Capstone Project

Juan Salvador Huertas Romero

Machine Learning Engineer

March 23th, 2017

Nanodegree

Definition

Project Overview

Build a Stock Tendency Price Predictor

Prediction of stock trend has long been an intriguing topic and is extensively studied by researchers from different fields.

Machine learning, a well-established algorithm in a wide range of applications, has been extensively studied for its potentials in prediction of financial markets. Popular algorithms, including support vector machine (SVM) and reinforcement learning, have been reported to be quite effective in tracing the stock market and help maximizing the profit of stock option purchase while keep the risk low [1-2]. However, in many of these literatures, the features selected for the inputs to the machine learning algorithms are mostly derived from the data within the same market under concern. Such isolation leaves out important information carried by other entities and make the prediction result more vulnerable to local perturbations.

Efforts have been done to break the boundaries by incorporating external information through fresh financial news or personal internet posts such as Twitter. These approaches, known as sentiment analysis, replies on the attitudes of several key figures or successful analysts in the markets to interpolate the minds of general investors. Despite its success in some occasions, sentiment analysis may fail when some of the people are biased, or positive opinions follow past good performance instead of suggesting promising future markets.

In this project, we propose the use of global stock data in associate with data of other financial products as the input features to machine learning algorithms such as SVM. In particular, we are interested in the correlation between the closing prices of the markets that stop trading right before or at the beginning of US markets. As the connections between worldwide economies are tightened by globalization, external perturbations to the financial markets are no longer domestic. It is to our belief that data of oversea stock and other financial markets, especially those having strong temporal correlation with the upcoming US trading day, should be useful to machine learning based predictor, and our speculation is verified by numerical results.

The volatile nature of the stock market makes it difficult to apply simple time-series or regression techniques. Financial institutions and traders have created various proprietary models to try and beat the market for themselves or their clients, but rarely has anyone achieved consistently higher-than-average returns on investment. Nevertheless, the challenge of stock forecasting is so appealing because an improvement of just a few percentage points can increase profit by millions of dollars for these institutions.

Traditionally, many prediction models have focused on linear statistical time series models such as ARIMA [3]. However, the variance underlying the movement of stocks and other assets makes linear

techniques suboptimal, and non-linear models like ARCH tend to have lower predictive error [4]. Recently, researchers have turned to techniques in the computer science fields of big data and machine learning for stock price forecasting. These apply computational power to extend theories in mathematics and statistics. Machine learning algorithms use given data to "figure out" the solution to a given problem. Big data and machine learning techniques are also the basis for algorithmic and high-frequency trading routines used by financial institutions.

Much economic research has been conducted into the Efficient Markets Hypothesis theory, which posits that stock prices already reflect all available information [5] and are therefore unpredictable.

According to the EMH, stock prices will only respond to new information and so will follow a random walk. If they only respond to new information, they cannot be predicted. That the stocks follow a random walk is actually a sign of market efficiency, since predictable movement would mean that information was not being reflected by the market prices.

There are three variants of this theory – weak, semi-strong, and strong. Most research has concluded that the semi-strong version holds true. This version claims that stock prices reflect all publicly available information, but private information can be used to unfairly predict profits. This is the basis behind strong insider trading laws.

Nevertheless, there are certain market phenomena that actually run contrary to EMH. These are known as market anomalies. Jegadeesh and Titman discovered that in the short term, stock prices tend to exhibit momentum [6]. Stocks that have recently been increasing continue to increase, and recently decreasing stocks continue to decrease. This type of trend implies some amount of predictability to future stock prices, contradicting the EMH.

The stock market also exhibits seasonal trends. Jacobsen and Zhang studied centuries' worth of data and found that trading strategies can exploit trends in high winter returns and low summer returns to beat the market [7][8].

If the EMH held perfectly true, then the direction of future stock prices could not be predicted with greater than 50% accuracy. That is, one should not be able to guess whether future prices will go up or down better than simple random guessing. However, the studies discussed in 2.3 are all able to predict price direction with greater than 50% accuracy, implying that machine learning techniques are able to take advantage of momentum and other price trends to forecast price direction.

Economic conditions greatly deteriorated in the Great Recession in late 2007. Unemployment increased drastically, making the Great Recession the worst "labour market downturn since the Great Depression" [9].

The S&P500 index dropped 38.49% in 2008 and then increased by 23.45% the next year [10]. In every year except 2011 the index had a double-digit increase in price.

We will use the S&P 500 Index (SPY) as the general index. The Standard & Poor's 500 Index (S&P 500) is an index of 500 stocks seen as a leading indicator of U.S. equities and a reflection of the performance of the large cap universe, made up of companies selected by economists. The S&P 500 is a market value weighted index and one of the common benchmarks for the U.S. stock market; other S&P indexes include small cap companies with market capitalization between $300 million and $2 billion, and an index of mid-cap companies

To make the list, a company must, among other things:

- Have a market capitalization of at least $5.3 billion
- Have a liquid stock -- a quarter of a million of its shares must change hands each month
- Be based in the United States
- And be "financially viable" -- meaning that it must generally earn a profit over the preceding 12 months

Globalization deepens the interaction between the financial markets around the world. Shock wave of US financial crisis hit the economy of almost every country and debt crisis originated in Greece brought down all major stock indices.

Nowadays, no financial market is isolated. Economic data, political perturbation and any other oversea affairs could cause dramatic fluctuation in domestic markets. Therefore, in this project, we propose to use world major stock indices as input features for our machine learning based predictor. In particular, the oversea markets that closes right before or at the beginning of the US market trading should provide valuable information on the trend of coming US trading day, as their movements already account for possible market sentiment on latest economic news or response to progress in major world affairs

In addition to stock markets, commodity prices and foreign currency data are also listed as potential features, as different financial markets are interconnected. For instance, slowdown in US economy will definitely cause a drop in US stock market. But at the same time, USD and JPY will increase with respect to its peers as people seek for asset havens. Such interplay implies the underlying relationship between these financial products and the possibility of using one or some of them to predict the move of the other ones.

Stock price and index data was obtained from Yahoo Finance[1] and Quandl Finance[2]. We use the daily closing prices.

Problem Statement

The aim of the project is to predict whether future daily returns of the index S&P 500 are going to be positive or negative.

Thus the problem I'm facing is a binary classification

The data set used in this project is collected from Yahoo Finance and Quandl Finance. It contains 19 sources as listed in Table 1 and covers daily price from 01-Jan-2003 to 01- Jan -2017: Since the markets are closed on holidays which vary from country to country, we use the own Index S&P 500 as a basis for data alignment and missing data in other data sources is replaced by coping the price of the day before.

[1] https://finance.yahoo.com/
[2] https://www.quandl.com/

TABLE 1	DATA SOURCE
ETF	SPY
Currency	EUR, AUD, JPY, GBP
Index	SP500, NASDAQ, DJIA, Nikkei 225, Hang Seng Index, FTSE100, DAX, ASX, WT1010, Paris CAC
Commodity	Gold, Silver, Platinum, Oil

Together with the algorithm, the most important question to be answered is defining the set of inputs or features to feed the model. Its turns out that it are very hard to define a priori a good set of features.

Technical analysis and fundamental analysis are the two main schools of thought when it comes to analyzing the financial markets. Technical analysis looks at the price movement of a security and uses this data to predict future price movements. Fundamental analysis instead looks at economic and financial factors that influence a business. For this project we are going to use indicators which are usually use in technical analysis, as Technical analysts believe that there's no reason to analyze a company's financial statements since the stock price already includes all relevant information. Instead, the analyst focuses on analyzing the stock chart itself for hints into where the price may be headed.

Definition of 'Intraday Return' or 'Daily Return'

One of the most popular technical indicators is the Daily Return, which measures the return generated by a stock during regular trading hours, based on its price change from the opening of a trading day to its close. Intraday return and overnight return together constitute the total daily return from a stock, which is based on the price change of a stock from the close of one trading day to the close of the next trading day. Also called daytime return.

Among the most popular technical indicators, moving averages are used to gauge the direction of the current trend. Every type of moving average, is a mathematical result that is calculated by averaging a number of past data points. Once determined, the resulting average is then plotted onto a chart in order to allow traders to look at smoothed data rather than focusing on the day-to-day price fluctuations that are inherent in all financial markets.

We will play as inputs with four technical indicators as parameters to feed the model:

Daily return Moving Average

This is a widely used indicator in technical analysis that helps smooth out price action by filtering out the "noise" from random price fluctuations. A moving average (MA) is a trend-following or lagging indicator because it is based on past prices.

Daily return Exponential Moving Average

An exponential moving average (EMA) is a type of moving average that is similar to a simple moving average, except that more weight is given to the latest data. It's also known as the exponentially weighted moving average. This type of moving average reacts faster to recent price changes than a simple moving average.

Momentum

Momentum is the rate of acceleration of a security's price or volume. In technical analysis, momentum is considered an oscillator and is used to help identify trend lines.

In general, momentum refers to the force or speed of movement; it is usually defined as a rate. In the world of investments, momentum refers to the rate of change on price movements for a particular asset – that is, the speed at which the price is changing. Studies have found that mutual fund inflows are positively correlated with market returns. Momentum plays a part in the decision to invest and when more people invest, the market goes up, encouraging even more people to buy. It's a positive feedback loop.

Volatility

Volatility is a statistical measure of the dispersion of returns for a given security or market index. Volatility can either be measured by using the standard deviation or variance between returns from that same security or market index. It is commonly understood that, the higher the volatility, the riskier the security.

We will use also moving averages for Momentum and Volatility, as this way we will not use a different view of understand the tendency of the market

My Hypothesis is that using moving averages of these technical indicators we can predict the trend of the index S&P500, if the trend is increase or the trend is decrease. Using moving averages we mitigate the natural oscillation of markets and the peaks that prices or rates can have in between days. Choosing an adequate window for the rolling average we can attenuate even more the peaks and discover if for example Crude Oil trends have any influence on S&P index trend.

Metrics

Accuracy: Classification accuracy is the number of correct predictions made as a ratio of all predictions made.

Precision: The precision is the ratio tp / (tp + fp) where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Recall: The recall is the ratio tp / (tp + fn) where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

F1_score: The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0

Support: The support is the number of occurrences of each class in y_true

Matthews's correlation coefficient

The Matthews correlation coefficient is used in machine learning as a measure of the quality of binary (two-class) classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 an average random prediction and -1 an inverse prediction. The statistic is also known as the phi coefficient."

If tp, tn, $fp$and $fn$are respectively the number of true positives, true negatives, false positives and false negatives, the MCC coefficient is defined as

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}.$$

Roc_auc_score. Receiver operating characteristic (ROC)

A receiver operating characteristic (ROC), or simply ROC curve, is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting the fraction of true positives out of the positives (TPR = true positive rate) vs. the fraction of false positives out of the negatives (FPR = false positive rate), at various threshold settings. TPR is also known as sensitivity, and FPR is one minus the specificity or true negative rate

The roc_auc_score function computes the area under the receiver operating characteristic (ROC) curve, which is also denoted by AUC or AUROC. By computing the area under the roc curve, the curve information is summarized in one number

Analysis

Data Exploration:

The data set used in this project is collected from Yahoo Finance and Quandl Finance. It contains 19 sources as listed in Table 1 and covers daily price from 01-Jan-2003 to 01- Jan -2017: Since the markets are closed on holidays which vary from country to country, we use the own Index S&P 500 as a basis for data alignment and missing data in other data sources is replaced by coping the price of the day before.

TABLE 1	DATA SOURCE
ETF	SPY
Currency	EUR, AUD, JPY, GBP
Index	SP500, NASDAQ, DJIA, Nikkei 225, Hang Seng Index, FTSE100, DAX, ASX, WT1010, Paris CAC
Commodity	Gold, Silver, Platinum, Oil

S&P 500 (^GSPC Yahoo Finance)

SPDR S&P 500 ETF (SPY Yahoo Finance)

NASDAQ Composite (^IXIC Yahoo Finance)

Dow Jones Industrial Average (^DJI Yahoo Finance)

Frankfurt DAX (^GDAXI Yahoo Finance)

London FTSE-100 (^FTSE Yahoo Finance)

Paris CAC 40 (^FCHI Yahoo Finance)

Tokyo Nikkei-225 (^N225 Yahoo Finance)

Hong Kong Hang Seng (^HSI Yahoo Finance)

Australia ASX-200 (^AXJO Yahoo Finance)

Daily yield of the current 10 year German Federal bond (BUNDESBANK/BBK01_WT1010)

Commodities = ["LBMA/SILVER", "LBMA/GOLD", "JOHNMATT/PLAT","OPEC/ORB"]

Currencies = ["CUR/EUR","CUR/AUD","CUR/GBP","CUR/JPY"]

Yahoo information files are Time series which contains the following columns: Date, Open, High, Low, Close, Volume, Adj Close. And we will take the Adj Close Price

An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time prior to the next day's open. The adjusted closing price is often used when examining historical returns or performing a detailed analysis on historical returns.

Each market has their own closing time, but clearly we can differentiate in between those stock exchanges which are still opens when NYSE opens and which not.

BUNDESBANK/BBK01_WT1010 à The Daily yield of the current 10 year German Federal bond, is fixed every day and available at closing time of Frankfurt DAX. Generally, a government bond is issued by a national government and is denominated in the country`s own currency. Bonds issued by national governments in foreign currencies are normally referred to as sovereign bonds. The yield required by investors to loan funds to governments reflects inflation expectations and the likelihood that the debt will be repaid

Currency rates are downloaded from Quandl Finance and they are also Time series which contains the closing day exchange rate of the currency versus Dollar.

The forex is the largest market in the world and is considered a 24-hour market because currencies are traded around the world in various markets, providing traders with the constant ability to trade currencies. The forex opens at 5pm EST on Sunday and runs until 5pm EST on Friday, running 24 hours a day during this time. But between the Friday close and the Sunday open, the forex market does not trade.

The opening prices for the week are the initial trading prices on Sunday and the closing prices for the week are those of the last trade on Friday. However, over the course of the week, there really are no closing prices for the forex as there is at least one market open at some place in the world at all times.

However, we often hear quotes for the opening and closing prices for currency pairs in the financial media. For example, a news article might state how the U.S. dollar closed down against the Canadian dollar during trading on Wednesday. The price being quoted is the closing price for an individual market within the forex market. There are three main regions - North America, Asia and Europe - and within each there are several forex markets. In North America, the main market is in New York, in Asia it is in Tokyo and in Europe it is in London. There are many other individual markets within these regions that are part of the forex market, and each individual market has an open and close (i.e. does not trade 24 hours a day). The New York market, for example, trades from 8am EST until 3pm EST. In North American media, the closing price will often refer to the closing price of the New York forex market.

Clearly in our project AUD and JPY are in Asia region and EUR and GBP in Europe Region. Asia region will be close at the time NYSE opens

Commodities Prices are downloaded as well from Quandl, and here each commodity has its own structure and particular closing time

Gold (LBMA/GOLD): The gold, silver, platinum and palladium price auctions take place in London on a daily basis. All of these prices are internationally regarded as the pricing mechanism for a variety of precious metal transactions and products The LBMA Gold price auction takes place twice daily by ICE Benchmark Administration (IBA) at 10:30 and 15:00 with the price set in US dollars per fine troy ounce. We use USD (AM) price in our project

Silver (LBMA/SILVER): The LBMA Silver price auction is operated by CME and administered by Thomson Reuters. The price is set daily in US dollars per ounce at 12:00 noon London time and is displayed on the LBMA's website with a 15 minute delay

Platinum (JOHNMATT/PLAT): All prices are in US$ per troy oz. The Johnson Matthey Base Prices are the company`s quoted selling prices for wholesale quantities of platinum group metals set by our trading desks in the USA, Hong Kong and London, based on market offer prices. We are taking for our project price of London desk 8:00 AM

Crude Oil (OPEC/ORB): Reference Price for the OPEC Crude Oil Basket. Currently includes: Saharan Blend (Algeria), Girassol (Angola), Oriente (Ecuador), Iran Heavy (Islamic Republic of Iran), Basra Light (Iraq), Kuwait Export (Kuwait), Es Sider (Libya), Bonny Light (Nigeria), Qatar Marine (Qatar), Arab Light (Saudi Arabia), Murban (UAE) and Merey (Venezuela).

On the file prices.xlsx we have an example of the original Dataframe, whit the prices use on the project

df_index.shape

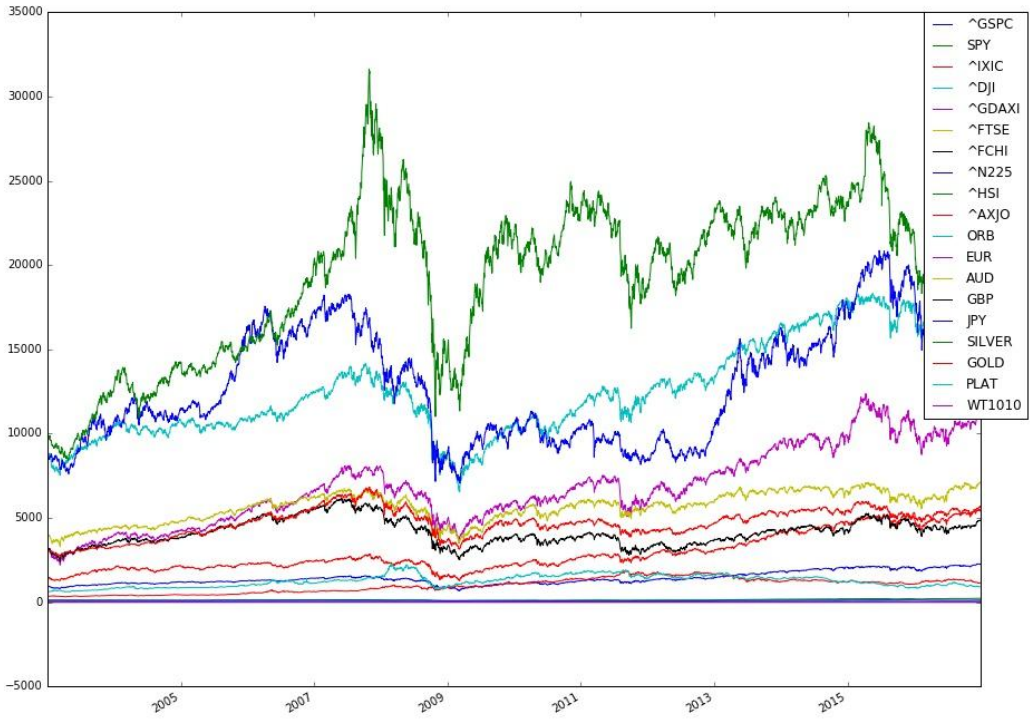
(3525, 19)

df_index.head()

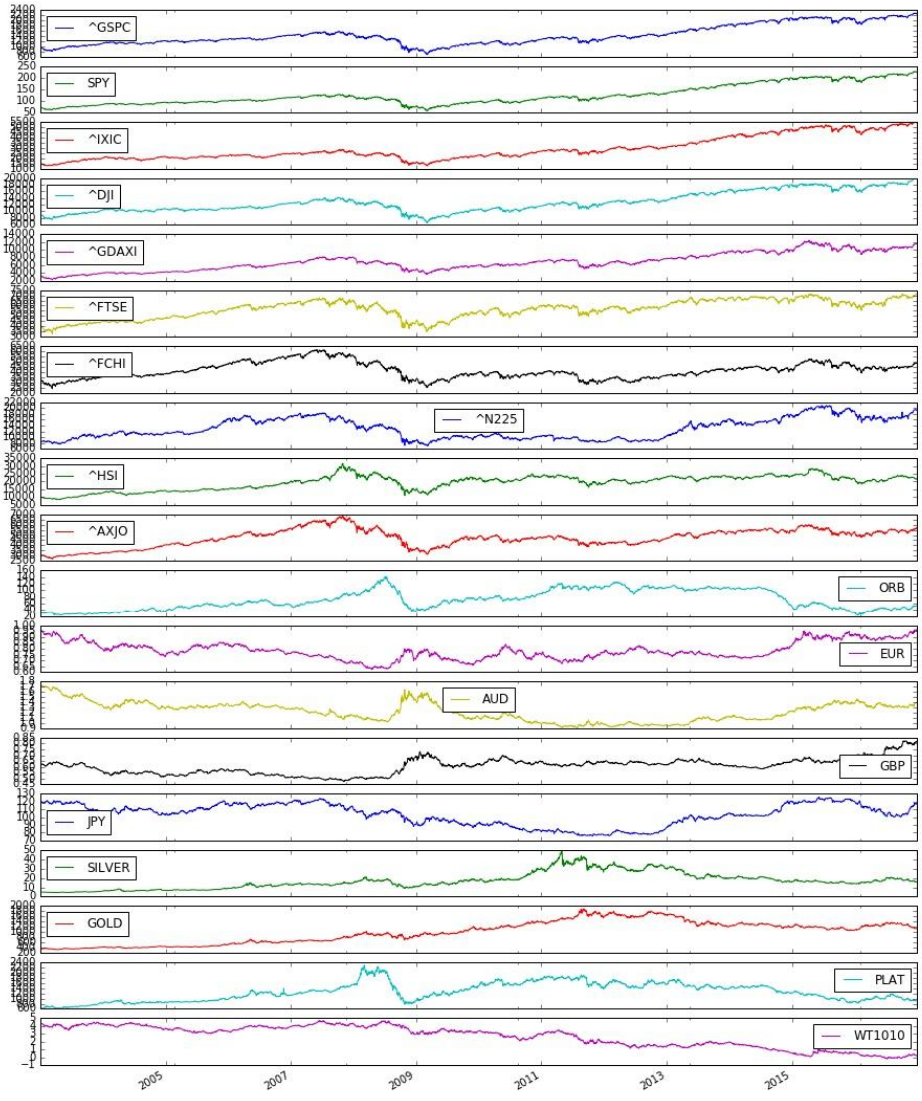
	^GSPC	SPY	^IXIC	^DJI	^GDAXI	^FTSE	^FCHI	^N225	^HSI	^AXJO	ORB	EUR	AUD	GBP	JPY	SILVER	GOLD	PLAT	WT1010
2003-01-02 00:00:00	909,03	68,60	1384,85	8607,52	3105,04	4009,50	3195,02	8713,33	9365,52	3027,60	30,05	0,96	1,78	0,63	119,60	4,74	342,20	602,00	4,22
2003-01-03 00:00:00	908,59	68,81	1387,08	8601,69	3092,94	4005,00	3187,88	8713,33	9583,85	3055,50	30,83	0,96	1,77	0,62	119,83	4,79	344,00	608,00	4,31
2003-01-06 00:00:00	929,01	70,02	1421,32	8773,57	3157,25	4001,40	3210,27	8713,33	9665,96	3075,40	30,71	0,95	1,74	0,62	118,89	4,91	356,10	608,00	4,35
2003-01-07 00:00:00	922,93	69,85	1431,57	8740,59	3112,77	3957,40	3160,99	8656,50	9652,40	3074,50	29,72	0,96	1,74	0,62	120,12	4,84	348,70	609,00	4,31
2003-01-08 00:00:00	909,93	68,84	1401,07	8595,31	2993,00	3924,80	3094,09	8517,80	9688,21	3074,70	28,86	0,96	1,74	0,62	119,62	4,78	346,75	607,00	4,29

df_index.mean()	df_index.min()	df_index.max()	df_index.std()
SPY 124.247170	^GSPC 676.530029	^GSPC 2271.719971	^GSPC 377.738299
^IXIC 2887.485241	SPY 57.440272	SPY 225.444643	SPY 42.716304
^DJI 12676.749688	^IXIC 1268.640015	^IXIC 5487.439941	^IXIC 1111.738679
^GDAXI 6792.568694	^DJI 6547.049805	^DJI 19974.619141	^DJI 3019.752104
^FTSE 5650.562809	^GDAXI 2202.959961	^GDAXI 12374.730469	^GDAXI 2287.877716
^FCHI 4129.860792	^FTSE 3287.000000	^FTSE 7142.799805	^FTSE 879.682560
^N225 12943.241037	^FCHI 2403.040039	^FCHI 6168.149902	^FCHI 771.859724
^HSI 19393.224227	^N225 7054.979980	^N225 20868.029297	^N225 3481.737428
^AXJO 4716.276737	^HSI 8409.009766	^HSI 31638.220703	^HSI 4634.787880
ORB 70.492400	^AXJO 2700.399902	^AXJO 6828.700195	^AXJO 855.984222
EUR 0.784911	ORB 22.480000	ORB 140.730000	ORB 29.223803
AUD 1.218466	EUR 0.625360	EUR 0.963904	EUR 0.073974
GBP 0.605921	AUD 0.906698	AUD 1.776110	AUD 0.180926
JPY 103.269039	GBP 0.474330	GBP 0.822470	GBP 0.066218
SILVER 16.828950	JPY 75.751979	JPY 125.630900	JPY 13.772426
GOLD 993.842156	SILVER 4.370000	SILVER 48.700000	SILVER 8.787288
PLAT 1251.970780	GOLD 319.750000	GOLD 1891.000000	GOLD 427.827158
WT1010 2.667748	PLAT 602.000000	PLAT 2280.000000	PLAT 349.284512
dtype: float64	WT1010 -0.200000	WT1010 4.690000	WT1010 1.383530
	dtype: float64	dtype: float64	dtype: float64

Graph 1 Series without Normalization



Graph 2 Series no Normalized Individual



As here we are mixing indexes, with prices of ETFs, with currency rates and with prices of commodities, we need to normalize the inputs to be able to compare them and this way no one will have more weight than others

The method I use is: I get the maximum and the minimum per each time series and I normalize each individual entry applying the formula

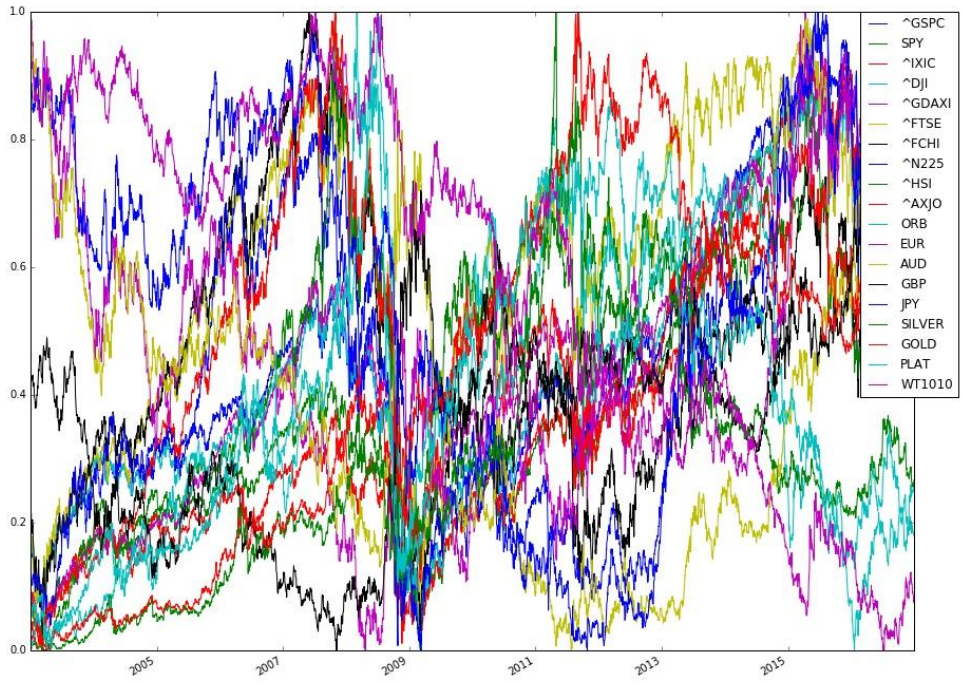
(X- min_value) / Max_value – min_value

All individual entries are normalizes to values on the range -1 to 1

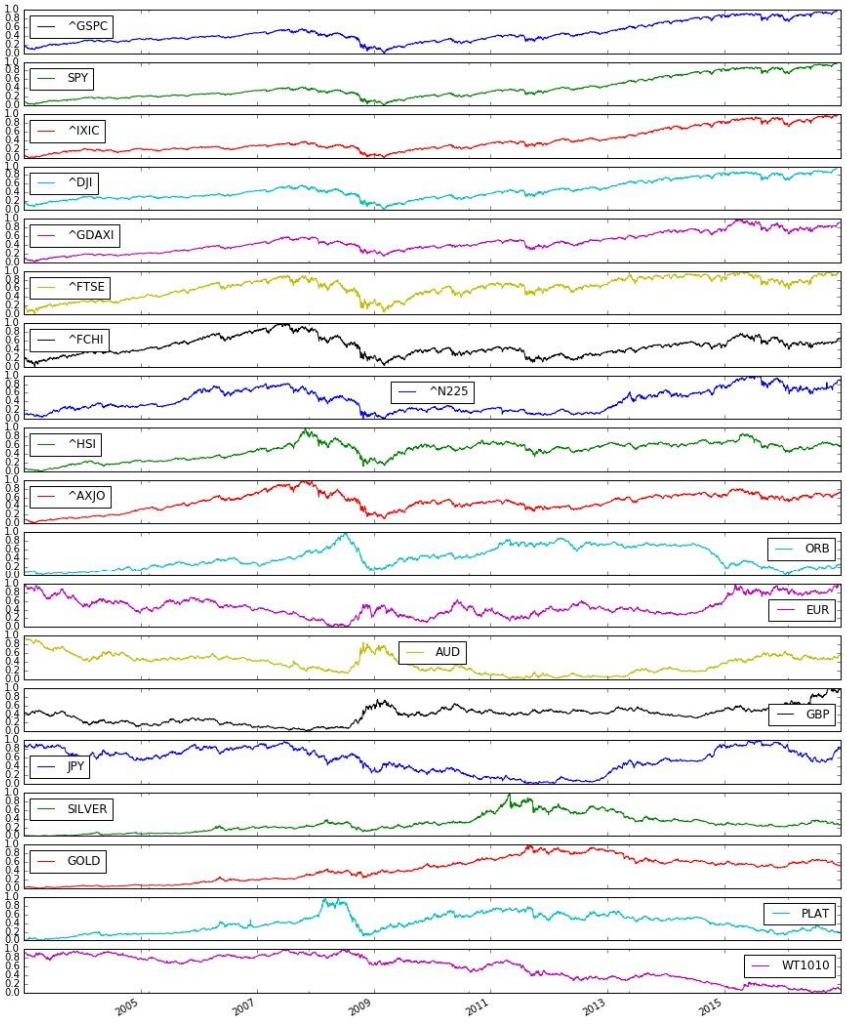
On the file prices_normalized.xlsx, we have an the Dataframe transferred to excel with prices normalized according to the explained formula

	^GSPC	SPY	^IXIC	^DJI	^GDAXI	^FTSE	^FCHI	^N225	^HSI
2003-01-02 00:00:00	0,146	0,066	0,028	0,153	0,089	0,187	0,210	0,120	0,0
2003-01-03 00:00:00	0,145	0,068	0,028	0,153	0,087	0,186	0,208	0,120	0,0
2003-01-06 00:00:00	0,158	0,075	0,036	0,166	0,094	0,185	0,214	0,120	0,0
2003-01-07 00:00:00	0,154	0,074	0,039	0,163	0,089	0,174	0,201	0,116	0,0
2003-01-08 00:00:00	0,146	0,068	0,031	0,153	0,078	0,165	0,184	0,106	0,0

Graph 3 Series Normalized



Graph 4 Series Normalized Individual



This file it will be the base to create the test cases which finally ends up as I explain here

TEST 1: Simple rolling average Series of Daily returns with a window of 5 Days

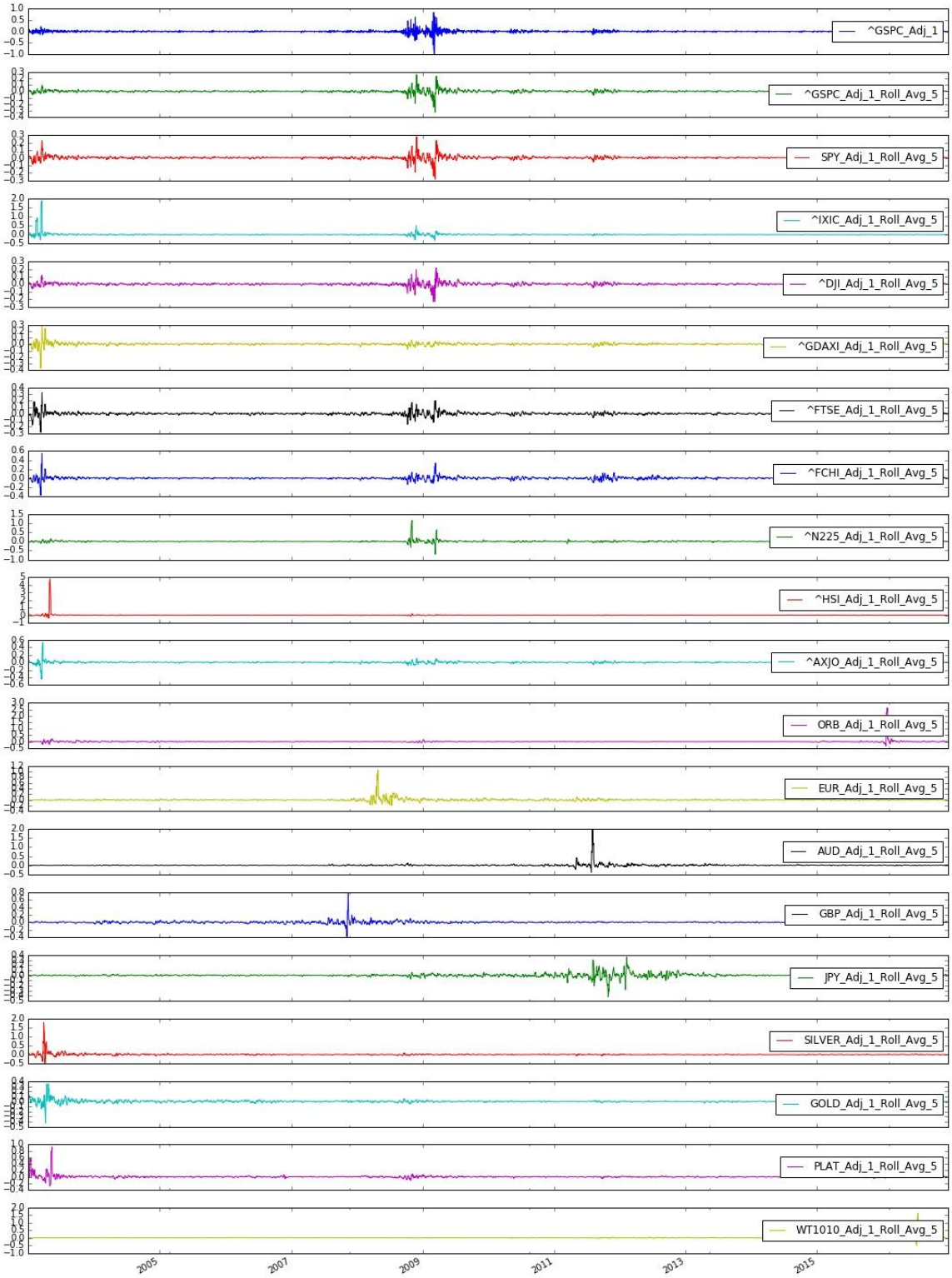
TEST 2: Simple rolling average Series of Momentum with a window of 5 Days

TEST 3: Simple rolling average Series of Volatility with a window of 5 Days

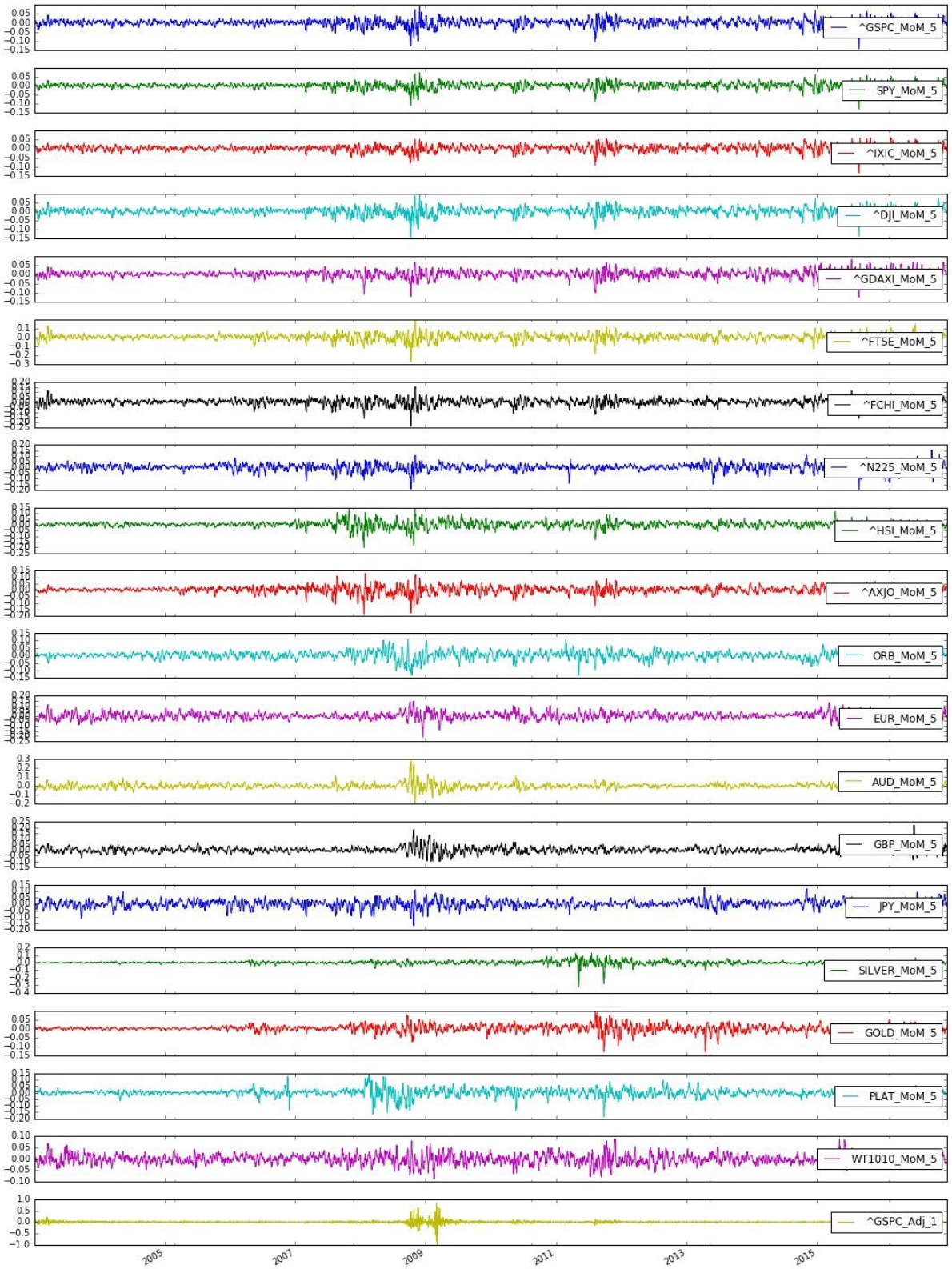
TEST 4: Exponential weighted rolling average Series of Daily returns with a window of 5 Days

TEST 6: Simple rolling average Series of Daily returns with a window of 5 Days + Simple rolling average Series of Momentum with a window of 21 Days + Simple rolling average Series of Volatility with a window of 2 Days+ Exponential weighted rolling average Series of Daily returns with a window of 5 Days. File prices_adjusted_rolling_test6.xlsx

Test 1: Simple rolling average Series of Daily returns with a window of 5 Days
File à prices_adjusted_rolling_test1.xlsx

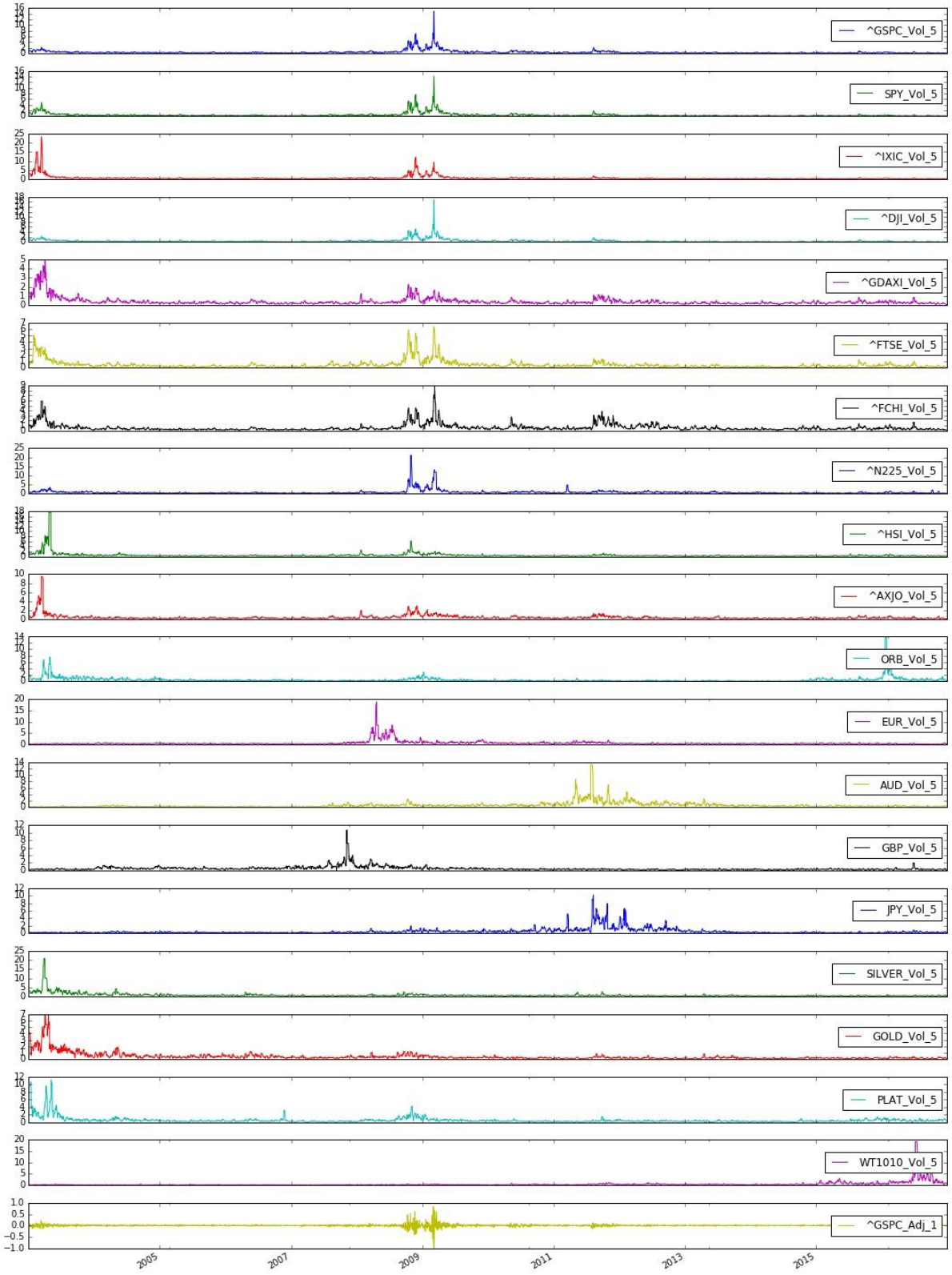


Test 2: Simple rolling average Series of Momentum with a window of 5 Days
File à prices_adjusted_rolling_test2.xlsx



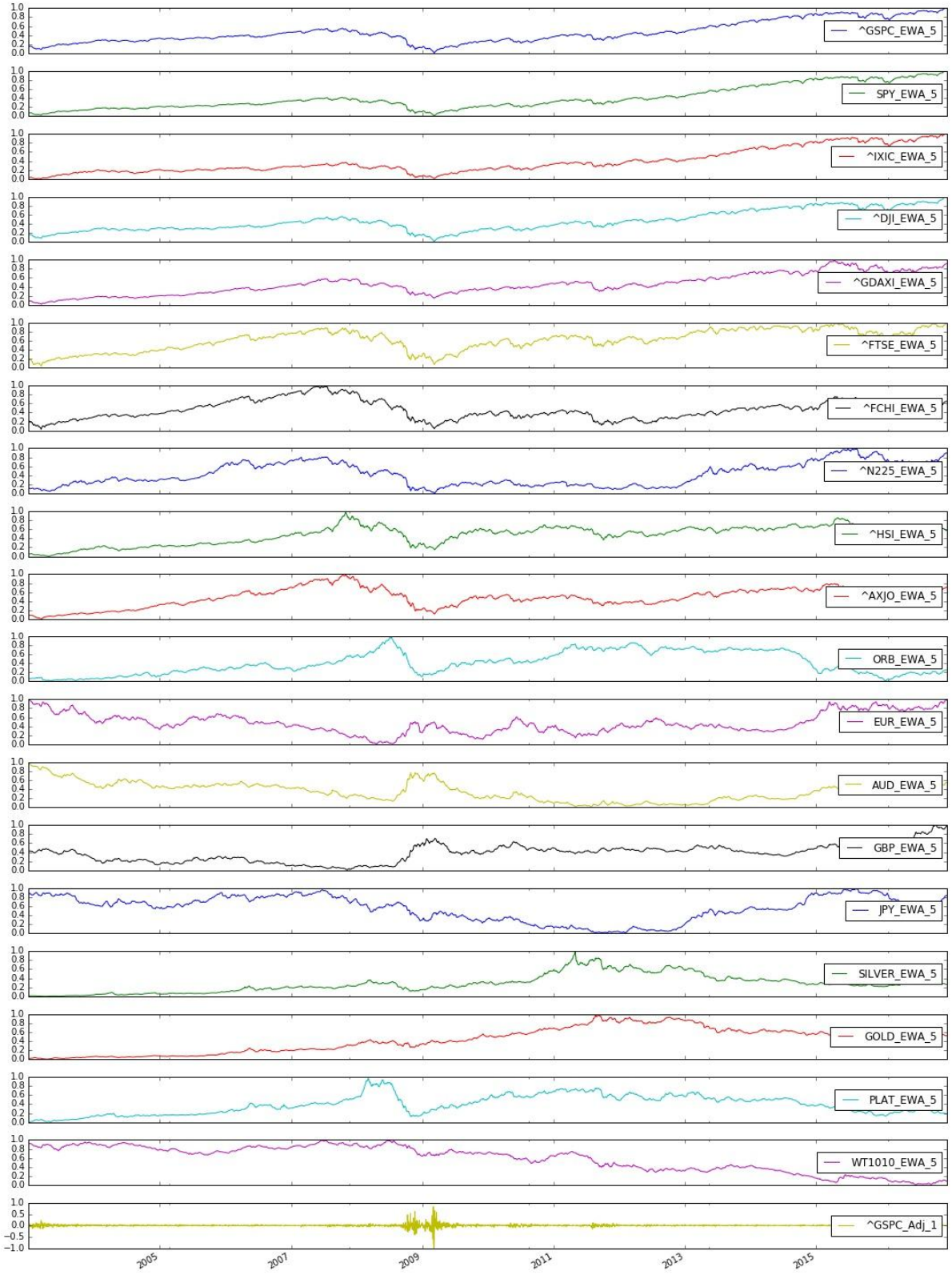
Test 3: Simple rolling average Series of Volatility with a window of 5 Days

File à prices_adjusted_rolling_test3.xlsx



TEST 4: Exponential weighted rolling average Series of Daily returns with a window of 5 Days

File à prices_adjusted_rolling_test4.xlsx



Exploratory Visualization

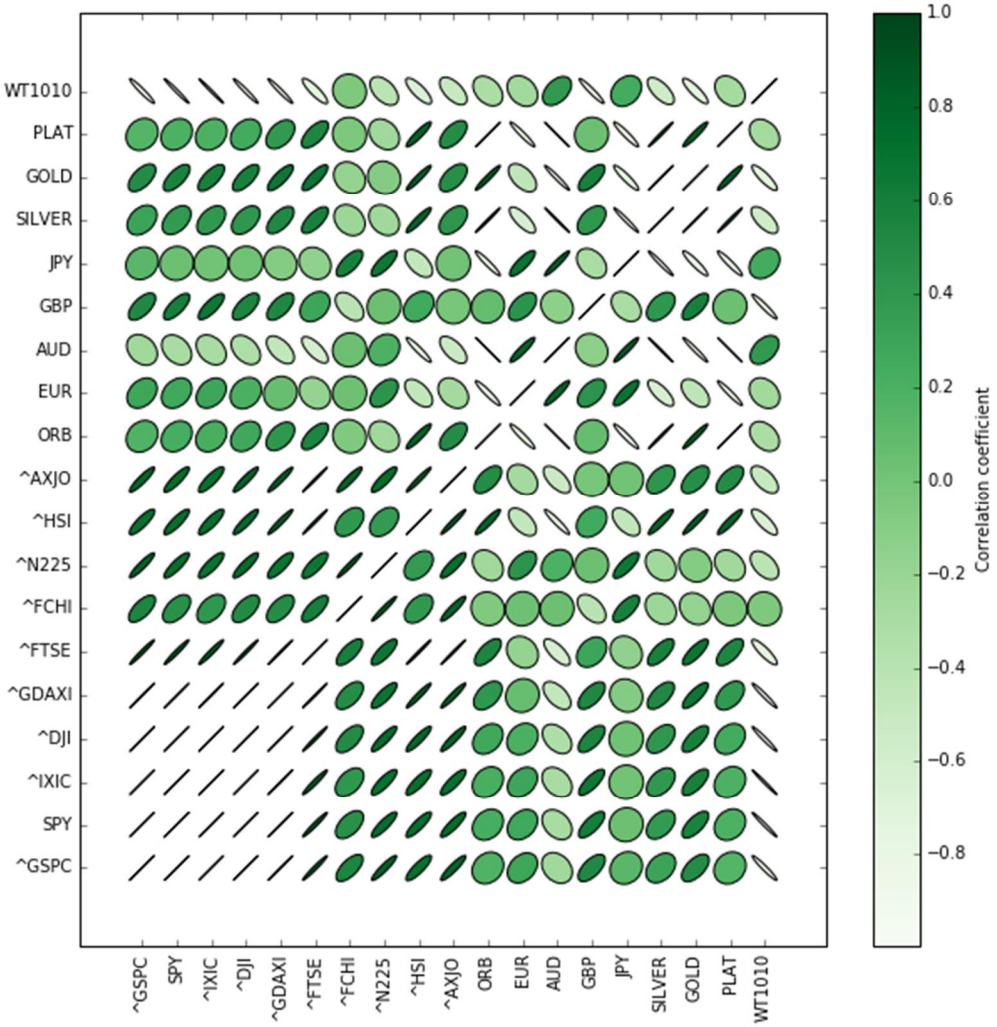
In this project, as we focus on the prediction of the trend of stock market (either increase or decrease). Therefore, the change of a feature over time is more important than the absolute value of each feature. The performance of a stock market predictor heavily depends on the correlation between the data used for training and the current input for prediction. Intuitively, if the trend of stock price is always an extension to yesterday, the accuracy of prediction should be fairly high. To select input features with high temporal correlation, we calculated the autocorrelation and cross-correlation of different market trends (increase or decrease)

Correlation Matrix

	^GSPC	SPY	^IXIC	^DJI	^GDAXI	^FTSE	^FCHI	^N225	^HSI	^AXJO	ORB	EUR	AUD	GBP	JPY	SILVER	GOLD	PLAT	WT1010
^GSPC	1,0000	0,9900	0,9761	0,9932	0,9459	0,8589	0,5137	0,7931	0,6555	0,6971	0,1212	0,3240	-0,1690	0,3541	0,2775	0,2322	0,4282	0,0860	-0,7856
SPY	0,9900	1,0000	0,9944	0,9895	0,9567	0,8300	0,4130	0,7252	0,6635	0,6467	0,1428	0,3332	-0,1882	0,4651	0,1895	0,2949	0,5123	0,1099	-0,8533
^IXIC	0,9761	0,9944	1,0000	0,9762	0,9473	0,7959	0,3521	0,6881	0,6477	0,5938	0,1279	0,3619	-0,1833	0,5142	0,1549	0,3092	0,5328	0,0954	-0,8772
^DJI	0,9932	0,9895	0,9762	1,0000	0,9551	0,8721	0,4711	0,7439	0,6943	0,6937	0,1969	0,2761	-0,2476	0,3750	0,1958	0,3130	0,5033	0,1572	-0,8024
^GDAXI	0,9459	0,9567	0,9473	0,9551	1,0000	0,8997	0,5066	0,7270	0,8181	0,7754	0,2853	0,1616	-0,3213	0,3960	0,1135	0,4005	0,5896	0,2922	-0,7869
^FTSE	0,8589	0,8300	0,7959	0,8721	0,8997	1,0000	0,6807	0,7394	0,8411	0,8971	0,4454	-0,0429	-0,4903	0,1305	0,0948	0,4377	0,5254	0,4390	-0,5860
^FCHI	0,5137	0,4130	0,3521	0,4711	0,5066	0,6807	1,0000	0,8480	0,4855	0,8495	0,0038	-0,0436	-0,0266	-0,4050	0,5811	-0,1562	-0,1738	0,0821	0,0279
^N225	0,7931	0,7252	0,6881	0,7439	0,7270	0,7394	0,8480	1,0000	0,4735	0,7740	-0,1436	0,3079	0,1326	-0,0398	0,6897	-0,1839	-0,0565	-0,1197	-0,3528
^HSI	0,6555	0,6635	0,6477	0,6943	0,8181	0,8411	0,4855	0,4735	1,0000	0,8316	0,6099	-0,3248	-0,6706	0,1370	-0,1841	0,6000	0,6762	0,6895	-0,4965
^AXJO	0,6971	0,6467	0,5938	0,6937	0,7754	0,8971	0,8495	0,7740	0,8316	1,0000	0,3623	-0,2050	-0,3822	-0,1166	0,2369	0,2468	0,3054	0,4294	-0,3093
ORB	0,1212	0,1428	0,1279	0,1969	0,2853	0,4454	0,0038	-0,1436	0,6099	0,3623	1,0000	-0,6806	-0,8945	-0,0801	-0,6161	0,7927	0,7060	0,8893	-0,1548
EUR	0,3240	0,3332	0,3619	0,2761	0,1616	-0,0429	-0,0436	0,3079	-0,3248	-0,2050	-0,6806	1,0000	0,6558	0,4924	0,4933	-0,3731	-0,2080	-0,7060	-0,4017
AUD	-0,1690	-0,1882	-0,1833	-0,2476	-0,3213	-0,4903	-0,0266	0,1326	-0,6706	-0,3822	-0,8945	0,6558	1,0000	0,0466	0,6910	-0,8348	-0,7502	-0,8857	0,2276
GBP	0,3541	0,4651	0,5142	0,3750	0,3960	0,1305	-0,4050	-0,0398	0,1370	-0,1166	-0,0801	0,4924	0,0466	1,0000	-0,2781	0,3111	0,5353	-0,0602	-0,7460
JPY	0,2775	0,1895	0,1549	0,1958	0,1135	0,0948	0,5811	0,6897	-0,1841	0,2369	-0,6161	0,4933	0,6910	-0,2781	1,0000	-0,7535	-0,6809	-0,6169	0,1484
SILVER	0,2322	0,2949	0,3092	0,3130	0,4005	0,4377	-0,1562	-0,1839	0,6000	0,2468	0,7927	-0,3731	-0,8348	0,3111	-0,7535	1,0000	0,9220	0,7899	-0,4489
GOLD	0,4282	0,5123	0,5328	0,5033	0,5896	0,5254	-0,1738	-0,0565	0,6762	0,3054	0,7060	-0,2080	-0,7502	0,5353	-0,6809	0,9220	1,0000	0,6941	-0,7023
PLAT	0,0860	0,1099	0,0954	0,1572	0,2922	0,4390	0,0821	-0,1197	0,6895	0,4294	0,8893	-0,7060	-0,8857	-0,0602	-0,6169	0,7899	0,6941	1,0000	-0,1061
WT1010	-0,7856	-0,8533	-0,8772	-0,8024	-0,7869	-0,5860	0,0279	-0,3528	-0,4965	-0,3093	-0,1548	-0,4017	0,2276	-0,7460	0,1484	-0,4489	-0,7023	-0,1061	1,0000

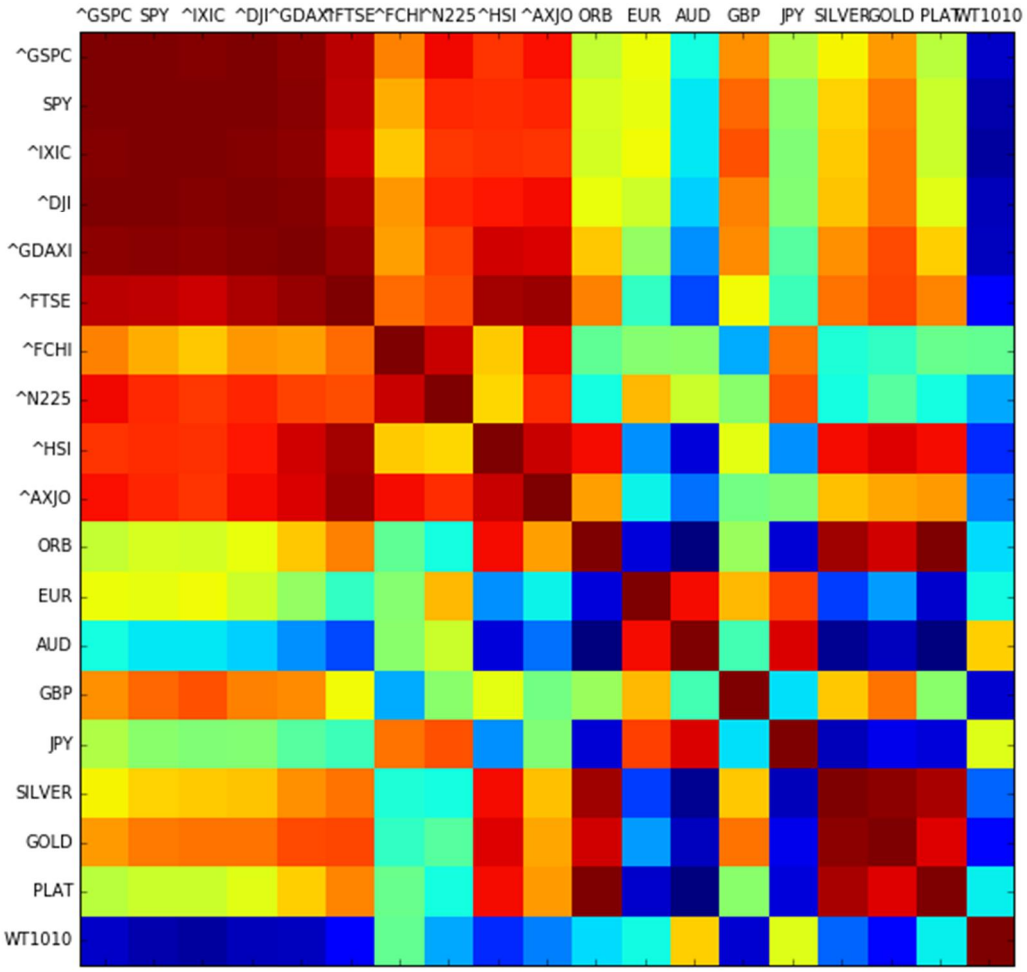
As we could see in our Graph 3 Series Normalized some Series are strongly correlated than others

On the Correlation matrix we are working with adjusted closing prices normalized, to emphasise the correlation in between series I established a threshold of .7 to paint data in red, meaning that is heavy correlated, correlation factor for 0 to 0.7 is painted in blue and negative correlation is painted in green.



In this Correlation Graph we can observe the same. Strong correlated data is represented by dark green narrow ellipses and light correlated data are wide ellipses with light green.

Orientation represents the sign, being positive when the ellipse is left to right, and negative when the ellipse is right to left



In this other model of correlation Graph we observe, obviously the same, but probably is more direct the interpretation.

We can observe how SP 500 is strongly correlated with:

NASDAQ Composite (^IXIC Yahoo Finance), Dow Jones Industrial Average (^DJI Yahoo Finance), Frankfurt DAX (^GDAXI Yahoo Finance), London FTSE-100 (^FTSE Yahoo Finance), and Tokyo Nikkei-225 (^N225 Yahoo Finance)

And medium correlated with:

Paris CAC 40 (^FCHI Yahoo Finance), Hong Kong Hang Seng (^HSI Yahoo Finance), Australia ASX-200 (^AXJO Yahoo Finance), and the Currencies = ["CUR/EUR","CUR/JPY"]

The German Bond Yield Index is not correlated at all, but I keep on the set

Algorithms and Techniques

Text from http://scikit-learn.org

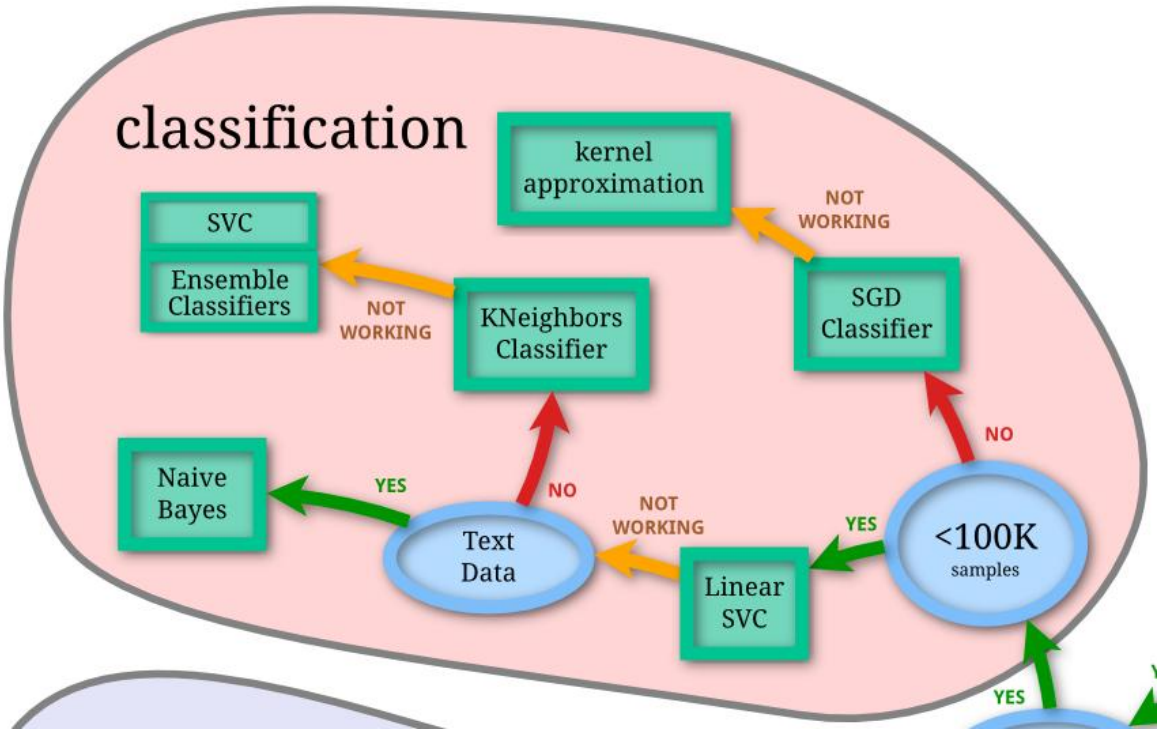
Often the hardest part of solving a machine learning problem can be finding the right estimator for the job.

Different estimators are better suited for different types of data and different problems.

The flowchart below is designed to give users a bit of a rough guide on how to approach problems with regard to which estimators to try on your data.

As I mentioned before I am facing this problem as a Classification problem and to made it easier, I will do binary classifications this means the algorithm only will say if the index will be up or down

Looking at the classification algorithms I decide as exercise to try all important ones together with ensemble methods as I think is a good practice for me and also to compare outcomes in between them. A priori I don't know an algorithm which perform better than other working with Time Series



I provide a description of the algorithms I am trying. Just from http://scikit-learn.org

Linear Discriminant Analysis

It is a classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix.

The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions.

Support vector machines (SVMs)

Are a set of supervised learning methods used for classification, regression and outlier's detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Nearest Neighbors Classification

Neighbours-based classification is a type of *instance-based learning* or *non-generalizing learning*: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbours of the point.

The k-Neighbor's classification in K-Neighbors Classifier is the more commonly used of the two techniques. The optimal choice of the value $k$is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct

Logistic Regression

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function

Stochastic Gradient Descent (SGD)

Is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyper parameters such as the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

Decision Trees (DTs)

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by Boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called over-fitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Ensemble methods

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability / robustness over a single estimator.

Two families of ensemble methods are usually distinguished:

In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced.

Examples: Bagging methods, Forests of randomized trees, etc.

By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble.

Examples: Ada-Boost, Gradient Tree Boosting, etc.

Random Forest Classifier

A random forest is a Meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

In random forests each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but,

due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model.

Ada-Boost

The core principle of Ada-Boost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights w_1, w_2, ..., w_N to each of the training samples. Initially, those weights are all set to $w_i = 1/N$, so that the first step simply trains a weak learner on the original data. For each successive iteration, the sample weights are individually modified and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly. As iterations proceed, examples that are difficult to predict receive ever-increasing influence. Each subsequent weak learner is thereby forced to concentrate on the examples that are missed by the previous ones in the sequence.

Gradient Tree Boosting

Gradient Tree Boosting or Gradient Boosted Regression Trees (GBRT) is a generalization of boosting to arbitrary differentiable loss functions. GBRT is an accurate and effective off-the-shelf procedure that can be used for both regression and classification problems. Gradient Tree Boosting models are used in a variety of areas including Web search ranking and ecology.

The advantages of GBRT are:

- Natural handling of data of mixed type (= heterogeneous features)
- Predictive power
- Robustness to outliers in output space (via robust loss functions)

The disadvantages of GBRT are:

- Scalability, due to the sequential nature of boosting it can hardly be parallelized

Voting Classifier

The idea behind the voting classifier implementation is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses

Benchmark

In order to Benchmark the different classifiers, I am going to measure time for training and testing with best hyper-parameters and compare all metrics versus the Logistic Regression estimators. I am going to use the Data frame use in TEST 6 and all the estimators will be setup to the best hyper-parameters

The Data frame contains: Simple Rolling Average Series of daily return of each feature with window 5 days + Momentum Rolling Average Series of each feature with window 21 days + Volatility rolling average Series of each feature with window of 2 days and Exponential Rolling Average Series of each feature with window of 21 days.

Logistic Regression gets the following scores:

First line is when the estimators predict label 0, second line when estimators predict label 1. F-1 Score is 0.921328671328671

Accuracy	Algorithm	Date	Features	KFolds	Predicting	Training	Version	f1_score	matthews_corrcoef	precision	recall	roc_auc_score	support
0.910714286	LRC	17-05-21-18-18-36	Completed	0	00:00.0	00:00.2	SP Adjust Price 1 + Combination of Series	0.84	0.825740381	0	0.97	0.905698006	0.9
0.910714286	LRC	17-05-21-18-18-36	Completed	0	00:00.0	00:00.2	SP Adjust Price 1 + Combination of Series	0.98	0.825740381	1	0.87	0.905698006	0.92

In green those algorithms with better Accuracy than Logistic Regression

Algorithm	Accuracy	Parameters	version	Features	F1Score
GTB	0.926587302	Default	Completed	SP Adjust Price 1 + Combination of Series	0.933810376
RFC	0.920634921	Default	Completed	SP Adjust Price 1 + Combination of Series	0.927140255
VOT	0.907738095	Default	Completed	SP Adjust Price 1 + Combination of Series	0.918777293
ADA Bost	0.902777778	Default	Completed	SP Adjust Price 1 + Combination of Series	0.914035088
SGD	0.850198413	Default	Completed	SP Adjust Price 1 + Combination of Series	0.875103391
DTC	0.830357143	Default	Completed	SP Adjust Price 1 + Combination of Series	0.838831291
SVM	0.762896825	Default	Completed	SP Adjust Price 1 + Combination of Series	0.816012317
LDA	0.682539683	Default	Completed	SP Adjust Price 1 + Combination of Series	0.648351648
KNN	0.535714286	Default	Completed	SP Adjust Price 1 + Combination of Series	0.697674419

Accuracy	Algorithm	Date	Features	KFolds	Predicting	Training	Version	f1_score	matthews_corrcoef	precision	recall	roc_auc_score	support
0.926587302	GTB	17-05-21-18-18-35	Completed	0	00:00.0	00:05.0	SP Adjust Price 1 + Combination of Series	0.88	0.854101436	0	0.96	0.923504274	0.92
0.926587302	GTB	17-05-21-18-18-35	Completed	0	00:00.0	00:05.0	SP Adjust Price 1 + Combination of Series	0.97	0.854101436	1	0.9	0.923504274	0.93
0.920634921	RFC	17-05-21-18-16-28	Completed	0	00:00.2	00:15.3	SP Adjust Price 1 + Combination of Series	0.9	0.840590269	0	0.93	0.918945869	0.91
0.920634921	RFC	17-05-21-18-16-28	Completed	0	00:00.2	00:15.3	SP Adjust Price 1 + Combination of Series	0.94	0.840590269	1	0.91	0.918945869	0.93
0.907738095	VOT	17-05-21-18-22-43	Completed	0	00:00.8	04:05.7	SP Adjust Price 1 + Combination of Series	0.83	0.819844466	0	0.97	0.902635328	0.89
0.907738095	VOT	17-05-21-18-22-43	Completed	0	00:00.8	04:05.7	SP Adjust Price 1 + Combination of Series	0.97	0.819844466	1	0.87	0.902635328	0.92
0.902777778	ADA Bost	17-05-21-18-18-30	Completed	0	00:00.1	00:05.3	SP Adjust Price 1 + Combination of Series	0.83	0.808785553	0	0.95	0.898005698	0.89
0.902777778	ADA Bost	17-05-21-18-18-30	Completed	0	00:00.1	00:05.3	SP Adjust Price 1 + Combination of Series	0.96	0.808785553	1	0.87	0.898005698	0.91
0.850198413	SGD	17-05-21-18-18-35	Completed	0	00:00.0	00:00.0	SP Adjust Price 1 + Combination of Series	0.7	0.718331692	0	0.97	0.840242165	0.81
0.850198413	SGD	17-05-21-18-18-35	Completed	0	00:00.0	00:00.0	SP Adjust Price 1 + Combination of Series	0.98	0.718331692	1	0.79	0.840242165	0.88
0.830357143	DTC	17-05-21-18-22-43	Completed	0	00:00:00	00:00.1	SP Adjust Price 1 + Combination of Series	0.84	0.660366556	0	0.8	0.830840456	0.82
0.830357143	DTC	17-05-21-18-22-43	Completed	0	00:00:00	00:00.1	SP Adjust Price 1 + Combination of Series	0.82	0.660366556	1	0.85	0.830840456	0.84
0.762896825	SVM	17-05-21-18-18-24	Completed	0	00:00.1	01:56.3	SP Adjust Price 1 + Combination of Series	0.51	0.569128304	0	0.96	0.746082621	0.67
0.762896825	SVM	17-05-21-18-18-24	Completed	0	00:00.1	01:56.3	SP Adjust Price 1 + Combination of Series	0.98	0.569128304	1	0.7	0.746082621	0.82
0.682539683	LDA	17-05-21-18-18-35	Completed	0	00:00.0	00:00.2	SP Adjust Price 1 + Combination of Series	0.84	0.399430095	0	0.62	0.693019943	0.71
0.682539683	LDA	17-05-21-18-18-35	Completed	0	00:00.0	00:00.2	SP Adjust Price 1 + Combination of Series	0.55	0.399430095	1	0.8	0.693019943	0.65
0.535714286	KNN	17-05-21-18-16-12	Completed	0	00:00.7	00:00.0	SP Adjust Price 1 + Combination of Series	0	0	0	0	0.5	0
0.535714286	KNN	17-05-21-18-16-12	Completed	0	00:00.7	00:00.0	SP Adjust Price 1 + Combination of Series	1	0	1	0.54	0.5	0.7

All the metrics in Gradient Tree Boosting and Random Forest Classification are also better; all of them, but in GTB Training the estimator is 25 times slower and in RFC 75 times slower. Prediction time is slower in RFC

GTB gets the best accuracy and training and predicting time are good considering a Data frames with 76 columns and 3526 rows. Get and accuracy of 92.6% is a good outcome and confirm partially my hypothesis of being able to forecast the tendency of an Index looking at the values of certain global indicators referenced on the previous day. Same time, its shows up that the Efficient Markets Hypothesis theory is not completely true, specially strong version which states that stock prices will only respond to new information and so will follow a random walk.

This estimator also is getting an f1 score of 88% in predicting the label 0, and 97% predicting label 1

We are getting a Matthews Correlation coefficient of 0.85, which is really close to 1 meaning perfect prediction. The roc auc score of 92.2 confirm the outcomes

The split in between training and test set is 10 years for training from 2003 to 2013 and rest from 01012013 to 01012017 for test. The dataset contains 76 features which are defined as Simple Rolling average Series of daily returns with 5 days window, Simple rolling average Series of momentum with a 21 days window, simple rolling average Series of volatility with 2 days windows and Exponential Rolling average Series of daily return with a window of 5 days. I decided to use this new dataset after observing the issue of Multicollinearity found on the initial TEST 1, 2 3 and 4 when using Series of 5,21 and 63 days of the same technical Indicator

All the partial outcomes and in the file conbined.xlsx on the folder \out\Benchmarks

Multicollinearity

Doing the Cross Validation and searching for the best hyper-parameters, y got many times warning messages advising that my features suffered Multicollinearity. I was really intrigued by this fact and I did some research about the topic. On my initial test 1, 2 3 and 4 I was using dataset with series of 5,21 and 63 days all together, which produced this effect.

Definition

In statistics, Multicollinearity (also collinearity) is a phenomenon in which two or more predictor variables in a multiple regression model are highly correlated, meaning that one can be linearly predicted from the others with a substantial degree of accuracy. In this situation the coefficient estimates of the multiple regressions may change erratically in response to small changes in the model or the data. Multicollinearity does not reduce the predictive power or reliability of the model as a whole, at least within the sample data set; it only affects calculations regarding individual predictors. That is, a multiple regression model with correlated predictors can indicate how well the entire bundle of predictors predicts the outcome variable, but it may not give valid results about any individual predictor, or about which predictors are redundant with respect to others.

I found in forums that one effective mechanism is the Cholesky decomposition to the correlation matrix

In linear algebra, the Cholesky decomposition or Cholesky factorization is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose, which is useful e.g. for efficient numerical solutions and Monte Carlo simulations. It was discovered by André-Louis Cholesky for real matrices. When it is applicable, the Cholesky

decomposition is roughly twice as efficient as the LU decomposition for solving systems of linear equations

Multicollinearity means that your predictors are correlated. Why is this bad?

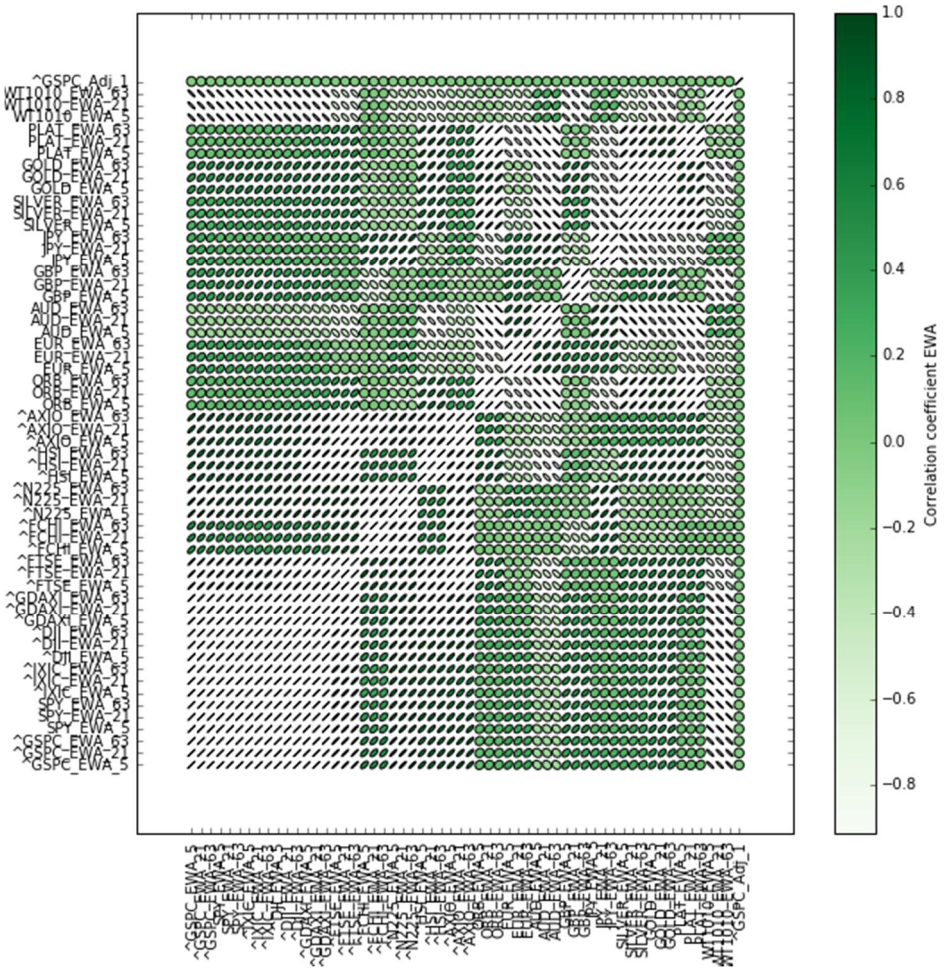
Because LDA, like regression techniques involves computing a matrix inversion, which is inaccurate if the determinant is close to 0 (*i.e.* two or more variables are almost a linear combination of each other). One way to detect Multicollinearity is to check the Cholesky decomposition of the correlation matrix - if there's Multicollinearity there will be some diagonal elements that are close to zero

I have done a Cholesky decomposition of the 4 case Test Dataframes, founding that the option to take 5, 21, 63 days as windows ranges for creating the rolling averages series create this collinearity, which make absolutely sense as each rolling average with more days means a soft version of the previous one.

Matrix and graphics are in folder CV\Cholesky

Example: Dataframe TEST 4

Looking at the Diagonal left- Down to Right-Up is easily identifiable all series that produce Multicollinearity. Visually all the “Islands” with looks like white are showing up the issue. This is the correlation Matrix of EWA rolling Average 5, 21, 63



I have some options to deal with this issue, even when working with Time Series increase the problem , 1 is use only a windows per series and mix them in a single data frame or try with dimensionality reduction. I am going to start for the first choice and combining different Features in a single Dataframe , which drove me to create the TEST number 6 , which at the end it would be which produce the best scores.

I also with SVM estimator on the Function "performSVM_PCAClass(X_train, y_train, X_test, y_test)" use the PCA dimensionality reduction, but As I got good outcomes mixing the rolling Average Series I decided to continue with that

Principal component analysis (PCA)

PCA is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. In scikit-learn, PCA is implemented as a *transformer* object that learns $n$components in its fit method, and can be used on new data to project it on these components.

The optional parameters whiten=True makes it possible to project the data onto the singular space while scaling each component to unit variance. This is often useful if the models down-stream make strong assumptions on the isotropy of the signal

Refinement

From Scikit

Cross Validation Grid Search: Searching for estimator parameters

Parameters that are not directly learnt within estimators can be set by searching a parameter space for the best Cross-validation: evaluating estimator performance score. Typical examples include C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc.

I have implemented functions to search the optimal parameters for each algorithm and I choose those which get the best validation scores. All the intermediary test are save it to an excel file for debugging and cross checking. In the folder CV are all the intermediary outcomes of running the CV Grid Search with each algorithm versus the four datasets

Support Vector Machines

params_map = [{ 'C': [0.001,0.005, 0.01,0.02, 0.1,0.4, 0.5,0.6,0.7, 1.0, 2.0, 5.0, 10.0, 100.0],'kernel': ['rbf'],'gamma': ['auto',0.001,0.01, 0.05,0.1, 0.5, 1.0, 5.0, 10.0,50.0,100.0,150.0]},

{'kernel': ['linear'], 'C': [1, 10, 100, 1000, 1500, 2000, 2500,3000, 3500,4000]},

{'kernel': ['poly'], 'C': [1, 10, 100, 1000, 1500, 2000, 2500,3000, 3500,4000],'degree':[2,3,4,5,6,7,8,9,10,15,20], 'gamma': ['auto',0.001,0.01, 0.05,0.1, 0.5, 1.0, 5.0, 10.0]},

{'kernel': ['sigmoid'], 'C': [1, 10, 100, 1000, 1500, 2000, 2500,3000, 3500,4000], 'gamma': ['auto',0.001,0.01, 0.05,0.1, 0.5, 1.0, 5.0, 10.0]}]

K-Nearest Neighbors

tuned_parameters = [{'n_neighbors': [50,100,150,200,250,300,350], 'weights': ['distance', 'uniform'],'algorithm': ['ball_tree', 'kd_tree', 'brute'], 'leaf_size': [30,40,50,60] }]

Random Forest Classifier

tuned_parameters = [{'n_estimators': [500, 1000, 2000 ,2500, 3000, 3500, 4000, 4500,5000], 'criterion':['gini','entropy'] }]

Gradient Tree Boosting

param_test1 = {'n_estimators':[20,40,50,60,70,80,90,100,150,200],'learning_rate':[0.1,0.2], min_samples_split':[100,200,300,400,500], 'min_samples_leaf':[10,30,40, 50, 75],'max_depth':[10,20,30,40],'max_features':['auto','sqrt'], 'subsample':[0.6,0.8,1.0], 'random_state':[10,20,30]}

Ada Boosting

param_grid = [{'n_estimators':[20,40,50,60,70,80,90,100,150,200],'algorithm' : ['SAMME', 'SAMME.R'] }]

Linear Discriminant Analysis

params_map = [{'solver':['lsqr','eigen'],'shrinkage':['auto',0.1,0.3,0.5,0.7,0.9],'store_covariance':['True', 'False'],

'n_components':[2,3,4,5,6,7,8,9,10,11,12,13,14,15]}, {'solver':['svd'], 'tol':[0.3,0.45,0.5,0.55,0.8, 1.0],'store_covariance':['True', 'False'], 'n_components':[2,3,4,5,6,7,8,9,10,11,12,13,14,15]}]

Stochastic Gradient Descent (SGD)

param_test1=[{'loss':['hinge', 'log', 'modified_huber', 'squared_hinge', 'perceptron'], 'penalty':['none', 'l2', 'l1', 'elasticnet'], 'learning_rate':['constant','optimal','invscaling'], 'eta0':[0.1,0.2]}]

Decision Trees (DTs)

param_grid = [{'criterion':['gini','entropy'],'max_features':['auto','sqrt','log2'] }]

Logistic regression

param_grid = {'C':[0.001, 0.01, 0.1, 1, 10, 100, 1000], 'penalty' :["l1","l2"], 'solver' :["newton-cg", "lbfgs", "liblinear", "sag"] }

Voting Classifier

This specific Classifier is composed of 4 classifiers from which I took the optimal parameters found after the Grid search,

The Classifiers are Support Vector Machines, KNN , Gaussian Naïve Bayes and Stochastic Gradient Descent

In the excel File Best parameters.xlsx (in CV folder), I include the hyper-parameters which shot the better scores using the 5 defined datasets. First and evident outcome of the CV grid search is that the Dataframe 1 and 6 are producing the best scores

DataFrame 1 is Simple Rolling Average Series of each feature with window 5 days

DataFrame 6 is Simple Rolling Average Series of each feature with window 5 days + Momentum Rolling Average Series of each feature with window 21 days + Volatility rolling average Series of each feature with window of 2 days and Exponential Rolling Average Series of each feature with window of 21 days

The top fifteen are in the table below, and my next step is setup all the estimators with the hyper-parameters which produce the better scores and I will run the 5 TEST datasets versus all estimators optimized to Benchmark the model

ALG	TEST	Parameters	mean_validation_score	cv_validation_scores
GTB	6	{'max_features': 'auto', 'min_samples_split': 300, 'min_samples_leaf': 75, 'n_estimators': 90, 'learning_rate': 0.1}	0,933034212	[0.93527803 0.92102928 0.93933824 0.93789379 0.93163172]
RFC	6	{'n_estimators': 500, 'criterion': 'gini'}	0,929653041	[0.92078285 0.92896175 0.93922652]
LRC	6	{'penalty': 'l1', 'C': 1, 'solver': 'liblinear'}	0,92403653	[0.91901408 0.92907801 0.92401747]
SGD	6	{'learning_rate': 'invscaling', 'alpha': 0.01, 'loss': 'perceptron', 'penalty': 'elasticnet', 'eta0': 0.2}	0,860205599	[0.81024096 0.90156394 0.85079365 0.82986913 0.90856031]
DTC	6	{'max_features': 'sqrt', 'criterion': 'entropy'}	0,819078905	[0.84900285 0.77835052 0.82989691]
ADA	6	{'algorithm': 'SAMME.R', 'n_estimators': 80, 'learning_rate': 0.7}	0,816515589	[0.8310992 0.83870968 0.77969175]
LRC	1	{'C': 10, 'penalty': 'l1', 'solver': 'liblinear'}	0,768637384	[0.77 0.77 0.77]
GTB	1	{'min_samples_leaf': 50, 'n_estimators': 70, 'max_features': 'auto', 'min_samples_split': 100, 'learning_rate': 0.1}	0,760618655	[0.77 0.77 0.75 0.76 0.75]
SGD	1	{'penalty': 'elasticnet', 'eta0': 0.1, 'alpha': 0.001, 'loss': 'perceptron', 'learning_rate': 'invscaling'}	0,75509702	[0.77 0.77 0.77 0.77 0.7]
RFC	1	{'criterion': 'gini', 'n_estimators': 4000}	0,754964671	[0.75 0.76 0.75]
SVM	1	{'C': 3500, 'kernel': 'sigmoid', 'gamma': 0.05}	0,753804383	[0.76 0.76 0.74]
SVM	6	{'C': 2000, 'kernel': 'linear'}	0,741688173	[0.85483871 0.67153285 0.69863901]
LDA	1	{'store_covariance': 'True', 'n_components': 1, 'solver': 'lsqr', 'shrinkage': 0.1}	0,723913	[0.72 0.74 0.71]
KNN	1	{'leaf_size': 30, 'algorithm': 'ball_tree', 'n_neighbors': 350, 'weights': 'distance'}	0,710429381	[0.71 0.71 0.71 0.71 0.7 0.71 0.71 0.71 0.71 0.71]
KNN	1	{'leaf_size': 40, 'algorithm': 'ball_tree', 'n_neighbors': 350, 'weights': 'distance'}	0,710429381	[0.71 0.71 0.71 0.71 0.7 0.71 0.71 0.71 0.71 0.71]

Methodology

Data Pre-processing

1) Download the time series from Yahoo and Quandl and save them to CSV files
2) Iterate over each individual file, starting with S&P 500, which will be used as a master for trading days. From all indexes we take only the column Adjusted Price. For the rest, the column which represent when possible, a price which is known before NYSE opens (example: Silver take "London 08:00"). Join all the individual columns in a pandas Dataframes
3) As not all stock exchanges have the same trading dates, and not all commodities as well, we have some gaps on the individual columns due to we take as reference the trading days of NYSE. These holes are with NaN,s and we fill them up with the previous value in the Time Series. For those gaps at the beginning of the Time Series we copy back the first initial value
4) We Normalize all the columns to values in between -1 to 1
5) Calculate Daily returns of S&P 500 normalized, which will be the value to predict. Shift the hole series 1 day back in order to match the value of S&P500 with the calculates values of the rest of the Series. This way we are sure that all data we use to predict the value of the index is known in advance to the NYSE opens. We calculate daily returns of each feature and we shift 1 day up to those which its value is known before the NYSE opens. We do that in order to balance the comparison, and compare all indexes and prices on the same day. Not doing this we have index on day x and index on day x-1, which is not a consistent approach.
6) Depending of the test to run create the Dataframes with Moving Averages (Daily Return n, Volatility, Momentum or Exponential Weight average) using windows of 5, 21 and 63
7) Remove the initial 3 months of the Dataframes due to empty values caused for the rolling averages with windows 63. We can do that as loosing 3 month for 12 years of data, is better to fill up back with the first value produced by the rolling average
8) Clean NaN and +Infinite or – Infinite Values just in case
9) Create and additional column on the Dataframes with the Value UP or Down, applying the following value. If daily return of S&P 500 is negative Down, if it is positive UP
10) Selecting the features used for prediction (basically all the columns except for the first one – float returns – and the last one – string returns)
11) Splitting the whole Dataframes into train and test set (based on a date passed as argument) and returns for each of them predictors and actual prediction. In this project we use 4 years as test Set and 10 years as Training set. We want to capture on the training set the year 2007, 2008 and 2009 as they were the Great recession years on the Markets, whit big losses and high volatility
12) I run a prediction with all estimators using default hyper- parameters. I use a numbers of selectors for test to run (Variable TEST) and each test produce a report with the metrics and I write accuracy and F-1 Score together with the information of the test in a excel saved in raw_data named accuracy.xlsx. all the individual outcomes are saved in folder raw_data\archive
13) After this test a made a search of the best hyper parameters for each estimator. I run the search using the 5 Different Dataframes. I have another selector (variable optimize) with value 0 call the estimators, with value 1 call the CVgridsearch function with a grid of parameters to test. Each estimator have their own optimizer. All the individual outcomes are saved in excel files , and I save the grid_scores_ detailed with verbosity 100, to later choose the best setup

14) Each estimator is configured with the best hyper parameters for TEST 6 where I got the best scores in all estimators
15) I run 2 Cross Validation techniques described on further section to validates the outcomes, with all estimators using the best hyper parameters for TEST 6
16) Exploration of other techniques like stacking and blending

Note for step 6. After my initial test and observe the issue of Multicollinearity, the final Data Frames for each use Case (TEST1, TEST2, TEST3 and TEST4) were created only with single rolling average with window of 5 days as combining series with different windows produced this effect

Implementation

1. I create correlation matrix of each index versus S&P500 before and after pre-processing, this if also when creating the rolling average series
2. Test all estimators versus all independent Rolling average Series (Daily returns, Momentum, Volatility and Exponential Weighted Average) with a windows size of 5. This correspond to 1 trading week. All estimators will run with default parameters. The outcomes of each individual test are in the folder out\archive
3. Tune hyper-parameters of every estimator with the four different dataset produced. The outcomes are in the section Refinement and the individual test are on the folder CV
4. Benchmarking the estimators with the tuned hyper-parameters versus the 5 datasets. Outcomes of benchmarking are in folder out\Benchmarks
5. Evaluate the model. For this section I use 2 techniques tailored to Time Series. Multiple Train-Test and Walk-Forward Validation. The explanations and outcomes produced are explained on the section Model Evaluation and Validation. The intermediate outputs of the cross-validation are in folders out\ Cross_Validation_Kfolds_TS and out\ Cross_Validation_WFV. Additionally I am recording on the excel raw_data\accuracy.xlsx, F1-score and accuracy of each individual test done.
6. Improve the model combining estimators via stacking and blending
7. Final test are saved on the folder out\Final Test
8. The file BuildaStockPricePredictor.py contains the main body of the application
9. The file library.py contains all support classes, classes than implements estimators, classes to create the Rolling average Series, Classes which implements different Cross Validation techniques and Cross Validation to search best hyper-parameters, classes for plotting graphs and print reports to csv or xlsx.
10. The file CORRELATION.PY contains classes to plot correlation graphics and calculate Cholesky matrix
11. The file carga_datos.py download the financial information from yahoo financial and Quandl and save it on csv files
12. The file Data_analysys.py contains some classes to help and plot in the data analysis section

13. The files prices.xlsx (contains adjusted price of all features without normalization),
prices_normalized.xlsx(contains adjusted price of all features with normalization),
prices_adjusted_rolling_test1.xlsx(Contains Series for TEST 1),
prices_adjusted_rolling_test2.xlsx(Contains Series for TEST 2),
prices_adjusted_rolling_test3.xlsx(Contains Series for TEST 3),
prices_adjusted_rolling_test4.xlsx(Contains Series for TEST 5),
prices_adjusted_rolling_test6.xlsx(Contains Series for TEST 6)

TEST 5 , was used in some intermediate TEST and on final TEST not used

Results

Model Evaluation and Validation

When I came over this point I was intrigued if the conventional Cross-validation techniques using k-folds apply for time series and I was researching and I found that we can't use them. I found two interesting tutorials on the topic

http://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

http://francescopochetti.com/pythonic-cross-validation-time-series-pandas-scikit-learn/

And I applied what I learnt on these two places. I reproduce here part of the first tutorial with the explanations of why Time series required a bit special technique to cross validate the outcomes due to we need respect the temporal order in which values were observed when we split the data

We could evaluate it on the data used to train it. This would be invalid. It might provide insight into how the selected model works, and even how it may be improved. But, any estimate of performance on this data would be optimistic, and any decisions based on this performance would be biased.

Why?

It is helpful to take it to an extreme:

A model that remembered the timestamps and value for each observation would achieve perfect performance.

All real models we prepare will report a pale version of this result.

When evaluating a model for time series forecasting, we are interested in the performance of the model on data that was not used to train it. In machine learning, we call this unseen or out of sample data.

We can do this by splitting up the data that we do have available. We use some to prepare the model and we hold back some data and ask the model to make predictions for that period. The evaluation of these predictions will provide a good proxy for how the model will perform when we use it operationally.

In applied machine learning, we often split our data into a train and a test set: the training set used to prepare the model and the test set used to evaluate it. We may even use k-fold cross validation that repeats this process by systematically splitting the data into k groups, each given a chance to be a held out model.

These methods cannot be directly used with time series data.

This is because they assume that there is no relationship between the observations, that each observation is independent.

This is not true of time series data, where the time dimension of observations means that we cannot randomly split them into groups. Instead, we must split data up and respect the temporal order in which values were observed.

In time series forecasting, this evaluation of models on historical data is called back-testing. In some time series domains, such as meteorology, this is called hind-casting, as opposed to forecasting.

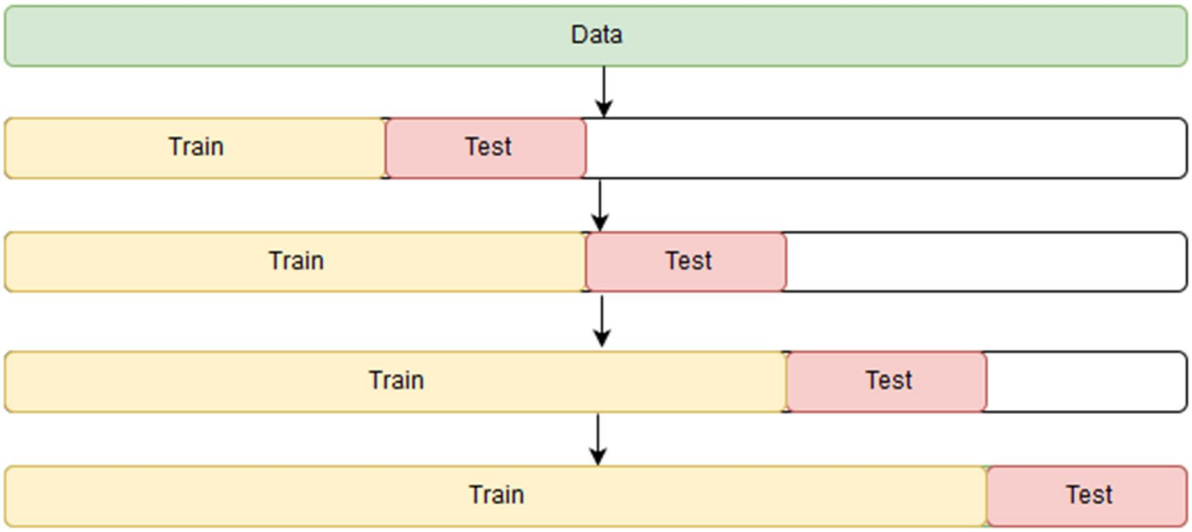
We will look at three different methods that you can use to back-test your machine learning models on time series problems. They are:

1. **Train-Test split** that respect temporal order of observations.
2. **Multiple Train-Test splits** that respect temporal order of observations.
3. **Walk-Forward Validation** where a model may be updated each time step new data is received.

In this project we are going to use number 2 and number 3

Explanation on Multiple Train-Test Splits

1. Split data in train and test set given a Date (i.e. test set is what happens after 2 April 2014 included).
2. Split train set (i.e. what happens before 2 April 2014 not included) in for example 10 consecutive time folds.
3. Then, in order not to lose the time information, perform the following steps:
4. Train on fold 1 –> Test on fold 2
5. Train on fold 1+2 –> Test on fold 3
6. Train on fold 1+2+3 –> Test on fold 4
7. Train on fold 1+2+3+4 –> Test on fold 5
8. Train on fold 1+2+3+4+5 –> Test on fold 6
9. Train on fold 1+2+3+4+5+6 –> Test on fold 7
10. Train on fold 1+2+3+4+5+6+7 –> Test on fold 8
11. Train on fold 1+2+3+4+5+6+7+8 –> Test on fold 9
12. Train on fold 1+2+3+4+5+6+7+8+9 –> Test on fold 10
13. Compute the average of the accuracies of the 9 test folds (number of folds – 1)

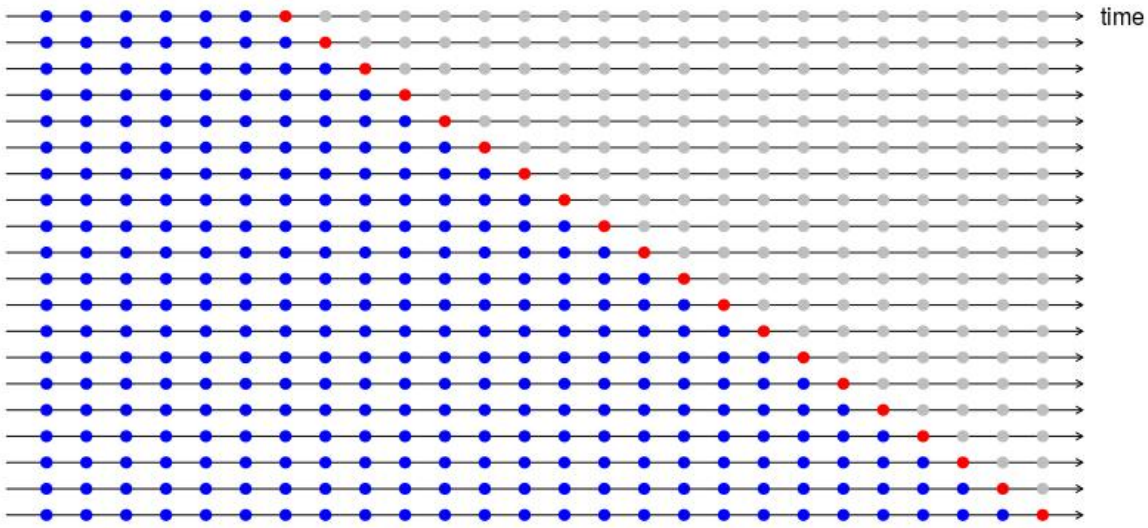


Accuracy	Algorithm	Date	Features	K-Folds	Predicting	Training	Version	f1_score	Matthews coef	precision	recall	roc_auc_score	support
0.932998324958	RFC	17-05-12-16-52-09	Completed	K-folds 4	0:00:00.079963	0:00:05.259593	SP Adjust Price 1 + Combination of Series	0.93	0.864859622539	0.0	0.93	0.931878974556	0.93
0.932998324958	RFC	17-05-12-16-52-09	Completed	K-folds 4	0:00:00.079963	0:00:05.259593	SP Adjust Price 1 + Combination of Series	0.94	0.864859622539	1.0	0.94	0.931878974556	0.94
0.928870292887	LRC	17-05-12-16-57-31	Completed	K-folds 2	00:00:00	0:00:00.029987	SP Adjust Price 1 + Combination of Series	0.91	0.856625616882	0.0	0.94	0.92705716302	0.92
0.928870292887	LRC	17-05-12-16-57-31	Completed	K-folds 2	00:00:00	0:00:00.029987	SP Adjust Price 1 + Combination of Series	0.95	0.856625616882	1.0	0.92	0.92705716302	0.94
0.928391959799	RFC	17-05-12-16-56-17	Completed	K-folds 3	0:00:00.099954	0:00:04.533930	SP Adjust Price 1 + Combination of Series	0.92	0.855522568227	0.0	0.92	0.927051930187	0.92
0.928391959799	RFC	17-05-12-16-56-17	Completed	K-folds 3	0:00:00.099954	0:00:04.533930	SP Adjust Price 1 + Combination of Series	0.94	0.855522568227	1.0	0.93	0.927051930187	0.93
0.928033472803	GTB	17-05-12-16-57-30	Completed	K-folds 2	00:00:00	0:00:00.771652	SP Adjust Price 1 + Combination of Series	0.92	0.854947071277	0.0	0.92	0.927473535639	0.92
0.928033472803	GTB	17-05-12-16-57-30	Completed	K-folds 2	00:00:00	0:00:00.771652	SP Adjust Price 1 + Combination of Series	0.93	0.854947071277	1.0	0.93	0.927473535639	0.93
0.926100628931	RFC	17-05-12-16-53-48	Completed	K-folds 5	0:00:00.069967	0:00:05.819359	SP Adjust Price 1 + Combination of Series	0.94	0.850924166861	0.0	0.91	0.923254660029	0.93
0.926100628931	RFC	17-05-12-16-53-48	Completed	K-folds 5	0:00:00.069967	0:00:05.819359	SP Adjust Price 1 + Combination of Series	0.92	0.850924166861	1.0	0.95	0.923254660029	0.93
0.925523012552	RFC	17-05-12-16-57-28	Completed	K-folds 2	0:00:00.119944	0:00:03.398457	SP Adjust Price 1 + Combination of Series	0.92	0.849964667542	0.0	0.92	0.925165843331	0.92
0.925523012552	RFC	17-05-12-16-57-28	Completed	K-folds 2	0:00:00.119944	0:00:03.398457	SP Adjust Price 1 + Combination of Series	0.93	0.849964667542	1.0	0.93	0.925165843331	0.93
0.925052410901	GTB	17-05-12-16-53-57	Completed	K-folds 5	00:00:00	0:00:01.413353	SP Adjust Price 1 + Combination of Series	0.93	0.84888089303	0.0	0.93	0.921575717241	0.93
0.925052410901	GTB	17-05-12-16-53-57	Completed	K-folds 5	00:00:00	0:00:01.413353	SP Adjust Price 1 + Combination of Series	0.94	0.84888089303	1.0	0.94	0.921575717241	0.94

Best scores RFC with 4 folds gets 93, 2% accuracy. GTB holds the top position on the benchmark test here is getting also 92.6% of accuracy with 2 and 5 Folds

Walk-Forward Validation

In this procedure, there is a series of test sets, each consisting of a single observation. The corresponding training set consists only of observations that occurred prior to the observation that forms the test set. Thus, no future observations can be used in constructing the forecast. The following diagram illustrates the series of training and test sets, where the blue observations form the training sets, and the red observations form the test sets.



The forecast accuracy is computed by averaging over the test sets. This procedure is sometimes known as “evaluation on a rolling forecasting origin” because the “origin” at which the forecast is based rolls forward in time.

Validation Matrix

Accuracy	Algorithm	Date	Features	KFolds	Predicting	Training	Version	f1_score	matthews_corrcoef	precision	recall	roc_auc_score	support
1	ADA Bost	17-05-13-00-59-15	Completed	WFV	00:00:00	00:00,1	SP Adjust Price 1 + Combination of Series	1	0	0	1	0	1
1	GTB	17-05-13-01-21-11	Completed	WFV	00:00:00	00:01,6	SP Adjust Price 1 + Combination of Series	1	0	0	1	0	1
1	RFC	17-05-13-00-33-31	Completed	WFV	00:00,1	00:15,2	SP Adjust Price 1 + Combination of Series	1	0	0	1	0	1
0,969215492	LRC	17-05-13-01-25-28	Completed	WFV	00:00:00	00:00,2	SP Adjust Price 1 + Combination of Series	0,94	0	0	0,99	0	0,97
0,961271102	SVM	17-05-17-01-26-42	Completed	WFV	00:00,0	00:51,6	SP Adjust Price 1 + Combination of Series	0,95	0	0	0,96	0	0,96
0,932998325	RFC	17-05-12-16-52-09	Completed	K-folds 4	00:00,1	00:05,3	SP Adjust Price 1 + Combination of Series	0,93	0,864859623	0	0,93	0,931878975	0,93
0,930486594	DTC	17-05-13-02-15-34	Completed	WFV	00:00:00	00:00,1	SP Adjust Price 1 + Combination of Series	0,94	0	0	0,91	0	0,93
0,928870293	LRC	17-05-12-16-57-31	Completed	K-folds 2	00:00:00	00:00,0	SP Adjust Price 1 + Combination of Series	0,91	0,856625617	0	0,94	0,927057163	0,92
0,92839196	RFC	17-05-12-16-56-17	Completed	K-folds 3	00:00,1	00:04,5	SP Adjust Price 1 + Combination of Series	0,92	0,855522568	0	0,92	0,92705193	0,92
0,928033473	GTB	17-05-12-16-57-30	Completed	K-folds 2	00:00:00	00:00,8	SP Adjust Price 1 + Combination of Series	0,92	0,854947071	0	0,92	0,927473536	0,92
0,926587302	GTB	17-05-17-09-58-57	Completed	TEST 6	00:00:00	00:01,9	SP Adjust Price 1 + Combination of Series	0,88	0	0	0,96	0	0,92
0,926100629	RFC	17-05-12-16-53-48	Completed	K-folds 5	00:00,1	00:05,8	SP Adjust Price 1 + Combination of Series	0,94	0,850924167	0	0,91	0,92325466	0,93
0,925523013	RFC	17-05-12-16-57-28	Completed	K-folds 2	00:00,1	00:03,4	SP Adjust Price 1 + Combination of Series	0,92	0,849964668	0	0,92	0,925165843	0,92
0,925052411	GTB	17-05-12-16-53-57	Completed	K-folds 5	00:00:00	00:01,4	SP Adjust Price 1 + Combination of Series	0,93	0,848880893	0	0,93	0,921575717	0,93
0,924623116	GTB	17-05-12-16-56-22	Completed	K-folds 3	00:00:00	00:01,2	SP Adjust Price 1 + Combination of Series	0,92	0,848254093	0	0,93	0,922543244	0,92
0,923611111	RFC	17-05-17-09-58-01	Completed	TEST 6	00:00,1	00:07,9	SP Adjust Price 1 + Combination of Series	0,9	0	0	0,93	0	0,92
0,922431866	LRC	17-05-12-16-53-58	Completed	K-folds 5	00:00,0	00:00,0	SP Adjust Price 1 + Combination of Series	0,93	0,844135737	0	0,92	0,91846352	0,93
0,922418136	RFC	17-05-12-16-54-30	Completed	K-folds 6	00:00,1	00:05,8	SP Adjust Price 1 + Combination of Series	0,91	0,843177401	0	0,92	0,920888615	0,92
0,922389726	GTB	17-05-12-16-52-16	Completed	K-folds 4	00:00:00	00:01,3	SP Adjust Price 1 + Combination of Series	0,93	0,84447694	0	0,91	0,920668313	0,92
0,919395466	GTB	17-05-12-16-54-42	Completed	K-folds 6	00:00:00	00:01,5	SP Adjust Price 1 + Combination of Series	0,92	0,837806672	0	0,91	0,916854764	0,92
0,917713568	LRC	17-05-12-16-56-22	Completed	K-folds 3	00:00,0	00:00,0	SP Adjust Price 1 + Combination of Series	0,91	0,834273545	0	0,92	0,915651392	0,92
0,917154812	VOT	17-05-12-16-57-32	Completed	K-folds 2	00:00,4	00:00,3	SP Adjust Price 1 + Combination of Series	0,91	0,833046393	0	0,91	0,916584333	0,91
0,916876574	LRC	17-05-12-16-54-43	Completed	K-folds 6	00:00,0	00:00,0	SP Adjust Price 1 + Combination of Series	0,93	0,833423082	0	0,92	0,913161345	0,92
0,91468254	VOT	17-05-17-10-00-38	Completed	TEST 6	00:00,5	01:38,5	SP Adjust Price 1 + Combination of Series	0,85	0	0	0,97	0	0,9

After the cross validation we can observe that either with Walk-Forward Validation or in the K-folds adapted to time Series, both produce the same outcomes than we got in previous tests and in some algorithms, even improved its scores. Working with Time Series Data is quite specific due its nature and I observed than specially using the Walk forward Validation can be really time consuming depending of the window chosen (in my test 1 day) , the minimum size of the initial training set and the size of the dataset. The benefit in the other hand is providing a much more robust estimation of how the chosen modeling method and parameters will perform in practice. This improved estimate comes at the computational cost of creating many models. We can observe like on the Walk forward Validation some estimators hits 100% accuracy like GTB, Ada Bost and Random Forest

On the Validation tests, we can observe as the GTB and other methods even improved their scores, being the case of Support Vector Machines curious as how with a linear Kernel with a "C" of 2000 performs really well in terms of accuracy and time.

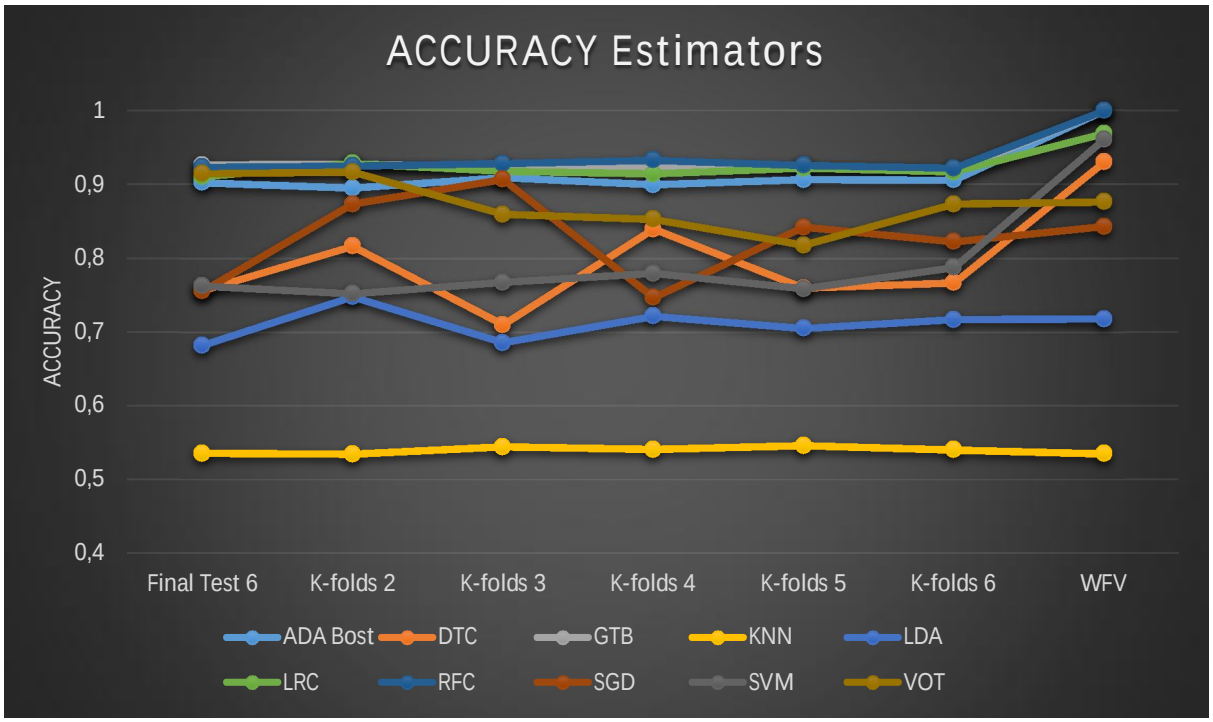
> The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyper-plane if that hyper-plane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyper-plane, even if that hyper-plane misclassifies more points. For very tiny values of C, you should get misclassified examples, often even if your training data is linearly separable.

I think that some time tuning the algorithms and using stacking and Bagging it will be possible increase the prediction scores

In the Validation tests, I have observed as well that, from the four features chosen at the beginning rolling averages of daily returns and rolling averages of Momentum gets the better scores, which somehow confirm the theory of Jegadeesh and Titman, who discovered that in the short term, stock prices tend to exhibit momentum [6]. Stocks that have recently been increasing continue to increase, and recently decreasing stocks continue to decrease. This type of trend implies some amount of predictability to future stock prices, contradicting the EMH. Combining carefully some Series get good outcomes

Accuracy Matrix

	Final Test 6	K-folds 2	K-folds 3	K-folds 4	K-folds 5	K-folds 6	WFV
ADA Bost	0,902777778	0,89539749	0,909547739	0,899497487	0,906708595	0,906297229	1
DTC	0,756944444	0,816736402	0,710427136	0,839754327	0,759958071	0,767254408	0,930486594
GTB	0,926587302	0,928033473	0,924623116	0,922389726	0,925052411	0,919395466	1
KNN	0,535714286	0,534728033	0,54459799	0,541038526	0,546121593	0,540554156	0,535253227
LDA	0,682539683	0,748117155	0,685929648	0,721943049	0,705450734	0,717380353	0,717974181
LRC	0,910714286	0,928870293	0,917713568	0,914572864	0,922431866	0,916876574	0,969215492
RFC	0,923611111	0,925523013	0,92839196	0,932998325	0,926100629	0,922418136	1
SGD	0,755952381	0,873640167	0,907035176	0,747068677	0,842243187	0,822670025	0,843098312
SVM	0,762896825	0,752301255	0,76758794	0,780011167	0,758909853	0,78790932	0,961271102
VOT	0,91468254	0,917154812	0,859924623	0,853154662	0,818134172	0,873551637	0,876861966



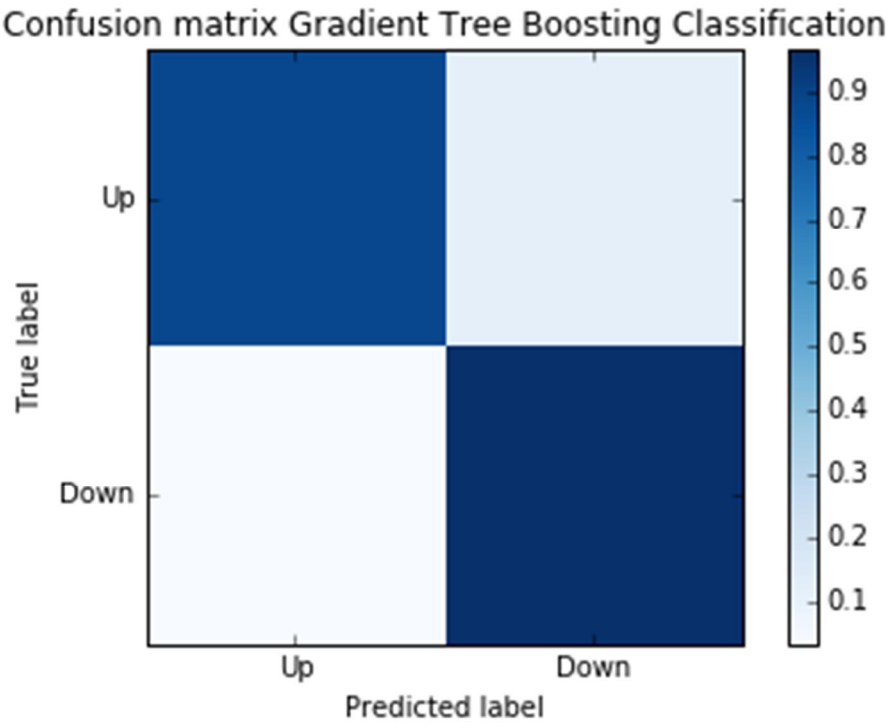
Confusion Matrix

In the field of machine learning and specifically the problem of statistical classification, a confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa).[2] The name stems from the fact that it makes it easy to see if the system is confusing two classes (i.e. commonly mislabelling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions

The diagonal elements represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabelled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.

I plot here the confusion Matrix of GTB, which show up the better scores. Rest are plotted in annex 2



Conclusion

The outcomes I presented in this projects, after almost 3 months of time, and not being a specialist on the matter besides that I work for more than 15 years in finance sector are quite promising. Choosing the right combination of technical features, using global indicators and align them correctly on time in order to have all the information need it previous to the opening to the stock exchange market, is able to forecast the tendency of the index with an accuracy of 100%. Many traders and portfolio managers would argue that predicting is not enough, and if would be necessary to estimate the % of increase or decrease. In my opinion, I think is possible at least to confirm if the investment strategy is adequate, allowing the portfolio manager play on the safe side. If we look at the assets in a managed portfolio and we are able to forecast if they are going to go up or down beforehand, will give us the opportunity at least to react quickly in some scenarios and specially in those where is forecasted all going down for example, minimize the losses. Here there is still many questions to be answer in how to use this predictor when we have portfolios mix in different currencies, having assets which are valuated in different time schedules for example and how to choose right technical features for each asset. Using Rolling Averages Series, we are soften the effects of jumps for consecutive days, but it would be challenging to choose correct indicators for more heterogeneous assets

The Final results after running the Cross Validation test confirm the outcomes on the benchmark test GTB and RFC beat the LR estimator in all scores. We can observe that in the Walk-Forward Validation which is very similar to the real prediction, we are getting 1 in accuracy. Why I say this? Because in this validation technique testing set was established as 1 day window and it is offered for training all the observation previous to the testing day. In our WFV process, we repeat the predictions 1008 times in a row as I started the process in 01/01/2013 forward up to 30/12/2016, and the outcome I am showing in the table is the mean of all individual predictions, which is 1.

	Final Test 6	K-folds 2	K-folds 3	K-folds 4	K-folds 5	K-folds 6	WFV
GTB	0,926587302	0,928033473	0,924623116	0,922389726	0,925052411	0,919395466	1
RFC	0,923611111	0,925523013	0,92839196	0,932998325	0,926100629	0,922418136	1
LRC	0,910714286	0,928870293	0,917713568	0,914572864	0,922431866	0,916876574	0,969215492

F1-score, precision and recall is also is 1 and we could not measure here Matthews correlation coefficient and Roc Auc Score as is not possible to calculate when the test set has only 1 observation

Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public datasets were found
2. The data was downloaded and pre-processed
3. Metric were defined and selected different estimators
4. Initial test were carried out with all estimators and all dataset
5. the estimators were optimized
6. A benchmark was created for the classifiers and test ran to benchmarking the optimized estimators
7. The model was evaluated and validated
8. It was explored alternatives to improve scores

I found the step 7 and 8 the most difficult, as I had to familiarize myself with Time Series and how to cross-validate and split Time Series and how to approach conceptually stacking and blending

As for the most interesting aspects of the project, getting involve in understanding estimators , ML techniques , together with understanding how portfolio managers or traders do their job to create a program which forecast an Index and everything with a direct and real use, it has been very challenging and interesting. Many hours reading, exploring and testing different possibilities have been very gratifying and rewarding.

Improvement

Stacked Generalization and Blending

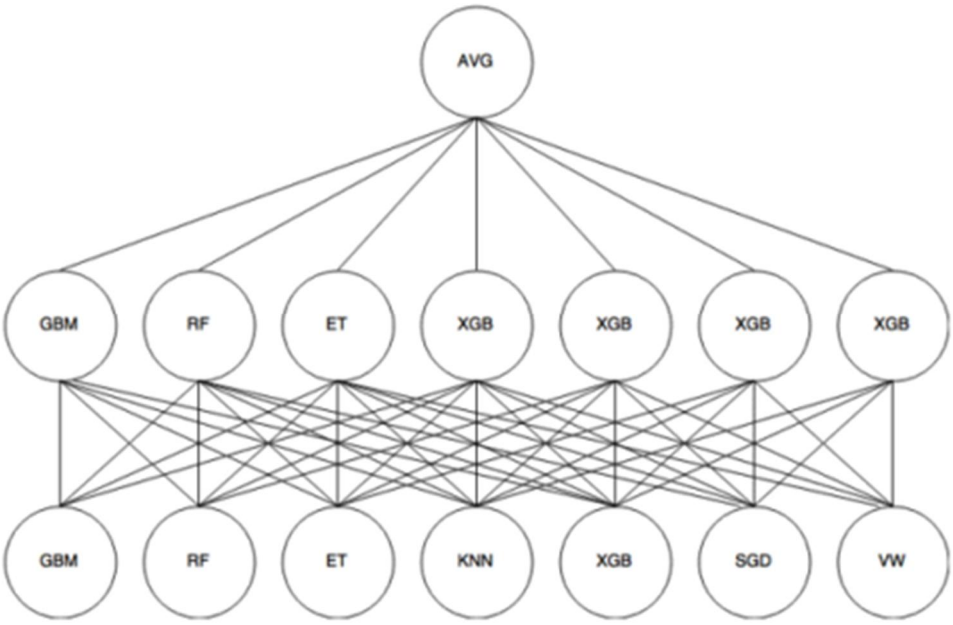
Stacked Generalization or stacking is an ensemble algorithm where a new model is trained to combine the predictions from two or more models already trained or your dataset.

The predictions from the existing models or sub-models are combined using a new model, and as such stacking is often referred to as blending, as the predictions from sub-models are blended together.

It is typical to use a simple linear method to combine the predictions for sub-models such as simple averaging or voting, to a weighted sum using linear regression or logistic regression.

Models that have their predictions combined must have skill on the problem, but do not need to be the best possible models. This means that you do not need to tune the sub-models intently, as long as the model shows some advantage over a baseline prediction.

It is important that sub-models produce different predictions, so-called uncorrelated predictions. Stacking works best when the predictions that are combined are all skillful, but skillful in different ways. This may be achieved by using algorithms that use very different internal representations (trees compared to instances) and/or models trained on different representations or projections of the training data.



In the class performEnsemberBlendingClass, I have created my class with 7 estimators for the first layer (used during the initial test), with the best hyper parameters according to Data Frame Test 1. With predictions each predictions of the test set become a feature of a new Dataframe. I introduce the fit and predict of these 7 estimators in a loop and I execute the loop 50 times. Finally I join the 350 columns to the original y_test

With this new dataset, y train 2 new estimators, first I look for the best hyper parameters with CVGridSearch and one I found them (to tailor the estimators to this specific Dataset)

The estimators in second layer are Logistic regression and SVM, the best accurary spots with 3 different sets in Logistic Regression and 1 set in Support vector Machines

Logistic regression

parameters	mean_validation_score	cv_validation_scores
{'penalty': 'l2', 'C': 0.001, 'solver': 'newton-cg'}	0,754373357	[0.75223881 0.76012461 0.75075075]
{'penalty': 'l2', 'C': 0.001, 'solver': 'lbfgs'}	0,754373357	[0.75223881 0.76012461 0.75075075]
{'penalty': 'l2', 'C': 0.001, 'solver': 'sag'}	0,754373357	[0.75223881 0.76012461 0.75075075]

Support Vector Machines

parameters	mean_validation_score	cv_validation_scores
{'kernel': 'rbf', 'C': 100.0, 'gamma': 0.1}	0,758143672	[0.7752443 0.75342466 0.7456446]

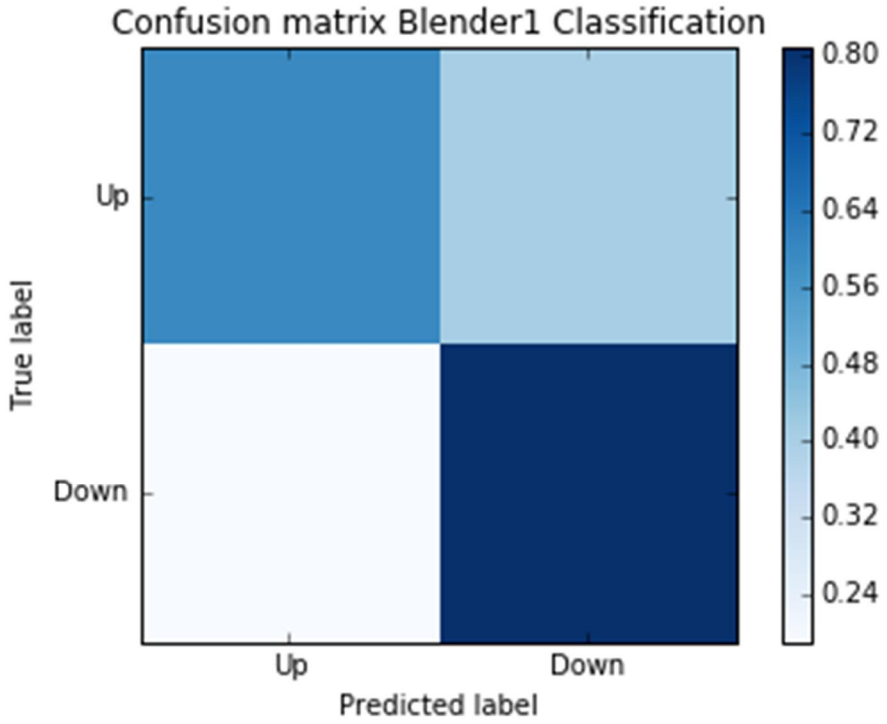
Now I set up the estimators second layer with the best hyper parameters and run a cycle of again of stacking and blending

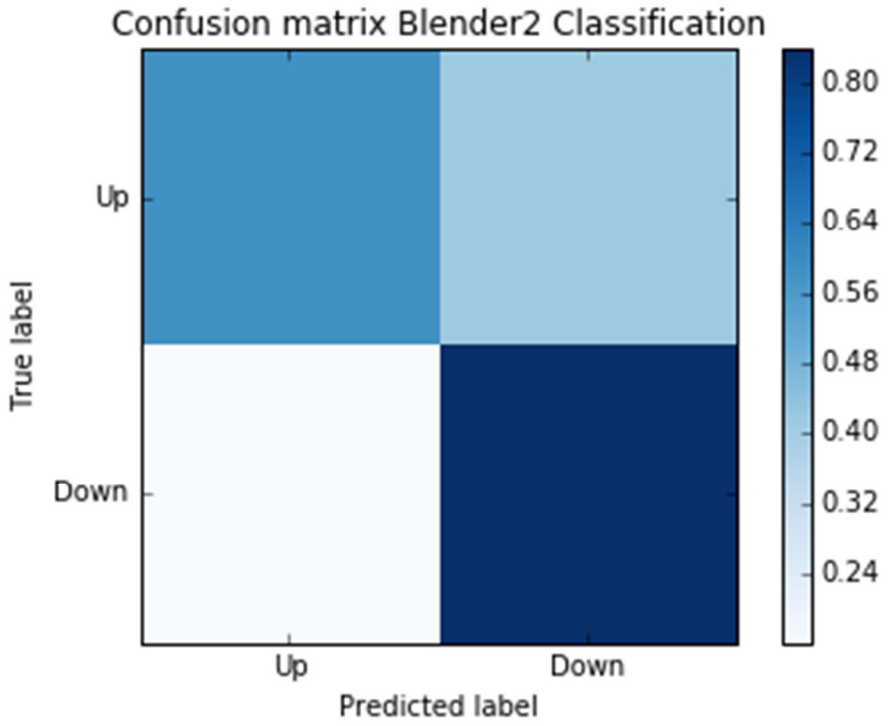
To train the estimators of second layer I use 3 years and to test 1 year, or 75%/25%

Algorithm	Accuracy	Parameters	version	F1Score	Feautures
BLE1	0,706349206	K-folds	Completed	0,741258741	SP Adjust Price 1 + Rest Adjust Rolling Average 5
BLE2	0,718253968	K-folds	Completed	0,756013746	SP Adjust Price 1 + Rest Adjust Rolling Average 5

Both estimators are better that it's optimized version using TEST 1 Dataframe

LRC got accuracy of 0.65 and here 0.71 and SVM 0.69 and the blended version 0.72. However the time to stack and blend is considerably superior and barely beat the best estimator on that test which was GTB with 0.72 too.





In a second test I use only GTB and Ada Boosting estimators for first layer, Iterate in a loop 250 times training and predicting, then with same configuration for blender 1 and 2 I got slightly better outcomes for blender 1

BLE1	0,734126984	K-folds	Completed	0,750929368	SP Adjust Price 1 + Rest Adjust Rolling Average 5
BLE2	0,702380952	K-folds	Completed	0,757281553	SP Adjust Price 1 + Rest Adjust Rolling Average 5

In my opinion investing some more time in tuning and researching probably I will get better outcomes

REFERENCE

[1] W. Huang et al., “Forecasting stock market movement direction with support vector machine,” Computers & Operations Research, 32, pp. 2513–2522005, 2005

[2] J. Moody, et al., “Learning to trade via direct reinforcement,” IEEE Transactions on Neural Networks, vol. 12, no. 4, Jul. 2001.

[3] Y.-A. L. B. Gianluca Bontempi, Souhaib Ben Taieb, Machine Learning Strategies for Time Series Forecasting, Machine Learning Group, Computer Science Department, Univerite Libre de Bruxelles.

[4] J. Zhang, “Applying time series analysis builds stock price forecast model,” Modern Applied Science, vol. 3, no. 5, 2009.

[5] A. M. Zvi Bodie, Alex Kane, Investments. McGraw-Hill, 2014.

[6] S. T. Narasimhan Jegadeesh, “Returns to buying winners and selling losers: Implications for stock market efficiency,” The Journal of Finance, vol. 48, no. 1, pp. 65–91, March 1993.

[7] C. Y. Z. Ben Jacobsen, “Are monthly seasonals real? a three century perspective,” 2012.

[8] ——, “The halloween indicator, ’sell in May and go away’: An even bigger puzzle,” October 2012.

[9] Long-Term Unemployment in the Great Recession. U.S. Congress Joint Economic Committee, 2010.

[10] “S&P Dow jones indices,” PDF, March 31 2015.

[11] Equity forecast: Predicting long term stock price movement using machine learning

Nikola Milosevic School of Computer Science, University of Manchester, UK

[12]Predicting Stock Price Direction using Support Vector Machines, Saahil Madge

[13] Stock Market Forecasting Using Machine Learning Algorithms, Shunrong Shen, Haomiao Jiang

Department of Electrical Engineering, Stanford University

http://scikit-learn.org

https://www.quantopian.com/posts/technical-analysis-indicators-without-talib-code

http://machinelearningmastery.com/time-series-data-visualization-with-python/

https://www.quantinsti.com/blog/volatility-and-measures-of-risk-adjusted-return-based-on-volatility/

http://www.quantatrisk.com/tag/time-series/

http://machinelearningmastery.com/ensemble-machine-learning-algorithms-python-scikit-learn/

http://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it

http://stackoverflow.com/questions/39156506/alpha-beta-for-linear-regression-calculations-output-nan

https://www.quantinsti.com/blog/volatility-and-measures-of-risk-adjusted-return-based-on-volatility/

https://www.quantopian.com/posts/realised-volatility-function

http://stackoverflow.com/questions/26366021/pandas-aligning-multiple-dataframes-with-timestamp-index

https://mlwave.com/kaggle-ensembling-guide/

http://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

http://machinelearningmastery.com/implementing-stacking-scratch-python/

https://www.quora.com/What-are-examples-of-blending-and-stacking-in-Machine-Learning

http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

https://mlwave.com/kaggle-ensembling-guide/

Machine Learning Engineer Nanodegree, Udacity

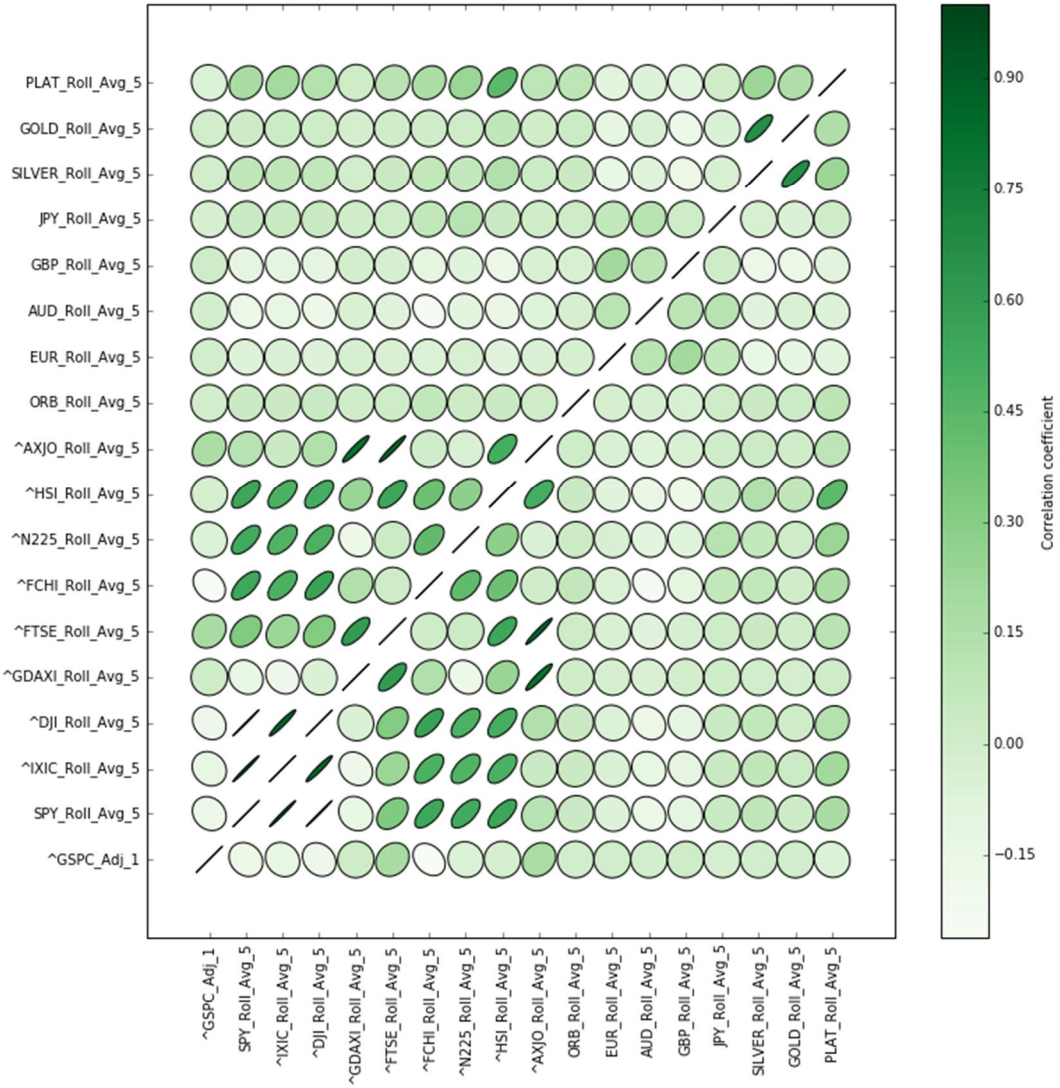
Machine Learning for Trading, Udacity

Annex 1

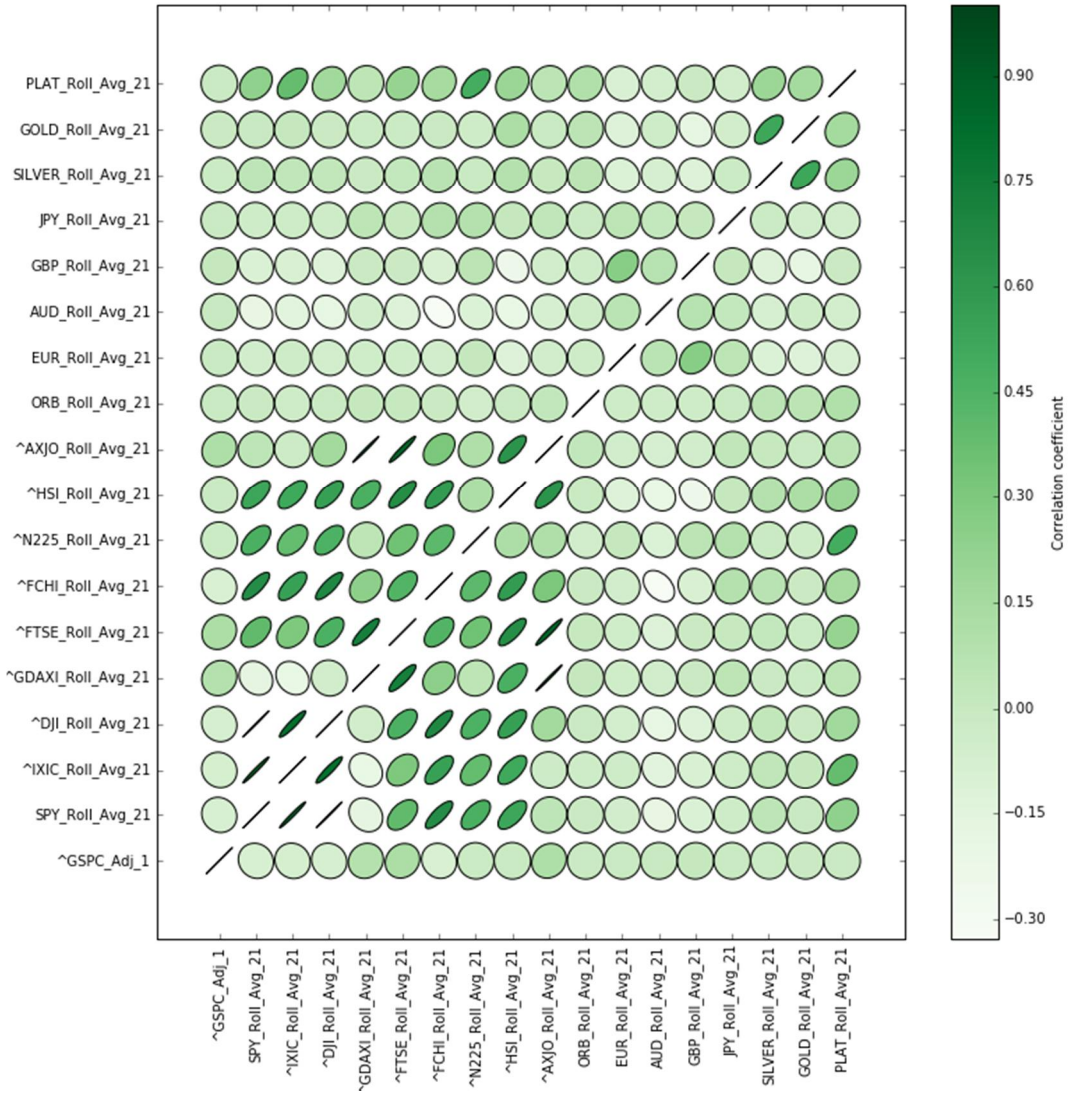
Exploratory Visualization Extended

In this annex, we are visualizing the correlation in between the different Rolling average series of Adjusted Price, Momentum, Volatility and Exponential Rolling Average of Adjusted prices, all with windows of 5, 21 and 63 days respectively. The reason of use this days is because 5 correspond to a week of trading on markets, 21 to a month and 63 to a quarter. It is true and some can argue that not all technical indicators like volatility can be used with such extended windows as 63 days, but this argument apply to very liquid assets and not indexes which take a long time to reflex the conditions of specific very volatile assets

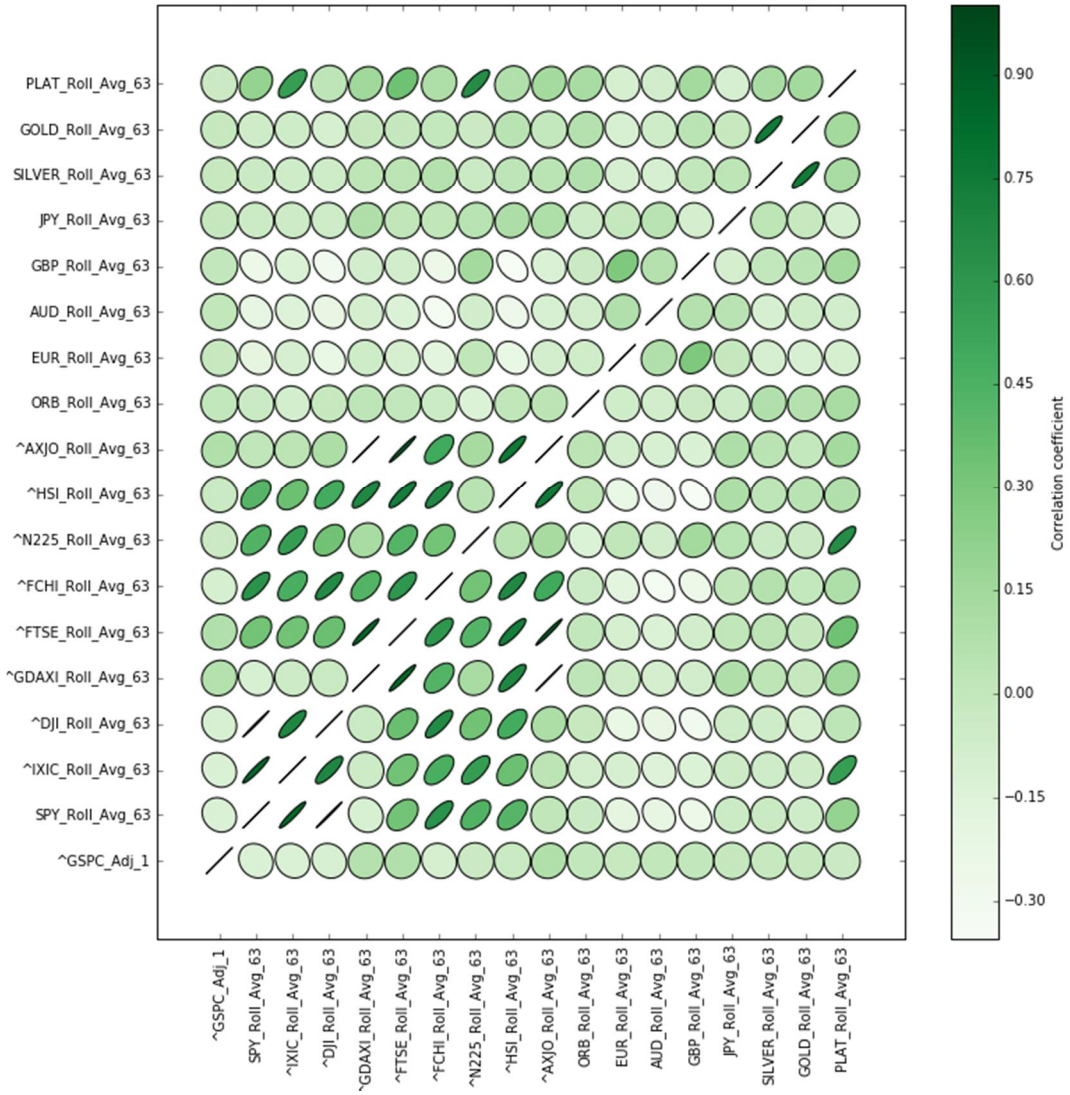
Correlation Rolling Average Adjusted Price Window 5 days



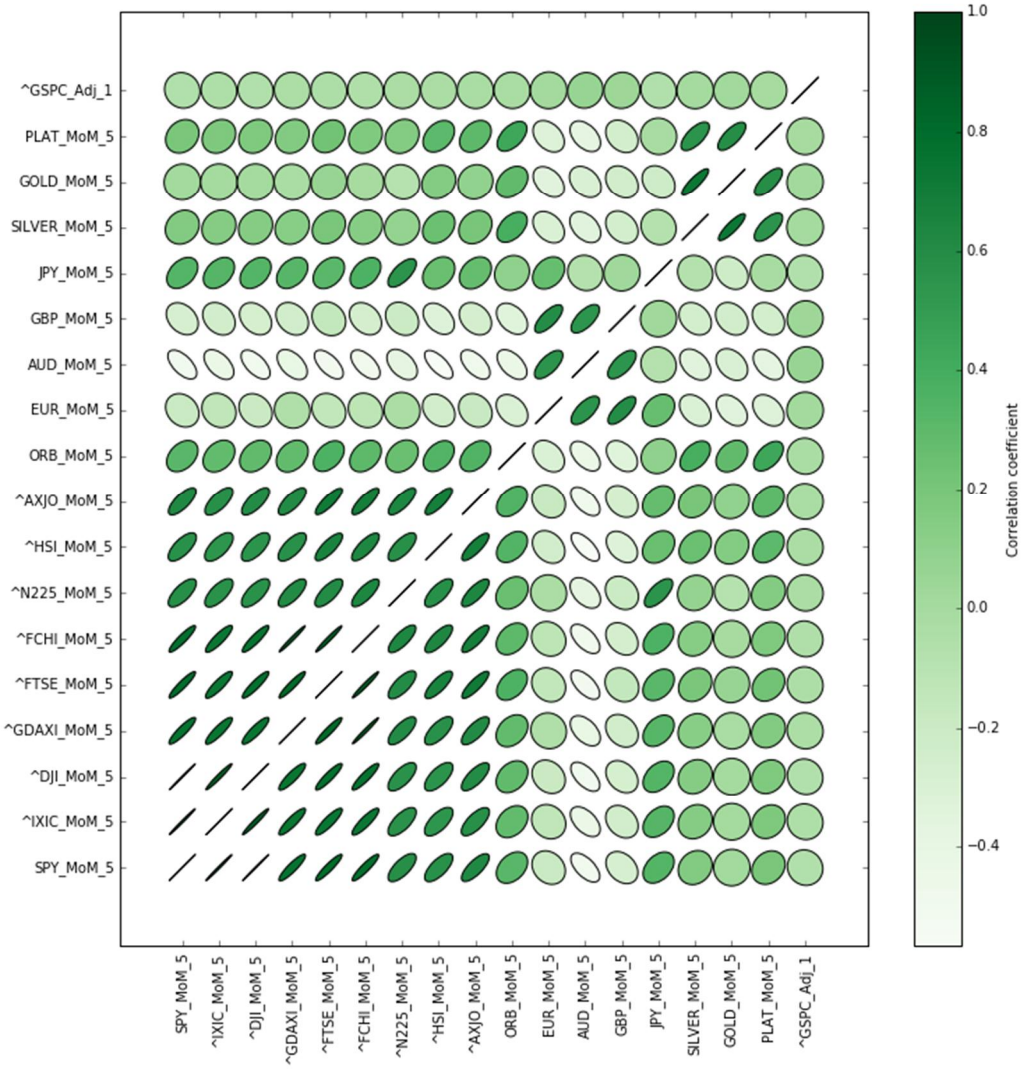
Correlation Rolling Average Adjusted Price Window 21 days



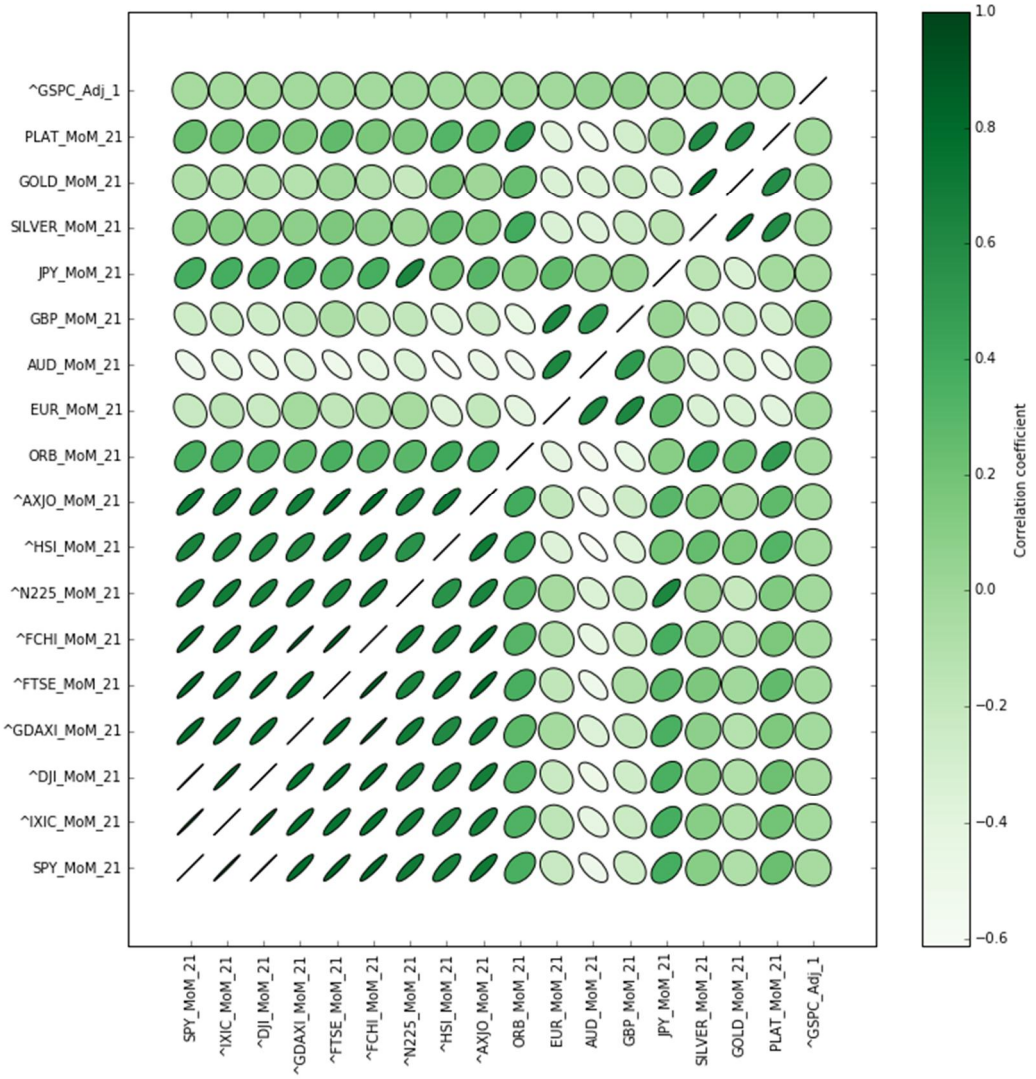
Correlation Rolling Average Adjusted Price Window 63 days



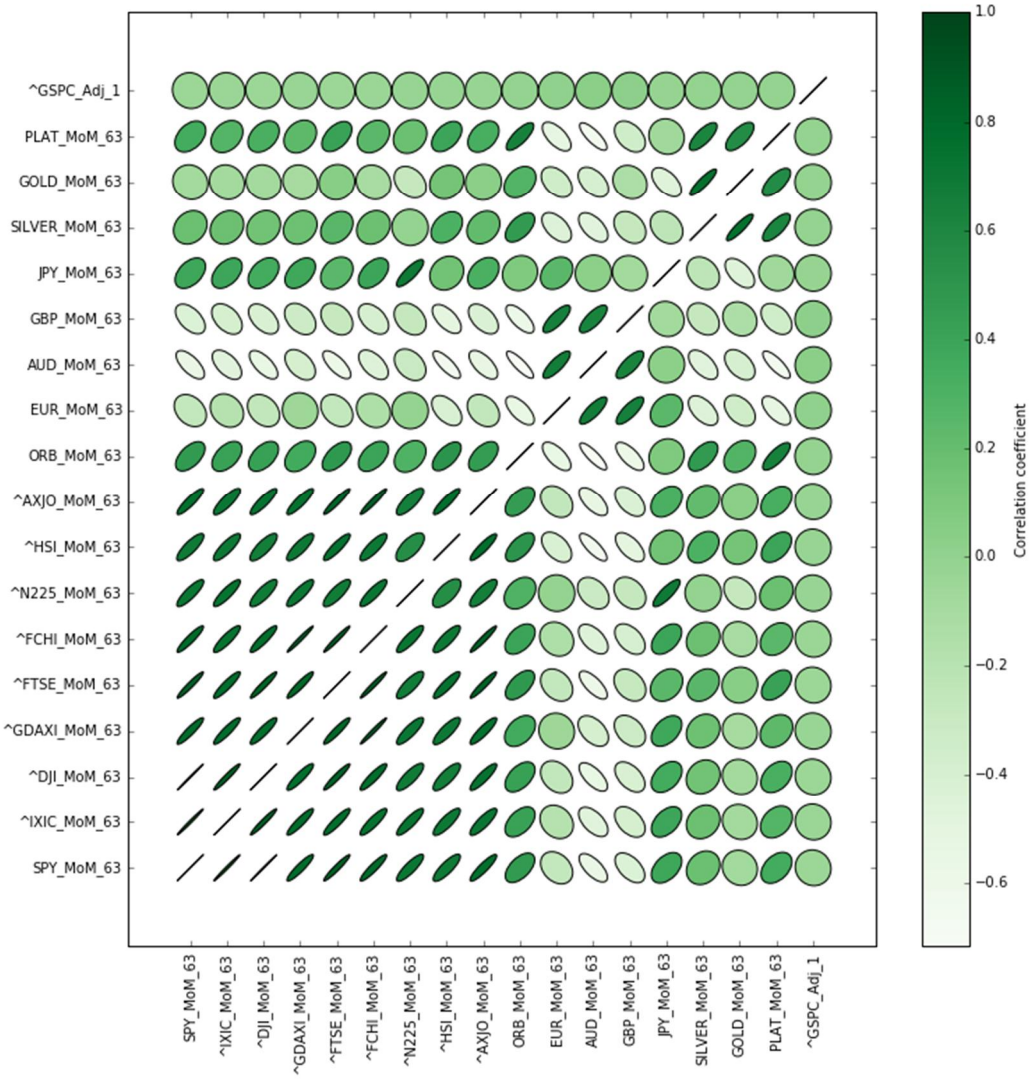
Correlation Rolling Average Momentum Window 5 days



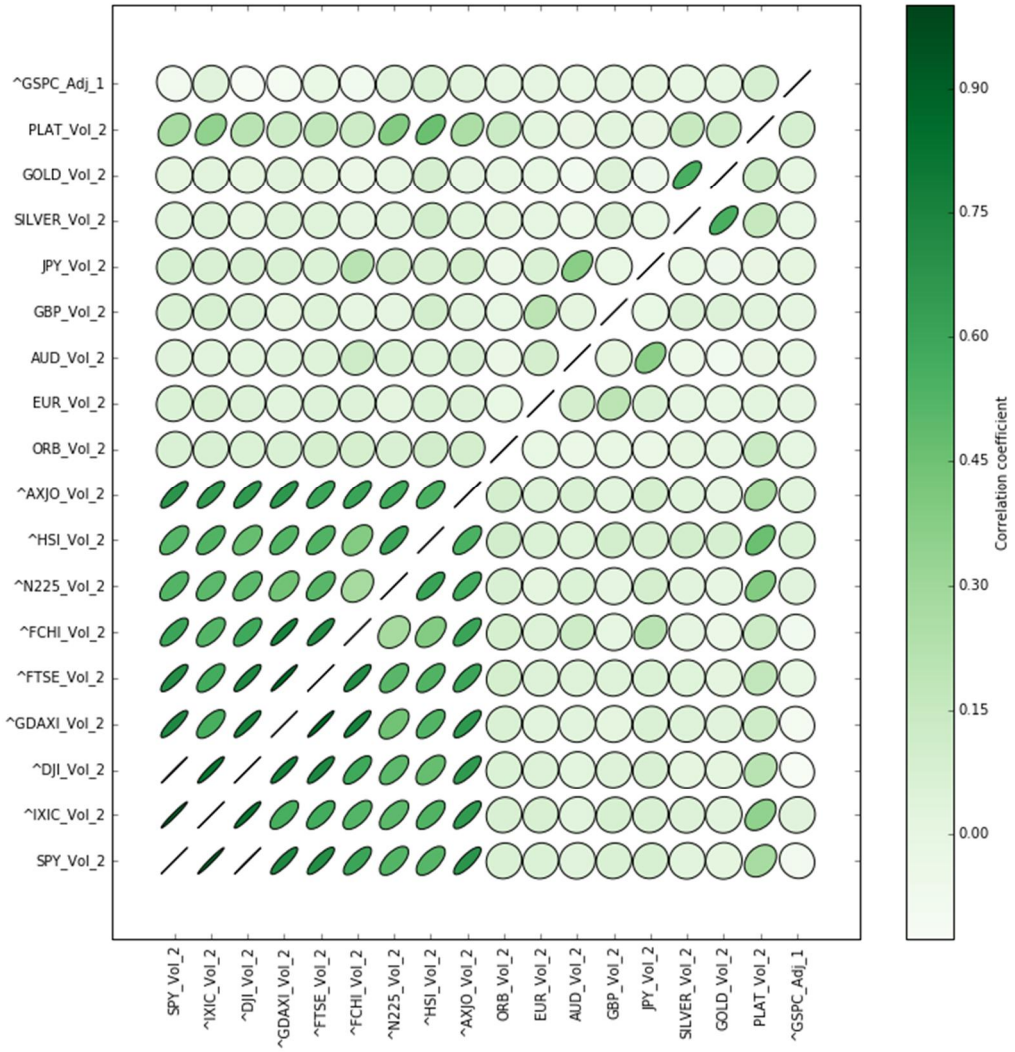
Correlation Rolling Average Momentum Window 21 days



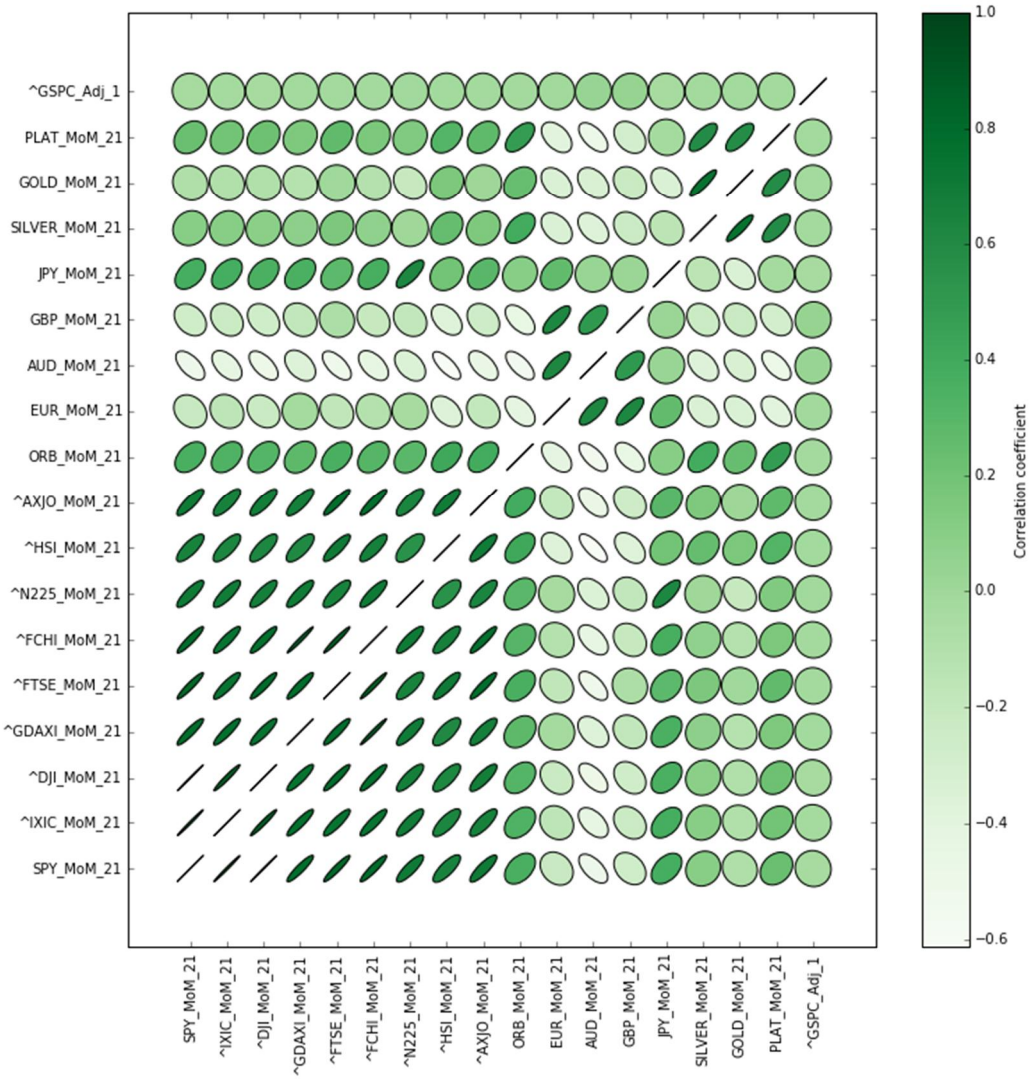
Correlation Rolling Average Momentum Window 63 days



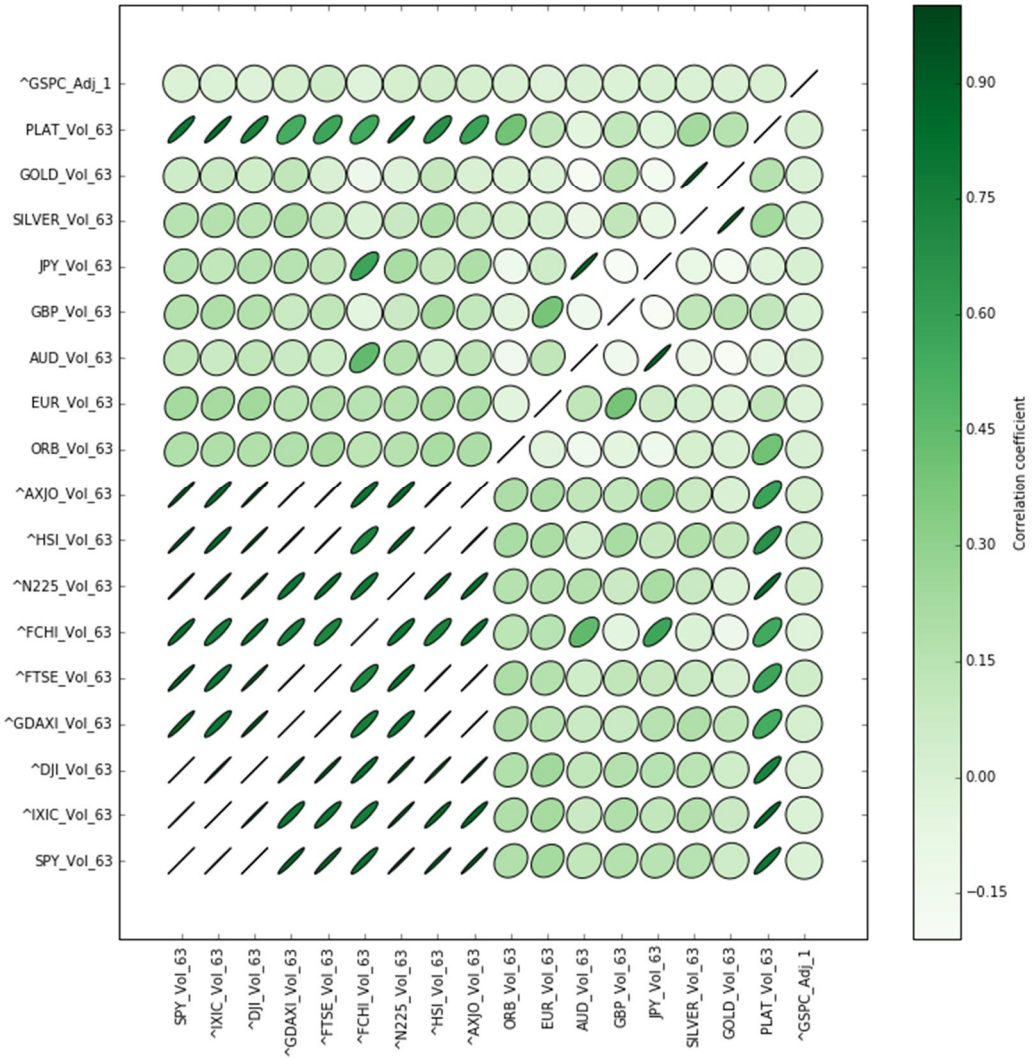
Correlation Rolling Average Volatility Window 2 days



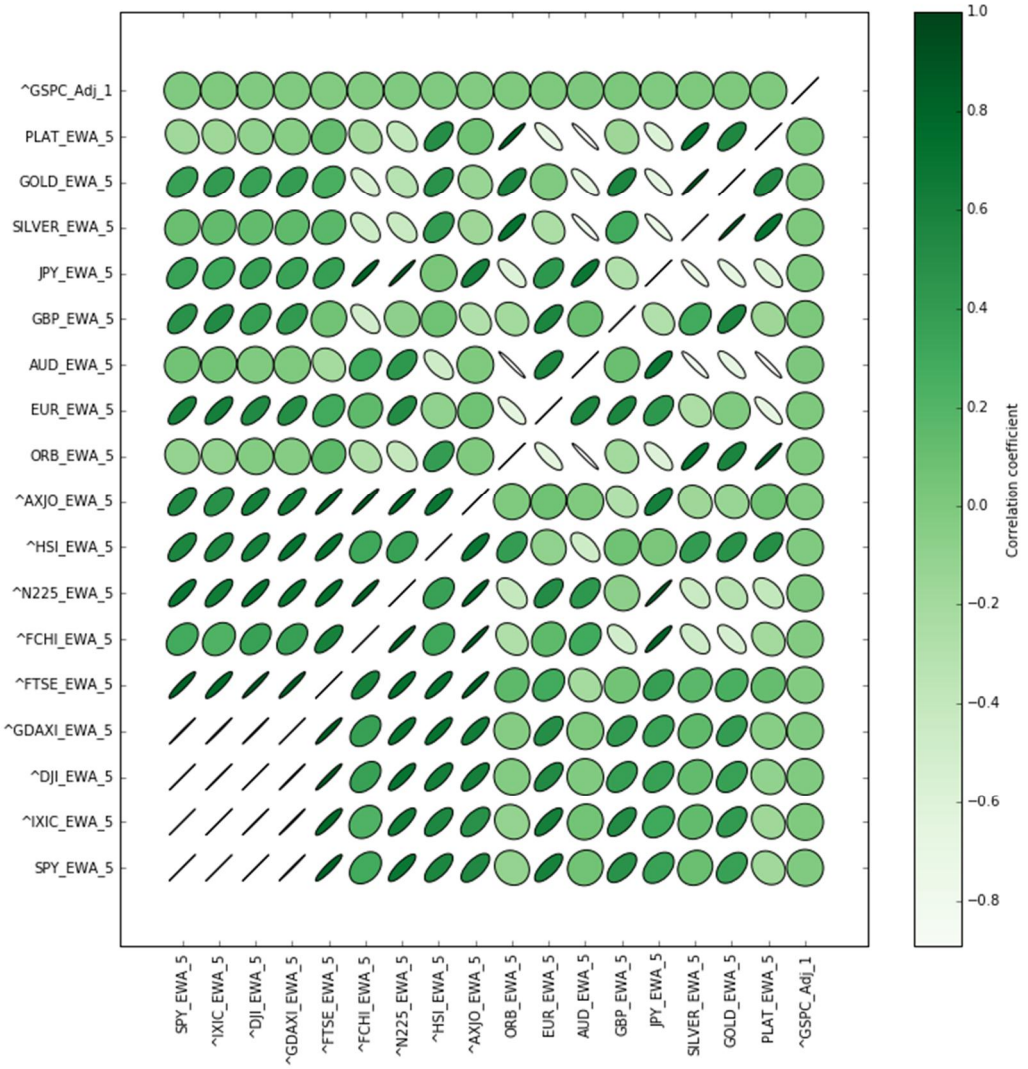
Correlation Rolling Average Volatility Window 21 days



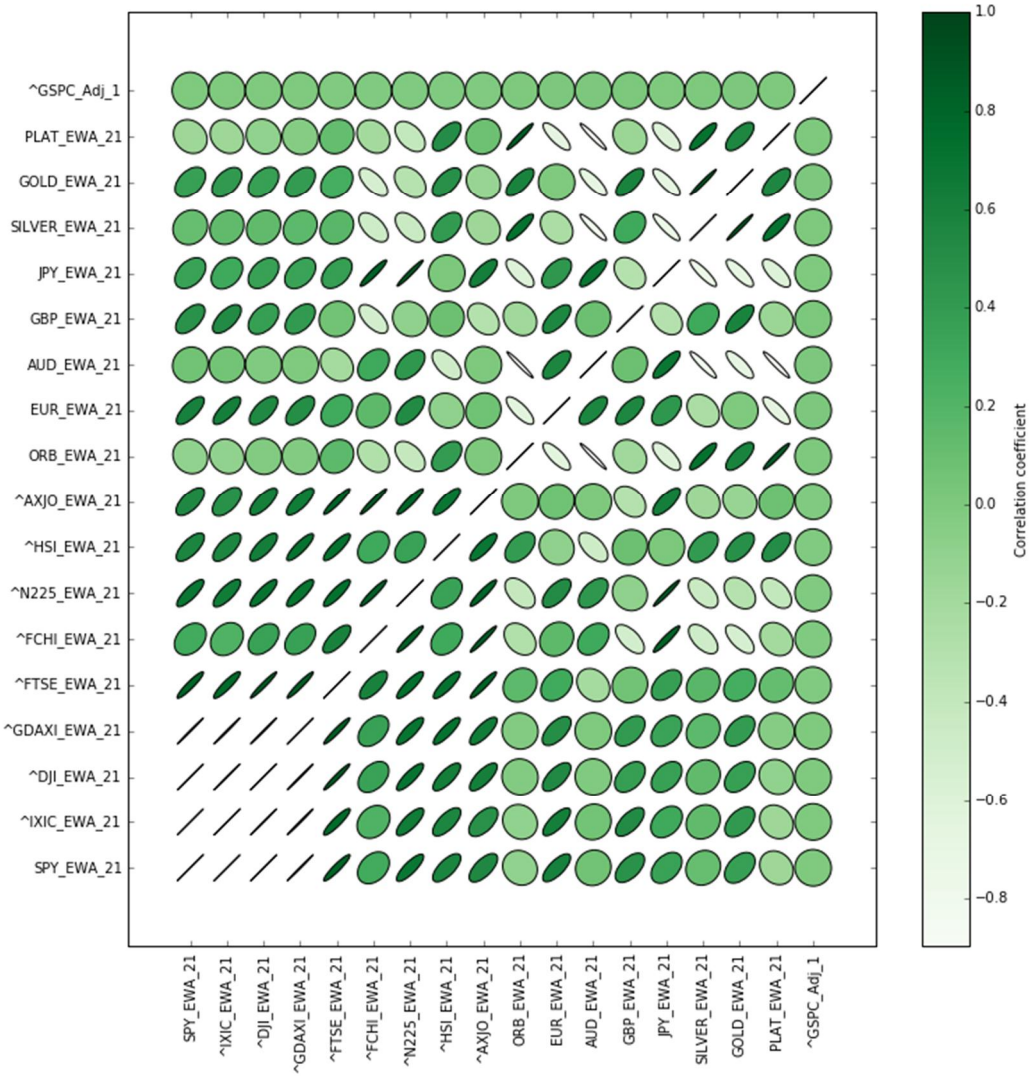
Correlation Rolling Average Volatility Window 63 days



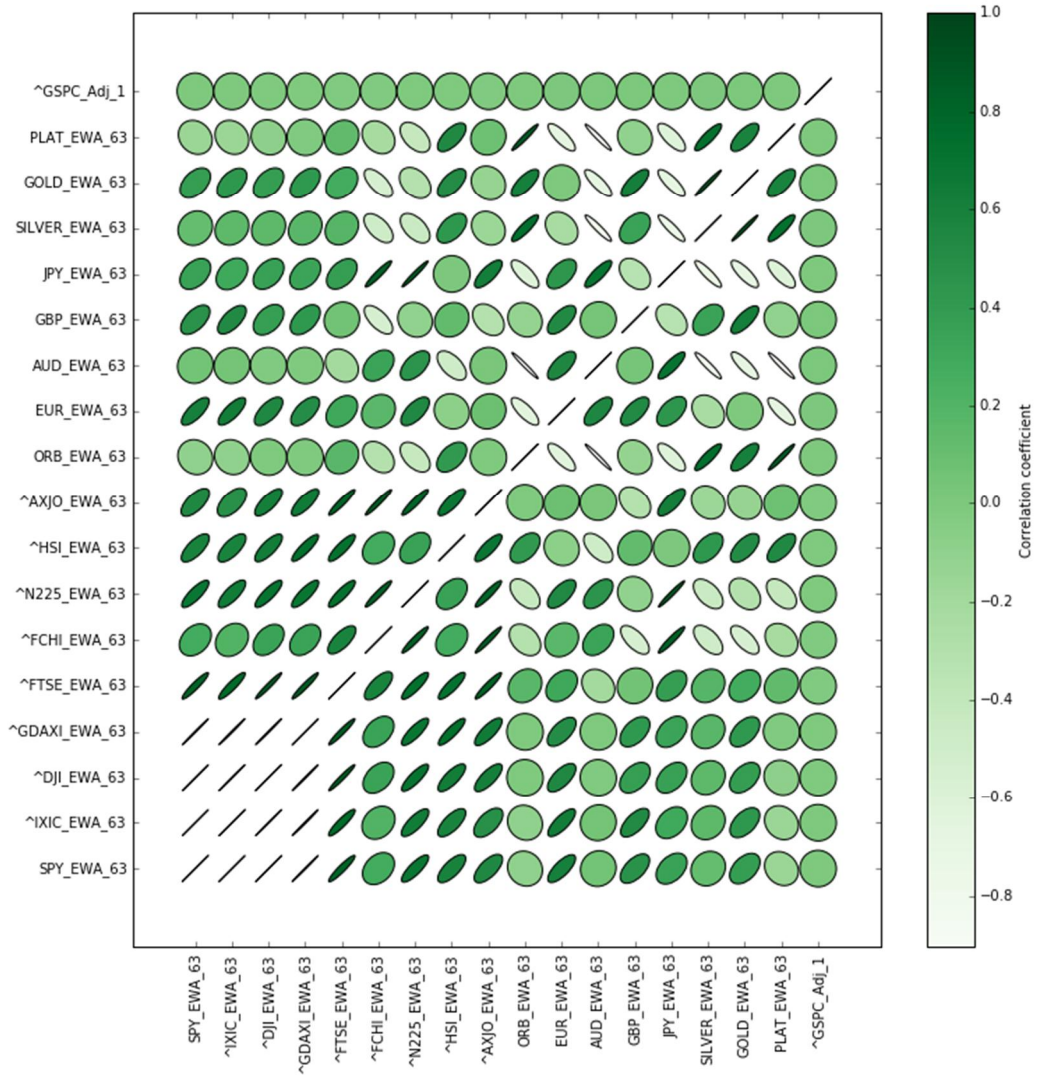
Correlation Exponential Rolling Average Window 5 days



Correlation Exponential Rolling Average Window 21 days



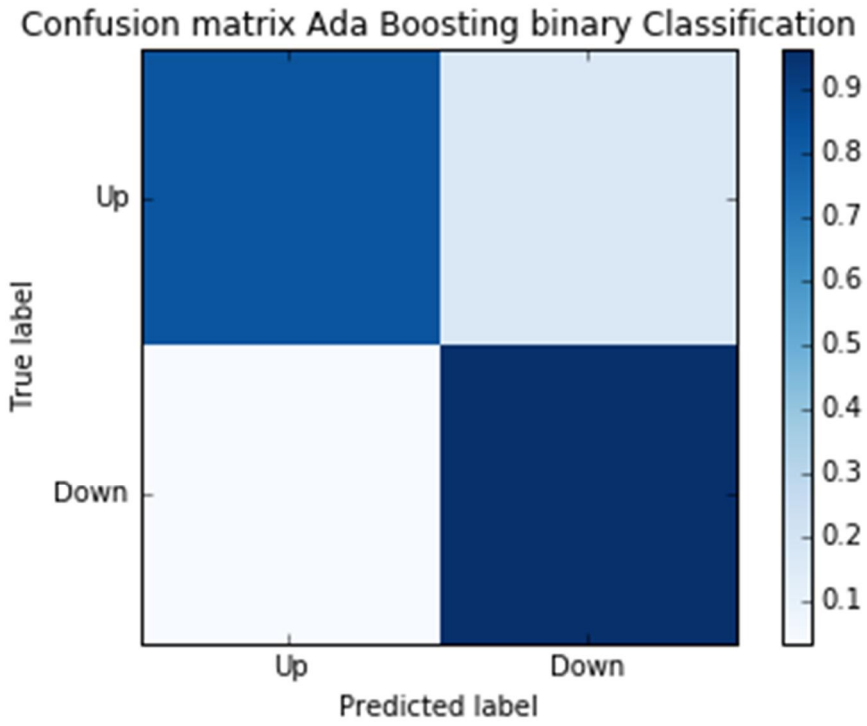
Correlation Exponential Rolling Average Window 63 days



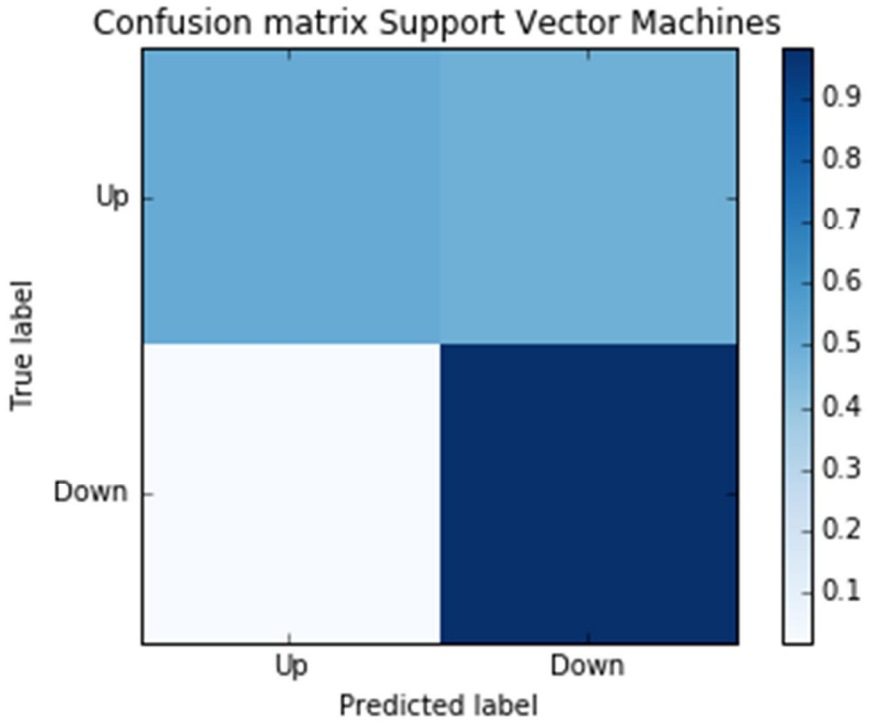
Annex 2 Confusion Matrix

Here I plot all the Confusion Matrix starting the better accuracy after top up to the worst accuracy. We can see than at least 4 algorithms perform on the 70% up, which is not bad. The worst are KNN and Naive Bayes, which probably are not correctly setup for this data

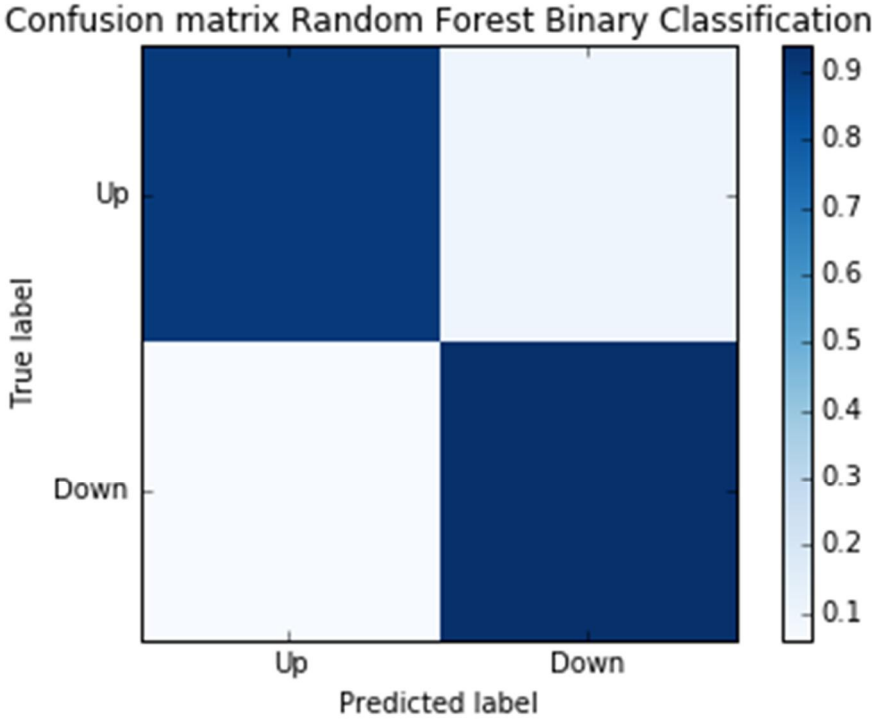
Random Ada Boosting Binary Classification



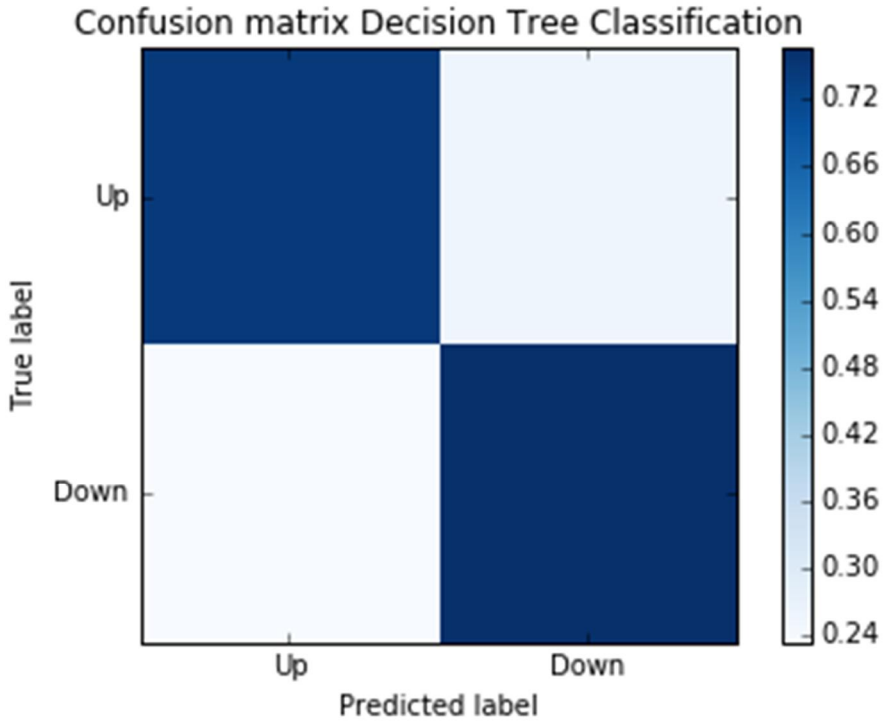
Support Vector Machines



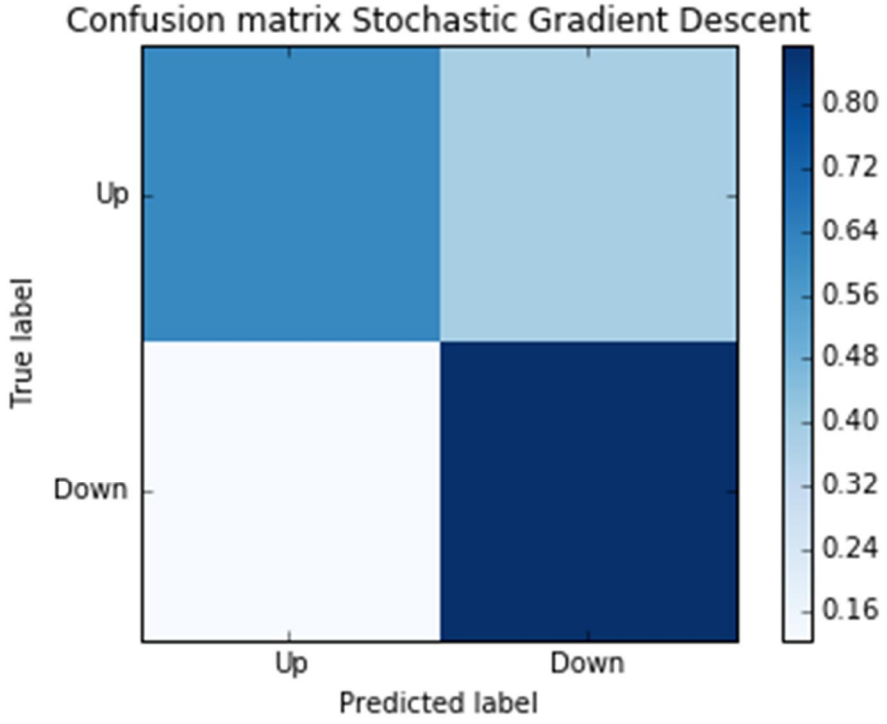
Random Forest Binary Accuracy



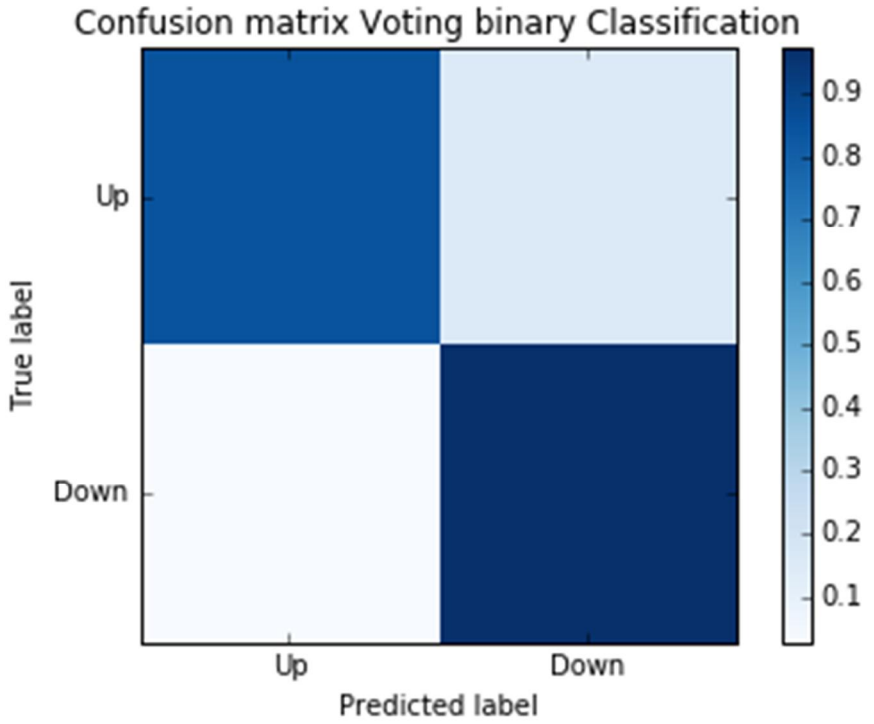
Decision Tree Classification



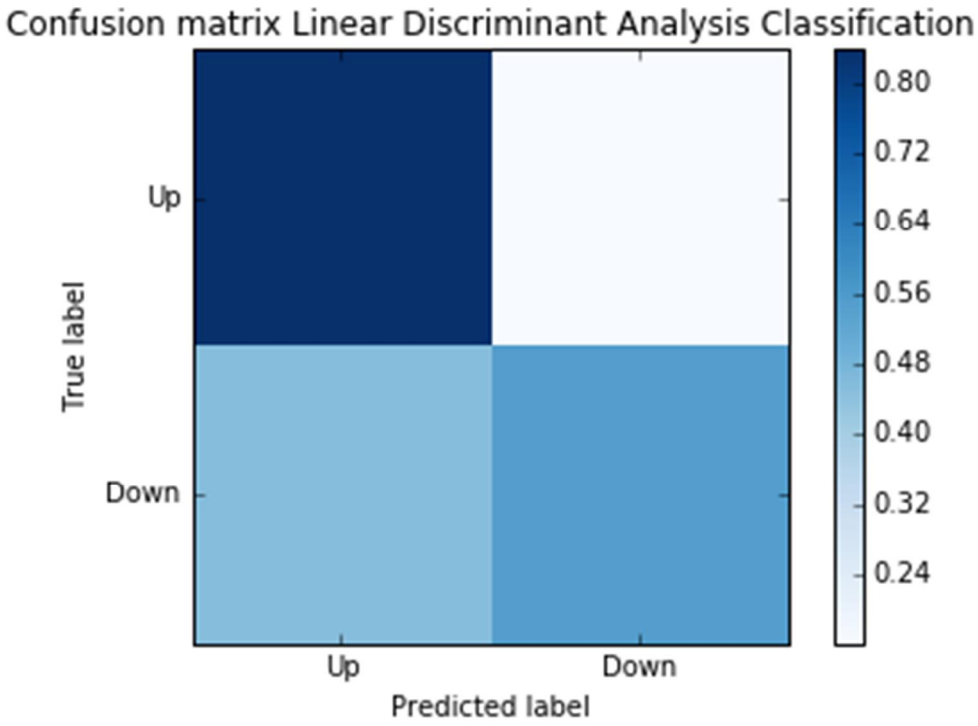
Stochastic Gradient Descent



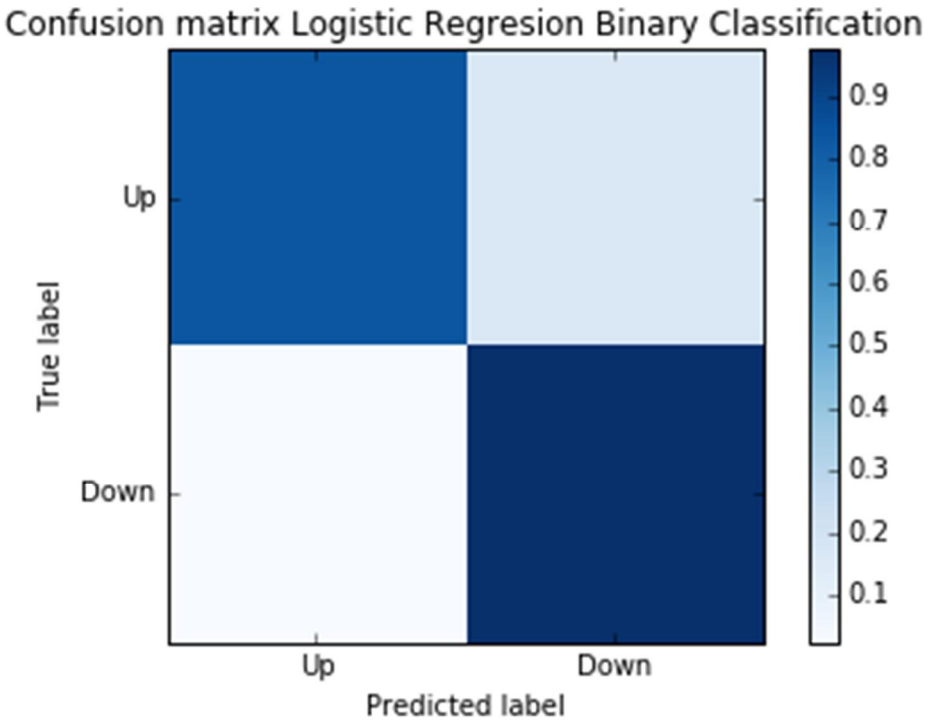
Voting Binary Classification



Linear Discriminant Analysis



Logistic Regression



K Nearest Neighbour Classification