameters$hyper_parameters$iterations)
    the_decay = c(optimised_parameters$hyper_parameters$decay)
  }
  #merge the training and test sets to generate the lag time-series
  full_data = rbind(training_data,test_data)
  #generate columns lists
  cols = c('Log_returns')
  cols_no_return = c()
  for (attribute in all.vars(formula))
  {
    if (attribute != "Direction"){
```

```
    cols[length(cols)+1] = attribute
    cols_no_return[length(cols_no_return)+1] = attribute
  }
}
#reduce the col list to the list of interest
full_data = full_data [,cols]
#scale the data
full_data = scale(full_data, center = TRUE, scale = TRUE)

#get the lagged explained variables
full_explanatory = full_data[,cols_no_return]
full_explanatory = as.zoo(full_explanatory)
X0 = lag(full_explanatory, k=0)
full_explanatory=cbind(X0)
full_explanatory= pre_process(df=data.frame(full_explanatory),method = "knnImpute")
test_range = (dim(full_explanatory)[1]-dim(test_data)[1]+1):dim(full_explanatory)[1]
training_explanatory = full_explanatory[-test_range,]
training_response = full_data[-test_range,c('Log_returns')]
test_explanatory = full_explanatory[test_range,]
test_response = full_data[(dim(full_data)[1]-length(test_range)+1):dim(full_data)[1],c('Log_returns')]
curr_accuracy_rate = 0
curr_size = the_size[1]
curr_iterations = the_iterations[1]
curr_decay = the_decay[1]
init_confusion_matrix = load_from_csv(paste(getwd(),"/schemas/",sep=""),"confusion_matrix")
curr_confusion_matrix = init_confusion_matrix
for (size in the_size)
{
  for (iterations in the_iterations)
  {
    for (decay in the_decay)
    {
      print (paste ("size:", size, "- iterations:", iterations,"- decay: ", decay, paste=""))
      set.seed(1)
      rnn.fit = NULL
      if (rnn_model == "jordan"){
        rnn.fit = tryCatch( jordan (  x=training_explanatory,
                                      y=training_response,
                                     size =c(size),
                                     learnFuncParams =c(decay),
                                     maxit =iterations,
                                     linOut=TRUE,
                                     inputsTest = test_explanatory,
                                     targetsTest = test_response),
                            error=function(e) next)
      }else{
        rnn.fit = tryCatch( elman ( x=training_explanatory,
                                    y=training_response ,
                                    size =c(size),
                                    learnFuncParams =c(decay),
                                    maxit =iterations,
                                    linOut=TRUE,
                                    inputsTest = test_explanatory,
                                    targetsTest = test_response),
                            error=function(e) next)
      }
      if (is.null(rnn.fit) == FALSE){
        #generate the prediction
        pred = predict (rnn.fit, test_explanatory)
        #reset the confusion matrix
        confusion_matrix = init_confusion_matrix
        #create the confusion matrix
        #print(paste("response:", test_response, sep=""))
        print(paste("response len:", pred, sep=""))
        for (index in 1:length(test_response)){
          expected = as.numeric(test_response[index])
          predicted = as.numeric(pred[index])
```

```
          #do not update the confusion matrix if no prediction
          if (is.na(predicted) == TRUE){
            print ("predicted value is NA")}
          else{
            #print(paste("expected: ", expected))
            #print(paste("predicted: ", predicted))
            if (expected < 0 && predicted < 0){
              confusion_matrix[1,1] = confusion_matrix[1,1] + 1
              #print("exp:down pred:down")
              #print(confusion_matrix[1,1])
            }else if (expected < 0 && predicted > 0){
              confusion_matrix[2,1] = confusion_matrix[2,1] + 1
              #print("exp:down pred:up")
              #print(confusion_matrix[2,1])
            }else if (expected > 0 && predicted > 0){
              confusion_matrix[2,2] = confusion_matrix[2,2] + 1
              #print("exp:up pred:up")
              #print(confusion_matrix[2,2])
            }else{
              confusion_matrix[1,2] = confusion_matrix[1,2] + 1
              #print("exp:up pred:down")
              #print(confusion_matrix[1,2])
            }
          }
        }
        print(confusion_matrix)
        #calculate the accuracy rate
        accuracy_rate = (confusion_matrix[1,1] + confusion_matrix[2,2])/sum(confusion_matrix)
        if (is.na(accuracy_rate)){accuracy_rate = 0}
        if (accuracy_rate > curr_accuracy_rate){
            curr_accuracy_rate = accuracy_rate
            curr_confusion_matrix = confusion_matrix
            curr_size = size
            curr_iterations = iterations
            curr_decay = decay
        }
       }
      }
     }
    #Set the best hyper params
    optimised_parameters$hyper_parameters$size = curr_size
    optimised_parameters$hyper_parameters$iterations = curr_iterations
    optimised_parameters$hyper_parameters$decay = curr_decay
    #Save confusion matrix
    file_name_root = paste(underlying_name, "_",curr_size,"_",curr_decay,"_",curr_iterations,"_", formula_idx,sep="")
    file_name = paste(file_name_root ,"_", tw_index,".csv",sep="")
    cat (paste("\nSave confusion matrix for ", model_short_name, ": ", file_name, " in: ", output_dir, "\n", sep=""))
    save_to_csv(curr_confusion_matrix, output_dir,file_name)
    }
  return (optimised_parameters)
 }
```

Code - ../models/elman_jordan.r

Appendix 10 – The Data Exploration Service

This code corresponds to the time window generation used by the machine learning models. The code can be found in the following places:

Code Location	Description
../data.explorator.batch.svc.rmd	Generates the specialise analysis result sets and graphs (e.g. ks test, missing value tests, etc..)
../data.explorator.r	This is a helper function for data exploration

````
---
#Markdown doc generation: http://rmarkdown.rstudio.com/word_document_format.html
#Markdown cheat sheet:
#http://nestacms.com/docs/creating-content/markdown-cheat-sheet
#http://rmarkdown.rstudio.com/authoring_basics.html
title: "Asset DataSet Geneator Service"
author: Frederic Marechal
date: April-Sep, 2017
output: word_document
---

################################################################################
################################################################################
#######              Clean-Up
################################################################################
################################################################################
```{r cleanup}
#Clear console
cat("\014")
```

################################################################################
################################################################################
#######              Directory Config
################################################################################
################################################################################
```{r working_directory_config}
is_server_run = FALSE
if(is_server_run){
  setwd('/host/dsm1/fmare001/stats/svm/deliverables')
}else{
  setwd('C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/Thesis/Finance/z_Code')
}

index_name = 'XLE'
number_sliding_windows = 100

input_dir = paste(getwd(), "/data/", index_name,sep="")

schema_dir = paste(getwd(), "/schemas/",sep="")
input_dir = paste(getwd(), "/data/",index_name,"/processed/first_cut/",sep="")
input_final_dir = paste(getwd(), "/data/",index_name,"/processed/final/",sep="")
input_models_dir = paste(getwd(), "/data/",index_name,"/processed/models/",sep="")
input_dir_weights = paste(getwd(), "/data/",index_name,"/raw/constituents.weight/",sep="")
input_dir_results = paste(getwd(), "/results/",sep="")
input_dir_models_rf_vip = paste(getwd(), "/results/vip_rf/models/",sep="")
time_windows_input_dir = paste(getwd(), "/data/", index_name, "/processed/time.windows/",number_sliding_windows,
"/",sep="")
output_dir = paste(getwd(), "/data/",index_name,"/processed/data.exploration/",sep="")
graph_output_dir = paste(getwd(), "/graphs/",index_name, "/data.exploration/",sep="")
```
````

```
################################################################################
################################################################################
#######              Load Dependencies
################################################################################
################################################################################
```{r load_depedencies}
source("utilities.r")
source("data.explorator.r")
source("statisitical.tests.r")
require(dplyr)
require(reshape2)
require(ggplot2)
require(Hmisc)
require(fBasics)
require(caTools)
require(corrplot)
```

##
##
#######   Directory Clean-up
##
##
```{r load_directory_cleanup}
#delete_all_files_in_directory(output_dir)
```

##
##
#######   Generator As Batch
##
##
```{r dataset_generator}
weights_df = load_from_csv(input_dir_weights, index_name)
schema_df = load_from_csv(schema_dir, "default")
exploration_files = list.files(input_dir)

len = length(exploration_files)
for (i in 1:len) {
  stock_name = as.character(exploration_files[i])
  print(stock_name)

  df = load_from_csv(input_dir, stock_name,TRUE )

  schema_df[i,"underlying_name"] = stock_name
  schema_df[i,"total_row_count"] = get_total_row_count(df)
  schema_df[i,"start_date_index"] = 1
  schema_df[i,"start_date"] = get_start_date(df)
  end_date_index = get_end_date_index(df)
  schema_df[i,"end_date_index"] = end_date_index
  schema_df[i,"end_date"] = get_date(end_date_index)
  start_date_sentiment_index = get_start_date_index_sentiment(df)
  schema_df[i,"start_date_index_sentiment"] = start_date_sentiment_index
  schema_df[i,"start_date_sentiment"] = get_date(start_date_sentiment_index)
  schema_df[i,"count_days_sentiment"] = end_date_index - start_date_sentiment_index
  start_date_fundamental_index = get_start_date_index_fundamental(df)
  schema_df[i,"start_date_index_fundamental"] = start_date_fundamental_index
  schema_df[i,"20days_ma_missing_perc"] = get_missing_perc(20,end_date_index)
  schema_df[i,"50days_ma_missing_perc"] = get_missing_perc(50,end_date_index)
  schema_df[i,"100days_ma_missing_perc"] = get_missing_perc(100,end_date_index)
  schema_df[i,"200days_ma_missing_perc"] = get_missing_perc(200,end_date_index)
  schema_df[i,"sentiment_missing_perc"] = 100
  if (is.na(start_date_sentiment_index) == FALSE){
    schema_df[i,"sentiment_missing_perc"] =
get_range_missing_perc(df,start_date_sentiment_index,end_date_index,"Article_sentiment" )
    generate_missing_graph(stock_name, df, "Log_returns", "Article_sentiment", start_date_sentiment_index,
end_date_index,output_dir, "Sentiment")
```
```

```
  }
  schema_df[i,"fundamental_missing_perc"] = 100
  if (is.na(start_date_fundamental_index) == FALSE){
    schema_df[i,"fundamental_missing_perc"] =
get_range_missing_perc(df,start_date_fundamental_index,end_date_index,"Price_earning_ratio" )
    generate_missing_graph(stock_name, df, "Log_returns", "Price_earning_ratio", start_date_fundamental_index,
end_date_index,output_dir, "Fundamental")
  }
  weight_index = get_weight_index(weights_df, remove_file_extension(stock_name))
  schema_df[i,"weight_perc"] = NA
  if (is.na(weight_index) == FALSE){
    weight_str = as.character(weights_df[weight_index, "Index.Weight"])
    weight = as.numeric(substr(weight_str,1,nchar(weight_str)-1))
    schema_df[i,"weight_perc"] = weight
  }
  schema_df[i,"zero_log_returns_count"] = get_zero_log_returns_count(df)
  schema_df[i,"zero_log_returns_perc"] =
get_perc(schema_df[i,"zero_log_returns_count"],schema_df[i,"total_row_count"])
  schema_df[i,"pos_log_returns_count"] = get_pos_log_returns_count(df)
  schema_df[i,"pos_log_returns_perc"] =
get_perc(schema_df[i,"pos_log_returns_count"],schema_df[i,"total_row_count"])
  schema_df[i,"neg_log_returns_count"] = get_neg_log_returns_count(df)
  schema_df[i,"neg_log_returns_perc"] =
get_perc(schema_df[i,"neg_log_returns_count"],schema_df[i,"total_row_count"])
 }

 #Save
 file_name = "data_exploration.csv"
 cat (paste("\nSave data exploration: ", file_name, " to: ", output_dir, "\n", sep=""))
 save_to_csv(schema_df, output_dir, file_name)
 ```

 ##
 ##
 #######   Explanatory Graph Generators
 ##
 ##
 ```{r dataset_graph}
 weights_df = load_from_csv(input_dir_weights, index_name)
 schema_df = load_from_csv(schema_dir, "default")
 exploration_files = list.files(input_dir)

 len = length(exploration_files)
 for (i in 1:len) {
  stock_name = as.character(exploration_files[i])
  print(stock_name)

  df = read.csv(paste(input_dir,stock_name, sep=""), stringsAsFactors=FALSE)
  df = df[1:(dim(df)[1]-1), c("DateAsDate","Log_returns", "Close")]

  par(mfrow=c(1,2))

  date_format = "%d-%m-%Y"
  from_date = as.Date("31-12-1980",date_format)
  to_date = as.Date("30-12-2016",date_format)
  plot_time_series(remove_file_extension(stock_name), df,from_date, to_date, "Close", graph_output_dir)
  plot_time_series(remove_file_extension(stock_name), df,from_date, to_date, "Log_returns" , graph_output_dir)

  par(mfrow=c(1,1))
 }
 ```
```
```

```
################################################################################
################################################################################
#######              Descriptive Stats
################################################################################
################################################################################
```{r dataset_stats}
weights_df = load_from_csv(input_dir_weights, index_name)
schema_df = load_from_csv(schema_dir, "default")
exploration_files = list.files(input_dir)
res = {}

len = length(exploration_files)
for (i in 1:len) {
  stock_name = as.character(exploration_files[i])
  print(stock_name)
  df = read.csv(paste(input_dir,stock_name, sep=""), stringsAsFactors=FALSE)
  df = df[1:(dim(df)[1]-1), c("Log_returns")]
  stats =   basicStats(df, ci = 0.95)
  if (i == 1) {
    res = stats
  }else{
    res = cbind(res,stats)
  }
}
res
```
##
##
#######   The K-S test
##
##
```{r dataset_graph}
weights_df = load_from_csv(input_dir_weights, index_name)
schema_df = load_from_csv(schema_dir, "ks_results")
exploration_files = list.files(input_dir)

directions = c('Up','Down')
attributes =
c('Close','Close_price_1day_lag','Close_price_2day_lag','Close_price_3day_lag','Close_price_4day_lag','Close_price_5day_lag','Sma20','Sma50','Sma100','Sma200','Ema20','Ema50','Ema100','Ema200','Roc_1day','Roc_2day','Roc_5day','Roc_10day','Roc_20day','Momentum_5day','Momentum_10day','Momentum_20day','MomentumAbs_5day','MomentumAbs_10day','MomentumAbs_20day','Wpr_5day','Wpr_10day','Wpr_20day','Rsi_5day','Rsi_10day','Rsi_14day','Rsi_20day','Cmo_5day','Cmo_10day','Cmo_20day','Atr_tr','Atr_atr','Atr_trueHigh','Atr_trueLow','Smi_smi','Smi_signal','Stoch_fastk','Stoch_fastd','Stoch_slowd','Macd_macd','Macd_signal','Bb_dn','Bb_mavg','Bb_up','Bb_pctB','Volatility','Mfi','Sar','Volume')
idx = 1
len = length(exploration_files)
for (i in 1:len) {
  stock_name = as.character(exploration_files[i])

  for (attribute in attributes){
    print(paste (stock_name, "-", attribute), sep="")

    df = read.csv(paste(input_dir,stock_name, sep=""), stringsAsFactors=FALSE)
    print('**********************************')
    for (direction in directions) {
      df_test =  subset(subset(df, select=c("Direction",attribute)), df$Direction == direction)
      res = kolmogorov_smirnov_normal_distribution_test(df_test[attribute], paste("", attribute, "/Direction = '",
direction, " is normally distributed", sep=""))

      schema_df[idx,"code"] = remove_file_extension(stock_name)
      schema_df[idx,"attribute_name"] = attribute
      schema_df[idx,"direction"] = direction
      schema_df[idx,"result"] = res
      idx = idx+1
    }
```
```

```
    print('**********************************')
  }

  save_to_csv(schema_df, output_dir, "ks_test_results.csv")
}

#a few graphs
stock_name = "XLE.csv"
df = read.csv(paste(input_dir,stock_name, sep=""), stringsAsFactors=FALSE)
par(mfrow=c(2,2))
stock_name = remove_file_extension(stock_name)
plot_qqplot (df$Rsi_20day, paste(stock_name, " - Rsi_20day/Direction = Up", sep=""))
plot_qqplot (df$Rsi_20day, paste(stock_name, " - Rsi_20day/Direction = Down", sep=""))
plot_qqplot (df$Sma20, paste(stock_name, " - Sma20/Direction = Up", sep=""))
plot_qqplot (df$Sma20, paste(stock_name, " - Sma20/Direction = Down", sep=""))
par(mfrow=c(1,1))
```
```
################################################################################
################################################################################
#######               Result Aggregator
################################################################################
################################################################################
```{r dataset_stats}
add_to_report_list = function(input_dir, model_name, feature_selection_type,type, scenario, target_folder, weights_df){
  len = dim(weights_df)[1]
  for (i in 1:len) {
    name = weights_df$Symbol[i]
    long_name = weights_df$Company.Name[i]
    dir = paste(input_dir,feature_selection_type, "/models/",model_name, target_folder, type, "/", scenario,"/", sep="")
    #get file
    file_name = dir(dir,pattern = paste("summary_performance_",name, sep=""))
    #select the last but one file when there is more than one available
    nb_files = length(file_name)
    if (nb_files>1){
      file_name = file_name[nb_files-1]
    }
    #get the data
    idx= dim(results_schema_df)[1]+1
    #share data
    results_schema_df[idx,"code"] = name
    results_schema_df[idx,"type"] = type
    results_schema_df[idx,"scenario"]= scenario
    results_schema_df[idx,"model_name"]= model_name
    results_schema_df[idx,"asset_type"] = "stock"
    results_schema_df[idx,"feature_selection_type"]= feature_selection_type
    if (name == "XLE"){
       results_schema_df[idx,"asset_type"] = "index"
    }
    results_schema_df[idx,"long_name"] = long_name
    #specific data
     if (file.exists(paste(dir,file_name, sep=""))){
      #load summary data
      report_results_df =  load_from_csv(dir, file_name[1], TRUE)
      root = toString(report_results_df[1,"file_name_root"])
      #load formula data
      formula_dir = paste(input_dir,feature_selection_type, "/models/", sep="")
      formula_file_name = dir(formula_dir,pattern = paste("Formula_",root, sep=""))

      report_formula_df =  load_from_csv(formula_dir, formula_file_name, TRUE)
      #perform mapping
      results_schema_df[idx,"state"] = "OK"
      results_schema_df[idx,"feature_list"]= paste(toString(report_formula_df$formula_as_srtring))

      results_schema_df[idx,"best_in_class"]= 0
      results_schema_df[idx,"hyper_parameters"]= "NA"

      results_schema_df[idx,"file_name_root"]= root
```
```

```
    results_schema_df[idx,"accuracy_rate_avg"]=report_results_df[1,"accuracy_rate_avg"]
    results_schema_df[idx,"accuracy_rate_std"]=report_results_df[1,"accuracy_rate_std"]
    results_schema_df[idx,"accuracy_rate_med"]=report_results_df[1,"accuracy_rate_med"]
    results_schema_df[idx,"accuracy_rate_min"]=report_results_df[1,"accuracy_rate_min"]
    results_schema_df[idx,"accuracy_rate_max"]=report_results_df[1,"accuracy_rate_max"]
    results_schema_df[idx,"error_rate_avg"]=report_results_df[1,"error_rate_avg"]
    results_schema_df[idx,"consider_pabak"]=report_results_df[1,"consider_pabak"]
    results_schema_df[idx,"kappa"]=report_results_df[1,"kappa"]
    results_schema_df[idx,"kappa_lower"]=report_results_df[1,"kappa_lower"]
    results_schema_df[idx,"kappa_upper"]=report_results_df[1,"kappa_upper"]
    results_schema_df[idx,"pabak"]=report_results_df[1,"pabak"]
    results_schema_df[idx,"pabak_lower"]=report_results_df[1,"pabak_lower"]
    results_schema_df[idx,"pabak_upper"]=report_results_df[1,"pabak_upper"]
    results_schema_df[idx,"sens"]=report_results_df[1,"sens"]
    results_schema_df[idx,"spec"]=report_results_df[1,"spec"]
    results_schema_df[idx,"auc"]=report_results_df[1,"auc"]
    results_schema_df[idx,"dw"]=report_results_df[1,"dw"]
   } else {
    results_schema_df[idx,"state"] = "MISSING"
    print (paste(name, " DOES NOT EXIST!!!!"))}
  }
  return (results_schema_df)
}

weights_df = read.csv(paste(input_dir_weights,index_name,".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df = read.csv(paste(schema_dir,"results",".csv", sep=""), stringsAsFactors=FALSE)
target_folder = "/100/confusion.matrix/"
len = dim(weights_df)[1]
weights_df[len+1,"Symbol"] = "XLE"
weights_df[len+1,"Company.Name"] = "Energy Select Sector"

##########################################################
###### Original - Permutated
##########################################################
results_schema_df = add_to_report_list(input_dir_results,"elman", "permutated", "train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"elman", "permutated", "test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"jordan", "permutated","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"jordan", "permutated","test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"mlp_weight_decay", "permutated","train","original",
target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"mlp_weight_decay", "permutated","test","original",
target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"lda", "permutated","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"lda", "permutated","test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"qda", "permutated","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"qda", "permutated","test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"pda", "permutated","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"pda", "permutated","test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"rf_500", "permutated","train","original", target_folder,
weights_df)
```

```
results_schema_df = add_to_report_list(input_dir_results,"rf_500", "permutated","test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"svmLinear", "permutated","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"svmLinear", "permutated","test","original", target_folder,
weights_df)

#########################################################
###### Original - Gini
#########################################################
results_schema_df = add_to_report_list(input_dir_results,"elman", "gini", "train","original", target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"elman", "gini", "test","original", target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"jordan", "gini","train","original", target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"jordan", "gini","test","original", target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"mlp_weight_decay", "gini","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"mlp_weight_decay", "gini","test","original", target_folder,
weights_df)

results_schema_df = add_to_report_list(input_dir_results,"lda", "gini","train","original", target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"lda", "gini","test","original", target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"qda", "gini","train","original", target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"qda", "gini","test","original", target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"pda", "gini","train","original", target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"pda", "gini","test","original", target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"rf_500", "gini","train","original", target_folder, weights_df)
results_schema_df = add_to_report_list(input_dir_results,"rf_500", "gini","test","original", target_folder, weights_df)

results_schema_df = add_to_report_list(input_dir_results,"svmLinear", "gini","train","original", target_folder,
weights_df)
results_schema_df = add_to_report_list(input_dir_results,"svmLinear", "gini","test","original", target_folder,
weights_df)

#########################################################
###### Create Base file
#########################################################

#join the result sets
results_schema_df_test = results_schema_df[which(results_schema_df$type == "test"),]
results_schema_df_train = results_schema_df[which(results_schema_df$type == "train"),]

results_schema_join= left_join( results_schema_df_test,
                        results_schema_df_train,
                        by = c( "code" = "code",
                              "scenario" = "scenario",
                              "feature_selection_type" = "feature_selection_type",
                              "model_name" = "model_name"))
#Order the cols
results_schema_join = results_schema_join[order( results_schema_join$state.x,
                                    results_schema_join$code,
                                    results_schema_join$type.x,
                                    results_schema_join$scenario,
                                    -results_schema_join$accuracy_rate_avg.x,
                                    results_schema_join$model_name),]

#define cols to keep
columns_to_keep = c( "state.x","code","long_name.x",
                "asset_type.x","feature_selection_type",
                "file_name_root.x", "feature_list.x",
                "scenario","model_name",
                "type.x",
```

```
                "accuracy_rate_avg.x",
                "kappa.x",
                "sens.x","spec.x",
                "type.y",
                "accuracy_rate_avg.y",
                "pabak.y",
                "sens.y","spec.y",
                "best_in_class.x" )

#Generate ordered test accuracy per code and model
res_df = results_schema_join[which( results_schema_join$state.x == "OK"),
                             columns_to_keep,
                             drop = FALSE]

#reset the best_in_class.x flag, to indicate the selected code/formula combination
current_code = res_df[1,"code"]
res_df[1,"best_in_class.x"] = 1
len = dim(res_df)[1]
for (i in 1:len) {
  if (res_df[i,"code"] != current_code){
    res_df[i,"best_in_class.x"] = 1
    current_code =  res_df[i,"code"]
  }
}

#rename col
res_df$type.y = "validation"

#save to csv
save_to_csv(res_df, input_dir_results,"test_accuracy_per_code_and_model_all.csv", row_names =FALSE)

#save to csv
res_df_best_in_class = res_df[which( res_df$best_in_class.x == 1),columns_to_keep,  drop = FALSE]
save_to_csv(res_df_best_in_class, input_dir_results,"test_accuracy_per_code_and_model_best_in_class.csv",
row_names =FALSE)

############################################################
###### Create sentiment aggregated file
############################################################
aggregrate_res = function(model, type, scenario)
{
   the_sent_res_dir = paste(input_models_dir, model, "/100/confusion.matrix/train/",scenario, sep="")
   sent_res_train_files = list.files(path = the_sent_res_dir, pattern = "^summary_performance_")

   for (file_name in sent_res_train_files){

     #train
     results_train = read.csv(paste(the_sent_res_dir,"/",file_name, sep=""), stringsAsFactors=FALSE)
     results_train$code = strsplit(results_train$file_name_root,"_")[[1]][1]
     results_train$model_name=model
     results_train$type="validation"
     results_train$scenario=scenario

     #test
     the_sent_res_dir = paste(input_models_dir, model, "/100/confusion.matrix/test/",scenario, sep="")
     results_test = read.csv(paste(the_sent_res_dir,"/",file_name, sep=""), stringsAsFactors=FALSE)
     results_test$code = strsplit(results_test$file_name_root,"_")[[1]][1]
     results_test$model_name=model
     results_test$type="test"
     results_test$scenario=scenario

     results_join = left_join( results_test,
                          results_train,
                          by = c( "code" = "code",
                                  "scenario" = "scenario",
                                  "model_name" = "model_name"))
```

```
      #left join and accunulate
      if (is.null(results)){
        results <<- results_join
      } else{
        results <<- rbind(results, results_join)
      }
    }
}
sent_res_gen = function(scenario_df){

   scenario_df = scenario_df[order(scenario_df$code),]

 columns_to_keep = c( "scenario",
                      "code",
                      "type.x",
                      "accuracy_rate_avg.x",
                      "kappa.x",
                      "sens.x","spec.x",
                      "type.y",
                      "accuracy_rate_avg.y",
                      "pabak.y",
                      "sens.y","spec.y")

 #Generate ordered test accuracy per code and model
 scenario_df = scenario_df[,columns_to_keep]

 #res =  cbind(base_df,scenario_df)

 return(scenario_df)
}

#read the template schema
result_sent_schemas = read.csv(paste(schema_dir,"sent_results",".csv", sep=""), stringsAsFactors=FALSE)

#load the sent results
results = NULL
models = c("elman", "jordan","lda","mlp_weight_decay", "pda", "qda","rf_500","svmLinear")
scenarios = c("sentiment","sentiment_m1","sentiment_m2","sentiment_m3","sentiment_momentum",
"sentiment_momentum_mm1","sentiment_momentum_mm1m2", "sentiment_momentum_mm1m2m3")
for (model in models)
{
 for (scenario in scenarios)
 {
  aggregrate_res (model, type, scenario)
 }
}

dim(results[results$model_name %in% models & results$scenario == "sentiment" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_m1" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_m2" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_m3" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_momentum" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_momentum_mm1" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_momentum_mm1m2" ,])
dim(results[results$model_name %in% models & results$scenario == "sentiment_momentum_mm1m2m3" ,])

res_df_best_in_class = res_df_best_in_class[order(res_df_best_in_class$code),]

#sentiment
sent = results[results$model_name %in% models & results$scenario == "sentiment" ,]
sent_res = sent_res_gen(sent)
save_to_csv(sent_res, input_dir_results,"sentiment.csv", row_names =FALSE)
```

```
#sentiment_m1
sent_m1 = results[results$model_name %in% models & results$scenario == "sentiment_m1" ,]
sent_m1_res = sent_res_gen(sent_m1)
save_to_csv(sent_m1_res, input_dir_results,"sentiment_m1.csv", row_names =FALSE)

#sentiment_m2
sent_m2 = results[results$model_name %in% models & results$scenario == "sentiment_m2" ,]
sent_m2_res = sent_res_gen(sent_m2)
save_to_csv(sent_m2_res, input_dir_results,"sentiment_m2.csv", row_names =FALSE)


#sentiment_m3
sent_m3 = results[results$model_name %in% models & results$scenario == "sentiment_m3" ,]
sent_m3_res = sent_res_gen(sent_m3)
save_to_csv(sent_m3_res, input_dir_results,"sentiment_m3.csv", row_names =FALSE)

#sent_momemtum
sent_momemtum = results[results$model_name %in% models & results$scenario == "sentiment_momentum" ,]
sent_momentum_res = sent_res_gen(sent_momemtum)
save_to_csv(sent_momentum_res, input_dir_results,"sent_momemtum.csv", row_names =FALSE)

#sent_momemtum_mm1
sent_momemtum = results[results$model_name %in% models & results$scenario == "sentiment_momentum_mm1" ,]
sent_momentum_res = sent_res_gen(sent_momemtum)
save_to_csv(sent_momentum_res, input_dir_results,"sent_momemtum_mm1.csv", row_names =FALSE)

#sentiment_momentum_mm1m2
sent_momemtum = results[results$model_name %in% models & results$scenario == "sentiment_momentum_mm1m2"
,]
sent_momentum_res = sent_res_gen(sent_momemtum)
save_to_csv(sent_momentum_res, input_dir_results,"sentiment_momentum_mm1m2.csv", row_names =FALSE)

#sentiment_momentum_mm1m2m3
sent_momemtum = results[results$model_name %in% models & results$scenario ==
"sentiment_momentum_mm1m2m3" ,]
sent_momentum_res = sent_res_gen(sent_momemtum)
save_to_csv(sent_momentum_res, input_dir_results,"sentiment_momentum_mm1m2m3.csv", row_names =FALSE)

```
##
##
#######   Result Aggregator
##
##
```{r dataset_stats}

result_summary = function(model_name, scenario,type,weights_df,results_schema_df, input_models_dir){

 input_dir = paste(input_models_dir, model_name, "/", number_sliding_windows,"/mse/", type,"/",scenario,sep="")

 len = dim(weights_df)[1]
 for (i in 1:len) {

  idx = dim(results_schema_df)[1]+1

  #get the asset name
  name = weights_df$Symbol[i]
  long_name = weights_df$Company.Name[i]
  res = list.files(path = input_dir, pattern = paste("^",name, sep=""))
  if (length(res)>0){
   #read the feat selection from the file name
   feat_selection = strsplit(strsplit(res, '.csv')[[1]][3], '.txt')
   #load the data, ignoring the first and last line
   df = read.csv(paste(input_dir,"/",res, sep=""), stringsAsFactors=FALSE)
   df = df[2:(dim(df)[1]-1),] #ignore the first and last entry
   #mse = mean(df$MSE)
   #rmse = sqrt(avg_mse)
```

```
    mse = tail(df$MSE,1)
    #sd_mse = sd(df$MSE)
    rmse = tail(df$RMSE,1)

    asset_type = "stock"
    if (name == "XLE"){
        asset_type = "index"
      }

    results_schema_df[idx, "code"] = name
    results_schema_df[idx, "long_name"] = long_name
    results_schema_df[idx, "asset_type"] = asset_type
    results_schema_df[idx, "feature_selection_type"] = "corr"
    results_schema_df[idx, "feature_list"] = feat_selection
    results_schema_df[idx, "scenario"] = scenario
    results_schema_df[idx, "model_name"] = model_name
    results_schema_df[idx, "type"] = type
    results_schema_df[idx, "mse"] = mse
    #results_schema_df[idx, "sd_mse"] = sd_mse
    results_schema_df[idx, "rmse"] = rmse
    results_schema_df[idx, "best_in_class"] = 0
   }
  }
   results_schema_df = results_schema_df[order( results_schema_df$code,
                                        results_schema_df$model_name,
                                        results_schema_df$scenario,
                                        results_schema_df$type),]
   return(results_schema_df)
}

weights_df = read.csv(paste(input_dir_weights,index_name,".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df = read.csv(paste(schema_dir,"results_vol",".csv", sep=""), stringsAsFactors=FALSE)

#add the index to the list
len = dim(weights_df)[1]
weights_df[len+1,"Symbol"] = "XLE"
weights_df[len+1,"Company.Name"] = "Energy Select Sector"
weights_df[len+1,"Index.Weight"] = "100.00%"

#redce the list to the first 20 assets + index
assets_under_analysis = c("XLE",
"XOM","CVX","SLB","PXD","EOG","HAL","OXY","COP","APC","VLO","KMI","PSX","SE","BHI","TSO","APA",
"DVN","WMB","MPC","NBL", "XLE", "XOM","OXY","COP","APC","VLO","KMI")
weights_df = weights_df[weights_df$Symbol %in% assets_under_analysis, ]

#load the original scenarios
empty_results_schema_df = read.csv(paste(schema_dir,"results_vol",".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df_train_j = result_summary ("jordan", "original", "train", weights_df,results_schema_df,
input_models_dir)
empty_results_schema_df = read.csv(paste(schema_dir,"results_vol",".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df_test_j = result_summary ("jordan", "original", "test", weights_df,results_schema_df,
input_models_dir)
results_schema_df_j = cbind( results_schema_df_train_j, results_schema_df_test_j)

empty_results_schema_df = read.csv(paste(schema_dir,"results_vol",".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df_train_e = result_summary ("elman", "original","train", weights_df,results_schema_df,
input_models_dir)
empty_results_schema_df = read.csv(paste(schema_dir,"results_vol",".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df_test_e = result_summary ("elman", "original","test", weights_df,results_schema_df,
input_models_dir)
results_schema_df_e = cbind( results_schema_df_train_e, results_schema_df_test_e)

results_schema_df = rbind( results_schema_df_j, results_schema_df_e)
results_schema_df = results_schema_df[order( results_schema_df$code,
                                        results_schema_df$model_name,
                                        results_schema_df$scenario,
                                        results_schema_df$type),]
```

```
#set the best class flag to the asset/model showing the smaller MSE
len = dim(weights_df)[1]
for (i in 1:len){
  name = weights_df$Symbol[i]
  idx_jordan = which(results_schema_df$code == name & results_schema_df$model_name == "jordan")
  idx_elman = which(results_schema_df$code == name & results_schema_df$model_name == "elman")
  if (length(idx_jordan) == 1 && length(idx_elman) == 1){
    mse_jordan = results_schema_df[idx_jordan ,"mse"]
    mse_elman = results_schema_df[idx_elman ,"mse"]
    if (mse_elman < mse_jordan){
      results_schema_df[idx_elman,"best_in_class"] = 1
    }else{
      results_schema_df[idx_jordan,"best_in_class"] = 1
    }
  }
}

#load the scenarios
#jordan
empty_results_schema_df = read.csv(paste(schema_dir,"results_vol",".csv", sep=""), stringsAsFactors=FALSE)
results_schema_df_j_s1_train = result_summary ("jordan", "sentiment_m1","train",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario", "type","mse", "rmse")]
results_schema_df_j_s1_test = result_summary ("jordan", "sentiment_m1","test", weights_df,empty_results_schema_df,
input_models_dir)[,c("code","model_name", "scenario", "type", "mse", "rmse")]
results_schema_df_j_s1 = cbind( results_schema_df_j_s1_train, results_schema_df_j_s1_test)

results_schema_df_j_mm1_train = result_summary ("jordan", "sentiment_momentum_mm1","train",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_j_mm1_test = result_summary ("jordan", "sentiment_momentum_mm1","test",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_j_mm1 = cbind( results_schema_df_j_mm1_train, results_schema_df_j_mm1_test)

results_schema_df_j_mm1m2_train = result_summary ("jordan", "sentiment_momentum_mm1m2","train",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_j_mm1m2_test = result_summary ("jordan", "sentiment_momentum_mm1m2","test",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_j_mm1m2 = cbind( results_schema_df_j_mm1m2_train, results_schema_df_j_mm1m2_test)

#elman
results_schema_df_e_s1_train = result_summary ("elman", "sentiment_m1","train",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_e_s1_test = result_summary ("elman", "sentiment_m1","test", weights_df,empty_results_schema_df,
input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_e_s1 = cbind( results_schema_df_e_s1_train, results_schema_df_e_s1_test)

results_schema_df_e_mm1_train = result_summary ("elman", "sentiment_momentum_mm1","train",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_e_mm1_test = result_summary ("elman", "sentiment_momentum_mm1","test",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_e_mm1 = cbind( results_schema_df_e_mm1_train, results_schema_df_e_mm1_test)

results_schema_df_e_mm1m2_train = result_summary ("elman", "sentiment_momentum_mm1m2","train",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_e_mm1m2_test = result_summary ("elman", "sentiment_momentum_mm1m2","test",
weights_df,empty_results_schema_df, input_models_dir)[,c("code","model_name", "scenario","type", "mse", "rmse")]
results_schema_df_e_mm1m2 = cbind( results_schema_df_e_mm1m2_train, results_schema_df_e_mm1m2_test)

#column bind the base scenario and other scenarios where model = "jordan"
res_jordan = results_schema_df[which(results_schema_df$model_name == "jordan"),]
res_jordan = cbind( res_jordan,
                    results_schema_df_j_s1,
                    results_schema_df_j_mm1,
                    results_schema_df_j_mm1m2)

#column bind the base scenario and other scenarios where model = "elman"
res_elman = results_schema_df[which(results_schema_df$model_name == "elman"),]
```

```
res_elman = cbind( res_elman,
                   results_schema_df_e_s1,
                   results_schema_df_e_mm1,
                   results_schema_df_e_mm1m2)

#row bind both sets and get the best_in_class = 1 resutls all across
final_res = rbind(res_jordan, res_elman)
final_res = final_res[which(final_res$best_in_class == 1),]
save_to_csv(final_res, input_dir_results,"res_vol.csv", row_names =FALSE )

```
```

Code - ../ data.explorator.batch.svc.rmd

```
#author: Frederic Marechal
#date: April-Sep, 2017
get_total_row_count = function(df){
  return(NROW(df))
}
get_start_date = function(df){
  return(as.character(df[1,"DateAsDate"]))
}
get_end_date_index = function(df){
  return(dim(df)[1])
}
get_start_date_index_sentiment = function(df){
  return(which(!is.na(df$Article_sentiment))[1])
}
get_start_date_index_fundamental = function(df){
  return(which(!is.na(df$Price_earning_ratio))[1])
}

get_date = function(idx){
  return(as.character(df[idx,"DateAsDate"]))
}
get_missing_perc = function(count, total){
  return(round(100.0*count/total, digits = 2))
}
count_na = function(df){
  return(sum(is.na(df)))
}
get_range_missing_perc = function(df,start_idx, end_idx, col_name){
  missing_df = df[start_idx:end_idx,col_name]
  count = count_na(missing_df)
  total = get_total_row_count(missing_df)
  res = get_missing_perc(count, total)
  return(res)
}
#Generate missing data graph
generate_missing_graph = function (underlying_name,df, col_name_1, col_name_2, start_date_index,
end_date_index,output_dir,title){
  index_col_name_1 = which(names(df)==col_name_1)
  index_col_name_2 = which(names(df)==col_name_2)
  plot_df = cbind(df[start_date_index:end_date_index,index_col_name_1 ],
df[start_date_index:end_date_index,index_col_name_2])
  colnames(plot_df) <- c(col_name_1, col_name_2)
  name = remove_file_extension(underlying_name)
  ggplot_missing(plot_df,name,title)
}
#Code copied from http://www.njtierney.com/r/missing%20data/rbloggers/2015/12/01/ggplot-missing-data/
#and modfiied by Fred Marechal Jun-sep 2017
ggplot_missing = function(x,underlying_name,title){
    x %>%
    is.na %>%
    melt %>%
    ggplot(data = ., aes(x = Var2,y = Var1)) +
    ggtitle(paste ("   ", title, " missing data for: ", underlying_name, sep="")) +
    geom_raster(aes(fill = value)) +
```
```

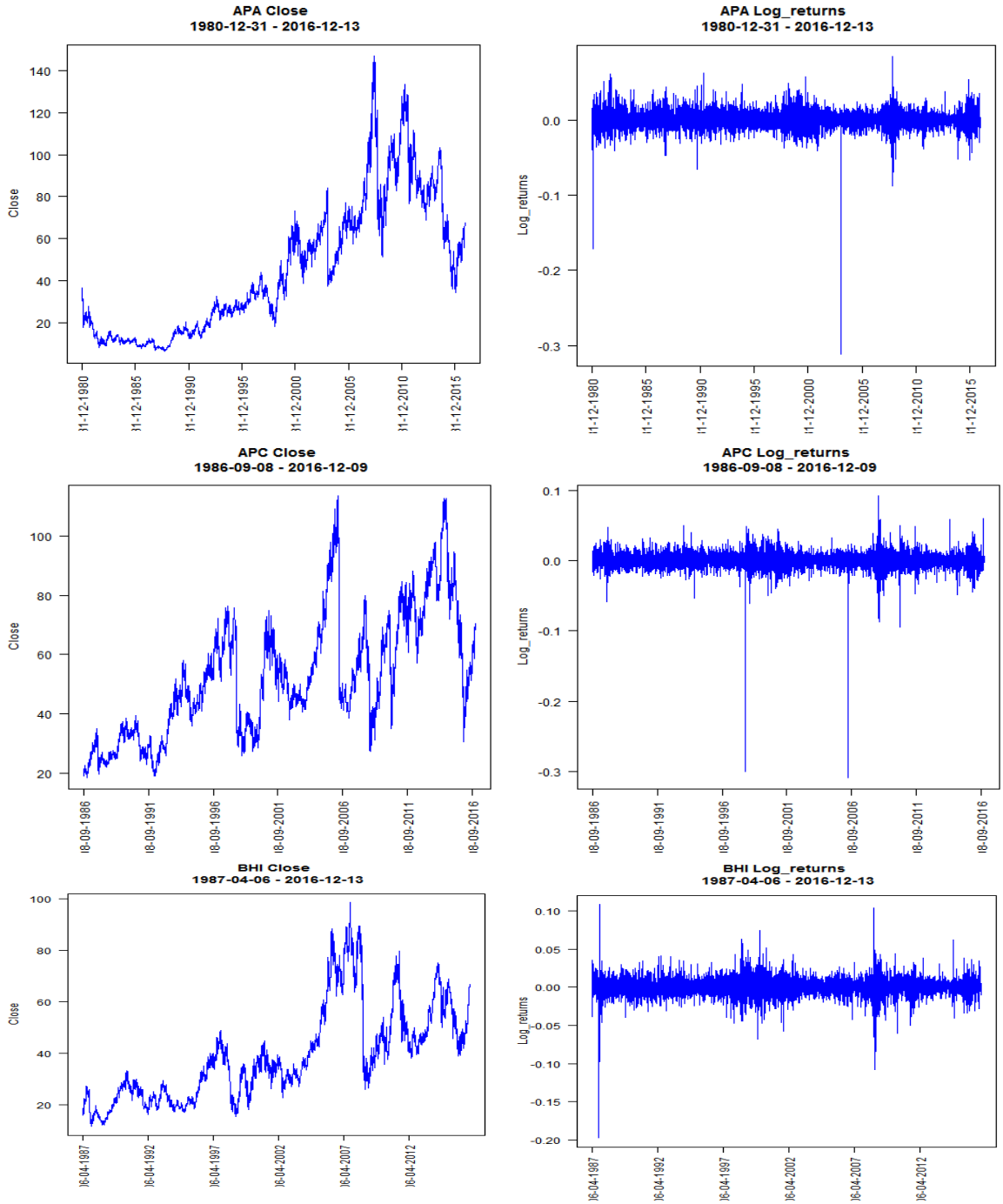
```
    scale_fill_grey(name = "", labels = c("Present","Missing")) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle=45, vjust=0.5)) +
    labs(x = "Variables in Dataset", y = "Rows / observations")
}
get_weight_index = function(df, underlying_name){
  return(which(toupper(df$Symbol) == toupper(underlying_name)))
}
get_zero_log_returns_count = function (df){
  return ( length( which( df$Log_returns == 0 ) ) )
}
get_pos_log_returns_count = function (df){
  return (length(which(df$Log_returns > 0)))
}
get_neg_log_returns_count = function(df){
  return (length(which(df$Log_returns < 0)))
}
get_perc = function (count,total){
  return (100*count/total)
}
##############################################
################ Graphs ######################
##############################################
get_from_date = function (stock_df, from_date){
  #If the min date of the dataset > from_date , then reset the from_date to min_date
  min_date = min(as.Date(stock_df$DateAsDate))
  if (min_date>from_date) {
    from_date = min_date
  }
  return (as.Date(from_date))
}
get_to_date = function (stock_df, to_date){
  #If to_date no           t provided, then default to max date
  if (length(to_date)<0){
    to_date = max(as.Date(stock_df$DateAsDate))
  }
  else {
    #If the max date of the dataset < to_date , then reset the to_date to max_date
    max_date = max(as.Date(stock_df$DateAsDate))
    if (to_date > max_date ) {to_date = max_date }
  }
  return (as.Date(to_date))
}
#This function plots the relative strength index (rsi)
plot_time_series = function (stock_name, stock_df, from_date, to_date, y_col_name, graph_output_dir){
  step_by = "60 mon"
  date_format = "%d-%m-%Y"
  from_date = get_from_date(stock_df, from_date);
  to_date = get_to_date(stock_df, to_date);
  #Only plot when dates are consistent
  if(to_date> from_date){
    #Reduce the data range to the required time range
    stock_df = stock_df[as.Date(stock_df$DateAsDate) >=from_date & as.Date(stock_df$DateAsDate) <=to_date, ]
     #Get the highest y value for the plot
    max_height = max(stock_df[,c(y_col_name)], na.rm = TRUE)
    min_height = min(stock_df[,c(y_col_name)], na.rm = TRUE)
    #Plot
    png(filename=paste(graph_output_dir,paste(stock_name, "_",y_col_name, sep="") ,".png",sep=""))
    plot( as.Date(stock_df$DateAsDate),
          stock_df[,c(y_col_name)],
          col="blue",
          type="l",
          xlab= "",
          xlim=c(from_date,to_date),
          ylab= y_col_name,
          ylim=c(min_height,max_height),
          main=paste(stock_name,y_col_name,"\n", from_date, "-", to_date, sep=" "),
```

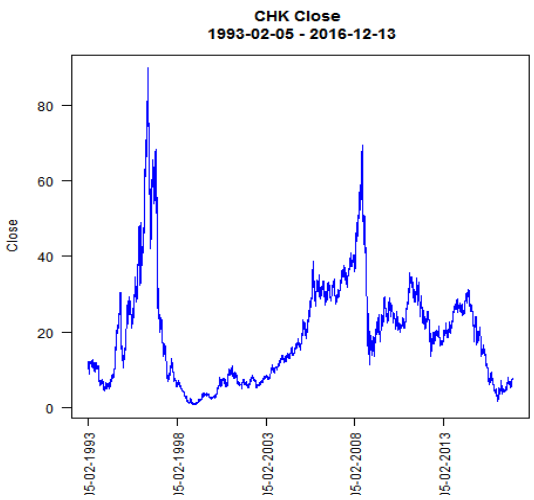
```
        las=2,
        xaxt='n') #this removes the default x-axis label
    axis.Date(1, at=seq(from_date,to_date, by=step_by), format=date_format, las=2)
    dev.off()
  }
}
#Produce the QQ plot
plot_qqplot = function (data, ylab_param) {
  title = ylab_param
  qqnorm(data, main=paste(title, " \n (centered & scaled) - QQplot", sep=""), ylab=paste(ylab_param, paste=""))
  qqline(data, col="gold", lwd=2)
}
```

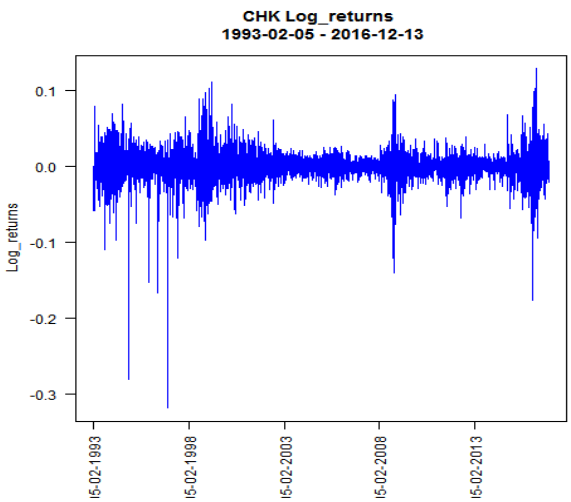
Code - ../ data.explorator.rmd

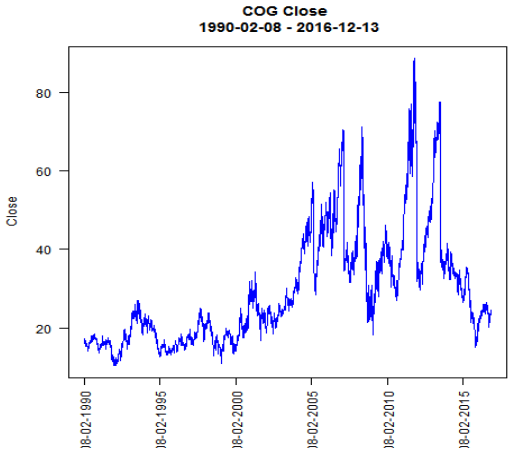
Appendix 11 – Log returns and Price Time Series Graphs

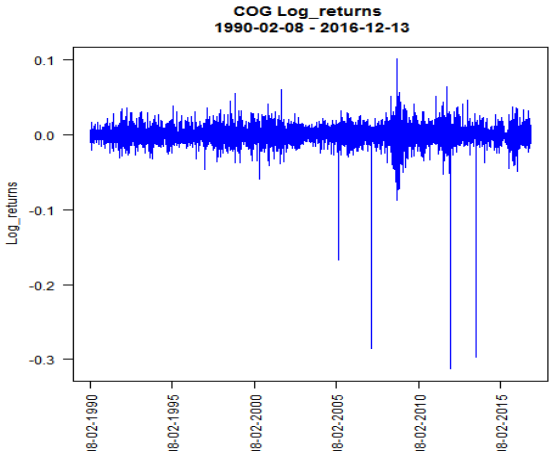
These graphs show the close price and log returns time series graphs for each asset. The output can be found in the folder ..\graphs\XLE\data.exploration.

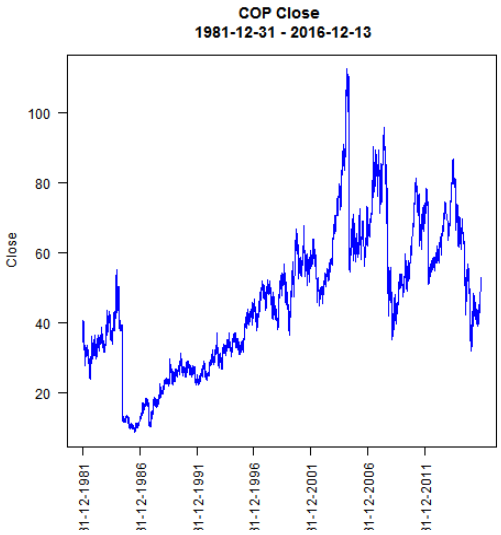


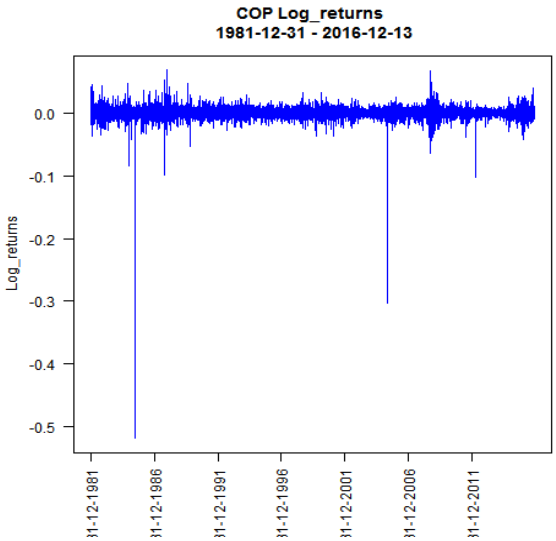

CHK Close
1993-02-05 - 2016-12-13
Close
0
20
40
60
80
05-02-1993
05-02-1998
05-02-2003
05-02-2008
05-02-2013

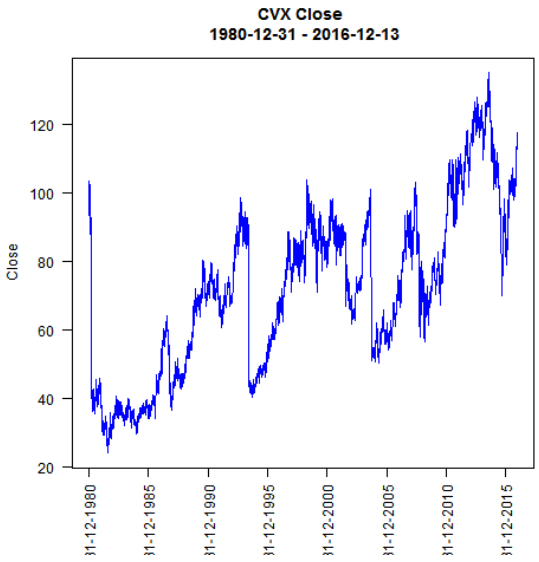

CHK Log_returns
1993-02-05 - 2016-12-13
Log_returns
-0.3
-0.2
-0.1
0.0
0.1
05-02-1993
05-02-1998
05-02-2003
05-02-2008
05-02-2013

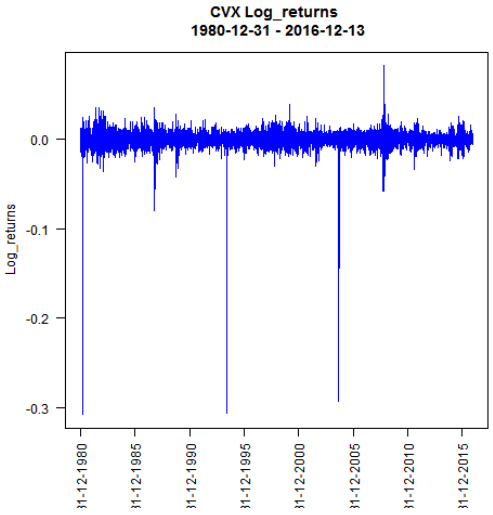

COG Close
1990-02-08 - 2016-12-13
Close
20
40
60
80
08-02-1990
08-02-1995
08-02-2000
08-02-2005
08-02-2010
08-02-2015

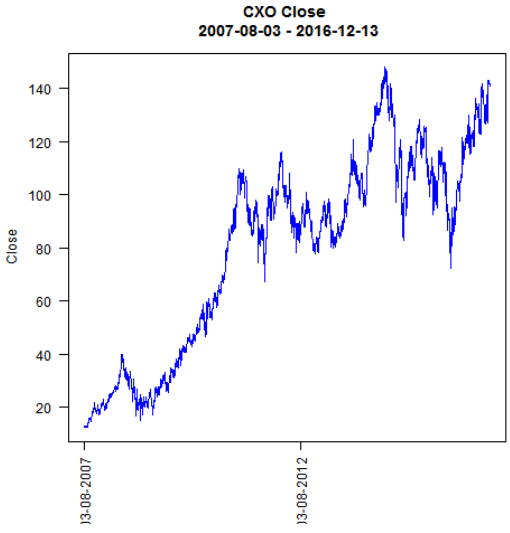

COG Log_returns
1990-02-08 - 2016-12-13
Log_returns
-0.3
-0.2
-0.1
0.0
0.1
08-02-1990
08-02-1995
08-02-2000
08-02-2005
08-02-2010
08-02-2015

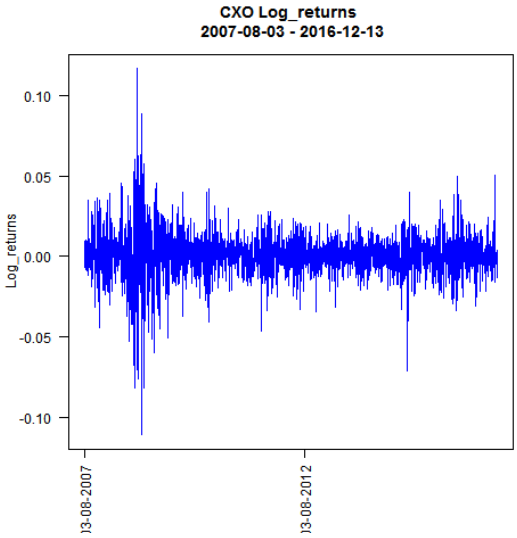

COP Close
1981-12-31 - 2016-12-13
Close
20
40
60
80
100
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

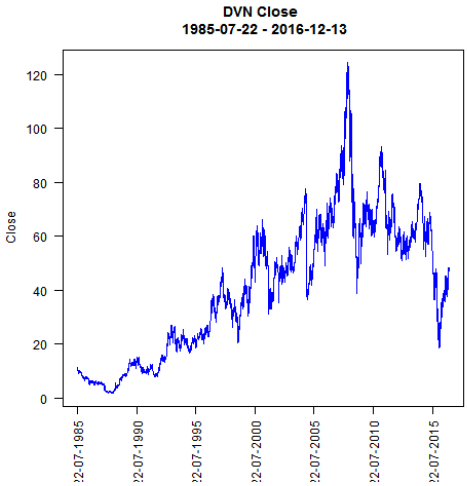

COP Log_returns
1981-12-31 - 2016-12-13
Log_returns
-0.5
-0.4
-0.3
-0.2
-0.1
0.0
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

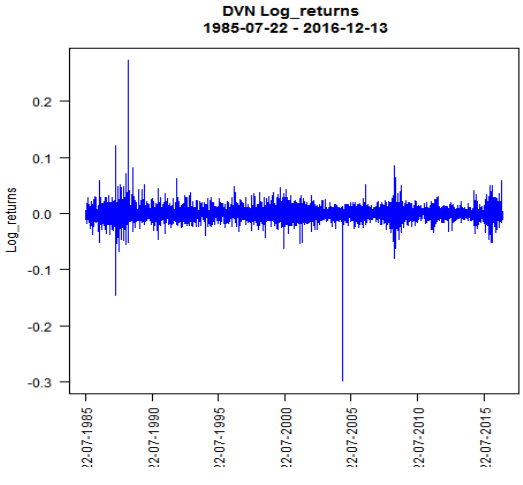

CVX Close
1980-12-31 - 2016-12-13
Close
120
100
80
60
40
20
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

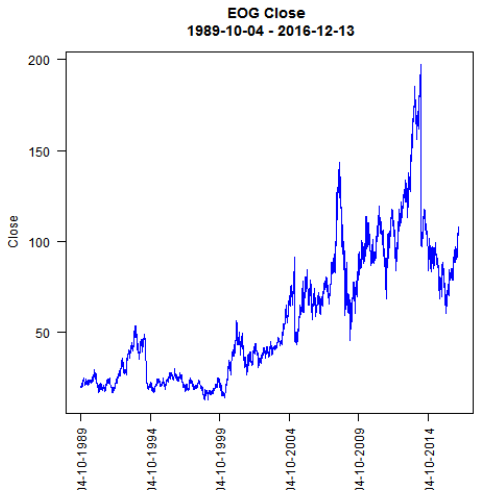

CVX Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

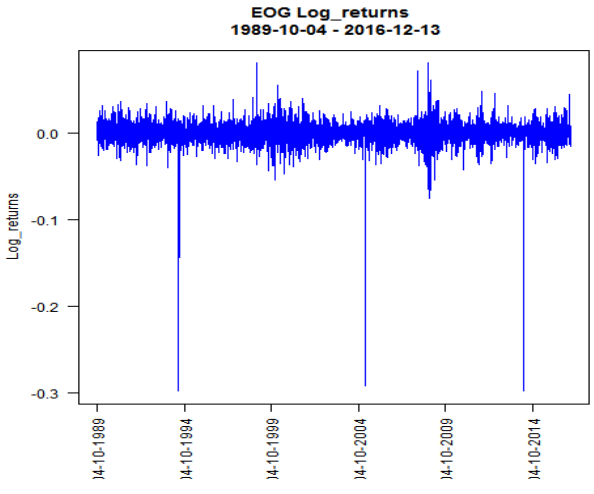

CXO Close
2007-08-03 - 2016-12-13
Close
140
120
100
80
60
40
20
03-08-2007
03-08-2012

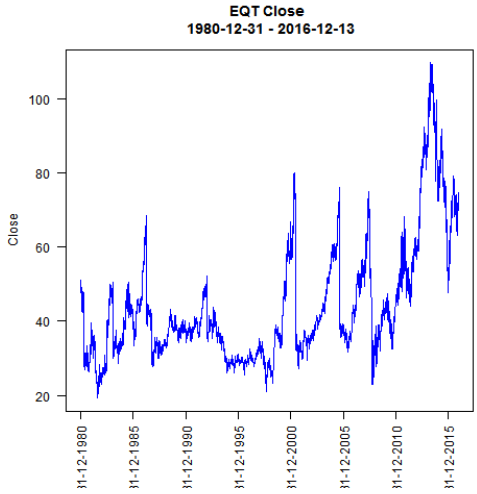

CXO Log_returns
2007-08-03 - 2016-12-13
Log_returns
0.10
0.05
0.00
-0.05
-0.10
03-08-2007
03-08-2012

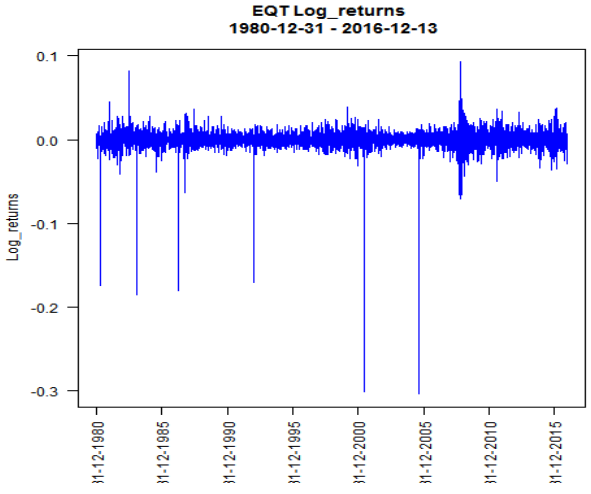

DVN Close
1985-07-22 - 2016-12-13
Close
120
100
80
60
40
20
0
22-07-1985
22-07-1990
22-07-1995
22-07-2000
22-07-2005
22-07-2010
22-07-2015

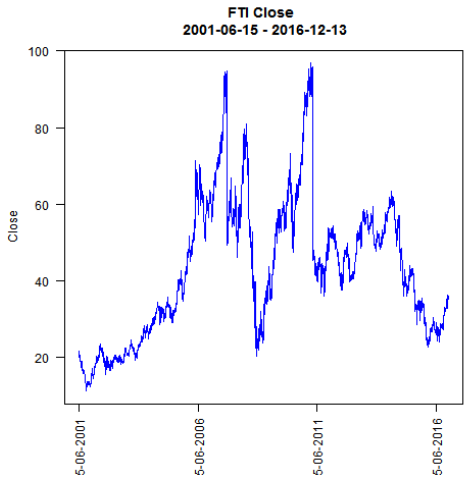

DVN Log_returns
1985-07-22 - 2016-12-13
Log_returns
0.2
0.1
0.0
-0.1
-0.2
-0.3
22-07-1985
22-07-1990
22-07-1995
22-07-2000
22-07-2005
22-07-2010
22-07-2015

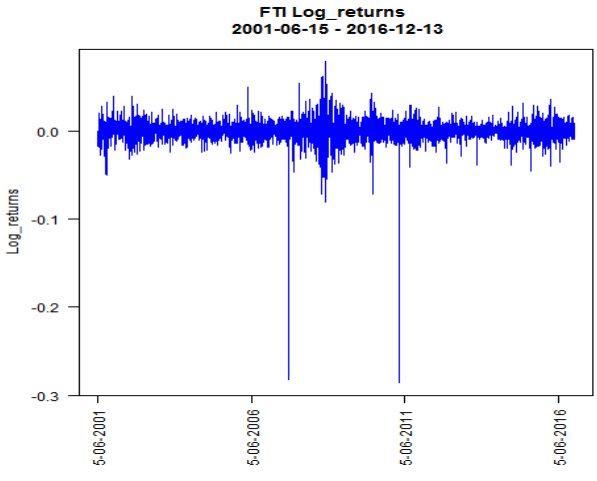

EOG Close
1989-10-04 - 2016-12-13
Close
200
150
100
50
04-10-1989
04-10-1994
04-10-1999
04-10-2004
04-10-2009
04-10-2014

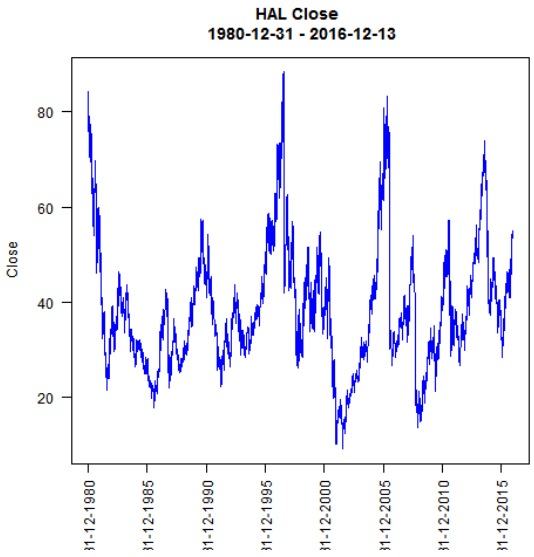

EOG Log_returns
1989-10-04 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
04-10-1989
04-10-1994
04-10-1999
04-10-2004
04-10-2009
04-10-2014

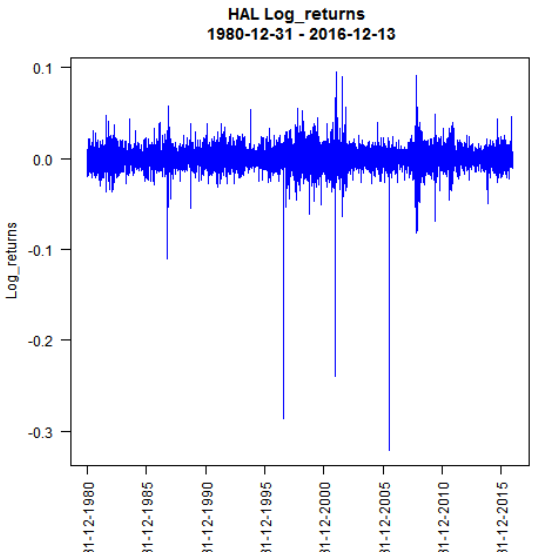

EQT Close
1980-12-31 - 2016-12-13
Close
100
80
60
40
20
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

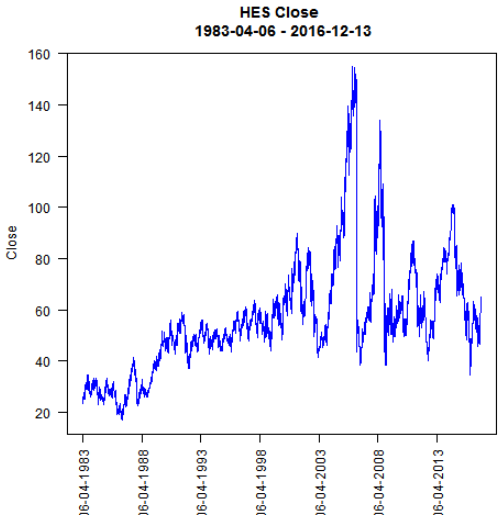

EQT Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

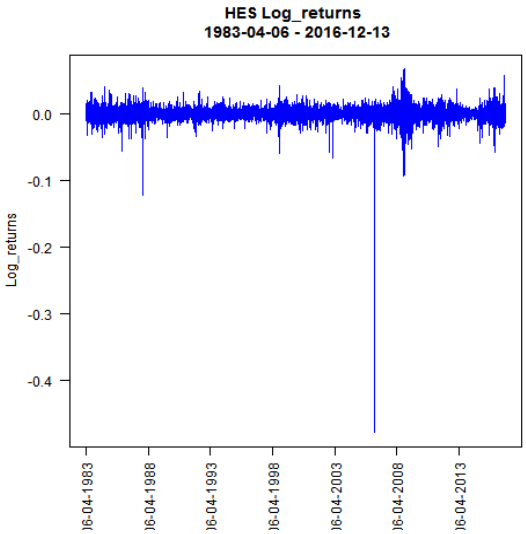

FTI Close
2001-06-15 - 2016-12-13
Close
100
80
60
40
20
15-06-2001
15-06-2006
15-06-2011
15-06-2016

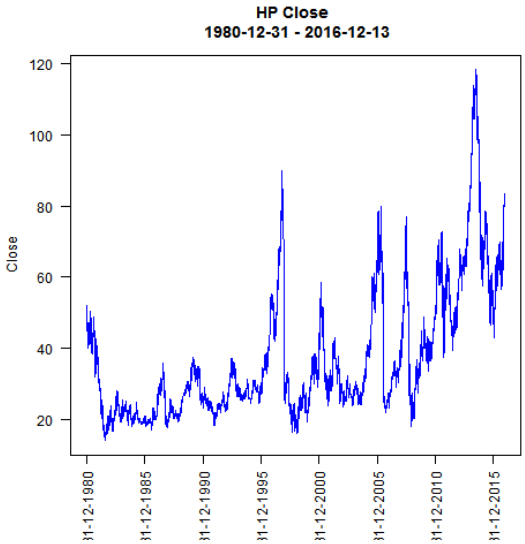

FTI Log_returns
2001-06-15 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
15-06-2001
15-06-2006
15-06-2011
15-06-2016

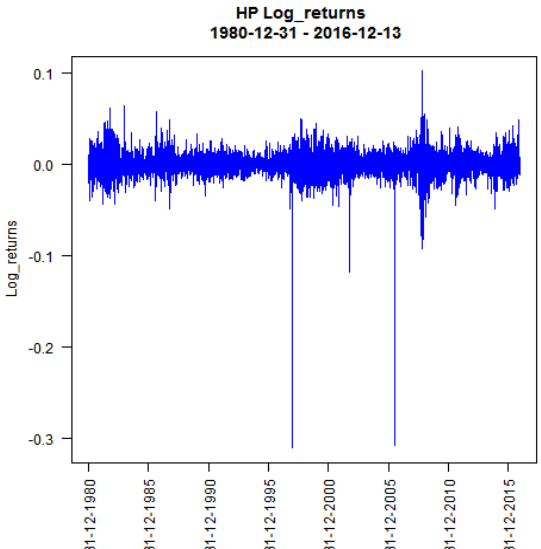

HAL Close
1980-12-31 - 2016-12-13
Close
80
60
40
20
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

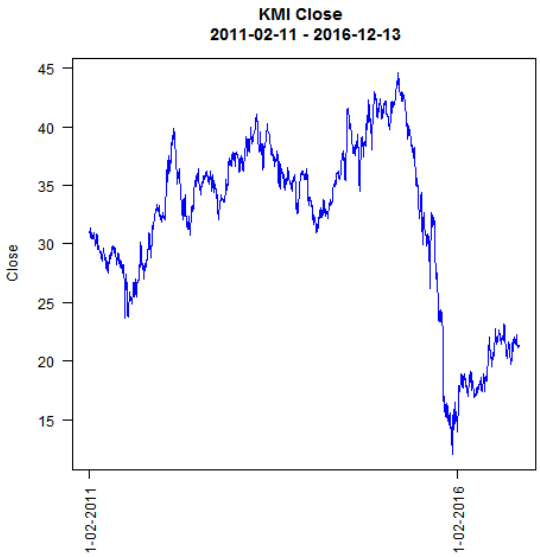

HAL Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

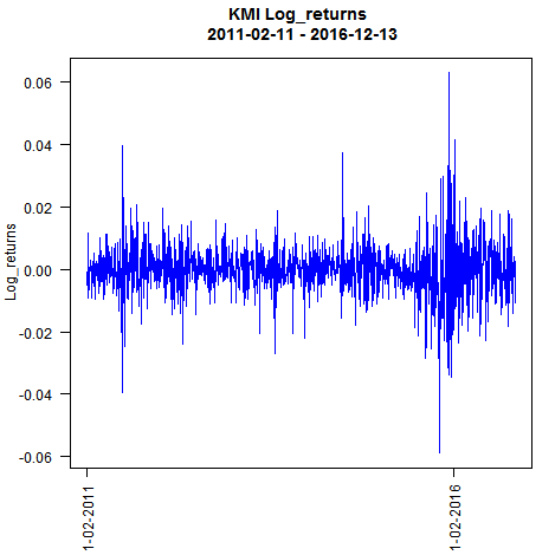

HES Close
1983-04-06 - 2016-12-13
Close
160
140
120
100
80
60
40
20
06-04-1983
06-04-1988
06-04-1993
06-04-1998
06-04-2003
06-04-2008
06-04-2013

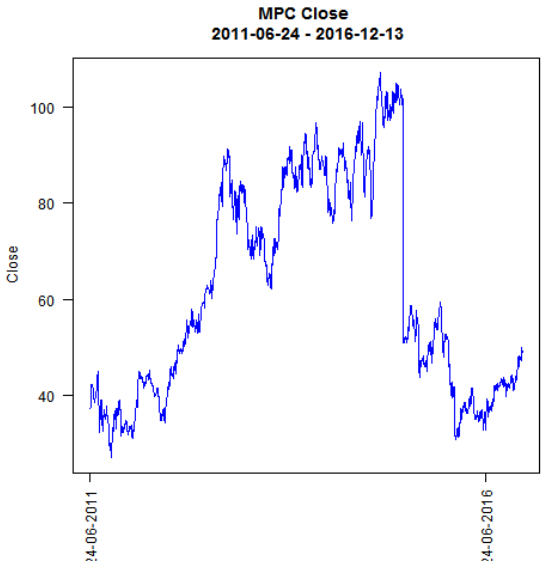

HES Log_returns
1983-04-06 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
-0.4
06-04-1983
06-04-1988
06-04-1993
06-04-1998
06-04-2003
06-04-2008
06-04-2013

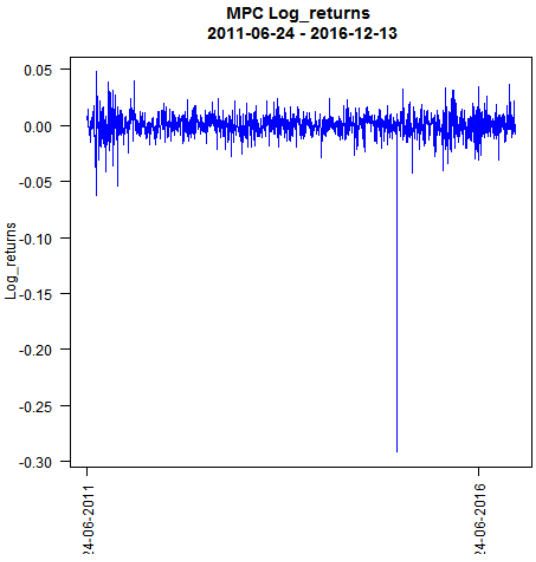

HP Close
1980-12-31 - 2016-12-13
Close
120
100
80
60
40
20
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

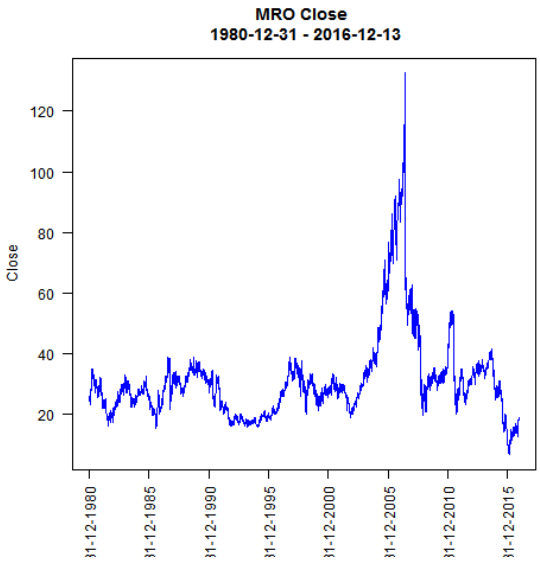

HP Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

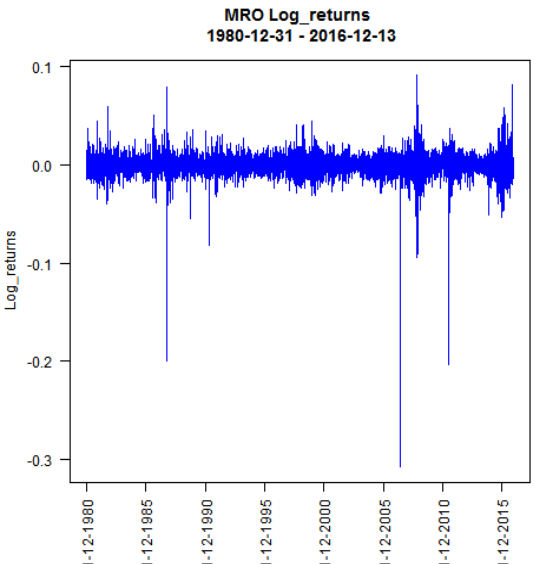

KMI Close
2011-02-11 - 2016-12-13
Close
45
40
35
30
25
20
15
1-02-2011
1-02-2016

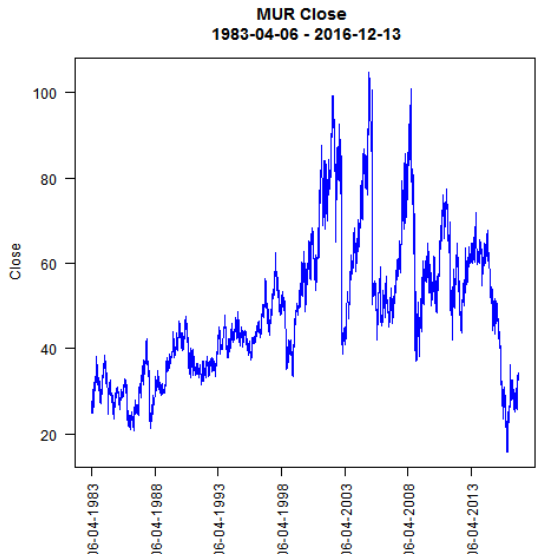

KMI Log_returns
2011-02-11 - 2016-12-13
Log_returns
0.06
0.04
0.02
0.00
-0.02
-0.04
-0.06
1-02-2011
1-02-2016

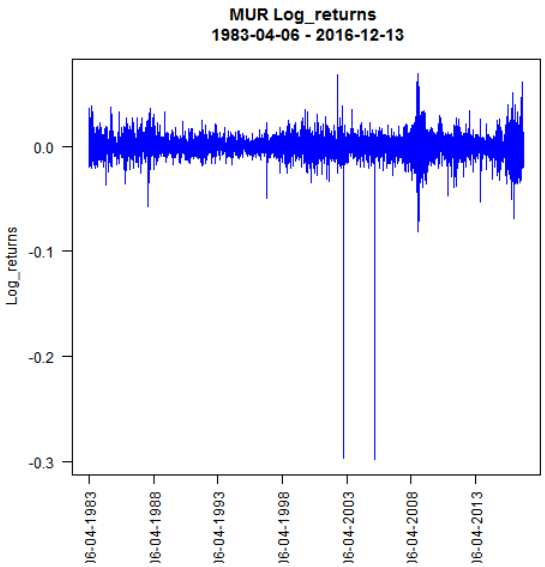

MPC Close
2011-06-24 - 2016-12-13
Close
100
80
60
40
24-06-2011
24-06-2016

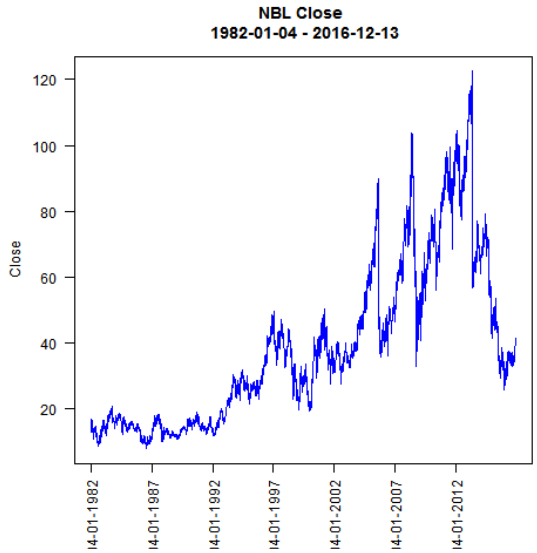

MPC Log_returns
2011-06-24 - 2016-12-13
Log_returns
0.05
0.00
-0.05
-0.10
-0.15
-0.20
-0.25
-0.30
24-06-2011
24-06-2016

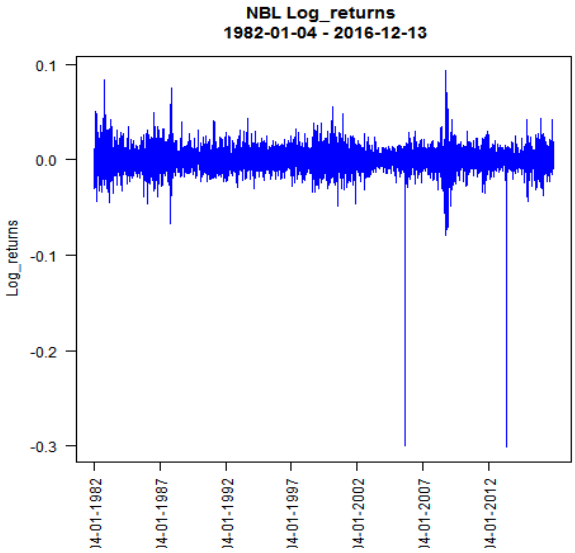

MRO Close
1980-12-31 - 2016-12-13
Close
120
100
80
60
40
20
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

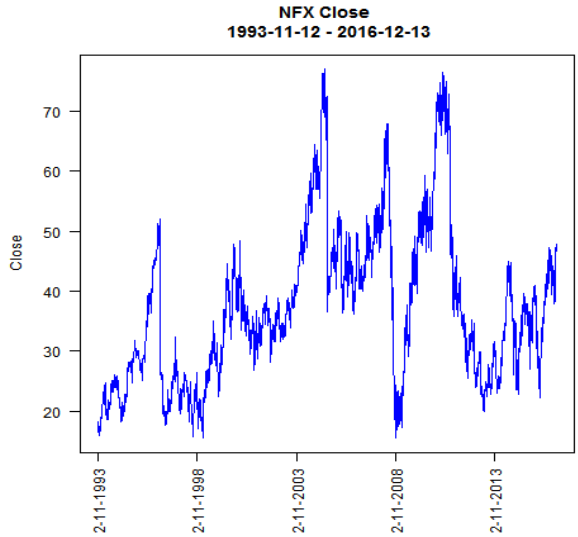

MRO Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

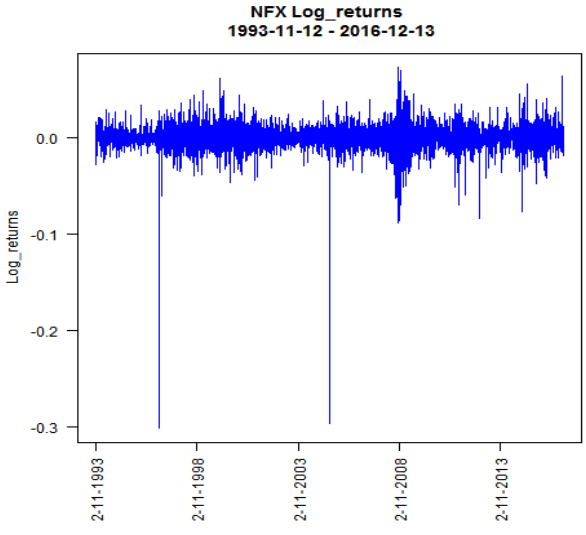

MUR Close
1983-04-06 - 2016-12-13
Close
100
80
60
40
20
06-04-1983
06-04-1988
06-04-1993
06-04-1998
06-04-2003
06-04-2008
06-04-2013

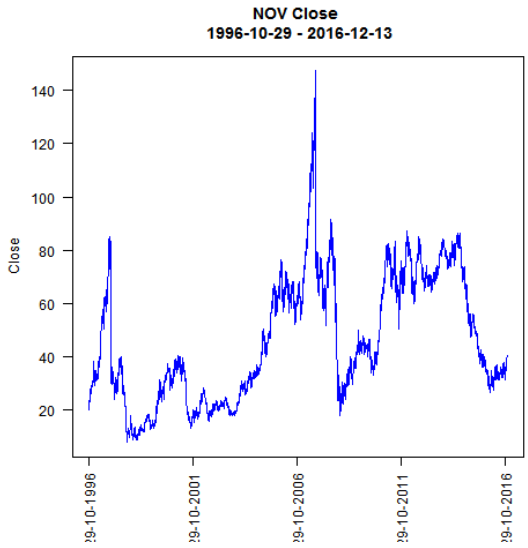

MUR Log_returns
1983-04-06 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
06-04-1983
06-04-1988
06-04-1993
06-04-1998
06-04-2003
06-04-2008
06-04-2013

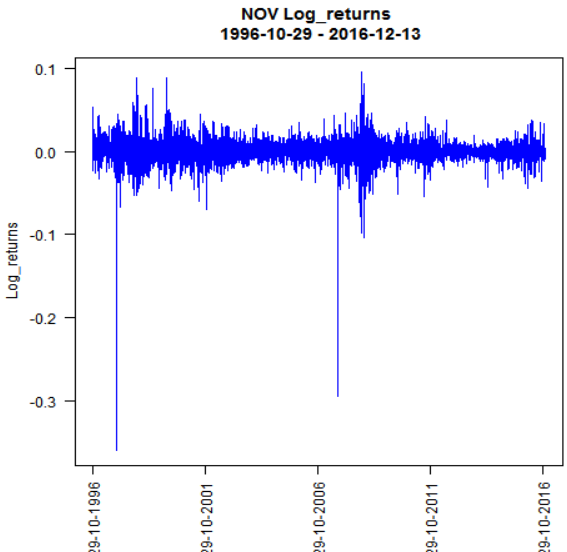

NBL Close
1982-01-04 - 2016-12-13
Close
120
100
80
60
40
20
04-01-1982
04-01-1987
04-01-1992
04-01-1997
04-01-2002
04-01-2007
04-01-2012

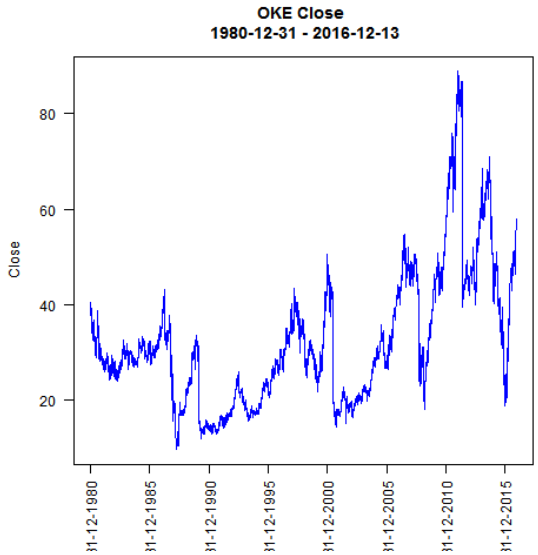

NBL Log_returns
1982-01-04 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
04-01-1982
04-01-1987
04-01-1992
04-01-1997
04-01-2002
04-01-2007
04-01-2012

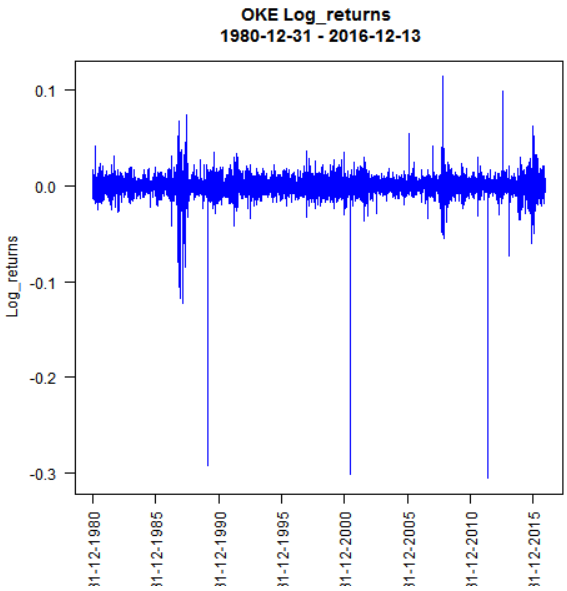

NFX Close
1993-11-12 - 2016-12-13
Close
70
60
50
40
30
20
12-11-1993
12-11-1998
12-11-2003
12-11-2008
12-11-2013

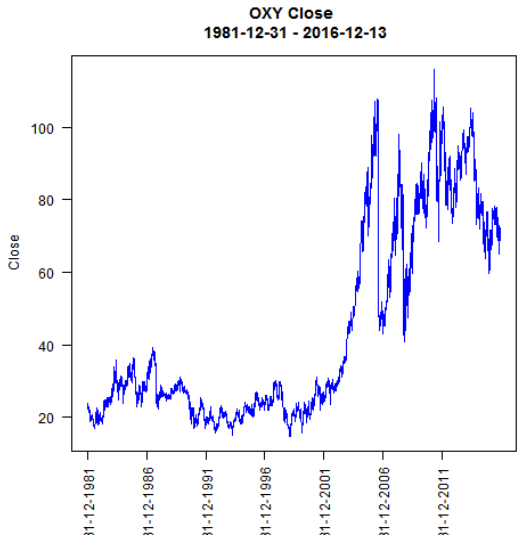

NFX Log_returns
1993-11-12 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
12-11-1993
12-11-1998
12-11-2003
12-11-2008
12-11-2013

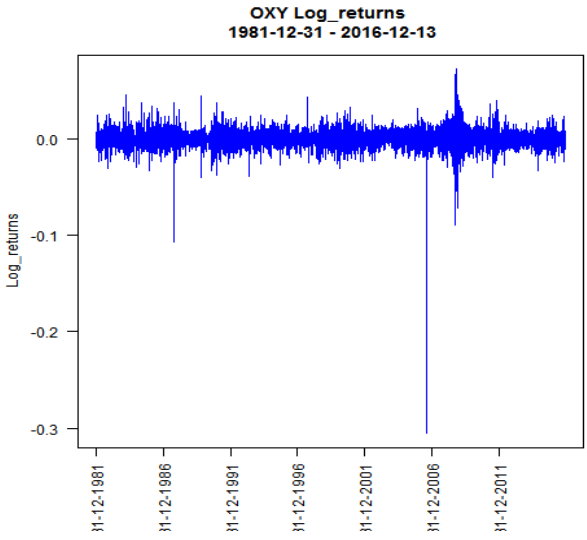

NOV Close
1996-10-29 - 2016-12-13
Close
140
120
100
80
60
40
20
29-10-1996
29-10-2001
29-10-2006
29-10-2011
29-10-2016

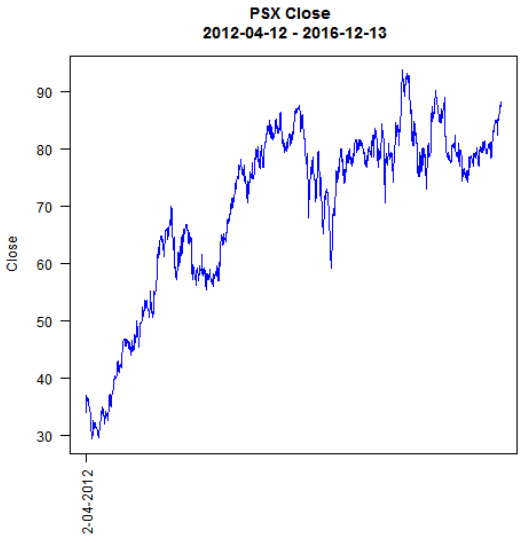

NOV Log_returns
1996-10-29 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
29-10-1996
29-10-2001
29-10-2006
29-10-2011
29-10-2016

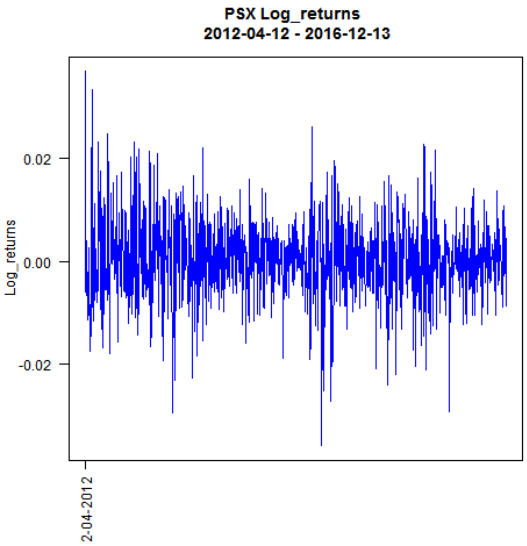

OKE Close
1980-12-31 - 2016-12-13
Close
80
60
40
20
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

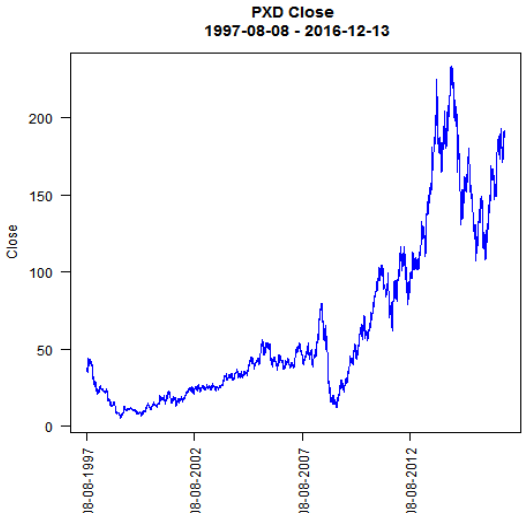

OKE Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

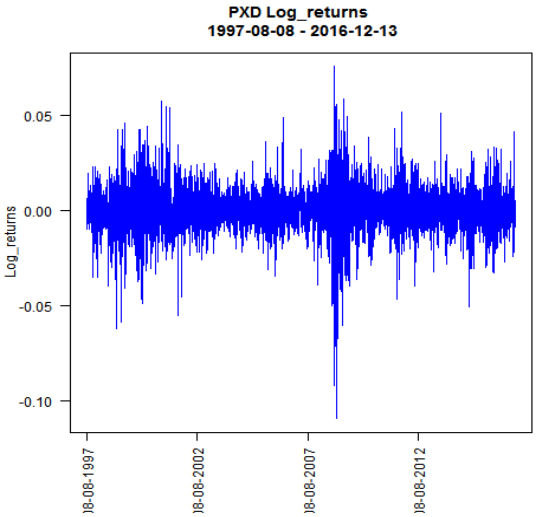

OXY Close
1981-12-31 - 2016-12-13
Close
100
80
60
40
20
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

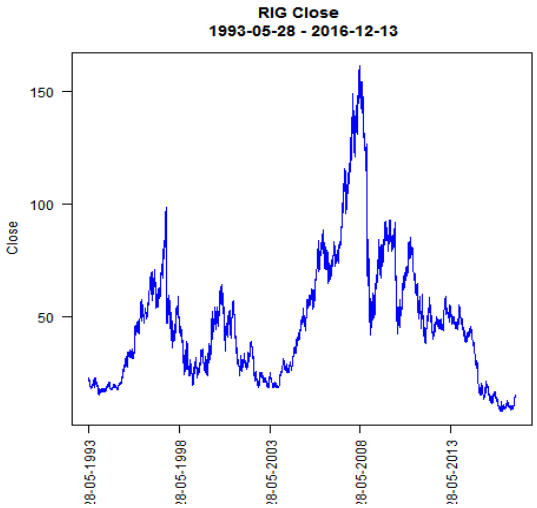

OXY Log_returns
1981-12-31 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

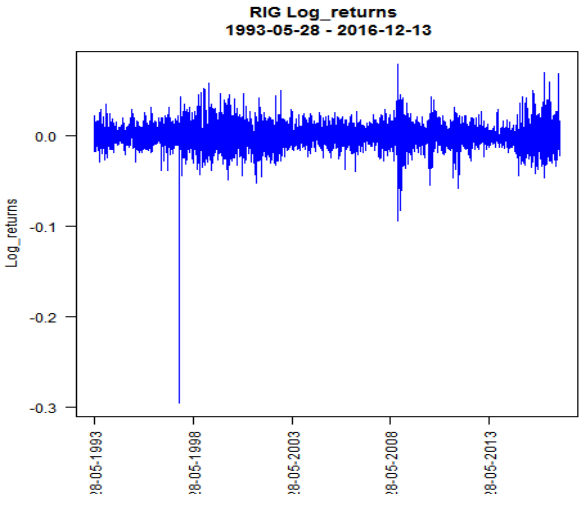

PSX Close
2012-04-12 - 2016-12-13
Close
90
80
70
60
50
40
30
2-04-2012

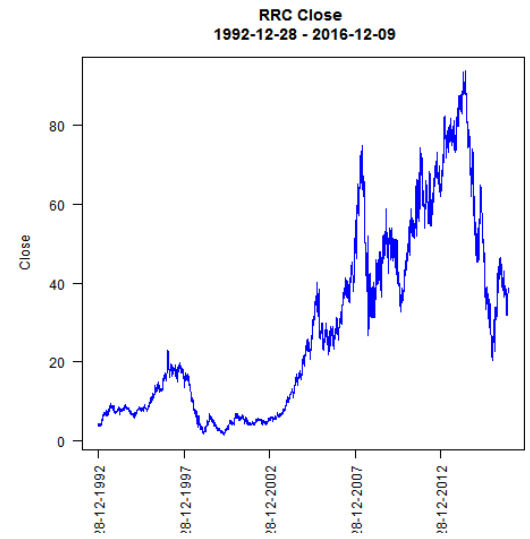

PSX Log_returns
2012-04-12 - 2016-12-13
Log_returns
0.02
0.00
-0.02
2-04-2012

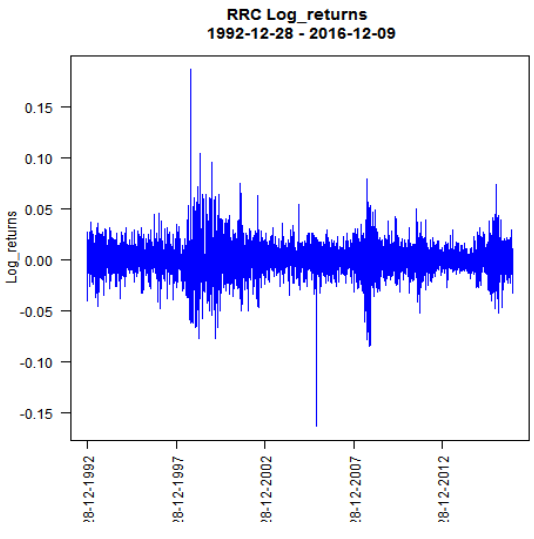

PXD Close
1997-08-08 - 2016-12-13
Close
200
150
100
50
0
8-08-1997
8-08-2002
8-08-2007
8-08-2012

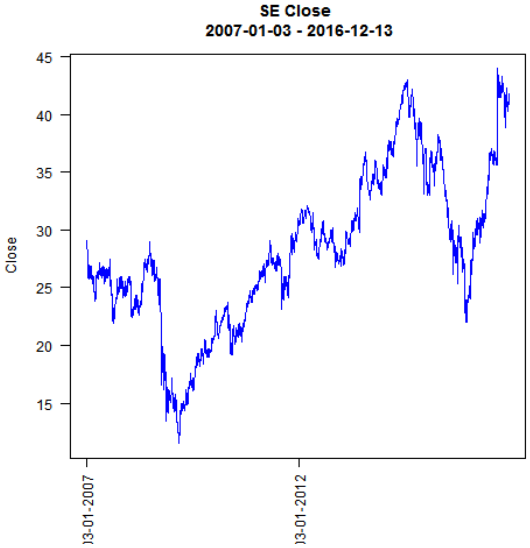

PXD Log_returns
1997-08-08 - 2016-12-13
Log_returns
0.05
0.00
-0.05
-0.10
8-08-1997
8-08-2002
8-08-2007
8-08-2012

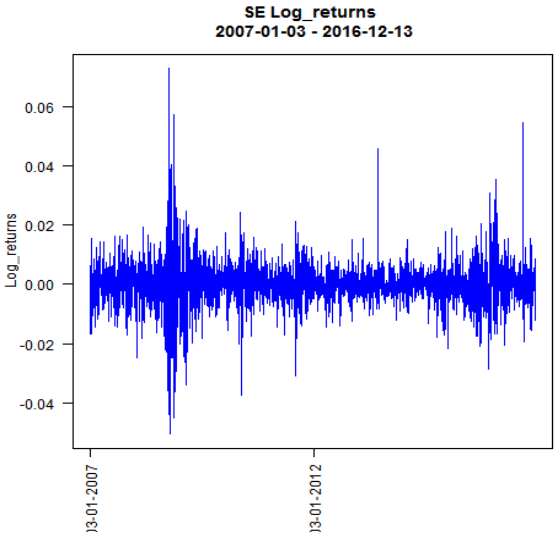

RIG Close
1993-05-28 - 2016-12-13
Close
150
100
50
8-05-1993
8-05-1998
8-05-2003
8-05-2008
8-05-2013

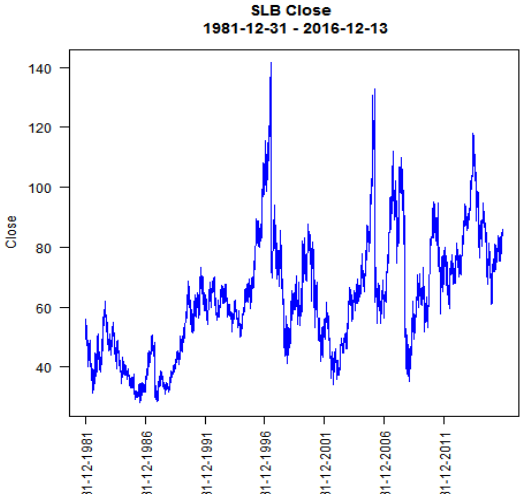

RIG Log_returns
1993-05-28 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
8-05-1993
8-05-1998
8-05-2003
8-05-2008
8-05-2013

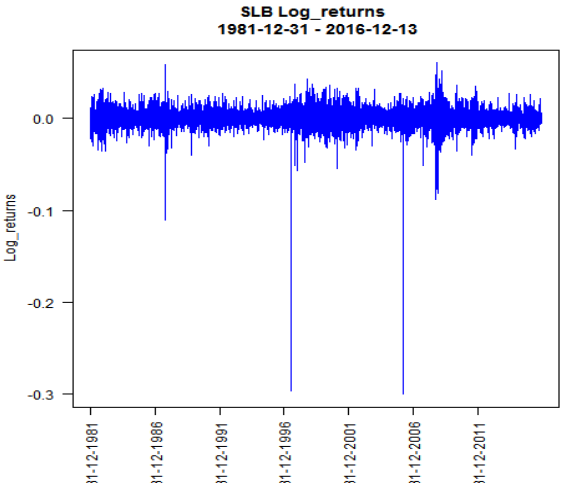

RRC Close
1992-12-28 - 2016-12-09
Close
0
20
40
60
80
28-12-1992
28-12-1997
28-12-2002
28-12-2007
28-12-2012

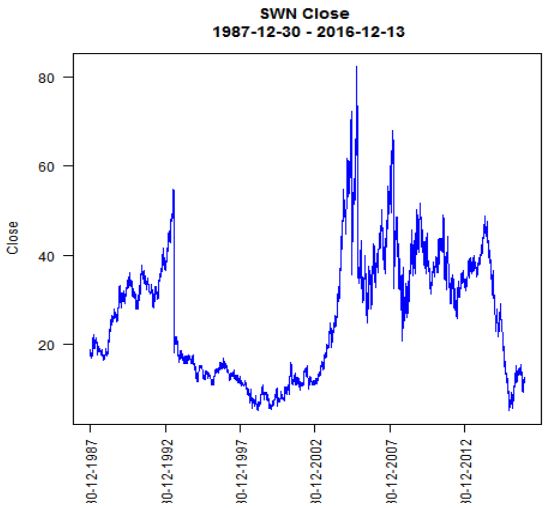

RRC Log_returns
1992-12-28 - 2016-12-09
Log_returns
-0.15
-0.10
-0.05
0.00
0.05
0.10
0.15
28-12-1992
28-12-1997
28-12-2002
28-12-2007
28-12-2012

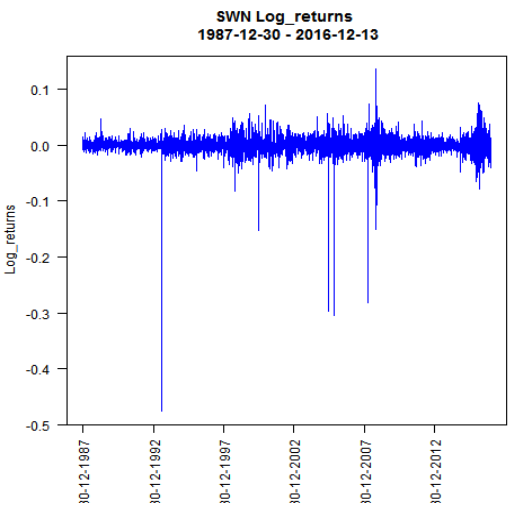

SE Close
2007-01-03 - 2016-12-13
Close
15
20
25
30
35
40
45
03-01-2007
03-01-2012

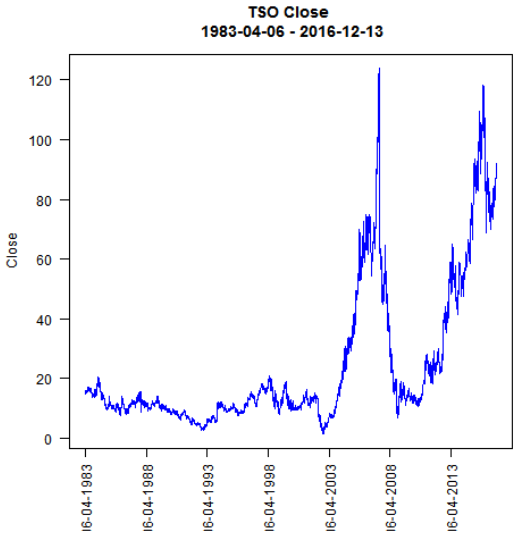

SE Log_returns
2007-01-03 - 2016-12-13
Log_returns
-0.04
-0.02
0.00
0.02
0.04
0.06
03-01-2007
03-01-2012

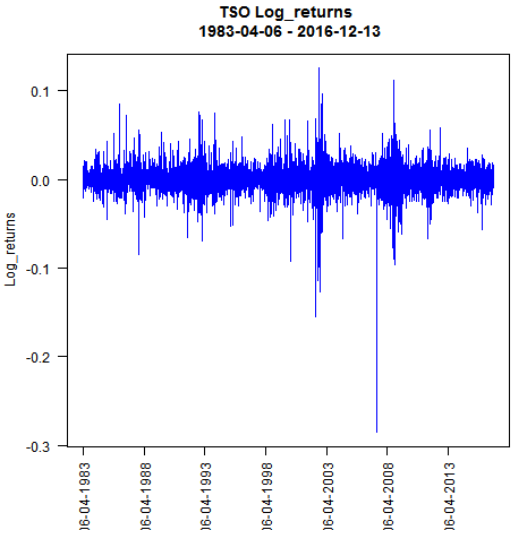

SLB Close
1981-12-31 - 2016-12-13
Close
40
60
80
100
120
140
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

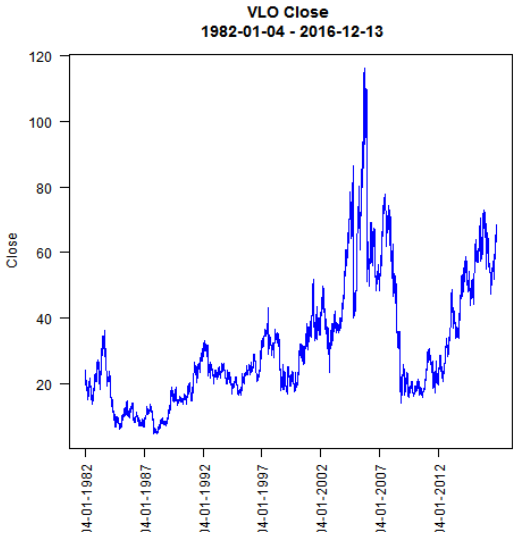

SLB Log_returns
1981-12-31 - 2016-12-13
Log_returns
-0.3
-0.2
-0.1
0.0
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

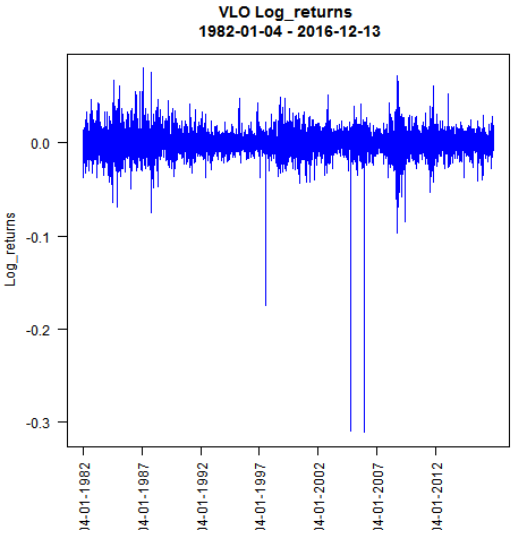

SWN Close
1987-12-30 - 2016-12-13
Close
80
60
40
20

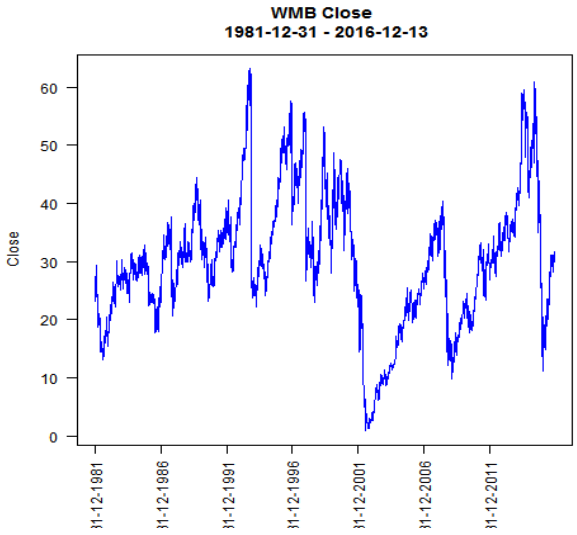

SWN Log_returns
1987-12-30 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3
-0.4
-0.5

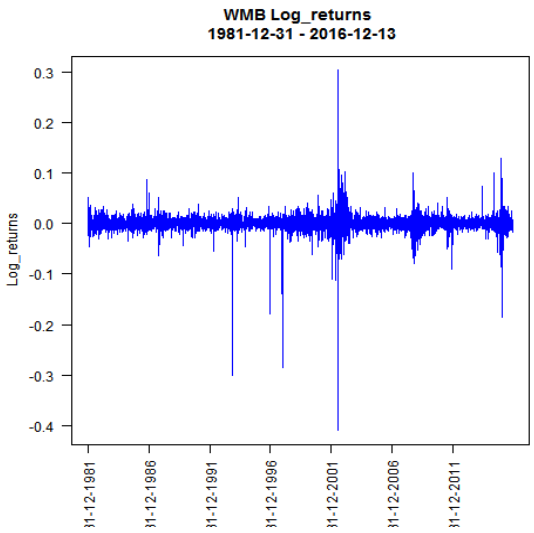

TSO Close
1983-04-06 - 2016-12-13
Close
120
100
80
60
40
20
0

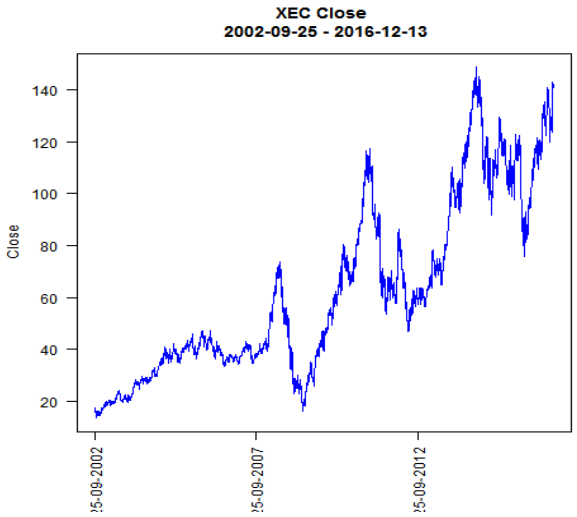

TSO Log_returns
1983-04-06 - 2016-12-13
Log_returns
0.1
0.0
-0.1
-0.2
-0.3

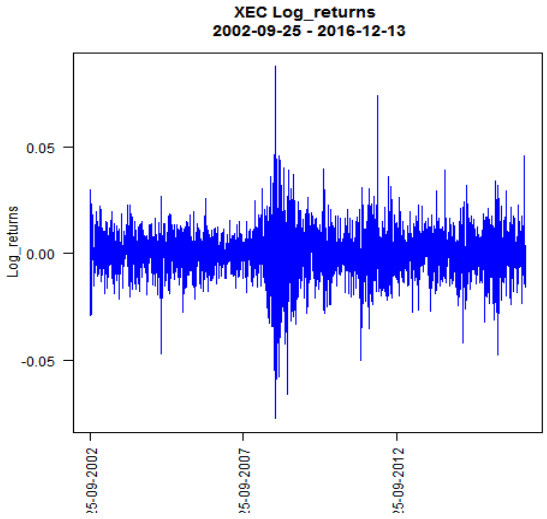

VLO Close
1982-01-04 - 2016-12-13
Close
120
100
80
60
40
20

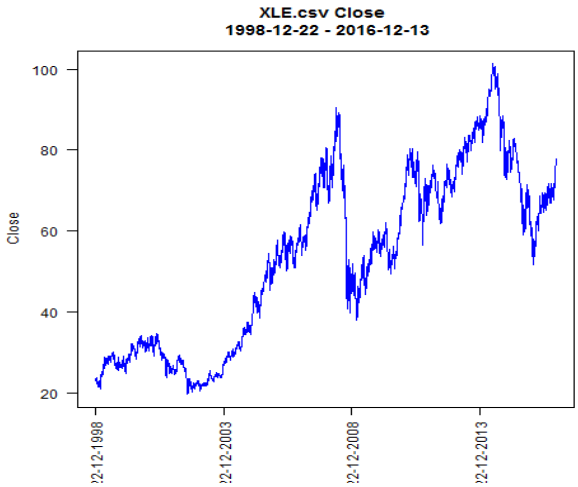

VLO Log_returns
1982-01-04 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3

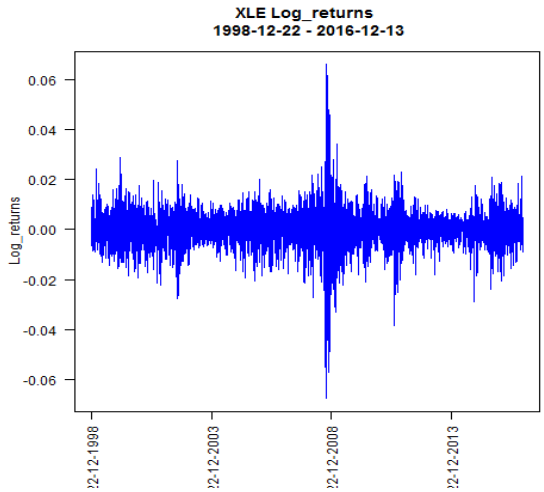

WMB Close
1981-12-31 - 2016-12-13
Close
0
10
20
30
40
50
60
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

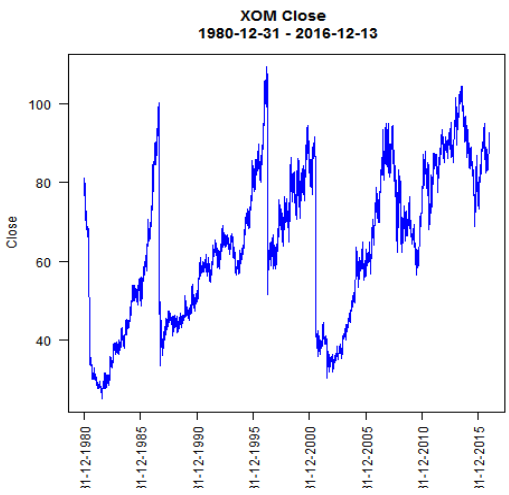

WMB Log_returns
1981-12-31 - 2016-12-13
Log_returns
-0.4
-0.3
-0.2
-0.1
0.0
0.1
0.2
0.3
31-12-1981
31-12-1986
31-12-1991
31-12-1996
31-12-2001
31-12-2006
31-12-2011

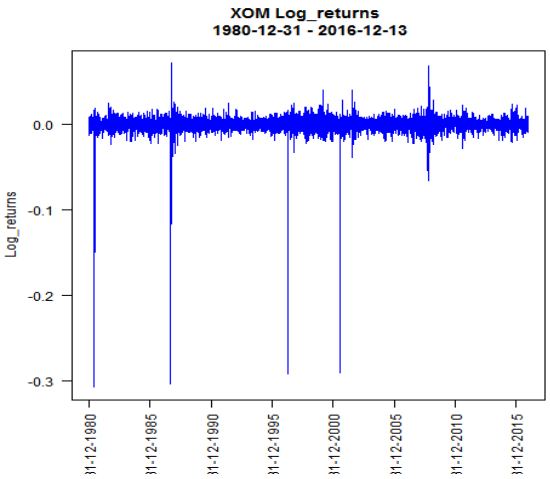

XEC Close
2002-09-25 - 2016-12-13
Close
20
40
60
80
100
120
140
25-09-2002
25-09-2007
25-09-2012


XEC Log_returns
2002-09-25 - 2016-12-13
Log_returns
-0.05
0.00
0.05
25-09-2002
25-09-2007
25-09-2012


XLE.csv Close
1998-12-22 - 2016-12-13
Close
20
40
60
80
100
22-12-1998
22-12-2003
22-12-2008
22-12-2013


XLE Log_returns
1998-12-22 - 2016-12-13
Log_returns
-0.06
-0.04
-0.02
0.00
0.02
0.04
0.06
22-12-1998
22-12-2003
22-12-2008
22-12-2013


XOM Close
1980-12-31 - 2016-12-13
Close
100
80
60
40
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015


XOM Log_returns
1980-12-31 - 2016-12-13
Log_returns
0.0
-0.1
-0.2
-0.3
31-12-1980
31-12-1985
31-12-1990
31-12-1995
31-12-2000
31-12-2005
31-12-2010
31-12-2015

Appendix 12 – Kolmogorov–Smirnov Test Results

The *ks* test was run for each asset and the list of below attributes:

'Close','Close_price_1day_lag','Close_price_2day_lag','Close_price_3day_lag','Close_price_4day_lag','Close_price_5day_lag','Sma20','Sma50','Sma100','Sma200','Ema20','Ema50','Ema100','Ema200','Roc_1day','Roc_2day','Roc_5day','Roc_10day','Roc_20day','Momentum_5day','Momentum_10day','Momentum_20day','MomentumAbs_5day','MomentumAbs_10day','MomentumAbs_20day','Wpr_5day','Wpr_10day','Wpr_20day','Rsi_5day','Rsi_10day','Rsi_14day','Rsi_20day','Cmo_5day','Cmo_10day','Cmo_20day','Atr_tr','Atr_atr','Atr_trueHigh','Atr_trueLow','Smi_smi','Smi_signal','Stoch_fastk','Stoch_fastd','Stoch_slowd','Macd_macd','Macd_signal','Bb_dn','Bb_mavg','Bb_up','Bb_pctB','Volatility','Mfi','Sar','Volume'

The below table shows all the case where the ks did not reject the *Null* hypothesis. All the other cases (i.e. 3654 cases) the *Null* hypothesis was rejected. The raw data can be found in ..\data\XLE\processed\data.exploration\ks_test_results.csv and ks_test_results_analysis.csv.

code	attribute_name	direction	code	attribute_name	direction
APA	Roc_1day	Down	NFX	Momentum_10day	Down
APA	Roc_5day	Down	NFX	Rsi_10day	Down
APA	Roc_10day	Down	NFX	Rsi_20day	Up
APA	Roc_20day	Down	NFX	Rsi_20day	Down
APA	Rsi_20day	Down	NFX	Cmo_10day	Down
APA	Cmo_20day	Down	NFX	Cmo_20day	Up
APC	Momentum_5day	Up	NFX	Cmo_20day	Down
APC	Momentum_5day	Down	NFX	Mfi	Up
APC	Momentum_10day	Up	NOV	Rsi_10day	Down
APC	Rsi_10day	Up	NOV	Rsi_14day	Up
APC	Rsi_10day	Down	NOV	Rsi_14day	Down
APC	Rsi_14day	Up	NOV	Rsi_20day	Up
APC	Rsi_20day	Up	NOV	Rsi_20day	Down
APC	Cmo_10day	Up	NOV	Cmo_10day	Down
APC	Cmo_10day	Down	NOV	Cmo_20day	Up
APC	Cmo_20day	Up	NOV	Cmo_20day	Down
APC	Mfi	Up	NOV	Mfi	Up
APC	Mfi	Down	NOV	Mfi	Down
BHI	Rsi_10day	Up	OKE	Rsi_10day	Up
BHI	Rsi_10day	Down	OKE	Rsi_14day	Up
BHI	Rsi_14day	Up	OKE	Rsi_20day	Up
BHI	Rsi_14day	Down	OKE	Cmo_10day	Up
BHI	Rsi_20day	Up	OKE	Cmo_20day	Up
BHI	Rsi_20day	Down	OXY	Roc_1day	Down
BHI	Cmo_10day	Up	OXY	Roc_2day	Down
BHI	Cmo_10day	Down	OXY	Rsi_10day	Up
BHI	Cmo_20day	Up	OXY	Rsi_14day	Up
BHI	Cmo_20day	Down	OXY	Cmo_10day	Up
CHK	Rsi_10day	Up	PSX	Roc_1day	Up
CHK	Rsi_10day	Down	PSX	Roc_1day	Down

CHK	Rsi_14day	Up	PSX	Roc_2day	Up
CHK	Rsi_14day	Down	PSX	Roc_5day	Up
CHK	Rsi_20day	Up	PSX	Roc_5day	Down
CHK	Rsi_20day	Down	PSX	Roc_20day	Up
CHK	Cmo_10day	Up	PSX	Roc_20day	Down
CHK	Cmo_10day	Down	PSX	Momentum_5day	Up
CHK	Cmo_20day	Up	PSX	Momentum_5day	Down
CHK	Cmo_20day	Down	PSX	Momentum_20day	Up
COG	Rsi_10day	Up	PSX	Momentum_20day	Down
COG	Cmo_10day	Up	PSX	Rsi_5day	Up
COG	Mfi	Up	PSX	Rsi_5day	Down
COG	Mfi	Down	PSX	Rsi_10day	Up
COP	Rsi_10day	Up	PSX	Rsi_10day	Down
COP	Rsi_10day	Down	PSX	Rsi_14day	Up
COP	Rsi_14day	Up	PSX	Rsi_14day	Down
COP	Rsi_14day	Down	PSX	Rsi_20day	Up
COP	Cmo_10day	Up	PSX	Rsi_20day	Down
COP	Cmo_10day	Down	PSX	Cmo_5day	Up
CVX	Momentum_5day	Up	PSX	Cmo_5day	Down
CVX	Rsi_10day	Up	PSX	Cmo_10day	Up
CVX	Rsi_10day	Down	PSX	Cmo_10day	Down
CVX	Rsi_14day	Down	PSX	Cmo_20day	Up
CVX	Cmo_10day	Up	PSX	Cmo_20day	Down
CVX	Cmo_10day	Down	PSX	Smi_smi	Up
CXO	Roc_10day	Up	PSX	Smi_smi	Down
CXO	Roc_10day	Down	PSX	Smi_signal	Up
CXO	Roc_20day	Up	PSX	Smi_signal	Down
CXO	Roc_20day	Down	PSX	Macd_macd	Down
CXO	Rsi_5day	Down	PSX	Macd_signal	Up
CXO	Rsi_10day	Up	PSX	Macd_signal	Down
CXO	Rsi_10day	Down	PSX	Volatility	Up
CXO	Rsi_14day	Up	PSX	Mfi	Up
CXO	Rsi_14day	Down	PXD	Rsi_10day	Up
CXO	Rsi_20day	Down	PXD	Rsi_10day	Down
CXO	Cmo_5day	Down	PXD	Rsi_14day	Up
CXO	Cmo_10day	Up	PXD	Rsi_14day	Down
CXO	Cmo_10day	Down	PXD	Rsi_20day	Up
CXO	Cmo_20day	Down	PXD	Rsi_20day	Down
CXO	Mfi	Up	PXD	Cmo_10day	Up
CXO	Mfi	Down	PXD	Cmo_10day	Down
DVN	Momentum_5day	Up	PXD	Cmo_20day	Up
DVN	Rsi_10day	Up	PXD	Cmo_20day	Down
DVN	Rsi_10day	Down	PXD	Mfi	Up

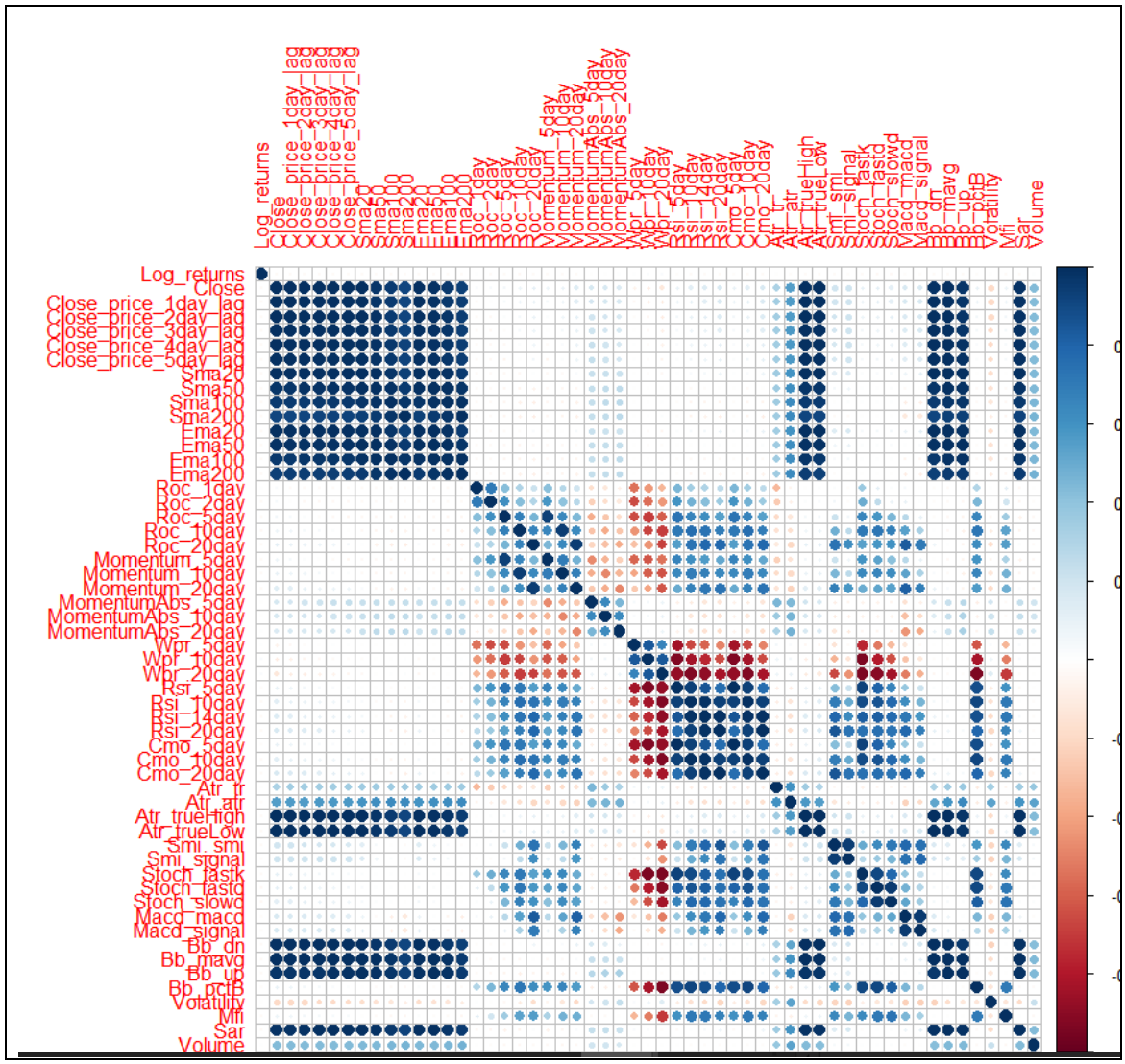
DVN	Rsi_14day	Up	RIG	Roc_20day	Down
DVN	Rsi_14day	Down	RIG	Rsi_5day	Up
DVN	Rsi_20day	Up	RIG	Rsi_10day	Up
DVN	Cmo_10day	Up	RIG	Rsi_10day	Down
DVN	Cmo_10day	Down	RIG	Rsi_14day	Up
DVN	Cmo_20day	Up	RIG	Rsi_14day	Down
DVN	Mfi	Up	RIG	Rsi_20day	Up
DVN	Mfi	Down	RIG	Rsi_20day	Down
EOG	Rsi_10day	Up	RIG	Cmo_5day	Up
EOG	Rsi_10day	Down	RIG	Cmo_10day	Up
EOG	Rsi_14day	Up	RIG	Cmo_10day	Down
EOG	Rsi_14day	Down	RIG	Cmo_20day	Up
EOG	Rsi_20day	Up	RIG	Cmo_20day	Down
EOG	Rsi_20day	Down	RIG	Mfi	Up
EOG	Cmo_10day	Up	RRC	Rsi_10day	Up
EOG	Cmo_10day	Down	RRC	Rsi_10day	Down
EOG	Cmo_20day	Up	RRC	Rsi_14day	Up
EOG	Cmo_20day	Down	RRC	Rsi_14day	Down
EOG	Mfi	Up	RRC	Rsi_20day	Up
EQT	Rsi_10day	Up	RRC	Rsi_20day	Down
EQT	Rsi_14day	Up	RRC	Cmo_10day	Up
EQT	Cmo_10day	Up	RRC	Cmo_10day	Down
FTI	Rsi_5day	Up	RRC	Cmo_20day	Up
FTI	Rsi_10day	Up	RRC	Cmo_20day	Down
FTI	Rsi_10day	Down	RRC	Mfi	Down
FTI	Cmo_5day	Up	SE	Close_price_1day_lag	Down
FTI	Cmo_10day	Up	SE	Momentum_5day	Up
FTI	Cmo_10day	Down	SE	Momentum_5day	Down
FTI	Mfi	Up	SE	Momentum_10day	Up
FTI	Mfi	Down	SE	Momentum_10day	Down
HAL	Rsi_20day	Down	SE	Momentum_20day	Up
HAL	Cmo_20day	Down	SE	Momentum_20day	Down
HP	Roc_10day	Down	SE	Rsi_10day	Up
HP	Rsi_10day	Up	SE	Rsi_10day	Down
HP	Rsi_10day	Down	SE	Rsi_14day	Up
HP	Rsi_14day	Up	SE	Rsi_14day	Down
HP	Rsi_14day	Down	SE	Rsi_20day	Down
HP	Rsi_20day	Up	SE	Cmo_10day	Up
HP	Rsi_20day	Down	SE	Cmo_10day	Down
HP	Cmo_10day	Up	SE	Cmo_20day	Down
HP	Cmo_10day	Down	SE	Mfi	Up
HP	Cmo_20day	Up	SE	Mfi	Down
HP	Cmo_20day	Down	SE	Sar	Up

KMI	Momentum_5day	Up	SLB	Momentum_5day	Down
KMI	Momentum_5day	Down	SLB	Momentum_10day	Down
KMI	Momentum_10day	Up	SLB	Momentum_20day	Down
KMI	Momentum_10day	Down	SLB	Rsi_14day	Up
KMI	Momentum_20day	Up	SLB	Rsi_20day	Up
KMI	Momentum_20day	Down	SLB	Cmo_20day	Up
KMI	Rsi_5day	Up	SLB	Mfi	Up
KMI	Rsi_5day	Down	SLB	Mfi	Down
KMI	Rsi_10day	Up	SWN	Rsi_10day	Up
KMI	Rsi_10day	Down	SWN	Rsi_10day	Down
KMI	Rsi_14day	Up	SWN	Rsi_14day	Up
KMI	Rsi_14day	Down	SWN	Rsi_14day	Down
KMI	Rsi_20day	Up	SWN	Rsi_20day	Up
KMI	Rsi_20day	Down	SWN	Rsi_20day	Down
KMI	Cmo_5day	Up	SWN	Cmo_10day	Up
KMI	Cmo_5day	Down	SWN	Cmo_10day	Down
KMI	Cmo_10day	Up	SWN	Cmo_20day	Up
KMI	Cmo_10day	Down	SWN	Cmo_20day	Down
KMI	Cmo_20day	Up	TSO	Rsi_10day	Up
KMI	Cmo_20day	Down	TSO	Rsi_14day	Up
KMI	Mfi	Up	TSO	Rsi_20day	Up
KMI	Mfi	Down	TSO	Cmo_10day	Up
MPC	Roc_1day	Up	TSO	Cmo_20day	Up
MPC	Rsi_5day	Up	TSO	Mfi	Up
MPC	Rsi_5day	Down	VLO	Rsi_14day	Up
MPC	Rsi_10day	Up	VLO	Rsi_14day	Down
MPC	Rsi_10day	Down	VLO	Rsi_20day	Up
MPC	Rsi_14day	Up	VLO	Rsi_20day	Down
MPC	Rsi_14day	Down	VLO	Cmo_20day	Up
MPC	Rsi_20day	Up	VLO	Cmo_20day	Down
MPC	Rsi_20day	Down	XEC	Roc_10day	Down
MPC	Cmo_5day	Up	XEC	Wpr_10day	Down
MPC	Cmo_5day	Down	XEC	Wpr_20day	Down
MPC	Cmo_10day	Up	XEC	Rsi_14day	Down
MPC	Cmo_10day	Down	XEC	Rsi_20day	Down
MPC	Cmo_20day	Up	XEC	Cmo_20day	Down
MPC	Cmo_20day	Down	XEC	Stoch_fastk	Down
MPC	Smi_signal	Down	XEC	Stoch_fastd	Down
MPC	Mfi	Up	XEC	Stoch_slowd	Down
MPC	Mfi	Down	XEC	Mfi	Up
MRO	Rsi_10day	Up	XEC	Mfi	Down
MRO	Rsi_14day	Up	XLE	Roc_20day	Down
MRO	Rsi_14day	Down	XLE	Rsi_10day	Up

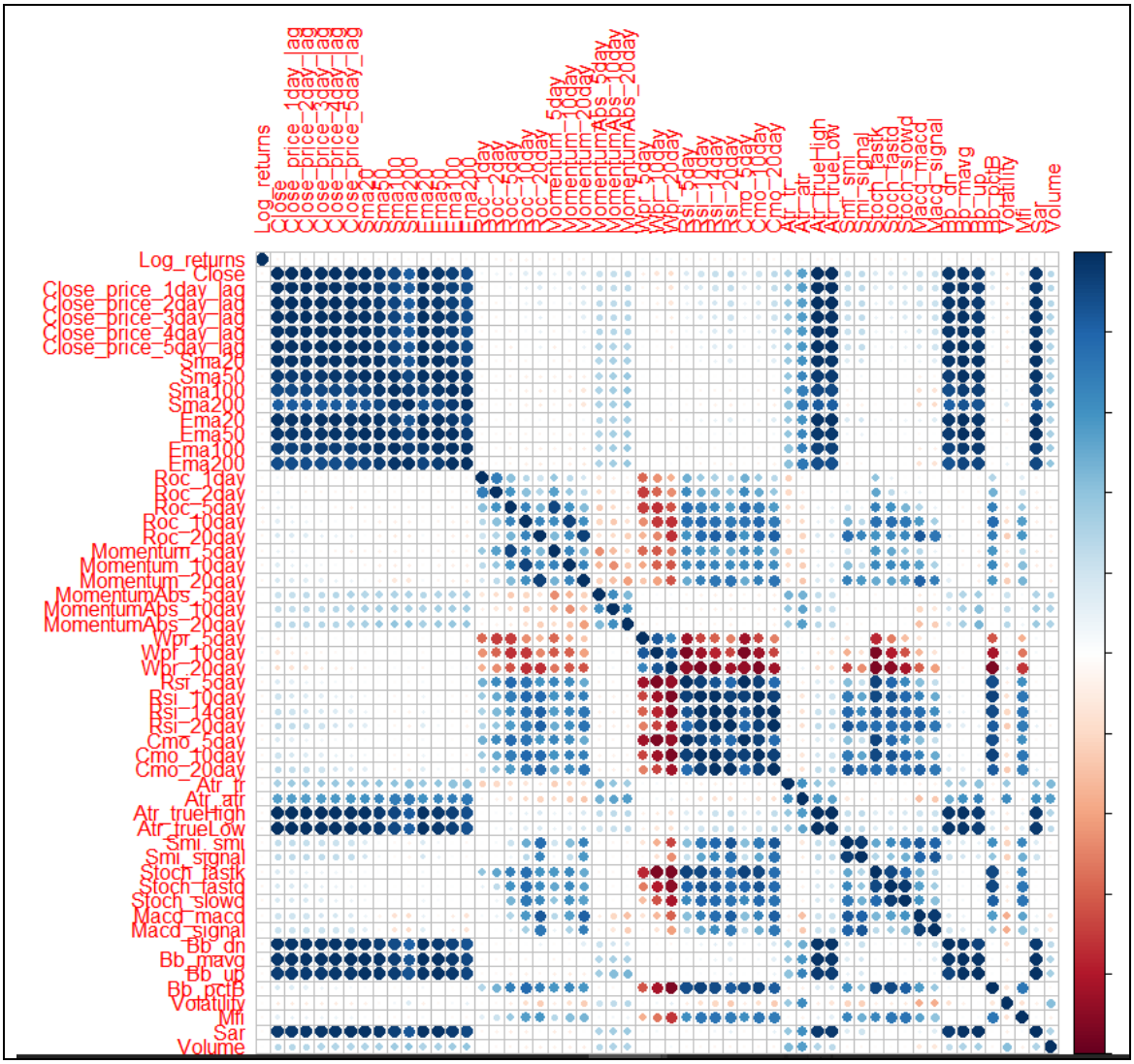
MRO	Rsi_20day	Up	XLE	Rsi_14day	Up
MRO	Rsi_20day	Down	XLE	Rsi_14day	Down
MRO	Cmo_10day	Up	XLE	Rsi_20day	Up
MRO	Cmo_20day	Up	XLE	Rsi_20day	Down
MRO	Cmo_20day	Down	XLE	Cmo_10day	Up
MUR	Rsi_20day	Up	XLE	Cmo_20day	Up
MUR	Cmo_20day	Up	XLE	Cmo_20day	Down
NBL	Roc_20day	Down	XLE	Mfi	Up
NBL	Rsi_10day	Up	XLE	Mfi	Down
NBL	Cmo_10day	Up	XOM	Rsi_10day	Up
NBL	Mfi	Up	XOM	Cmo_10day	Up
NBL	Mfi	Down			

Appendix 13 - Correlation Graphs

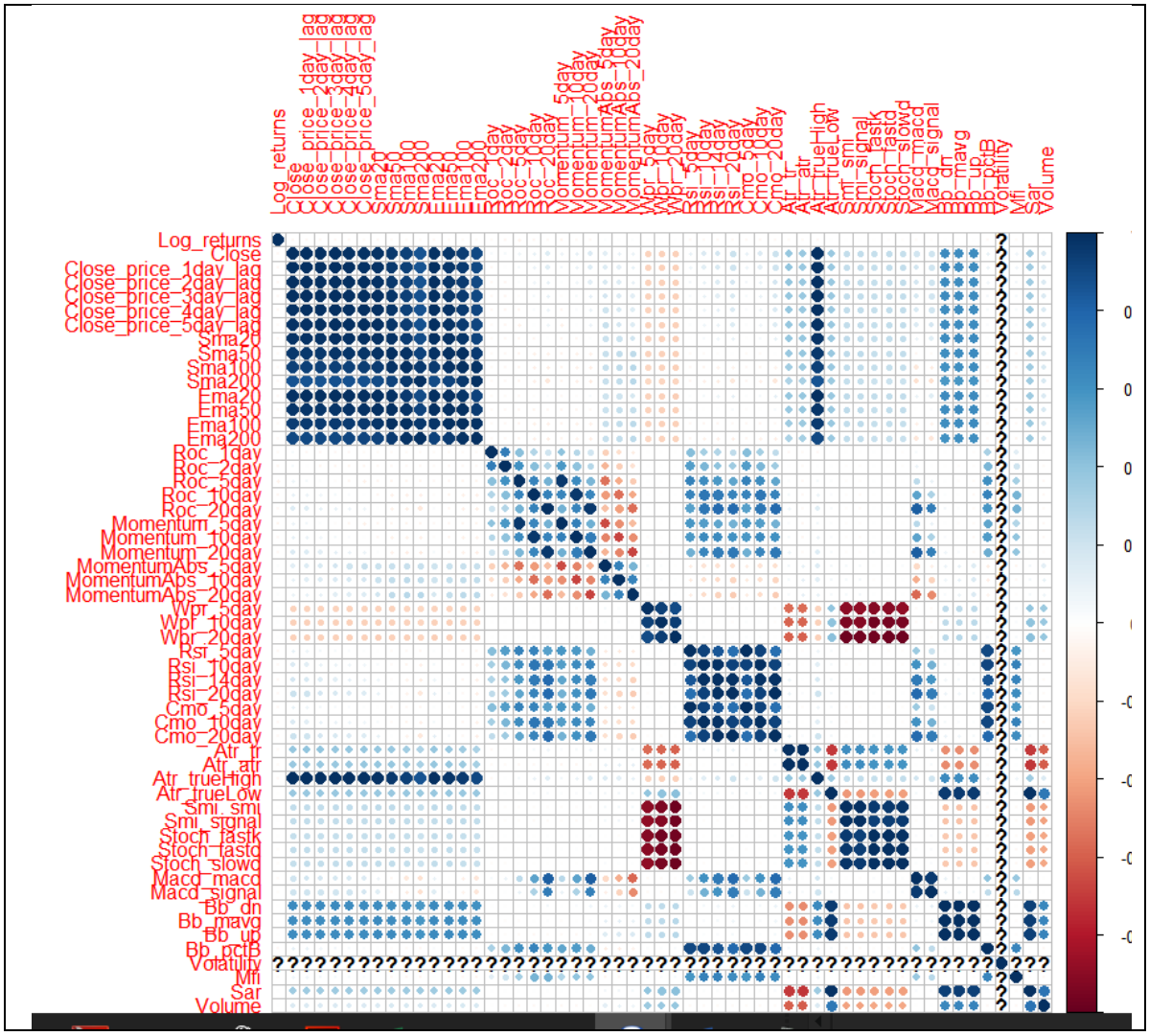
The three below graphs show the degree of correlation within the explanatory variables and with the response variable (the log_returns). The have been chosen as they represent the three largest weights in the index.



CVX Correlation Graph



SLB Correlation Graph



XOM Correlation Graph

Appendix 14 – Class Imbalance Details

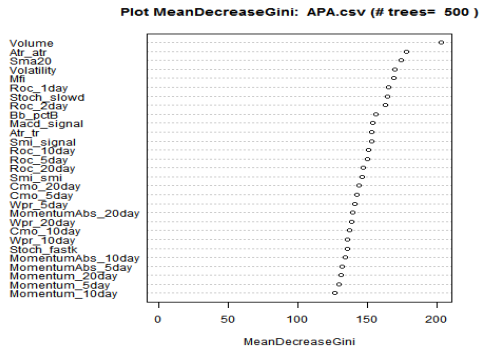
This table shows the frequency of appearance of zero, positive and negative returns for each underlying in the XLE index.

Underlying Name /Frequency	returns = 0	returns > 0	returns < 0
APA	9.55%	45.32%	45.12%
APC	4.02%	47.86%	48.11%
BHI	5.34%	47.14%	47.51%
CHK	5.99%	47.54%	46.46%
COG	7.14%	46.51%	46.34%
COP	6.46%	47.45%	46.07%
CVX	5.72%	48.25%	46.02%
CXO	0.34%	51.99%	47.63%
DVN	10.80%	44.75%	44.44%
EOG	4.16%	49.00%	46.83%
EQT	8.12%	46.66%	45.21%
FTI	0.69%	51.03%	48.26%
HAL	4.48%	47.78%	47.74%
HES	4.39%	48.19%	47.41%
HP	7.54%	46.48%	45.97%
KMI	1.63%	47.65%	50.65%
MPC	0.29%	53.37%	46.27%
MRO	8.61%	45.33%	46.05%
MUR	6.17%	46.87%	46.95%
NBL	7.61%	46.02%	46.36%
NFX	3.85%	49.11%	47.02%
NOV	1.38%	50.17%	48.43%
OKE	9.20%	46.62%	44.17%
OXY	7.93%	46.15%	45.91%
PSX	0.76%	51.87%	47.28%
PXD	1.87%	49.33%	48.78%
RIG	2.71%	48.10%	49.17%
RRC	7.14%	46.52%	46.32%
SE	1.52%	50.58%	47.87%
SLB	3.39%	48.12%	48.47%
SWN	8.66%	45.23%	46.10%
TSO	12.96%	42.79%	44.25%
VLO	8.34%	45.92%	45.74%
WMB	6.26%	47.61%	46.12%
XEC	0.70%	52.15%	47.12%
XOM	5.53%	48.41%	46.05%

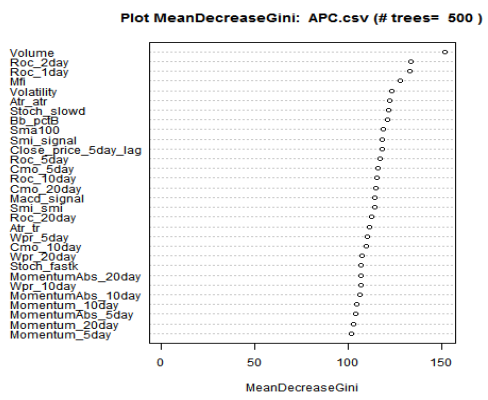
Appendix 15 – Feature Selections Results

These tables represent the results of the two types of feature selection methodologies, for each of the assets. The left graphs show the Wrapper method (i.e. Gini index). The right tables present the Filter method (i.e permutated Relief). The original data can respectively be found under following folders:

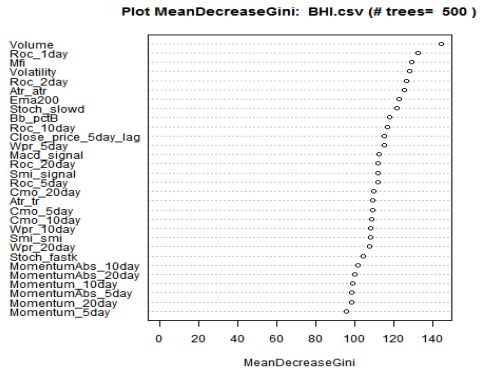
- … \graphs\XLE\feature.selection\mean.decrease.gini\100, and
- …\data\XLE\processed\feature.selection\permuted_relief\100



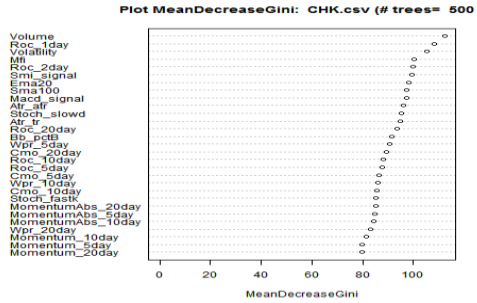
	perm.standardized
Roc_2day	2.950159033
Wpr_5day	1.819855382
Mfi	1.603640416
Stoch_slowd	1.189247204
Sma20	1.122471284
Atr_tr	0.93957407
Stoch_fastk	0.805239559
Macd_signal	0.706551862
Smi_smi	0.67233736
MomentumAbs_20day	0.635004268
Smi_signal	0.634436322
Cmo_20day	0.46521378
Wpr_10day	0.453875082
Wpr_20day	0.228918264
Roc_1day	0.220432195
Volume	0.2035086
MomentumAbs_10day	0.048499225
Momentum_10day	-0.052757481
Cmo_10day	-0.162132208
Volatility	-0.372411154
Momentum_20day	-0.387380631
Momentum_5day	-0.431177495
Atr_atr	-0.509773616
Roc_20day	-0.611969111
Cmo_5day	-0.747572751
MomentumAbs_5day	-0.754592272
Roc_10day	-0.782822676
Bb_pctB	-0.833176746
Roc_5day	-1.443345038



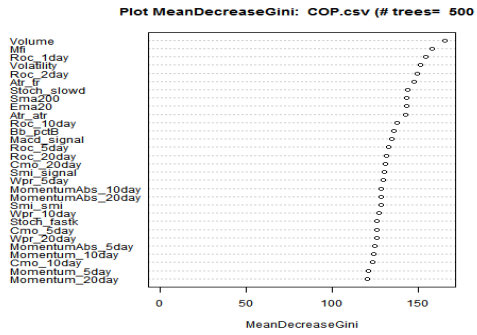
	perm.standardized
Bb_pctB	2.181147196
Cmo_5day	1.072010262
Atr_tr	1.061563028
MomentumAbs_5day	1.01039473
Mfi	0.972461267
Momentum_20day	0.874122706
Roc_20day	0.543258818
MomentumAbs_20day	0.420046586
Roc_2day	0.282110174
Roc_1day	0.151117971
Cmo_20day	0.141275173
Volume	0.140754511
Sma100	-0.035514747
Momentum_5day	-0.109368578
Smi_smi	-0.17158365
Wpr_10day	-0.210038074
Smi_signal	-0.269214283
Atr_atr	-0.287909493
Cmo_10day	-0.368169645
Macd_signal	-0.371737514
Volatility	-0.397512876
Roc_5day	-0.411112363
Momentum_10day	-0.531295731
Wpr_5day	-0.777069942
Roc_10day	-0.910322837
MomentumAbs_10day	-1.021954138
Close_price_5day_lag	-1.045942782
Wpr_20day	-1.089955254
Stoch_slowd	-1.185339424
Stoch_fastk	-1.613221399



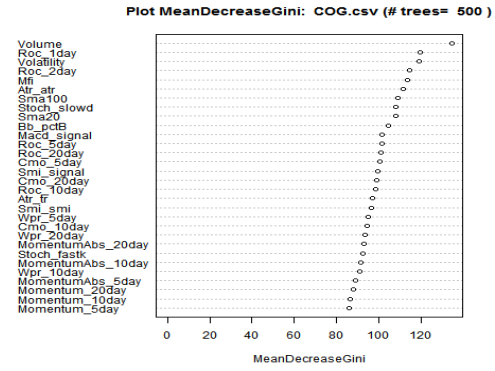
	perm.standardized
Momentum_10day	2.74946816
MomentumAbs_10day	2.437014877
Cmo_10day	1.892093686
Cmo_5day	1.763147551
Roc_1day	1.532870623
Roc_10day	1.450776433
Close_price_5day_lag	1.424518013
Atr_atr	1.158589145
Cmo_20day	1.153963864
Atr_tr	1.050495206
Mfi	1.040346147
Ema200	0.680387707
Momentum_5day	0.494595387
Macd_signal	0.492138058
Volatility	0.065313879
Stoch_fastk	0.062893615
Smi_signal	-0.037481182
Wpr_10day	-0.116639218
Stoch_slowd	-0.149584289
Wpr_5day	-0.251050242
Smi_smi	-0.5021822
Roc_5day	-0.581056661
Volume	-0.595962315
Roc_20day	-0.601729116
MomentumAbs_5day	-0.746415961
Roc_2day	-0.850460305
Bb_pctB	-0.853303642
Momentum_20day	-0.921782552
MomentumAbs_20day	-1.695816024
Wpr_20day	-1.903139476



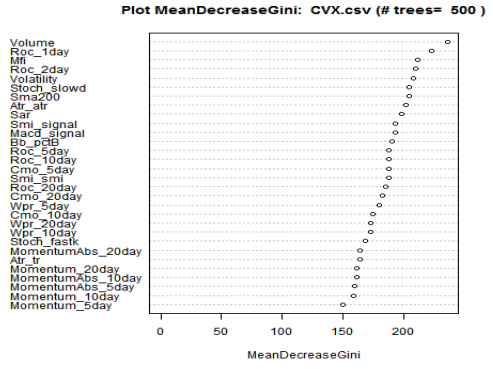
	perm.standardized
Roc_2day	2.676259779
MomentumAbs_10day	2.211144241
Sma100	2.021183041
Momentum_10day	1.346852552
Momentum_20day	1.202100974
Ema20	1.024460623
Stoch_slowd	0.842977733
MomentumAbs_20day	0.829257775
Atr_tr	0.579947893
Roc_20day	0.514914604
Roc_1day	0.492281522
Cmo_20day	0.219245577
Bb_pctB	-0.072031331
Wpr_20day	-0.105043347
Cmo_5day	-0.237886353
Cmo_10day	-0.256519899
Roc_10day	-0.279165892
Mfi	-0.339939275
Smi_signal	-0.346396659
Roc_5day	-0.350723996
Macd_signal	-0.374151348
Momentum_5day	-0.533194545
MomentumAbs_5day	-0.538384904
Stoch_fastk	-0.616593607
Atr_atr	-0.897646337
Volume	-1.181090698
Wpr_10day	-1.257937247
Wpr_5day	-1.948479438
Volatility	-2.515372278



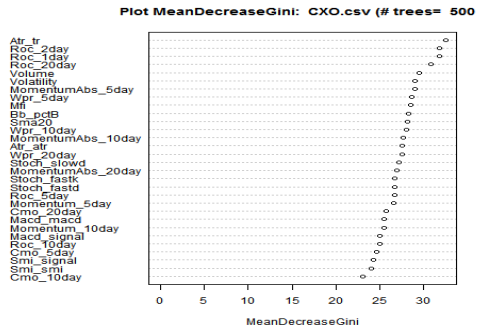
	perm.standardized
Cmo_5day	2.522614329
Smi_smi	1.701051451
Cmo_10day	1.472511013
Stoch_slowd	1.151776533
Wpr_5day	1.033045306
Smi_signal	0.845793324
Roc_2day	0.718115779
Wpr_10day	0.59845374
Ema20	0.533729907
Momentum_5day	0.527646892
Macd_signal	0.339059534
Cmo_20day	-0.032646085
MomentumAbs_10day	-0.156628804
MomentumAbs_5day	-0.172882044
Sma200	-0.265837951
Atr_tr	-0.27887354
Roc_5day	-0.398928592
Roc_10day	-0.680534588
MomentumAbs_20day	-0.723086094
Roc_1day	-0.735077397
Roc_20day	-0.846708923
Momentum_10day	-0.992639452
Mfi	-1.062139092
Stoch_fastk	-1.079241723
Volatility	-1.289423111
Momentum_20day	-1.377616742
Wpr_20day	-1.411674528
Volume	-1.714353557
Atr_atr	-1.768582364
Bb_pctB	-1.934596131



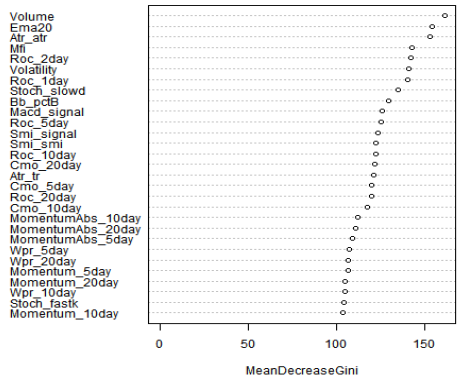
	perm.standardized
Cmo_20day	2.549096422
Smi_smi	1.870412042
Cmo_10day	1.395425797
Roc_2day	1.333547368
Smi_signal	1.149933793
Macd_signal	1.059320242
Wpr_20day	0.922033894
Bb_pctB	0.607436655
Cmo_5day	0.323478479
Wpr_10day	0.260100579
Stoch_fastk	0.107955859
Stoch_slowd	-0.057379404
Atr_atr	-0.074626007
Sma100	-0.078926978
Roc_10day	-0.085621347
MomentumAbs_5day	-0.199491824
Momentum_20day	-0.221250616
Volatility	-0.306425066
Mfi	-0.360430832
Roc_1day	-0.396614604
Wpr_5day	-0.458603056
MomentumAbs_20day	-0.594432281
Sma20	-0.610529752
MomentumAbs_10day	-0.663887528
Momentum_10day	-0.850565237
Volume	-1.046266337
Roc_5day	-1.099181707
Momentum_5day	-1.473251698
Roc_20day	-1.704660279
Atr_tr	-3.339980505



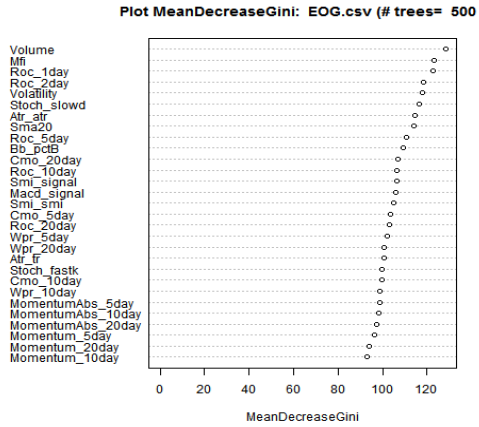
	perm.standardized
Roc_1day	2.020862069
Momentum_20day	1.794154618
Wpr_10day	1.478062365
Roc_2day	1.346778648
Cmo_5day	1.239280656
MomentumAbs_20day	0.666710955
Volume	0.620245634
Volatility	0.417343637
Stoch_fastk	0.165344462
Roc_5day	0.153929085
Stoch_slowd	-0.027153407
Sar	-0.16518585
Smi_signal	-0.215754908
Momentum_5day	-0.357374368
Roc_20day	-0.465943624
Wpr_5day	-0.470276137
Smi_smi	-0.506890787
Cmo_10day	-0.559351491
Bb_pctB	-0.586388819
Sma200	-0.813633675
MomentumAbs_10day	-0.920636666
MomentumAbs_5day	-1.154956486
Momentum_10day	-1.166577573
Wpr_20day	-1.208440043
Atr_tr	-1.245356962
Mfi	-1.559519738
Macd_signal	-1.701531894
Cmo_20day	-1.929089983
Atr_atr	-2.309139988
Roc_10day	-2.528410001



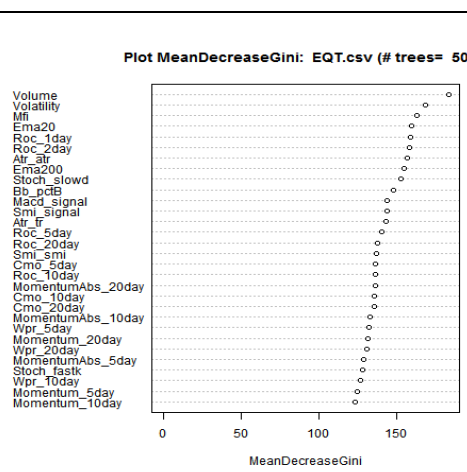
	perm.standardized
Cmo_20day	1.69731569
Sma20	1.419471514
Macd_macd	1.369532082
Smi_smi	1.299650226
Macd_signal	0.664243111
Momentum_5day	0.477637871
Cmo_10day	0.422953668
Stoch_fastd	0.394020578
Wpr_5day	0.240981422
Smi_signal	0.240401731
MomentumAbs_20day	0.231967962
MomentumAbs_10day	0.102244199
Roc_1day	0.099471281
Roc_2day	0.061724906
MomentumAbs_5day	-0.066516914
Wpr_20day	-0.160219443
Volatility	-0.218009676
Roc_20day	-0.279879478
Roc_5day	-0.331133692
Cmo_5day	-0.35296938
Atr_tr	-0.397922775
Volume	-0.406308506
Atr_atr	-0.451864981
Stoch_slowd	-0.496295132
Bb_pctB	-0.578787471
Mfi	-0.822555155
Momentum_20day	-0.959332772
Wpr_10day	-1.015904603
Stoch_fastk	-1.260313119
Momentum_10day	-1.337401677
Roc_10day	-2.240395696



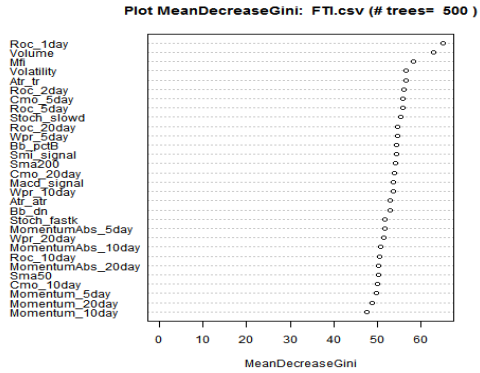
	perm.standardized
Roc_1day	2.899878088
Roc_2day	2.759403319
Mfi	1.516543214
Macd_signal	1.368248195
Momentum_10day	1.196843392
Atr_tr	1.060245869
Smi_smi	0.959776105
Roc_10day	0.620675722
Wpr_5day	0.437838429
Stoch_fastk	0.392408324
Smi_signal	0.322306795
MomentumAbs_10day	-0.059194509
Cmo_20day	-0.092008239
Ema20	-0.112475144
Stoch_slowd	-0.362106577
Atr_atr	-0.424925853
MomentumAbs_5day	-0.453835723
Volume	-0.486740578
MomentumAbs_20day	-0.606146925
Momentum_5day	-0.763699472
Cmo_10day	-0.79916553
Wpr_20day	-0.969467215
Momentum_20day	-1.047314832
Roc_20day	-1.229080464
Roc_5day	-1.390347742
Volatility	-1.41869242
Wpr_10day	-1.508575949
Bb_pctB	-1.753988602
Cmo_5day	-2.080991782



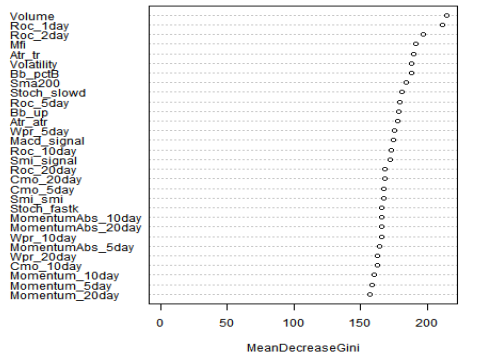
A	B
	perm.standardized
Wpr_20day	2.588025795
Wpr_10day	1.520872448
Roc_1day	1.318715205
Smi_signal	1.144793898
Wpr_5day	1.096423048
Stoch_fastk	0.954473658
Volatility	0.847208273
Volume	0.805386388
Macd_signal	0.444968898
Bb_pctB	0.297313325
MomentumAbs_20day	0.152127166
Mfi	-0.12856941
Atr_atr	-0.203866533
Momentum_20day	-0.27080811
Smi_smi	-0.405244126
Roc_5day	-0.424084176
Roc_20day	-0.424647475
MomentumAbs_5day	-0.687131728
Cmo_20day	-0.793421162
Momentum_5day	-0.807830299
Momentum_10day	-0.808166689
Atr_tr	-0.810326466
Cmo_5day	-0.926468844
Sma20	-1.317688705
MomentumAbs_10day	-1.860630447
Roc_10day	-1.921831503
Roc_2day	-2.145700401
Cmo_10day	-2.177561727
Stoch_slowd	-2.398206316



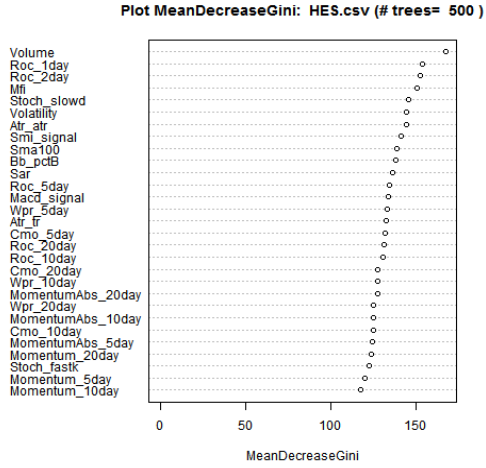
A	B
	perm.standardized
Volatility	2.205145532
Roc_2day	2.195188471
Macd_signal	2.187618782
Atr_atr	1.948991897
MomentumAbs_10day	1.860453972
Momentum_10day	1.852687107
Roc_1day	1.521310074
MomentumAbs_5day	1.135872155
Roc_10day	1.049212177
Roc_20day	0.87072353
Mfi	0.710577194
Ema200	0.605937103
MomentumAbs_20day	0.595014784
Momentum_20day	0.574718051
Wpr_20day	0.259225514
Momentum_5day	0.233200155
Ema20	0.206874433
Volume	0.140908236
Cmo_10day	0.061746541
Stoch_fastk	-0.089698169
Atr_tr	-0.492822306
Stoch_slowd	-0.620225547
Bb_pctB	-0.646192427
Roc_5day	-0.664629146
Smi_signal	-0.724714048
Cmo_20day	-0.915484156
Smi_smi	-1.175050972
Cmo_5day	-1.538516858
Wpr_10day	-1.665595232
Wpr_5day	-2.474722932



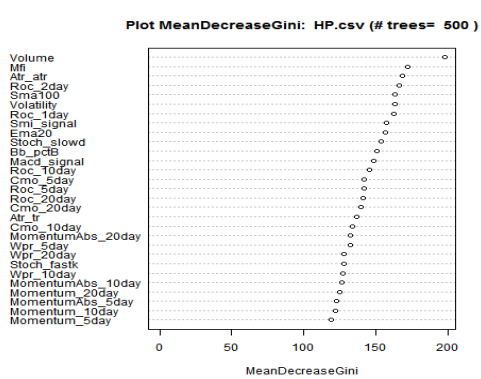
	perm.standardized
Stoch_fastk	2.432753832
Smi_signal	1.444030453
Wpr_20day	1.089391856
Wpr_5day	0.809306854
Mfi	0.656688654
Sma200	0.620982765
Roc_20day	0.456621526
Macd_signal	0.44064788
Volatility	0.400230335
Atr_tr	0.371682883
Roc_1day	0.358448528
Wpr_10day	0.224135834
Bb_pctB	0.198726157
Volume	0.190395068
MomentumAbs_20day	0.131143296
Momentum_5day	-0.008189545
MomentumAbs_5day	-0.042176616
Roc_5day	-0.069798096
Roc_10day	-0.079304546
Momentum_20day	-0.157585535
Atr_atr	-0.174178432
Cmo_5day	-0.300597305
Stoch_slowd	-0.301685902
Cmo_20day	-0.308735067
Cmo_10day	-0.411275991
Sma50	-0.592519718
MomentumAbs_10day	-0.675205321
Momentum_10day	-0.689110973
Roc_2day	-0.762966708
Bb_dn	-0.970941003



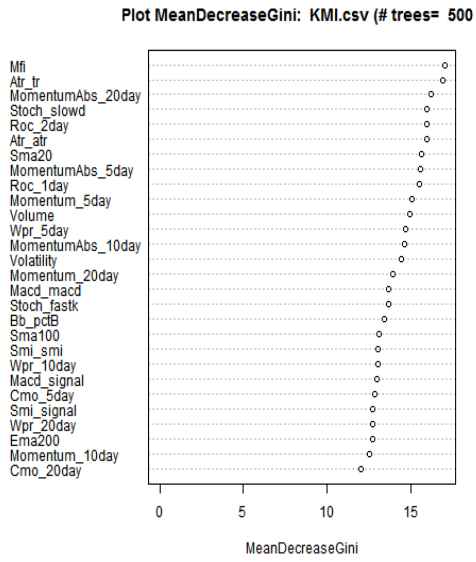
	perm.standardized
Mfi	2.354459381
Sma200	1.723569159
MomentumAbs_5day	1.527857465
Momentum_5day	1.306369428
Roc_10day	1.125770701
MomentumAbs_10day	1.082979865
Smi_smi	1.003782568
Wpr_10day	0.907306763
Bb_up	0.837442388
Cmo_5day	0.693980888
Cmo_10day	0.66741679
Momentum_10day	0.630013203
Roc_20day	0.59577399
Stoch_fastk	0.580645893
Atr_atr	0.49502556
Atr_tr	0.449169058
Smi_signal	0.153109608
Roc_1day	0.14281032
Stoch_slowd	-0.021955877
Roc_2day	-0.093087
Wpr_5day	-0.329153507
MomentumAbs_20day	-0.370462559
Volume	-0.423743542
Momentum_20day	-0.510640846
Volatility	-0.727023416
Wpr_20day	-0.878528293
Bb_pctB	-0.890847166
Cmo_20day	-0.933704741
Macd_signal	-1.873569789
Roc_5day	-2.337866855



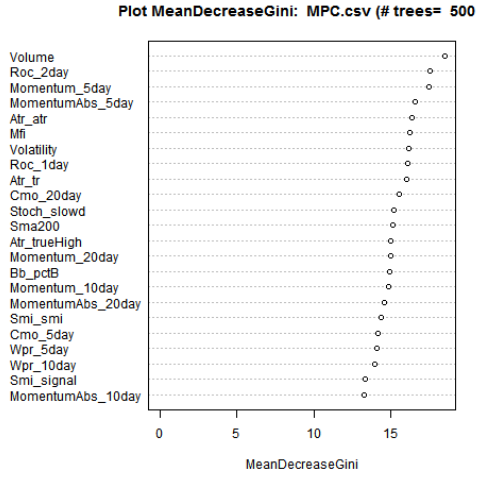
	perm.standardized
MomentumAbs_5day	1.773110637
Sar	1.527450214
Momentum_5day	1.358828921
Momentum_20day	1.336377876
MomentumAbs_20day	1.122182303
Cmo_5day	1.079448289
Roc_5day	0.992310448
Wpr_5day	0.992178907
Cmo_10day	0.945993665
Wpr_10day	0.891687137
Mfi	0.766935038
MomentumAbs_10day	0.59793567
Cmo_20day	0.574895743
Roc_10day	0.524040786
Roc_20day	0.508163741
Atr_atr	0.42457552
Sma100	0.423478284
Momentum_10day	0.380291534
Macd_signal	0.076582794
Atr_tr	0.037957744
Bb_pctB	0.035456252
Volatility	-0.371744762
Stoch_slowd	-0.418594609
Stoch_fastk	-0.448043138
Wpr_20day	-0.778319812
Volume	-1.045168358
Smi_signal	-1.410426384
Roc_2day	-2.179956078
Roc_1day	-2.456553865



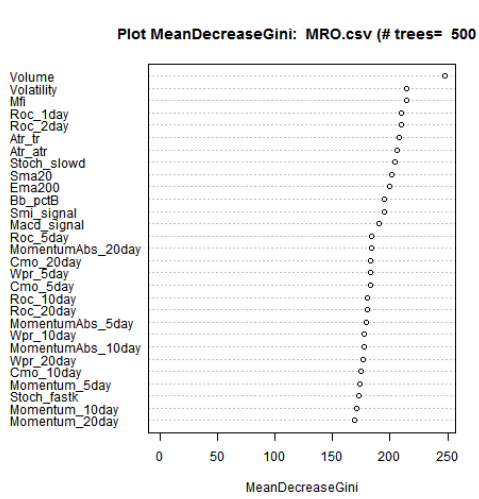
	perm.standardized
Roc_2day	1.987108169
Roc_5day	1.605210248
Roc_10day	0.959033675
MomentumAbs_10day	0.763088146
Sma100	0.736687523
Momentum_20day	0.660242974
Momentum_10day	0.606714246
Cmo_20day	0.571947796
Smi_signal	0.537113596
Roc_20day	0.501180857
MomentumAbs_20day	0.197486584
Macd_signal	-0.025329105
Volatility	-0.085951587
Wpr_5day	-0.149399352
Cmo_10day	-0.155893112
Stoch_fastk	-0.198043698
Wpr_10day	-0.274687884
Wpr_20day	-0.283062315
Mfi	-0.288374577
Cmo_5day	-0.480122402
Roc_1day	-0.519682898
Atr_atr	-0.702499624
Ema20	-0.722095951
Momentum_5day	-0.860604015
Stoch_slowd	-0.985943463
MomentumAbs_5day	-1.112682645
Atr_tr	-1.457634816
Volume	-1.562982652
Bb_pctB	-2.005586461



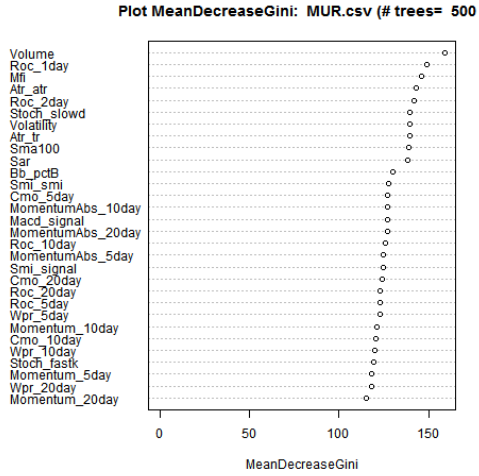
	perm.standardized
Roc_1day	2.617898748
Atr_tr	1.647807099
Roc_2day	1.25933535
Sma100	0.986747302
Wpr_10day	0.829138729
Ema200	0.654902329
Wpr_5day	0.481006206
Stoch_fastk	0.354355741
Momentum_20day	0.340718026
Macd_macd	0.324458955
MomentumAbs_20day	0.269400051
Sma20	0.055915316
Smi_signal	0.043905975
Mfi	-0.049841018
Wpr_20day	-0.140351561
Smi_smi	-0.142370002
Macd_signal	-0.156939618
Stoch_slowd	-0.266055368
Momentum_5day	-0.27819864
MomentumAbs_5day	-0.290589495
Cmo_5day	-0.600854757
Bb_pctB	-0.661857476
MomentumAbs_10day	-0.683655288
Atr_atr	-0.87816916
Cmo_20day	-1.03969306
Volume	-1.04042354
Momentum_10day	-1.354010881
Volatility	-1.638848995



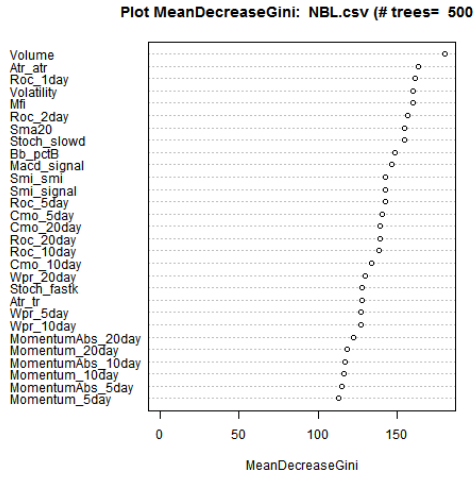
	A	B
		perm.standardized
Roc_2day		1.515547224
Cmo_5day		1.351850383
Wpr_10day		0.870660019
Momentum_5day		0.859850643
Bb_pctB		0.77114741
Roc_1day		0.737505978
Atr_tr		0.551516361
MomentumAbs_10day		0.543418509
Stoch_slowd		0.404392723
Volatility		0.159506573
Smi_signal		-0.108013376
Momentum_10day		-0.312632831
Mfi		-0.314644836
Sma200		-0.393087811
Smi_smi		-0.412250464
Wpr_5day		-0.420555683
Volume		-0.601999081
Atr_trueHigh		-0.717474116
Atr_atr		-1.038583141
MomentumAbs_20day		-1.479455188
MomentumAbs_5day		-1.557953815
Cmo_20day		-2.105568768
Momentum_20day		-2.452725035



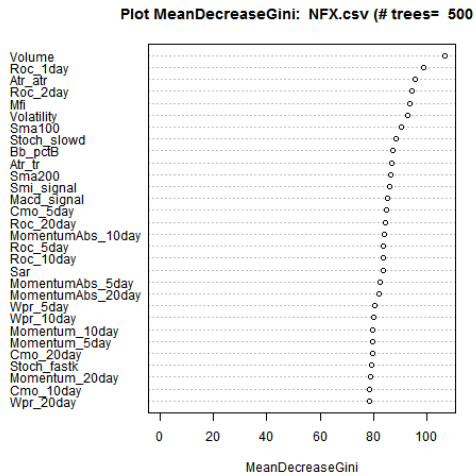
	perm.standardized
Roc_2day	1.931517035
Wpr_20day	1.133244749
Cmo_5day	1.053830835
Volatility	1.030089626
Wpr_5day	1.025139629
Ema200	0.91158807
Sma20	0.702223207
MomentumAbs_5day	0.673654113
Mfi	0.611247413
Stoch_slowd	0.590226784
Bb_pctB	0.528991346
Stoch_fastk	0.503004489
Momentum_5day	0.4260893
Roc_20day	0.376655434
Wpr_10day	0.197800662
Macd_signal	0.153897406
Roc_5day	0.078769558
Momentum_20day	0.017574373
Cmo_20day	0.000894487
Roc_10day	0.000705641
MomentumAbs_20day	-0.069055463
MomentumAbs_10day	-0.41565769
Roc_1day	-0.444006258
Momentum_10day	-0.679438164
Cmo_10day	-0.975978114
Volume	-1.135708447
Atr_atr	-1.205132453
Atr_tr	-2.173080057
Smi_signal	-2.42602834



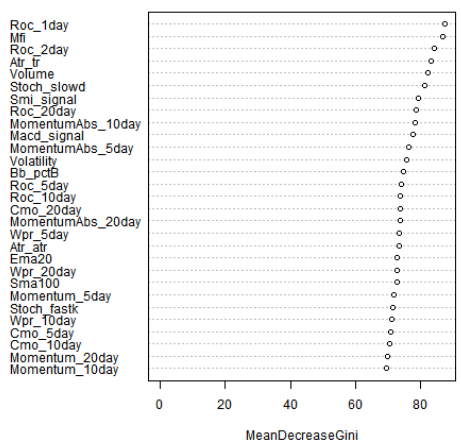
	perm.standardized
Atr_tr	2.100144894
Bb_pctB	1.826884127
Wpr_5day	1.558384944
Volume	1.339763092
Stoch_fastk	1.203589451
Wpr_20day	1.199486783
Cmo_5day	1.179089447
Momentum_10day	1.146659028
Wpr_10day	0.813136702
Roc_5day	0.779984725
Sar	0.699002161
Cmo_20day	0.556993745
MomentumAbs_10day	0.472431868
Volatility	0.373794624
Smi_smi	0.347688026
Momentum_5day	0.279636723
Mfi	0.145351791
Atr_atr	0.098086361
MomentumAbs_5day	0.020317277
Smi_signal	-0.053411055
Roc_20day	-0.064019143
Roc_10day	-0.073949595
Cmo_10day	-0.102575566
Stoch_slowd	-0.216535674
Sma100	-0.300374558
Roc_1day	-0.384014184
Roc_2day	-0.781904474
MomentumAbs_20day	-1.063334437
Macd_signal	-1.102193897
Momentum_20day	-1.581405734



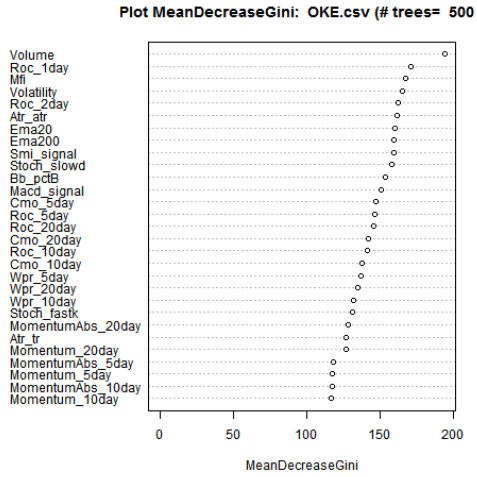
	A	B
		perm.standardized
Cmo_10day		1.902650238
MomentumAbs_5day		1.409633714
Cmo_20day		1.193380011
Roc_20day		1.013030629
Wpr_5day		0.977602602
Momentum_20day		0.937725454
Stoch_fastk		0.902025187
Stoch_slowd		0.79258742
Momentum_5day		0.747589923
Roc_1day		0.664301232
Smi_smi		0.395529349
Wpr_10day		0.368694088
MomentumAbs_20day		0.358218787
Bb_pctB		0.356498402
Roc_5day		0.299175975
Roc_10day		0.210586552
Smi_signal		0.181266631
Atr_tr		0.060534326
Roc_2day		0.044046861
Volume		-0.169481203
Volatility		-0.231221584
Macd_signal		-0.330465496
Wpr_20day		-0.402388452
MomentumAbs_10day		-0.668256135
Cmo_5day		-0.674978811
Momentum_10day		-0.696820199
Sma20		-0.930303572
Mfi		-1.62212284
Atr_atr		-2.596468812



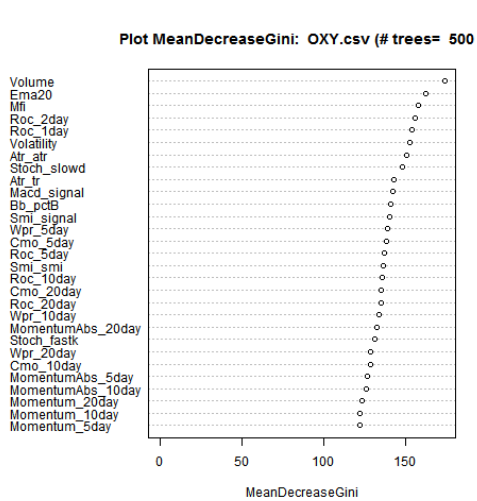
	A	B
		perm.standardized
Bb_pctB		2.21375796
Stoch_slowd		1.668326018
MomentumAbs_20day		1.325798479
Smi_signal		1.321873218
Macd_signal		1.227877133
Sar		1.107663247
Volume		1.01409597
Sma200		0.908095342
Atr_atr		0.751699709
Roc_20day		0.536511185
Momentum_20day		0.454041315
Wpr_5day		0.352933415
Cmo_5day		0.341370347
Sma100		0.316874494
Wpr_20day		0.238059588
Roc_10day		0.223281704
Mfi		0.209506409
MomentumAbs_10day		0.027346884
Volatility		-0.032491286
Momentum_5day		-0.2825171
Atr_tr		-0.314708616
MomentumAbs_5day		-0.402119435
Momentum_10day		-0.45731093
Roc_2day		-0.49037015
Wpr_10day		-0.595533465
Stoch_fastk		-0.662577148
Cmo_20day		-0.70090606
Roc_5day		-1.10482664
Roc_1day		-1.426326579
Cmo_10day		-1.542018577



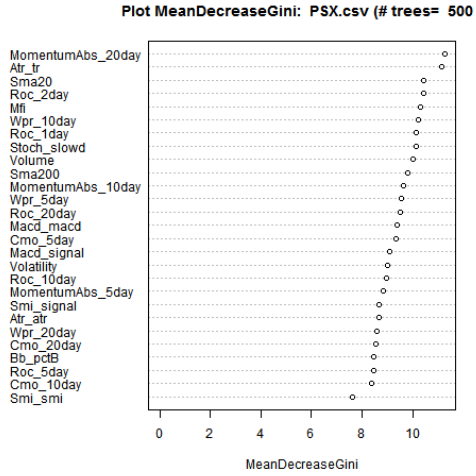
	perm.standardized
Roc_5day	3.31564751
Momentum_5day	2.61019121
Ema20	2.155462892
Atr_atr	1.457346541
MomentumAbs_5day	1.054016773
Sma100	1.00880843
Roc_2day	0.89478737
Roc_10day	0.876544666
Volatility	0.586718272
Atr_tr	0.560238082
Volume	0.478165258
MomentumAbs_20day	0.414310313
Stoch_slowd	0.188619046
Macd_signal	-0.015078289
Wpr_5day	-0.047899934
Mfi	-0.263552202
Roc_20day	-0.306277224
Cmo_5day	-0.32504838
Wpr_20day	-0.384815911
Momentum_20day	-0.407569459
Bb_pctB	-0.603875802
Stoch_fastk	-0.609751465
Cmo_10day	-0.909095962
MomentumAbs_10day	-1.118618379
Momentum_10day	-1.246189364
Smi_signal	-1.252213774
Roc_1day	-1.479553344
Wpr_10day	-2.186781127
Cmo_20day	-2.265529971



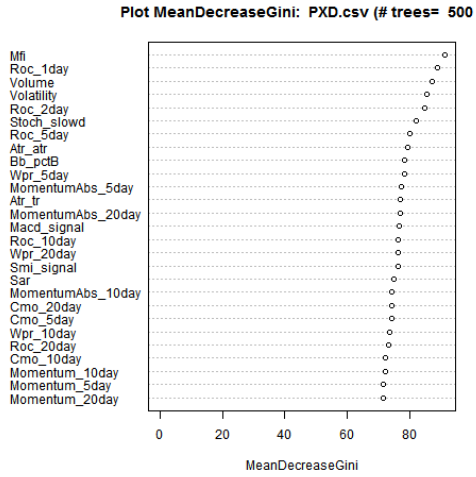
	perm.standardized
Volatility	2.280070575
Cmo_20day	1.956294111
Cmo_10day	1.869757826
Atr_tr	1.284358464
Atr_atr	1.098864161
Macd_signal	1.04157031
Smi_signal	0.941000786
Roc_1day	0.795781839
Wpr_20day	0.785167152
Bb_pctB	0.440215413
Roc_5day	0.408325263
Ema200	0.333391962
MomentumAbs_5day	0.296378154
Roc_20day	0.294049114
Cmo_5day	0.27711024
Stoch_slowd	0.226161165
Momentum_20day	0.225804808
Mfi	0.129527575
Momentum_5day	0.107196714
MomentumAbs_20day	-0.053734908
MomentumAbs_10day	-0.094341488
Momentum_10day	-0.228687116
Wpr_5day	-0.2717924
Ema20	-0.524745903
Wpr_10day	-0.554070217
Stoch_fastk	-0.820960774
Volume	-0.856329421
Roc_2day	-1.021736809
Roc_10day	-1.871649599



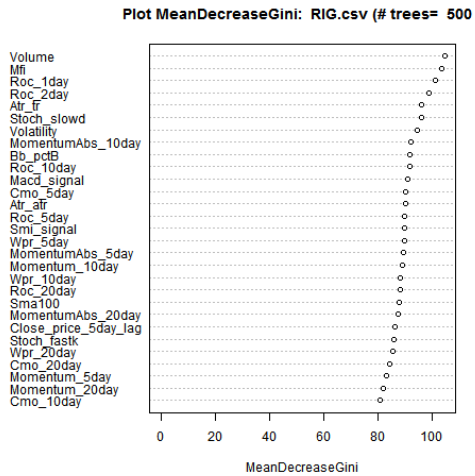
	perm.standardized
Volume	2.273218743
Roc_5day	1.986430355
Wpr_10day	1.352507542
Mfi	1.251602995
Atr_atr	0.999739284
Cmo_5day	0.66933057
Smi_smi	0.50337372
Stoch_fastk	0.498488081
Momentum_10day	0.465712858
Ema20	0.464719392
Wpr_5day	0.449188504
MomentumAbs_10day	0.40111865
Atr_tr	0.22804993
Bb_pctB	0.186548535
Roc_20day	0.179632029
Volatility	0.143485409
Wpr_20day	0.043656732
MomentumAbs_20day	-0.1255392
Roc_2day	-0.186669737
Smi_signal	-0.288050068
Momentum_20day	-0.495723368
Momentum_5day	-0.512518825
Cmo_20day	-0.639467059
Roc_10day	-0.809784639
MomentumAbs_5day	-0.831316525
Cmo_10day	-0.989379544
Macd_signal	-1.140742754
Roc_1day	-1.304754138
Stoch_slowd	-1.360426519



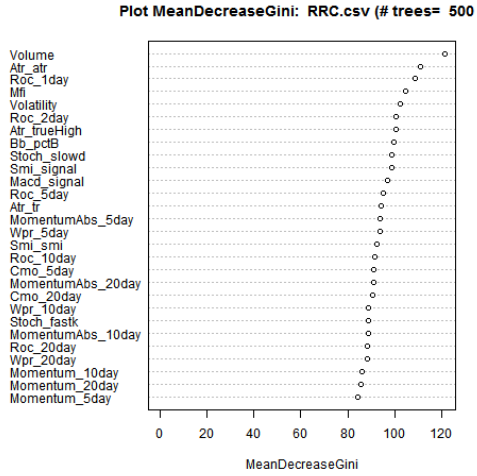
	A	B
		perm.standardized
Cmo_5day		1.823216644
Cmo_10day		1.606533331
Wpr_20day		1.264096415
Wpr_5day		1.195661438
Atr_tr		0.983800267
Smi_smi		0.878041009
MomentumAbs_20day		0.707125394
MomentumAbs_5day		0.563434059
Atr_atr		0.542421054
Roc_2day		0.31887319
Roc_1day		0.310790673
Bb_pctB		0.201006926
Wpr_10day		0.180005689
Mfi		0.05827088
Smi_signal		0.050200842
Stoch_slowd		0.015180027
Cmo_20day		-0.252828701
Volatility		-0.286104571
Roc_10day		-0.37000987
Volume		-0.479863689
Macd_signal		-0.479936046
Sma200		-0.62109319
Roc_5day		-0.626798119
Roc_20day		-0.677943323
Sma20		-0.739469469
Macd_macd		-1.351774554
MomentumAbs_10day		-1.781500611



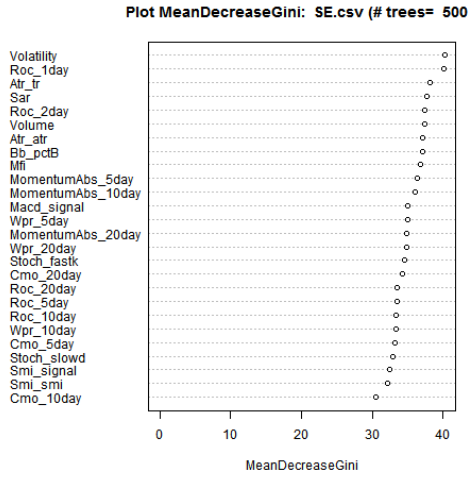
	A	B
		perm.standardized
Roc_20day		2.342192375
Sar		1.872342767
Momentum_10day		1.765861596
MomentumAbs_20day		1.184607928
MomentumAbs_10day		1.000462817
Cmo_20day		0.905032669
Volume		0.896331871
Macd_signal		0.5389566
Volatility		0.510652802
Cmo_10day		0.500874997
Smi_signal		0.499378771
Atr_atr		0.441477965
Atr_tr		0.402169982
Bb_pctB		0.29645728
Cmo_5day		-0.04535571
Mfi		-0.079740223
Roc_10day		-0.094805402
Momentum_20day		-0.157805261
Momentum_5day		-0.278513056
Wpr_20day		-0.340706962
Roc_5day		-0.593742789
Roc_1day		-0.889420107
Wpr_10day		-0.993389218
MomentumAbs_5day		-1.043266442
Stoch_slowd		-1.810118893
Wpr_5day		-2.668548148
Roc_2day		-2.764630433



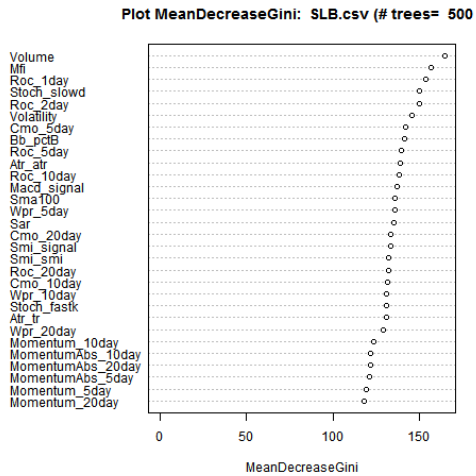
	A	B
		perm.standardized
Roc_1day		2.340052745
Volume		1.64865478
Wpr_5day		1.341467019
Cmo_10day		1.19426065
Smi_signal		1.169695089
Momentum_10day		1.119001497
MomentumAbs_10day		0.851606003
Atr_atr		0.758036316
MomentumAbs_20day		0.590150147
Wpr_10day		0.485984756
Cmo_20day		0.459114657
Cmo_5day		0.361060572
Bb_pctB		0.263504614
Macd_signal		0.162771234
Atr_tr		0.109787116
Roc_10day		0.10391694
Sma100		0.001369244
Roc_2day		-0.040004172
Stoch_slowd		-0.080386435
Momentum_5day		-0.159255128
Roc_20day		-0.18196838
Mfi		-0.202870221
Volatility		-0.368299894
Momentum_20day		-0.430991378
Roc_5day		-0.60500907
Wpr_20day		-1.238881356
MomentumAbs_5day		-1.388008126
Stoch_fastk		-1.563959533
Close_price_5day_lag		-1.760647073



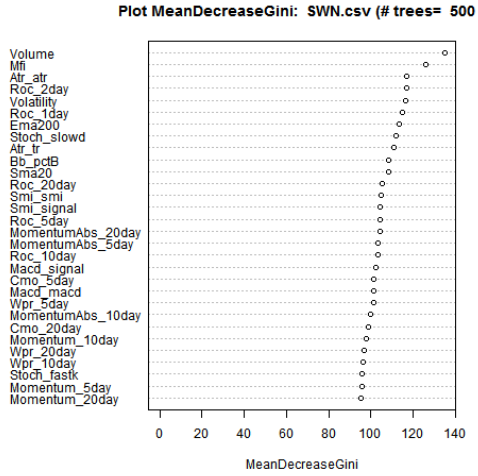
	perm.standardized
Atr_trueHigh	2.568701616
Atr_atr	1.867815228
Smi_smi	1.750055457
Smi_signal	1.608270509
Bb_pctB	1.567520304
Roc_20day	1.310720361
Momentum_5day	1.157722539
Atr_tr	0.741638857
Volatility	0.719079961
Stoch_slowd	0.694815785
Momentum_10day	0.692829551
Macd_signal	0.618080871
Wpr_10day	0.565514422
MomentumAbs_20day	0.54087133
Roc_10day	0.44975278
Cmo_20day	0.374122672
MomentumAbs_5day	0.18247601
Roc_2day	-0.026217246
Stoch_fastk	-0.088128193
Roc_5day	-0.093210994
Volume	-0.2941277
Wpr_5day	-0.397639196
Wpr_20day	-0.507435117
Cmo_5day	-0.533840746
Momentum_20day	-0.826611683
Roc_1day	-0.891770851
Mfi	-0.965399284
MomentumAbs_10day	-1.300672161



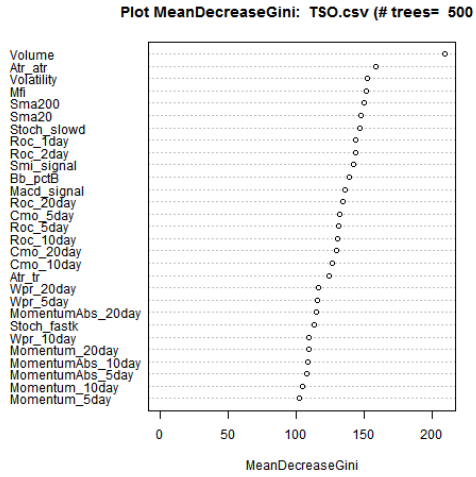
	perm.standardized
Wpr_10day	1.740651499
Stoch_fastk	1.195495241
Volume	1.065163468
Roc_2day	0.858794923
Roc_1day	0.799042279
Wpr_20day	0.487761507
Mfi	0.407524884
Roc_20day	0.268935615
Wpr_5day	0.260558808
Stoch_slowd	-0.062651945
MomentumAbs_5day	-0.193333958
MomentumAbs_10day	-0.341363488
Volatility	-0.493668641
Bb_pctB	-0.537966012
Cmo_5day	-0.538957557
Cmo_20day	-0.57231
Cmo_10day	-0.635351752
Atr_atr	-0.706898426
Macd_signal	-0.862554932
Sar	-0.963405063
Roc_10day	-1.073807906
Roc_5day	-1.1063715
Smi_smi	-1.262005713
MomentumAbs_20day	-1.601675525
Atr_tr	-1.853283538
Smi_signal	-2.325988658



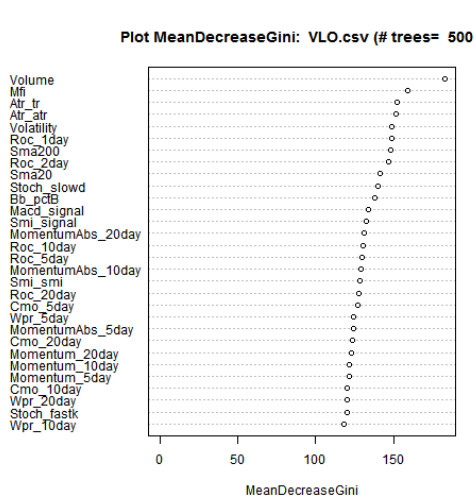
	perm.standardized
Momentum_10day	2.664748283
Mfi	2.533915789
MomentumAbs_10day	1.757010554
Sar	1.232350448
Wpr_20day	1.165613006
Volatility	0.877637487
MomentumAbs_20day	0.823319206
Macd_signal	0.799742166
Wpr_5day	0.671655528
Sma100	0.499768401
Cmo_20day	0.306327459
Roc_10day	0.222113945
Stoch_fastk	0.161606094
Roc_20day	0.066545484
Wpr_10day	0.011248883
Atr_tr	0.006777782
Smi_signal	-0.004800104
Stoch_slowd	-0.012010646
Atr_atr	-0.056876391
Bb_pctB	-0.120956723
Momentum_20day	-0.255556812
Cmo_10day	-0.788857399
Momentum_5day	-0.951087853
Cmo_5day	-0.963320088
Roc_2day	-1.11590785
MomentumAbs_5day	-1.193268042
Roc_5day	-1.212939748
Roc_1day	-1.327280816
Volume	-1.472801663
Smi_smi	-1.512372729



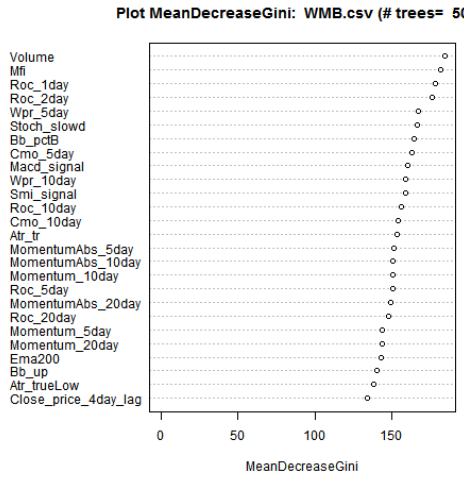
	A	B
		perm.standardized
	Roc_5day	1.545799431
	Wpr_20day	1.29095823
	Stoch_slowd	1.170945242
	MomentumAbs_20day	0.870020733
	Wpr_5day	0.788706144
	Momentum_20day	0.624275031
	Roc_10day	0.623099705
	Roc_2day	0.622249115
	MomentumAbs_10day	0.507157272
	Roc_1day	0.473848386
	Mfi	0.450465813
	Smi_smi	0.407569167
	Smi_signal	0.390096641
	Macd_macd	0.277557194
	Stoch_fastk	0.018227278
	Volume	-0.170014652
	Volatility	-0.209052145
	Roc_20day	-0.228233277
	Momentum_5day	-0.334722603
	Macd_signal	-0.395957443
	Ema200	-0.651882815
	Wpr_10day	-0.706902379
	Sma20	-0.918884528
	Bb_pctB	-1.067787775
	Cmo_10day	-1.078154808
	Cmo_20day	-1.332416887
	Atr_tr	-1.42766874
	Cmo_5day	-1.543373315
	MomentumAbs_5day	-1.556716445
	Momentum_10day	-1.815495928
	Atr_atr	-1.948085811



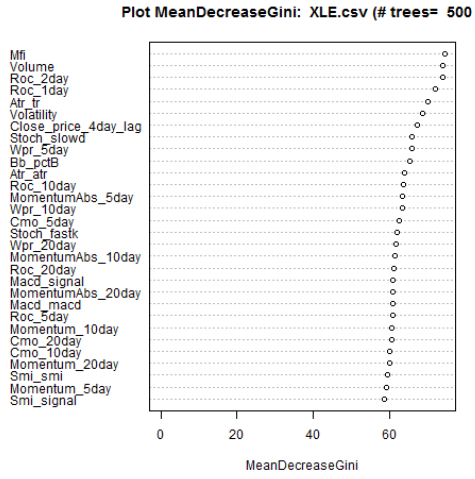
	A	B
		perm.standardized
	Roc_20day	1.604437113
	Wpr_20day	1.554924154
	Cmo_10day	1.270080985
	Roc_5day	1.119483781
	Cmo_5day	1.087991962
	Bb_pctB	1.032308008
	Momentum_10day	1.008765804
	Wpr_10day	0.904513715
	MomentumAbs_10day	0.529141177
	Volatility	0.440550265
	Macd_signal	0.427807479
	Smi_signal	0.307154621
	Momentum_20day	0.219783662
	Atr_tr	0.173885894
	Mfi	0.035131085
	Stoch_fastk	0.019966381
	Wpr_5day	-0.0384412
	Sma200	-0.069096232
	Momentum_5day	-0.125518065
	Roc_1day	-0.137689472
	Cmo_20day	-0.195720512
	Roc_10day	-0.228013808
	Roc_2day	-0.303251854
	Stoch_slowd	-0.343510548
	Sma20	-0.539738255
	MomentumAbs_5day	-0.628299126
	Atr_atr	-0.631340653
	Volume	-0.859099049
	MomentumAbs_20day	-1.078179694



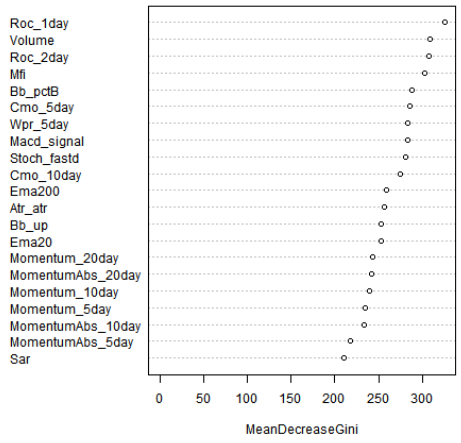
	A	B
		perm.standardized
	Sma20	2.168634685
	Sma200	1.634251331
	Volume	1.474555878
	Volatility	1.145022631
	Wpr_10day	0.970217532
	Atr_atr	0.66798491
	Smi_signal	0.593357323
	Cmo_5day	0.282896244
	Roc_20day	0.271589875
	Roc_1day	0.245618709
	Stoch_fastk	0.207768109
	MomentumAbs_20day	0.117693813
	MomentumAbs_5day	0.04285846
	Mfi	0.001860427
	Roc_5day	-0.124210268
	Momentum_10day	-0.166286407
	MomentumAbs_10day	-0.295573566
	Smi_smi	-0.306375
	Cmo_10day	-0.448323508
	Momentum_20day	-0.49771359
	Roc_10day	-0.537522557
	Atr_tr	-0.653510708
	Macd_signal	-0.669665585
	Momentum_5day	-0.824060181
	Stoch_slowd	-0.880228607
	Cmo_20day	-0.930037105
	Bb_pctB	-0.965652898
	Roc_2day	-1.278503495
	Wpr_5day	-1.59029976
	Wpr_20day	-3.355984871



A	B
	perm.standardized
Smi_signal	1.485183485
Stoch_slowd	1.328885332
MomentumAbs_5day	1.202710087
MomentumAbs_10day	0.929110772
Wpr_10day	0.652022742
Roc_10day	0.53865343
Momentum_10day	0.510146809
Mfi	0.478235238
Close_price_4day_lag	0.442109978
Roc_20day	0.268349916
Momentum_5day	0.213590694
Volume	0.135723619
Roc_2day	0.048531493
MomentumAbs_20day	-0.070261027
Cmo_5day	-0.149168225
Roc_5day	-0.18590673
Wpr_5day	-0.240361407
Atr_tr	-0.316986737
Ema200	-0.324527205
Momentum_20day	-0.379490563
Atr_trueLow	-0.467748002
Bb_pctB	-0.499769348
Macd_signal	-0.599959148
Bb_up	-0.641770316
Roc_1day	-1.060398842
Cmo_10day	-1.08515346



A	B
	perm.standardized
Momentum_10day	2.074645265
MomentumAbs_10day	1.793285857
Roc_10day	1.448368777
Smi_smi	1.398109204
MomentumAbs_5day	1.084293572
Wpr_5day	1.056260694
Smi_signal	0.929895318
Momentum_5day	0.860893427
Roc_20day	0.842843114
Bb_pctB	0.667610504
Stoch_slowd	0.362669476
Stoch_fastk	0.278135083
Roc_5day	0.105564989
Cmo_5day	-0.062617825
Roc_1day	-0.124363492
Wpr_20day	-0.198234982
Macd_macd	-0.458359156
Mfi	-0.48361069
Wpr_10day	-0.601287442
Momentum_20day	-0.906115256
Atr_atr	-0.942968961
Atr_tr	-1.005896359
MomentumAbs_20day	-1.03254572
Close_price_4day_lag	-1.281339064
Macd_signal	-1.432551332
Volatility	-1.692683785
Cmo_10day	-2.226379389
Roc_2day	-2.534294844
Cmo_20day	-2.545368963
Volume	-2.823705385



A	B
	perm.standardized
Atr_atr	2.290827729
MomentumAbs_10day	1.802155477
Momentum_10day	1.774480801
Momentum_5day	1.279728791
MomentumAbs_5day	1.015502074
Cmo_5day	0.7298667
Volume	0.671767938
MomentumAbs_20day	0.490549966
Roc_2day	0.222214327
Sar	0.207008148
Roc_1day	-0.127715589
Ema20	-0.143430502
Momentum_20day	-0.208500054
Bb_up	-0.24304883
Ema200	-0.386664074
Stoch_fastd	-0.422497154
Bb_pctB	-0.775105983
Wpr_5day	-0.842781025
Mfi	-1.670509041
Macd_signal	-1.898323931
Cmo_10day	-1.990904665

Appendix 16 – The Volatility Prediction Code

This code refers to the generation of the volatility prediction. It is dependent on past EGARCH volatilities, volume, sentiment and sentiment momentum. The displayed piece of code only shows one example of the below listed implementation (i.e. ../models/vol_rnn_elman_sent.r). The other pieces of code listed in the below table are carbon copies of this one with different set of parameters.

Code Location	Description
../models/vol_rnn_elman.r	Generates the base volatility prediction for an Elman RNN model (i.e. without the influence of sentiment).
../models/vol_rnn_elman_sent.r	Generates the scenario volatility prediction for an Elman RNN model. In this case the sentiment at t-1 is added as an explanatory variable.
../models/vol_rnn_elman_sent_momentum1.r	Generates a scenario volatility prediction for an Elman RNN model. In this case the sentiment momentum a t and t-1 are added as explanatory variables.
../models/vol_rnn_elman_sent_momentum1_2.r	Generates a scenario volatility prediction for an Elman RNN model. In this case the sentiment momentum a t, t-1 and t-2 are added as explanatory variables.
../models/vol_rnn_jordan.r	Generates the base volatility prediction for a Jordan RNN model (i.e. without the influence of sentiment).
../models/vol_rnn_jordan_sent.r	Generates the scenario volatility prediction for a Jordan RNN model. In this case the sentiment at t-1 is added as an explanatory variable.
../models/vol_rnn_jordan_sent_momentum1.r	Generates a scenario volatility prediction for a Jordan RNN model. In this case the sentiment momentum a t and t-1 are added as explanatory variables.
../models/vol_rnn_jordan_sent_momentum1_2.r	Generates a scenario volatility prediction for a Jordan RNN model. In this case the sentiment momentum a t, t-1 and t-2 are added as explanatory variables.

```
. is_server_run = FALSE
if(is_server_run){
  setwd('/host/dsm1/fmare001/Thesis/Finance/z_Code')
}else{
  setwd('C:/Users/Fred/Desktop/Studies/MSc-DataScience/Research/Thesis/Finance/z_Code')
}

require(caret)
require(psych)
require(epiR)
require(randomForest)
require(MASS)
require(gbm)
require (e1071)
require(nnet)
require(rugarch)
source("utilities.r")
require(RSNNS)

source("models/model_utils.r")
source("utilities.r")

#author: Frederic Marechal
#date: Aug-Sep, 2017
```

```
#copied from https://ctszkin.com/2012/03/11/generating-a-laglead-variables/
shift = function(x,shift_by){
  stopifnot(is.numeric(shift_by))
  stopifnot(is.numeric(x))

  if (length(shift_by)>1)
    return(sapply(shift_by,shift, x=x))

  out=NULL
  abs_shift_by=abs(shift_by)
  if (shift_by > 0 )
    out=c(tail(x,-abs_shift_by),rep(NA,abs_shift_by))
  else if (shift_by < 0 )
    out=c(rep(NA,abs_shift_by), head(x,-abs_shift_by))
  else
    out=x
  out
}

normalise = function(x){
  res = (x-min(x))/(max(x)-min(x))
  return (res)
}

get_col_list = function(file_name, b_sent){

  cols = c()

  if (file_name == "EOG.csv" ||
      file_name == "HAL.csv" ||
      file_name == "OXY.csv" ||
      file_name == "WMB.csv" ||
      file_name == "MPC.csv" ||
      file_name == "NBL.csv" ||
      file_name == "COG.csv" ||
      file_name == "MRO.csv" ||
      file_name == "OKE.csv" ||
      file_name == "EQT.csv" ||
      file_name == "NFX.csv"){
    if (b_sent == TRUE){
      cols = c("m_1_vol","m_2_vol","m_3_vol",
"m_1_volume","m_2_volume","m_3_volume","Article_sentiment_1day_lag")
    }else{
      cols = c("m_1_vol","m_2_vol","m_3_vol", "m_1_volume","m_2_volume","m_3_volume")
    }
  }else{
    #default
    if (b_sent == TRUE){
      cols = c("m_1_vol", "m_1_volume","m_2_volume","m_3_volume","Article_sentiment_1day_lag")
    }else{
      cols = c("m_1_vol", "m_1_volume","m_2_volume","m_3_volume")
    }
  }

  return(cols)
}

index_name = "XLE"
number_sliding_windows = 100
file_names = c("XLE.csv",
"XOM.csv","CVX.csv","SLB.csv","PXD.csv","EOG.csv","HAL.csv","OXY.csv","COP.csv","APC.csv","VLO.csv","K
MI.csv","PSX.csv","SE.csv","BHI.csv","TSO.csv","APA.csv","DVN.csv","WMB.csv","MPC.csv","NBL.csv","COG.cs
v","HES.csv","CXO.csv","NOV.csv","MRO.csv","FTI.csv","XEC.csv","OKE.csv","EQT.csv","RRC.csv","NFX.csv","
MUR.csv","CHK.csv","SWN.csv","RIG.csv","HP.csv")

rnn_model = "elman"
b_sent = TRUE
```

```
for (file_name in file_names){

  cols = get_col_list(file_name, b_sent)

  test_results=data.frame(SIZE = -1 , ITERATIONS  = -1 , DECAY = -1, MSE = -1, RMSE = -1)
  train_results=data.frame(SIZE = -1 , ITERATIONS  = -1 , DECAY = -1, MSE = -1, RMSE = -1)
  input_dir = paste(getwd(), "/data/",index_name,"/processed/first_cut/",sep="")
  time_windows_input_dir = paste(getwd(), "/data/", index_name,
"/processed/time.windows/",number_sliding_windows,sep="")
  data_df = load_from_csv(input_dir, file_name,TRUE )
  time_windows_df = load_from_csv(paste(time_windows_input_dir, "/",sep=""), file_name,TRUE)

  #generate the daily return at t
  close = data_df$Close
  close_m1 = shift(close, -1)
  data_df$return =  log(close/close_m1)
  return = as.data.frame(data_df$return)[2:length(data_df$return),]

  #generate the EGCARCH vol
  gspec.ru <-ugarchspec(# variance equation
    variance.model=list(model="eGARCH",garchOrder=c(1,1)),
    # eGARCH would stand for exponential GARCH model
    # mean equation
    mean.model=list(armaOrder=c(0,0),include.mean=F),
    # assumed distribution of errors
    distribution.model="norm")

  gfit.ru <- ugarchfit(gspec.ru, return)

  #generate the volatility lags
  data_df = data_df[2:dim(data_df)[1],]
  data_df$pres_vol = gfit.ru@fit$sigma^2
  data_df$m_1_vol =  shift(data_df$pres_vol,-1)
  data_df$m_2_vol =  shift(data_df$pres_vol,-2)
  data_df$m_3_vol =  shift(data_df$pres_vol,-3)
  #data_df$Volatility =  data_df$Volatility
  data_df$Volatility = return^2
  data_df$m_1_Volatility =  shift(data_df$Volatility,-1)

  #generate the volume lags
  data_df$m_1_volume =  shift(data_df$Volume,-1)
  data_df$m_2_volume =  shift(data_df$Volume,-2)
  data_df$m_3_volume =  shift(data_df$Volume,-3)

  data_df = data_df[5:dim(data_df)[1],]

  train_rmse_acc = 0.0
  test_rmse_acc = 0.0
  time_windows_seq = seq(dim(time_windows_df)[1])
  for (tw_index in time_windows_seq)
  {
    print (paste("file_name", file_name, " - tw_index ", tw_index, sep=""))

    training_range =
    time_windows_df[tw_index,"training_start_index"]:time_windows_df[tw_index,"training_end_index"]
    training_data = data_df[training_range,]

   validation_range =
   time_windows_df[tw_index,"validation_start_index"]:time_windows_df[tw_index,"validation_end_index"]
    validation_data = data_df[validation_range,]

   #impute missing data
    training_data = pre_process(df=training_data,method = "medianImpute")
    validation_data = pre_process(df=validation_data,method = "medianImpute")
```

```
training_data = training_data[,c("Volatility",
                          "m_1_vol", "m_2_vol", "m_3_vol",
                          "m_1_volume", "m_2_volume", "m_3_volume",
                          "Article_sentiment",
                          "Article_sentiment_1day_lag", "Article_sentiment_2day_lag","Article_sentiment_3day_lag",
                          "Article_sentiment_momentum", "Article_sentiment_momentum_minus_1_day",
                          "Article_sentiment_momentum_minus_2_day","Article_sentiment_momentum_minus_3_day",
                          "Atr_tr", "Volume")]

validation_data = validation_data[,c("Volatility",
                           "m_1_vol", "m_2_vol", "m_3_vol",
                           "m_1_volume", "m_2_volume", "m_3_volume",
                           "Article_sentiment",
                           "Article_sentiment_1day_lag", "Article_sentiment_2day_lag","Article_sentiment_3day_lag",
                           "Article_sentiment_momentum", "Article_sentiment_momentum_minus_1_day",
                           "Article_sentiment_momentum_minus_2_day",
                           "Article_sentiment_momentum_minus_3_day",
                           "Atr_tr", "Volume")]

training_data = normalise(training_data)
validation_data = validation_data

the_size = c(5,7,10,15,20)
the_iterations = c(100,500,1000,1500,2000)
the_decay=c(0,0.001,0.01,0.1,1)
best_mse =  1000000.0
best_size = the_size[1]
best_iterations = the_iterations[1]
best_decay =  the_decay[1]

print ("do the training/validation/optmisation phase....")
for (size in the_size)
{
 for (iterations in the_iterations)
 {
  for (decay in the_decay)
  {
   print (paste ("tw_index", tw_index, "- size:", size, "- iterations:", iterations,"- decay: ", decay, paste=""))
   set.seed(1)
   rnn.fit = NULL
   if (rnn_model == "jordan"){
    rnn.fit = tryCatch( jordan (  x=training_data[,cols],
                           y=training_data[,c("Volatility")],
                           size =c(size),
                           learnFuncParams =c(decay),
                           maxit =iterations,
                           metric='RSME',linOut=FALSE),
                 error=function(e) next)
   }else{
    rnn.fit = tryCatch( elman ( x=training_data[,cols],
                         y=training_data[,c("Volatility")],
                         size =c(size),
                         learnFuncParams =c(decay),
                         maxit =iterations,
                         metric='RSME',linOut=FALSE),
                 error=function(e) next)
   }

   if (is.null(rnn.fit) == FALSE){
    #generate the prediction
    pred = predict (rnn.fit, validation_data[,cols])
    t.mse = (1/nrow(validation_data))*sum((validation_data[,c("Volatility")] - pred)^2)
    print(paste("t.mse", t.mse, spe=""))
   }

   print(paste("current best_mse", best_mse, sep=""))
```

```
        if (is.na(t.mse) == FALSE){
          if (t.mse < best_mse){
            best_mse = t.mse
            best_size = size
            best_iterations = iterations
            best_decay =  decay
            print(paste("best_mse", best_mse, sep=""))

          }
        }
       }
      }
    }

    train_results[(dim(train_results)[1]+1),] = c(SIZE = best_size , ITERATIONS  = best_iterations , DECAY =
best_decay, MSE = best_mse, RMSE = sqrt(best_mse),row.names=T)

    train_rmse_acc = train_rmse_acc + best_mse

    print ("do the test phase now....")
    training_range =
    time_windows_df[tw_index,"training_start_index"]:time_windows_df[tw_index,"validation_end_index"]
    training_data = data_df[training_range,]

    test_range = time_windows_df[tw_index,"test_start_index"]:time_windows_df[tw_index,"test_end_index"]
    test_data = data_df[test_range,]

   #impute missing data
    training_data = pre_process(df=training_data,method = "medianImpute")
    test_data = pre_process(df=test_data,method = "medianImpute")

    set.seed(1)
    if (rnn_model == "jordan"){
      rnn.fit = tryCatch( jordan (  x=normalise(training_data[,cols]),
                                    y=training_data[,c("Volatility")],
                                    size =c(best_size),
                                    learnFuncParams =c(best_decay),
                                    maxit =best_iterations,
                                    metric='RSME',linOut=FALSE),
                          error=function(e) next)
    }else{
      rnn.fit = tryCatch( elman ( x=normalise(training_data[,cols]),
                                  y=training_data[,c("Volatility")],
                                  size =c(best_size),
                                  learnFuncParams =c(best_decay),
                                  maxit =best_iterations,
                                  metric='RSME',linOut=FALSE),
                          error=function(e) next)
    }

    if (is.null(rnn.fit) == FALSE){
      #generate the prediction
      pred = predict (rnn.fit, test_data[,cols])
      t.mse = (1/nrow(test_data))*sum((test_data[,c("Volatility")] - pred)^2)

      test_results[(dim(test_results)[1]+1),] = c(SIZE = best_size , ITERATIONS  = best_iterations , DECAY =
best_decay, MSE = t.mse, RMSE = sqrt(t.mse),row.names=T)
    }

    test_rmse_acc = test_rmse_acc + t.mse

  }
```

```
  train_rmse_avg = train_rmse_acc / number_sliding_windows
  test_rmse_avg = test_rmse_acc / number_sliding_windows

  train_results[(dim(train_results)[1]+1),] = c(SIZE = -1 , ITERATIONS  = -1 , DECAY = -1, MSE = train_rmse_avg,
RMSE = sqrt(train_rmse_avg),row.names=T)
  write.table(train_results, file= paste(file_name, rnn_model, "_final_train_", file_name,"_",toString(cols), ".txt",
sep=""), sep=',', row.names=T)

  test_results[(dim(test_results)[1]+1),] = c(SIZE = -1 , ITERATIONS  = -1 , DECAY = -1, MSE = test_rmse_avg,
RMSE = sqrt(test_rmse_avg),row.names=T)
  write.table(test_results, file= paste(file_name, rnn_model, "_final_test_", file_name,"_",toString(cols), ".txt", sep=""),
sep=',', row.names=T)
 }
```

../models/vol_rnn_elman_sent.r