COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MASTER

Portfolio management system based on technical indicators and artificial intelligence strategies

Autor: Jean Nadal
Director: Carlos Maté Jiménez

Madrid
Agosto de 2020

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título Portfolio management system based on technical indicators and artificial intelligence strategies

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Jean Nadal Fecha: 29/ 08/ 2020

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

[illegible]

Firma: Carlos Maté

Fdo.: Professor **CARLOS MATÉ JIMÉNEZ** Fecha: 31/08/2020

Department of Industrial Organization
Alberto Aguilera, 25 - 28015 Madrid
Tel. +34 91 542 28 00 Ext. 2430
cmate@comillas.edu;

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MASTER

Portfolio management system based on technical indicators and artificial intelligence strategies

Autor: Jean Nadal
Director: Carlos Maté Jiménez

Madrid
Agosto de 2020

AGRADECIMIENTOS

Quiero agradecer en primer lugar, a Carlos Maté Jiménez y a Jaime de Rábago Marín por todo lo aprendido durante el desarrollo de esta tesis al igual que por su seguimiento y ayuda a lo largo de este año. En segundo lugar a la Universidad Pontificia Comillas ICAI – ICADE por la oportunidad de poder realizar este proyecto y por la gran calidad de su enseñanza y profesorado, gracias a los cuales me he podido desarrollar a nivel personal y académico durante los últimos dos años. Por último, pero no por ello menos importante, un agradecimiento especial a mi familia y amigos por su apoyo incondicional durante toda mi etapa académica .

SISTEMA DE GESTION DE CARTERAS BASADO EN INDICADORES TECNICOS Y ESTRATEGIAS DE INTELIGENCIA ARTIFICIAL

Autor: Nadal, Jean.

Director: Maté Jiménez, Carlos.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Los mercados financieros tienen una importancia vital en el correcto funcionamiento de una sociedad desarrollada. Permiten asignar los recursos económicos de la forma más eficiente posible, proporcionan financiación a compañías y gobiernos y permiten canalizar el ahorro de los ciudadanos. Uno de los mercados financieros que más relevancia tienen es el mercado de acciones, donde las empresas que cotizan en bolsa emiten acciones para recibir financiación.

La inversión en este tipo de activos es especialmente atractiva para los inversores, principalmente por dos razones. Tiene un alto grado de liquidez, lo que permite que sea rápido y sencillo comprar y vender acciones, esto reduce el riesgo debido a que permite al inversor deshacerse de las posiciones que no desea mantener en su cartera. Esta ventaja es especialmente importante en situaciones de estrés en los mercados financieros, caracterizadas por caídas severas en la mayoría de activos. La segunda ventaja principal es que históricamente se trata del activo financiero que mayores retornos a proporcionado a los inversores a largo plazo, tal y como se aprecia en la Figura 1 que compara el retorno obtenido al invertir 10.000 dólares en diferentes activos de riesgo durante los últimos 200 años.

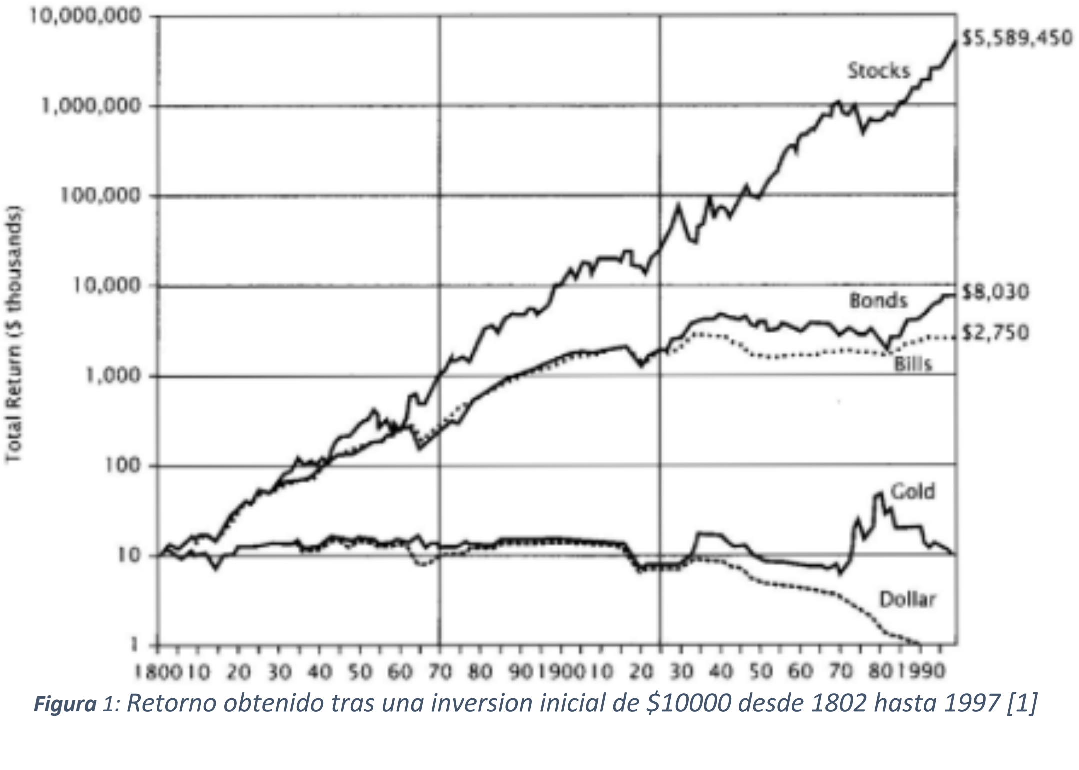


Figura 1: Retorno obtenido tras una inversion inicial de $10000 desde 1802 hasta 1997 [1]

La previsión de la evolución de los mercados financieros es una tarea extremadamente compleja, debido a la cantidad de variables y factores que tienen un efecto en el movimiento de los precios y debido a que se trata de una labor con una gran dependencia en fenómenos aleatorios.

El gran número de participantes en los mercados de acciones, aumenta la dificultad de conseguir resultados por encima de la media. Es necesario obtener mejores rendimientos que el resto de inversores para batir al mercado y hacerlo de forma recurrente es un trabajo complicado. Requiere de un profundo conocimiento de la inversion y de la psicología necesaria para no cometer errores y caer en los diferentes sesgos que pueden provocar malos resultados.

La creciente sofisticación de los mercados financieros y aceleración del ritmo al cual la sociedad evoluciona, de un punto de vista empresarial, del desarrollo de nuevas tendencias y tecnologías y de la aparición de nuevas empresas hace que parezca cada vez más difícil entender y desarrollar una visión sobre como será el mundo dentro de unas décadas.

Estos fenómenos tienen como consecuencia que sea complicado comprometerse con una estrategia de inversión para los próximos 20 o 30 años. No obstante, esto es lo que tiene que hacer cualquier inversor que quiera cumplir con sus objetivos financieros en el largo plazo.

Esta es seguramente una de las razones por las cuales la inversión de forma pasiva es cada vez más importante a nivel mundial. Cada vez más inversores consideran que este tipo de inversion es la solución más inteligente ya que les evita tener que dedicar demasiado tiempo a un problema que parece demasiado complejo, predecir como será el mundo del mañana.

Sin embargo, el creciente porcentaje de las inversiones canalizadas de forma pasiva, puede provocar la aparición de oportunidades de inversión, debido a la menor competencia por superar el rendimiento del mercado. Debido a que cada vez más dinero es invertido simplemente en las acciones con una mayor capitalización bursátil, el precio del resto de acciones no ha pasado a través del mecanismo de la oferta y la demanda que ajusta el precio de una acción al valor real de la compañía. Esto aumenta las ineficiencias del mercado y permite a aquellos inversores que sigan una estrategia activa comprar acciones a precios inferiores a su valor intrínseco.

El problema que trata de resolver este trabajo es por lo tanto determinar si es posible desarrollar una estrategia de inversión activa que sea capaz de aprovechar esas ineficiencias y superar de forma consistente en el largo plazo los rendimientos del mercado.

Dentro de la gestión activa, existen diferentes enfoques y una multitud de estrategias de inversion distintas. Un enfoque muy utilizado es realizando un análisis fundamental de la compañía, con el objetivo de medir su valor intrínseco mediante los balances e informaciones financieras de la empresa. Existen también metodologías basadas en análisis técnico o con un enfoque cuantitativo, en las cuales con la ayuda de indicadores basados en el precio de la acción y desarrollando modelos matemáticos y estadísticos se determina la futura evolución del precio de una acción.

Gracias al creciente desarrollo de herramientas de inteligencia artificial y mejoras en el manejo de grandes cantidades de datos, son cada vez más estudiadas las aplicaciones de este tipo de algoritmos a los mercados financieros. El objetivo de estas aplicaciones es aprovechar la alta capacidad de adaptación a nuevas condiciones y conjuntos de datos de estos modelos y su capacidad de detectar patrones para tratar de determinar la evolución de un activo financiero.

Para poder cumplir el objetivo principal propuesto en este trabajo, es necesario en primer lugar desarrollar estrategias de inversión y a continuación comprobar su validez. Es necesario demostrar que las estrategias de inversión permiten obtener resultados satisfactorios de forma consistente.

Para ello se han recopilado los precios diarios de las acciones que componen 8 de los mercados financieros desarrollados más importantes, el S&P 500, el DAX 30, el CAC 40, el Mercado Continuo Español, el FTSE MIB, el FTSE 100, el S&P TSX 60 y el ATX. Los datos se remontan al 1 de septiembre de 2010 y llegan hasta el 30 de agosto de 2019. La construcción de esta base de datos se realiza gracias al provedor de datos financieros SIX Financial Information.

A continuación se ha desarrollado una herramienta en MATLAB que permita hacer el backtesting de las estrategias desarrolladas sobre esa base de datos. Esta herramienta permitirá recuperar y realizar el pre-procesamiento de los datos de cada acción, crear las estrategias de inversión, aplicarlas al conjunto de datos seleccionado, realizar la gestión de la cartera con las acciones obtenidas gracias a la estrategia y finalmente analizar los resultados conseguidos.

Uno de los peligros que puede aparecer al realizar el backtesting de una estrategia, es que haya funcionado por casualidad durante el periodo de tiempo seleccionado y con las acciones que se han escogido. Para obtener una estrategia más robusta se ha separado el periodo temporal utilizado en dos, un primer periodo de aproximadamente 7 años y otro de 2 años. Además de esto se realiza la prueba de forma independiente en cada uno de los 8 índices seleccionados. La idea es por lo tanto comprobar si la estrategia ha funcionado en ambos periodos de tiempo y en todos los mercados financieros, independientemente de las condiciones particulares de cada uno de ellos. De esta forma se obtiene una estrategia más robusta, capaz de funcionar de forma satisfactoria en el mayor numero posible de entornos distintos.

Se han seleccionado cinco estrategias de inversión, debido a los buenos resultados que han proporcionado en todos los mercados financieros seleccionados y en ambos periodos de tiempo.

Todas las estrategias son sistemáticas, es decir que es necesario definir unas ciertas reglas basadas en parámetros o algún tipo de algoritmo que genere de forma automática las señales de compra o de venta de las acciones. Una de las ventajas de este tipo de estrategias se basa en que en numerosas ocasiones es el propio inversor el que entorpece su éxito en los mercados financieros. Ya sea porque toma demasiadas decisiones, porque considera que sus habilidades para predecir la evolución de los mercados financieros son superiores a la media y a lo que realmente son, o porque es victima de alguno de sus sesgos. Estableciendo una estrategia sistemática, se elimina un gran numero de decisiones, en particular las más peligrosas, aquellas que se toman en momentos de crisis, cuando una acción ha sufrido una caída importante. En esos momentos es normal que el inversor tenga dudas acerca de cual es la mejor decisión que puede tomar. Gracias a una estrategia sistemática todas esas dudas desaparecen. Aún hay que tomar ciertas decisiones, cuales son las reglas que hay que seguir, o en que se basa el algoritmo, no obstante este proceso se puede realizar de forma tranquila, con tiempo, sin la presión de tener que actuar de forma inmediata.

La primera de las estrategias, se basa en la combinación de dos reglas. La primera de ellas se basa en el RSI, Relative Strength Index, de la acción. Se trata de un indicador que permite medir la magnitud de las tendencias a corto plazo de una acción. Gracias a esta regla se pretende determinar que acciones están siendo demasiado compradas recientemente y cuales de ellas están siendo demasiado vendidas. La idea es que cuando una acción proporciona una de estas señales, su tendencia es revertir a la media. Por lo que una acción que ha sido demasiado comprada en los últimos días y cuyo precio por lo tanto ha subido, generalmente suele ser vendida en los días siguientes haciendo que baje el precio. La segunda de las reglas se basa en el retorno anual obtenido por la acción. Se clasifican las acciones de menor a mayor retorno y se invierte en aquellas que tengan conseguido un mayor retorno en los últimos 12 meses. Ambas reglas tratan de aprovecharse de un factor llamado momentum. Este fenómeno se basa en la idea de que el mercado se mueve por tendencias, las cuales perduran en el corto a medio plazo. Esta estrategia se ha llamado por lo tanto Pure Momentum Strategy.

La segunda estrategia también se basa en la combinación de dos reglas. La primera de ellas es una vez más el retorno anual obtenido por la acción, generando una señal de compra para aquellas acciones que hayan obtenido un mayor retorno en ese periodo de tiempo. La segunda regla clasifica a las acciones en función de la volatilidad que tengan con respecto al mercado. Este indicador permite saber como se ha comportado la acción frente a movimientos del índice al que pertenece. La idea que se ha querido poner a prueba es que históricamente aquellas acciones cuya beta es menor, es decir que sus variaciones suelen ser menores en comparación a las del mercado, han obtenido mayores retornos que aquellas con una beta mayor. Por lo tanto la regla emite una señal de compra para aquellas acciones con menor beta. Al tratarse de una estrategia que pretende aprovecharse del momentum y de la baja beta de las acciones se ha llamado, Momentum and Low Beta Strategy.

La tercera estrategia utiliza un modelo de inteligencia artificial. Se basa en el algoritmo de Machine Learning conocido como clasificador de Naive Bayes. Se trata de un clasificador probabilístico que permite identificar a que sub conjunto de categorías pertenece una muestra mediante una aplicación del teorema de Bayes. De esta forma el objetivo del algoritmo es clasificar cada acción determinando si se debe comprar, vender o mantener. Para ello el modelo necesita una serie de indicadores para cada acción, indicadores temporales, retornos de la acción a diferentes plazos, el RSI y finalmente el MACD de la acción. El MACD es un indicador que permite determinar el momentum de una acción comparando la relación entre dos medias móviles de diferentes plazos y el precio de la acción. Esta estrategia se ha llamado Naive – Bayes Classifier Strategy.

La cuarta estrategia se fundamenta en un algoritmo de Deep Learning. Se trata de una red neuronal de tipo Long Short – Term Memory la que proporciona las señales de compra y de venta de cada acción. Se trata de una red neuronal recurrente, es decir que tiene bucles cerrados que van de las salidas a las entradas. De esta forma se permite que la información persista durante ciertas iteraciones, una parte de los datos de las salidas previas se convierta en las entradas de las entradas de la siguiente iteración, de esta forma la información es recordada, haciendo que parezca que la red neuronal tiene una memoria, de ahí su nombre. Estos bucles permiten además que la red neuronal sea capaz de mostrar comportamientos temporales, lo que la convierte en una herramienta adecuada para la toma de decisiones basadas en series temporales. Este algoritmo necesita también un cierto número de indicadores sobre los cuales hacer las previsiones. Se ha desarrollado el modelo de forma que utilice los mismos indicadores que la estrategia basada en el clasificador de Naive – Bayes. Además de las señales proporcionadas por el modelo, esta estrategia realiza un filtrado adicional basado en la regla de la beta de la segunda estrategia. De esta forma se privilegian también aquellas acciones con una beta menor. Esta estrategia se ha llamado LSTM and Low Beta Strategy.

La última estrategia utiliza un modelo diferente de Deep Learning. Se basa en una red neuronal más simple llamada Feedforward. Esta red neuronal tiene una arquitectura abierta, sin retroalimentación de las salidas hacia las entradas. La información se mueve de las entradas a las salidas de forma unidireccional y hacia delante, lo que da el nombre a la red neuronal. Se trata de la más simple de las redes neuronales. Para que la red neuronal funcione de forma satisfactoria requiere también de un cierto numero de indicadores, se ha decidido utilizar los mismos que en las dos estrategias anteriores. Finalmente esta estrategia también añade el filtro adicional basado en la beta de las acciones. Esta estrategia se ha llamado FeedForward Neural Network and Low Beta Strategy.

Finalmente se ha desarrollado una herramienta que permite la optimización de las carteras, independientemente de la estrategia utilizada. El eventual uso de esta herramienta es una de las decisiones de inversión que se deben tomar y cualquiera de las estrategias puede funcionar tanto haciendo uso de la optimización como sin ella. Esta herramienta permite determinar de forma optimizada la asignación de capital a cada una de las acciones en cartera, de forma que el Sharpe ratio estimado sea máximo. El Sharpe ratio es un indicador muy utilizado para determinar si una estrategia de inversion ha obtenido buenos resultados ya que permite comparar el retorno obtenido con el riesgo asumido.

A continuación se muestra un ejemplo con el resultado obtenido al realizar el backtesting en el segundo periodo temporal de Julio 2017 a Agosto 2019 con las acciones del FTSE 100 mediante la estrategia basada en el clasificador de Naive – Bayes,

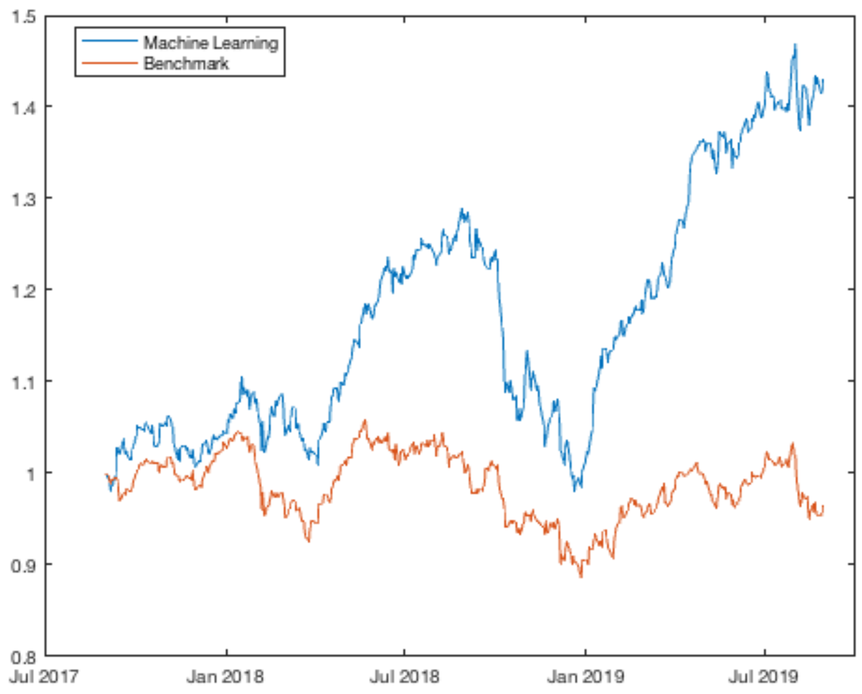


Figura 2: Equity Curve – Naive – Bayes Classifier – FTSE 100

	Sharpe Ratio	P&L Final	Max Drawdown Estrategia	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
FTSE 100	1,2303	14.300.900	23,98%	16.41%	7,99e-02%	10,90%	0,74%

Tabla 1: Resumen indicadores de rendimiento – Naive Bayes Classifier Strategy

Una vez que se han seleccionado las estrategias de inversión que han sido capaces de proporcionar buenos resultados de forma consistente, es decir superando una estrategia pasiva en ambos periodos temporales y en todos los mercados financieros, se implementan las estrategias con los datos financieros de cada acción en tiempo real.

La herramienta para la implementación de las estrategias y la gestión de las carteras en tiempo real se ha realizado también en MATLAB y siguiendo la misma estructura que en la herramienta de backtesting. La principal diferencia entre ambas radica en la forma en la que se consiguen los datos. Debido a que para la herramienta en tiempo real es necesario obtener de forma habitual los últimos datos con las cotizaciones de las empresas que pertenezcan al universo de inversión, se ha decidido utilizar Yahoo – Finance como provedor de los datos. Para ello la herramienta es capaz de descargar los datos directamente desde MATLAB.

A continuación se muestran a día 27 de agosto 2020 la evolución de las carteras obtenidas gracias a la implementación de cada una de las estrategias,

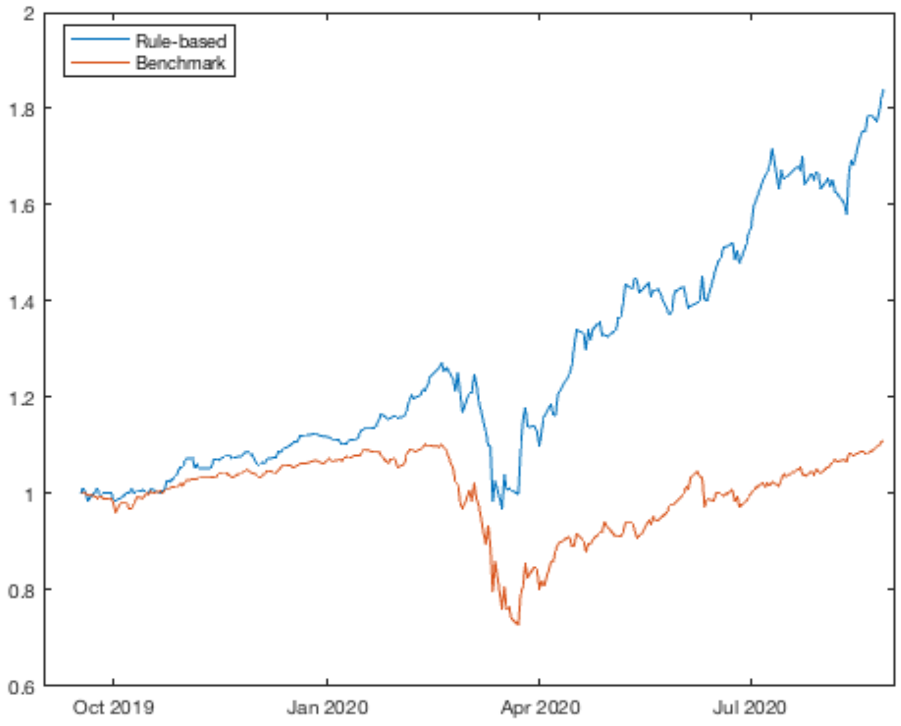


Figura 3: Equity curve Momentum Strategy – 27 Agosto 2020

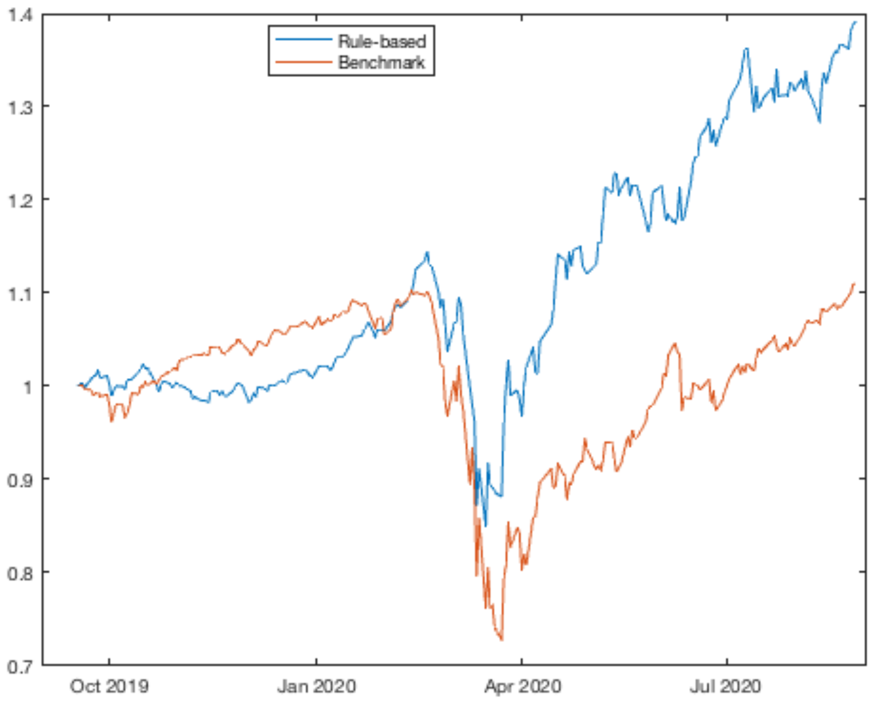


Figura 4: Equity curve Momentum – Low Beta Strategy – 27 Agosto 2020

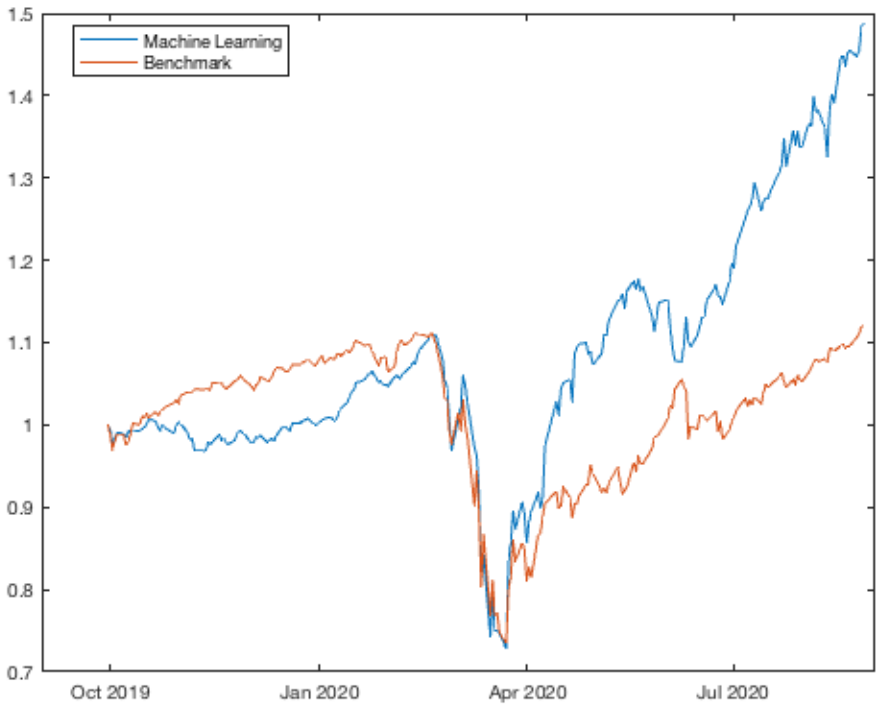


Figura 5: Equity curve Naïve Bayes Classifier Strategy – 27 Agosto 2020

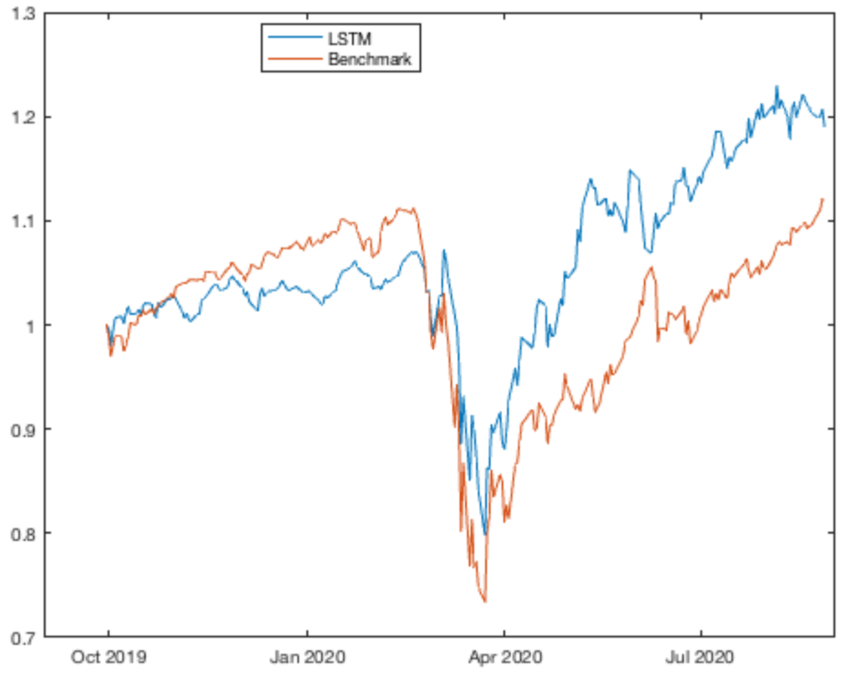


Figura 6: Equity curve – LSTM Low Beta Strategy – 27 Agosto 2020

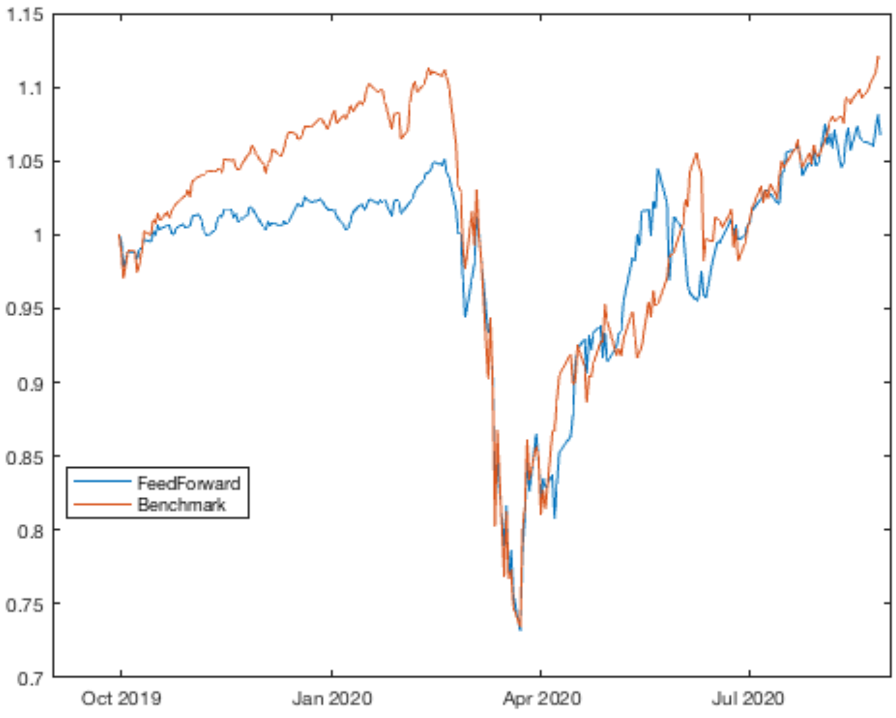


Figura 7: Equity curve - Feedforward Strategy Low beta – 27 Agosto 2020

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
Pure Momentum	2,3018	18.409.400	23,70%	34,01%	0,24%	13,25%	1,64%
Momentum – Low Beta	1,4192	13.904.800	25,73%	34,01%	0,12%	6,18%	1,49%
Naive - Bayes	1,4097	11.400.800	35,41%	34,01%	0,15%	8,59%	1,55%
LSTM – Low Beta	0,8302	11.895.600	25,55%	34,01%	5,02e-02%	1,44%	1,52%
Feedforward – Low Beta	0,4080	10.665.860	30,38%	34,01%	3,02e-04%	-1,97%	1,46%

Tabla 2: Resumen de los indicadores de rendimiento obtenidos a día 27 de Agosto 2020

A la vista de los resultados obtenidos se concluye que, a excepción de la estrategia basada en la red neuronal Feedforward, las estrategias de inversión han obtenido muy buenos resultados hasta el día 27 de Agosto 2020. Cabe destacar que aunque los resultados obtenidos con la última estrategia de inversión no son superiores a los de su índice de referencia, son muy similares y se han obtenido con una menor volatilidad.

Una vez finalizado el proyecto se puede concluir que el objetivo principal que se había propuesto se ha cumplido. Se han conseguido desarrollar estrategias de inversion basadas en gestión activa capaces de obtener resultados mejores que aquellos obtenidos mediante la inversión pasiva en un índice de referencia global. Dicho objetivo no solo se ha cumplido del punto de vista de los retornos obtenidos sino también del riesgo asumido. Las estrategias desarrolladas son capaces de obtener un buen rendimiento manteniendo un nivel de volatilidad similar al del mercado y en ocasiones incluso menor.

PORTFOLIO MANAGEMENT SYSTEM BASED ON TECHNICAL INDICATORS AND ARTIFICIAL INTELLIGENCE STRATEGIES

Author: Nadal, Jean.

Director: Maté Jiménez, Carlos.

Collaborating Institution: ICAI – Universidad Pontificia Comillas.

PROJECT SUMMARY

Financial markets play a vital role in every capitalist economy. They allow economic resources to be allocated as efficiently as possible, provide financing to companies and governments and enable the channelling of individual savings. The stock market is one of those financial markets, where listed companies issue shares to receive financing.

The investment in this type of asset is particularly attractive for investors, mainly for two reasons. It has a high degree of liquidity, which makes it fast and easy to buy and sell shares, this reduces the risk as it allows the investor to get rid of the positions he does not want to hold in his portfolio. This advantage is especially important in stressful situations in the financial markets, characterised by severe declines in most assets. The second main advantage is that it has historically been the financial asset that has provided the highest returns to investors over the long term, as can be seen in Figure 1 which compares the return obtained by investing $10,000 in different risk assets over the last 200 years.

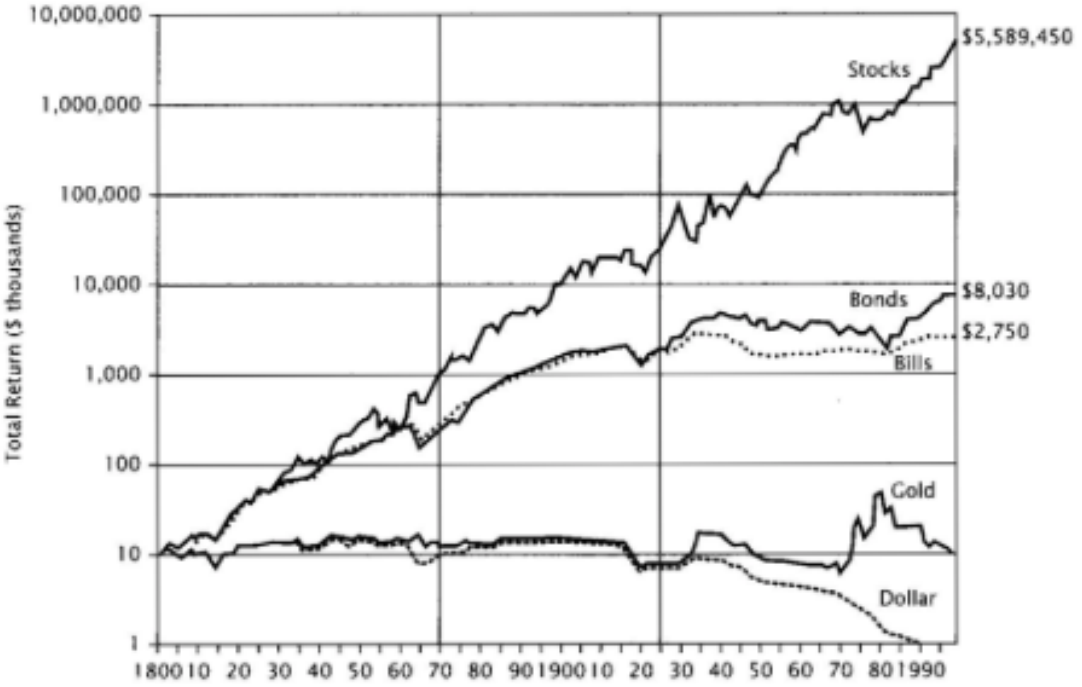


Figure 8: Total real return on $10000 initial investment from 1802 to 1997 [1]

The forecasting of financial market is an extremely complex task given the number of variables and factors that have an effect on price movements and because it is highly dependent on random phenomena.

The large number of participants in the stock markets increases the difficulty of achieving above-average results. It is necessary to obtain better returns than other investors in order to beat the market and doing so on a recurring basis is a complicated task. It requires a deep understanding of investment and the psychology necessary to avoid making mistakes and falling into the various biases that can lead to poor results.

The growing sophistication of the financial markets and the acceleration of the pace at which society is evolving, from a business point of view, the development of new trends and technologies and the appearance of new companies make it increasingly difficult to understand and develop a vision of what the world will be like in a few decades.

These phenomena make it difficult to commit to an investment strategy for the next 20 or 30 years. However, this is what any investor who wants to meet his financial goals in the long term has to do.

This is surely one of the reasons why passive investment is becoming increasingly important worldwide. More and more investors see this type of investment as the most intelligent solution as it avoids them having to spend too much time on a problem that seems too complex, predicting what tomorrow's world will be like.

However, the increasing share of investments that are passively channelled may lead to the emergence of investment opportunities due to reduced competition for market returns. Because more and more money is simply invested in shares with a larger market capitalisation, the price of other shares has not passed through the supply and demand mechanism that adjusts the price of a share to the real value of the company. This increases market inefficiencies and allows those investors who follow an active strategy to buy shares at prices below their intrinsic value.

The problem this work seeks to address is therefore to determine whether it is possible to develop an active investment strategy that is capable of taking advantage of these inefficiencies and consistently outperforming market returns over the long term.

Within active management, there are different approaches and a multitude of different investment strategies. One widely used is to conduct a fundamental analysis of the company, with the aim of measuring its intrinsic value through the company's balance sheets and financial information. There are also methodologies based on technical analysis or with a quantitative approach, in which the future evolution of a stock is determined with the help of indicators based its price and by developing mathematical and statistical models.

Thanks to the growing development of artificial intelligence tools and improvements in the management of large amounts of data, the applications of this type of algorithms to the financial markets are being increasingly studied. The aim of these applications is to take advantage of the high capacity of these models to adapt to new conditions and data sets and their ability to detect patterns in order to try to determine the evolution of a financial asset.

In order to meet the main objective proposed in this work, it is first necessary to develop investment strategies and then to check their validity. It is necessary to demonstrate that the investment strategies consistently deliver satisfactory results.

To this end, the daily prices of the shares that make up 8 of the most important developed financial markets, the S&P 500, the DAX 30, the CAC 40, the Spanish Mercado Continuo, the FTSE MIB, the FTSE 100, the S&P TSX 60 and the ATX, have been compiled. The data dates back to 1 September 2010 and runs through to 30 August 2019. The construction of this database is carried out using the financial data provider SIX Financial Information.

A tool was then developed in MATLAB to allow the backtesting of the strategies developed on the basis of this database. This tool will allow the recovery and pre-processing of the data on each share, create the investment strategies, apply them to the selected data set, manage the portfolio with the shares obtained thanks to the strategy and finally analyse the results achieved.

One of the dangers that can arise when back-testing a strategy is that it may have worked by chance over the period of time selected and with the stocks that have been chosen. To obtain a more robust strategy, the time period used has been separated into two, a first period of approximately 7 years and another of 2 years. In addition to this, the test is carried out independently on each of the 8 selected indices. The idea is therefore to check whether the strategy has worked in both time periods and in all financial markets, regardless of the particular conditions of each one of them. In this way a more robust strategy is obtained, capable of working satisfactorily in the greatest possible number of different environments.

Five investment strategies have been selected, due to the good results they have provided in all the selected financial markets and in both periods of time.

All strategies are systematic, i.e. it is necessary to define certain rules based on parameters or an algorithm that automatically generates the signals to buy or sell shares. One of the advantages of this type of strategy is that on many occasions it is the investor himself who hinders his success in the financial markets. Either because he makes too many decisions, because he considers his skills in predicting the evolution of the financial markets to be above average and above what they really are, or because he is a victim of one of their biases. By establishing a systematic strategy, a large number of decisions are eliminated, particularly the most dangerous ones, those taken in times of crisis, when a stock has suffered a significant fall. In those moments it is normal for the investor to have doubts about which is the best decision to take. Thanks to a systematic strategy all those doubts disappear. Certain decisions still have to be made, as which rules must be followed, or on what criteria the algorithm is based, but this process can be carried out calmly, with time, without the pressure of having to act immediately.

The first strategy is based on a combination of two rules. The first is based on the RSI, Relative Strength Index, of the stock. This is an indicator that allows the magnitude of a stock's short-term trends to be measured. This rule is intended to determine which stocks are being overbought recently and which are being over-sold. The idea is that when a stock provides one of these signals, its trend is to revert to the average. So a stock that has been overbought in recent days and whose price has therefore risen, is usually sold in the following days causing the price to fall. The second rule is based on the annual return obtained by the stock. The shares are classified from lowest to highest return and the investment is made in those with the highest return in the last 12 months. Both rules try to take advantage of a factor called momentum. This phenomenon is based on the idea that the market moves by trends, which last in the short to medium term. This strategy has therefore been called Pure Momentum Strategy.

The second strategy is also based on the combination of two rules. The first one is once again the annual return obtained by the stock, generating a buy signal for those stocks that have obtained a higher return in that period of time. The second rule classifies the shares according to the volatility they have with respect to the market. This indicator allows to know how the stock has behaved in relation to movements in the index to which it belongs. The idea that has been put to the test is that historically those stocks whose beta is lower, meaning that their variations are usually smaller in comparison to those of the market, have obtained greater returns than those with a higher beta. Therefore the rule gives a buy signal for those stocks with lower beta. As this is a strategy that aims to take advantage of the momentum and low beta of the shares, it has been called, Momentum and Low Beta Strategy.

The third strategy uses an artificial intelligence model. It is based on the Machine Learning algorithm known as Naive Bayes' classifier. It is a probabilistic classifier that allows identifying to which sub-set of categories a sample belongs by means of an application of Bayes' theorem. In this way the objective of the algorithm is to classify each action determining whether it should be bought, sold or held. To do so, the model needs a series of indicators for each stock, temporary indicators, returns of the stock at different terms, the RSI and finally the MACD of the stock. The MACD is an indicator that allows the momentum of a stock to be determined by comparing the relationship between two moving averages of different terms and the price of the stock. This strategy has been called the Naive - Bayes Classifier Strategy.

The fourth strategy is based on a deep learning algorithm. It is composed of a Long Short - Term Memory neural network which provides the buy and sell signals for each share. It is a recurrent neural network, i.e. it has closed loops that go from outputs to inputs. In this way the information is allowed to persist during certain iterations, a part of the data from the previous outputs becomes the incoming data of the next iteration, in this way the information is remembered, making it seem that the neural network has a memory, hence its name. These loops also allow the neural network to be able to show temporal behaviour, which makes it a suitable tool for making decisions based on time series. This algorithm also needs a certain number of indicators on which to make forecasts. The model has been developed so that it uses the same indicators as the strategy based on the Naive-Bayes classifier. In addition to the signals provided by the model, this strategy performs additional filtering based on the beta rule of the second strategy. In this way, shares with a lower beta are also privileged. This strategy has been called LSTM and Low Beta Strategy.

The last strategy uses a different model of Deep Learning. It is based on a simpler neural network called Feedforward. This neural network has an open architecture, without feedback from the outputs to the inputs. Information moves from inputs to outputs in a unidirectional way and forward, which gives the neural network its name. It is the simplest of all neural networks. For the neural network to function satisfactorily it also requires a certain number of indicators, it has been decided to use the same ones as in the two previous strategies. Finally this strategy also adds the additional filter based on the beta of the actions. This strategy has been called FeedForward Neural Network and Low Beta Strategy.

Finally, a tool has been developed that allows portfolios to be optimised, regardless of the strategy used. The eventual use of this tool is one of the investment decisions that must be made and any of the strategies can work both with and without the use of optimization. This tool allows the capital allocation to each of the shares in the portfolio to be determined in an optimized way, so that the estimated Sharpe ratio is maximum. The Sharpe ratio is a widely used indicator to determine whether an investment strategy has performed well as it allows the return obtained to be compared with the risk taken.

Below is an example of the result obtained when backtesting the FTSE 100 shares in the second period from July 2017 to August 2019 using the strategy based on the Naive-Bayes classifier,

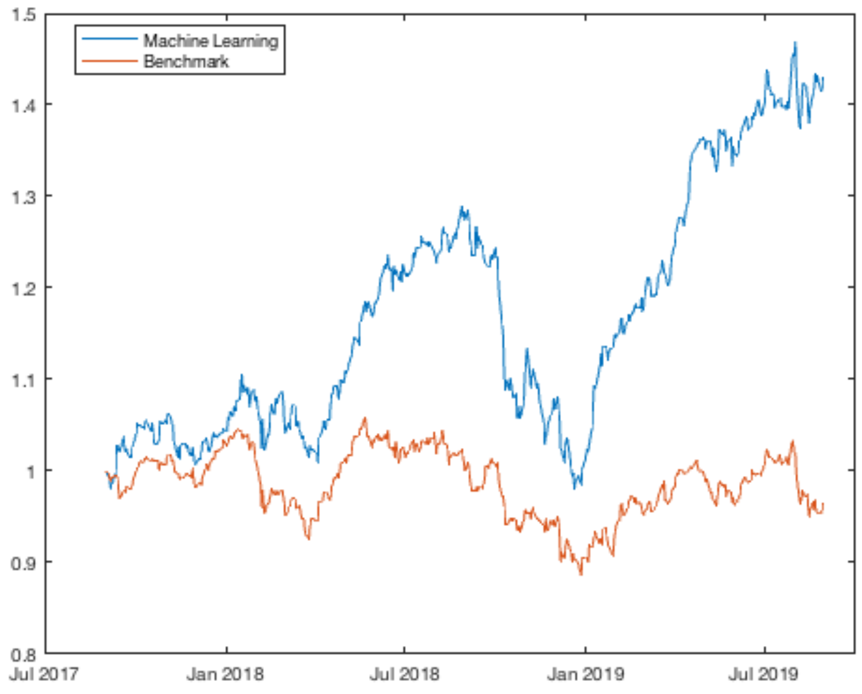


Figure 9: Equity Curve – Naive – Bayes Classifier – FTSE 100

	Sharpe Ratio	P&L Final	Max Drawdown Estrategia	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
FTSE 100	1,2303	14.300.900	23,98%	16.41%	7,99e-02%	10,90%	0,74%

Table 3: Performance indicators – Naive Bayes Classifier Strategy

Once the investment strategies that have been able to provide good results consistently i.e. outperforming a passive strategy in both time periods and in all financial markets have been selected, the strategies are implemented with the financial data of each stock in real time.

The tool for implementing the strategies and managing the portfolios in real time has also been implemented in MATLAB and following the same structure as the backtesting tool. The main difference between the two lies in the way the data is obtained. Because the real-time tool requires obtaining the latest data on a regular basis for the quotes of companies belonging to the investment universe, it has been decided to use Yahoo - Finance as the data provider. For this purpose the tool is designed to download the data directly from MATLAB.

The evolution of the portfolios obtained thanks to the implementation of each of the strategies is shown below as of 27 August 2020,

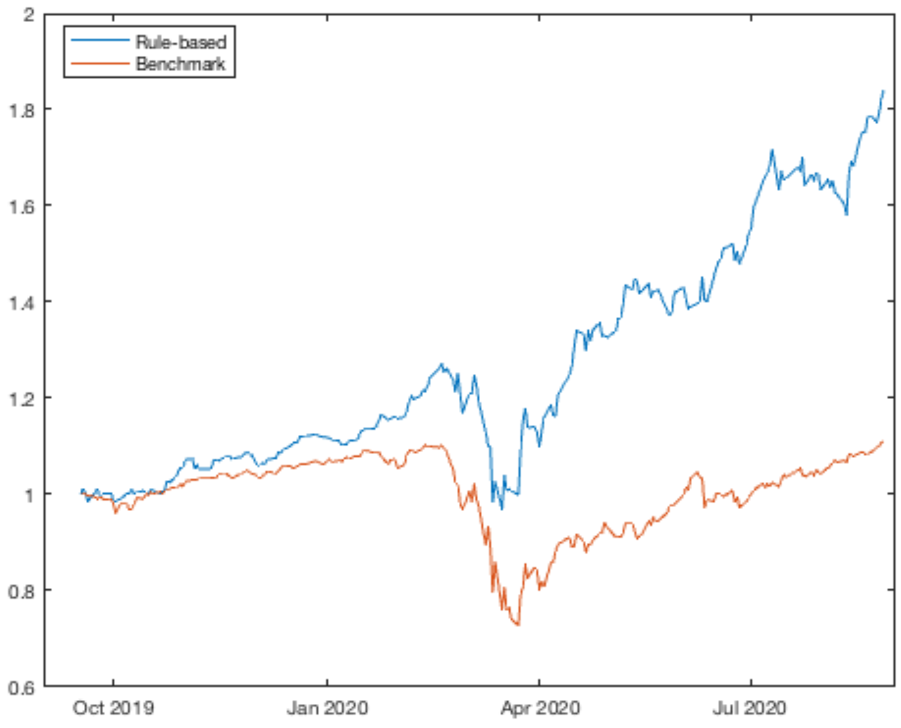


Figure 10: Equity curve Momentum Strategy – 27 August 2020

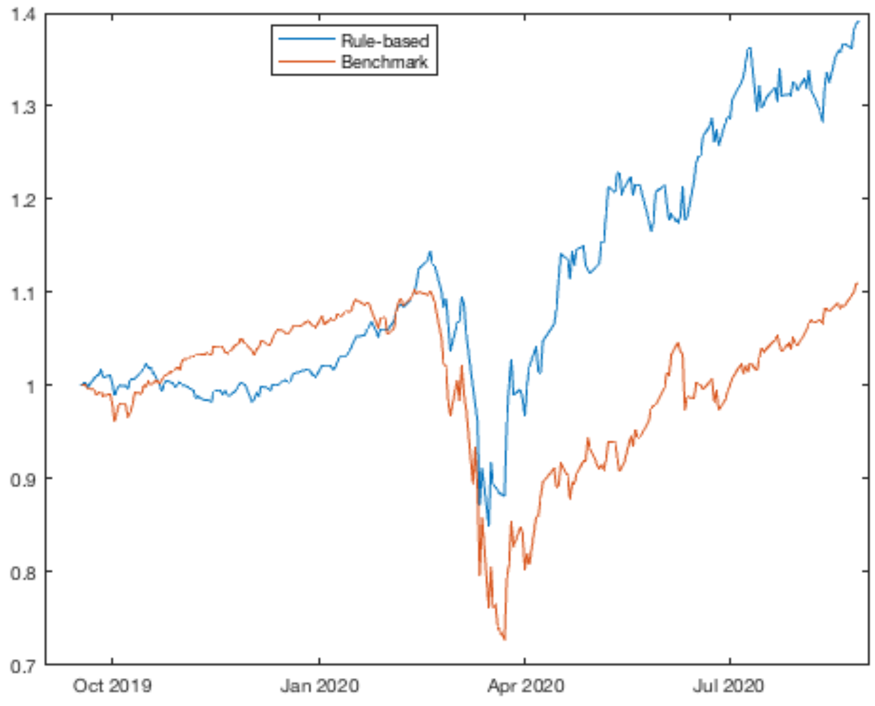


Figure 11: Equity curve Momentum – Low Beta Strategy – 27 August 2020

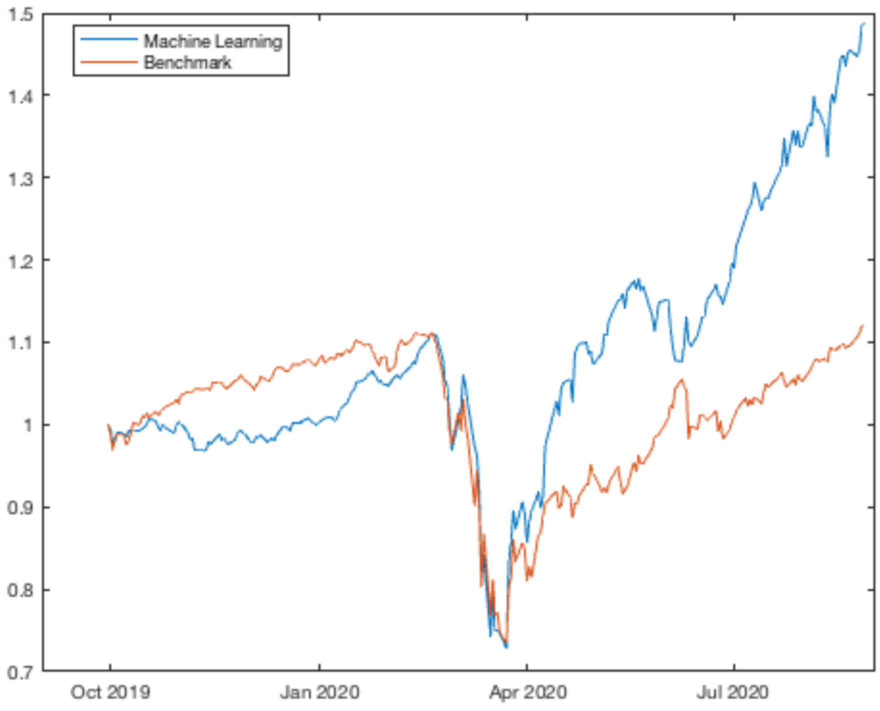


Figure 12: Equity curve Naïve Bayes Classifier Strategy – 27 August 2020

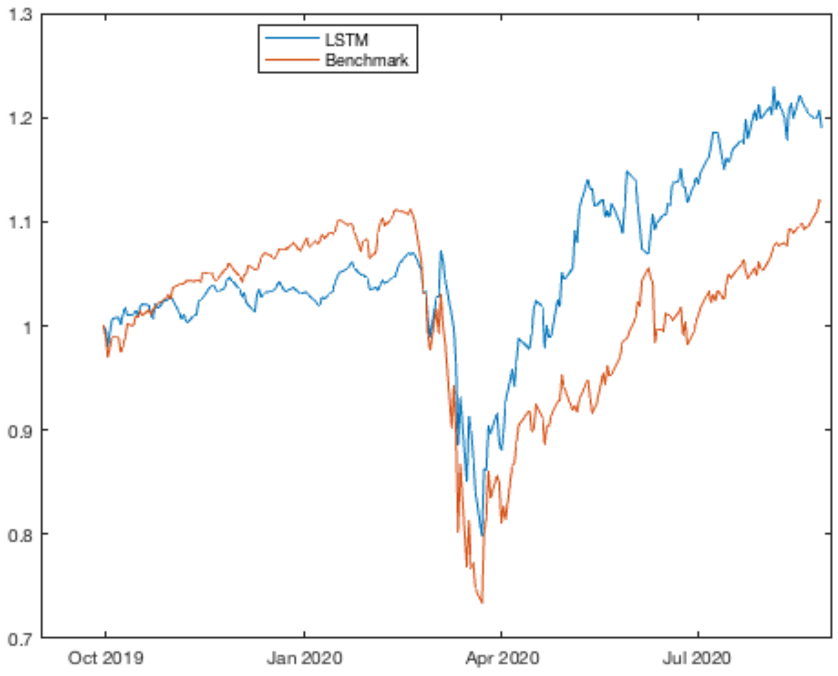


Figure 13: Equity curve – LSTM Low Beta Strategy – 27 August 2020

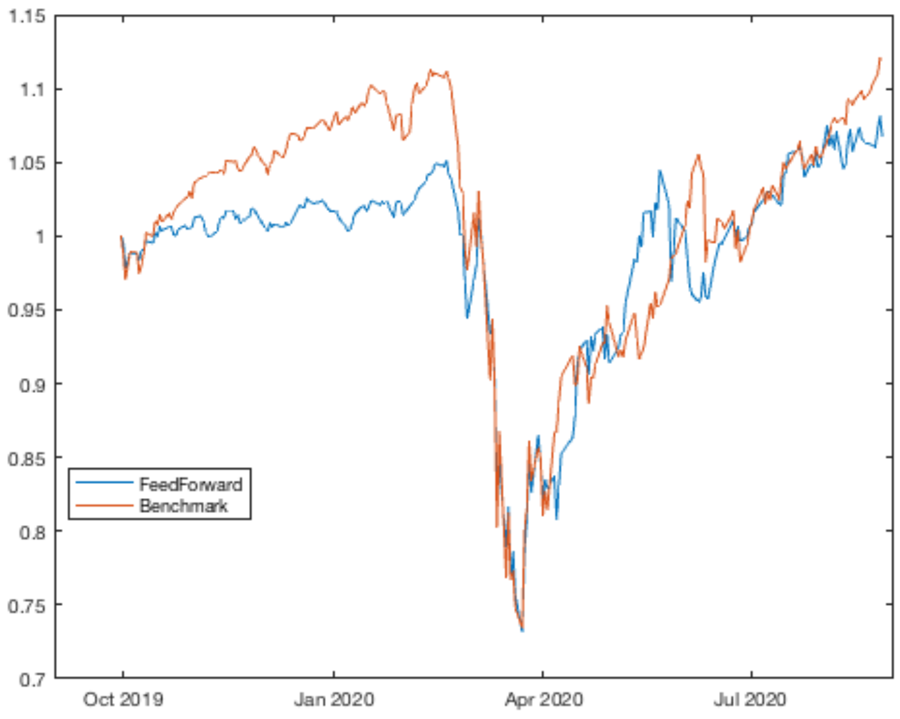


Figure 14: Equity curve - Feedforward Strategy Low beta – 27 August 2020

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
Pure Momentum	2,3018	18.409.400	23,70%	34,01%	0,24%	13,25%	1,64%
Momentum – Low Beta	1,4192	13.904.800	25,73%	34,01%	0,12%	6,18%	1,49%
Naive - Bayes	1,4097	11.400.800	35,41%	34,01%	0,15%	8,59%	1,55%
LSTM – Low Beta	0,8302	11.895.600	25,55%	34,01%	5,02e-02%	1,44%	1,52%
Feedforward – Low Beta	0,4080	10.665.860	30,38%	34,01%	3,02e-04%	-1,97%	1,46%

Table 4: Performance indicators obtained as of 27 August 2020

In view of the results obtained, it can be concluded that, with the exception of the strategy based on the Feedforward neural network, the investment strategies have obtained very good results up to 27 August 2020. It should be noted that although the results obtained with the latest investment strategy are not higher than those of its benchmark index, they are very similar and have been obtained with lower volatility.

Once the project has been completed, it can be concluded that the main objective that had been proposed has been fulfilled. It has been possible to develop investment strategies based on active management capable of obtaining better results than those obtained through passive investment in a global reference index. This objective has been achieved not only in terms of the returns obtained but also in terms of the risk assumed. The strategies developed are capable of obtaining a good return while maintaining a level of volatility similar to that of the market and sometimes even lower.

Table of Contents

Table of Figures

List of Tables

1. Introduction

1.1 History

Since 1970, financial markets have an increasing relevance in our daily life. They play a vital role in every capitalist economy, allowing to allocate resources where they are needed the most. They provide finance for business, governments and allow people's savings to be channelled. There are numerous financial markets, depending on the location, the type of financial instrument that can be bought or sold, or whether it is a new security or an existing one.

The stock market is one of those financial markets, it allows publicly traded companies to issue shares or common stock and raise money for finance their operations and grow. The stock market has two characteristics that make common stocks a very attractive type of asset for investors, its high degree of liquidity and that historically is the asset that has provided the highest return to investors over the long time. The liquidity enables the stockholder to easily and quickly sell his shares, this reduces risk because it allows the investor to get rid of the securities he no longer wants, particularly during a stressed situation. Despite its high volatility and the fact that it is a risky asset, which could lead to the loss of money in the short term, investing in equity is historically the best idea, in the sense that it is what has made the most money for investors. As you can see in the following chart, which compares the total return on an initial investment of 10000 dollars during the last two centuries, the investment in stock is undoubtedly the best choice.

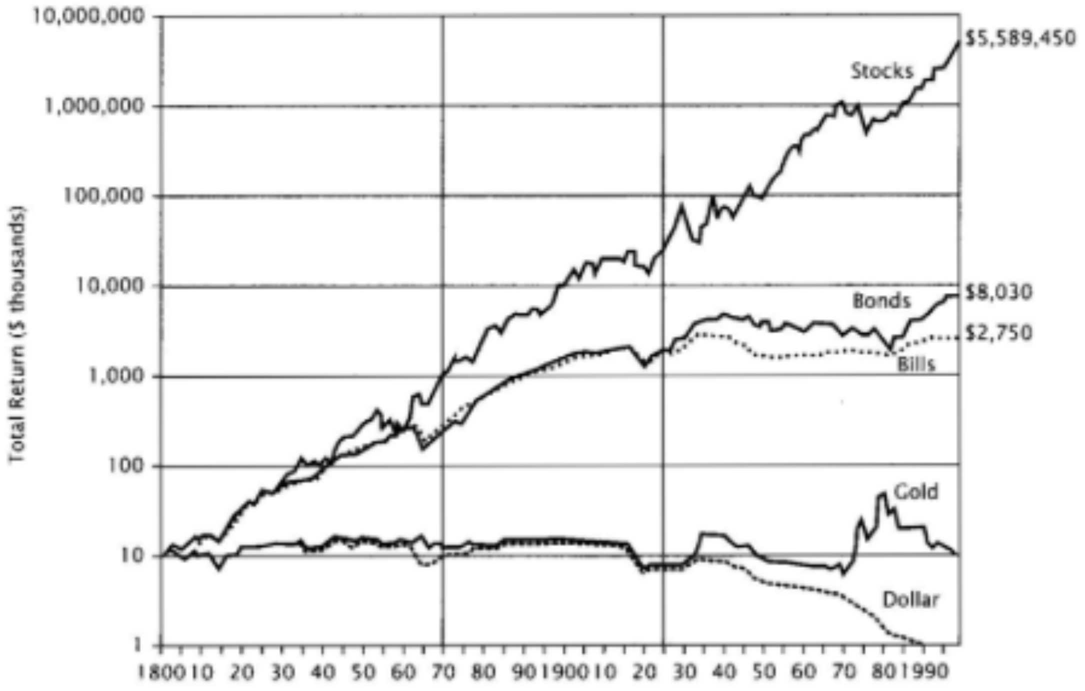


Figure 15: Total real return on $10000 initial investment from 1802 to 1997 [1]

Because of this greater upside potential, the focus in this work will be made on how to correctly invest in stocks. The different shares held by an investor are said to make up a portfolio, the control and different decision-making processes to configure a portfolio and define the exits and entries is called portfolio management.

One of the greatest discoveries in portfolio management occurred in 1952 thanks to Harry Markowitz’s theory of portfolio selection (Markowitz, 1952 [2]). Previously, the creation of the portfolio was done by analyzing the assets individually, considering only the possible return and choosing simply the one with the highest expected return. His work brought two important changes, first he emphasizes that risk is an inherent part of the return, the only way to higher return is through higher risk and second he analyzes the possible investments looking at the effects on the overall portfolio. In other words, the focus is on the marginal contribution it would have to the return and risk characteristics of the portfolio. Going back to what was done before, it is easy to deduce that if investors owned a portfolio composed by the stocks with simply the highest expected return, they also held the portfolio with the most risky stocks. Thanks to Markowitz's work, investors began to monitor and control the actual return and risk characteristics to which they were exposed.

This work was the beginning of what we call today the Modern Portfolio Theory. This theory is a mathematical approach for the construction and management of a portfolio. It assumes that investors are risk averse, the expected return must then be maximized for a given level of risk or the risk incurred minimized for a given level of expected return. Modern Portfolio Theory also integrate the idea that through diversification is possible to lower the risk incurred while maintaining the same amount of expected return. The main idea is that risk and return should not be assessed for each new investment individually but rather looking at the contribution to the overall portfolio. It is then possible to create an “efficient frontier” made with optimal portfolios, which offer the maximum possible expected return for a given level of risk. The efficient frontier is a graphic representation of such optimal portfolios. On the X-axis the standard deviation is represented as a proxy for risk and on the Y-axis the expected return of the investment is usually considered.

In portfolio management, it is therefore crucial to understand the concept of expected return, how it is calculated and what are the reasons that make the expected returns different depending on the assets. In order to explain the stock return Jack Treynor (1961 [3], 1962 [4]), William F. Sharpe (1964 [5]), John Lintner (1965a,b [6]) and Jan Mossin (1966 [7]) developed independently the Capital Asset Pricing Model, known as CAPM, based on the Modern Portfolio Theory. This model describes the sensitivity of the asset’s return to the systematic risk, or market risk, which is the risk due to the exposure to the financial market. This risk is often called beta (β) in the financial industry and every stock has one. The expected return of an asset is therefore due to its exposure to the market and to his sensitivity to the market’s movements, according to the following formula,

$$ER_i = Rf + \beta_i \times (ER_m - Rf)$$

Where,

ER_i is the expected return of the investment i.
Rf is the Risk free rate.
$\beta_{i.}$ is the beta of the investment i.
ER_m is the expected return of the market.
$(ER_m - Rf)$ is usually considered as the market risk premium, which is the premium obtained for being invested in the market.

The simplicity of this model is what made it popular and useful, however some assumptions made have been proven to be wrong. Firstly, it is assumed that the market is efficient, i.e. any relevant information about a share is immediately absorbed and reflected in its price. There is still an ongoing debate about whether or not markets are efficient, as well as to what extent. It is generally assumed that financial markets are considerably efficient in the long term, but that there are nevertheless some inefficiencies in the short term and for some securities. It is these inefficiencies that allow investors to make money. Without them, the best investment decision would simply be to own a passive asset that replicates the market. The second assumption made is about the rationality of investors. As shown above, it is generally assumed that investors are rational and seek to maximize the return obtained for a given level of risk or alternatively minimize the risk assumed for a given level of return. However, in recent years a new field has gained importance called behavioral finance in which the different biases of investors are analyzed. These biases, mixed with the investor's emotions, could cause their behavior to not always be rational, leading to the appearance of anomalies in the market, such as bubbles in the prices of certain assets.

To solve this, in 1992 researchers Eugene Fama and Kenneth French expanded the CAPM with the three factor model (Fama, French, 1992 [8]), which add two additional factors, the size and the value of the stock in addition to the exposure to the market. The size factor represents the difference in return obtained between a portfolio composed by small firms and one composed by big firms. Analogously, the value factor, is represented in this work by the book-to-market value of the firm, and it is equal to the difference in return between a portfolio of high and low book-to-market firms. The book-to-market is a financial ratio, and it is considered as an indicator of the value of the firm. It is computed as the book value over the market capitalization of the company. The book value is simply the value of the assets divided by the value of the liabilities. And the market capitalization is obtained by multiplying the number of shares outstanding by the price of one share.

This model allows to disaggregate the expected return into more factors, in order to understand with greater precision the drivers behind it. By combining the additional factors, the following relationship is obtained:

$$ER_i = Rf + \beta_1 \times (ER_m - Rf) + \beta_2 \times SMB + \beta_3 \times HML + \alpha_i + e_i$$

Where,

ER_i is the expected return of the investment i.
Rf is the Risk free rate.
$\beta_{1.}$ is the beta of the investment i, i.e. the sensitivity to the market.
β_2 is the sensitivity to the size factor
β_3 is the sensitivity to the value factor

ER_m is the expected return of the market.
$(ER_m - Rf)$ is the market risk premium.
SMB is the firm size factor, Small minus Big.
HML is the value factor, High minus Low.
α_i is the intercept term of the investment i, estimated with historical return data, assuming the factors capture all systematic risk, if the market is in equilibrium, it should be 0.
$e_{i.}$ is the abnormal return of the investment i, represents the portion of the return unexplained by the factors.

This improvement achieve to explain up to 95% on average of an expected return against 70% explained by the CAPM. Due to this significant improvement, researchers have continued in the same direction and have kept adding factors creating a new investment approach called Factor Investing. One of the most important factor is momentum, the idea was developed shortly after the three factor model (Titman and Jegadeesh, 1993 [9]) and explains that the market has the tendency of continuing with the last trend for a while. The strategy therefore is to buy stocks that have had positive returns over the past few months, and sell those with negative returns.

There is a wide variety of tools that allow to identify and quantify the level of these factors. One of the most widely used momentum indicator is the Relative Strength Index, it measures the recent changes in price and provides an indication of whether the stock is overbought or oversold. There are plenty of indicators like the RSI that provide trade signals for the portfolio manager, allowing him to know when to buy, sell or maintain an asset.

1.2 Managing uncertainty

The forecasting of financial markets is an extremely complex problem, since many variables, factors and noise affect the movement of prices. It is impossible to determine with certainty what is behind a short-term movement in the price of a given asset. Despite this volatility, the movements are not completely random, the financial market can be represented as a non-linear, dynamic and chaotic system. This is because a mixture of two important factors come into play, on the one hand the gradual change in the balance of power between buyers and sellers. On the other hand, unexpected events and variations that arise in daily negotiations.

That is why the use of automatic strategies based on rules, and the use of investment strategies based on artificial intelligence tools can help to better deal with all this uncertainty. In the first case, as they are automatic strategies, with clear rules, the effect that the investor's biases or emotions can have is eliminated or limited. In the second case, tools based on artificial intelligence have the ability to learn from the past and identify patterns that go unnoticed by a human investor. In this way, price forecast models can be produced that include all types of indicators and with a high capacity to adapt to different market conditions.

Strategies with this approach will therefore be developed throughout this work, always taking into account the importance of simplicity and robustness of a strategy, to avoid falling into the trap of developing a strategy that is too adapted to market conditions in the past, and that may not be so in the market of the future. This phenomenon is one of the main dangers of excessive data mining.

1.3 Motivation

Investment management, finding the most efficient allocation of capital, is one of the most complex and stimulating intellectual challenges you can find. This is due to many reasons such as the great variety of financial instruments and companies in which it is possible to invest. The competitiveness is an important element, it is difficult to beat the market because it requires to obtain better results than the other investors. Otherwise the strategy obtain average returns, or even worse than the market and in that case it is more profitable to invest passively, thus obtaining the return of the market, with very low fees and management costs. It is therefore absolutely crucial for every active portfolio manager to have a strategy that works in the long term.

This is the most important debate in asset management in recent years, is it possible to consistently outperform market returns in the long term through active management? This is one of the questions this paper will try to answer. As explained above, in recent years there has been a flow of funds from active management to passive management, it seems that investors have lost confidence in active management and have made their decision. However, this phenomenon may allow the remaining active investors to obtain higher returns, due to less competition to outperform the market, as a large number of participants simply buy the largest companies without thinking about whether or not it is the best and most efficient way to allocate their assets. As more and more people buy the same assets, in this case the shares with a higher capitalization, the price of the remaining assets will not have gone through the mechanism of supply and demand that makes the price move closer to the value of the company. This leads to more and more inefficiencies, i.e. more opportunities to buy a share whose price is below its intrinsic value.

That is why this work, that ultimately seeks to create an investment strategy based on technical indicators, is so interesting. The challenge is to solve one of the greatest financial challenges, to develop an active management strategy that works and is consistent over the long term. The idea is totally updated, in addition to that, because when a researcher or portfolio manager finds an investment strategy that works, generally other investors put it into practice and cause the inefficiencies that have allowed the strategy to work to become smaller and smaller in size and frequency. Due to this phenomenon, it is necessary to constantly investigate and search for new investment strategies that allow the investor to create a portfolio that consistently outperform the market.

Personally I find it a very interesting and stimulating master's thesis. Since it touches on many issues that I find fascinating. Depending on the strategy that is being followed, it uses mathematical and economic concepts, one of the aspects that I like most about investment is that generally a very deep knowledge is developed on a large number of topics and sectors, it is necessary to be aware of what the innovations are in each sector, who are the most important actors, in which direction they are going.

I also like the idea of managing uncertainty as whatever the strategy is, it will be one of the most important aspects. As Ian Wilson, a former GE executive said,

" No amount of sophistication is going to allay the fact that all of your knowledge is about the past and all your decisions are about the future."

For these reasons I would like to start my professional career in investment and portfolio management and this master's thesis has been very helpful in allowing me to understand and put into practice many important concepts related to both topics.

1.4 Objectives

The main objective of this work is to develop an original, basic and visual portfolio management system that will allow the investor to maximize his return and finally to consistently outperform the market.

In addition, a series of secondary objectives will be implemented which will enable the main objective to be achieved. This work will also study the framework of portfolio management. We will begin by searching and understanding the general concepts, which are the most important trends, methodologies and investment strategies to be able to gain a global overview. That will serve as a starting point for the development of the work. Once there is a solid foundation, we will focus on the strategies based on technical indicators as well as on the different strategies that incorporate in some way such indicators. The focus will be placed on those strategies that have been most successful historically.

The current validity of these existing strategies will be checked, as well as the validity of the strategies that are developed along this work, using backtesting. Therefore, one of the outcomes of the paper will be a battery of investment strategies that provide satisfactory returns and with a proven track record.

Finally, the purpose is to be able to develop a system that effectively manages several investment portfolios, each following a previously tested investment strategy.

1.5 Work Methodology

In order to achieve the main objective a series of intermediate levels will be performed. The paper can be divided into two main parts, on the one hand there will be an analysis and study of existing investment strategies as well as trying to develop new investment strategies. The strategies will be tested to see if they work in the long term. On the other hand, the second part consists in the development of a tool for the creation and management of portfolios. The procedure to be implemented in each of the parties is detailed below.

The first step will be the analysis and study of technical indicators and existing strategies. In order to do so, the different articles that exist on the matter will be searched and analyzed. From this study will derive the development of new investment strategies, either combining some existing ones, optimizing them or developing a new and original approach.

Then it will be necessary to make a small study of all the information that can be obtained from the stocks that are included in this work. For this purpose, a search will be carried out for the data offered by the different providers of financial information, which are their strong and weak points, and we will explain which of them has been chosen and why. The data to be downloaded for each stock will be crisp and inter-valued data such as opening and closing prices, maximum and minimum values, etc. The information will be needed for each day of the last 10 years. By doing this, we will obtain what is called an interval time series. As shown in Maté, C. (2012 [17]) El análisis de intervalos. Aplicaciones en ingeniería, the use of this type of data will reduce the risk and uncertainty when making decisions. Another benefit is that this will provide a feasible and less complex solution than classical distribution-based approaches.

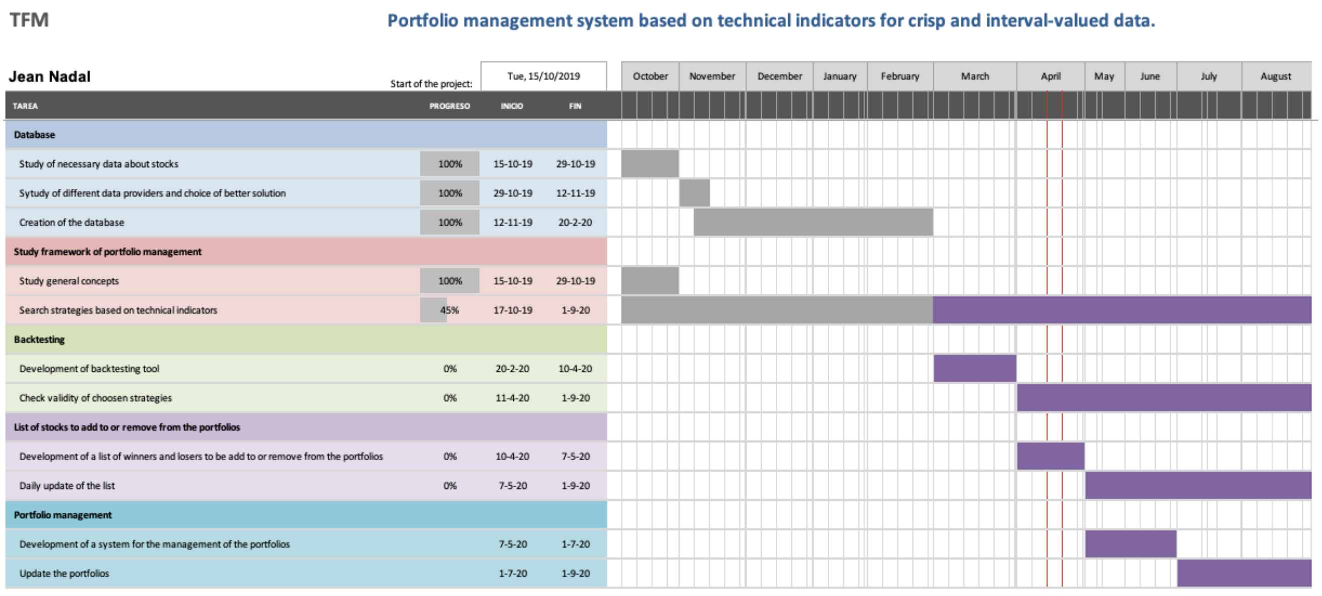
Once there is an investment strategy, either existing or developed within the scope of this work, and the necessary data is available, it will be necessary to test it. This test will be carried out through backtesting. This can be done with Excel, with some other program such as MATLAB or even doing some program in C or R. In this work it has been decided to use MATLAB. To perform backtesting it will be necessary to download the necessary data and indicators of each share on a daily basis and for the last 10 years. The list of shares will be formed by stocks constituting the indexes from the largest and most important occidental stock exchanges, namely the Spanish equities market, the S&P 500 and the Nasdaq 100 from the United States, the FTSE 100 from London, the CAC 40 from Paris, the DAX from Frankfurt, the S&P/TSX from Toronto, the ATX from Vienna, The AEX Index from Amsterdam and the FTSE MIB from Milan.

At the end of this part, a set of investment strategies that have been tested and are ready to be implemented in a real investment portfolio will be available.

Once an investment strategy is ready to be implemented in a real portfolio, we proceed to the second part of this work, the creation and management of the portfolios.

A program, also developed in the MATLAB environment, will be created for this purpose. The objective is obtain a file where the different portfolios will be seen in their current state. It will display which shares make up the portfolio, at what price they were bought and the date, their current price and other information relating to the shares. In that file or in an underlying one, a historic of the shares that have been in the portfolio, their entry and exit price and their characteristics will be added. In this way, profitability can be calculated as well as a whole series of subsequent analyses such as risk attribution, performance attribution, etc... These analyses make it possible to understand how the portfolio has behaved and why, in order to make eventual adjustments in the investment strategy and gain insights for the development of new ones.

Below you will find a Gantt chart showing the expected time frame of the different tasks that will be carried out within this project.


TFM

Portfolio management system based on technical indicators for crisp and interval-valued data.

Jean Nadal

Start of the project: Tue, 15/10/2019

TAREA	PROGRESO	INICIO	FIN
Database			
Study of necessary data about stocks	100%	15-10-19	29-10-19
Sytudy of different data providers and choice of better solution	100%	29-10-19	12-11-19
Creation of the database	100%	12-11-19	20-2-20
Study framework of portfolio management			
Study general concepts	100%	15-10-19	29-10-19
Search strategies based on technical indicators	45%	17-10-19	1-9-20
Backtesting			
Development of backtesting tool	0%	20-2-20	10-4-20
Check validity of choosen strategies	0%	11-4-20	1-9-20
List of stocks to add to or remove from the portfolios			
Development of a list of winners and losers to be add to or remove from the portfolios	0%	10-4-20	7-5-20
Daily update of the list	0%	7-5-20	1-9-20
Portfolio management			
Development of a system for the management of the portfolios		7-5-20	1-7-20
Update the portfolios		1-7-20	1-9-20

Figure 16: Gantt Chart of the project.

1.6 Resources

As has been seen in the previous section, a fundamental aspect of the work is to have financial information as opening price, closing price, maximum price of the day, minimum price of the day, volume etc. on a daily basis for each of the approximately 1500 actions included in the work. To get this information it is necessary to have access to a financial information provider such as SIX Financial Information, Bloomberg or Yahoo Finance. In this work we will SIX Financial Information and Yahoo Finance.

Another fundamental element is the database with information about the approximately 1000 companies included in the investment universe. Those databases will be made through Excel files.

Finally, as indicated above, both the backtesting tool and the portfolio management and analysis tool have been developed in MATLAB.

2. State of the art

Given the well-known benefits of investing in equities, and as you may have noticed, a great deal of research has been done in this area. The main objective is to find the most efficient way to allocate capital. There are hundreds of different strategies and thousands of technical indicators and trade signals that allow investors to define their investment philosophy.

One strategy is to perform a fundamental analysis, measuring the intrinsic value of the security using financial information of the company. More possibilities are through technical analysis or based in a quantitative approach, using mathematical and statistical models in order to predict the future trend of the security. All these strategies are included in what is called active investment, where someone is in charge of making the decisions about how to allocate the money. However, there is also passive investment, where the investor simply replicates an index, so he does not make decisions nor has to try to predict the next trends in prices. The principal benefits of this strategy is the assurance of obtaining the same returns as the benchmark, which is not the case with an active approach where one of the risks is to obtain below-average returns, and the capacity of doing it with ultra-low fees, which is really beneficial for the client or the investor.

Recently a new strategy has been developed, which combines the positive aspects of active and passive investment, it's called Factor Investing. It is based in the investment in passive instruments that captures the profitability of one or several factor. This factors are usually considered to be value, momentum, size, volatility, quality and dividends. The investor only needs to choose one or several of this factors, and invest in an ETF that is able to pick up the excess return that the investor believe this factor will obtain in comparison to the global benchmark. It is an active strategy because the investor choose one factor, so he can beat the benchmark and obtain higher returns, but is somehow also a passive strategy because he invest through a passive ETF that only follows the index of the factor. For example, a value ETF is replicating a value index as for example the MSCI World Value Index.

Over the past years, there has been an increase in the flow of funds from active to passive strategies. However, active management is still the most used way of investing. In this strategy we find any conceivable combination of technical indicators with all kinds of trade signals. For example, a widely used indicator is the RSI. The Relative Strength Index is a momentum indicator, therefore it allows to measure the magnitude of the recent trends in price. It was developed by J. Welles Wilder in his 1978 work, New Concepts in Technical Trading Systems (Wilder, 1978 [10]). It is presented as an oscillator, a line that moves between two extremes, 0 and 100, and it represents the ratio of higher closes to lower closes. Therefore it shows the current strength or weakness of the security, if it is above 50 it means that recently, the gains are greater than the losses and vice versa. The classic approach, developed by Anson et al. (2012 [11]), is to use the 14-day timeframe with two trading signals, at 70 the security it is told to be overbought and at 30 or below, oversold. In both situations a reversal is expected, therefore if the security is oversold the trading signal is to buy it and likewise, if it is overbought the trade signal is to sold the security.

A more recent work, Optimization and Testing of RSI by Patrice Marek and Blanka Šedivá (2017 [12]) try to answer whether the RSI can be still a good strategy. For this they compare the returns using a strategy based on RSI with others strategies. Regardless of the outcome of this research, the mere fact that it is carried out demonstrates that there is still room to more research in this field. For that reason, one of the investment strategies that will be used in this work will be based on the RSI – 14, eventually combined with other technical indicators.

Due to the extended use of technical indicators and thanks to the increasing development of computational tools and artificial intelligence, an increasing number of studies are being carried out that try to combine the two concepts. An example of this is the incorporation of the so-called genetic algorithms to the application of technical indicators in the search for signals to buy and sell financial assets.

In Adopting genetic algorithms for technical analysis and portfolio management by Tak-chung Fu, Chi-pang Chung and Fu-lai Chung (2013 [13]), they use a Genetic Algorithm to solve an optimization problem, which decides the best investment allocation considering the risk and return of the portfolio by optimizing the Sharpe ratio of the portfolio. The algorithm provides also the best combination of several technical indicators which are selected for decision-making. Finally, the Genetic Algorithm improves the RSI-14, by providing the values at which a buy or sell signal is given, i.e. the 70 and 30 values of the classical approach.

We found another approach in the work of António Gorgulho, Rui Neves, Nuno Horta, Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition (2011 [14]). In this case, the algorithm determines a Classifier equation which rank each stock within the market and pick the best stocks for defining the portfolio.

Other artificial intelligence tools have been applied to the investment process such as the increasingly used neural networks. In CAST: Using neural networks to improve trading systems based on technical analysis by means of the RSI financial indicator by Alejandro Rodríguez-González, Ángel García-Crespo, Ricardo Colomo-Palacio, Fernando Guldrís Iglesias and Juan Miguel Gómez-Berbís (2011 [15]), a double objective is pursued. On the one hand to handle uncertainty and noise, by learning from the past to forecast future prices. On the other hand the improvement of the RSI, to do so, the aim is to find the best number of days in the computation of the RSI for each stock and for that particular moment. Instead of using 14 days at all times and for any stock, the neural network look for the most appropriate one.

A different application of neural networks is studied in the work Neural Network Models for Stock Selection Based on Fundamental Analysis by Yuxuan Huang, Luiz Fernando Capretz and Danny Ho (2019 [16]). The objective is to develop an algorithm that resemble the decision-making process of a human expert, and will therefore predict which stocks within a market are going to be the winners and losers based on their return and finally build and manage a portfolio.

There seems to be an extremely high number of investment strategies and approaches, the question is how many of them are capable of consistently outperforming the market over the long term.

3. Data Collection

The first task undertaken to complete the objective is the creation of a database with the different prices related to each of the shares that make up the investment universe. This database will be used to feed the backtesting tool.

As previously mentioned, the backtesting investment universe is made up of the stocks that constitute the S&P 500 of the United States, the S&P TSX 60 of Canada, the ATX 20 of Austria, the CAC 40 of France, the DAX 30 of Germany, the Mercado Continuo of Spain, the FTSE 100 of the United Kingdom and the FTSE MIB of Italy.

The idea behind choosing such a high number of shares and especially such a high number of different markets, is that thereby the strategy is expected to be as robust as possible. One of the most important dangers when backtesting a strategy, is that it may only have worked by chance during the period of time covered and with the particular stocks that have been chosen. One of the most common and particularly dangerous biases for investors is the commonly known as hindsight bias. What happens in this case is that past events always seem less random than they really were, and the reality is that randomness is much more present than is thought. When working with a system as complex as the stock market, the dependence on randomness is much more important than in other types of applications, therefore it is essential to be extremely careful with the conclusions drawn from backtesting. This is one of the ideas developed by Nassim Nicholas Taleb in his book Fooled By Randomness [18] , in which he explains how a strategy can work simply by chance, leading people to incur too much risk and above all to believe that the success of the strategy is due to them without taking into account the randomness of the markets. The fate of these types of strategies is always that sooner or later they collapse, leading to the loss of everything that has been gained in the past. The great investors usually mention how relatively easy it is to make money in the markets simply by increasing the risk incurred, the problem is that, once again, by doing so there is a risk of losing everything, which can cause not only the overall return to drop considerably, but can even mean the possibility of no longer being able to invest, either professionally because the fund has ceased to exist or personally because all the savings have been lost. Nassim Nicholas Taleb uses an analogy with a Russian roulette wheel that allows this phenomenon to be understood in a very simple way. In the well-known game of Russian roulette, assuming a gun with 6 bullet chambers, there are 6 alternative stories, in five of them the player gets rich, in one of them he dies. If one imagines a young player, who has just finished his studies at the age of 25, and plays regularly, it is very unlikely that he will reach the age of 50. However, if a large enough number of participants is assumed, it should be expected that there will be a small number of extremely rich and successful people, (as well as a large cemetery). This phenomenon is very similar to the universe of hedge funds, which have an average life of approximately 5 years. The vast majority of these funds are based on very particular strategies, usually with significant leverage, which means that the strategies usually have a very high risk and therefore a particularly high dependence on randomness. Similarly to the Russian roulette, there are a small number of extremely rich and successful hedge fund managers and a large number of hedge funds that disappear every year. Of course there are some of these funds that have robust strategies and whose success is not only based on luck, there is a well-known joke that 15 years ago there were about 200 hedge funds, of which about 20 were consistently achieving good returns and today there are about 10,000 hedge funds, and the same 20 are consistently achieving good returns.

The lesson of this metaphor is that despite the complexity of financial markets, and the fact that markets generally rise for several years until there is a rapid and significant decline, they should not make people forget the existence of the bullet.

It is therefore important that the strategies developed in this work are simple and robust, so that they can withstand as many different scenarios as possible. By testing the strategies, separately and independently, in 8 different markets and not only in a few stocks of the S&P 500 or the national market of the person doing the research, which is what happens in most cases, it is expected to obtain only those models that really have some value to select the stocks that in the long run can outperform the market.

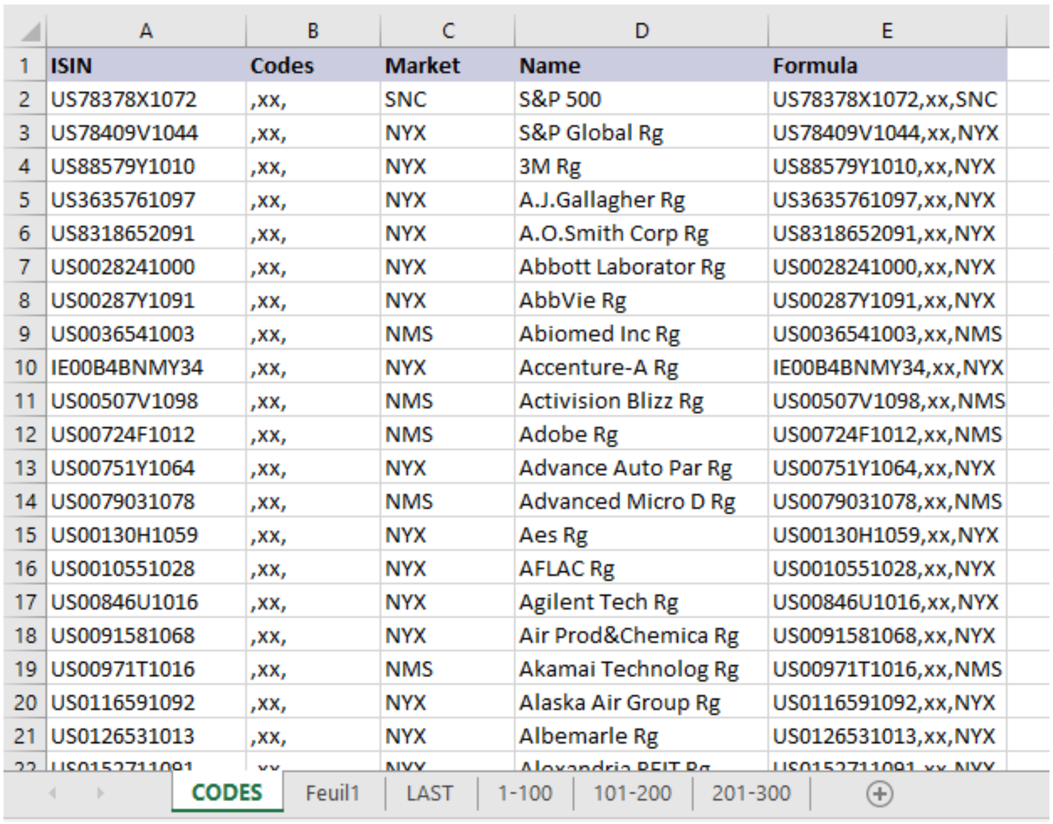
To obtain the necessary data, the financial data provider SIX Financial Information is used. It is one of the most important real-time financial data providers and provides all kinds of services and solutions to a large number of clients and companies in many different industries, particularly in the investment and asset management sector.

One of its most important products is SIX iD, a terminal, like the one shown in figure XX, in which it is possible to search for practically any financial data, be it different types of prices, financial ratios, information from company balance sheets, news, etc, from any company listed on any financial market and any type of asset, this is equivalent to more than 20 million financial instruments. They obtain the information directly from the different stock exchanges around the world so they are able to provide it back to their clients in real time.

One of the advantages of this platform is that it has a complement for Excel, called SIX iD FinXL. This is an add-in that allows to download any type of data available on the SIX iD platform, even data and prices in real time, and open it directly in Excel. This feature is especially useful for this work as it facilitates the creation of the database. One of the most important challenges faced by any person or company that wants to test the performance of investment strategies or do any kind of research about strategies, characteristics or history of financial markets is the creation of its database. Data scientist are increasingly in demand, since the creation, maintenance and manipulation of huge amounts of data is very complex. As for financial time series databases, it is generally assumed that the larger the database, i.e. the more financial instruments and the further back in time it arrives, the better. Obviously this has a cost since the larger the database is, the greater the amount of effort that has to be devoted to cure it, in such a way that there is no data that is wrong, that there is no lack of data etc.

In this work it has been decided that the database will be formed by the above-mentioned stocks, starting on September 1, 2010 until August 30, 2019. Therefore, we obtain a database large enough to be able to draw some interesting conclusions, while maintaining a relatively manageable size.

Thanks to FinXL it is therefore possible to download in a simple way the prices required for each share in an Excel spreadsheet. A spreadsheet is created for each of the indexes, which is called with the index name followed by 'Histo', as these are the historical prices, for example in the case of the S&P 500 it is called SP 500 Histo. The spreadsheets follow the same structure for each of the indexes. Three tabs are added to the spreadsheet. The first one, called CODES, contains the columns shown in Figure 17.



	A	B	C	D	E
1	**ISIN**	**Codes**	**Market**	**Name**	**Formula**
2	US78378X1072	,xx,	SNC	S&P 500	US78378X1072,xx,SNC
3	US78409V1044	,xx,	NYX	S&P Global Rg	US78409V1044,xx,NYX
4	US88579Y1010	,xx,	NYX	3M Rg	US88579Y1010,xx,NYX
5	US3635761097	,xx,	NYX	A.J.Gallagher Rg	US3635761097,xx,NYX
6	US8318652091	,xx,	NYX	A.O.Smith Corp Rg	US8318652091,xx,NYX
7	US0028241000	,xx,	NYX	Abbott Laborator Rg	US0028241000,xx,NYX
8	US00287Y1091	,xx,	NYX	AbbVie Rg	US00287Y1091,xx,NYX
9	US0036541003	,xx,	NMS	Abiomed Inc Rg	US0036541003,xx,NMS
10	IE00B4BNMY34	,xx,	NYX	Accenture-A Rg	IE00B4BNMY34,xx,NYX
11	US00507V1098	,xx,	NMS	Activision Blizz Rg	US00507V1098,xx,NMS
12	US00724F1012	,xx,	NMS	Adobe Rg	US00724F1012,xx,NMS
13	US00751Y1064	,xx,	NYX	Advance Auto Par Rg	US00751Y1064,xx,NYX
14	US0079031078	,xx,	NMS	Advanced Micro D Rg	US0079031078,xx,NMS
15	US00130H1059	,xx,	NYX	Aes Rg	US00130H1059,xx,NYX
16	US0010551028	,xx,	NYX	AFLAC Rg	US0010551028,xx,NYX
17	US00846U1016	,xx,	NYX	Agilent Tech Rg	US00846U1016,xx,NYX
18	US0091581068	,xx,	NYX	Air Prod&Chemica Rg	US0091581068,xx,NYX
19	US00971T1016	,xx,	NMS	Akamai Technolog Rg	US00971T1016,xx,NMS
20	US0116591092	,xx,	NYX	Alaska Air Group Rg	US0116591092,xx,NYX
21	US0126531013	,xx,	NYX	Albemarle Rg	US0126531013,xx,NYX
22	US0152711091	,xx,	NYX	Alexandria REIT Rg	US0152711091,xx,NYX

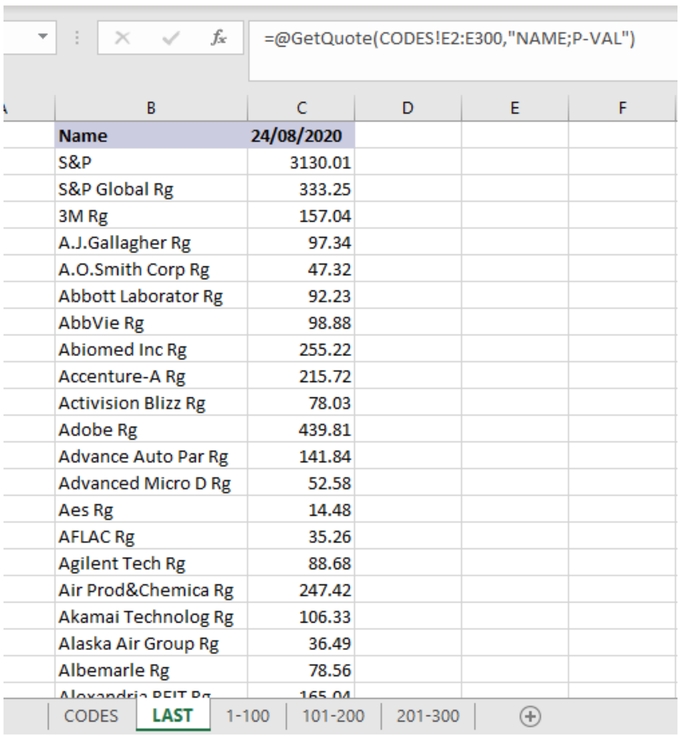
Figure 17: S&P 500 Histo - CODES

The first column contains the ISIN code for each of the stocks that compose the index. The ISIN code, from International Securities Identification Number, is a code that allows any financial instrument to be uniquely identified. Its structure follows the ISO 6166 standard and it is a 12-character alphanumeric code. The first two characters are alphabetical and are used to identify the country issuing the financial instrument. The next 9 characters, numeric, identify the instrument and a last character, also numeric, allows to detect if there has been an error when writing the code. This character is called check digit. The next column, Codes, contains the code ,xx,. This is the SIX iD code that enables the program to know that the user wants to search for an instrument by providing the ISIN. The third column is the market in which the share is quoted. The fourth column is the name of the instrument in question. Finally, the formula that will be used to create a request for information to SIX iD is defined. This formula is simply the sum of the cells containing the ISIN, the codes, and the market. For example in the case of Activision Blizzard, the formula would be, US00507V1098,xx,NMS as seen in Figure 17.

In the next tab, called LAST, the closing price of the previous day is downloaded for each of the shares that make up the index. This is done to check that the download and update of historical data has worked. To do this, a request is made to SIX iD by entering the following formula in cell B2,

=@GetQuote(CODES!E2:E33, "NAME; P-VAL")

GetQuote is one of the most important functions of FinXL and takes care of displaying the real time data and other features of the financial instruments. The first thing that is passed as an argument are the securities that the user wants to search, this can be done in several ways, in this case, the formulas created in the tab CODES are used, which in the case of the DAX 30 go from cell E2 to cell E33, then the fields needed by the function are written, in this case the name of the instrument and P-VAL which corresponds to the previous day valuation Price. The table shown in Figure 18 is obtained,



```
=@GetQuote(CODES!E2:E300,"NAME;P-VAL")
```

Name	24/08/2020
S&P	3130.01
S&P Global Rg	333.25
3M Rg	157.04
A.J.Gallagher Rg	97.34
A.O.Smith Corp Rg	47.32
Abbott Laborator Rg	92.23
AbbVie Rg	98.88
Abiomed Inc Rg	255.22
Accenture-A Rg	215.72
Activision Blizz Rg	78.03
Adobe Rg	439.81
Advance Auto Par Rg	141.84
Advanced Micro D Rg	52.58
Aes Rg	14.48
AFLAC Rg	35.26
Agilent Tech Rg	88.68
Air Prod&Chemica Rg	247.42
Akamai Technolog Rg	106.33
Alaska Air Group Rg	36.49
Albemarle Rg	78.56
Alexandria REIT Rg	165.04

Figure 18: S&P 500 Histo – LAST

The third and most important tab, called HISTO contains for each of the stocks that make up the index the closing price, opening price, minimum price of the day, maximum, and volume (defined as the number of trades). All this information is obtained thanks to the formula entered in cell A3,

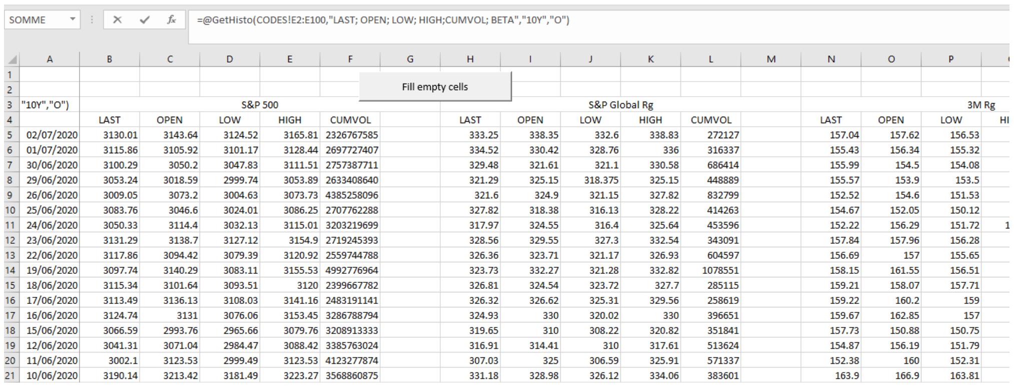
=@GetHisto(CODES!E2:E33,”LAST;OPEN;LOW;HIGH;QTYTRADES”,”10Y”,” O”).

GetHisto() is the FinXL's function that allows to download historical time series from SIX iD directly into Excel. Its construction is very simple and similar to that of the GetQuote() function. First it is necessary to specify the securities from which the financial time series will be obtained. This step can be done in multiple ways, in this case it is done in the same way as before, using the formulas created in the CODES tab, cells E2 to E33 for the case of the DAX. Next the fields commented above are specified. Next, the period to be downloaded is specified, in this case 10 years. Finally by adding 'O' the remaining parameters are set to their default values.

The result is display in the Figure 19 for the DAX 30.

One of the problems faced by anyone who has built a database with financial time series is that of missing entries. While it is true that most of the financial assets included in this work, with the exception of some shares that make up the Mercado Continuo, are large, highly liquid shares, generally traded every day and for which SIX Financial Information always has the necessary price. It is inevitable that from time to time such missing entries will appear, either because for some reason that company wasn't trading that day for its own reasons or because SIX Financial Information doesn't have that data. To solve this problem, a macro has been created that fills in the missing entry. The macro goes through the table and when it finds a missing entry it fills it with the average value between the immediately previous and next values. This is a quite usual way of proceeding in this kind of cases, in particular the way they proceed in the article [16]. In the cases where there are several missing entries in a row, this method may not be the most appropriate, since by doing the average value successively, too low values may be obtained. When this happens, the macro is repeated, but instead of calculating the average value, the last available value is used. This way there is no need to delete that day from the database. The use of the mean is based on the assumption that since it is a daily time series, the variations are usually not important so it is a reasonably satisfactory approximation, especially considering that this does not happen often and it is an observation selected randomly, as we see in J.W. Graham's article, "Missing data analysis: making it work in the real world, 2009" [19].

Figure 19 shows the button that launches the macro that automatically fills the empty cells.



=@GetHisto(CODES!E2:E100,"LAST; OPEN; LOW; HIGH;CUMVOL; BETA","10Y","O")

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																	
2																	
3	"10Y","O")			S&P 500						S&P Global Rg						3M Rg	
4		LAST	OPEN	LOW	HIGH	CUMVOL		LAST	OPEN	LOW	HIGH	CUMVOL		LAST	OPEN	LOW	HI
5	02/07/2020	3130.01	3143.64	3124.52	3165.81	2326767585		333.25	338.35	332.6	338.83	272127		157.04	157.62	156.53	
6	01/07/2020	3115.86	3105.92	3101.17	3128.44	2697727407		334.52	330.42	328.76	336	316337		155.43	156.34	155.32	
7	30/06/2020	3100.29	3050.2	3047.83	3111.51	2757387711		329.48	321.61	321.1	330.58	686414		155.99	154.5	154.08	
8	29/06/2020	3053.24	3018.59	2999.74	3053.89	2633408640		321.29	325.15	318.375	325.15	448889		155.57	153.9	153.5	
9	26/06/2020	3009.05	3073.2	3004.63	3073.73	4385258096		321.6	324.9	321.15	327.82	832799		152.52	154.6	151.53	
10	25/06/2020	3083.76	3046.6	3024.01	3086.25	2707762288		327.82	318.38	316.13	328.22	414263		154.67	152.05	150.12	
11	24/06/2020	3050.33	3114.4	3032.13	3115.01	3203219699		317.97	324.55	316.4	325.64	453596		152.22	156.29	151.72	1
12	23/06/2020	3131.29	3138.7	3127.12	3154.9	2719245393		328.56	329.55	327.3	332.54	343091		157.84	157.96	156.28	
13	22/06/2020	3117.86	3094.42	3079.39	3120.92	2559744788		326.36	323.71	321.17	326.93	604597		156.69	157	155.65	
14	19/06/2020	3097.74	3140.29	3083.11	3155.53	4992776964		323.73	332.27	321.28	332.82	1078551		158.15	161.55	156.51	
15	18/06/2020	3115.34	3101.64	3093.51	3120	2399667782		326.81	324.54	323.72	327.7	285115		159.21	158.07	157.71	
16	17/06/2020	3113.49	3136.13	3108.03	3141.16	2483191141		326.32	326.62	325.31	329.56	258619		159.22	160.2	159	
17	16/06/2020	3124.74	3131	3076.06	3153.45	3286788794		324.93	330	320.02	330	396651		159.67	162.85	157	
18	15/06/2020	3066.59	2993.76	2965.66	3079.76	3208913333		319.65	310	308.22	320.82	351841		157.73	150.88	150.75	
19	12/06/2020	3041.31	3071.04	2984.47	3088.42	3385763024		316.91	314.41	310	317.61	513624		154.87	156.19	151.79	
20	11/06/2020	3002.1	3123.53	2999.49	3123.53	4123277874		307.03	325	306.59	325.91	571337		152.38	160	152.31	
21	10/06/2020	3190.14	3213.42	3181.49	3223.27	3568860875		331.18	328.98	326.12	334.06	383601		163.9	166.9	163.81	

Figure 19: S&P 500 Histo - HISTO

The rest of the spreadsheets that make up the database drink directly from this Histo tab that was just mentioned.

In particular the spreadsheet whose name is composed of the name of the index followed by RSI, such as CAC40 RSI in which the RSI of each share is calculated on a daily basis. To do this, in the first tab, called RSI 14, the following columns are created. The first one contains the date, so that it is updated directly with the last values downloaded in the source file 'Histo' instead of writing a function that returns the date of the last 10 years a call is made to cell A5 of the Histo file, in which there will always be the date of the last data downloaded. The next column calculates the change in absolute terms between the trading day for which the RSI is to be determined and the previous trading day. The next column shows the gain with respect to the previous day, it is simply the value of the change in case it is positive, or 0 in case it is negative. The next column shows it, in an analogous way, for the losses. This is the absolute value of the change in the case of a negative change, 0 in the case of a positive change. Then in the next two columns the average gain and the average loss are calculated respectively. Since this is RSI 14, these averages are calculated with respect to the last 14 sessions. The calculated average represents an exponential moving average. For this purpose, the first of the averages that can be calculated, i.e. after the first 14 days, is calculated using the traditional formula of the arithmetic mean,

$$Mean = \frac{\sum_{i=1}^{i=14} Prices_i}{14}$$

Then for the remaining days the calculation of the average gains and losses is based on an exponential moving average, i.e. giving greater weight to recent values, thus the RSI is expected to more effectively adapt the calculated value to the most recent fluctuations. The formula used is the classical one for the exponential moving average with a

N-days smooting = 14 = $\frac{1}{\alpha}$

$$Mean_i = \alpha \times Prices_i + (1-\alpha) \times Mean_{i-1}$$

Which is equivalent to,

$$Mean_i = \frac{(Mean_{i-1} * (14-1) \ + Prices_i)}{14}$$

Once the average gains and losses have been obtained, the term RS, Relative Strenght, can be calculated. This is the ratio of the average gains to the average losses, thereby, the greater the relative strength, the greater the force of the upward movements relative to the downward movements over the last 14 days.

$$RS = \frac{Average\ Gain}{Average\ Loss}$$

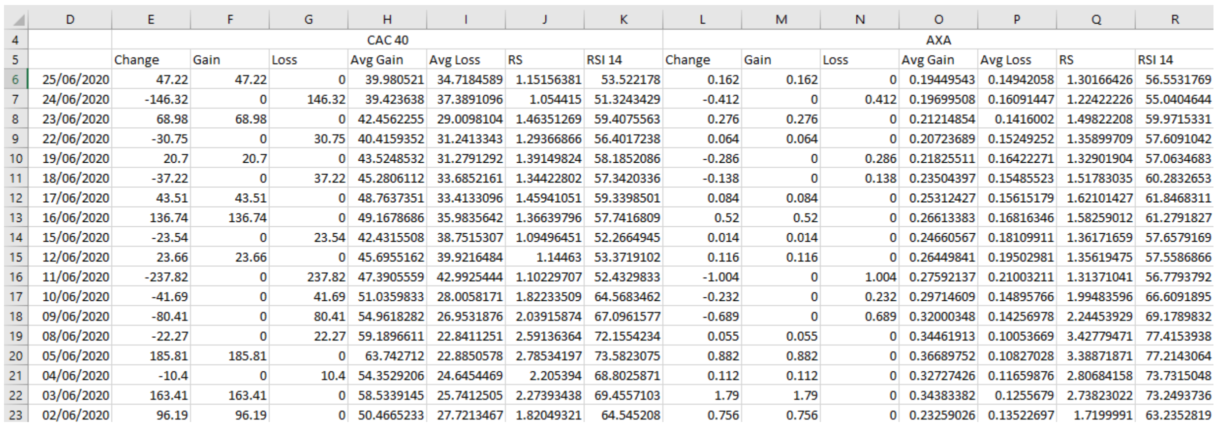
Finally we obtain the RSI by following this formula,

$$RSI = 100 - \frac{100}{1+RS}$$

This converts the Relative Strength into an index, an oscillator whose value varies between 0 and 100 representing the weight or strength that the recent increases have had on the total of recent increases and decreases.

In the case that the average loss for the last 14 days is 0, the formula for the RS term would not work, in which case, it is assumed that the value of the RSI is 100.

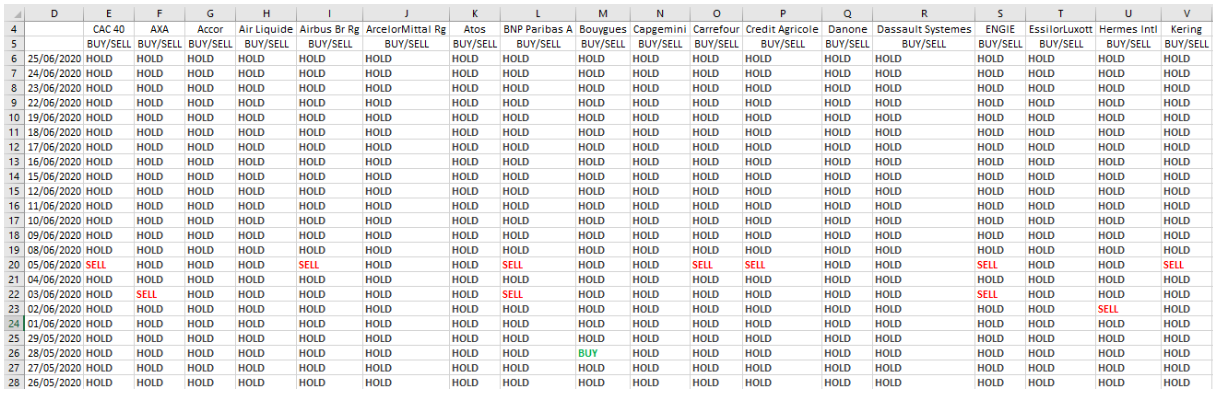
This calculation is made for all the shares that make up the index, thus obtaining the RSI on a daily basis for each share as shown in the Figure 20.



	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
4		CAC 40							AXA						
5		Change	Gain	Loss	Avg Gain	Avg Loss	RS	RSI 14	Change	Gain	Loss	Avg Gain	Avg Loss	RS	RSI 14
6	25/06/2020	47.22	47.22	0	39.980521	34.7184589	1.15156381	53.522178	0.162	0.162	0	0.19449543	0.14942058	1.30166426	56.5531769
7	24/06/2020	-146.32	0	146.32	39.423638	37.3891096	1.054415	51.3243429	-0.412	0	0.412	0.19699508	0.16091447	1.22422226	55.0404644
8	23/06/2020	68.98	68.98	0	42.4562255	29.0098104	1.46351269	59.4075563	0.276	0.276	0	0.21214854	0.1416002	1.49822208	59.9715331
9	22/06/2020	-30.75	0	30.75	40.4159352	31.2413343	1.29366866	56.4017238	0.064	0.064	0	0.20723689	0.15249252	1.35899709	57.6091042
10	19/06/2020	20.7	20.7	0	43.5248532	31.2791292	1.39149824	58.1852086	-0.286	0	0.286	0.21825511	0.16422271	1.32901904	57.0634683
11	18/06/2020	-37.22	0	37.22	45.2806112	33.6852161	1.34422802	57.3420336	-0.138	0	0.138	0.23504397	0.15485523	1.51783035	60.2832653
12	17/06/2020	43.51	43.51	0	48.7637351	33.4133096	1.45941051	59.3398501	0.084	0.084	0	0.25312427	0.15615179	1.62101427	61.8468311
13	16/06/2020	136.74	136.74	0	49.1678686	35.9835642	1.36639796	57.7416809	0.52	0.52	0	0.26613383	0.16816346	1.58259012	61.2791827
14	15/06/2020	-23.54	0	23.54	42.4315508	38.7515307	1.09496451	52.2664945	0.014	0.014	0	0.24660567	0.18109911	1.36171659	57.6579169
15	12/06/2020	23.66	23.66	0	45.6955162	39.9216484	1.14463	53.3719102	0.116	0.116	0	0.26449841	0.19502981	1.35619475	57.5586866
16	11/06/2020	-237.82	0	237.82	47.3905559	42.9925444	1.10229707	52.4329833	-1.004	0	1.004	0.27592137	0.21003211	1.31371041	56.7793792
17	10/06/2020	-41.69	0	41.69	51.0359833	28.0058171	1.82233509	64.5683462	-0.232	0	0.232	0.29714609	0.14895766	1.99483596	66.6091895
18	09/06/2020	-80.41	0	80.41	54.9618282	26.9531876	2.03915874	67.0961577	-0.689	0	0.689	0.32000348	0.14256978	2.24453929	69.1789832
19	08/06/2020	-22.27	0	22.27	59.1896611	22.8411251	2.59136364	72.1554234	0.055	0.055	0	0.34461913	0.10053669	3.42779471	77.4153938
20	05/06/2020	185.81	185.81	0	63.742712	22.8850578	2.78534197	73.5823075	0.882	0.882	0	0.36689752	0.10827028	3.38871871	77.2143064
21	04/06/2020	-10.4	0	10.4	54.3529206	24.6454469	2.205394	68.8025871	0.112	0.112	0	0.32727426	0.11659876	2.80684158	73.7315048
22	03/06/2020	163.41	163.41	0	58.5339145	25.7412505	2.27393438	69.4557103	1.79	1.79	0	0.34383382	0.1255679	2.73823022	73.2493736
23	02/06/2020	96.19	96.19	0	50.4665233	27.7213467	1.82049321	64.545208	0.756	0.756	0	0.23259026	0.13522697	1.7199991	63.2352819

Figure 20: CAC 40 RSI Data

In the next two tabs, the buy and sell signals are calculated and displayed visually so that it is easy and straightforward to check if the calculation that has been made seems correct. If there are too many buy or sell signals or none at all, there is probably an error somewhere in the process, either with the formulas, the references or in the source of the data. At the same time, it is possible to visualize how many signals there have been in the last few years, and how many there are now to get a general idea. The first tab shows the BUY, SELL or HOLD signals for each day of the last 10 years and for each stock in the index. The formula in each cell returns BUY if the RSI for that day is below 30 and the previous day's value is above 30. In other words, the buy signal is shown when the RSI of the stock goes below 30. Similarly, the sell signal is created when the RSI of the stock goes from below 70 to above 70. The following Figure 21 shows the screen with the signals.



	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
4		CAC 40	AXA	Accor	Air Liquide	Airbus Br Rg	ArcelorMittal Rg	Atos	BNP Paribas A	Bouygues	Capgemini	Carrefour	Credit Agricole	Danone	Dassault Systemes	ENGIE	EssilorLuxott	Hermes Intl	Kering
5		BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL	BUY/SELL
6	25/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
7	24/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
8	23/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
9	22/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
10	19/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
11	18/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
12	17/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
13	16/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
14	15/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
15	12/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
16	11/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
17	10/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
18	09/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
19	08/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
20	05/06/2020	SELL	HOLD	HOLD	HOLD	SELL	HOLD	HOLD	SELL	HOLD	HOLD	SELL	SELL	HOLD	HOLD	SELL	HOLD	HOLD	SELL
21	04/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
22	03/06/2020	HOLD	SELL	HOLD	HOLD	HOLD	HOLD	HOLD	SELL	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	SELL	HOLD	HOLD	HOLD
23	02/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	SELL	HOLD
24	01/06/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
25	29/05/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
26	28/05/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	BUY	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
27	27/05/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD
28	26/05/2020	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD	HOLD

Figure 21: CAC 40 RSI – Signals

Finally, the third tab, called Overview, simply shows the current state of the signals on the last day for which data is available by simply referring to that cell.

In the spreadsheets whose name is made up of the name of the index followed by Beta, such as FTSE100 Beta, the beta of each share is calculated in relation to its reference index, i.e. the index to which it belongs, also on a daily basis. The beta of a stock, as mentioned above, represents the systematic risk that the stock has taken with respect to its reference index. That is, how much on average the price of a share has moved in relation to the movement of the index. This is a very useful indicator since it allows to know the relative volatility of a share with respect to the market and thus to have a better understanding of how the share can behave in the face of future price movements. This also allows to determine more precisely what the risk of the portfolio is and how it is expected to fluctuate in the future, if the investor is comfortable with these variations and to evaluate how this risk will vary when adding new investments.

To calculate this indicator, the first step is to determine the return of the stock with respect to the previous day using the formula,

$$Return_i = \frac{(Price_i - Price_{i-1})}{Price_{i-1}} \times 100$$

This return is calculated, in percentage terms for each share and on a daily basis, in addition to the returns of the benchmark whose value is necessary to calculate the beta of each share. Subsequently, the beta of the reference index is set at 1, which is its value by definition.

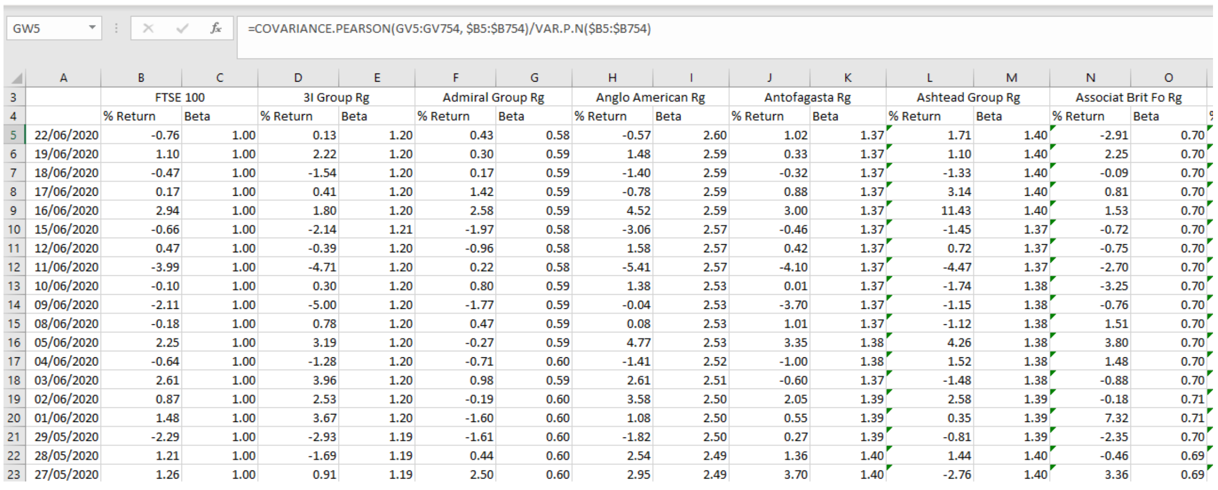
Finally, the beta of each stock is calculated using the following formula,

$$Beta = \frac{Covariance(Return_{stock}\ , Return_{index})}{Variance(Return_{index})}$$

It is necessary to determine which will be the time frame over which the beta will be determined. In this case there is a tradeoff between the fact that the greater the time frame, the greater the amount of data and hence, the greater stability of the beta obtained is expected. On the other hand if the period is too long there is a risk that at the company or market level there has been some change in between that makes the characteristics of the relative volatility between the two change, making it difficult to estimate the value of the calculated beta. In this case, it was decided to use the last 750 sessions, i.e. approximately 3 years. Although it is true that different ways of calculating the beta can lead to different results, for example, in some places it is calculated with a time frame of 5 years or 3 months, with daily data as in this case or with monthly values. However, what is important about this indicator is not so much the absolute value as the relative one. What is being compared is the beta of one stock with respect to another, and whatever formula is chosen, this relationship should be maintained in general. In any case, the final formula used in Excel to calculate the beta is the following,

$$Beta = \frac{Covariance.Pearson(Return_{stock}(1)\colon Return_{stock}(754),\ Return_{index}(1)\colon Return_{index}(754))}{VAR.P.N(Return_{index}(1)\colon Return_{index}(754))}$$

The final outcome of this table is shown in Figure 22,



GW5 =COVARIANCE.PEARSON(GV5:GV754, $B5:$B754)/VAR.P.N($B5:$B754)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
3		FTSE 100		3I Group Rg		Admiral Group Rg		Anglo American Rg		Antofagasta Rg		Ashtead Group Rg		Associat Brit Fo Rg	
4		% Return	Beta	% Return	Beta	% Return	Beta	% Return	Beta	% Return	Beta	% Return	Beta	% Return	Beta
5	22/06/2020	-0.76	1.00	0.13	1.20	0.43	0.58	-0.57	2.60	1.02	1.37	1.71	1.40	-2.91	0.70
6	19/06/2020	1.10	1.00	2.22	1.20	0.30	0.59	1.48	2.59	0.33	1.37	1.10	1.40	2.25	0.70
7	18/06/2020	-0.47	1.00	-1.54	1.20	0.17	0.59	-1.40	2.59	-0.32	1.37	-1.33	1.40	-0.09	0.70
8	17/06/2020	0.17	1.00	0.41	1.20	1.42	0.59	-0.78	2.59	0.88	1.37	3.14	1.40	0.81	0.70
9	16/06/2020	2.94	1.00	1.80	1.20	2.58	0.59	4.52	2.59	3.00	1.37	11.43	1.40	1.53	0.70
10	15/06/2020	-0.66	1.00	-2.14	1.21	-1.97	0.58	-3.06	2.57	-0.46	1.37	-1.45	1.37	-0.72	0.70
11	12/06/2020	0.47	1.00	-0.39	1.20	-0.96	0.58	1.58	2.57	0.42	1.37	0.72	1.37	-0.75	0.70
12	11/06/2020	-3.99	1.00	-4.71	1.20	0.22	0.58	-5.41	2.57	-4.10	1.37	-4.47	1.37	-2.70	0.70
13	10/06/2020	-0.10	1.00	0.30	1.20	0.80	0.59	1.38	2.53	0.01	1.37	-1.74	1.38	-3.25	0.70
14	09/06/2020	-2.11	1.00	-5.00	1.20	-1.77	0.59	-0.04	2.53	-3.70	1.37	-1.15	1.38	-0.76	0.70
15	08/06/2020	-0.18	1.00	0.78	1.20	0.47	0.59	0.08	2.53	1.01	1.37	-1.12	1.38	1.51	0.70
16	05/06/2020	2.25	1.00	3.19	1.20	-0.27	0.59	4.77	2.53	3.35	1.38	4.26	1.38	3.80	0.70
17	04/06/2020	-0.64	1.00	-1.28	1.20	-0.71	0.60	-1.41	2.52	-1.00	1.38	1.52	1.38	1.48	0.70
18	03/06/2020	2.61	1.00	3.96	1.20	0.98	0.59	2.61	2.51	-0.60	1.37	-1.48	1.38	-0.88	0.70
19	02/06/2020	0.87	1.00	2.53	1.20	-0.19	0.60	3.58	2.50	2.05	1.39	2.58	1.39	-0.18	0.71
20	01/06/2020	1.48	1.00	3.67	1.20	-1.60	0.60	1.08	2.50	0.55	1.39	0.35	1.39	7.32	0.71
21	29/05/2020	-2.29	1.00	-2.93	1.19	-1.61	0.60	-1.82	2.50	0.27	1.39	-0.81	1.39	-2.35	0.70
22	28/05/2020	1.21	1.00	-1.69	1.19	0.44	0.60	2.54	2.49	1.36	1.40	1.44	1.40	-0.46	0.69
23	27/05/2020	1.26	1.00	0.91	1.19	2.50	0.60	2.95	2.49	3.70	1.40	-2.76	1.40	3.36	0.69

Figure 22: FTSE 100 – Beta

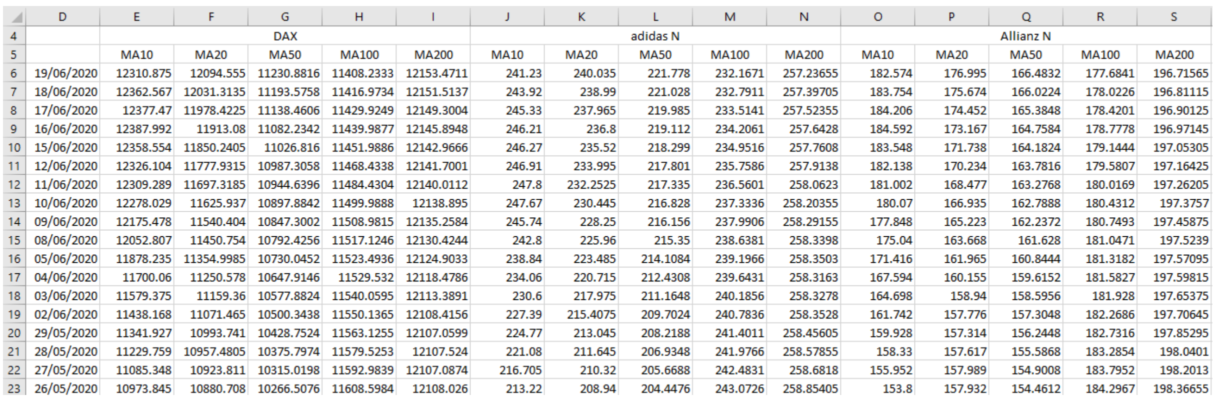
The second tab of this spreadsheet simply shows an overview with the last daily beta value of each stock.

Finally, a spreadsheet is added to the database containing the moving averages of the last 10, 20, 50, 100 and 200 sessions on a daily basis and for each stock. Generally for strategies that use moving averages only two are needed, a fast one that takes into account a small number of last sessions and a slow one, with a larger number of sessions. Therefore, the fast one will represent the recent movements and will adapt to them quickly while the slow one will represent the more general tendency, being much more stable. Since the calculation of a moving average is not very complicated, several moving averages with different values are calculated so that the choice can be made between a greater number of possibilities when developing the strategy.

The formula for the moving average used is shown below,

$$MA_i = \frac{Price\ (1) : Price(i)}{i}$$

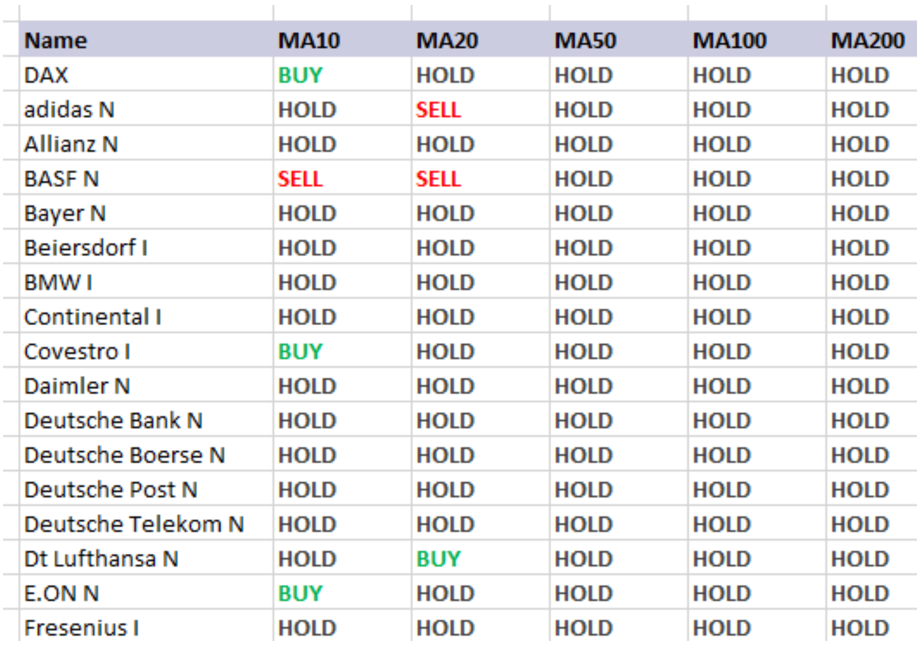
The final outcome is shown in Figure 23.



	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
4		DAX					adidas N					Allianz N				
5		MA10	MA20	MA50	MA100	MA200	MA10	MA20	MA50	MA100	MA200	MA10	MA20	MA50	MA100	MA200
6	19/06/2020	12310.875	12094.555	11230.8816	11408.2333	12153.4711	241.23	240.035	221.778	232.1671	257.23655	182.574	176.995	166.4832	177.6841	196.71565
7	18/06/2020	12362.567	12031.3135	11193.5758	11416.9734	12151.5137	243.92	238.99	221.028	232.7911	257.39705	183.754	175.674	166.0224	178.0226	196.81115
8	17/06/2020	12377.47	11978.4225	11138.4606	11429.9249	12149.3004	245.33	237.965	219.985	233.5141	257.52355	184.206	174.452	165.3848	178.4201	196.90125
9	16/06/2020	12387.992	11913.08	11082.2342	11439.9877	12145.8948	246.21	236.8	219.112	234.2061	257.6428	184.592	173.167	164.7584	178.7778	196.97145
10	15/06/2020	12358.554	11850.2405	11026.816	11451.9886	12142.9666	246.27	235.52	218.299	234.9516	257.7608	183.548	171.738	164.1824	179.1444	197.05305
11	12/06/2020	12326.104	11777.9315	10987.3058	11468.4338	12141.7001	246.91	233.995	217.801	235.7586	257.9138	182.138	170.234	163.7816	179.5807	197.16425
12	11/06/2020	12309.289	11697.3185	10944.6396	11484.4304	12140.0112	247.8	232.2525	217.335	236.5601	258.0623	181.002	168.477	163.2768	180.0169	197.26205
13	10/06/2020	12278.029	11625.937	10897.8842	11499.9888	12138.895	247.67	230.445	216.828	237.3336	258.20355	180.07	166.935	162.7888	180.4312	197.3757
14	09/06/2020	12175.478	11540.404	10847.3002	11508.9815	12135.2584	245.74	228.25	216.156	237.9906	258.29155	177.848	165.223	162.2372	180.7493	197.45875
15	08/06/2020	12052.807	11450.754	10792.4256	11517.1246	12130.4244	242.8	225.96	215.35	238.6381	258.3398	175.04	163.668	161.628	181.0471	197.5239
16	05/06/2020	11878.235	11354.9985	10730.0452	11523.4936	12124.9033	238.84	223.485	214.1084	239.1966	258.3503	171.416	161.965	160.8444	181.3182	197.57095
17	04/06/2020	11700.06	11250.578	10647.9146	11529.532	12118.4786	234.06	220.715	212.4308	239.6431	258.3163	167.594	160.155	159.6152	181.5827	197.59815
18	03/06/2020	11579.375	11159.36	10577.8824	11540.0595	12113.3891	230.6	217.975	211.1648	240.1856	258.3278	164.698	158.94	158.5956	181.928	197.65375
19	02/06/2020	11438.168	11071.465	10500.3438	11550.1365	12108.4156	227.39	215.4075	209.7024	240.7836	258.3528	161.742	157.776	157.3048	182.2686	197.70645
20	29/05/2020	11341.927	10993.741	10428.7524	11563.1255	12107.0599	224.77	213.045	208.2188	241.4011	258.45605	159.928	157.314	156.2448	182.7316	197.85295
21	28/05/2020	11229.759	10957.4805	10375.7974	11579.5253	12107.524	221.08	211.645	206.9348	241.9766	258.57855	158.33	157.617	155.5868	183.2854	198.0401
22	27/05/2020	11085.348	10923.811	10315.0198	11592.9839	12107.0874	216.705	210.32	205.6688	242.4831	258.6818	155.952	157.989	154.9008	183.7952	198.2013
23	26/05/2020	10973.845	10880.708	10266.5076	11608.5984	12108.026	213.22	208.94	204.4476	243.0726	258.85405	153.8	157.932	154.4612	184.2967	198.36655

Figure 23: DAX 30 – MA Data

Finally in the second tab of this spreadsheet is also added an Overview that enables to know which are the BUY, SELL or HOLD signals that can be deduced from the last data downloaded. In this case the signals are created separately, simply showing a buy signal if the last price downloaded is higher than the moving average, whatever its period, when the price the day before was lower than the moving average. This is done in an analogous way for the sell signal. SELL is returned if the last price downloaded is lower than the moving average while the previous day's price was higher than the moving average. In all other cases HOLD is returned. Figure 24 shows the final result.



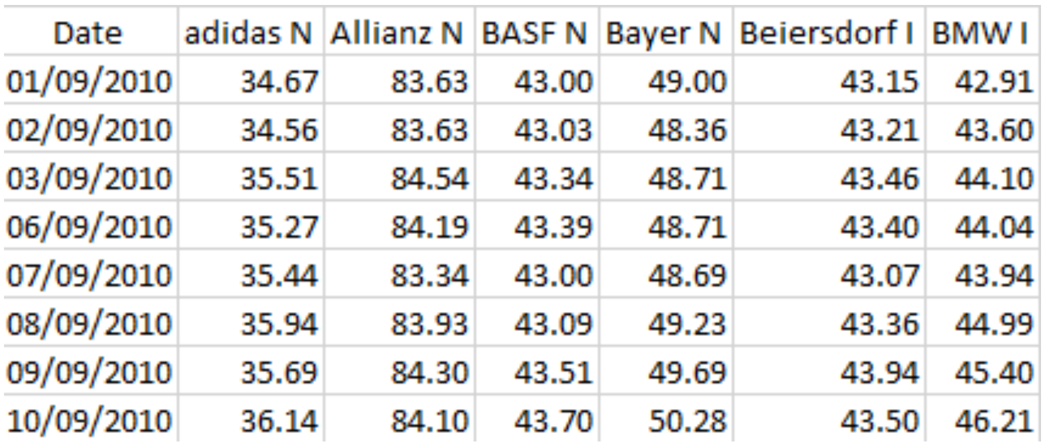
Name	MA10	MA20	MA50	MA100	MA200
DAX	BUY	HOLD	HOLD	HOLD	HOLD
adidas N	HOLD	SELL	HOLD	HOLD	HOLD
Allianz N	HOLD	HOLD	HOLD	HOLD	HOLD
BASF N	SELL	SELL	HOLD	HOLD	HOLD
Bayer N	HOLD	HOLD	HOLD	HOLD	HOLD
Beiersdorf I	HOLD	HOLD	HOLD	HOLD	HOLD
BMW I	HOLD	HOLD	HOLD	HOLD	HOLD
Continental I	HOLD	HOLD	HOLD	HOLD	HOLD
Covestro I	BUY	HOLD	HOLD	HOLD	HOLD
Daimler N	HOLD	HOLD	HOLD	HOLD	HOLD
Deutsche Bank N	HOLD	HOLD	HOLD	HOLD	HOLD
Deutsche Boerse N	HOLD	HOLD	HOLD	HOLD	HOLD
Deutsche Post N	HOLD	HOLD	HOLD	HOLD	HOLD
Deutsche Telekom N	HOLD	HOLD	HOLD	HOLD	HOLD
Dt Lufthansa N	HOLD	BUY	HOLD	HOLD	HOLD
E.ON N	BUY	HOLD	HOLD	HOLD	HOLD
Fresenius I	HOLD	HOLD	HOLD	HOLD	HOLD

Figure 24: DAX 30 – MA – Signals

This concludes the creation of the database. Below is shown how a simplified sub-database has been created, developed entirely from the first and main database, with only the values that have been finally used in the final tool developed in MATLAB. While it is true that the first database could have been used directly by selecting the necessary values from MATLAB, the decision was made to make a first selection that would facilitate and lighten the code of the backtesting tool.

The backtesting tool calls from MATLAB 6 different files for each index. Using the DAX 30 as an example these 8 files would be, DAX30, DAX30_Beta, DAX30_RSI, DAX30_Benchmark, DAX30_Benchmark_Beta, DAX30_Benchmark_RSI. The name of the file and its contents follow the same structure for each of the indexes.

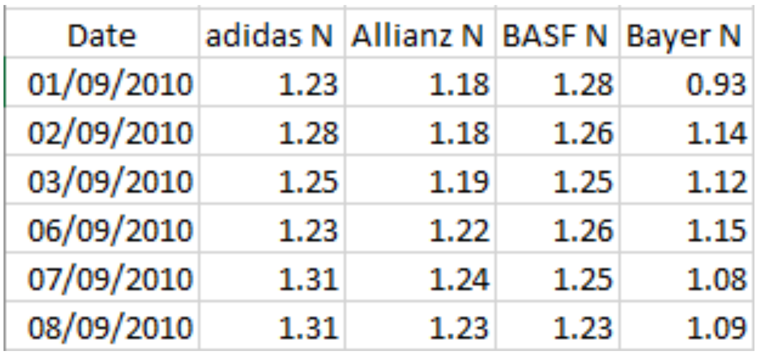
The first of the files, DAX30, contains only the column called LAST included in the files of type Histo, with the closing price of each of the DAX 30 shares, not including the closing prices of the index. The period is daily and from 01/09/2010 to 30/08/2019, the direction in which the table is organized is from older to recent, which is important because it is in this way that MATLAB requires the tables that make up the *Timetable* data structure to be organized. This is the structure that has been chosen for this type of table in the entire backtesting tool. The table in Excel also includes the company names above the closing values because they will be needed later when doing the backtest. The result is shown in Figure 25.



Date	adidas N	Allianz N	BASF N	Bayer N	Beiersdorf I	BMW I
01/09/2010	34.67	83.63	43.00	49.00	43.15	42.91
02/09/2010	34.56	83.63	43.03	48.36	43.21	43.60
03/09/2010	35.51	84.54	43.34	48.71	43.46	44.10
06/09/2010	35.27	84.19	43.39	48.71	43.40	44.04
07/09/2010	35.44	83.34	43.00	48.69	43.07	43.94
08/09/2010	35.94	83.93	43.09	49.23	43.36	44.99
09/09/2010	35.69	84.30	43.51	49.69	43.94	45.40
10/09/2010	36.14	84.10	43.70	50.28	43.50	46.21

Figure 25: Table containing the closing prices for the DAX 30 sub - database

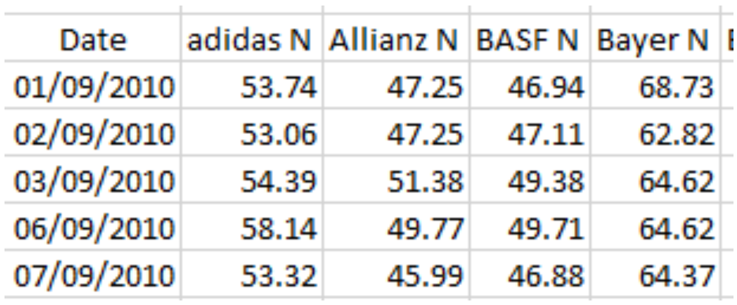
The second of the files, DAX30_Beta, contains only the column with the beta of each stock, again without including the beta of the index and also ordered from oldest to recent starting on 01/09/2010 until 30/08/2019.In this case it's also important that the first cell of each column contains the name of the company, which has to follow the same order in all the files of the same index. Figure 26 below shows the final outcome.



Date	adidas N	Allianz N	BASF N	Bayer N
01/09/2010	1.23	1.18	1.28	0.93
02/09/2010	1.28	1.18	1.26	1.14
03/09/2010	1.25	1.19	1.25	1.12
06/09/2010	1.23	1.22	1.26	1.15
07/09/2010	1.31	1.24	1.25	1.08
08/09/2010	1.31	1.23	1.23	1.09

Figure 26: Table containing the daily Beta for the DAX 30 sub - database

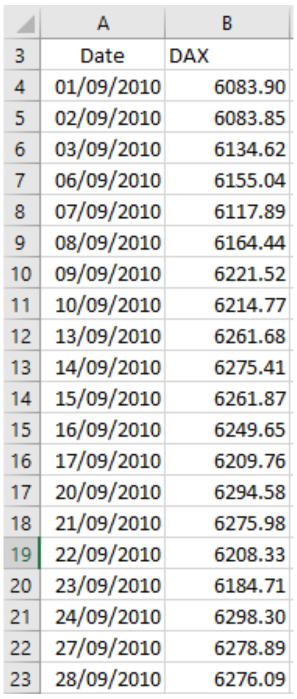
The third of the files, DAX30_RSI, contains only the column with the final result of the RSI, ignoring the intermediate calculations. Following the same structure as in the previous files, the final result is shown below in Figure 27.



Date	adidas N	Allianz N	BASF N	Bayer N
01/09/2010	53.74	47.25	46.94	68.73
02/09/2010	53.06	47.25	47.11	62.82
03/09/2010	54.39	51.38	49.38	64.62
06/09/2010	58.14	49.77	49.71	64.62
07/09/2010	53.32	45.99	46.88	64.37

Figure 27: Table containing the daily RSI for the DAX 30 sub - database

The fourth of the files, DAX30_Benchmark, contains only the column called LAST of the Histo files, exclusively with the closing prices of the benchmark index, in this case the S&P 500. The information related to the index has been separated from that of the rest of the stocks that compose it, since it facilitates the management of the data in the MATLAB tool and is necessary for the development of certain strategies. All Benchmark files also follow the same structure as the previous files, i.e. data starts on 01/09/2010 and is organized from older to more recent until 30/08/2019. The final result is shown below in Figure 28.

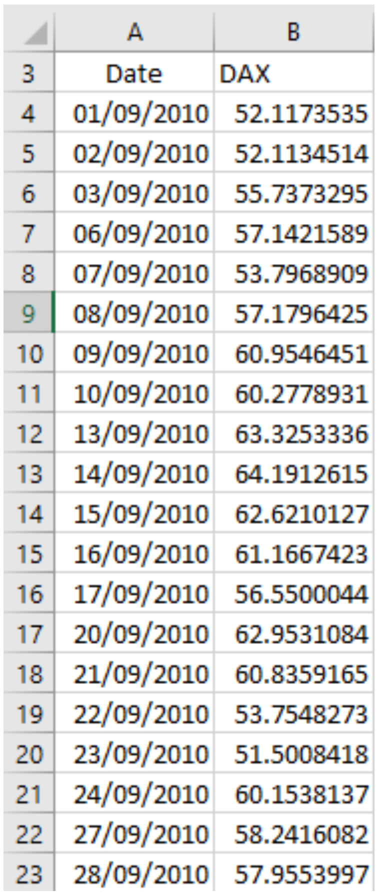


	A	B
3	Date	DAX
4	01/09/2010	6083.90
5	02/09/2010	6083.85
6	03/09/2010	6134.62
7	06/09/2010	6155.04
8	07/09/2010	6117.89
9	08/09/2010	6164.44
10	09/09/2010	6221.52
11	10/09/2010	6214.77
12	13/09/2010	6261.68
13	14/09/2010	6275.41
14	15/09/2010	6261.87
15	16/09/2010	6249.65
16	17/09/2010	6209.76
17	20/09/2010	6294.58
18	21/09/2010	6275.98
19	22/09/2010	6208.33
20	23/09/2010	6184.71
21	24/09/2010	6298.30
22	27/09/2010	6278.89
23	28/09/2010	6276.09

Figure 28: DAX 30 Benchmark – Closing Prices

The fifth file, DAX30_Benchmark_Beta, contains only the benchmark index beta, which by definition is always 1.

The last file, DAX30_Benchmark_RSI, contains only the RSI value of the reference index, ignoring the intermediate calculations. The final result is shown in Figure 29 below.



	A	B
3	Date	DAX
4	01/09/2010	52.1173535
5	02/09/2010	52.1134514
6	03/09/2010	55.7373295
7	06/09/2010	57.1421589
8	07/09/2010	53.7968909
9	08/09/2010	57.1796425
10	09/09/2010	60.9546451
11	10/09/2010	60.2778931
12	13/09/2010	63.3253336
13	14/09/2010	64.1912615
14	15/09/2010	62.6210127
15	16/09/2010	61.1667423
16	17/09/2010	56.5500044
17	20/09/2010	62.9531084
18	21/09/2010	60.8359165
19	22/09/2010	53.7548273
20	23/09/2010	51.5008418
21	24/09/2010	60.1538137
22	27/09/2010	58.2416082
23	28/09/2010	57.9553997

Figure 29: DAX 30 Benchmark – RSI

Once these databases have been established, it is possible to start developing the backtesting tool to evaluate investment strategies based on these data.

4. Implementation - Backtesting

4.1 Development of the Backtesting Tool

This section explains how the tool has been developed to allow the backtesting of the strategies. MATLAB will be used to prepare the data collected from the database mentioned in the previous section, apply the different strategies, manage the portfolio created from the strategy, and analyze the results. MATLAB is an extremely complete program, providing an integrated development environment in which it is possible to develop and test different scripts written in its own programming language, called M language. MATLAB is short for MATrix LABoratory, since matrices are the core element of MATLAB. Matrix manipulation is its strength, and whenever possible, approaching the code in terms of matrix manipulation rather than in loops will unlock the full power of MATLAB. Thus, the performance and computation speed of the programs will be optimal.

MATLAB has been chosen as the development environment for many reasons. First, one of the features of MATLAB is that in addition to the general basic development environment, there are a number of Toolboxes specifically designed to solve a particular type of problem or for a particular type of application. In particular for this work, the Statistics and Machine Learning, Financial, Optimization, and Deep Learning Toolboxes have been needed. These kind of specific libraries are especially useful because they allow the use of certain parts of code already developed, which lightens the workload, particularly in certain trivial functions such as the calculation of certain indicators, certain ratios and in the part of the analysis of results, such as the calculation of the Sharpe Ratio, for which, as we will see later, there is already a function belonging to the financial toolbox that allows to obtain this ratio from a time series of returns. It is also completely essential for the development of Machine Learning and Deep Learning algorithms or for the optimization of certain functions.

Another interesting element of MATLAB, related to the previous one, is that it has a very important and active community, in which there is a huge amount of webinars, videos, and file exchanges among MATLAB users, as well as between professional MATLAB developers and the rest of the users. All of this content is public, well documented, and explained, making it easy to download and use code developed by other programmers, as if it were a library. On the other hand, it also allows to be inspired and realize the enormous potential that MATLAB has in terms of applications for the financial sector and in the development of artificial intelligence tools. In particular the files and webinars “Walk-Forward Analysis: Using MATLAB to Backtest Your Trading Strategy” and “Classifying Trading Signals using Machine Learning and Deep Learning” by Kawee Numpacharoen have been very useful.

The first stage in the development of the backtesting tool is a series of scripts common to all strategies, they will have to be launched first and for any strategy, whatever its characteristics. It involves the preparation and pre-processing of the data necessary for the particular strategy to be implemented. This part has been developed in 3 parts, using 3 different MATLAB scripts.

The first one of the scripts, simply retrieves the information contained in the database regarding the closing prices of the different indices, as well as the closing prices of the shares that make them up. This script is called data1_Retrieving.mlx, most of the scripts in this work, and in particular all in which, due to its high relevance, the user spends a lot of time on them, have been developed in Live Code File format, i.e. .mlx files. This is mainly due to the fact that this type of scripts enables to insert text, as shown below, which makes the script much more understandable as well as more pleasant and simple to work with. This is especially important as the project grows in size and the number of scripts increases, since it allows the programmer to quickly know what the program he is opening is doing thanks to the initial explanations. This element is crucial and indispensable when a new programmer is analyzing the code, since if it were a whole block of code without any explanation it would be extremely complicated and tedious to understand and analyze. This type of format has more additional advantages as it is interoperable between each country's own MATLAB configurations, the Spanish and Russian versions do not have the same local configurations. In addition, it is an extensible file that is compatible with both previous and future versions of MATLAB.

The program therefore starts with a short explanation of what the program does and with the commands to clear the variables stored in the workspace at that time, the lines of code and their responses in the Command Window, and closing all the windows or graphs that MATLAB has open.

Load Data

The objective of this file is to load historical prices from our database into MATLAB work space and store them in TimeTable format.

```
clear;
close all;
clc;
```

Then the code of the first script starts. The way to proceed is the following, practically the same lines of code are executed for each of the benchmark indexes. This strategy will be followed for all the scripts. It has been chosen to do it in this way because it allows to do separately each one of the indexes, which is useful in the case in which it is only wanted to do the backesting of the strategy for a single index. It also allows to divide the code in different independent modules, making it more understandable. Below are the lines of code that enables to collect the information from the database and store it in the *Timetable* format, in the case of the S&P 500 to give an example.

S&P 500

```
SP_data = readtable('SP500.xlsx','PreserveVariableNames',true);
SP_price = SP_data(:,2:end);
SP_prices = table2array(SP_price);
SP500_nStocks = width(SP_price);
```

Store data into TimeTable format

```
SP500_Data = table2timetable(SP_data);
SP500_Data.Properties.DimensionNames{1} = 'time';
SP500_Data.Properties.DimensionNames{2} = 'prices';
namesSP = SP500_Data.Properties.VariableNames;
```

As it can be seen in the above code, being able to alternate between code and text makes the code easier to read. The same lines of code are repeated for each of the indices, firstly collecting the data of the shares that make it up and then collecting the information of the closing price of the indices themselves, i.e. from the 'Benchmark' files that we saw in the previous section.

S&P 500 Benchmark

To train the Deep Learning and Machine Learning with the indexes

```
SP500_Benchmark_data =
readtable('SP500_Benchmark.xlsx','PreserveVariableNames',true);
SP500_Benchmark_price = SP500_Benchmark_data(:,2:end);
SP500_Benchmark_prices = table2array(SP500_Benchmark_price);
SP500_Benchmark_nStocks = width(SP500_Benchmark_price);
```

Store data into TimeTable format

```
SP500_Benchmark_Data = table2timetable(SP500_Benchmark_data);
```

```
SP500_Benchmark_Data.Properties.DimensionNames{1} = 'time';
SP500_Benchmark_Data.Properties.DimensionNames{2} = 'prices';
namesSP500_Benchmark = SP500_Benchmark_Data.Properties.VariableNames;
```

Finally the auxiliary variables are deleted and the variables that will be needed later are stored in a matrix called data1.mat, as shown below.

SAVE

```
clearvars -except SP500_Data ATX_Data CAC40_Data DAX30_Data
FTSE_MIB_Data FTSE100_Data MC_Data SP_TSX60_Data ...
    namesSP namesATX namesCAC40 namesDAX30 namesFTSE_MIB namesFTSE100
namesMC namesSP_TSX60 SP500_nStocks ...
    ATX_nStocks CAC40_nStocks DAX30_nStocks FTSE_MIB_nStocks
FTSE100_nStocks MC_nStocks SP_TSX60_nStocks ...
    namesATX_Benchmark ATX_Benchmark_nStocks ATX_Benchmark_Data
namesSP500_Benchmark SP500_Benchmark_nStocks ...
    SP500_Benchmark_Data namesCAC_Benchmark CAC_Benchmark_nStocks
CAC_Benchmark_Data namesDAX_Benchmark DAX_Benchmark_nStocks ...
    DAX_Benchmark_Data namesFTSE_MIB_Benchmark
FTSE_MIB_Benchmark_nStocks FTSE_MIB_Benchmark_Data
namesFTSE100_Benchmark ...
    FTSE100_Benchmark_nStocks FTSE100_Benchmark_Data namesMC_Benchmark
MC_Benchmark_nStocks MC_Benchmark_Data namesSP_TSX60_Benchmark ...
    SP_TSX60_Benchmark_nStocks SP_TSX60_Benchmark_Data
save('data1.mat');
```

The second script is one of the most important, since it is used to pre-process the data. In other words, all the factors necessary for the implementation of the different investment strategies are calculated and created. This script is called data2_Preprocessing and it starts as the previous one, with a small description of what this program is intended to achieve, followed by the commands to clean the saved variables and the Command Window. It continues by loading the data1.mat matrix to be able to launch the code only with the information contained in that matrix, without other variables that could interfere with the correct functioning of the code. Here is how it is done,

Data Preprocessing

In this file, we are going to perform data preprocessing, including, data cleaning, data factor creation, data reformatting, and data partitioning.

```
clear;
close all;
clc;
load data1.mat
```

We call the function Preprocessing with each of the Timetables

This script is structured on the same principle as the previous one. The script is separated in modules, each one contains the function Preprocessing(), which allows to make the pre-processing of the data. Each one of the modules therefore pre-process the data for each index, both for the stocks that make up the index and for the benchmark index itself, separately. This means that each module calls the function only once, passing as an argument the data relative to the index in question. There is a total of 16 modules 8 for the files that contain the shares that compose the indexes and other 8 for the files that contain the data of the price movements of the correspondent benchmarks.

S&P500

```
c = 'SP500';
[SP500_Data, SP500_DataML, SP500_prices, SP500_Days, SP500_x, SP500_y,
SP500_prices_stats] = Preprocessing(SP500_Data, SP500_nStocks, c);
```

ATX

```
c = 'ATX';
[ATX_Data, ATX_DataML, ATX_prices, ATX_Days, ATX_x, ATX_y,
ATX_prices_stats] = Preprocessing(ATX_Data, ATX_nStocks, c);
```

Below, an explanation of how the Preprocessing() function works.

First of all, the rows in the Timetable that contain missing values are removed. This shouldn't happen since a cleanup has been done in the different steps of the database construction, however there's no way to know if for some reason there is still one or several missing values. If that's the case the program wouldn't work so in order to make it more robust it's important to remove the row, it is done with the next line of code.

```
function [T, TML, prices, Days, x, y, pricesmean] = Preprocessing(T,
nbStocks, c)

    %% Data cleaning: missing values could be because the company
wasn't yet listed

    T = fillmissing(T,"constant",0);
```

Secondly, the temporal factors are created, as they could have some effect on the movement of prices as can be seen in these two articles “The day-of-the-Week effects of stock markets in different countries” by Jilin Zhang Yongzeng Lai Jianghong Lin [20] and “The Impact of Public Information on the Stock Market” by Mark L. Mitchell J. Harold Mulherin [21]. It seems that there is no reason to believe that the day of the week, month, year or any other temporal factor should have an influence on the financial markets, there seems to be no economic reason for this. However, it does seem that they can be useful, not in strategies that directly rely on them alone, but in artificial intelligence strategies. The idea behind their use is that, although for a human they do not seem to have any relevance, a neural network or a classification method based on Machine Learning might appreciate some kind of time pattern and put it into practice. The time predictors that have been calculated are the day of the week, with day 1 being Sunday, the number of days left until the next trading day and the number of days that have passed since the last trading day. The date of each data is also separated in day, month and year, obtaining three additional indicators for a total of 6 temporary predictors.

Below is the code that enables this process to be performed.

```
% Factor Creation
    % Create datetime factors
    % Separate datetime into day, month, year

    [T.y,T.m,T.d] = ymd(T.time);

    % Create day of the week where day 1 of the week is Sunday

    T.dayOfWeek = day(T.time,'dayofweek');

    % Number of days to next trading day and number of day from
previous
    % trading day

    dayDiff = diff(datenum(T.time));
    T.nNextTD = [dayDiff; NaN];
    T.nPreviousTD = [NaN; dayDiff];
```

In addition to the temporal predictors, the program calculates and adds from the database some indicators based on the closing prices, which will also serve as predictors in the investment strategies. First of all, the returns with different time periods are calculated. The first one is daily, which is simply the return compared to the previous day. The second corresponds to the semi-annual return, i.e. for the last 6 months or the last 126 sessions. Finally, the annual return is calculated, i.e. from the last 252 sessions. To obtain these returns, the double for-loop shown below is required,

```
% Generate price related indicators as predictors
    % Generate n-day returns

    nDay1 = [1; 126; 252];
    [~,n] = size(T);
    for i = 1:numel(nDay1)
        for j = 1:(n-6)
            factorName =
append(string(T.Properties.VariableNames(j)),['Ret'
num2str(nDay1(i))]);
            T.(factorName) = nDayReturn(T.(j), nDay1(i));
        end
    end
```

In it the nDayReturn function that has been created is called, for each stock and each day, in addition it also allows to create and add the columns with the returns to the initial *Timetable* matrix. Below is the nDayreturn function.

Supporting functions

```
function returns = nDayReturn(prices, nDay)
    % Compute return from prices
    if nDay == 1
        returns = NaN(size(prices,1),1);
        returns(nDay+1:end) = prices(nDay+1:end)./ prices(1:end-nDay)-
1;
    else
        % For the longer term returns we exclude the last month to
avoid
        % microstructure and liquidity biases
        returns = NaN(size(prices,1),1);
        returns(nDay+1:end) = prices(nDay+1-21:end-21) ./ prices(1:end-
nDay) - 1;
    end
end
```

This function simply use the classic formula for the return,

$$Return_i = \frac{Prices_i}{Prices_{i-1}} - 1$$

In order to take advantage of MATLAB's full potential, the formula is in a matrix format so it is not required to use a for-loop as seen in the function. The function has been divided into 2 parts, one for the daily return, since it is a short term return, and another for the semi-annual and annual returns, since they are long term. For the second case, the prices of the last month are not taken into account since they can cause microstructure and liquidity biases as it is explained in the paper “Fact, Fiction and Momentum investing” by Clifford Asness, Andrea Frazzini, Ronen Israel, and Tobias Moskowitz [22]. That is, if, for example, we want the annual return of July 20, 2020, we use the prices from July 20, 2019 to June 20, 2020, this way of proceeding is usually noted as Momentum(12,2).

Then, the beta and RSI values included in the database are added to the initial Timetable. To do this, first of all, the following switch structure is used, to load only the necessary files, relative to the *Timetable* that was passed as an argument of the function.

```
switch c
        case 'SP500'
            dataRSI =
readtable("SP500_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("SP500_Beta.xlsx",'PreserveVariableNames',true);
        case 'ATX'
            dataRSI =
readtable("ATX_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("ATX_Beta.xlsx",'PreserveVariableNames',true);
        case 'CAC'
```

```
            dataRSI =
readtable("CAC40_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("CAC40_Beta.xlsx",'PreserveVariableNames',true);
        case 'DAX'
            dataRSI =
readtable("DAX30_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("DAX30_Beta.xlsx",'PreserveVariableNames',true);
        case 'FTSE_MIB'
            dataRSI =
readtable("FTSE_MIB_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("FTSE_MIB_Beta.xlsx",'PreserveVariableNames',true);
        case 'FTSE100'
            dataRSI =
readtable("FTSE100_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("FTSE100_Beta.xlsx",'PreserveVariableNames',true);
        case 'MC'
            dataRSI =
readtable("Mercado_Continuo_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("Mercado_Continuo_Beta.xlsx",'PreserveVariableNames',true);
        case 'SP_TSX60'
            dataRSI =
readtable("SP_TSX60_RSI.xlsx",'PreserveVariableNames',true);
            dataBeta =
readtable("SP_TSX60_Beta.xlsx",'PreserveVariableNames',true);
        case 'SP500_Benchmark'
            dataRSI = readtable("SP500_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("SP500_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
        case 'ATX_Benchmark'
            dataRSI = readtable("ATX_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("ATX_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
        case 'CAC_Benchmark'
            dataRSI = readtable("CAC40_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("CAC40_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
        case 'DAX_Benchmark'
            dataRSI = readtable("DAX30_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("DAX30_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
        case 'FTSE_MIB_Benchmark'
            dataRSI = readtable("FTSE_MIB_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("FTSE_MIB_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
```

```
        case 'FTSE100_Benchmark'
            dataRSI = readtable("FTSE100_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("FTSE100_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
        case 'MC_Benchmark'
            dataRSI = readtable("Mercado_Continuo_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta =
readtable("Mercado_Continuo_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
        case 'SP_TSX60_Benchmark'
            dataRSI = readtable("SP_TSX60_Benchmark_RSI.xlsx",
'PreserveVariableNames', true);
            dataBeta = readtable("SP_TSX60_Benchmark_Beta.xlsx",
'PreserveVariableNames', true);
    end
```

Then, using the following for loop, a column is created in the *Timetable* and the RSI data is added, putting as the name of the column the name of the stock followed by RSI, as seen below.

```
% Generate 14-day RSI

    RSI_14 = table2timetable(dataRSI);
    for f = 1:(nbStocks)
    RSI_14.Properties.VariableNames(f) =
append(string(RSI_14.Properties.VariableNames(f)),'RSI14');
    end
    T = [T RSI_14];
```

A new indicator, the MACD, is then generated and added to the Timetable. This is a Trend-following Momentum indicator, which comes from Moving Average Convergence Divergence. It allows to determine the relationship between two moving averages of different periods based on the price of a share. The MACD is the result of the subtraction between both moving averages. There are a great number of strategies based on the MACD, usually based on charts, in which in addition to the MACD line other averages and moving averages are plotted and depending on the crosses, the buy and sell signals are generated. In this case, only the numerical value of the MACD will be used. To calculate it, the MATLAB formula macd(), included in the Financial Toolbox, is used. The argument of this function is the price column of each share, which is obtained from the Timetable through the for loop shown below. The macd() function returns the MACD series, which is calculated by subtracting the 26-period exponential moving average from the 12-period moving average. This function also returns what is called SignalLine, which is a series with the 9-period exponential moving average, although it was not used in this work. The for loop also allows to create a column in the Timetable and add the calculated MACD, as shown below,

```
% Generate MACD
```

```
    for r = 1:nbStocks
        T.(append(string(T.Properties.VariableNames(r)),'MACD')) = 
macd(T.(r));
    end
```

Subsequently, the function is responsible for adding the data relating to the beta of each stock to the Timetable. Once again, a double for loop is used, similar to the one used for the different returns. The reason why a single for-loop is not used, that simply would allow to incorporate the data of the betas that have been previously loaded as it has been done with the RSI, is that it is wanted to calculate and to add the average beta of the last year. When observing the evolution of the beta of a stock with respect to time, it can be seen that its value changes significantly. As can be seen in Figure 30 which shows the evolution of BASF beta in the last 9 years.

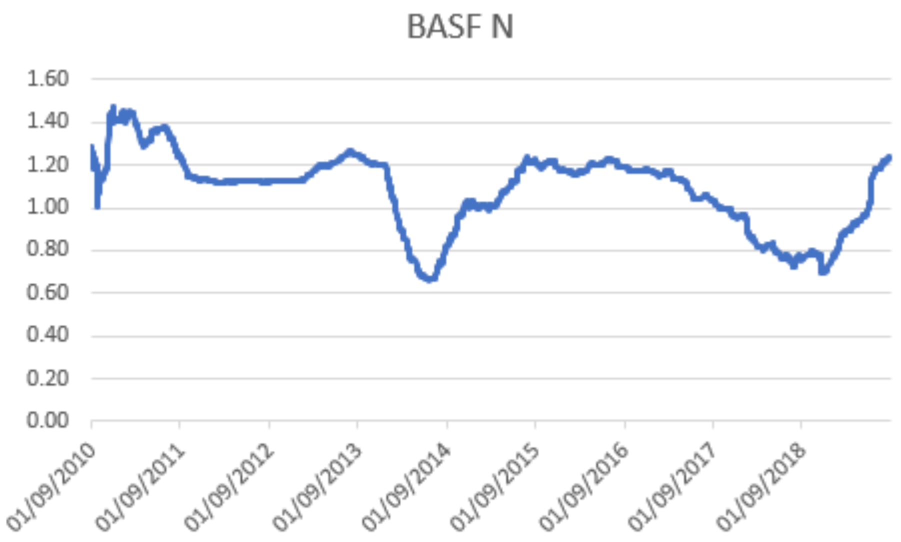


Figure 30: Evolution of the beta of BASF N for the last 9 years

It can therefore be seen that this indicator does not present the stability that one would like it to have in order to incorporate it into a robust investment strategy. A solution to this problem is found in the book “Los mercados financieros y sus matemáticas”, by Juan Pablo Jimeno Moreno in the chapter, “La inestabilidad de la beta. Formas “razonables” de inferir betas futuras a partir de las betas históricas” [23]. The Figure 30 shows, however, that despite the instability, the betas fluctuate between maximum and minimum values that are not very distant, the problem is to determine which is the value of the beta that best characterizes the stock, that is, what will be the future beta of the stock. If the different betas were added to the *Timetable* in the same way as the RSI, it would simply be the last beta available, which, although it is a widely used solution, does not seem to be the best solution for estimating the future beta. It seems more appropriate to use the data available regarding the evolution of the beta in recent sessions. In his book, Juan Pablo Jimeno Moreno calculates the beta of a stock at different terms and calculates the maximum, minimum and average values for each of the terms. The conclusion of his study is that in the short term, beta has a very important oscillation, from where it is concluded that any short term inference is really complicated. The second conclusion is that the longer the term, the smaller the oscillation, so with a long enough term, valuable information can be obtained for the future. The last and most important conclusion is that the average of each of the time periods is practically identical, so whatever the time period available, calculating the average gives practically the same value of beta. For these reasons it has been decided to calculate the annual average of the beta, which we remember has a time frame of 750 sessions, which is quite long (the longest time frame in the study carried out by Juan Pablo Jimeno Moreno is 180 sessions). In this way it is expected to obtain a stable value that enables to answer with the greatest possible confidence the question, what will be the company's beta in the future?

Therefore, the following double for loop is used, which will call the function averageBeta() created, for each stock and for each time period for the average wanted, although for the moment it is only calculated on an annual basis. Thanks to this structure, the columns are also created in the *Timetable* and the beta of each stock is added.

```
% Generate Beta
    nDay2 = [252];
    Beta = table2timetable(dataBeta);
    for k = 1:numel(nDay2)
        for g = 1:(nbStocks)
                factornamebeta = 
append(string(T.Properties.VariableNames(g)),'Beta');
                T.(factornamebeta) = averageBeta(Beta.(g), nDay2(k));
        end

    end
```

The avergaBeta() function is very simple, it collects as an argument the beta values loaded from the database and the number of sessions from which the average is to be obtained. Since what the user wants to obtain is a moving average, i.e. for each day calculate the average of the last 252 sessions, and as the function advances in time, the window moves accordingly, it uses the MATLAB function movmean(), which takes as an argument the vector from which the moving mean is to be obtained, in this case beta, the size of the window is entered, in this case 252, Next, the function is told to work in the first dimension of the beta vector and to calculate the average only when the window contains values of X, that is to say that it does not calculate the mean when it approaches the end and it only has partial values of the beta vector. For example, for the last 252 values the average is not calculated since it will not have 252 values, but less.

Supporting functions

```
function BetaAverage = averageBeta(Beta, nDay)
        % Compute the annual average of the beta of each stock
        BetaAverage = NaN(size(Beta, 1), 1);
        BetaAverage(nDay:end) = movmean(Beta, 252, 1, 
'Endpoints',"discard");
    end
```

This concludes the calculation of indicators or predictors that will be used as a basis for the investment strategies.

A response variable is then generated, which uses the next day's return to generate a Buy value, equal to 1, a Sell value equal to -1 and a Hold value equal to 0. Therefore, this variable allows to associate each row with indicators and prices to the return obtained on the following day. This is a variable that can be useful in the case of Machine Learning strategies in which the algorithm has associated a series of input values to their expected value for the output.

With the for-loop shown below, this variable is generated in such a way that the values from which the desired output is generated can be changed. The reason why it is not simply Buy > 0 and Sell < 0 is because to give it more robustness it is desired to add a margin of error in which the position is maintained or nothing is done, this margin of error is represented with these values and with the existence of the Hold variable. In this loop also the column is created in the Timetable and the response variable is added for each stock.

```
% Generate response variable (Y)
    % Use next day returs to generate "Buy (1)" if returns > 0 and
    % "Sell (-1) if returns < -0.002

     for a = 1:nbStocks

         Returns = T.(n+a);
         nextDayReturn = 1*(Returns(2:end) > 0) - 1*(Returns(2:end) < -
0.002);
         response = categorical(nextDayReturn,[1,-1,0],{'Buy', 'Sell',
'Hold'});
         T.(append(string(T.Properties.VariableNames(a)),'Response')) =
[response; missing];
     end
```

A similar step is then performed, but instead of using the next day's return as the expected output variable, the annual return up to that day is used. In a similar way to the previous one, the variable is composed by Buy that is equivalent to 1, Sell that is equivalent to -1 and Hold that is equivalent to 0. There is also here the notion of the margin of error whose values we can change to give more robustness to the strategy, and not buy directly when the annual return becomes positive and sell when it becomes negative, but give a time to the strategy. In this case another *Timetable* has been created, exactly the same as the previous one, in which this response variable is added so that it does not conflict with the previous one.

```
nDays = height(T);


        hasRet252 = ~cellfun('isempty',
regexp(T.Properties.VariableNames, 'Ret252', 'once'));
        Ret252 = T(:,T.Properties.VariableNames(hasRet252));
        Ret252 = table2array(Ret252);
        yearly_Return = Ret252;
        TML = T;


        for a = 1:nbStocks
           Returns = yearly_Return(:,a);
           Returns = 1*(Returns(255:end) > 0.025) - 1*(Returns(255:end)
< -0.02);
           response = categorical(Returns,[1,-1,0],{'Buy', 'Sell',
'Hold'});
           sizeR = size(response);
           sizeTML = height(TML);
           response(sizeR(1) + 1:sizeTML,:) = missing;

TML.(append(string(T.Properties.VariableNames(a)),'Response')) =
[response];
       end
```

Finally, the *Timetables* are divided into predictors and responses. To do this, the data with the daily prices of each share is stored in a different variable. All the rows that contain missing values are eliminated from the *Timetables*, the *Timetable* is transformed into a normal table and the column with the dates and prices is eliminated. Another table is also created which contains all the content calculated above except the betas, this table will be used for the strategies based on Deep Learning, which as will be explained does not require the beta to be contained in this table. Those steps are necessary only for the management of the tables by the different investment strategies. Below is the code that executes those steps.

```
% Split data into x (predictors) and y (response)

    pricesmean = zeros(height(T), nbStocks);
    for p = 1:nbStocks
        pricesmean(:,p) = T.(p);
    end
    T = rmmissing(T);
    T = timetable2table(T);
    nd = height(T);
    prices = zeros(nd,nbStocks);
    Days = T.time;
    T.time = []; % Remove time
    for b = 1:nbStocks
        prices(:,b) = T.(1);
        T.(1) = []; % Remove prices
    end

    TML = rmmissing(TML);
    TML = timetable2table(TML);
    TML.time = []; % Remove time
    for b2 = 1:nbStocks
        TML.(1) = []; % Remove prices
    end

    % We also remove beta as it won't be useful for the neural network
    T2 = T;

    hasbeta = ~cellfun('isempty', regexp(T2.Properties.VariableNames,
'Beta', 'once'));
    T2(:,T2.Properties.VariableNames(hasbeta)) = [];

    x = T2{:,1:end-nbStocks};
    y = T2{:,end-nbStocks+1:end};


    hasbeta = ~cellfun('isempty', regexp(TML.Properties.VariableNames,
'Beta', 'once'));
    TML(:,TML.Properties.VariableNames(hasbeta)) = [];


end
```

This concludes the pre-processing function. As said before, this function is called 16 times, for each of the indexes and for the shares that make up each of them.

Finally, all the information calculated is stored in the data2.mat matrix as shown below.

Save

```
clearvars c;
save('data2.mat')
```

The third of the scripts allows for the partitioning of the data into two time periods, called In Sample and Out of Sample, and for making final adjustments, since, as mentioned above, each strategy requires small differences in the shape of the data. First, as usual, both the Workspace and the Command Window are cleared and the preprocessing data, saved in the data2.mat matrix, is loaded. In this script it is also recommended to set a random seed using rng(0).

Data Partitioning

Each script required different forms of data. We need to create and partition data for each script.

In Sample: First 7 years

Out of Sample: Last 2 years

```
clear;
close all;
clc;
rng(0); % Fix random seed
load data2.mat
```

We call the function Partitioning with each of the timetables.

This script has a structure similar to the two previous ones, 2 functions have been created, Partitioning and Partitioning2 that are in charge of doing the partitioning firstly for the rules-based strategies and for the strategies based on machine learning. Secondly, the partition is made for the strategies based on Deep Learning.

CAC40

The first one wil divide the dataset into in Sample and out of Sample for the rule based strategy and the machine learning strategy

```
c = '0';


[CAC40_inSample, CAC40_trainML, CAC40_outSample, CAC40_yRetInSample,
CAC40_yRetOutSample, CAC40_pricesinSample, ...
    CAC40_pricesoutSample, CAC40_DaysinSample, CAC40_DaysoutSample] =
Partitioning(CAC40_Data, CAC40_DataML, CAC40_nStocks, CAC40_prices,
CAC40_Days, CAC40_x, CAC40_y, c);
```

The second one will do it for the deep learning strategies

```
[CAC40_xInSample, CAC40_xTrainFF, CAC40_Xin, CAC40_Xout,
CAC40_yInSample, CAC40_xOutSample, CAC40_yOutSample, ...
 CAC40_xTrain, CAC40_yTrain, CAC40_nFeatures, CAC40_yTrainFF,
CAC40_yTrainPredict]...
  = Partitioning2(CAC40_Data, CAC40_nStocks, CAC40_prices, CAC40_Days,
CAC40_x, CAC40_y);
```

DAX30

The first one wil divide the dataset into in Sample and out of Sample for the rule based strategy and the machine learning strategy

```
c = '0';


[DAX30_inSample, DAX30_trainML, DAX30_outSample, DAX30_yRetInSample,
DAX30_yRetOutSample, DAX30_pricesinSample, ...
    DAX30_pricesoutSample, DAX30_DaysinSample, DAX30_DaysoutSample] =
Partitioning(DAX30_Data, DAX30_DataML, DAX30_nStocks, DAX30_prices,
DAX30_Days, DAX30_x, DAX30_y, c);
```

The second one will do it for the deep learning strategies

```
[DAX30_xInSample, DAX30_xTrainFF, DAX30_Xin, DAX30_Xout,
DAX30_yInSample, DAX30_xOutSample, DAX30_yOutSample, ...
  DAX30_xTrain, DAX30_yTrain, DAX30_nFeatures, DAX30_yTrainFF,
DAX30_yTrainPredict]...
  = Partitioning2(DAX30_Data, DAX30_nStocks, DAX30_prices, DAX30_Days,
DAX30_x, DAX30_y);
```

For greater reliability, backtesting is not applied directly to the entire temporal universe but is divided into at least two parts. The first one, called In Sample, is the one that will be used to perform the training for the Machine Learning and Deep Learning techniques. For the strategies based on rules, for which there is no training, it would not be necessary to divide the data into In Sample and Out of Sample, since the algorithm does exactly the same in both periods, that is, follow the rule. However, it has been considered interesting because by dividing a relatively long period, 9 years, it allows to check visually faster the success of the strategy, both in the first 7 years and in the following 2.

The reason for the second period, the so-called Out of Sample, in the case of strategies where there is training, is that it serves to check that the strategy works correctly in a new set of data, and that no overfitting has occurred. In other words, an excessive adjustment of the algorithm to the data set in which the training has been performed. The problem with overfitting is that the resulting algorithm adapts very well to the initial data set, but once a new data set is introduced the algorithm is not able to adapt and very poor results are obtained. It should not be forgotten that the objective is to predict the return of new data sets, so obtaining an algorithm that works consistently in any data set is crucial. It is therefore better to get a robust algorithm that works well in as many environments as possible, than an algorithm that works extremely well only in a particular type of conditions but that works poorly when those conditions change.

As mentioned above, the total period for which data is available for the backtesting is 9 years, from 1 September 2010 to 30 August 2019. It has been decided to separate this set in a first universe of approximately 7 years, resulting in a total of about 1735 samples, we recall that on average a year has 252 trading days. This universe will be called In Sample. The second universe, Out of Sample, will be formed by the last two years, that is a total of 504 samples. In practice these two values can change from one index to another because not all of them are open the same days and it also depends on the number of missing values in each of the indexes, either because the market was closed or because of the calculations that have been made in the pre-processing of the data.

This partitioning is done as follows, in the Partitioning function.

```
function [inSample, trainingML, outSample, yRetInSample, yRetOutSample,
pricesinSample, pricesoutSample, DaysinSample, DaysoutSample] =
Partitioning(T, TML, nbStocks, prices, Days, x, y, c)
    % Setup partitioning parameters
    % Setup number of trading days for out of sample backtesting

    nOut = 252*2;

    % Define index for in-sample and out-of-sample
    N = height(T);
    nIn = N - nOut;
    idxIn = [1:nIn]';
    idxOut = [nIn + 1:N]';

    % Partition data for the Rules Based Strategy

    inSample = T(idxIn,:);
    outSample = T(idxOut,:);
    pricesinSample = prices(idxIn,:);
    pricesoutSample = prices(idxOut,:);
    DaysinSample = Days(idxIn,:);
    DaysoutSample = Days(idxOut,:);
        .
```

Then, in case the function is called with a Benchmark type data, that is, with the data about the behavior of the index itself and not about the shares that compose it, a resampling of the In Sample data set is performed, which will be used for the training of the Machine Learning algorithm. As it will be explained later, the Benchmark type data will be used for the training of such algorithms. The objective of this step is to obtain a balanced response variable. This is an important and delicate point in which there is no consensus about what should be done, and it is generally left to the developer of the algorithm to determine whether or not it is necessary to perform the resampling. There are certain studies and programmers who say that it is important that the algorithm is trained on a correctly balanced data set, that is, that all response variables have the same number of occurrence. In this case this would mean that the training data set has the same number of samples that require a response type Buy, Sell and Hold.

However, there are also studies that say it depends on the objective of the algorithm or even the data set and what it represents. What is clear is that whether or not it is balanced will produce a bias in the algorithm, the question is to determine whether it is better a bias caused by an unbalanced data set or a bias caused by an artificially balanced data set. It should also be noted that if the data are balanced, information about the frequency of occurrence of each of the variables is lost, which may be interesting information for the algorithm.

In this case, when trying the different benchmarks, the majority of signals are Hold signals. In addition, there are significantly more buy signals than sell signals. In the case of financial market time series, it is important to know that this frequency of occurrence is consistent with the behavior of financial markets. There is a bias towards the rise of financial markets because with time, processes are optimized, knowledge grows and technology improves, so that the productivity of companies increases, which in turn increases the profits of companies. This increase in profits and productivity is what drives the financial markets upwards, so it makes sense to have a greater number of buy signals than sell signals. The frequencies obtained therefore seem consistent with those that would be found in the total universe of financial market time series and do not seem to be due to the way in which the different samples have been selected.

In this work, the algorithms of the two forms have been tested, with a balanced training data set and an unbalanced one. The conclusion has been to use a balanced training data for the strategy based on a Machine Learning algorith. This was the procedure that provided the best results.

Below is the code that allows for balancing the data sets for Machine Learning based strategies.

```
% Partition data for the Machine Learning strategy

    trainingML = TML(idxIn,:);
    inSampleML = T(idxIn,:);
    outSampleML = T(idxOut,:);
    if c == 'Benchmark'

        hasResponse = ~cellfun('isempty', 
regexp(trainingML.Properties.VariableNames, 'Response', 'once'));
        nBuy = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse))) == 'Buy');
        nSell = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse))) == 'Sell');
        nHold = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse))) == 'Hold');

        nRemove = abs(nBuy-nSell);

        if nRemove > 0
            if nBuy > nSell
                idxRemove1 = 
find(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRe
sponse))) == 'Buy');
            elseif nSell > nBuy
                idxRemove1 = 
find(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRe
sponse))) == 'Sell');
            end
            idxRemove2 = datasample(idxRemove1,nRemove);
            trainingML(idxRemove2,:) = [];
        end

        hasResponse2 = ~cellfun('isempty', 
regexp(trainingML.Properties.VariableNames, 'Response', 'once'));
        nBuy2 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Buy');
        nRemove_Hold = abs(nHold - nBuy2);
        while nRemove_Hold > 0
            if nHold > nBuy2
               idxRemove_Hold1 = 
find(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRe
sponse2))) == 'Hold');
            elseif nBuy2 > nHold
                idxRemove_Hold1 = 
find(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRe
sponse2))) == 'Buy');
            end
            idxRemove_Hold2 = datasample(idxRemove_Hold1, 
nRemove_Hold);
            trainingML(idxRemove_Hold2,:) = [];
            nBuy3 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Buy');
```

```
            nHold3 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Hold');

            nRemove_Hold = abs(nHold3 - nBuy3);
        end

        nBuy4 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Buy');
        nSell4 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Sell');

        nRemove3 = abs(nBuy4-nSell4);

        while nRemove3 > 0
            if nBuy4 > nSell4
               idxRemove3 = 
find(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRe
sponse2))) == 'Buy');
            elseif nSell4 > nBuy4
                idxRemove3 = 
find(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRe
sponse2))) == 'Sell');
            end
            idxRemove4 = datasample(idxRemove3, nRemove3);
            trainingML(idxRemove4,:) = [];
            nBuy5 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Buy');
            nSell5 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Sell');

            nRemove3 = abs(nSell5 - nBuy5);
        end

        nBuy6 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Buy')
        nSell6 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Sell')
        nHold6 = 
sum(table2array(trainingML(:,trainingML.Properties.VariableNames(hasRes
ponse2))) == 'Hold')

end

end
```

The Partioning2 function follows the same procedure as the previous one but is used for strategies based on Deep Learning. Firstly, the total period is divided into the two sub-periods mentioned above in the same way as in the Partitioning function. Then there is the code for balancing the data. For strategies based on Deep Learning it has also been tested in both ways, with a balanced and an unbalanced data set. Finally, in this case, it has been chosen to use unbalanced data sets, since better data is obtained in backtesting both in the In Sample set and more importantly in the Out of Sample set. The reasons for this could be once again those mentioned above.

```
function [xInSample, xTrainFF, Xin, Xout, yInSample, xOutSample, 
yOutSample, ...
        xTrain, yTrain, nFeatures, yTrainFF, yTrainPredict] = 
Partitioning2(T, nbStocks, prices, Days, x, y)


% Partition data for Deep Learning Strategies

    nOut = 252*2;

    % Define index for in sample and out of sample
    N = height(T);
    nIn = N - nOut;
    idxIn = [1:nIn]';
    idxOut = [nIn + 1:N]';

    [nX,nFeatures] = size(x);
    xInSample = x(idxIn,:);
    yInSample = y(idxIn,:);
    xOutSample = x(idxOut,:);
    yOutSample = y(idxOut,:);
    nIn = size(xInSample,1);

    % For in-sample data, resample dataset to create a training data 
with balanced response variable.

% All the resample code is commented in this case

    xTrain = xInSample;
    yTrain = yInSample;

    nTrain = size(xTrain,1);
```

Finally, the training data set needs to be standardized. This is a necessary step for those strategies where there is some kind of neural network. What is done is standardize each of the predictors that are used to predict the outcome, i.e. each of the inputs to the neural network. Standardization is important, since the predictors have different units, some are returns, other temporary data etc. The different scales that these data have mean that they do not contribute in the same way during training, so there may be a bias towards those predictors with a scale that makes the values numerically higher. This bias has no financial basis, nor does it make any sense, so it is better to eliminate it. What it's done is subtract from each column its mean, a measure of location, and divide it by its standard deviation, a measure of scale. This step is done with the following code and using the stardizeByCol function that has been created for this purpose.

```
% Standardize training data

    [xTrain, muTrain, sigmaTrain] = standardizeByCol(xTrain);
    xInSample = (xInSample- repmat(muTrain,nIn,1)) ./
repmat(sigmaTrain,nIn,1);
    xOutSample = (xOutSample- repmat(muTrain,nOut,1)) ./
repmat(sigmaTrain,nOut,1);

    xTrainFF = xTrain;
    Xin = xInSample;
    Xout = xOutSample;

    %Convert from numerical array to cell format for LSTM.

    xTrain = mat2cell(xTrain', nFeatures, ones(nTrain,1))';
    xInSample = mat2cell(xInSample', nFeatures, ones(nIn,1))';
    xOutSample = mat2cell(xOutSample', nFeatures, ones(nOut,1))';

    % Partition data for Feed Forward Neural Network

    hasRet1 = ~cellfun('isempty', regexp(T.Properties.VariableNames,
'Ret1', 'once'));
    yTrainFF = T(2:nTrain+1,T.Properties.VariableNames(hasRet1));
    hasRet126 = ~cellfun('isempty',
regexp(yTrainFF.Properties.VariableNames, 'Ret126', 'once'));
    yTrainFF(:,yTrainFF.Properties.VariableNames(hasRet126)) = [];

    yTrainPredict1 = table2array(yTrainFF);
    yTrainPredict = mat2cell(yTrainPredict1', nbStocks,
ones(nTrain,1))';

end
```

Supporting Function

```
function [X, mu, sigma] = standardizeByCol(X)
% This function is used to standardize inputs by column

n = size(X,1);
mu = mean(X);
sigma = std(X);
X = (X - repmat(mu,n,1)) ./ repmat(sigma,n,1);
end
```

At the end, certain final adjustments are made so that the input and output values are appropriate and required for each case, the code is shown below

Once the two functions have been called for each required data set, the data is stored in 2 matrices, data3_demo1.mat and data3_demo3.mat.

This concludes the preparation of the data and the scripts for the different investment strategies can be made.

Before detailing the different investment strategies, it will be explained how the main script is structured. Although it is true that each strategy is different and has its particularities, which are reflected in some differences in the codes, a large part of the scripts are common.

The general scheme of the backtesting tool is as follows.

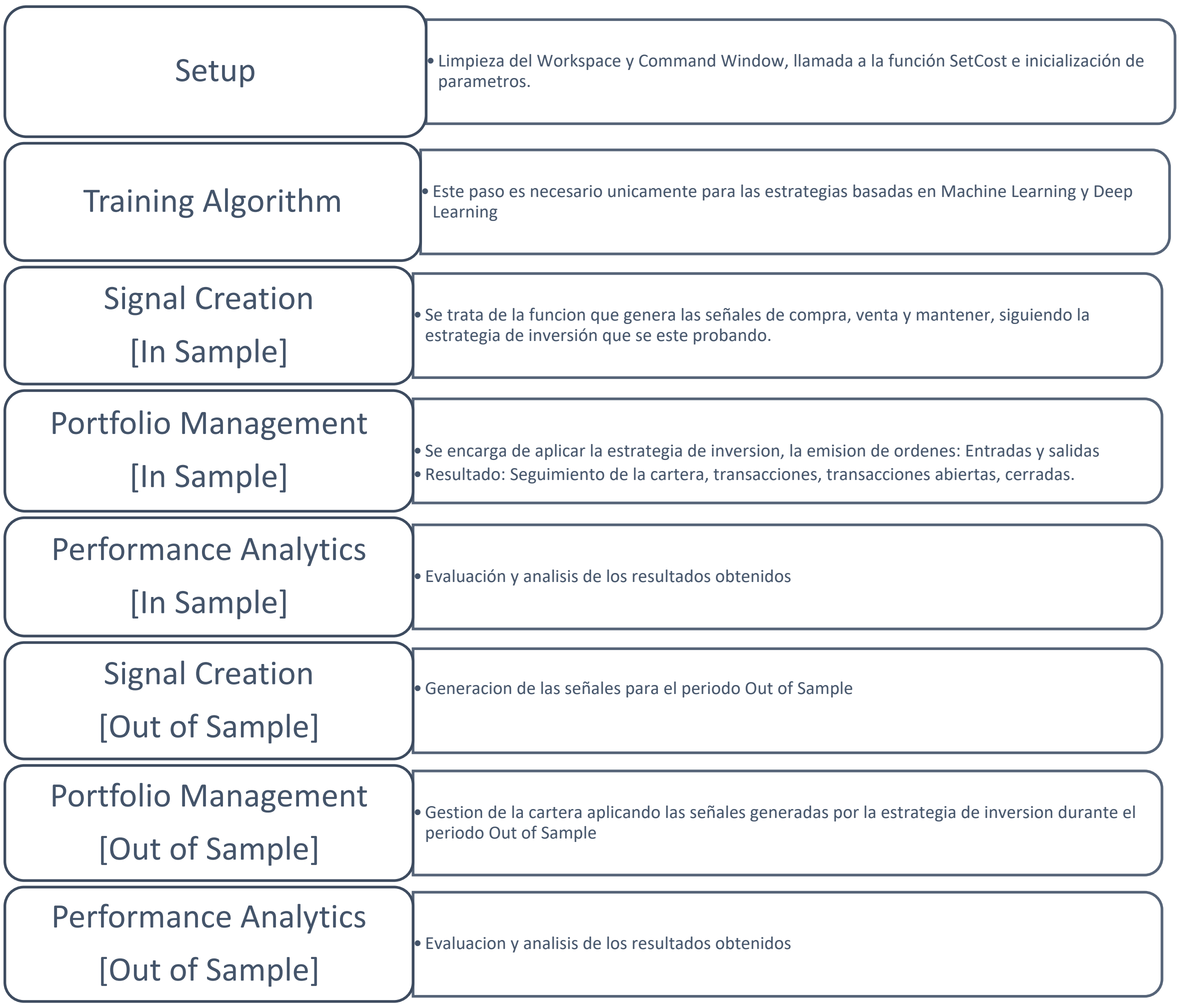

Setup
• Limpieza del Workspace y Command Window, llamada a la función SetCost e inicialización de parametros.
Training Algorithm
• Este paso es necesario unicamente para las estrategias basadas en Machine Learning y Deep Learning
Signal Creation [In Sample]
• Se trata de la funcion que genera las señales de compra, venta y mantener, siguiendo la estrategia de inversión que se este probando.
Portfolio Management [In Sample]
• Se encarga de aplicar la estrategia de inversion, la emision de ordenes: Entradas y salidas
• Resultado: Seguimiento de la cartera, transacciones, transacciones abiertas, cerradas.
Performance Analytics [In Sample]
• Evaluación y analisis de los resultados obtenidos
Signal Creation [Out of Sample]
• Generacion de las señales para el periodo Out of Sample
Portfolio Management [Out of Sample]
• Gestion de la cartera aplicando las señales generadas por la estrategia de inversion durante el periodo Out of Sample
Performance Analytics [Out of Sample]
• Evaluacion y analisis de los resultados obtenidos

Figure 31: General diagram of the Backtesting Tool

Once again, each script is structured in a modular way, where each module corresponds to an index. Each of the functions shown above are therefore called 8 times. The training function is called only in the case that it is a strategy based on a Machine Learning or Deep Learning algorithm. In this case the data relative to the benchmark index itself is passed as an argument since, as will be explained later when each strategy is detailed, only the benchmark index is used for training and not the data relative to the companies that compose it. The rest of the functions are also called 8 times, but in this case with the data relating to the companies that make up each index. The idea is as follows, the matrix is created with the Buy/Sell signals for the stocks that compose an index, for example the S&P 500. Previously training the algorithm if it is a Machine Learning or Deep Learning strategy. Then the Portfolio Management function is launched, only with the data from the S&P 500. Once the strategy has been simulated for the In Sample period, the results are analyzed with the Performance Analytics module, only with the data from the S&P 500. The entire process is then repeated by choosing another of the indices. By doing this separately and independently, both the development of the code, the simulation of the strategy and the analysis of the results are simpler, faster and more effective.

The training functions of the ML and DL algorithms and especially the creation of the signals represent the differential element of each strategy, so they will be seen in more detail when each one of them is presented.

The Setup, Portfolio Management and Performance Analytics parts and all the functions they contain are very similar for each strategy.

In the Setup part, first the clearing of the Workspace and the Command Window is performed, then the files necessary for the script to function correctly are loaded, this step changes slightly from one strategy to another as not all of them require the same data. The setCost function is then called and will be detailed below. Finally a certain number of variables are initialized that are needed in the script.

First the Setup used in the scripts of the strategies based on rules and in the strategies based on Machine Learning are shown, the only difference is the line of code that loads the matrix trainedML.mat, which is only present in the Setup of the strategy based on Machine Learning. Below the example of the Setup module of the Machine Learning Strategy

Machine Learning Strategy

The Classification Learner App is used to fit machine leanring algorithms on a balanced set of data. Then, the trained model is backtested during the In Sample and Out of Sample periods.

Setup

```
clear
clc
load data3_demo1.mat
load trainedML.mat
tDay = 252;
```

```
setCost;
Cash = [];
P =[];
T =[];
TOut =[];
TClose =[];
```

```
pnl=[];
```

Below is the code for the Setup module for the Deep Learning strategies. The only thing that changes are the loaded files and the fact that a random seed is set and the warning is disabled. This is necessary for the training of the neural networks.

FeedForward Neural Network Strategy

This script, will fit the deep learning algorithm, Feed-Forward Neural Network on the data for each financial market. The trained model will be backtested during In Sample and Out of Sample periods.

Setup

```
clear
clc
close all
rng(0);
warning off
load data3_demo3.mat
load data3_demo1.mat
tDay = 252;
```

```
setCost;
Cash = [];
P =[];
T =[];
TOut =[];
TClose =[];
pnl=[]
```

In both Setup modules, the setCost function has been called, this function is in charge of defining the initial cash, that is, the money with which the simulation starts at the beginning of each period, which is not invested and in this case is 10,000,000. It also initializes the risk-free rate, which is the theoretical value obtained by investing in a risk-free asset. It is generally obtained by subtracting inflation from the return provided by a bond from a solvent government with a duration similar to the investment time horizon, for example a Treasury bond. Historically the risk-free rate of return has been a positive value, however in this work a value equal to 0 is used as it is more representative of the current situation, despite the fact that inflation is low, so is the return provided by a risk-free asset such as the different sovereign bonds of historically solvent countries. In some countries such as Germany even providing negative returns. Finally, in this function the costs per transaction are fixed. While it is true that these costs do not stop falling due to the tremendous competition to which brokers are subjected, in the United States for American stocks and some financial instruments the cost is even zero, it is free to make transactions, in other countries like Spain there are still transaction costs. In this paper two types of transaction costs are considered, both of which are proportional to the volume of each transaction. The first of these, although not a real transaction cost, allows the simulation to behave in a more similar way to the real operative. In this one and practically any backtesting tool, it is considered that when a transaction is made on a particular day, it is completed at the closing price as it is impossible to know at what time of day it would have been made. Throughout the day the stock price fluctuates a bit so it is almost never really bought at the closing price, unless it is by chance. These intraday movements can be assumed to be random, so sometimes the investor would have bought at a higher price and sometimes at a lower price, with a frequency and magnitude that is considered random. Therefore, in approximately equal number of times, this fact should favor him and in others the opposite. However, since the results obtained in the backtest are generally better than real-time investing, it is a common practice to add a small expense called slippage to somehow represent this phenomenon. In this case the slippage is set at 5 basic points (bps), or what is the same 0.05% of the total value of the transaction. The other cost represents the cost that the broker would charge the investor for making the transaction and it is defined as 5 basic points. In this way we obtain a total cost of 10 basic points per transaction that allows the simulation to be close to the real time investing. Below is the code for the setCost function,

```
%% Set cost
% This script define the initial cash, risk-free rate, slipage, and
transaction cost

%% Set transaction costs
% The objective is to have an average of 10 bps of trading value, i.e. 0.1%
% Slipage is 0.05% of trading value
slipGlobal = 0.0005;

% Transaction is 0.05% of trading value
tCostGlobal = 0.0005;

%% Setup performance analytic variables
% initialCash is mainly used for performance analytics and
% determining the position size. Negative cash might occur due to
% unrealized gains which could lead to misleading returns and therefore
% Sharpe Ratios
initialCash = 10000000;
tDay = 252; %

% We will use 0 as risk free as currently it is more representative of the
% reality

riskFree = 0;
```

In the Setup part, a certain number of variables have been initialized, which are Cash, which monitor the cash available. P, a table that represents the portfolio, where the different stocks and some characteristics can be visualized. T, a table that represents all the transactions made and some of their characteristics. TOut, the transactions that remain open, outstanding transactions. TClose, the transactions that have already been closed. Finally, pnl, with the evolution of the gains and losses. These variables, plus an additional table called newPLTable, which includes the variable pnl and which shows the evolution of gains, losses and more characteristics are the outputs of the PortfolioManagement function. Its inputs are the matrix with the daily signals for buying, selling, holding, the matrix with the daily prices of each share, the column matrix containing the dates of all those daily prices, a matrix with the names of the shares, the number of shares, the number of days, the initial cash, a matrix with all the daily prices, not only those corresponding to the period in question. The difference with the other matrix lies in the fact that to calculate certain indicators, such as the annual return, since 252 values are needed, the first 251 days for which prices are available there is no annual return value. For those days, when eliminating the missing values from the Timetable, the prices are no longer available. Thanks to this matrix we can recover them. We also pass as an argument the initialised and empty variables P, T, TClose, TOut and pnl. The code of this function is explained below.

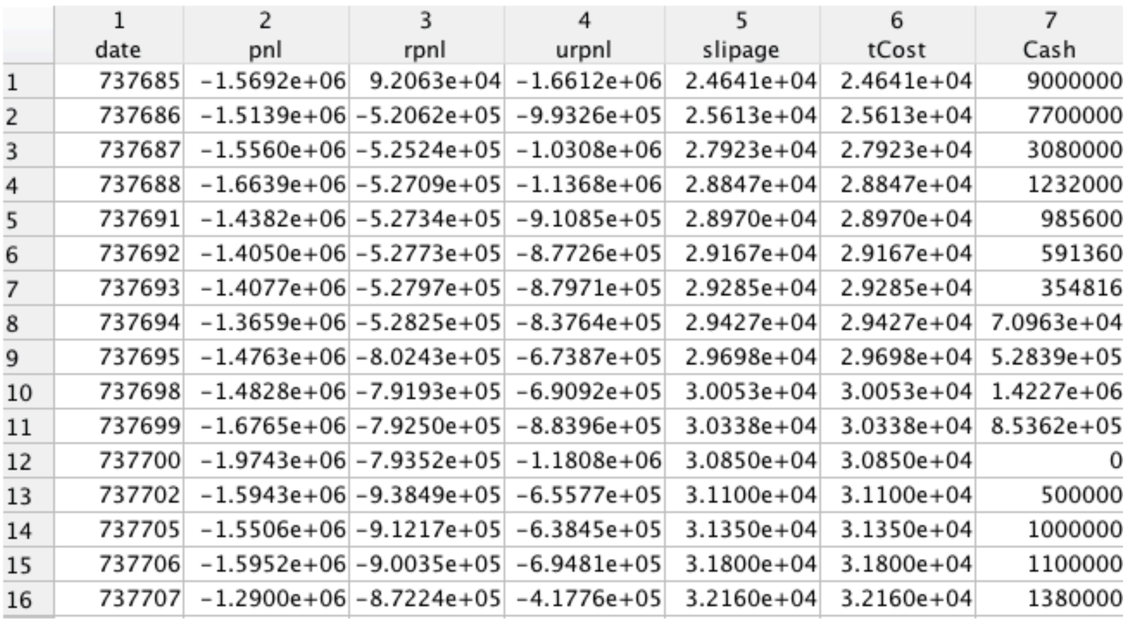
Portfolio Management

S&P 500

```
[P, T, TClose, TOut, newPLTable, pnl] =
PortfolioManagement(SP500_signalinSample, SP500_pricesinSample,
SP500_DaysinSample, namesSP, SP500_nStocks, SP500_nDay, initialCash,
SP500_prices_stats, P, T, TClose, TOut, pnl);
```

First, the variable newPLTable is created, it is a table with 7 columns where the number of rows corresponds to the number of days in the period, i. e. the number of days of the simulation. The columns correspond to the date, the pnl i.e. the total profits, rpnl that corresponds to the realized profits, that is the profits or losses obtained when selling a share that was in portfolio, urpnl, i.e. the unrealized profits. When having a certain amount of money invested in a share, its value changes over time, this value has not yet been translated into a profit or a loss, however, to be able to account for it, the variable urpnl is used. The pnl variable is the sum of the rpnl and urpnl variables. The next column is the slippage variable which shows the evolution of this cost. The next is tCost and shows the evolution of the second of the transaction costs. Finally, the last column, Cash, monitors the liquidity.

Below is an example of a newPLTable obtained,



	1 date	2 pnl	3 rpnl	4 urpnl	5 slipage	6 tCost	7 Cash
1	737685	-1.5692e+06	9.2063e+04	-1.6612e+06	2.4641e+04	2.4641e+04	9000000
2	737686	-1.5139e+06	-5.2062e+05	-9.9326e+05	2.5613e+04	2.5613e+04	7700000
3	737687	-1.5560e+06	-5.2524e+05	-1.0308e+06	2.7923e+04	2.7923e+04	3080000
4	737688	-1.6639e+06	-5.2709e+05	-1.1368e+06	2.8847e+04	2.8847e+04	1232000
5	737691	-1.4382e+06	-5.2734e+05	-9.1085e+05	2.8970e+04	2.8970e+04	985600
6	737692	-1.4050e+06	-5.2773e+05	-8.7726e+05	2.9167e+04	2.9167e+04	591360
7	737693	-1.4077e+06	-5.2797e+05	-8.7971e+05	2.9285e+04	2.9285e+04	354816
8	737694	-1.3659e+06	-5.2825e+05	-8.3764e+05	2.9427e+04	2.9427e+04	7.0963e+04
9	737695	-1.4763e+06	-8.0243e+05	-6.7387e+05	2.9698e+04	2.9698e+04	5.2839e+05
10	737698	-1.4828e+06	-7.9193e+05	-6.9092e+05	3.0053e+04	3.0053e+04	1.4227e+06
11	737699	-1.6765e+06	-7.9250e+05	-8.8396e+05	3.0338e+04	3.0338e+04	8.5362e+05
12	737700	-1.9743e+06	-7.9352e+05	-1.1808e+06	3.0850e+04	3.0850e+04	0
13	737702	-1.5943e+06	-9.3849e+05	-6.5577e+05	3.1100e+04	3.1100e+04	500000
14	737705	-1.5506e+06	-9.1217e+05	-6.3845e+05	3.1350e+04	3.1350e+04	1000000
15	737706	-1.5952e+06	-9.0035e+05	-6.9481e+05	3.1800e+04	3.1800e+04	1100000
16	737707	-1.2900e+06	-8.7224e+05	-4.1776e+05	3.2160e+04	3.2160e+04	1380000

Figure 32: Example of a variable newPLTable

Portfolio Management

Once we have the matrix containing the trade signals, 1 = Buy, -1 = Sell and 0 = Hold, we can create and manage the portfolio as well as the entries and exits of the assets. This function also maintains a record of the portfolio, the transactions and the evolution of the P&L of the portfolio.

```
function [P, T, TClose, TOut, newPLTable, pnl] =
PortfolioManagement(signal1, prices, Days, names, nStocks, nDay,
initialCash, prices_stats, P, T, TClose, TOut, pnl)


    newPLTable = array2table(zeros(nDay,7),'VariableNames',...
        {'date','pnl','rpnl','urpnl','slipage','tCost', 'Cash'});

    Cash = initialCash;
    m = 0;
    f = 0;
```

Then a for loop is opened which will allow to go through the Buy/Sell signal matrix, in this case in the dimension of the rows. In other words, the for-loop allows to advance from day 1 of the simulation to the last day. This for-loop remains open until the end of the function. Inside it, firstly, it initializes the size of the buy and sell positions. That is, the size in the currency of the index that each buy and sell position will have. These values can change from one strategy to another as will be seen later although generally it is around 0.15 times the cash available for purchases and 0.05 times the initial cash for sales. The initial cash for sales is used so that the sales are not too small if at any time the cash available is too little. In the case of purchases, it must necessarily be the cash that is available, otherwise it would be necessary to borrow money, which is not contemplated in this work. The position is not sold in its entirety in the case of a sale position, nor is all or most of the money invested in the case of a purchase position. The reason is that since this is a systematic strategy, without any human control behind each operation, the idea is that small or relatively small investments are made every time there is a signal. Therefore, if it is a false signal, the investment or the sale are not very important. On the other hand, if the signal is repeated several days in a row, it is confirmed that it was a valid signal, and eventually a strong position is built up in the portfolio, or, if it was a sell signal, the position is sold.

Next, the variables that will store the lists with the shares to buy and sell each day, the amount to buy and sell and their price are initialized. These variables are respectively BuyList, SellList, BuyQty, SellQty, BuyPrice and SellPrice.

An if structure is then introduced. It only allows to cancel the sizes of the buy and sell positions, this is done for investment strategies where it is necessary to optimize the positions in the portfolio. This procedure will be discussed later when the portfolio optimization part is explained.

```
for i = 1:nDay

        Buy_posSize = 0.15*Cash;
        Sell_posSize = 0.05*initialCash;
        BuyList = cell(nStocks,1);
        SellList = cell(nStocks,1);
        BuyQty = cell(nStocks,1);
        SellQty = cell(nStocks,1);
        BuyPrice = cell(nStocks,1);
        SellPrice = cell(nStocks,1);

        if i == 20*m+1
            Buy_posSize = 0;
            Sell_posSize = 0;
        end
```

After that, a second for-loop is opened, which allows to go through the signal matrix but in this case horizontally. Starting with the first stock until the last one. The function analyzes for each day, for each stock, which is the signal and thus building the lists with the shares to buy and sell. The procedure is simple, first it analyzes if the for the day and stock in which the loop is, the signal is 1 and if there is cash to make a purchase. If this is the case, it proceeds by adding this share to the BuyList variable. It also adds its price to the BuyPrice variable as well as the number of shares to buy. This amount is equivalent to the value in the corresponding currency divided by the price of the share. This result is added to the BuyQty variable. The code analyzes, in a similar way, if there are sell signals, in that case by looking if the signal is a -1 and also if the stock that is being analyzed is already in the portfolio or not. If there is a sale signal and the asset is already in the portfolio, the sale process is continued by adding the asset to the different variables. If it is not in the portfolio it is not sold. This is done to avoid short sales, i.e. selling shares that are not in the portfolio, borrowing from the broker, and buying them later. In this way, the investor can take advantage not only of price increases but also of price decreases. However, this is a particularly risky operation since when borrowing the shares, it is necessary to pay interest to the broker. To be invested in short therefore costs money, the more time the investor has a short position, the greater the expense. Not only he needs to be right about the direction in which the share price will vary, but also about the timing. This added difficulty is the reason why in this work short selling is not allowed. However, by simply removing this restriction, i.e. this line of code, the backtest tool allows to test strategies that invest in short. A transaction cost would have to be added to represent the interest to be paid.

```
for j = 1:nStocks
            caseBuy = 0;
            caseSell = 0;
            if signal1(i,j) == 1 && Cash ~= 0
                caseBuy = 1;
                BuyList{j} = names(j);
            end
            if signal1(i,j) == -1
                % We first check if this asset is in the portfolio, if
it isn't we didn't sell short
                ch_P = isempty(P);
                if ch_P == 0
                    Assets = P.name';
                    isinPF = strcmp(upper(names(j)),string(Assets));
                    if any(isinPF == 1)
```

```
                        caseSell = 1;
                        SellList{j} = names(j);
                    end
                end
            end
            % Set buying/selling prices if there is any action

            if caseBuy == 1
                Exprice = prices(i,j);
                BuyPrice{j} = Exprice;
                BuyQty{j} = Buy_posSize/BuyPrice{j};
            end

            if caseSell == 1
                Exprice = prices(i,j);
                SellPrice{j} = Exprice;
                SellQty{j} = Sell_posSize/SellPrice{j};
            end

        end
```

The empty rows and columns of the variables containing the shares to be bought and sold are then removed. This step is necessary since the variables have been initialized by allocating a larger space than the one that is finally needed. The previous allocation is done because it allows to optimize the code and make it go faster.

```
% Delete empty row
        delBuy = false(nStocks,1);
        for b = 1:nStocks
            if isempty(BuyList{b}) == true
                delBuy(b) = true;
            end
        end

        delSell = false(nStocks,1);
        for c = 1:nStocks
            if isempty(SellList{c}) == true
                delSell(c) = true;
            end
        end

        if max(delBuy) == 1
            BuyList(delBuy) = [];
            BuyQty(delBuy) = [];
            BuyPrice(delBuy) = [];
        end

        if max(delSell) == 1
            SellList(delSell) = [];
            SellQty(delSell) = [];
            SellPrice(delSell) = [];
        end
```

Then, once the final lists are available with the shares to be bought and sold, their prices and quantities for that day. The function submitOrder is called. This function will manage the execution of the orders and incorporate them into the variables that represent the portfolio, the transactions, both open and closed and total. This function will be explained in detail later, when the PortfolioManagement() function is finished.

The option is given in the code to define the frequency with which this function is to be called. If it is called daily, every day the orders will be submitted, if the frequency is lower, for example every 21 days, what is achieved is that the portfolio is rebalanced on a monthly basis which is a very common practice in systematic portfolio management as it allows to reduce the turnover and transaction costs as well as to give a time to the shares that are in the portfolio for their price to rise. It is a way to increase the time horizon as it limits rotations. This will be explained in more detail when analyzing the different investment strategies.

```
% We submit the orders only once a month if it is 20.
        if i == 20*f+1
            % Submit the order
            [P, T, TOut, TClose] = submitOrder(prices, P, T, TOut, 
TClose, Days(i), BuyList, SellList, BuyPrice, SellPrice, BuyQty, 
SellQty);

            [nb_Buy, ~] = size(BuyList);
            [nb_Sell, ~] = size(SellList);

            Cash = Cash - Buy_posSize*nb_Buy + Sell_posSize*nb_Sell;

            if Cash < 0
                Cash = 0;
            end

            f = f + 1;
        end
```

Once the orders have been issued and the portfolio is updated, the remaining cash is updated. Simply by taking into account how much has been bought and how much has been sold. In the event that the cash is negative, it is set to 0 so that no further purchases are made.

Next, the information regarding the gains, losses and expenses associated with the transactions is calculated and stored. To do this the function calcPL is called. This function will be seen in detail later on when the explanation of the PortfolioManagement function is finished. It returns the updated pnl, rpnl, urpnl, slippage and tCost variables. Once these variables are updated they are added to the newPLTable, in a new line. Remembering that each line represents one day of the simulation, so it is easy to analyze the evolution of these elements of the simulation.

```
% Calculate and store P&L
        [pnl,rpnl,urpnl,slipage,tCost] = calcPL(prices,Days(i), TOut, 
TClose, names, i);
```

```
        date = datenum(Days(i));
        newPLTable(i,:) = table(date,pnl, rpnl, urpnl, slipage, tCost,
Cash);
```

Then comes the portion of code that manages the portfolio optimization discussed above. All this part is included within an if structure that determine if the optimisation is to be done, in a similar way to the code that allowed to set the sizes of the buy and sell positions to 0, which is a necessary step for the optimisation of the portfolio. It is necessary because the optimization consists precisely in determining the optimal size of the buy and sell of each stock. The initial if structure not only allows to establish if the optimization is done or not, but also its frequency, daily, monthly, quarterly etc.

This optimization is done using the PortfolioOptimization function, but before calling this function it is necessary to obtain certain information that will be passed to the function as arguments. Firstly, the list of assets in the portfolio which is stored in the AssetList variable. Then the monetary amount of each asset in the portfolio. To do this, the number of shares for each asset is multiplied by the current price of each share. To determine the current price, the GetPrice function has been created, which will be studied in detail later on. This creates the variable AssetQty, which stores, in the same order as in AssetList, the amounts in monetary value available in our portfolio for each share. Then the variable TotalQty is calculated, which is the sum of all the elements of AssetQty and represents the total value of the portfolio. With this information the variable InitPort is created, in which the weight of each share relative to the total value of the portfolio is stored in the same order as AssetList. In the case in which TotalQty is 0, that is to say that the portfolio has not yet been built, a value of 80% of the available cash is associated with it, in this way the initial optimized portfolio will have a total value of 99% of the initial capital.

Once this information is available, the PortfolioOptimization function is called. This function returns the updated BuyList, SellList, BuyQty, SellQty, BuyPrice, SellPrice, Cash variables with the list of shares, their price and especially the optimized amount to buy or sell to obtain a portfolio that is as close as possible to the optimization parameters that have been defined in the function and that will be detailed later.

The DeleteOrders function is then called. This is a necessary step to remove any 0-size orders that have been made before the optimization but that are not actually made and therefore are not desired to be reflected either in the portfolio or in the transaction records.

Once the buy and sell lists are available and the 0-size orders have been deleted, the submitOrder function can be called. Which as explained before is in charge of updating the portfolio and the transaction records from the list of stocks to be bought and sold, this time with position sizes different from 0 and determined by an optimization algorithm.

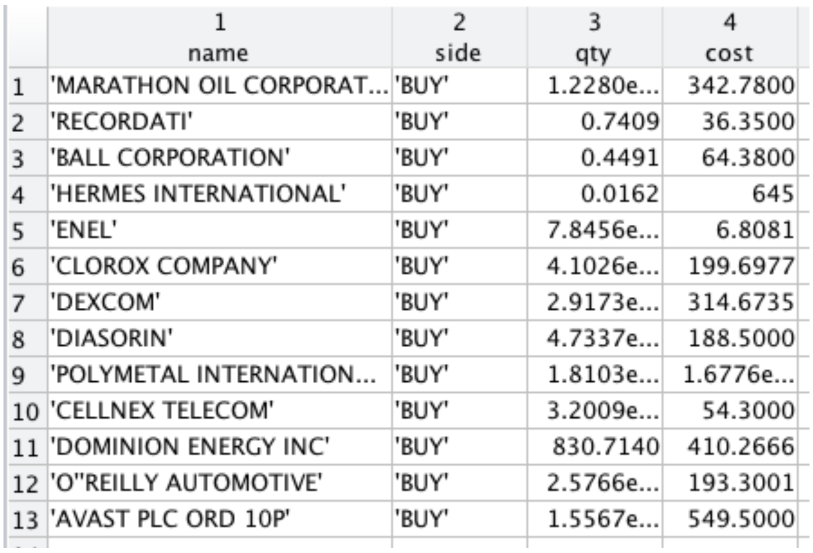
Thereafter, with the portfolio already updated, following the same process and order as described above, the gains, losses and expenses associated with the operation are calculated by using the CalcPL function. The newPLTable is then updated with the variables date, pnl, rpnl. Urpnl, slipage, tCost, Cash recently updated.

Below is the code that manages the call of the optimization function, of the delete orders function, to submit the orders and to finally compute the evolution of the P&L of the portfolio,

```
if i == 20*m+1
            checkP = isempty(P);
            if checkP == 0
                AssetList = P.name';
                [~, nAssets] = size(AssetList);
                actual_prices = zeros(nAssets, 1);
                for k = 1:nAssets
                    actual_prices(k) = getPrice(prices, AssetList{k},
names, i);
                end
                AssetQty = P.qty.*actual_prices;
                TotalQty = sum(AssetQty);
                InitPort = AssetQty./TotalQty;
                ind = find(strcmp(P.side, 'SELL'));
                InitPort(ind,:) = -InitPort(ind,:);

                if TotalQty == 0
                    InitPort(isnan(InitPort)) = 0;
                    [numAssets, ~] = size(InitPort);
                    %TotalQty = 0.8*Cash*numAssets;
                    TotalQty = 0.99*Cash;
                end

                [BuyList, SellList, BuyQty, SellQty, BuyPrice,
SellPrice, Cash] = PortfolioOptimization(AssetList, InitPort, prices,
prices_stats, names, i, Cash, TotalQty);

                [P, T, TOut, TClose] = DeleteOrders(P, T, TOut,
TClose);

                [P, T, TOut, TClose] = submitOrder(prices, P, T, TOut,
TClose, Days(i), BuyList, SellList, BuyPrice, SellPrice, BuyQty,
SellQty);

                % Calculate and store P&L
                [pnl,rpnl,urpnl,slipage,tCost] = calcPL(prices,Days(i),
TOut, TClose, names, i);
                date = datenum(Days(i));
                newPLTable(i,:) = table(date,pnl, rpnl, urpnl, slipage,
tCost, Cash);
            end
            m = m + 1;

        end
```

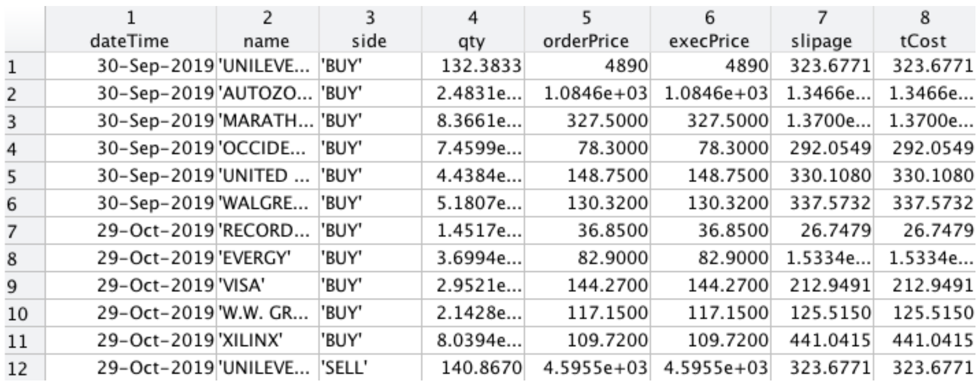
At this point, the initial for loop that allowed to go through the days of the simulation comes to an end, thus increasing by one the value of the days and repeating this whole process for the next day. Once all this management has been done for all the period of the simulation, the function is finished, returning as mentioned before the final portfolio, P, whose table shows the name of the shares, their side, i.e. if the initial operation was a purchase or a sale. In the case in which short sales are not allowed, they will always be purchases. The table also includes the number of shares of each stock and the average cost per share, in case the whole position was not bought the same day but built up over several days. The PortfolioManagement function also returns a record of all the transactions, T, in this table are shown the date of each transaction, the name of the stock, the side of the transaction, i.e. buy or sell, the number of shares that have been bought or sold, the order and execution price which in this work is the same. It could be considered that the execution price is the order price plus the slippage although in the whole project it has been preferred to show it separately. It also shows us the slippage of the operation and the transaction cost paid to the broker. Both costs, taking into account the volume of the operation, showing the total cost that had to be paid. The variable with the transactions still open, TOut, whose columns are the same as for table T. The function also returns the already closed trades, TClose, which in addition to the columns already mentioned in T, also includes a column with the date of closing the trade, the amount of shares that have been sold and the profit or loss, Capital Gain, that has been made by closing the trade.

Below are shown some examples of the variables P, T, TOut and TClose respectively,



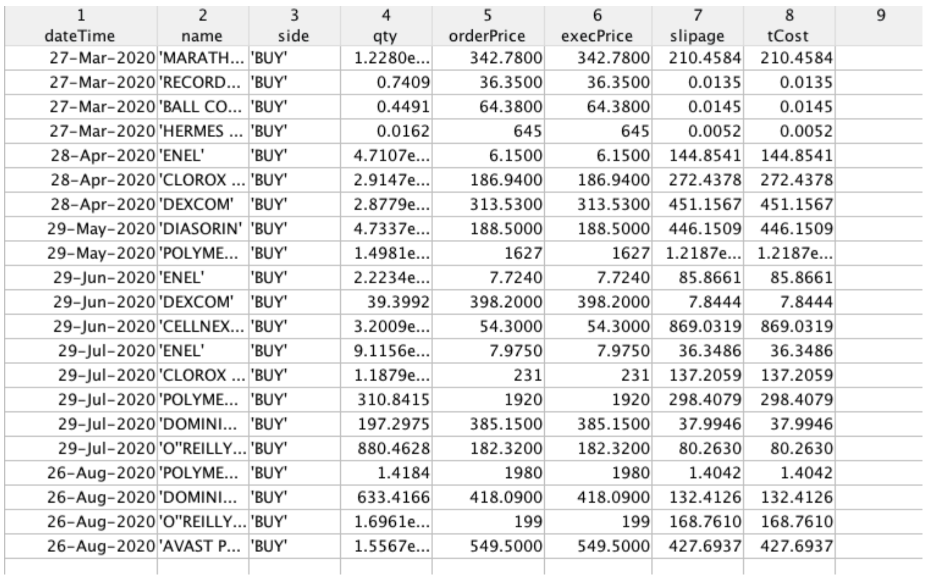
	1 name	2 side	3 qty	4 cost
1	'MARATHON OIL CORPORAT...	'BUY'	1.2280e...	342.7800
2	'RECORDATI'	'BUY'	0.7409	36.3500
3	'BALL CORPORATION'	'BUY'	0.4491	64.3800
4	'HERMES INTERNATIONAL'	'BUY'	0.0162	645
5	'ENEL'	'BUY'	7.8456e...	6.8081
6	'CLOROX COMPANY'	'BUY'	4.1026e...	199.6977
7	'DEXCOM'	'BUY'	2.9173e...	314.6735
8	'DIASORIN'	'BUY'	4.7337e...	188.5000
9	'POLYMETAL INTERNATION...	'BUY'	1.8103e...	1.6776e...
10	'CELLNEX TELECOM'	'BUY'	3.2009e...	54.3000
11	'DOMINION ENERGY INC'	'BUY'	830.7140	410.2666
12	'O''REILLY AUTOMOTIVE'	'BUY'	2.5766e...	193.3001
13	'AVAST PLC ORD 10P'	'BUY'	1.5567e...	549.5000

Figure 33: Example of a P variable obtained



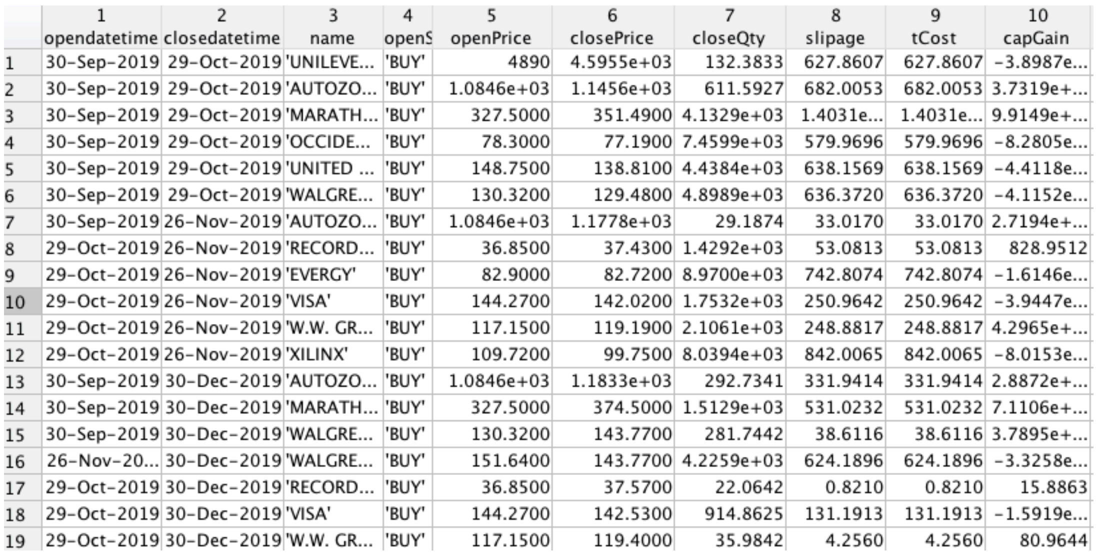
	1 dateTime	2 name	3 side	4 qty	5 orderPrice	6 execPrice	7 slipage	8 tCost
1	30-Sep-2019	'UNILEVE...	'BUY'	132.3833	4890	4890	323.6771	323.6771
2	30-Sep-2019	'AUTOZO...	'BUY'	2.4831e...	1.0846e+03	1.0846e+03	1.3466e...	1.3466e...
3	30-Sep-2019	'MARATH...	'BUY'	8.3661e...	327.5000	327.5000	1.3700e...	1.3700e...
4	30-Sep-2019	'OCCIDE...	'BUY'	7.4599e...	78.3000	78.3000	292.0549	292.0549
5	30-Sep-2019	'UNITED ...	'BUY'	4.4384e...	148.7500	148.7500	330.1080	330.1080
6	30-Sep-2019	'WALGRE...	'BUY'	5.1807e...	130.3200	130.3200	337.5732	337.5732
7	29-Oct-2019	'RECORD...	'BUY'	1.4517e...	36.8500	36.8500	26.7479	26.7479
8	29-Oct-2019	'EVERGY'	'BUY'	3.6994e...	82.9000	82.9000	1.5334e...	1.5334e...
9	29-Oct-2019	'VISA'	'BUY'	2.9521e...	144.2700	144.2700	212.9491	212.9491
10	29-Oct-2019	'W.W. GR...	'BUY'	2.1428e...	117.1500	117.1500	125.5150	125.5150
11	29-Oct-2019	'XILINX'	'BUY'	8.0394e...	109.7200	109.7200	441.0415	441.0415
12	29-Oct-2019	'UNILEVE...	'SELL'	140.8670	4.5955e+03	4.5955e+03	323.6771	323.6771

Figure 34: Example of a T variable obtained



1 dateTime	2 name	3 side	4 qty	5 orderPrice	6 execPrice	7 slipage	8 tCost	9
27-Mar-2020	'MARATH...	'BUY'	1.2280e...	342.7800	342.7800	210.4584	210.4584	
27-Mar-2020	'RECORD...	'BUY'	0.7409	36.3500	36.3500	0.0135	0.0135	
27-Mar-2020	'BALL CO...	'BUY'	0.4491	64.3800	64.3800	0.0145	0.0145	
27-Mar-2020	'HERMES ...	'BUY'	0.0162	645	645	0.0052	0.0052	
28-Apr-2020	'ENEL'	'BUY'	4.7107e...	6.1500	6.1500	144.8541	144.8541	
28-Apr-2020	'CLOROX ...	'BUY'	2.9147e...	186.9400	186.9400	272.4378	272.4378	
28-Apr-2020	'DEXCOM'	'BUY'	2.8779e...	313.5300	313.5300	451.1567	451.1567	
29-May-2020	'DIASORIN'	'BUY'	4.7337e...	188.5000	188.5000	446.1509	446.1509	
29-May-2020	'POLYME...	'BUY'	1.4981e...	1627	1627	1.2187e...	1.2187e...	
29-Jun-2020	'ENEL'	'BUY'	2.2234e...	7.7240	7.7240	85.8661	85.8661	
29-Jun-2020	'DEXCOM'	'BUY'	39.3992	398.2000	398.2000	7.8444	7.8444	
29-Jun-2020	'CELLNEX...	'BUY'	3.2009e...	54.3000	54.3000	869.0319	869.0319	
29-Jul-2020	'ENEL'	'BUY'	9.1156e...	7.9750	7.9750	36.3486	36.3486	
29-Jul-2020	'CLOROX ...	'BUY'	1.1879e...	231	231	137.2059	137.2059	
29-Jul-2020	'POLYME...	'BUY'	310.8415	1920	1920	298.4079	298.4079	
29-Jul-2020	'DOMINI...	'BUY'	197.2975	385.1500	385.1500	37.9946	37.9946	
29-Jul-2020	'O''REILLY...	'BUY'	880.4628	182.3200	182.3200	80.2630	80.2630	
26-Aug-2020	'POLYME...	'BUY'	1.4184	1980	1980	1.4042	1.4042	
26-Aug-2020	'DOMINI...	'BUY'	633.4166	418.0900	418.0900	132.4126	132.4126	
26-Aug-2020	'O''REILLY...	'BUY'	1.6961e...	199	199	168.7610	168.7610	
26-Aug-2020	'AVAST P...	'BUY'	1.5567e...	549.5000	549.5000	427.6937	427.6937	

Figure 35: Example of a TOut variable obtained



	1 opendatetime	2 closedatetime	3 name	4 openS	5 openPrice	6 closePrice	7 closeQty	8 slipage	9 tCost	10 capGain
1	30-Sep-2019	29-Oct-2019	'UNILEVE...	'BUY'	4890	4.5955e+03	132.3833	627.8607	627.8607	-3.8987e...
2	30-Sep-2019	29-Oct-2019	'AUTOZO...	'BUY'	1.0846e+03	1.1456e+03	611.5927	682.0053	682.0053	3.7319e+...
3	30-Sep-2019	29-Oct-2019	'MARATH...	'BUY'	327.5000	351.4900	4.1329e+03	1.4031e...	1.4031e...	9.9149e+...
4	30-Sep-2019	29-Oct-2019	'OCCIDE...	'BUY'	78.3000	77.1900	7.4599e+03	579.9696	579.9696	-8.2805e...
5	30-Sep-2019	29-Oct-2019	'UNITED ...	'BUY'	148.7500	138.8100	4.4384e+03	638.1569	638.1569	-4.4118e...
6	30-Sep-2019	29-Oct-2019	'WALGRE...	'BUY'	130.3200	129.4800	4.8989e+03	636.3720	636.3720	-4.1152e...
7	30-Sep-2019	26-Nov-2019	'AUTOZO...	'BUY'	1.0846e+03	1.1778e+03	29.1874	33.0170	33.0170	2.7194e+...
8	29-Oct-2019	26-Nov-2019	'RECORD...	'BUY'	36.8500	37.4300	1.4292e+03	53.0813	53.0813	828.9512
9	29-Oct-2019	26-Nov-2019	'EVERGY'	'BUY'	82.9000	82.7200	8.9700e+03	742.8074	742.8074	-1.6146e...
10	29-Oct-2019	26-Nov-2019	'VISA'	'BUY'	144.2700	142.0200	1.7532e+03	250.9642	250.9642	-3.9447e...
11	29-Oct-2019	26-Nov-2019	'W.W. GR...	'BUY'	117.1500	119.1900	2.1061e+03	248.8817	248.8817	4.2965e+...
12	29-Oct-2019	26-Nov-2019	'XILINX'	'BUY'	109.7200	99.7500	8.0394e+03	842.0065	842.0065	-8.0153e...
13	30-Sep-2019	30-Dec-2019	'AUTOZO...	'BUY'	1.0846e+03	1.1833e+03	292.7341	331.9414	331.9414	2.8872e+...
14	30-Sep-2019	30-Dec-2019	'MARATH...	'BUY'	327.5000	374.5000	1.5129e+03	531.0232	531.0232	7.1106e+...
15	30-Sep-2019	30-Dec-2019	'WALGRE...	'BUY'	130.3200	143.7700	281.7442	38.6116	38.6116	3.7895e+...
16	26-Nov-20...	30-Dec-2019	'WALGRE...	'BUY'	151.6400	143.7700	4.2259e+03	624.1896	624.1896	-3.3258e...
17	29-Oct-2019	30-Dec-2019	'RECORD...	'BUY'	36.8500	37.5700	22.0642	0.8210	0.8210	15.8863
18	29-Oct-2019	30-Dec-2019	'VISA'	'BUY'	144.2700	142.5300	914.8625	131.1913	131.1913	-1.5919e...
19	29-Oct-2019	30-Dec-2019	'W.W. GR...	'BUY'	117.1500	119.4000	35.9842	4.2560	4.2560	80.9644

Figure 36: Example of a TClose variable obtained

In the following, the different functions included in the PortfolioManagement function will be studied in detail.

Firstly the function submitOrder is explained, its entries are the matrix with the daily prices of each share, the portfolio P, the list of transactions T, the list of outstanding transactions TOut, the list of closed transactions TClose, the current date, Days(i), the list with the shares to buy BuyList, the list of shares to sell SellList, the list of prices to buy BuyPrice, the list of prices to sell SellPrice, the list of number of shares to buy of each BuyQty share and the list of number of shares to sell of each SellQty share. The function returns updated P, T, TOut and TClose.

The first step is to call the setCost function explained above since it is necessary to have defined in the Workspace the values that are defined in setCost for the slippage and tCost.

The second step is to submit the purchase orders. This creates a for-loop that goes through the list of shares to be purchased. Inside the loop, two functions are called. The first one, updateTNew, updates the T and TNew lists, which represents a table with the new transactions to be done, with the buy orders as we will see later. Once both lists are updated the second function is called, updatePort which allows to update the portfolio P and the TOut and TClose lists based on the content of the TNew table. The for loop is shown below,

```
function [P,T,TOut,TClose] = submitOrder(prices,P,T,TOut,TClose, days,...
    BuyList,SellList,BuyPrice,SellPrice,bQty,sQty)
% submitOrder will update the portfolio and transaction lists based on
% the buy and sell lists

%% Inititalize transaction cost
setCost;
%%  Submit buy orders
if isempty(BuyList) == false % If buying list is not empty
    for i = 1:length(BuyList)
        % Put buy order in T (All Transaction) and TNew (New Transaction)
        TNew = updateTNew(prices, days,
BuyList{i},'Buy',bQty{i},BuyPrice{i},...
            slipGlobal,tCostGlobal);
        % Update Portfolio, TOut (Outstanding Transaction), and TClose
        % (Closed Transaction) based on the TNew
        [P,T,TOut,TClose] = updatePort(P,T,TOut,TClose,TNew);
    end
end
```

With the buy orders already submitted, the procedure is similar with the sell orders with the next loop.

```
%%  Submit sell orders
if isempty(SellList) == false % If selling list is not empty
    for i = 1:length(SellList)
        % Put sell order in T (All Transaction) and TNew (New Transaction)
        TNew = updateTNew(prices, days,
SellList{i},'Sell',sQty{i},SellPrice{i},...
            slipGlobal,tCostGlobal);
        % Update Portfolio, TOut (Outstanding Transaction), and TClose
        % (Closed Transaction) based on the TNew
        [P,T,TOut,TClose] = updatePort(P,T,TOut,TClose,TNew);
    end
end
end
```

The updateTNew function has as entries the matrix with the daily prices of each share, the current date, the name of the share to be added, the side of the operation, i. e. 'buy' or 'sell', the quantity to be bought, the price and value of slippage and tCost, the costs per transaction. The function creates and returns the TNew table containing the new transactions to be made. To do this we first convert the input data to the required format.

```
function TNew = updateTNew(prices,dateTime,name,side,qty,orderPrice,...
    slipglobal,tCostHlobal)
% updateTNew create new transaction list, called TNew based on the buy and
sell lists

%% Data conversion
% Convert to cell array with upper case
name = {upper(name)};
side = {upper(side)};

% Convert cell array to matrix (array)
if iscell(qty)
    qty = cell2mat(qty);
end
```

Subsequently, the new transaction is created, the execution price is defined, which in this case is equal to the order price which is equivalent to the closing price of the stock for that day. After that the total cost of slippage and transaction cost paid to the broker are calculated. To do this the 5 basis points that have been defined for each of these costs are multiplied by the amount of shares to be bought and by the price per share. Thus, the total value of each of the expenses is obtained. Finally, the TNew table is created, whose columns are the date, the name of the share, the side, the quantity of shares, the order price, the execution price, the value of the slippage and the tCost.

```
%% Create TNew (new transaction)

    execPrice = orderPrice;

% Calculate slipage and transaction for this trade
slipage = execPrice*qty*slipglobal;
tCost = execPrice*qty*tCostHlobal;

% Create TNew (new transaction)
TNew = table(dateTime, name,side,qty,orderPrice,execPrice,slipage,tCost);
end
```

The second function called in the for-loop is updatePort. Its inputs are the P portfolio, T lists, TOut, TClose and TNew and its outputs are P, T, TOut and TClose update. First it is checked that the transaction list to be included is not empty. If this is the case, a message is displayed and the function is abandoned. In the case that it is not empty, the TNew content is added to the list T. Either by adding it at the end or by initializing T with the content of TNew in the case that no transaction has been made yet. Finally, the variables are extracted from the TNew table.

```
function [P,T,TOut,TClose] = updatePort(P,T,TOut,TClose,TNew)
% updatePort updates the portfolio (P), all transaction (T),
% outstanding transaction (TOut), and closed transaction (TClose)
% based on existing lists and new transaction list (TNew)

%% update all transaction (T)
% Check if TNew is empty, otherwise update T and extract variable from TNew
if isempty(TNew) == true
    disp('No Transaction')
    return
else
    if isempty(T) == true
```

```
        T = TNew;
    else
        T = [T ; TNew];
    end
    name = TNew.name;
    side = TNew.side;
    qty = TNew.qty;
    cost = TNew.execPrice;
end
```

In case there are open transactions, i.e. TOut is not empty, the number of open transactions and the number of shares held are stored. The isOpen() function then looks to see if the trade is open and therefore in the portfolio and if the new trade closes it, i.e. is of the opposite sign or if it is simply a trade in which new shares are added to the existing position. The function returns a variable that allows to know if the transaction closes the already existing one or if it goes in the same direction and the position of the stock in the portfolio, 0 if it is not included and the corresponding location if it is in the portfolio.

```
%% update P, TOut, and TClose
if isempty(TOut) == false
    % Number of transactions in TOut
    L1 = height(TOut);

    % Number of assets in Portfolio (P)
    L2 = height(P);

    % Check if TNew is open(1)/close(0) and get the location in P where
    % names are matched (not neccessary open)
    [open,loc] = isOpen(name, side, P);
```

In case the stock is not already in the portfolio, i.e. the location (loc variable) is 0, it is simply added to both the list of open transactions and the portfolio.

```
% If new transaction is not an existing name in P,
    % add new data into TOut and P
    if loc == 0
        TOut = [TOut; TNew];
        Pnew = table(name,side,qty,cost);
        P = [P;Pnew];
```

If the stock is already in the portfolio, location different from 0, and the variable 'flag' activated, that is, the transaction is in the same direction as the open operation, it is necessary to update the information with the new quantities and prices in both TOut and P. To do this, simply add TNew after the TOut list, add the new quantity of shares to be bought to the existing quantity in the portfolio, and update the average cost per share, since the price at which the share will be bought that day will most likely be different from the price at which they were bought in the past.

```
% If new transaction is an existing name (open) in P,
    % add new data into TOut, add qty to P, and update cost in P
    elseif open == 1
        TOut = [TOut; TNew];
        netqty = P.qty(loc)+ qty;
        P.cost(loc) = (P.cost(loc)*P.qty(loc) + cost*qty)/netqty;
        P.qty(loc) = netqty;
```

Thereafter the code manages the scenario in which the new transaction concerns an already existing asset in the portfolio and in the opposite direction, i.e. closing the transaction, either in full or in part. First the function browse through the assets on the TOut list in search of the transaction that corresponds to the opposite side of the one to be made. Once it has been found, as long as the amount to buy or sell is still positive, if the amount in TOut is higher than the amount to trade, it means that only the new amount has to be deducted from the old one. This means that it is only partially closed and there is no need to remove the whole transaction. Even if it is a partial close, it is added to the list of closed trades TClose with the closed amount corresponding to the amount of the new TNew trade. This is done by calling the local updateTClose function which will be discussed later. Then a new variable, newqty equal to the subtraction between the already existing amount and the new amount is created and entered into the TOut list. The new transaction cost amounts, both the slippage and the tCost, are updated as follows,

$$NewCost = \frac{OldCost \times NewQuantity}{OldQuantity}$$

Finally the new amount is set to 0 to mark that the transaction is completely matched.

Alternatively, if the new transaction involves an amount greater than that already held, i.e. a complete closure of the position held, the transaction must be completely deleted from the TOut list. To do this, the first step is to calculate the amount of qty2 that will be unmatched. Then, the closed transaction is added to TClose by means of the local function updateTClose in a similar way to the previous one. The amount of the open transaction in TOut is cancelled, since it is already closed. And an auxiliary variable is marked in such a way that the function will completely delete the row of TOut whose amount is zero.

```
% If new transaction is an existing name (close) in P,
    % update TOut and P
    elseif open == 0
        qty2 = qty;
        delrow = zeros(L1,1);

        % Go into each transaction of Tout
        for i = 1:L1
            TOutName = TOut.name{i}; % extract the name
            if strcmp(TOutName,name{1}) % if name is matched, then:
                %if qty2 is still positive
                if qty2 > 0
                    % if qty in Tout is greater than qty2, meaning that we
                    % have to deduct the qty from Tout and no need to
                    % delete this whole transaction out
                    if TOut.qty(i) > qty2
                        % Add new transact with qty2 to Tclose
                        TClose = updateTClose(TOut(i,:),TClose,TNew,qty2);
                        % New qty = original qty minus qty2 from Tnew
                        newqty = TOut.qty(i) - qty2;

                        % Prorate slipage and tcost in Tout because qty is
                        % reduced.
                        % calculate new slipage in Tout
                        TOut.slipage(i) = TOut.slipage(i)*newqty/...
                            TOut.qty(i);
                        % calculate new tcost in Tout
                        TOut.tCost(i) = TOut.tCost(i)*newqty/TOut.qty(i);
                        TOut.qty(i) = newqty; % update qty in Tout
                        qty2 = 0; % Flag that qty2 is completely matched

                        % if qty in Tout <= qty2, meaning
                        % that we have to delete the transaction from Tout
                    else
                        % Calculate unmatch qty in Nnew
                        qty2 = qty2-TOut.qty(i);
                        % Add new transact with qty2 to Tclose based on
                        % original qty in Tout
                        TClose = updateTClose(TOut(i,:),TClose,TNew,...
                            TOut.qty(i));
                        % reduce qty in Tout to zero (already closed)
                        TOut.qty(i) = 0;
                        % no need to update slipage and t cost as this row
                        % will be deleted
                        delrow(i) = 1; % mark this row for deletion

                    end
                end
            end
        end
```

Then, thanks to the auxiliary variable, the row is deleted in TOut.

```
%Delete row in Tout that were flag due to completely closed
        idx = find(delrow);
        for i = idx
            TOut(i,:) = [];
        end
```

Once TOut is updated, the function starts with the update of the portfolio P. With a similar for-loop, the portfolio is browsed, as soon as the asset corresponding to the one to be traded is found, the new amount is deducted from the existing amount. If the amount in the portfolio is still positive, the purchase price is updated using the local updatePortCost function. In the opposite case, the auxiliary variable is marked so that the function will delete the row of the portfolio whose quantity is zero.

```
delrow = 0;
        delrow = zeros(L2,1);
        for i = 1:L2 % Deduct qty of TNew from Port and update cost
            Pname = P.name{i};
            if strcmp(Pname,name{1}) %is name match?
                % Deduct the portfolio qty by the closing qty
                P.qty(i) = P.qty(i) - qty;

                % If the qty of asset in the portfolio is still positive,
                % then update the new cost based on FIFO
                if round(P.qty(i),9)> 0
                    P.cost(i) = updatePortCost(TOut,name,side);
                else
                    % If the qty of asset in the portfolio is not positive,
                    % then flag this asset to be deleted
                    delrow(i) = 1;
                end
            end
        end
```

Next, the row is deleted.

```
%Delete row in portfolio that were flag due to completely closed
        idx = find(delrow);
        for i = idx
            P(i,:) = [];
        end
    end
```

It is recalled that this entire process is included in the general case where the list of open transactions is not empty. In the alternative case in which it is empty, the new transaction is simply added to the portfolio P and to the list of open transactions TOut.

```
else % If TOut is empty
    P = table(name,side,qty,cost); % Add Tnew to P
    TOut = TNew; % Add Tnew to TOut
    % Tclose remain unchanged
end

end
```

The local functions called in the updatePort function are now detailed. First, updatePortCost. This sub-function allows to update the cost in the portfolio when the amount is reduced. The way to proceed is following the FIFO concept, First In First Out, that is, it is assumed that the first shares that were bought are the first shares to be sold in the case that there is a set of shares that have not been bought on the same day and that a part of those shares is to be sold. The first step is to initialize the necessary variables and in particular to define which is the opposite side of the operation to be done, if a purchase is to be made, the opposite side will be a sale, etc. Then, the function goes through the list of open transactions TOut looking for the operation that has the same name and the opposite side. Once it has been found, the total number of shares and the total monetary value are calculated. Finally, the new cost is defined, simply by dividing the monetary value by the total number of shares.

```
%% updatePortCost (local function)
function newCost = updatePortCost(TOut,name,side)
% This subfunction used is for updating the portfolio cost when the qty is
% reduced. The updated cost is based on FIFO (First In, First Out)

% Initialing parameters
value = 0;
totalQty = 0;
newCost = 0;

% Number of outstanding transactions
L = height(TOut);
```

```
% Find the opposite side
if strcmp(side{1},'BUY')
    oppSide = {'SELL'};
elseif strcmp(side{1},'SELL')
    oppSide = {'BUY'};
else
    error('Unknow Side')
end

% Looping into outstanding transaction one-by-one
for j = 1:L
    % Check if the name is matched
    testname = strcmp(TOut.name{j},name{1});
    % Check if the side is opposite
    testside = strcmp(TOut.side{j},oppSide);

    % If names are match and sides are opposite, then calculate
    % total qty and value (value = sum of price*qty)
    if testname && testside
        value = value + TOut.qty(j)* TOut.execPrice(j);
        totalQty = totalQty + TOut.qty(j);
    end
end

if totalQty == 0
    % In case that there is no qty found (Not supposed to happen)
    error('No qty found in Tout')
else
    % Calculate new cost
    newCost = value/totalQty;
end
end
```

The second local function is updateTclose. It updates the list of closed transactions TClose when there is a new transaction with the opposite direction of one currently in the portfolio. Firstly, the opening and closing dates of the transaction, the name of the stock, the opening and closing sides and the opening and closing prices are extracted. In case the opening and closing sides are opposite, which is expected, the profit is calculated,

$$Capital\ Gains = Close\ Quantity\ \times\ (Close\ Price\ -\ Open\ Price)$$

In the event that portfolio shares are being sold.

$$Capital\ Gains = Close\ Quantity\ \times\ (Open\ Price\ -\ Close\ Price)$$

In the event that a stock is being bought after being shorted.

The new slipage charges and costs paid to the broker are updated subsequently.

$$New\ Global\ Cost = \frac{Old\ Cost\ \times\ Closed\ Quantity}{Old\ Quantity} + \frac{New\ Cost\ \times\ Closed\ Quantity}{New\ Quantity}$$

Finally, once this information has been calculated, the list of closed transactions TClose is updated by adding them to the list that already exists or by initializing the TClose list if none has been closed yet.

```
%% updateTClose (local function)
function TClose = updateTClose(TOut,TClose, TNew, closeQty)
% This function is used for updating the closed transaction when there is a
new
% transaction with the opposite side (close the transaction) from the one
in portfolio.

% Extract open/close datetime from Tout and Tnew respectively
opendatetime = TOut.dateTime;
closedatetime = TNew.dateTime;

% Compare the name in Tout and Tnew (should be the same)
name1 = TOut.name; % Extract name from Tout
name2 = TNew.name; % Extract name from Tnew
if strcmp(name1{1},name2{1})
    % create new variable ('name') if both names are the same
    name = name1;
else
    % Return error if name is not the same
    error('Unmatched name')
end


openSide = TOut.side; % Extract side from Tout as an open side
closeSide = TNew.side; % Extract side from Tnew
openPrice = TOut.execPrice; % Extract openprice from Tout
closePrice = TNew.execPrice; % Extract closeprice from Tnew

% Compare the side in Tout and Tnew (should be opposite)
if strcmp(closeSide,'BUY')
    if strcmp(openSide,'SELL')
        % Calculate capital gain when open side is sell
        capGain = closeQty*(openPrice-closePrice);
    else
        error('Same side')
    end
elseif strcmp(closeSide,'SELL')
    if strcmp(openSide,'BUY')
        % Calculate capital gain when open side is buy
        capGain = closeQty*(closePrice-openPrice);
    else
        error('Same side')
    end
else
    error('Side is unknown')
end

% Update slipage and tcost in outstanding transactions
% Concept here is to prorate the slipage based on the existing
qty1 = closeQty;
qty2 = TNew.qty;
qty3 = TOut.qty;
slipage = TOut.slipage*qty1/qty3 + TNew.slipage*qty1/qty2;
tCost = TOut.tCost*qty1/qty3 + TNew.tCost*qty1/qty2;
```

```
% Update the closing transaction table (add new one to the bottom if there
% is already an existing Tclose
TCloseNew = table(opendatetime,closedatetime,name,openSide,openPrice,...
    closePrice, closeQty, slipage, tCost, capGain);
if isempty(TClose)
    TClose = TCloseNew;
else
    TClose = [TClose; TCloseNew];
end
end
```

Finally, the third local function that has been called isOpen(). This function allows to determine if the new transaction to be made, TNew, involves an asset already in the portfolio, and in that case if it is adding more shares to the position or if it is closing the operation that had been previously opened. To do this, first the open variable is initialized to 1, this variable is used as a flag to know if the new trade closes or opens even more the already existing position. If its value is 0 it means that it closes, if it is 1 the opposite. In the same way the location of the eventual asset in the portfolio is initialized to 0. In this way, the most probable assumption is that the asset is not in the portfolio. The next step is to go through the portfolio to find out if one of the names corresponds to the name of the stock to be traded. If it is found, the location variable is updated with the position that the asset has in the portfolio. Then the sides of the new transaction and the asset in portfolio whose names match are compared. If they are opposite, i.e. the position is being closed, the open variable is given the value 0.

```
%% Check if TNew is open (same direction, adding more stocks to the
existing) close (opposite side) order
function [open, location] = isOpen(name, side, P)
    % Check if TNew is open (1) or close (0) transaction
    open = 1; % Set default as a new (1)
    location = 0; % Set default location
    L1 = height(P); % Number of transactions in P

    for i = 1:L1
        Pname = P.name{i}; % Extract name from P

        % Find the location in the table where names are matched
        % name  is supposed to be unique in P
        if strcmp(Pname,name{1})
            location = i;
            PSide = P.side{i}; % Extract side from P
            % Find opposite side of the transaction in P
            if strcmp(PSide,'BUY')
                oppSide1 = {'SELL'};
            elseif strcmp(PSide,'SELL')
                oppSide1 = {'BUY'};
            else
                error('Unknow Side')
            end

            % If sides are opposite, flag as a closing transaction
            if strcmp(side{1},oppSide1)
                open = 0;
            end
        end

    end
```

```
end
```

This concludes the explanation of the updatePort function and of the submitOrder function, one of the most important of the PortfolioManagement function.

The second function called from PortfolioManagement is calcPL. Its entries are the matrix with the daily closing prices of each stock, the date, the list of open transactions TOut, the list of closed transactions TClose, the matrix with the names of the stocks, and the number of the day the simulation is on. The purpose of this function is to calculate the profit and loss characteristics and transaction costs that have been incurred so far. Therefore, all the numbers are cumulative, i.e. it does not show the profit of day i, but the profit obtained from the first day to day i. The first step is as usual to initialize the variables that will be used in the function.

```
function [pnl, rpnl, urpnl, slipage, tCost] = calcPL(prices,Days,...
 TOut, TClose,names, nbDay)
% calcPL will calculate and update the P&L (pnl), realized P&L (rpnl),
unrealized P&L
% (urpnl), actual slipage, and actual transaction cost.
% Every number is cumulative, allowing to maintain a record of the
% evolution of those variables

% Set initial status of variable
urpnl = 0;
capGain = 0;
slipage = 0;
tCost = 0;
```

Then, in case the list of closed transactions TClose is not empty, the capital gains are calculated as the sum of the gains for each transaction, and both transaction fees as the sum of each fee also for each one of the transactions.

These variables are then calculated for transactions that remain open. Similarly, slippage and brokerage fees are obtained by adding these fees for each open transaction. The list of open transactions is then browsed extracting the name, side and amount of each asset. This is done in order to calculate unrealized gains. To do this, the opening price, which is stored in the list of open transactions TOut, and the price of the asset on that day, which is considered as the hypothetical closing price, are extracted, even though in this case the transaction is not being closed. To obtain the closing price, the getPrice function is used, which will be analyzed later. The unrealized profit is obtained in the following way,

$$Unrealizaed\ P\&L = Quantity\ \times\ Side\ \times\ (Close\ Price\ -\ Open\ Price)$$

Where Side = 1 in case it is a purchase and -1 if it is a short sale.

The realized profits are obtained in the following way,

$$Realizaed\ P\&L = Capital\ Gains - Slipage - Brokerage\ Cost$$

Finally, the total profit or loss is obtained by applying the following formula,

$$P\&L = Realized\ P\&L + Unrealized\ P\&L$$

This completes the calcPL() function, displayed below.

```
%% Calculate P&L related variables from closed transaction (TClose)
% No unrealized P&L here because it can only be in TOut
if ~isempty(TClose)
    capGain = sum(TClose.capGain);
    slipage = sum(TClose.slipage);
    tCost = sum(TClose.tCost);
end

%% Calculate P&L related variables from oustanding transaction (TOut)

% Calculate P&L if Tout is not empty
if ~isempty(TOut)

    % Find slipage and tCost in tOut
    slipage = slipage + sum(TOut.slipage);
    tCost = tCost + sum(TOut.tCost);

    nTOut = height(TOut);
    for i = 1:nTOut
        name = TOut.name{i};
        side = upper(TOut.side{i});
        qty = TOut.qty(i);

        % Assign side = 1 for BUY, -1 for SELL
        if strcmp(side,'BUY')
            side1 = 1;
        elseif strcmp(side,'SELL')
            side1 = -1;
        else
            error('Wrong side')
        end

        % calculate unrealized p&L
        openPrice = TOut.execPrice(i);
        closePrice = getPrice(prices,name,names, nbDay);
        urpnl = urpnl+ qty*side1*(closePrice-openPrice);
    end
end
%% Calculate realized P&L and Total P&L
rpnl = capGain - slipage - tCost;
pnl = rpnl + urpnl;
end
```

The next function explained is getPrice() which has as entries the matrix with the daily prices of each share, the name of the share from which the price will be obtained, the matrix with all the names of the shares and the number of the day for which the price will be obtained, (day 1 is equivalent to the first day of the simulation). The function therefore returns the price of the stock as an argument for the day ordered. This function is generally used to extract the current price of the stock.

The function first looks for the position in the columns of the daily price matrix of the stock from which the price will be obtained. Remembering that the columns of the daily price matrix correspond to the shares that make up the index and each of the rows is equivalent to one day of the simulation. To obtain the position of the stock in the matrix, the name is searched in the array with all the names. Once the location is known, the function returns the price with this location in the column and for the day required, as shown below.

```
function Price = getPrice(prices,name,names, i)
% This functio extracts the historical price of one stock based on date,

% Find the name in the universe of assets (names)
cellfind = @(string)(@(cell_contents)(strcmp(string,cell_contents)));
col = cellfun(cellfind(name),upper(names));

%% Extract price based on date

Price = prices(i,col);
end.
```

The next function that is explained is the one that allows to make the optimization of the portfolios, it will also be explained in which consists this optimization. Remember that this function does not have to work in all the simulations, but that it is part of the creation of the investment strategy to determine whether or not there will be an optimization of the strategy. The entries in the PortfolioOptimization() function are the list of assets held in the portfolio, including those assets which are to be bought or sold on that day. It also includes the weights of each asset relative to the total value of the portfolio, the daily price matrix, the daily price matrix not only with the dates of the simulation but with all the previous dates available, the matrix with all the names of the stocks, the number of the simulation day, the available cash and the total value of the portfolio.

The first step is to initialize the variables needed to optimize the algorithm so that it works faster and more efficiently. Then, from the extended daily price matrix, a daily price matrix is constructed only for the assets held in the portfolio. It is important to emphasize here that although the prices are available for the whole period of the simulation, it is only possible to use the prices that were available until that day. If all the prices were used, the strategy would be using prices that corresponded to future prices at that time and that, by definition, would not be possible. Using the tick2ret() function in MATLAB's Financial Toolbox, each price series is transformed into a series of daily returns. The tick2ret() function simply calculates the change in a day's price from the previous day's price. The mean() and cov() functions then calculate the matrix with the averages of the returns for each asset and the covariance matrix for the returns of the assets. This is the reason why the extended price matrix is used, since the greater the number of data available, the better the calculation of the averages and the covariance matrix will be.

Portfolio Optimization

We optimize the portfolio monthly, in order to obtain an expected maximum sharpe ratio.

```
function [BuyList, SellList, BuyQty, SellQty, BuyPrice, SellPrice,
Cash]= PortfolioOptimization(AssetList, InitPort, prices, prices_stats,
names, i, Cash, TotalQty)
```

```
    % Compute mean and covar of the assets, for this we need the 
columns of
    % prices for these assets

    [~, numAssets] = size(AssetList);
    [nDay, ~] = size(prices);
    BuyList = cell(numAssets,1);
    SellList = cell(numAssets,1);
    BuyQty = cell(numAssets,1);
    SellQty = cell(numAssets,1);
    BuyPrice = cell(numAssets,1);
    SellPrice = cell(numAssets,1);
    %Price = zeros(nDay, numAssets);

    % Find the position of the name in the vector of names of the index
    for l = 1:numAssets
        cellfind = 
@(string)(@(cell_contents)(strcmp(string,cell_contents)));
        col = cellfun(cellfind(AssetList{l}),upper(names));


        %% Extract column with prices of the asset
        if prices_stats(253, 1) == prices(1, 1)
            Price(:,l) = prices_stats(1:253-1+i,col);
        elseif prices_stats(end-504, 1) == prices(1, 1)
            Price(:,l) = prices_stats(1:end-504-1+i,col);
        else
            error('no known difference')
        end
    end
    Returns = tick2ret(Price);
    AssetMean = mean(Returns);
    AssetCovar = cov(Returns);

    AssetList_2 = string(AssetList);
    AssetList = cellstr(AssetList_2);
```

The next step is to create the portfolio object, p. This is one of the objects that can be created using MATLAB Financial Toolbox. It includes all of the following properties:

```
          BuyCost: []
         SellCost: []
     RiskFreeRate: []
        AssetMean: []
       AssetCovar: []
    TrackingError: []
     TrackingPort: []
         Turnover: []
      BuyTurnover: []
     SellTurnover: []
             Name: []
        NumAssets: []
        AssetList: []
         InitPort: []
      AInequality: []
      bInequality: []
        AEquality: []
        bEquality: []
       LowerBound: []
       UpperBound: []
      LowerBudget: []
      UpperBudget: []
      GroupMatrix: []
       LowerGroup: []
       UpperGroup: []
           GroupA: []
           GroupB: []
       LowerRatio: []
       UpperRatio: []
     MinNumAssets: []
     MaxNumAssets: []
        BoundType: []
```

In this case, the AssetList property is defined with the variable that contains the list of all the assets included in the portfolio. Thanks to the function setAssetMoments, the AssetMean and AssetCovar properties are also defined with the averages and the covariance matrix that were calculated in the previous step. Then the property InitPort is initialized, with the weights that each asset has in the portfolio and the number of assets in the portfolio through the formula setInitPort(). Thanks to the function setDefaultConstraints(), all the other essential properties are initialized with their default values.

```
    p = Portfolio('AssetList', AssetList);
    p = setAssetMoments(p, AssetMean, AssetCovar);
    p = setDefaultConstraints(p);
    p = setInitPort(p,InitPort, numAssets);
```

There follows the most important function of the PortfolioOptimization function and the one that actually performs this optimization. This is a function included in the MATLAB Financial Toolbox called estimateMaxSharpeRatio().

```
[pwgt, pbuy, psell] = estimateMaxSharpeRatio(p, 'Method', 'iterative');
```

The Sharpe Ratio is one of the most commonly used ratios for knowing whether an investment strategy is performing well because it allows to understand the return obtained compared to the risk taken. To obtain it, a value called risk-free rate is subtracted from the total return obtained. To calculate the Sharpe Ratio, the value of the American Treasury bills is generally used as the risk-free rate. In this case we have simply used 0 for the reasons mentioned above.

This excess return compared to the risk-free rate is divided by the risk or volatility which in this case is simply the standard deviation of the returns obtained.

$$\boldsymbol{Sharpe\ Ratio} = \frac{\boldsymbol{Return_{portfolio} - Return_{risk-free\ asset}}}{\boldsymbol{\sigma_{portfolio}}}$$

In order to understand what the optimization of a portfolio means, it is important to first explain the concept of the efficient frontier. From a collection of stocks, a set of optimal portfolios can be built. This is based on the mean-variance portfolio theory, that says that any investor will choose the portfolio that maximizes the return for a certain level of risk or that minimizes the risk of the portfolio for a certain level of return. It is possible to obtain the return and risk characteristics of each of the stocks that form a portfolio as well as the correlations between the returns of these assets. By shifting the allocation to each of the assets, several combinations of portfolios are obtained, each with different risk and return characteristics. These combinations can be represented graphically, with on the X-axis the standard deviation which is represented as a proxy for risk and on the Y-axis the expected return of the investment. For each level of expected return, the portfolio with the lowest risk is selected. In this way a minimum-variance frontier is created, i.e. the set of all portfolios with the minimum standard deviation for this level of return. All the portfolios that belong to that frontier produce the highest return per unit of risk. The leftmost point of the curve represents the portfolio with the lowest possible risk, known as the global minimum-variance portfolio. The portion of the curve above that point to the right is known as the efficient frontier or the Markowitz efficient frontier due to Howard Markowitz and his famous work in which he developed this concept and which has been studied in the introduction to this paper. Therefore, when this paper talks about an optimized portfolio, it refers to a portfolio that is located on the efficient frontier.

The estimateMaxSharpeRatio() function searches along this efficient border for the portfolio that maximizes the Sharpe Ratio. More precisely, it seeks to find the relative weight of each asset in the portfolio so that the portfolio is on the efficient frontier and is expected to maximise the Sharpe Ratio. The function has as inputs the portfolio object p and the computation method to be used. The function returns the weights of each asset so that they meet the optimization objective. It also returns those weights separated in two variables, one with the weight of each asset that has to be bought to reach the target weight. Another variable with the weights of each asset that have to be sold to reach the target weight.

The following is an example of an efficient frontier with an optimized portfolio, obtained with this optimization tool. It shows the location of the initial and optimized portfolios, as well as the curve of estimated sharpe ratios that enables to decide where the optimized portfolio should be located over the efficient frontier.

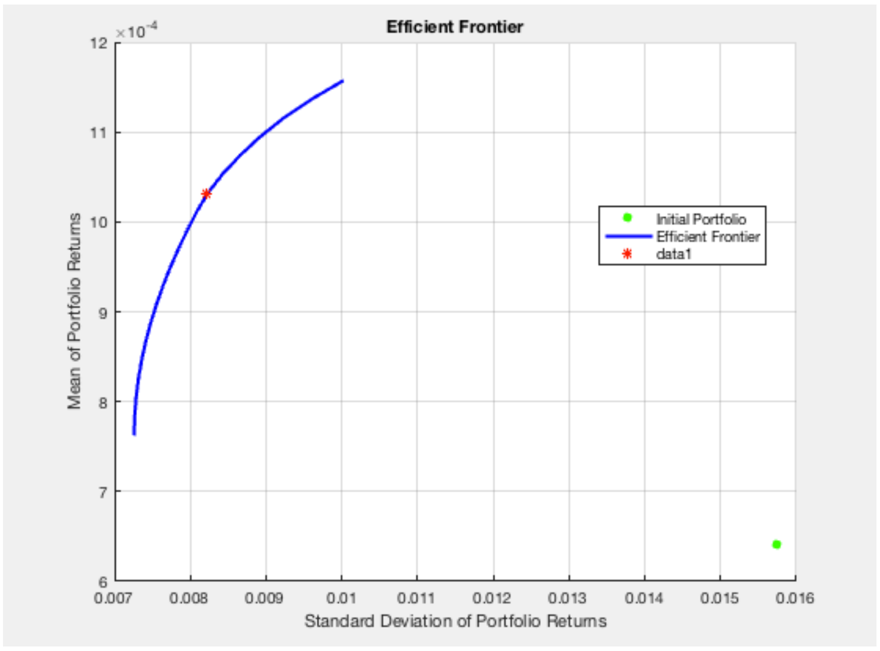


Figure 37: Efficient frontier with initial and optimized portfolios

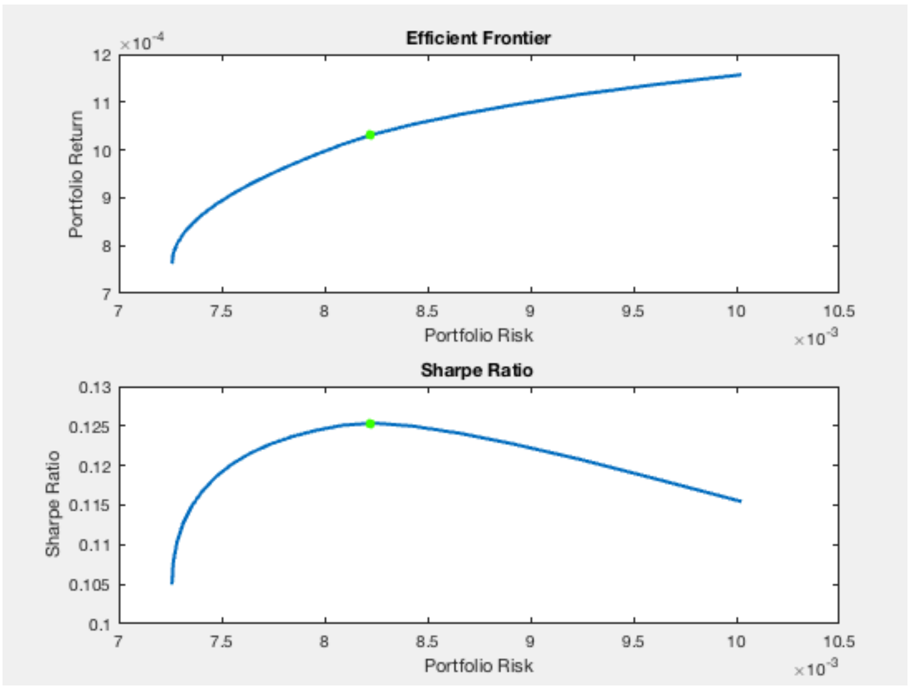


Figure 38: Efficient frontier and Sharpe ratio

From the two variables with the weights of the stocks that have to be bought and sold, the lists of those stocks are constructed. To do this, if the weight of the share is positive in the purchase variable, it is added to the list of shares to be bought and if the weight is positive in the sales variable, it is added to the list of shares to be sold. Likewise, to fill in the list with the amounts of shares to buy, the following formula is used to obtain each purchase amount, where Total Qty equals the total value of the portfolio. The procedure is similar for the sale amounts.

$$\boldsymbol{Buy\ Quantity} = \frac{\boldsymbol{Buy\ Weight * Total\ Qty}}{\boldsymbol{Price}}$$

```
for j = 1:numAssets
         caseBuy = 0;
         caseSell = 0;
         if round(pbuy(j),9) ~= 0
            caseBuy = 1;
            BuyList{j} = AssetList(j);
         end
         if round(psell(j),9) ~= 0
            caseSell = 1;
            SellList{j} = AssetList(j);
         end
      % Set buying/selling prices if there is any action

         if caseBuy == 1
            BuyPrice{j} = getPrice(prices, AssetList(j), names, i);
            BuyQty{j} = (pbuy(j)*TotalQty)/BuyPrice{j};
         end

            if caseSell == 1
                SellPrice{j} = getPrice(prices, AssetList(j), names,
i);
                SellQty{j} = (psell(j)*TotalQty)/SellPrice{j};
            end
     end
```

Then, as in the PortfolioManagement function, the empty rows are deleted from each list. Thus, the required amounts for each asset currently held in the portfolio are obtained in such a way that the portfolio is optimized, i.e. over the efficient frontier and with a maximum expected Sharpe Ratio.

```
% Delete empty row
        delBuy = false(numAssets,1);
        for b = 1:numAssets
            if isempty(BuyList{b}) == true
                delBuy(b) = true;
            end
        end

        delSell = false(numAssets,1);
        for c = 1:numAssets
            if isempty(SellList{c}) == true
                delSell(c) = true;
            end
        end

        if max(delBuy) == 1
            BuyList(delBuy) = [];
            BuyQty(delBuy) = [];
            BuyPrice(delBuy) = [];
        end

        if max(delSell) == 1
            SellList(delSell) = [];
            SellQty(delSell) = [];
            SellPrice(delSell) = [];
        end
```

The last auxiliary function called from PortfolioManagement is DeleteOrders, it is an extremely simple function that only deletes the rows of the portfolio tables P, T transactions and open transactions TOut for which there is a row with quantity 0.This is done because those rows were created to allow the PortfolioOptimization function to know which were the new stocks we wanted to put in the portfolio, but since the optimized amounts were not yet known, they were entered with a 0 size.

The function is shown below.

Delete Orders

We delete the orders with 0 quantity. They were made before rebalancing the portfolio.

```
function [P, T, TOut, TClose]= DeleteOrders(P, T, TOut, TClose)


    P_index = find(P.qty == 0);
    P(P_index,:) = [];

    T_index = find(T.qty == 0);
    T(T_index,:) = [];

    TOut_index = find(TOut.qty == 0);
    TOut(TOut_index,:) = [];


End
```

This is the end of the PortfolioManagement function. Therefore, the final portfolio on the last day of the simulation P, the list with all the transactions that have been made during the simulation, the list with all the transactions that are still open on the last day of the simulation, those that have been closed during the simulation and a table with the characteristics of P&L and their evolution throughout the simulation have been obtained.

Once these variables are available, the analysis of the results can be performed. The objective of this part is to determine whether the strategy has been successful or not. To do this, there are a certain number of ratios and indicators that can be calculated. Such as the Sharpe Ratio, the maximum drawdown, i.e. the greatest decline observed during a period of time, from a peak to trough, before a new peak is formed. This is a very useful measure as it allows to see how bad the strategy has been at its worst. It is important because one of the problems that most investors face in practice is not so much finding a good strategy, but being able to stick with the strategy that has been chosen in the bad times, which will always come regardless of the strategy. By studying the maximum drawdown, the investor gets an idea of what he will have to endure in the worst moments of the strategy, and if that drawdown is too much for him, it is better not to implement that investment strategy. Regardless of how good the long-term results are, he will not get them because he will probably sell all his positions when he experiences a greater drop than he can handle. Another widely used financial ratio, included in the MATLAB Financial Toolbox, is the information ratio. It allows to measure the returns obtained above those obtained by the benchmark index by comparing them to the volatilities of those returns. In general, it allows to determine how well the strategy is working in terms of generating excess return relative to the benchmark. This is ultimately the objective of any active investment strategy.

It is based on the following formula,

$$Information\ Ratio = \frac{Portfolio\ Return - Benchmark\ Return}{Tracking\ Error}$$

$$Tracking\ Error = Standard\ Deviation\ (Portfolio\ Return\ - Benchmark\ Return)$$

Finally the risk-adjusted alpha of the portfolio is calculated, it is also known as Jensen's Measures. Alpha is how the return obtained by the strategy above the return of the index is defined.

Performance analytics

```
portValue = table2array(newPLTable(:,2))+initialCash;
portReturn = tick2ret(portValue);
sharpeRatio = round(sharpe(portReturn,riskFree),4)*sqrt(tDay);
pnlFinal = pnl;
```

Visualize the equity curve wtih initial portfolio value of 1.

```
[Benchmark] = Benchmark(index, SP500_DaysinSample);


equitycurve = ret2tick(portReturn);
BenchReturn = tick2ret(Benchmark);
Benchmarkcurve = ret2tick(BenchReturn);


figure
plot(SP500_DaysinSample, equitycurve)
hold on
plot(SP500_DaysinSample, Benchmarkcurve)
legend('Rule-based', 'Benchmark','Location','best')

Max_Drawdown_Strategy = maxdrawdown(equitycurve);
Max_Drawdown_Benchmark = maxdrawdown(Benchmarkcurve);


cash = portReturn;
cash(:) = 0;
```

```
Portfolio_Alpha = portalpha(portReturn, BenchReturn, cash, 'capm');


[Information_Ratio Tracking_Error] = inforatio(portReturn,
BenchReturn);
```

Finally, the equity curve of the portfolio is displayed. The equity curve of a portfolio represents the evolution over time of the portfolio's value assuming that the initial value is 1. In this graph it is compared with the equity curve of the benchmark. In this way it is possible to quickly compare visually how the strategy compares to someone who would have bought and held the benchmark over the same period of time.

To do this in the first place the Benchmark function is used. This function has as entries a variable that contains the name of the index to be used and the matrix with the dates of the period required. The first step is to define the start and end time from the matrix with all the dates. Then using a switch structure select the index to be used and load the data of the closing prices from the database in Excel. These data are transformed to be in an array, that is to say a single column with the daily prices of the index during the specified period. This column is the variable returned by the function. This function is shown below,

```
function [Benchmark] = Benchmark(c, DaysinSample)

    % We obtain the benchmark for each case

    startTime = DaysinSample(1);
    endTime = DaysinSample(end);
    S = timerange(startTime, endTime,'closed');

    switch c
        case 'SP500'
            dataBenchmark =
readtable("SP500_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'ATX'
            dataBenchmark =
readtable("ATX_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'CAC'
            dataBenchmark =
readtable("CAC40_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'DAX'
            dataBenchmark =
readtable("DAX30_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'FTSE_MIB'
            dataBenchmark =
readtable("FTSE_MIB_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'FTSE100'
            dataBenchmark =
readtable("FTSE100_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'MC'
            dataBenchmark =
readtable("Mercado_Continuo_Benchmark.xlsx",'PreserveVariableNames',tru
e);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
        case 'SP_TSX60'
            dataBenchmark =
readtable("SP_TSX60_Benchmark.xlsx",'PreserveVariableNames',true);
            Benc = table2timetable(dataBenchmark);
            Ben = Benc(S,:);
            Be = timetable2table(Ben);
            B = Be(:,2);
            Benchmark = table2array(B);
    end

end
```

Then, using the functions ret2tick() and tick2ret() which allow to transform a series of returns into prices and a series of prices into returns respectively, the index prices and the profit evolution of the portfolio are transformed into a series of prices starting at value 1, i.e. both equity curves. This are the time series used for display the equity curves.

These functions are used in the same way for both the In Sample and Out of Sample periods as shown in Figure 31 **Figure 31**above, there is no change from one period to the next.

This concludes the explanation of the functions that are common to all the strategies.

In the next chapter, the different investment strategies used, their characteristics and how those differences translate into the different codes will be explained.

4.2 Successful strategies

4.2.1 Rule Based Strategies

The first and simplest of the sets of strategies that have been developed are the rule-based strategies. It includes strategies in which buy and sell signals are generated when one or more simple rules are met. It is therefore a systematic strategy, in which the investor decides what the rules are, but once the strategy is implemented he does not have to decide anything else, i.e. he does not have to make the decision whether to buy or sell certain shares individually. The investor may well not know which shares compose his portfolio. One of the advantages of this type of strategies is that, as has been said on a number of occasions during this work, on a large number of times it is the investor himself who hinders his success in the financial markets. Either because he makes too many decisions, because he considers his skills for predicting the evolution of the financial market to be above average and above what they really are, or because of any of his possible biases. By establishing a systematic strategy, it eliminates an important number of decisions, particularly the most dangerous ones, those that have to be made in the heat of the moment, when a stock has fallen by 25%, should I sell it or not? When a stock seems to be on a roll and everyone is talking about it but the strategy has not said that I have to buy it, do I have to make an exception or not? All these types of doubts are eliminated if the strategy is automated. There are still certain decisions to be made, such as which rules to incorporate, but this process can be done calmly and over time, without the pressure of having to act immediately.

Consequently, a certain number of strategies have been developed, combining some rules that can be created thanks to the indicators that have been calculated. All of these strategies are included in a set called Rule Based Strategy, and all of them have been tested in the same MATLAB script that has the same name even though they are different strategies. The implementation of these rules is done in the function called Signal Creation, which has been discussed above. This function is called before the PortfolioManagement function, and has as entries the table with the daily indicators of each stock, the number of stocks that make up the index, and the number of days that the period has. The function returns the matrix with the buy, sell and hold signals. It is a matrix where each column represents a stock and each row represents a day of the period to be simulated. The buy signals are represented with a 1, the sell signals with a -1 and the hold signals with a 0. So if in row 10, column 5 there is a 1, it means that on the tenth day of the simulation, the stock that has position 5 of that index has to be bought, reminding that the same order has been kept in all the variables so that there are no errors and it is easy to know what stock it is. The possible rules that have been used to create the strategies are the RSI, the classic version in which a share is bought when the RSI goes below 30 and sold when its value goes above 70. A rule based on momentum using the annual return of the share, for this purpose the shares are classified according to their annual return, the 20% with the highest annual return is bought and the 20% with the worst annual return is sold. Finally, a rule based on the share's beta has also been developed. Similar to the momentum indicator, the shares are classified according to their beta; in this case, those in the 50% with the lowest beta are bought and those in the 50% with the highest beta are sold. In this way, a strategy is followed that favors those with a lower beta. As shown in the article by the asset management company based on quantitative strategies AQR, “Betting Against Beta” by Andrea Frazzini a, Lasse Heje Pedersen [24], it has been demonstrated in various types of assets, over various markets and different periods that shares with a low beta tend to do better than those with a higher beta. This work also proves that such a factor exists as will be shown below. In the AQR article, they present some interesting reasons for the existence of such a factor.

The process that has been followed has been to generate two strategies, one purely of momentum, joining the rules of RSI and annual return, and another of momentum with low beta. The first strategy has the SignalCreation function associated, while the second is called SignalCreationLowBeta. First, the contents of the SignalCreation function are explained in detail.

Create rule-based signal and backtest (In-sample)

We obtain a list for each in sample day, if signal = 1, we buy, if signal = -1, we sell, if signal = 0, we hold.

The signals are created based on the RSI, the yearly return MOM(12, 2) momentum indicator.

```
index = 'SP500';
SP500_nDay = height(SP500_inSample);
[SP500_signalinSample] = SignalCreation(SP500_inSample, SP500_nStocks,
SP500_nDay);
```

The first step is to extract from the table with all the indicators the RSI of each stock, to do this the function looks into the table for the columns containing the word 'RSI'. Once the columns have been extracted using the formula,

$$\boldsymbol{Signal} = \mathbf{1} \times (\boldsymbol{RSI} < \mathbf{30}) - \mathbf{1} \times (\boldsymbol{RSI} > \mathbf{70})$$

A one is generated when the RSI is less than 30 and a -1 when the RSI is greater than 70.

Creation of the Signal Matrix

We obtain a list for each in sample day, if signal = 1, we buy, if signal = -1, we sell, if signal = 0, we hold.

Based in the RSI and a classification of the 12 month return.

```
function [signal] = SignalCreation(T, nbStocks, nDays)


   hasRSI = ~cellfun('isempty', regexp(T.Properties.VariableNames,
'RSI', 'once'));
   RSI =  T(:,T.Properties.VariableNames(hasRSI));
   RSI = table2array(RSI);
   signal1 = 1*(RSI < 30) - 1*(RSI > 70);
```

Afterwards, the annual return is extracted in a similar way, looking for the columns containing Ret252 which is the code that was assigned to it when the indicator was created. To classify the stocks according to their annual return, once the columns have been extracted, the sort() function is used, which simply sorts them from lowest to highest. Then, the upper and lower limits are created, both at 20% and through a for-loop that goes through the matrix in the vertical dimension, that is, in the days dimension, a 1 is assigned to the actions that that day are in the top 20% with the best return, a -1 to those that are in the worst 20% and a 0 to those that are in between.

```
hasRet252 = ~cellfun('isempty', regexp(T.Properties.VariableNames,
'Ret252', 'once'));
   Ret252 = T(:,T.Properties.VariableNames(hasRet252));
   Ret252 = table2array(Ret252);
   [~,I] = sort(Ret252,2);
   uplimit = round(0.8*nbStocks);
   lowlimit = round(0.2*nbStocks);
   signal2 = Ret252;
   for a = 1:nDays
       signal2(a,I(a,uplimit:end)) = 1;
       signal2(a,I(a,1:lowlimit)) = -1;
       signal2(a,I(a,lowlimit+1:uplimit+1)) = 0;
       signal = zeros(nDays,nbStocks);
  end
```

The last step is to put both rules together, to do this a double for-loop is created that goes through the two matrices that each rule has created, when both have a buy signal, the result will be a buy signal, when the RSI gives a sell signal it is sold.

```
for i = 1:nDays
     for j = 1:nbStocks
         if signal1(i,j) == 1 && signal2(i,j) == 1
             signal(i,j) = 1;
         elseif signal1(i,j) == -1 && signal2(i,j) == -1
             signal(i,j) = -1;
         elseif signal1(i,j) == -1
             signal(i,j) = -1;
         else
             signal(i,j) = 0;
        end
     end
```

```
  end
```

Thus the matrix with the buy and sell signals for this strategy is created.

The creation of this matrix is explained below in the case of the strategy that also incorporates low beta. The principle is exactly the same, the annual return is extracted and the signals are generated, in this case also the limits for the return, both the upper and the lower, are 20%.

Creation of the Signal Matrix

We obtain a list for each in sample day, if signal = 1, we buy, if signal = -1, we sell, if signal = 0, we hold.

Based on the 12 month return and a classification of the average beta of the stock

```
function [signal] = SignalCreationLowBeta(T, nbStocks, nDays)

   hasRet252 = ~cellfun('isempty', regexp(T.Properties.VariableNames,
'Ret252', 'once'));
   Ret252 = T(:,T.Properties.VariableNames(hasRet252));
   Ret252 = table2array(Ret252);
   [~,I] = sort(Ret252,2);
   uplimit = round(0.8*nbStocks);
   lowlimit = round(0.2*nbStocks);
   signal2 = Ret252;
   for a = 1:nDays
       signal2(a,I(a,uplimit:end)) = 1;
       signal2(a,I(a,1:lowlimit)) = -1;
       signal2(a,I(a,lowlimit+1:uplimit+1)) = 0;
   end
```

The management of the beta is done in a very similar way to that of the annual return. First, this indicator is extracted, also by searching the table with all the indicators. Then, it is classified from lower to higher beta, using the sort() function, the signals are also generated using a for loop. For each day, a -1 is assigned if the stock is in the top 50%, a 1 if it is in the 50% with the lowest beta. In this case there are no intermediate values, although the line of code that manages this case is included in case other percentages are chosen.

```
hasbeta = ~cellfun('isempty', regexp(T.Properties.VariableNames,
'Beta', 'once'));
   beta = T(:,T.Properties.VariableNames(hasbeta));
   beta = table2array(beta);
   [~,I] = sort(beta,2);
   uplimitbeta = round(0.5*nbStocks);
   lowlimitbeta = round(0.5*nbStocks);
   signal3 = beta;
   for b = 1:nDays
       signal3(b,I(b,uplimitbeta:end)) = -1;
       signal3(b,I(b,1:lowlimitbeta)) = 1;
       signal3(b,I(b,lowlimitbeta+1:uplimitbeta+1)) = 0;
    end
```

Then both rules are put together, in the case where both give a buy signal, a buy signal is generated, in the case where both give a sell signal, a sell signal is generated.

Putting both strategies together in this way generates too many signals, so a second filtering is done, by means of two for loops in which for each stock, if the day before there was a buy signal, and the day being analyzed also, a buy signal is generated. If the previous day was a buy signal, and the day being analysed is a hold signal, a sell signal is produced. The idea behind this signal is that when it is no longer a buy signal but a hold signal, it means either that it is no longer in the top in terms of return, or that its beta has risen too much, so it is no longer wanted in the portfolio. In the case that it goes from a buy signal to a sell signal, a sell signal is generated. If it goes from a hold signal to a buy signal, a buy signal is generated. If it goes from a hold signal to a hold signal, a hold signal is generated. If it goes from a hold signal to a sell signal, it is sold. If it switches from a sell signal to a buy signal, it is bought. If it goes from a sell signal to a hold signal, it is held, and finally if it goes from a sell signal to a sell signal, it is sold. The for loop is shown below.

```
for i = 1:nDays
     for j = 1:nbStocks
        if signal2(i,j) == 1 && signal3(i,j) == 1
            signal1(i,j) = 1;
        elseif signal2(i,j) == -1 && signal3(i,j) == -1
            signal1(i,j) = -1;
        end
     end
  end

     signal = signal1;

   for c = 2:nDays
       for d = 1:nbStocks
           if signal1(c-1,d) == 1 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == 0
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d)  == -1 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == -1
               signal(c,d) = -1;
           end
       end
```

```
    end


end
```

This concludes the generation of the matrices for each of the strategies. The next point that differentiates each strategy is the optimization of the portfolios. As previously mentioned, this optimization is not a mandatory process of the portfolio management. What must also be understood is that such optimization is an expected optimization, there is no way to know in advance if the Sharpe Ratio will be truly greater or not. It is very easy to fall into the idea that there is a way to always maximize the Sharpe Ratio but the reality is that this is not the case. In this work, strategies have been tested by optimizing the portfolio and without optimizing it, sometimes the optimization works well and sometimes not as much, as it is expected.

The results obtained by backtesting the two strategies mentioned above will be shown below. In the momentum strategy, associated with the SignalCreation function, the portfolio has not been optimized. While in the momentum strategy with low beta it has been optimized.

Firstly, the results obtained for the momentum strategy during the In-Sample period are presented. The equity curve obtained for each of the markets tested is displayed below.

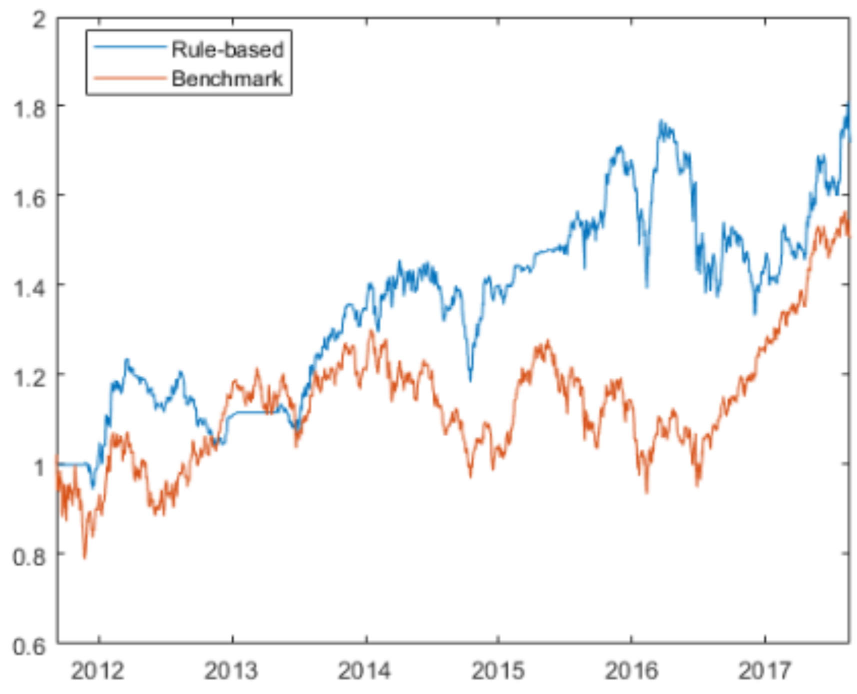


Figure 39: Equity curve – Momentum Strategy – In Sample – ATX

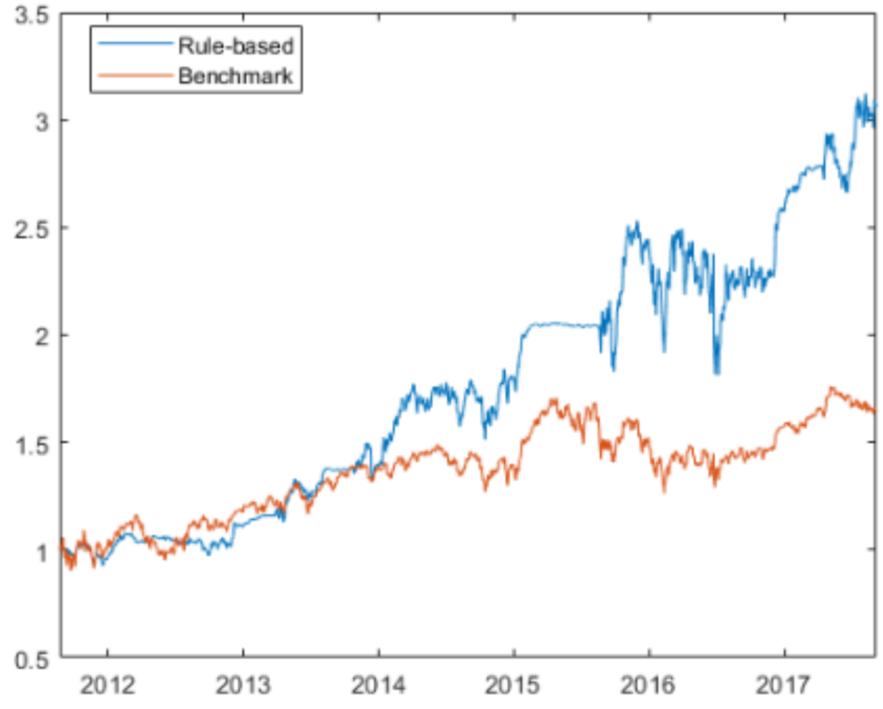


Figure 40: Equity curve – Momentum Strategy – In Sample – CAC 40

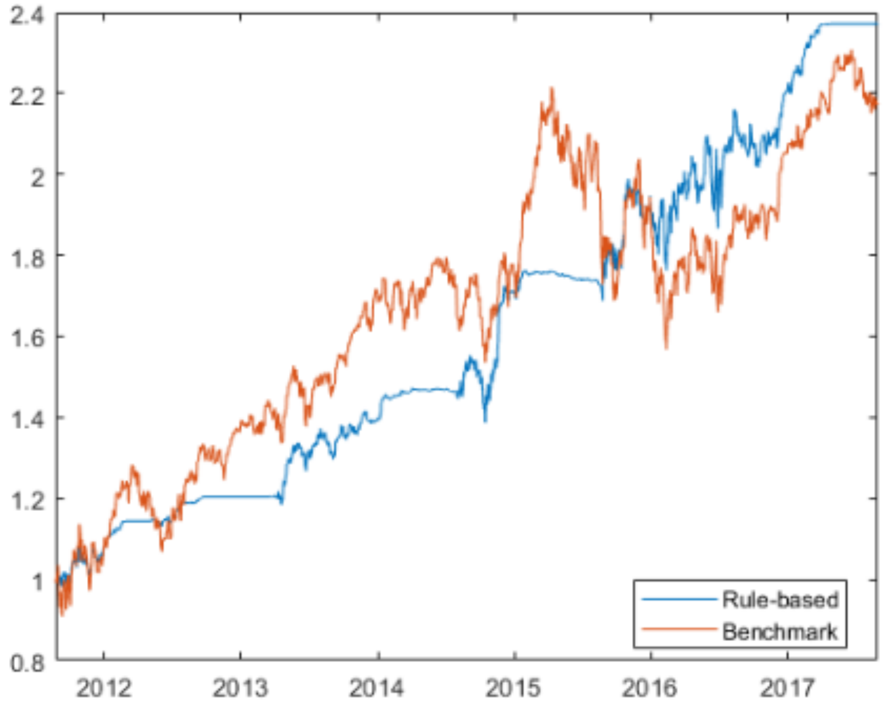


Figure 41: Equity curve – Momentum Strategy – In Sample – DAX 30

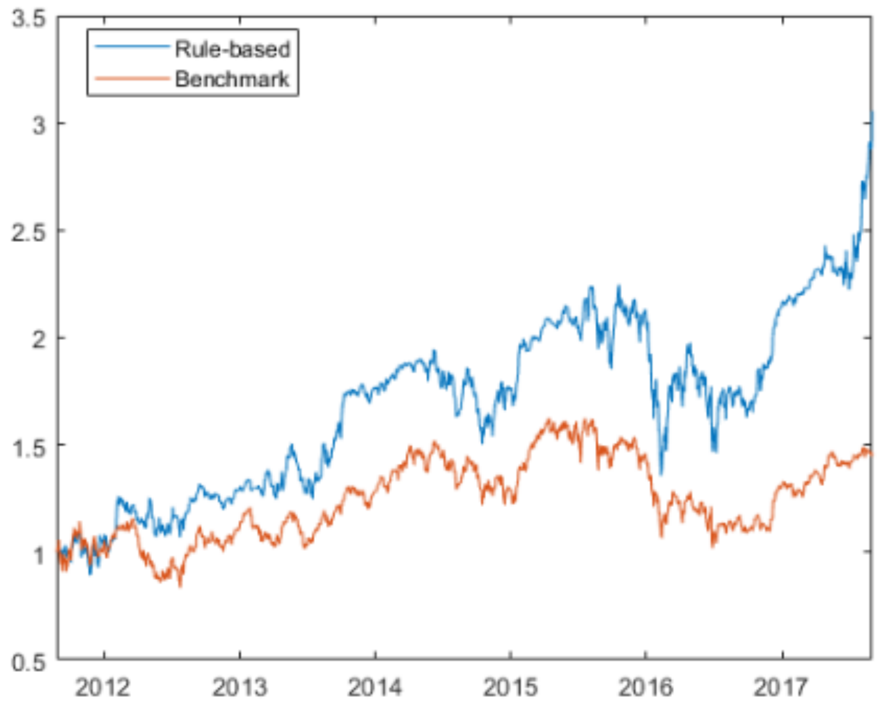


Figure 42: Equity curve – Momentum Strategy – In Sample – FTSE MIB

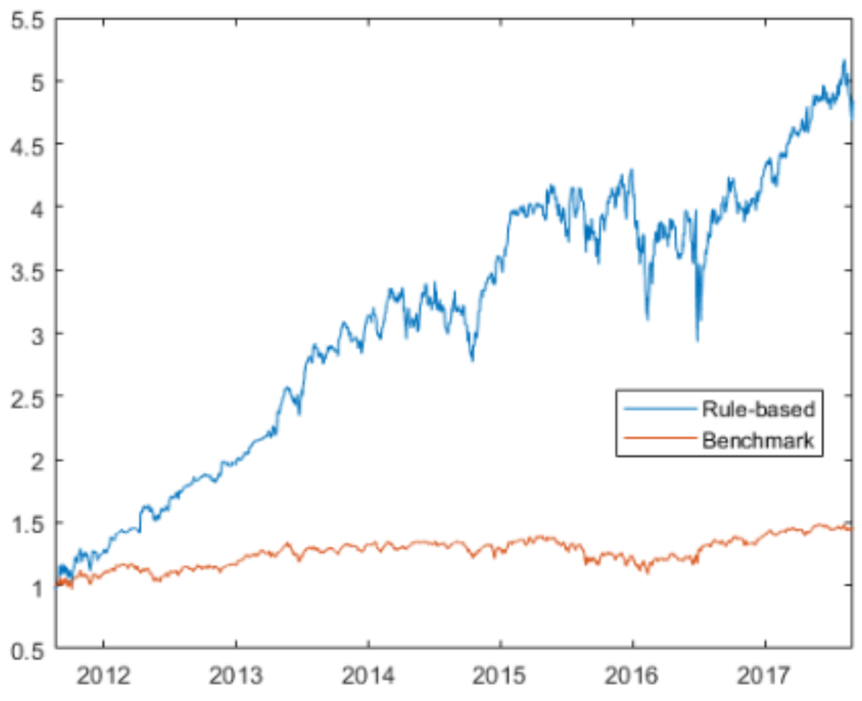


Figure 43: Equity curve – Momentum Strategy – In Sample – FTSE 100

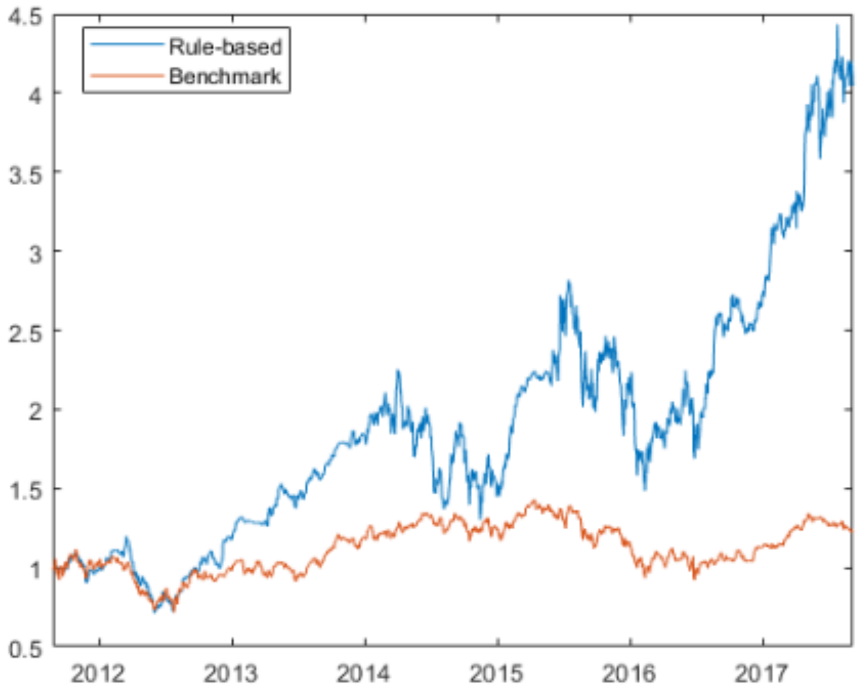


Figure 44: Equity curve – Momentum Strategy – In Sample – Mercado Continuo

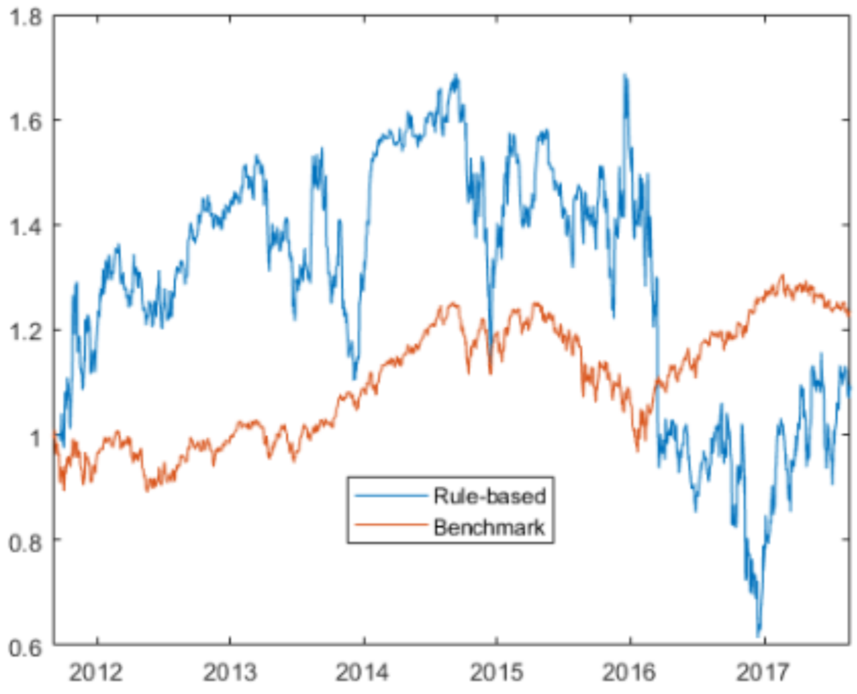


Figure 45: Equity curve – Momentum Strategy – In Sample – S&P TSX 60

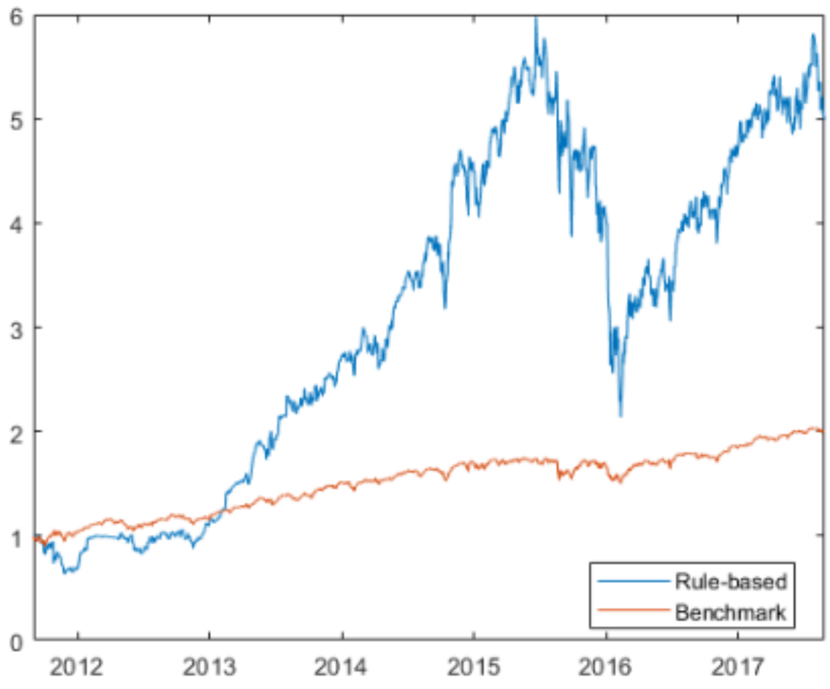


Figure 46: Equity curve – Momentum Strategy – In Sample – S&P 500

Below is a table compiling the different performance indicators calculated for each of the markets.

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
ATX	0,6651	17.196.400	24,74%	28,29%	5,42e-03%	0,5%	1,08%
CAC 40	0,9207	30.734.000	28,48%	26,04%	4,22e-02%	3,6%	1,17%
DAX 30	1,2017	23.681.000	11,35%	29,27%	8.1e-04%	8,9e-02%	0,91%
FTSE MIB	0,8128	30.552.000	39,65%	37,15%	4,94e-02%	4,19%	1,18%
FTSE 100	1,1811	48.384.000	31,91%	22.05%	8,54e-02%	8,33%	1,02%
Mercado Continuo	0,8239	40.434.000	47,29%	37,75%	9,17e-02%	5,14%	1,79%
S&P TSX 60	0,22	10.855.400	63,63%	22,86%	1,49e-02%	0,74%	2,01%
S&P 500	0,8953	49.992.200	64,20%	14,16%	8,63e-02%	4,45%	1,94%

Table 1: Compilation of Performance indicators – Momentum Strategy – In Sample

From the analysis of the different figures and the summary table, several things can be deduced about the momentum strategy. First of all and in absolute terms, i.e. without comparing with the benchmark, the strategy has managed to make money in all cases. However, in some cases it is clear from the charts and the maximum drawdown in particular that the strategy can have a very high volatility, so it may not be appropriate for a person with a high risk aversion. In the case of the S&P 500, it can be seen that in mid-2015 the shares in the portfolio suffered a very significant drop, causing the value of the portfolio to fall by up to 64%. This is well above the benchmark's maximum value of 14%. However, at the end of the period the value of the portfolio is much higher than a passive investor in the S&P 500 would have obtained. Only in the case of the S&P TSX 60 the strategy failed to outperform the index, in all other cases the strategy provides better results than its benchmark.

Secondly, the results obtained by backtesting the momentum with low beta strategy during the same period, i.e. the In Sample period are analyzed. Once again the analysis begins by showing the equity curves of the different markets.

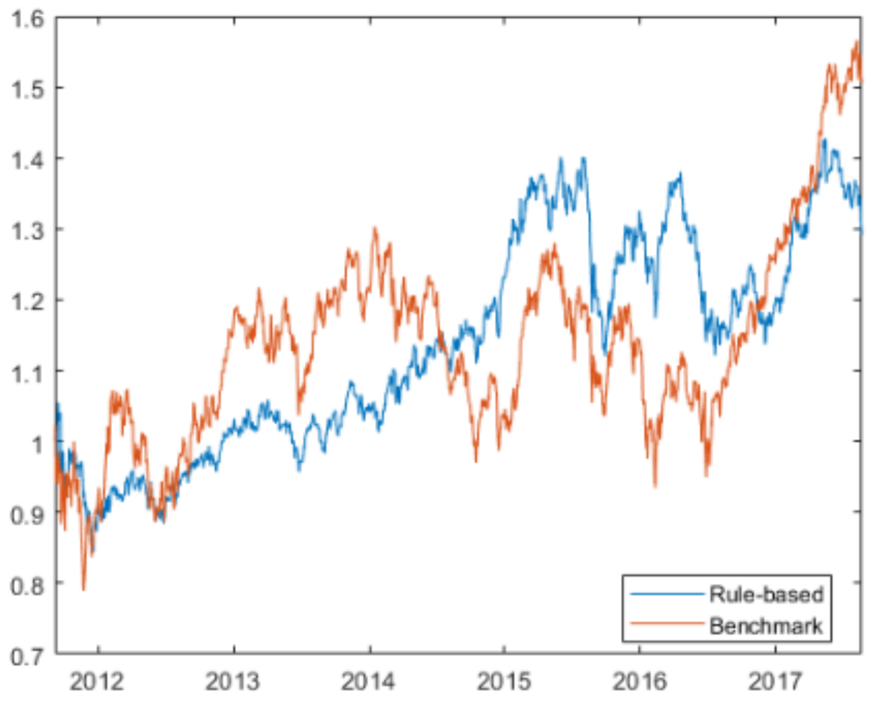


Figure 47: Equity curve – Momentum Low Beta Strategy – In Sample – ATX

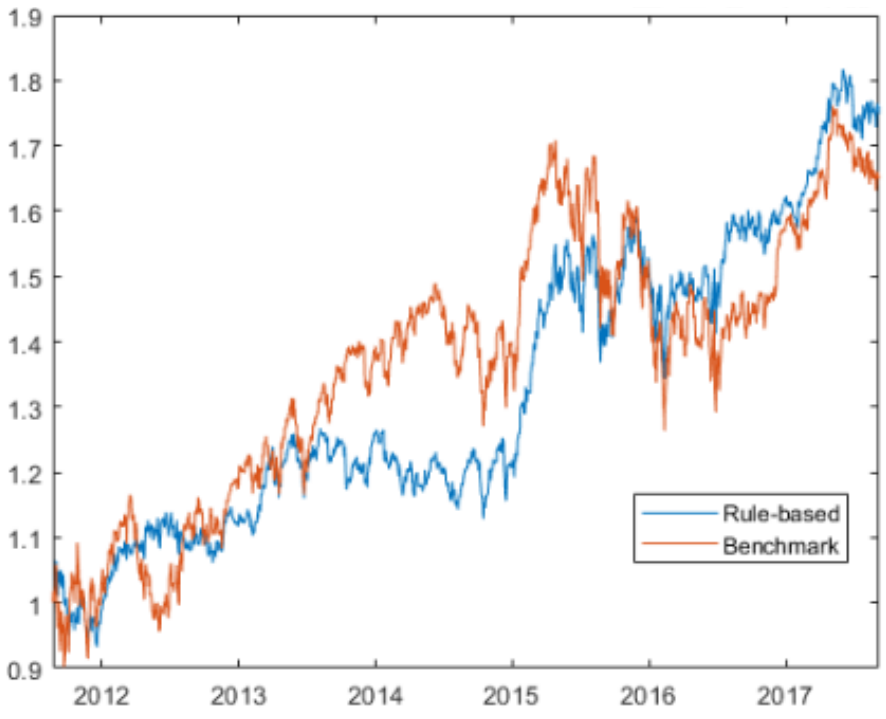


Figure 48: Equity curve – Momentum Low Beta Strategy – In Sample – CAC 40

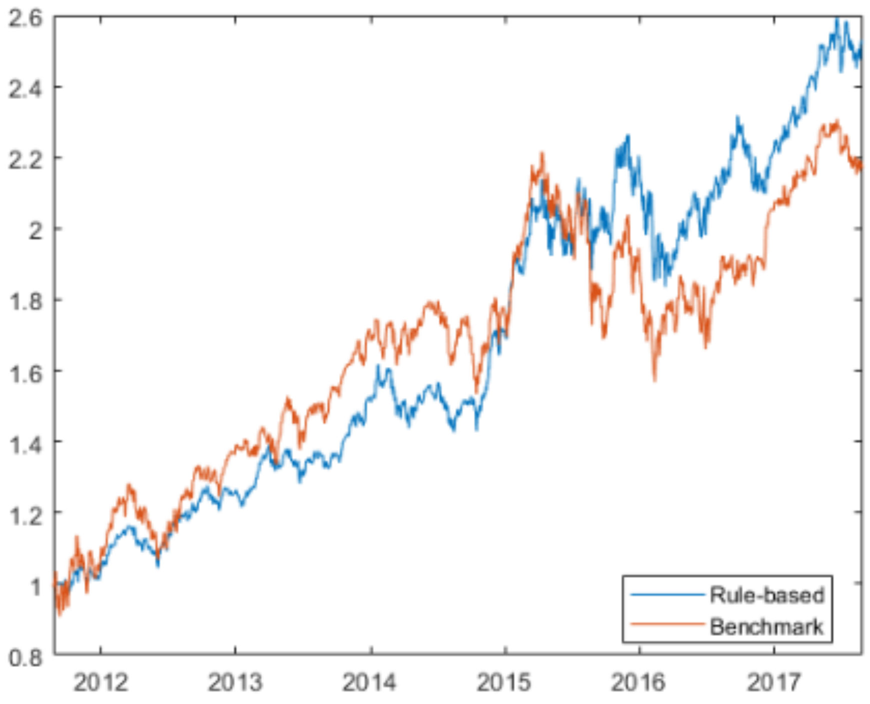


Figure 49: Equity curve – Momentum Low Beta Strategy – In Sample – DAX 30

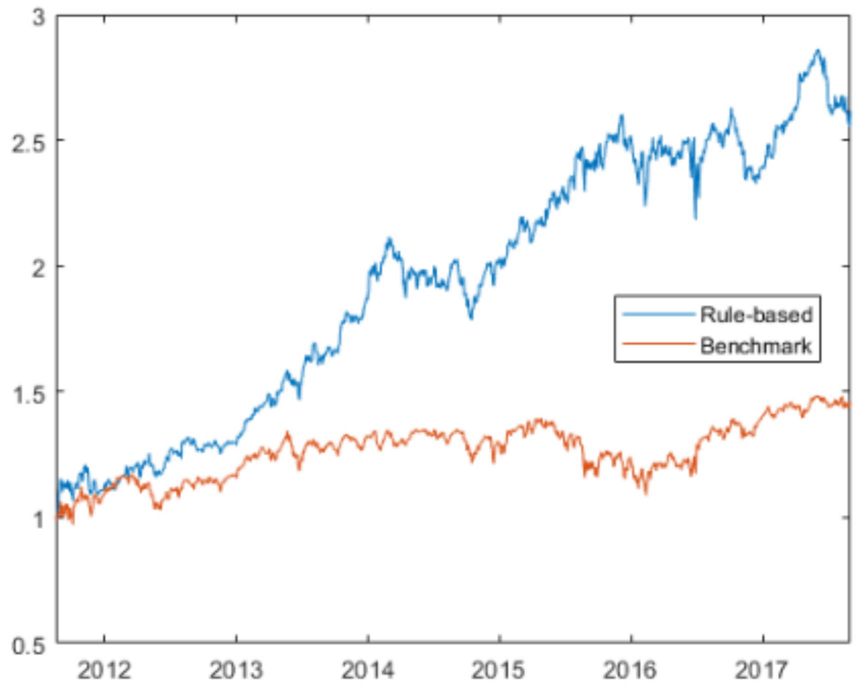


Figure 50: Equity curve – Momentum Low Beta Strategy – In Sample – FTSE 100

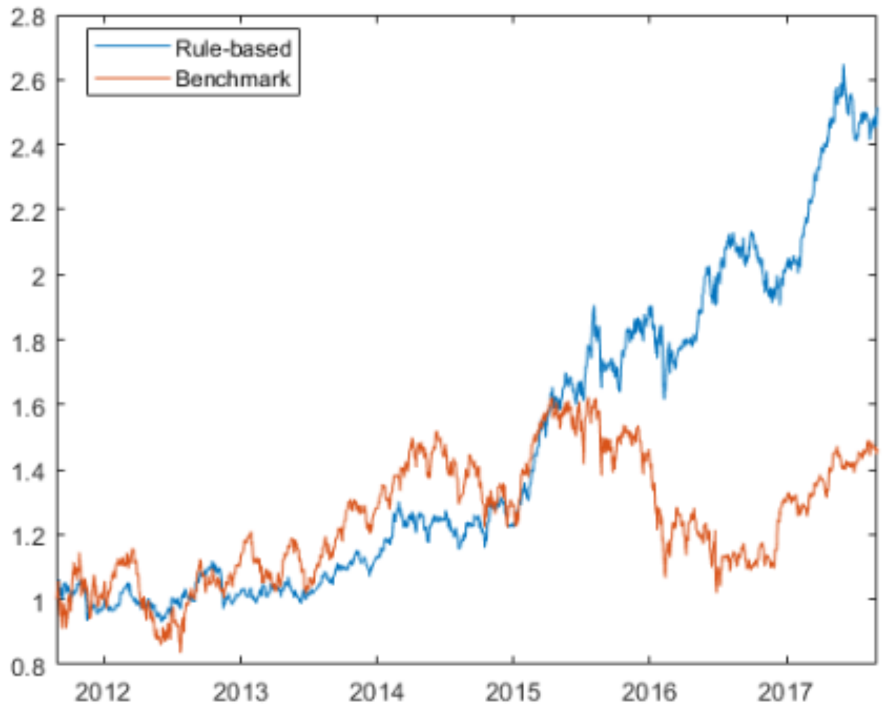


Figure 51: Equity curve – Momentum Low Beta Strategy – In Sample – FTSE MIB

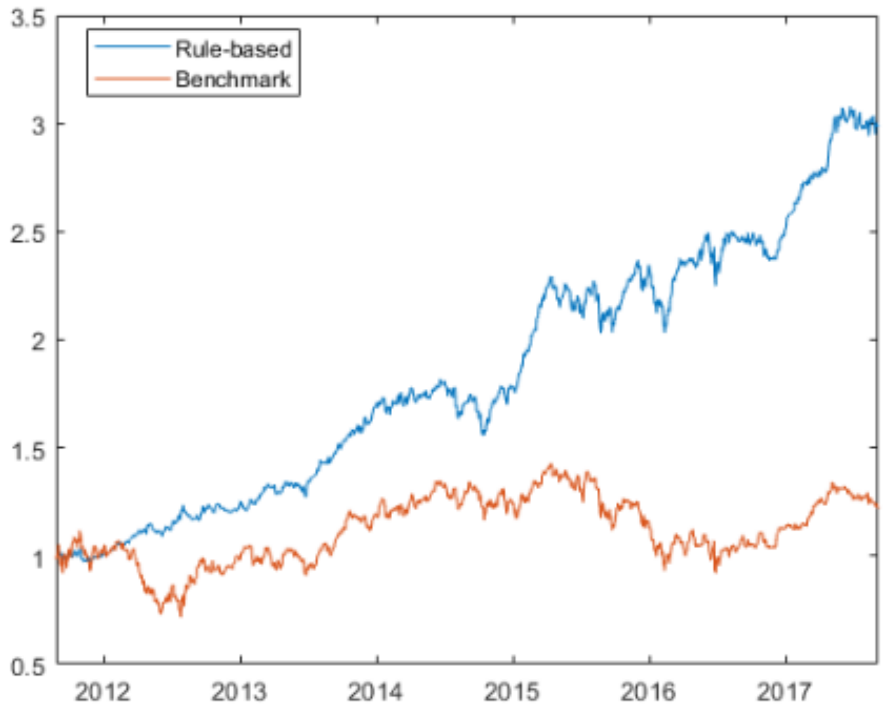


Figure 52: Equity curve – Momentum Low Beta Strategy – In Sample – Mercado Continuo

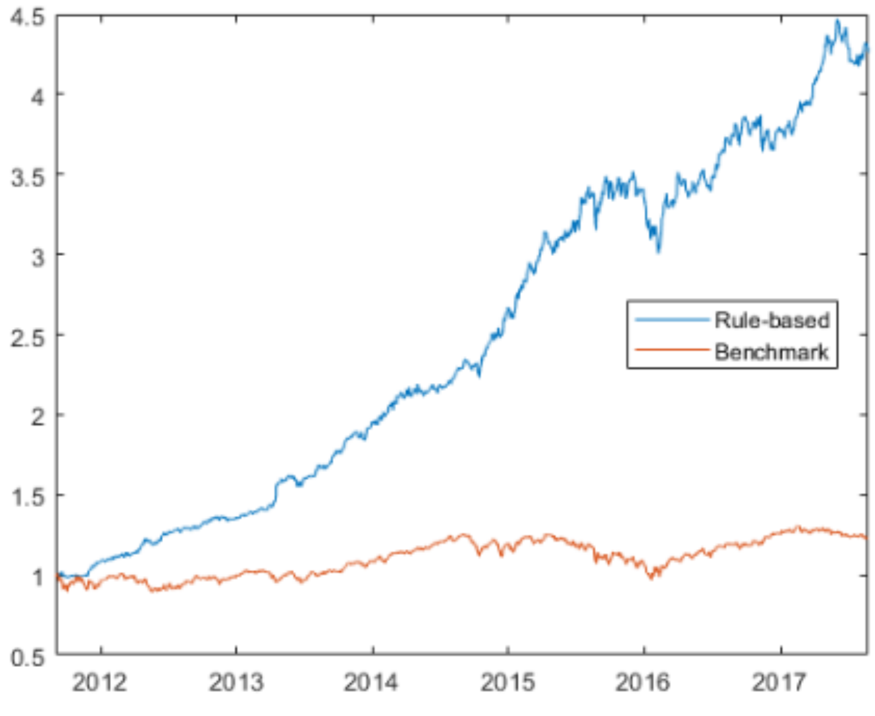


Figure 53: Equity curve – Momentum Low Beta Strategy – In Sample – S&P TSX 60

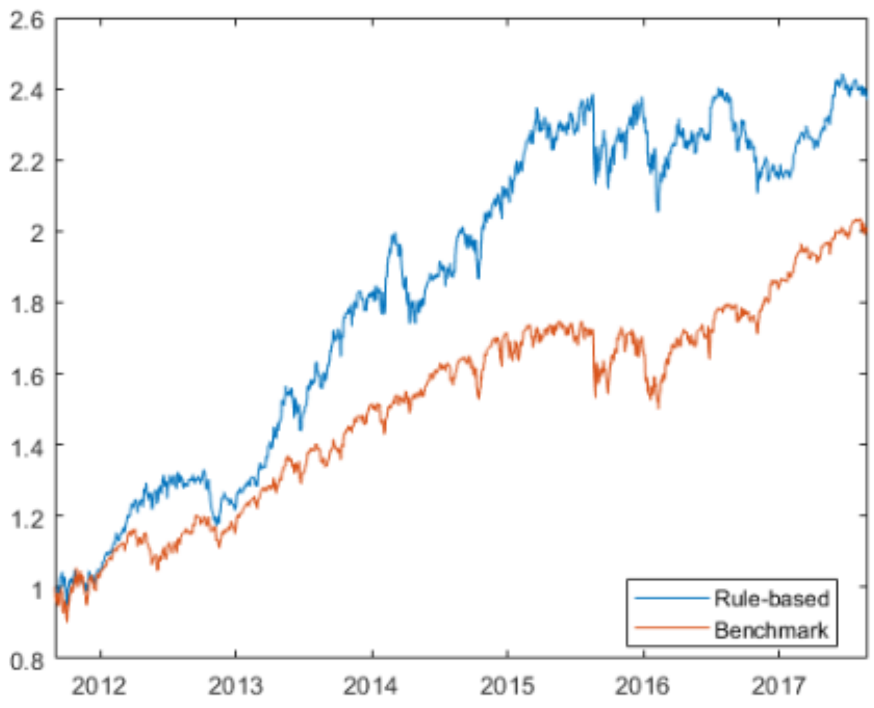


Figure 54: Equity curve – Momentum Low Beta Strategy – In Sample – S&P 500

Below is the table that compiles the different performance indicators calculated for each of the markets.

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
ATX	0,3699	12.905.200	20,08%	28,29%	7,72e-03%	-1,30%	1,12%
CAC 40	0,6985	17.481.900	15,57%	26,04%	1,83e-02%	-0,03%	0,84%
DAX 30	1,0334	25.235.000	18,90%	29,27%	3.3e-02%	0,7%	0,91%
FTSE MIB	0,9445	25.136.000	15,22%	37,15%	5,21e-02%	2,04%	1,38%
FTSE 100	1,1192	26.137.000	16,05%	22.06%	4,78e-02%	4,82%	0,79%
Mercado Continuo	1,5081	46.971.000	14,25%	37,75%	6,65e-02%	4,45%	1,14%
S&P TSX 60	2,2415	42.581.400	14,63%	22,86%	9,17e-02%	10,81%	0,76%
S&P 500	1,0731	23.663.000	13,94%	14,16%	2,11e-02%	1,88%	0,61%

Table 2: Compilation of Performance indicators – Momentum Low Beta Strategy – In Sample

Firstly, simply by looking at the different equity curves it can be concluded that this strategy has a very low volatility, the variations of the curve are relatively small.

They are much less important than those obtained with the previous strategy and most importantly, less than those that would have been obtained if the investor had invested passively in the benchmark. In all cases, with the exception of the ATX, the strategy achieved to outperform the benchmark at the end of the period. In most cases the outperformance is also very significant and during a large part of the period. Looking now at the summary table, the most important element that can be seen is that the strategy's maximum drawdown has been significantly reduced with respect to the previous strategy. The maximum drawdown is even much lower compared to the reference index in all cases. As for the Sharpe ratios, which show the risk-adjusted return of the strategy on some occasions has decreased and on others has increased, maintaining in any case good values in the majority of cases. The rest of the indicators are also mixed in comparison with the previous strategy, that is, on some occasions they are better and on others worse. In particular as regards the total money obtained at the end of the period, in more than half of the cases it has been lower than the previous strategy, in the rest it has ended up with more.

It seems therefore that with this strategy the results continue to be very good and the most important thing is that it continues to outperform the benchmark as well as doing so with less volatility and less risk of permanent loss of capital than investing in the index.

Because during the Out of Sample period, in the case of strategies belonging to the rules based strategies set, there is no change from the In Sample period in terms of how the strategy works. It has been decided to show only the result of the Out of Sample period for the momentum strategy with low beta as it has been considered more interesting and in this way this paper will not be saturate with figures. The figures for the Out of Sample period of the previous strategy will be available in the annex.

Thereafter, the results of the backtest during the Out of Sample period for the momentum strategy with low beta are shown below.

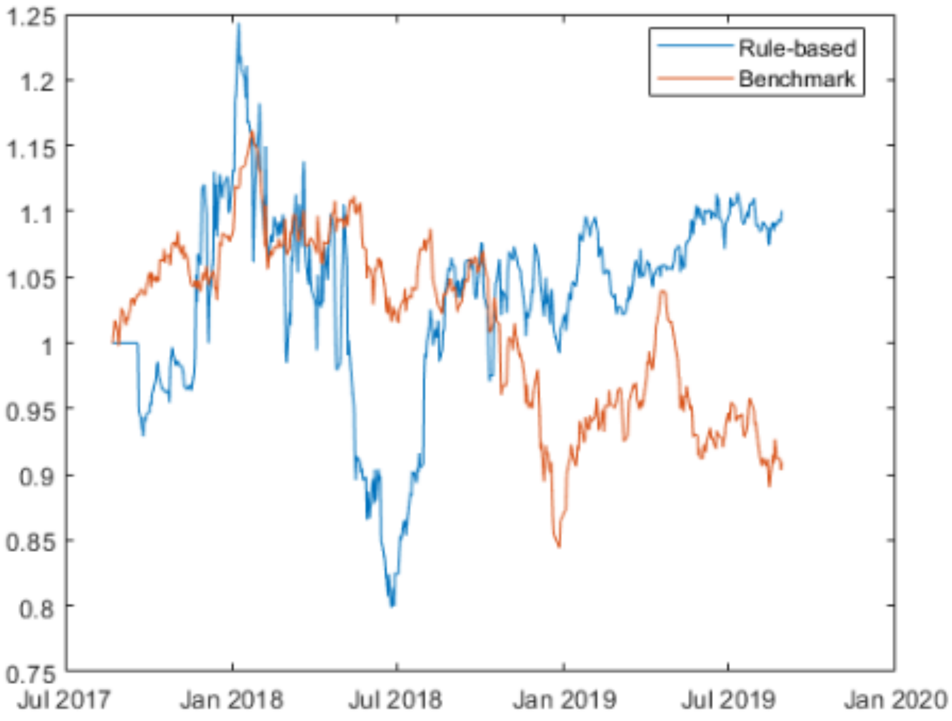


Figure 55: Equity curve – Momentum Low Beta Strategy – Out of Sample – ATX

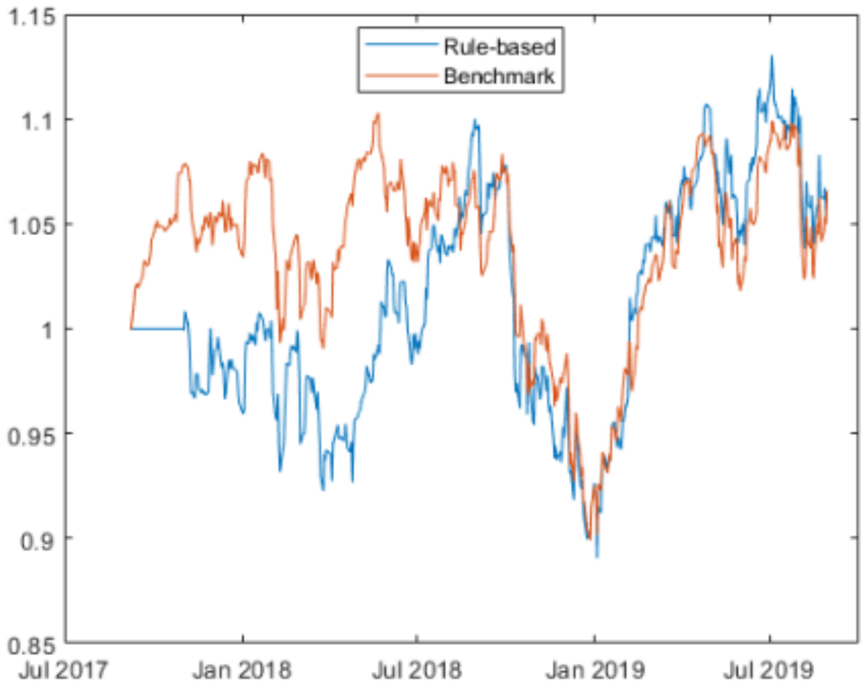


Figure 56: Equity curve – Momentum Low Beta Strategy – Out of Sample – CAC 40

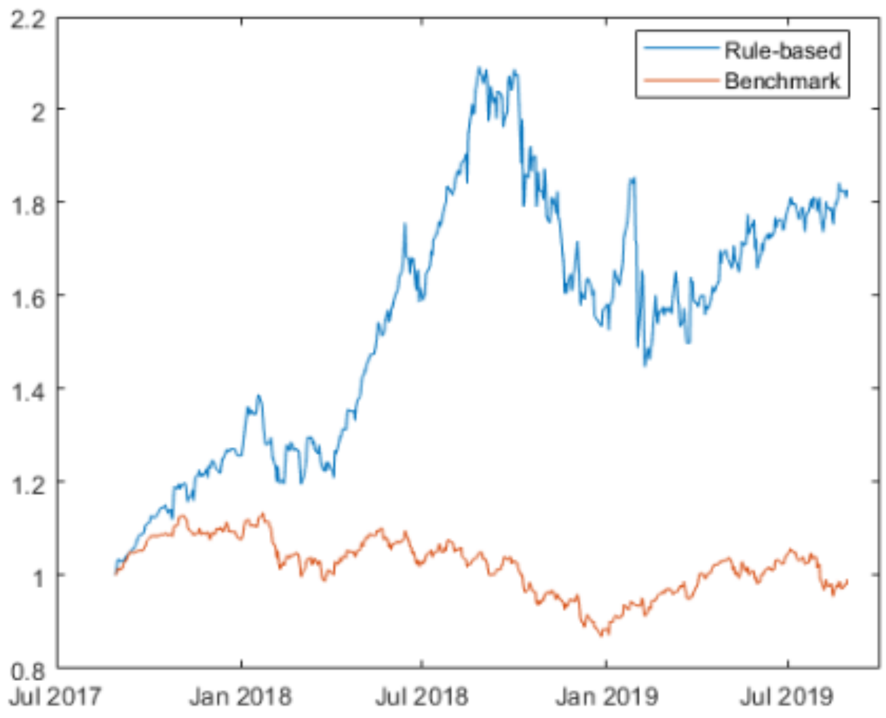


Figure 57: Equity curve – Momentum Low Beta Strategy – Out of Sample – DAX 30

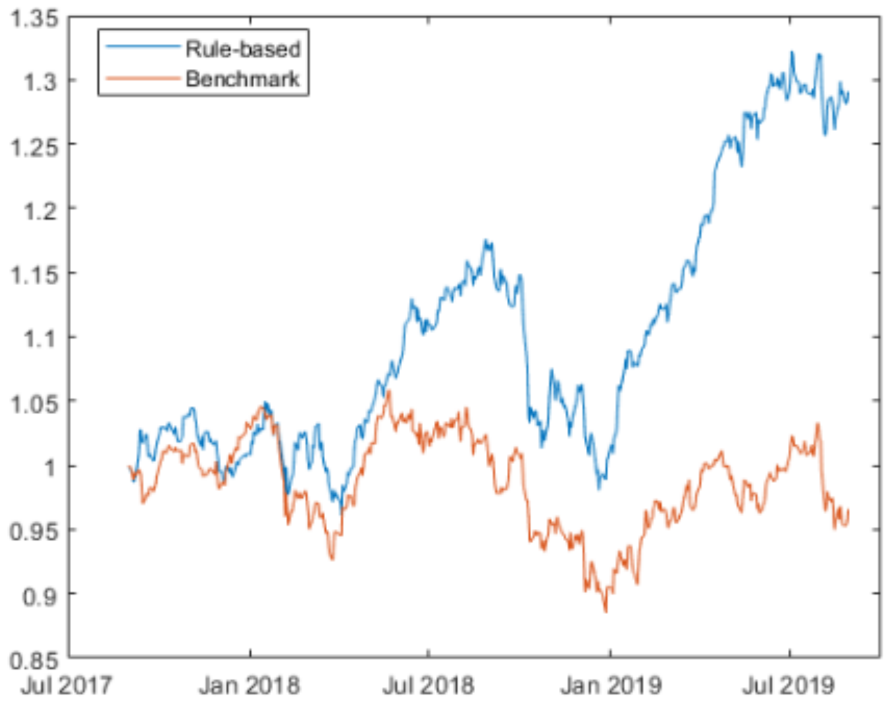


Figure 58: Equity curve – Momentum Low Beta Strategy – Out of Sample – FTSE 100

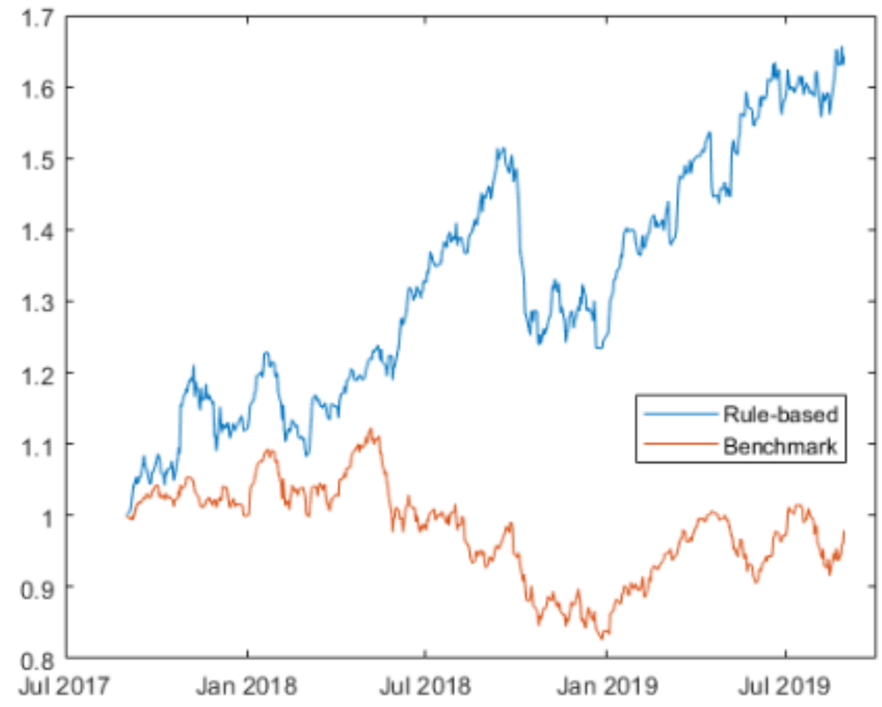


Figure 59: Equity curve – Momentum Low Beta Strategy – Out of Sample – FTSE MIB

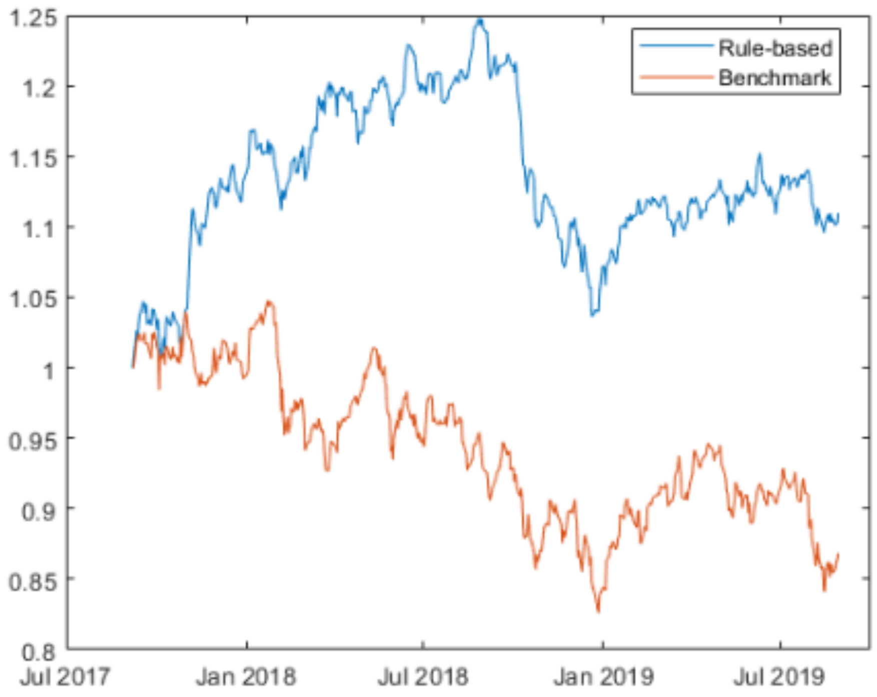


Figure 60: Equity curve – Momentum Low Beta Strategy – Out of Sample – Mercado Continuo

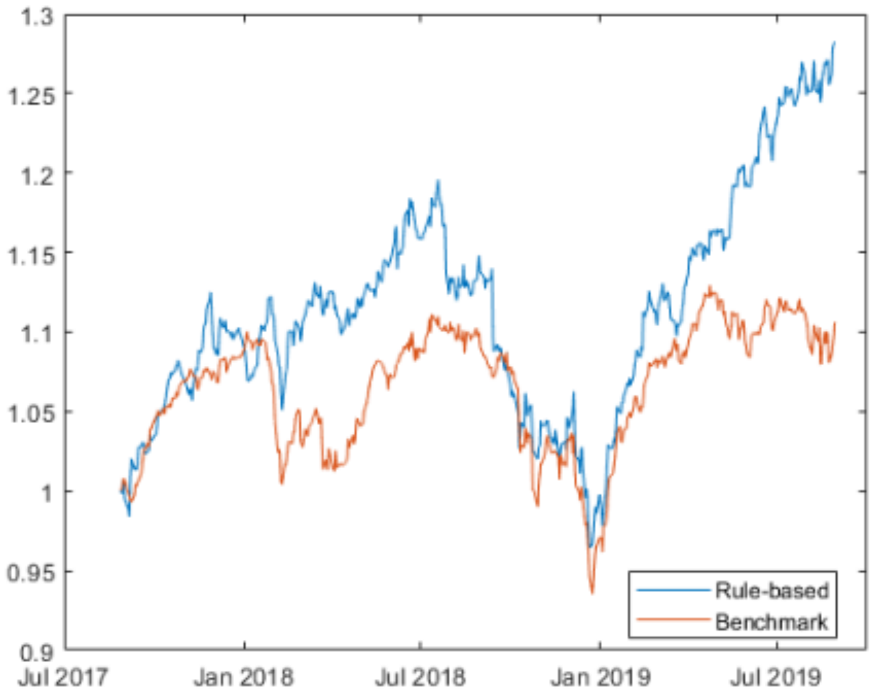


Figure 61: Equity curve – Momentum Low Beta Strategy – Out of Sample – S&P TSX 60

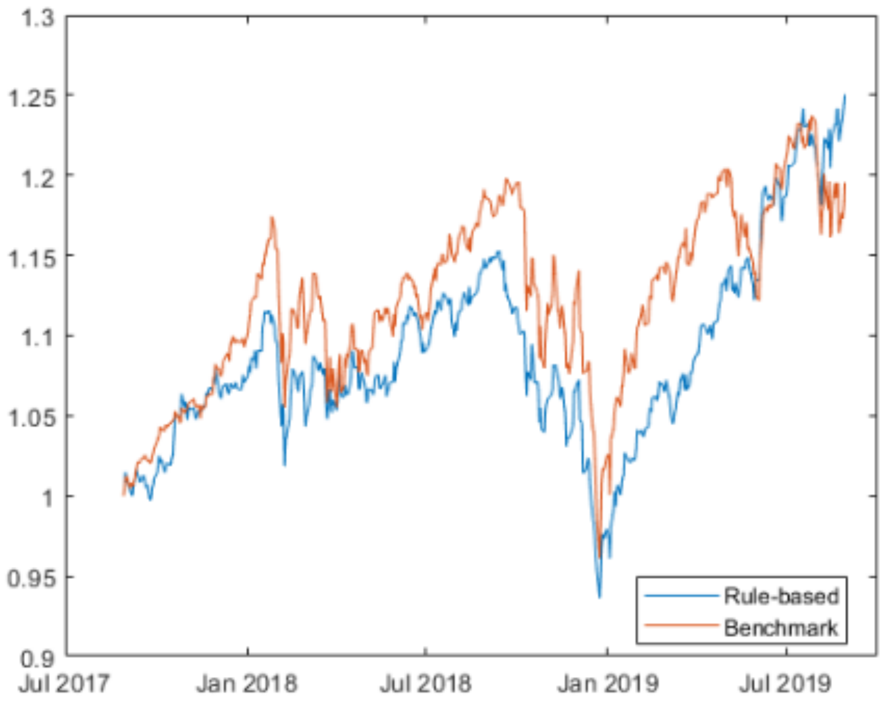


Figure 62: Equity curve – Momentum Low Beta Strategy – Out of Sample – S&P 500

Once again, a summary table is shown below with the different indicators obtained,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error

ATX	0,3111	11.005.300	35,76%	27,31%	4,60e-02%	3,03%	1,62%
CAC 40	0,2762	10.617.070	19,03%	18,47%	2,59e-03%	4,37e-02%	0,63%
DAX 30	1,1573	18.268.500	30,80%	23,44%	0,14%	8,58%	1,58%
FTSE MIB	1,4398	16.412.500	18,53%	26,40%	0,1%	9,47%	1,10%
FTSE 100	1,1128	12.898.500	16,62%	16.41%	5,66e-02%	9,40%	0,62%
Mercado Continuo	0,5493	11.084.800	16,93%	21,17%	3,34e-02%	6,51%	0,73%
S&P TSX 60	1,1620	12.812.900	19,38%	15,80%	3,70e-02%	4,96%	0,61%
S&P 500	1,0128	12.493.200	18,82%	19,78%	2,04e-02%	1,47%	0,50%

Table 3: Compilation of Performance indicators – Momentum Low Beta Strategy – Out of Sample

First, as can be seen in all the charts when looking at the evolution of the different benchmark indexes, the financial markets experienced a difficult time during these two years, from July 2017 to August 2019. In particular, in December 2018 all financial markets around the world fell considerably.

Despite these facts, the first conclusion that can be drawn is that in all cases the strategy manages to make money, and succeeds in beating the benchmark performance at the end of the period and once again in some cases it does so in a significant way and for a large part of the period analyzed. When looking at the table of indicators it can be seen that despite the bad period, good Sharpe ratios are achieved in most cases. As for the maximum drawdown in this case, values greater than those of the reference index were obtained in the majority of cases. All of them during the month of December 2018. However, these values are not particularly bad considering that this is a significant fall in all the financial markets of the world.

We can therefore finally conclude that two strategies are obtained that manage to consistently beat their respective benchmarks. The first of these is based exclusively on momentum and without optimization of the portfolio and achieves very good results in the long term although it has a high volatility so an investor who wants to carry out this strategy must have a low aversion to risk and be prepared for very important fluctuations both upwards and downwards in the value of his stock portfolio.

The second strategy, based on momentum with low beta, portfolio optimization and monthly rebalancing, also manages to consistently beat its benchmark indices and with a much lower level of volatility, making it more appropriate for most investors.

In the following chapter, strategies based on machine learning are studied.

4.2.2 Strategies based on Artificial Intelligence algorithms

The following chapters will explain in more detail the nature of the investment strategies based on artificial intelligence algorithms developed in this work.

The main idea for each of the strategies is to train the model only on the set of data with the prices and indicators that belong to the benchmark index. This model will then be applied to predict the evolution of the shares that make up this index. For example, the training of the algorithm is made based on the indicators and predictors that belong to the DAX 30 benchmark index. Then this model is used to predict the returns of each stock that compose the DAX, individually.

In this way, a general and robust model is obtained, independent of each company and which can be applied to any company in a country, even to a stock that has just been incorporated into the index. Without the need to retrain or update the model.

The reason why a model is made for each index and not a general one for all the indexes is that there are certain idiosyncrasies in each index resulting in a higher correlation between the shares that make up the index than with shares in other indexes. As there are not too many developed market indices, a manageable number of models are obtained.

4.2.2.1 Machine Learning: Naive – Bayes

The first of the strategies based on an artificial intelligence algorithm is detailed below. To develop this strategy, the statistics and machine learning toolbox has been used. As in the strategies detailed in the previous chapter, it is a systematic strategy. The strategy is based on the function, SignalCreationML(), which provides a matrix with the Buy and Sell signals for each share. In this way the algorithm based on Machine Learning is responsible for making investment decisions, and not the human investor.

The first step, prior to the creation of the matrix with the signals, is the training of the Machine Learning algorithm using the training data set. This is the balanced data set with the response variable for the reference index. For the training of Machine Learning models, MATLAB has a tool that makes the process very intuitive, using the Classification Learner app. The advantage of this app is that instead of having to write the code to do the training, all the steps are done through an interface that allows to test the effect of all the possible parameters and analyze the training results directly through the App. Once the results are satisfactory the model is simply exported and a new set of data can be applied to it to check its validity. This App has a large number of different models, from decision trees, support vector machines, nearest neighbor classifiers, and in particular the one that has been chosen in this work to develop the algorithms for each index, the Naive-Bayes Classifier.

It is a probabilistic classifier, i.e. it is used to identify to which subset of categories a given sample in a database belongs, and as its name indicates it is based on Bayes' Theorem. One of the characteristics of this algorithm is that it assumes independence between the predictors, in this case this means that there would be independence between the 11 predictors used, the temporal data, the returns, the RSI 14 and the MACD. While it is difficult to ensure full independence, particularly in this case as the data relate to the daily closing prices of the reference index, the predictors do have some significant degree of independence. The value of the RSI, whether high or low, does not influence the value of the MACD, nor the returns, nor obviously the date of the sample, for example.

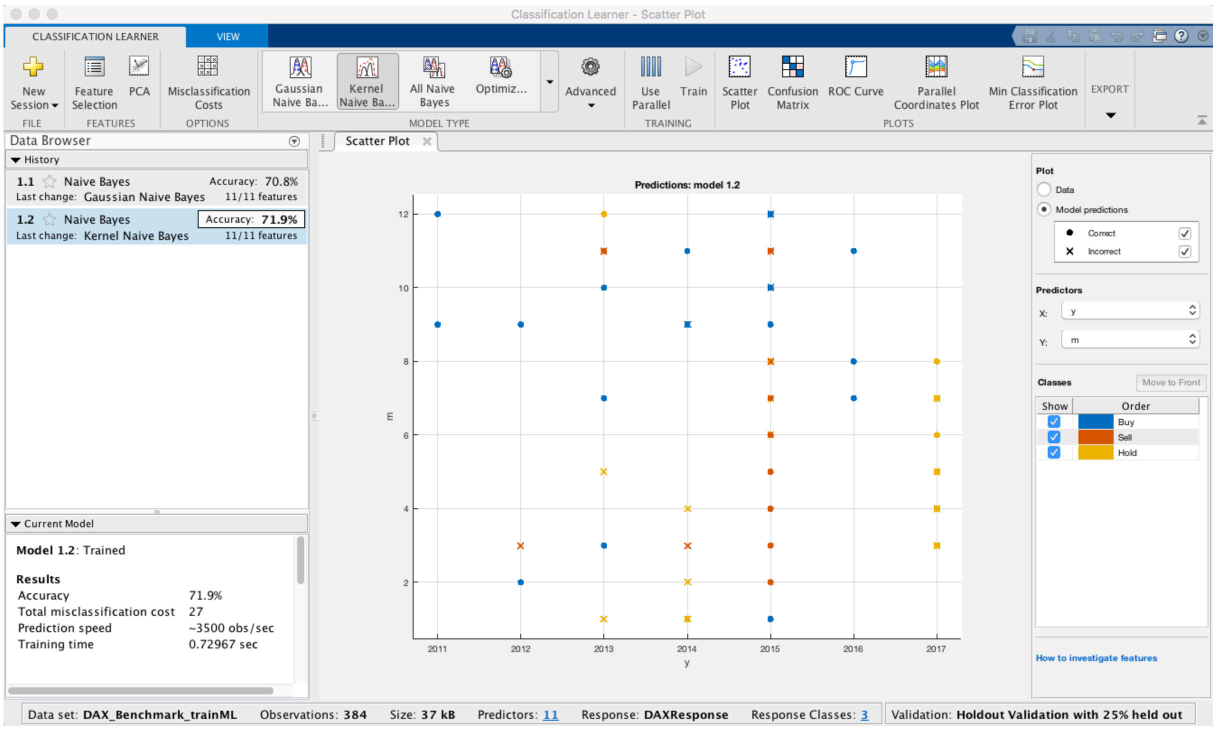
This is important as the Naive Bayes classifier considers that each of the predictors contributes independently to the probability that the sample will provide a Buy, Sell or Hold signal. This is exactly the objective, since they are independent indicators that can be complemented and it is not intended to associate a greater importance to any of them.

The algorithm is based on the probability that the answer belongs for example to the category Buy, knowing which are the values of the predictors, The result is determined by following Bayes' theorem as shown below,

$$P(Buy/A) = \frac{P(Buy) \times P(A/Buy)}{P(A)}$$

Being A the combination of the 11 predictors of that sample.

Below is an example of how the DAX data set is trained using the Classification Learner App.



1.1 Naive Bayes Accuracy: 70.8%
Last change: Gaussian Naive Bayes 11/11 features

1.2 Naive Bayes Accuracy: 71.9%
Last change: Kernel Naive Bayes 11/11 features

Model 1.2: Trained

Results

Accuracy	71.9%
Total misclassification cost	27
Prediction speed	~3500 obs/sec
Training time	0.72967 sec

Data set: DAX_Benchmark_trainML Observations: 384 Size: 37 kB Predictors: 11 Response: DAXResponse Response Classes: 3 Validation: Holdout Validation with 25% held out

Figure 63: Classification Learner App – DAX 30

As shown in the figure, the model had an accuracy of 71.9%. Of the 384 total samples contained in the training set, the algorithm has failed to predict 27 of them. For the validation of the model, the Holdout Validation method is used, which separates the set in two independent groups, one for training and another for validation. In this case the validation sub-set represents 25% of the total set. This process is carried out to have an additional protection against the overfitting of the algorithm to the training data, which represents the most important danger when working with prediction tools based on artificial intelligence algorithms. In this case a more than acceptable level of accuracy is obtained so the next step is to export the model so that it can be used to predict a new set of data and thus create the corresponding signal matrix.

For this purpose, the SignalCreationML() function has been created as mentioned above. Again, a modular structure is followed, in which the function is called independently for each index as shown below,

Create trading signals and backtest - In-sample

We create the signal matrix.

S&P 500

```
index = 'SP500';
SP500_nDay = height(SP500_inSample);
[SP500_signalinSampleML] = SignalCreationML(SP500_inSample,
SP500_nStocks, SP500_nDay, index, trainedModel_SP500);
```

ATX

```
index = 'ATX';
ATX_nDay = height(ATX_inSample);
[ATX_signalinSampleML] = SignalCreationML(ATX_inSample, ATX_nStocks,
ATX_nDay, index, trainedModel_ATX);
```

The following details the SignalCreationML function. First of all it is important to say that this strategy generates the signals only based on the outcome of the Naive Bayes Classification Model. There is no additional filter, in particular based on a Low Beta classification as is the case with the other artificial intelligence strategies developed in this work.

The first step is to remove the indicator with the beta from the matrix with the predictors as it will not be one of them nor will it be useful at any point in the strategy.

Creation of the Signal Matrix

We obtain a list for each in sample day, if signal = 1, we buy, if signal = -1, we sell, if signal = 0, we hold.

Based on the output of the previously trained Machine Learning model.

```
function [signal] = SignalCreationML(Xin, nbStocks, nDays, c,
trainedModel)


   % We want to build one matrix for each of the stocks, based on the
   % results predicted by the model based on Machine Learning for each
   % index.

   hasbeta = ~cellfun('isempty', regexp(Xin.Properties.VariableNames,
'Beta', 'once'));
   Xin(:,Xin.Properties.VariableNames(hasbeta)) = [];
```

Then, using a for loop, a data structure of the type cell is generated. Each cell represents a matrix with the 11 indicators for each of the stocks, as shown below. The result is therefore a structure with the same number of cells as the number of stocks that compose the index. Each of those cells with the predictors for that stock.

```
Temp_Pred = Xin(:,1:6);

   XinStocks = cell(1, nbStocks);
   yPredInSample = cell(1, nbStocks);

   for i = 1: nbStocks
       ColToKeep = 1;
       ColToDelete = nbStocks;
       Predictors = Xin(:,7:end-nbStocks);
       ColIndex = mod(0:size(Predictors,2)-1, ColToKeep+ColToDelete-1)
== i-1;
       Predictors(:,~ColIndex) = [];
       XinStocks{i} = [Temp_Pred Predictors];
    end
```

The trainedModel object, which is exported from the Classification Learner App, is then called using another for loop. Simply by using the function .predictFcn, included in the trainedModel variable, and passing the new data set as an argument, a variable is generated with the outputs calculated by the algorithm. It is therefore a very simple process, the only point in which it is necessary to be careful is to ensure that the new data set contains the predictors in the correct format, expected by the .predictFcn function, otherwise it will not work.

```
for j = 1:nbStocks
               Variables = trainedModel.RequiredVariables;
               XinStocks{j} =
XinStocks{j}(:,sort(XinStocks{j}.Properties.VariableNames));
               XinStocks{j}.Properties.VariableNames = Variables;
               yPredInSample{j} =
trainedModel.predictFcn(XinStocks{j});
end
```

The output calculated by the model is a categorical variable formed by three possible cases, Buy, Sell and Hold. Then, this categorical variable is transformed into a numeric one, associating the value 1 to the Buy category, -1 to the Sell category and 0 to the Hold category.

```
yPredinSample = [yPredInSample{:}];
signal1 = yPredinSample;
signal1 = 1*(yPredinSample == 'Buy') - 1*(yPredinSample == 'Sell');
```

Finally, by means of the following for loop, just like the one used in the strategy based on momentum and Low Beta, the definitive matrix is generated with the buy and sell signals that will be passed as an argument to the PortfolioManagement function.

Recalling how this for loop works, if the previous day delivered a buy signal, and the day being analyzed provides one as well, a buy signal is generated. If the previous day was a buy signal, and the day being analyzed is a hold signal, a sell signal is produced. The idea behind this signal is that when it is no longer a buy signal but a hold signal, it means that it is no longer in the top stocks in terms of the outcome of the Machine Learning algorithm so it is no longer wanted in the portfolio. In the case that the stocks goes from a buy signal to a sell signal, a sell signal is generated. If it goes from a hold signal to a buy signal, a buy signal is generated. If it goes from a hold signal to a hold signal, a hold signal is generated. If it goes from a hold signal to a sell signal, it is sold. If it switches from a sell signal to a buy signal, it is bought. If it goes from a sell signal to a hold signal, it is held, and finally if it goes from a sell signal to a sell signal, it is sold. The for loop is shown below.

```
signal = signal1;

   for c = 2:nDays
       for d = 1:nbStocks
           if signal1(c-1,d) == 1 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == 0
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d)  == -1 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == -1
               signal(c,d) = -1;
           end
       end
   end
end
```

Finally, this strategy follows the optimization of the portfolio according to the parameters seen previously. Therefore, it is updated monthly and always making the capital allocation in such a way that the portfolio is placed on the efficient border, in the point for which the maximum expected Sharpe Ratio is achieved.

The results obtained thanks to this strategy for the In Sample period are shown below,

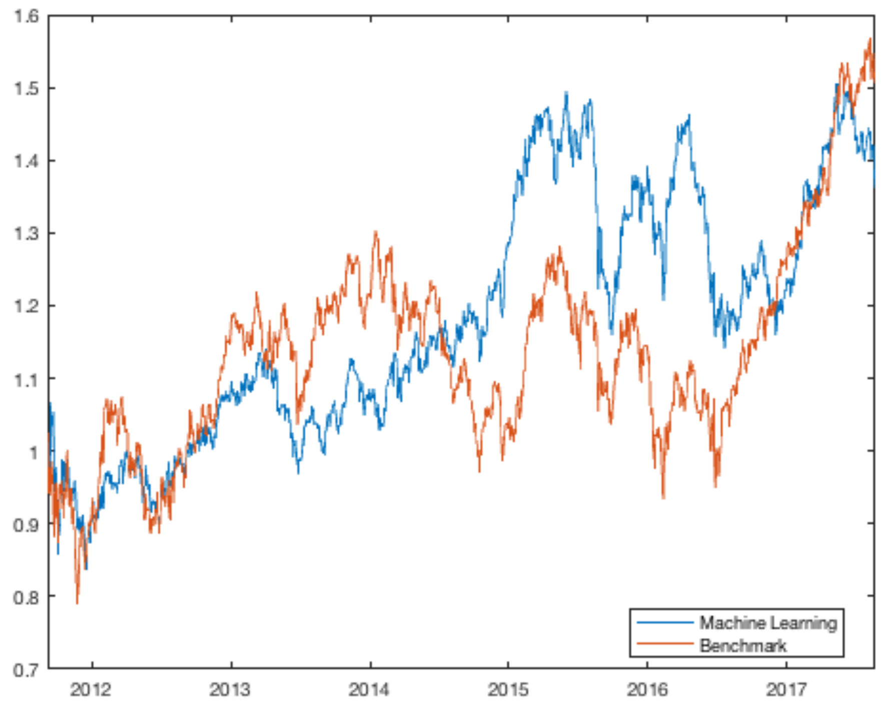


Figure 64: Equity curve – Naive Bayes Classifier Strategy – In Sample – ATX

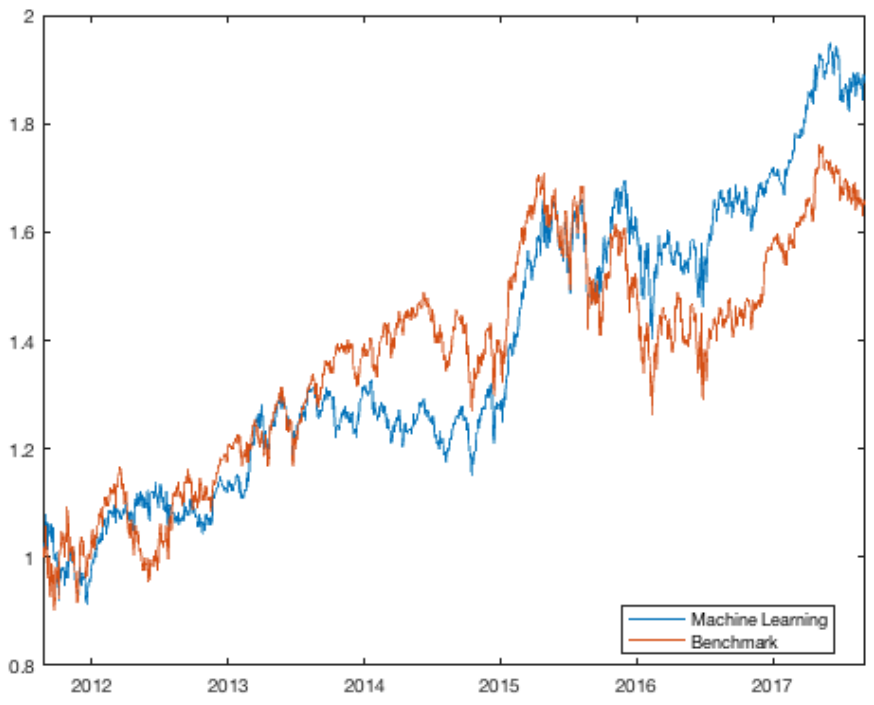


Figure 65: Equity curve – Naive Bayes Classifier Strategy – In Sample – CAC 40

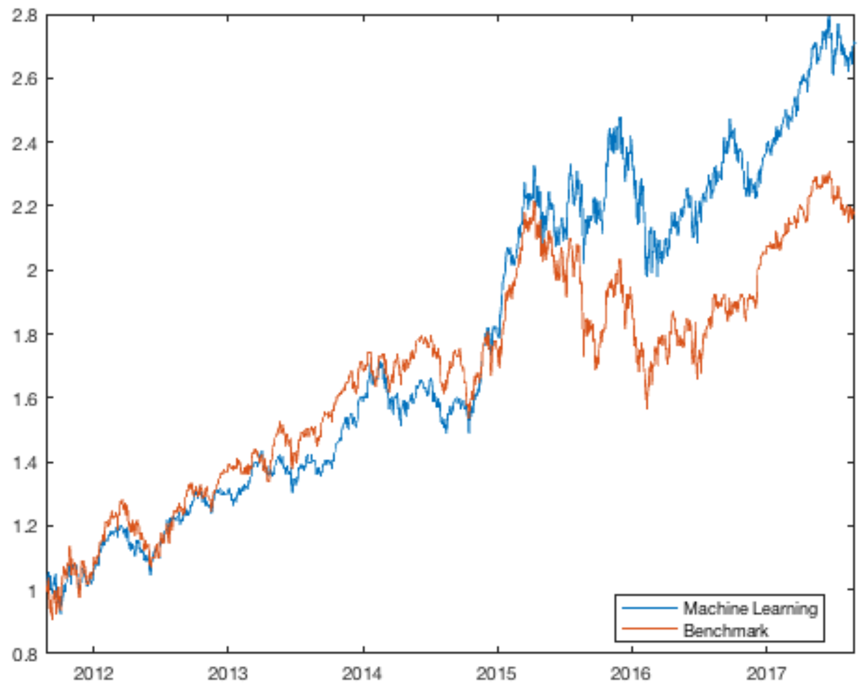


Figure 66: Equity curve – Naive Bayes Classifier Strategy – In Sample – DAX 30

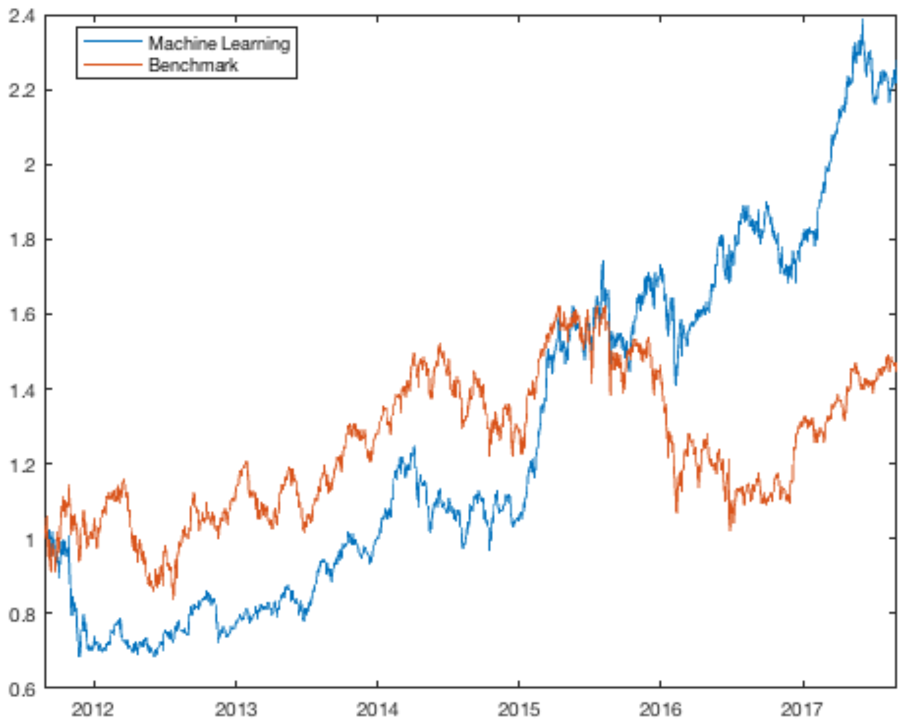


Figure 67: Equity curve – Naive Bayes Classifier Strategy – In Sample – FTSE MIB

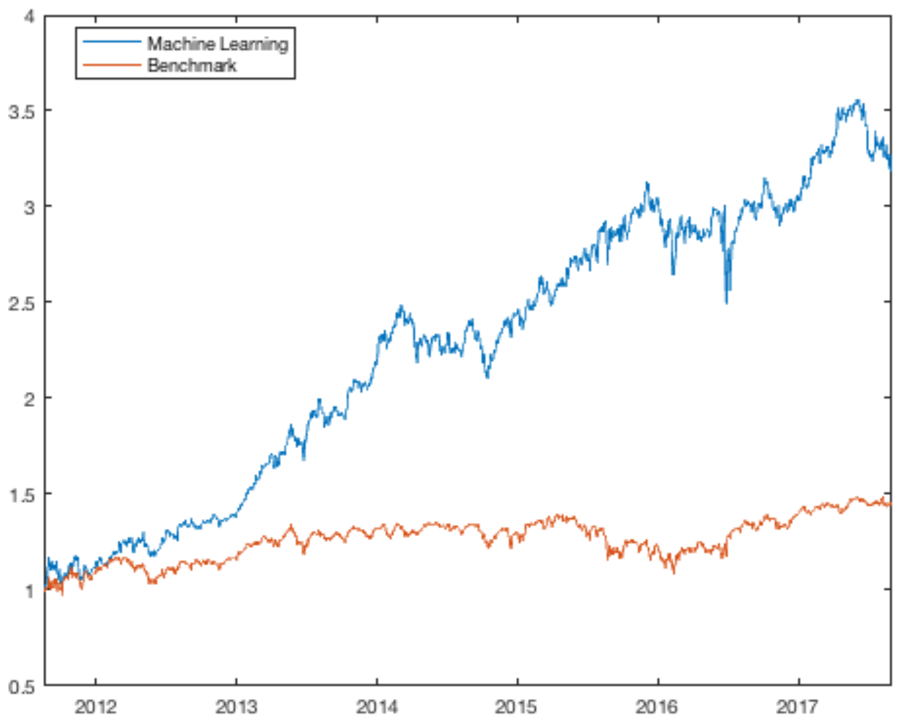


Figure 68: Equity curve – Naive Bayes Classifier Strategy – In Sample – FTSE 100

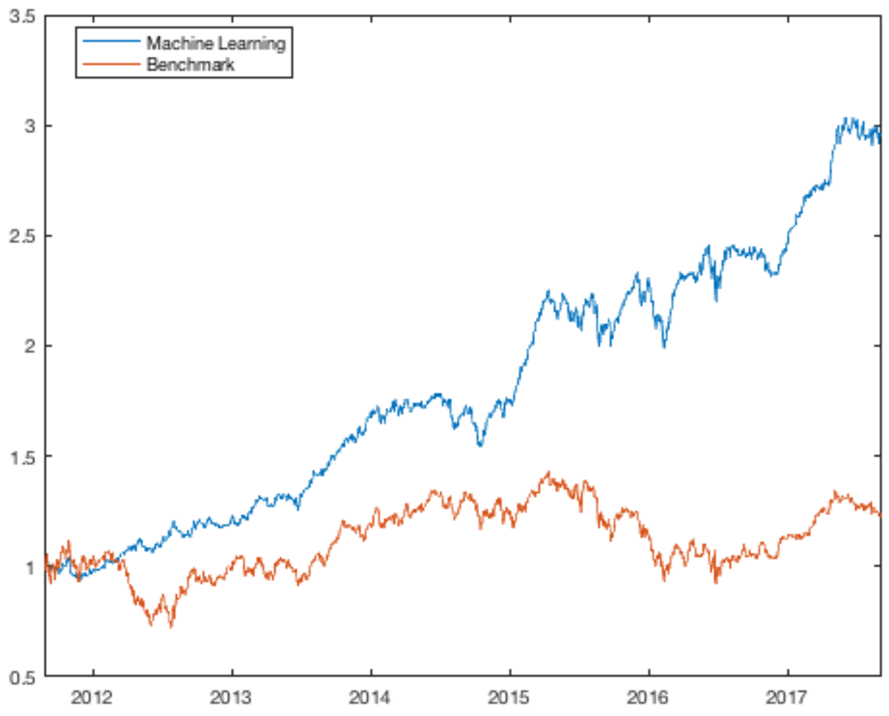


Figure 69: Equity curve – Naive Bayes Classifier Strategy – In Sample – Mercado Continuo

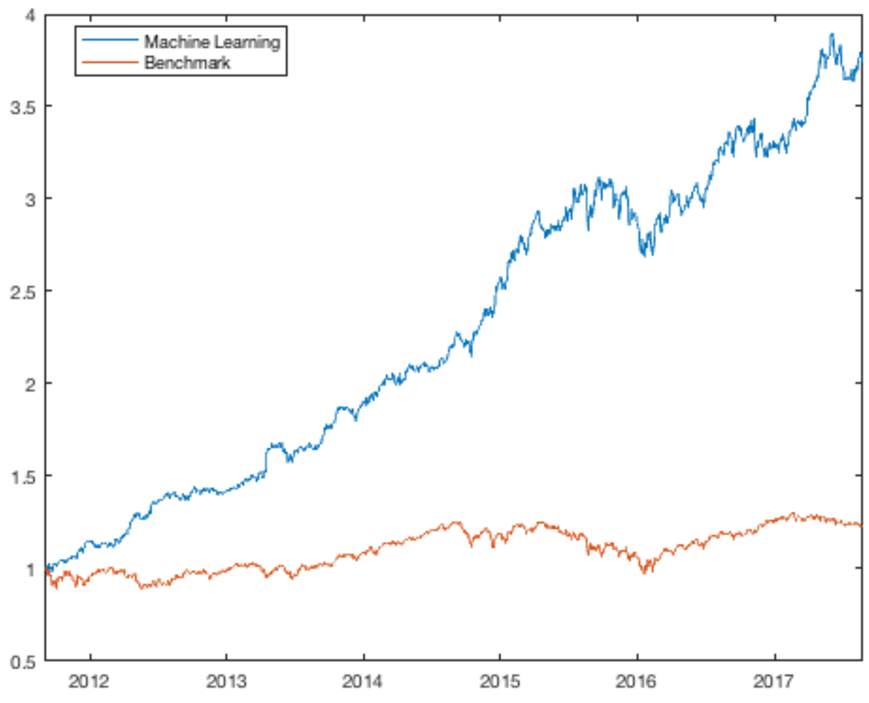


Figure 70: Equity curve – Naive Bayes Classifier Strategy – In Sample – S&P TSX 60

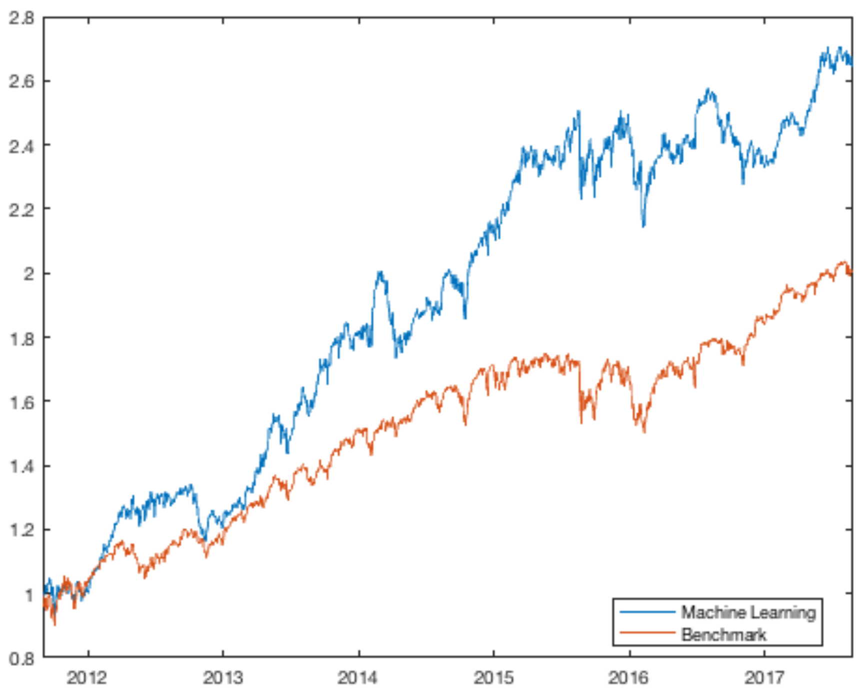


Figure 71: Equity curve – Naive Bayes Classifier Strategy – In Sample – S&P 500

The different indicators calculated are displayed below,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error

ATX	0,3842	13.590.000	23,73%	28,29%	9,00e-03%	-0,82%	1,12%
CAC 40	0,6912	18.738.300	17,39%	26,04%	2,01e-02%	1,62%	0,81%
DAX 30	0,9715	27.058.000	20,41%	29,27%	2,85e-02%	1,73%	0,78%
FTSE MIB	0,6826	21.275.600	35,32%	37,15%	4,12e-02%	1,27%	2,13%
FTSE 100	1,1573	32.756.000	20,20%	22.06%	5,96e-02%	6,40%	0,86%
Mercado Continuo	1,4398	29.552.000	14,83%	35,75%	6,54e-02%	4,50%	1,12%
S&P TSX 60	1,8748	37.267.000	13,87%	22,86%	8,17e-02%	9,90%	0,74%
S&P 500	1,1096	26.413.000	14,58%	14,16%	2,45e-02%	3,10%	0,63%

Table 4: Compilation of Performance indicators – Naive Bayes Classifier Strategy – In Sample

From the equity curves it is concluded that in all the markets analized, with the exception of the ATX, the portfolio achieves a higher value than the benchmark at the end of the period. In some cases, the outperformance is remarkable. It can also be seen that generally the curve of the strategy is smoother, therefore it seems that the strategy had less volatility than the index it is trying to beat. When analysing the table this is confirmed as in all markets without exception the maximum drawdown is lower than the index. This is very important as it means that the worst time of the strategy, the most dangerous one because it is where the investor generally makes mistakes that can be fatal to meeting his financial objectives, was better than the worst time of the benchmark index. It seems therefore a strategy with a good relationship between return and volatility. This can also be seen in the Sharpe Ratio values, which are good.

The results of the Out of Sample period are shown below, the most important period in the case of a strategy based on artificial intelligence. Since the model has been trained on certain samples of the in-sample period, it seems logical that it will perform well in this period. It is therefore crucial to confirm its validity, by confirming that it also works in a new data set, the Out of Sample set.

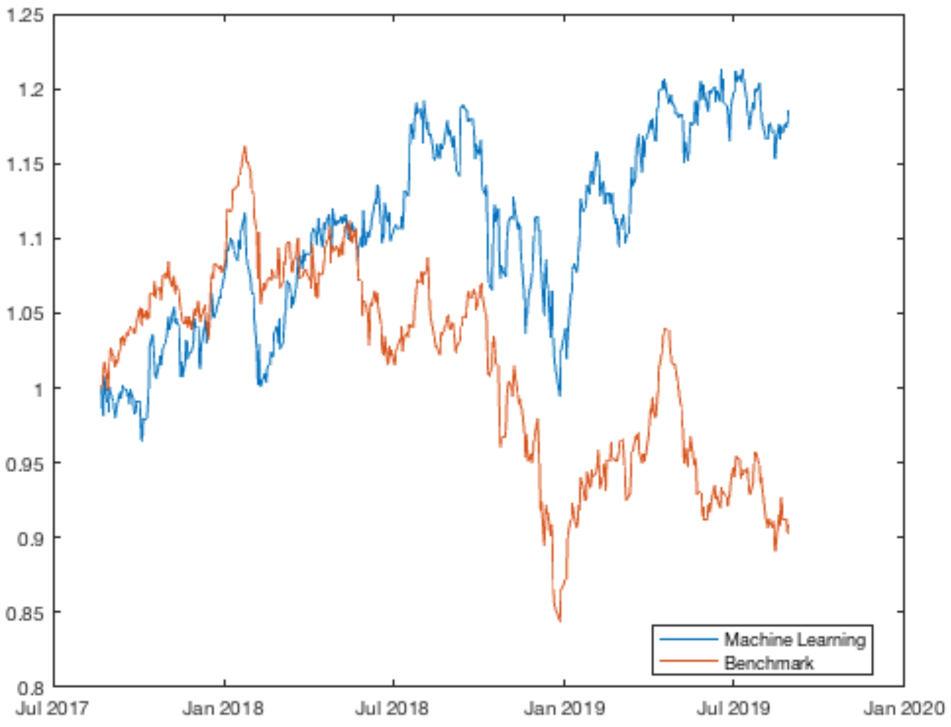


Figure 72: Equity curve – Naive Bayes Classifier Strategy – Out of Sample – ATX

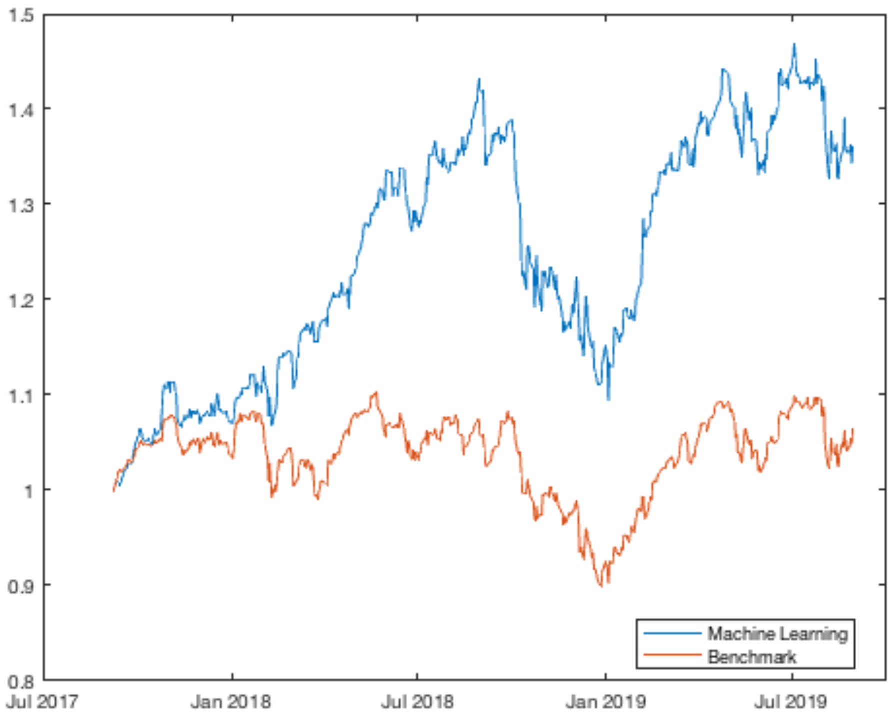


Figure 73: Equity curve – Naive Bayes Classifier Strategy – Out of Sample – CAC 40

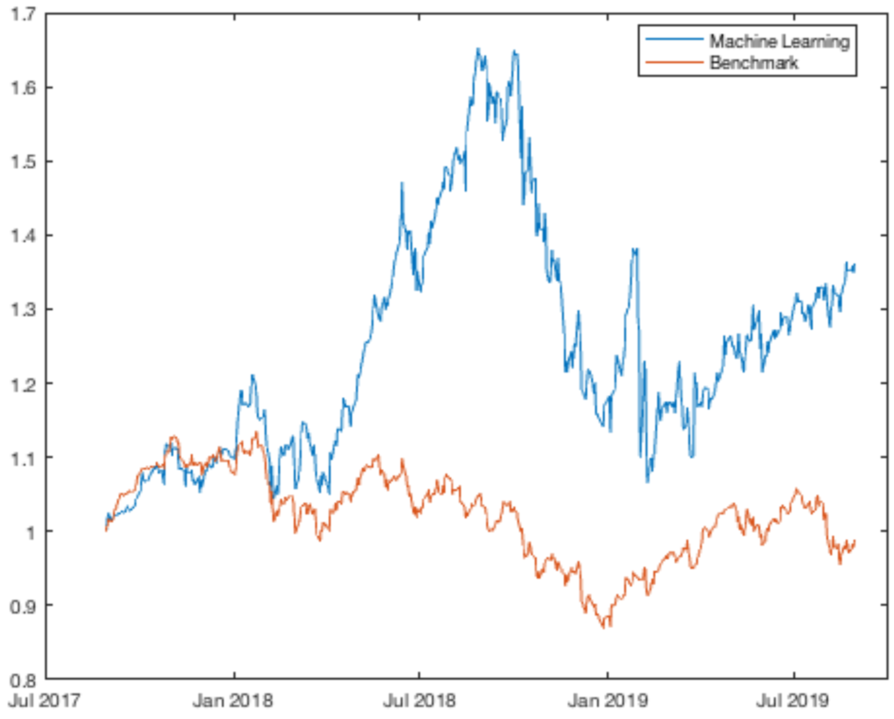


Figure 74: Equity curve – Naive Bayes Classifier Strategy – Out of Sample – DAX 30

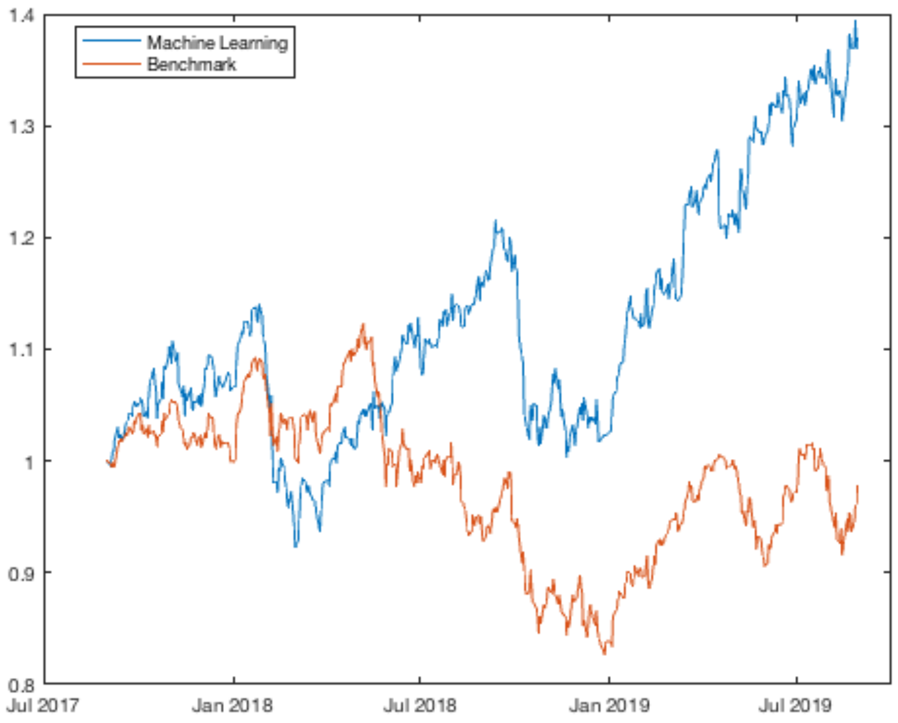


Figure 75: Equity curve – Naive Bayes Classifier Strategy – Out of Sample – FTSE MIB

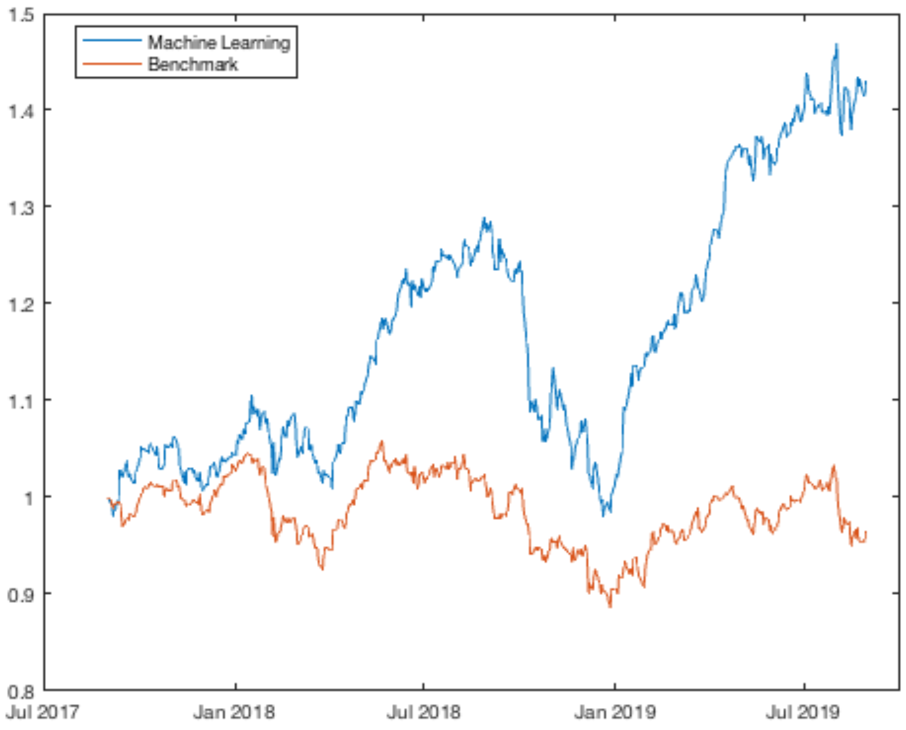


Figure 76: Equity curve – Naive Bayes Classifier Strategy – Out of Sample – FTSE 100

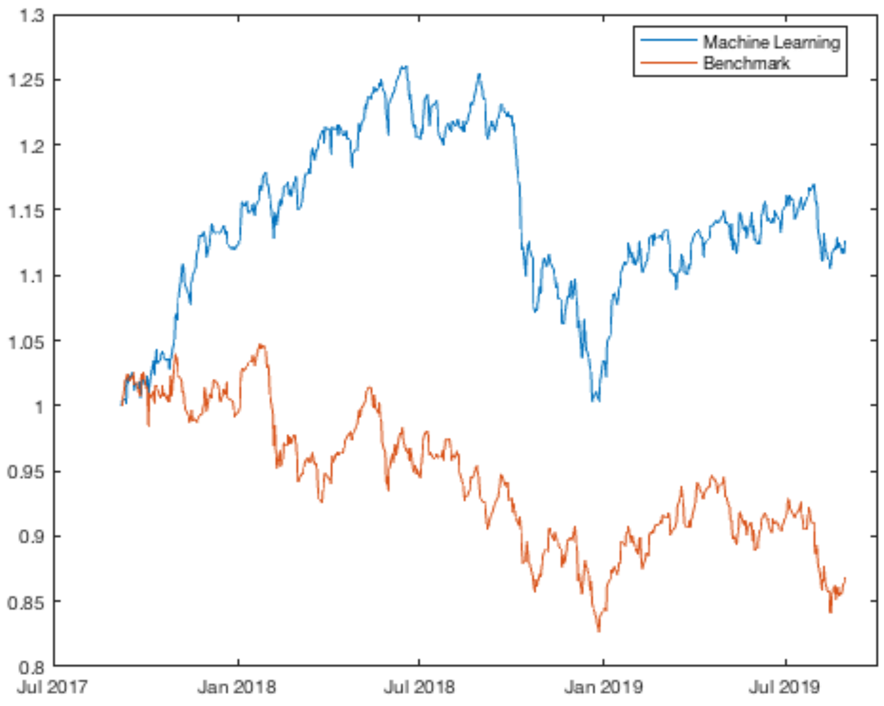


Figure 77: Equity curve – Naive Bayes Classifier Strategy – Out of Sample –Mercado Continuo

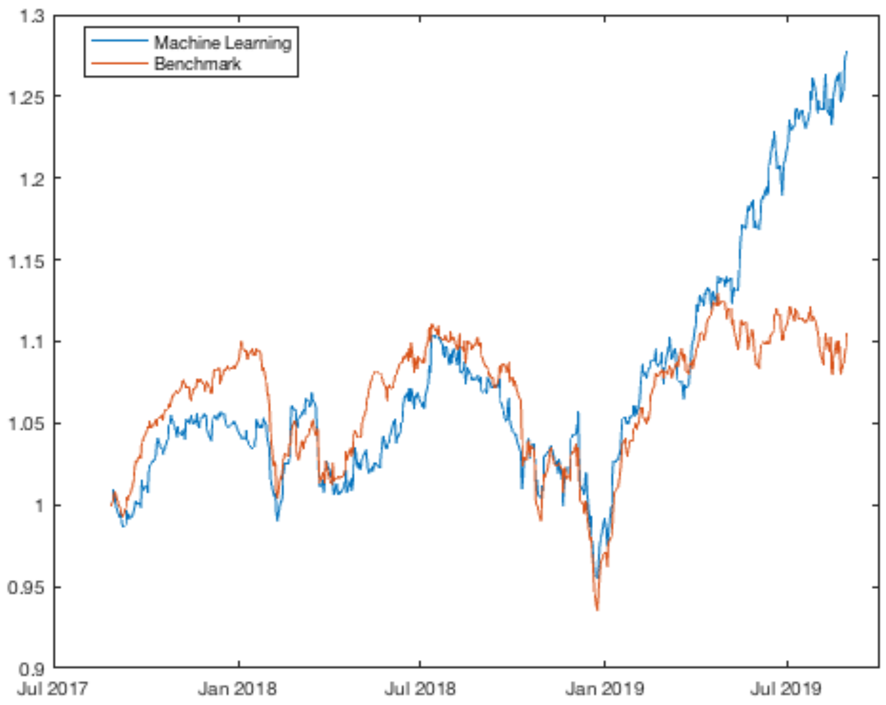


Figure 78: Equity curve – Naive Bayes Classifier Strategy – Out of Sample –S&P TSX 60

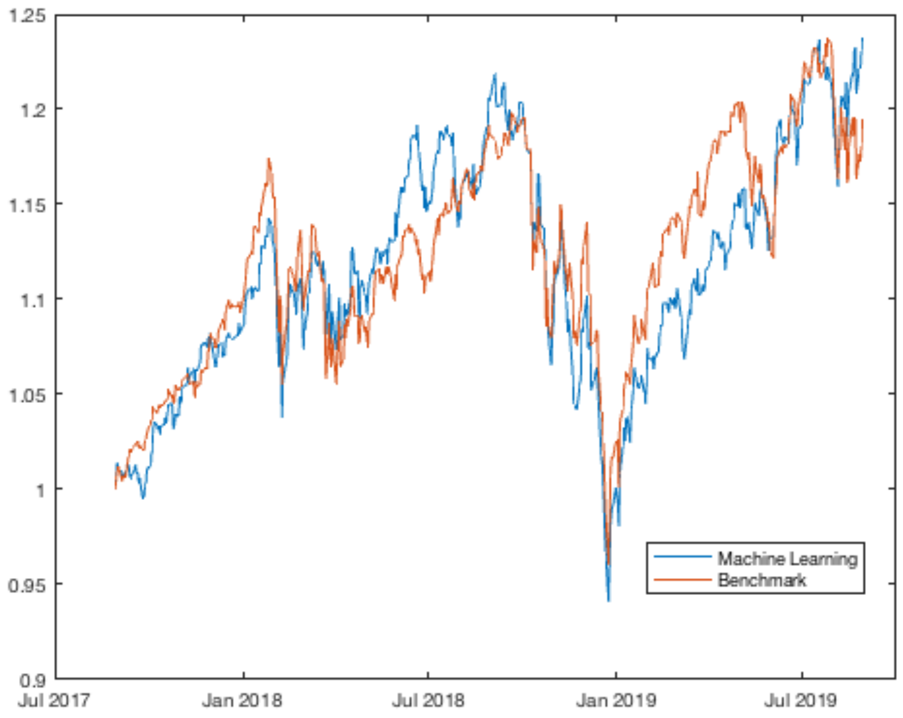


Figure 79: Equity curve – Naive Bayes Classifier Strategy – Out of Sample – S&P 500

The table with the different performance indicators obtained is then added,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error

ATX	0,6128	11.852.400	16,60%	27,31%	4,84e-02%	6,13%	0,87%
CAC 40	0,9937	13.611.300	23,56%	18,47%	5,07e-02%	4,79%	0,65%
DAX 30	0,6620	13.610.300	35,54%	23,44%	7,71e-02%	5,02%	1,55%
FTSE MIB	0,9763	13.787.800	19,12%	26,40%	6,99e-02%	7,03%	0,99%
FTSE 100	1,2303	14.300.900	23,98%	16.41%	7,99e-02%	10,90%	0,74%
Mercado Continuo	0,5556	12.560.000	20,42%	21,17%	4,07e-02%	7,46%	0,69%
S&P TSX 60	1,1969	12.767.700	13,93%	15,80%	3,43e-02%	5,64%	0,52%
S&P 500	0,8144	12.370.100	22,79%	19,78%	1,24e-02%	1,48%	0,46%

Table 5:: Compilation of Performance indicators – Naive Bayes Classifier Strategy – Out of Sample

The equity curves lead to the conclusion that the strategy has managed to beat the benchmarks of all the financial markets considered in the Out of Sample period, and moreover it does so very considerably in all cases except for the S&P 500. This is verified by analysing the information ratios, whose values are excellent in all cases except for the S&P 500 whose value is simply good. In this instance the strategy has more volatility than during the in-sample period and in only half the cases succeeds in beating the market with less drawdown. This is not surprising since as stated earlier 2018 was a particularly bad year for the stock markets, thus higher volatility falls within the expected behaviour.

In view of the results obtained both in the In Sample period and especially in the Out of Sample period, it can be concluded with a high level of confidence that the strategy based on the Machine Learning algorithm developed through the Naive Bayes classifier manages to consistently beat the market.

In the following chapter the strategies based on Deep Learning algorithms will be studied.

4.2.2.2 Deep Learning Strategies

These strategies are based on artificial neural networks. In both of the strategies developed, it is the neural network that will provide the buy and sell signals for each stock. It is therefore also a systematic strategy in which the investor does not take the investment decisions individually for each stock.

4.2.2.2.1 Long Short - Term Memory Strategy

This chapter will explain and define what a Long Short-Term Memory (LSTM) neural network is. This type of neural network is classified as a recurrent neural network. This is a neural network that has closed feedback loops, which go from outputs to inputs, allowing information to persist during some steps. These steps are called epochs. In this way, the previous outputs become part of the inputs, i.e. the outputs are remembered from one step to the next. That memory between consecutive epochs is what gives the name to this type of neural network, since it seems as if the neural network has some memory. The added interest of this type of loop is that it allows the neural network to show a temporary behavior. This type of neural network is therefore suitable for providing results that allow decisions to be made based on time series, as is the case in this work.

The first step in order to develop the strategy based on a Long Term Short Term Memory Neural Network is to perform the training of the algorithm on the selected data set. In this case, this data set contains also the predictors developed on the basis of the reference index. To do this, the TrainLSTM function is used, which trains and generates the neural network that will then be applied to a new data set to obtain the predictions.

Train LSTM Networks

S&P 500

```
c = 'SP500';
[net_SP500] = TrainLSTM(SP500_Benchmark_xTrain,
SP500_Benchmark_yTrainFF, SP500_Benchmark_nFeatures);
```

```
layers = 
  4x1 Layer array with layers:

     1   ''   Sequence Input      Sequence input with 11 dimensions
     2   ''   LSTM                LSTM with 100 hidden units
     3   ''   Fully Connected     1 fully connected layer
     4   ''   Regression Output   mean-squared-error
```

```
save net_SP500
```

ATX

```
c = 'ATX';
[net_ATX] = TrainLSTM(ATX_Benchmark_xTrain, ATX_Benchmark_yTrain2,
ATX_Benchmark_nFeatures);
```

```
layers = 
  4x1 Layer array with layers:

     1   ''   Sequence Input      Sequence input with 11 dimensions
     2   ''   LSTM                LSTM with 100 hidden units
     3   ''   Fully Connected     1 fully connected layer
     4   ''   Regression Output   mean-squared-error
```

```
save net_ATX
```

The first step in training the neural network is to define the architecture of the layers. In particular, it is considered that the intermediate layer will be made up of 100 hidden units, i.e. 100 "neurons". There is no clear rule for defining the optimal number of neurons that a hidden layer should contain. Generally what is done, and what has been done in this work, is to reach a satisfactory value by trial and error. In this case, with 100 neurons, satisfactory values have been obtained. Then the number of dimensions of the input layer is defined, which has to coincide with the number of predictors that are going to be used, in this case 11. Finally the output layer has only one dimension, which is obtained by regression and will provide the value of the next day's return for each of the stocks expected by the algorithm.

```
function [net] = TrainLSTM(xTrain, yTrain, nFeatures)
```

Define LSTM layers

```
hiddenUnit = 100;
layers = [ sequenceInputLayer(nFeatures)
           lstmLayer(hiddenUnit,"OutputMode","last")
           fullyConnectedLayer(1)
           regressionLayer]
```

The training options are defined below. Firstly, the solver to be used is defined. In this case the Adam Optimizer has been chosen, which stands for adaptive moment estimation and is one of the most widely used solvers for this type of neural network. Then it is established that the execution environment is the cpu. By setting the 'Verbose' option to false, the training progress information is not shown in the command window, since the aim is for the training progress to be shown in a plot, as specified below. Next the level of the gradient from which it will be clipped is defined, the threshold. Then the number of samples each iteration will have, in this case 27. Finally the maximum value of epochs is defined, that is the number of times that the algorithm evaluates the whole training data set, in this case 20.

Set training options

```
opts = trainingOptions('adam',...
    'ExecutionEnvironment','cpu',...
    'Verbose',false, ...
    'Plots','training-progress', ...
    'GradientThreshold', 1,...
    'MiniBatchSize',27,...
    'MaxEpochs',20);
```

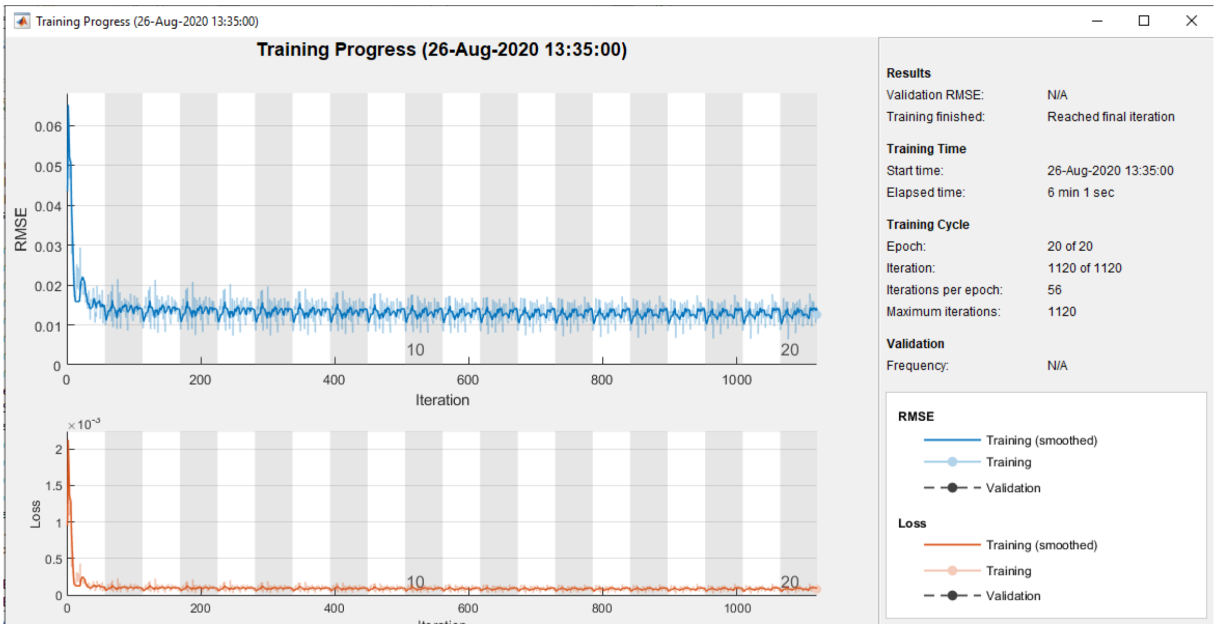
Once the LSTM neural network architecture and training options are defined, the training is launched using the following lines of code.

Train the network using traning data (a subset of in-sample data) with balanced class response variable.

```
yTrain = table2array(yTrain);


net = trainNetwork(xTrain,yTrain,layers, opts);


end
```

The net variable is the trained LSTM neural network, which will be used later with new data sets.

Below is a screenshot with the example of the training progress in the case of the S&P 500.



Training Progress (26-Aug-2020 13:35:00)

Results

Validation RMSE:	N/A
Training finished:	Reached final iteration

Training Time

Start time:	26-Aug-2020 13:35:00
Elapsed time:	6 min 1 sec

Training Cycle

Epoch:	20 of 20
Iteration:	1120 of 1120
Iterations per epoch:	56
Maximum iterations:	1120

Validation

Frequency:	N/A

RMSE
- Training (smoothed)
- Training
- Validation

Loss
- Training (smoothed)
- Training
- Validation

Figure 80: Training of the LSTM neural network

Once the trained LSTM neural network is available, the SignalCreationLSTM_LowBeta() function can be called, which is responsible for generating the matrix with the signals.

Create Long Short Term Model Neural Network signal and backtest - In-sample

For signal creation, buy and sell depending on the output of the neural network. If the return is in the top 10% we buy, if it is not we do nothing. When a stock we have in portfolio passes to be in the top 10% to not being in the top 10% we sell it. If the stock passes of not being in the top 10% to being in the top 10% we buy it.

An extra clasification is pursued to favor the stocks with a low average beta

We create the signal matrix for each of the index by calling the SignalCreationLSTM_LowBeta function for each of them.

CAC 40

```
index = 'CAC';
CAC40_nDay = height(CAC40_inSample);
[CAC40_signalinSampleLSTM] = SignalCreationLSTM(CAC40_xInSample,
CAC40_nStocks, CAC40_nDay, index);
```

DAX 30

```
index = 'DAX';
DAX30_nDay = height(DAX30_inSample);
```

```
[DAX30_signalinSampleLSTM] =
SignalCreationLSTM_LowBeta(DAX30_xInSample, DAX30_nStocks, DAX30_nDay,
DAX30_inSample,  index);
```

In the first step, which is similar to the strategy based on Machine Learning, a cell type data structure is created, called XinStocks, in which, thanks to a for loop, the 11 predictors associated with each share are filled in each of the cells.

Creation of the Signal Matrix

We obtain a list for each in sample day, if signal = 1, we buy, if signal = -1, we sell, if signal = 0, we hold.

Based on the output of the previously trained LSTM neural network and a classification of the average beta.

```
function [signal] = SignalCreationLSTM_LowBeta(xSample, nbStocks,
nDays, TR, c)


   % We want to build one matrix for each of the stocks, based on the
   % output of the neural network.

   % We convert xSample to an array from a cell structure


   xSample2 = cell2mat(xSample()');
   xSample = xSample2';

   nFeatures = 11;

   Temp_Pred = xSample(:,1:6);

   XinStocks = cell(1, nbStocks);
   yPredInSample = cell(1, nbStocks);

   for i = 1: nbStocks
       ColToKeep = 1;
       ColToDelete = nbStocks;
       Predictors = xSample(:,7:end);
       ColIndex = mod(0:size(Predictors,2)-1, ColToKeep+ColToDelete-1)
== i-1;
       Predictors(:,~ColIndex) = [];
       XinStocks{i} = [Temp_Pred Predictors];
    end
```

The second step is to create a switch structure, which, depending on the reference index passed as an argument for the function, is responsible for loading the LSTM neural network for each index. Then the predict() function is used. This function has as arguments the LSTM neural network and the new data set, derived directly from each of the cells of the XinStock structure. In this way, the predictions for the new data set are obtained, as shown below,

```
switch c
```

```
    case 'SP500'
        load net_SP500
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(net_SP500, input);
        end
    case 'ATX'
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(netATX_Prueba, input);
        end
    case 'CAC'
        load net_CAC
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(net_CAC, input);
        end
    case 'DAX'
        load net_DAX
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(net_DAX, input);
        end
    case 'FTSE_MIB'
        load net_FTSE_MIB
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(net_FTSE_MIB, input);
        end
    case 'FTSE100'
        load net_FTSE100
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(net_FTSE100, input);
        end
    case 'MC'
        load net_MC
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
            yPredInSample{j} = predict(net_MC, input);
        end
    case 'SP_TSX60'
        load net_SP_TSX60
        for j = 1:nbStocks
            input = XinStocks{j}';
            input = mat2cell(input,nFeatures, ones(nDays,1))';
```

```
                yPredInSample{j} = predict(net_SP_TSX60, input);
            end
     end
    yPredinSample = cell2mat(yPredInSample());
```

The returns predicted by the LSTM neural network are then classified, so that the first signal is composed by associating a Buy signal to the 10% of the shares with a higher predicted return. On the contrary, a sell signal is generated for the 10% of the shares that have a worse return predicted by the neural network. For the remaining 80% no signal is generated.

```
[~,I] = sort(yPredinSample,2);
    uplimit = round(0.9*nbStocks);
    lowlimit = round(0.1*nbStocks);
    signal2 = yPredinSample;
    for b = 1:nDays
        signal2(b,I(b,uplimit:end)) = 1;
        signal2(b,I(b,1:lowlimit)) = -1;
        signal2(b,I(b,lowlimit+1:uplimit+1)) = 0;
    end
```

Finally, similar to the Rules based strategy on momentum and low beta, the stocks are classified according to their average beta. In this way, a buy signal is generated for 50% of the shares with the lowest beta and a sell signal is generated for those with the highest beta, as shown below,

```
hasbeta = ~cellfun('isempty', regexp(TR.Properties.VariableNames,
'Beta', 'once'));
    beta = TR(:,TR.Properties.VariableNames(hasbeta));
    beta = table2array(beta);
    [~,I] = sort(beta,2);
    uplimitbeta = round(0.5*nbStocks);
    lowlimitbeta = round(0.5*nbStocks);
    signal3 = beta;
    for b = 1:nDays
        signal3(b,I(b,uplimitbeta:end)) = -1;
        signal3(b,I(b,1:lowlimitbeta)) = 1;
        signal3(b,I(b,lowlimitbeta+1:uplimitbeta+1)) = 0;
    end
```

Lastly, the two matrices are combined to create the final matrix in a similar way to that used in previous strategies as shown in the following double for loop,

```
signal1 = zeros(nDays,nbStocks);

    for i = 1:nDays
      for j = 1:nbStocks
            if signal2(i,j) == 1 && signal3(i,j) == 1
             signal1(i,j) = 1;
             elseif signal2(i,j) == -1 && signal3(i,j) == -1
             signal1(i,j) = -1;
            end
```

```
     end
  end

   signal = signal1;

   for c = 2:nDays
       for d = 1:nbStocks
           if signal1(c-1,d) == 1 && signal1(c,d) == 1
               signal(c,d) = 0;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == 0
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d)  == -1 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == -1
               signal(c,d) = 0;
           end
       end
   end
end
```

Thus, the creation of the matrix with the signals is completed, which is then used by the PortfolioManagement() function, in a similar way to the previous ones.

This strategy also uses the process of portfolio optimization, which means that the portfolio will also be updated on a monthly basis and in such a way that the portfolio is positioned on the efficient frontier, maximizing the Sharpe Ratio.

Below are the equity curves showing the evolution of the strategy based on a Long Short-Term Memory neural network and low beta for the In Sample period,

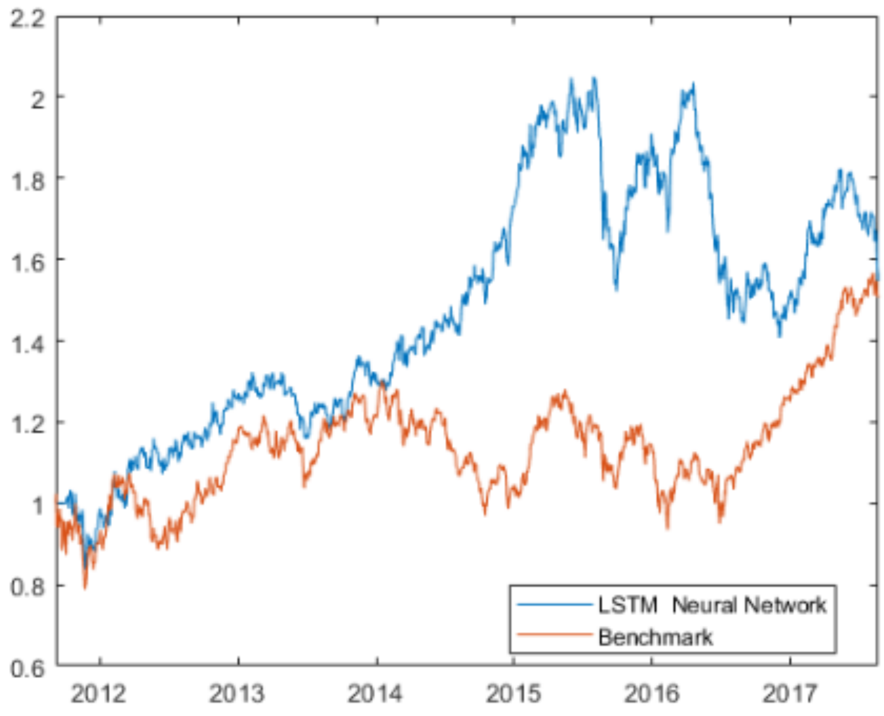


Figure 81: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – ATX

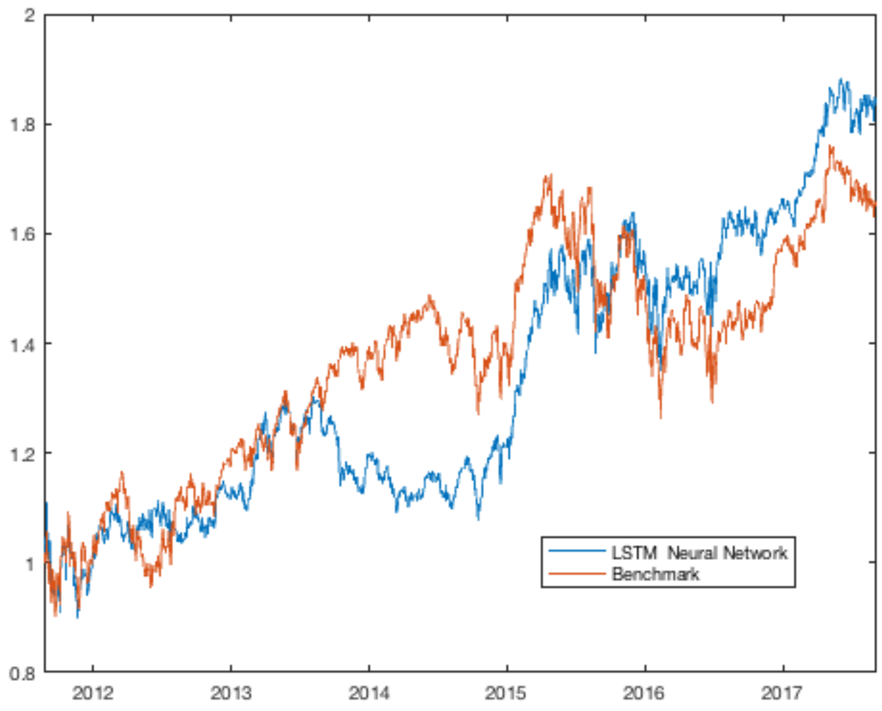


Figure 82: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – CAC40

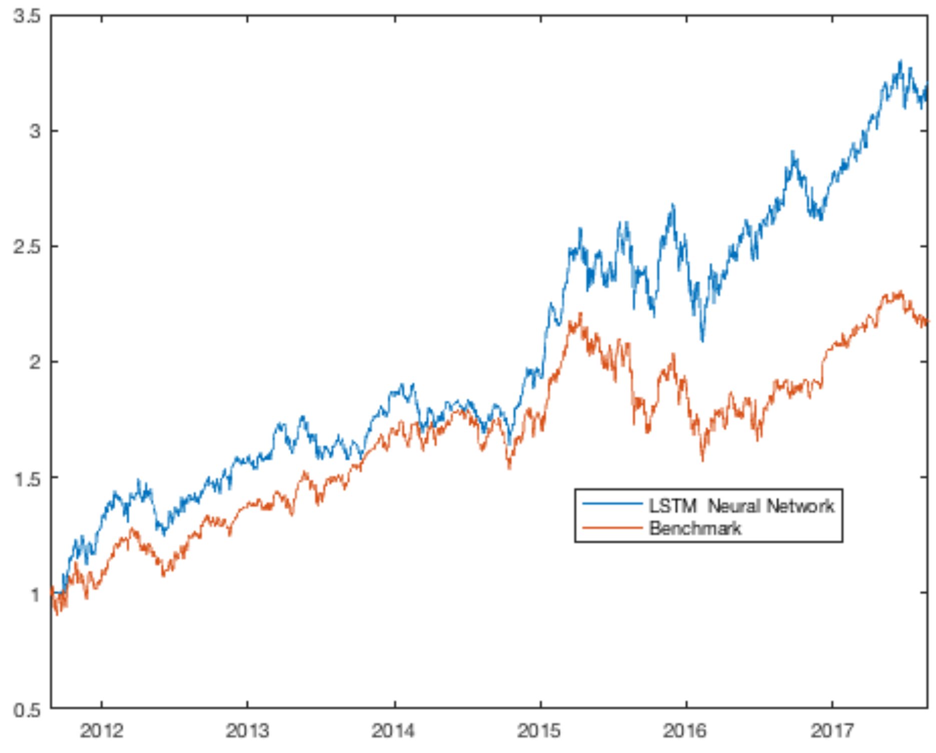


Figure 83: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – DAX 30

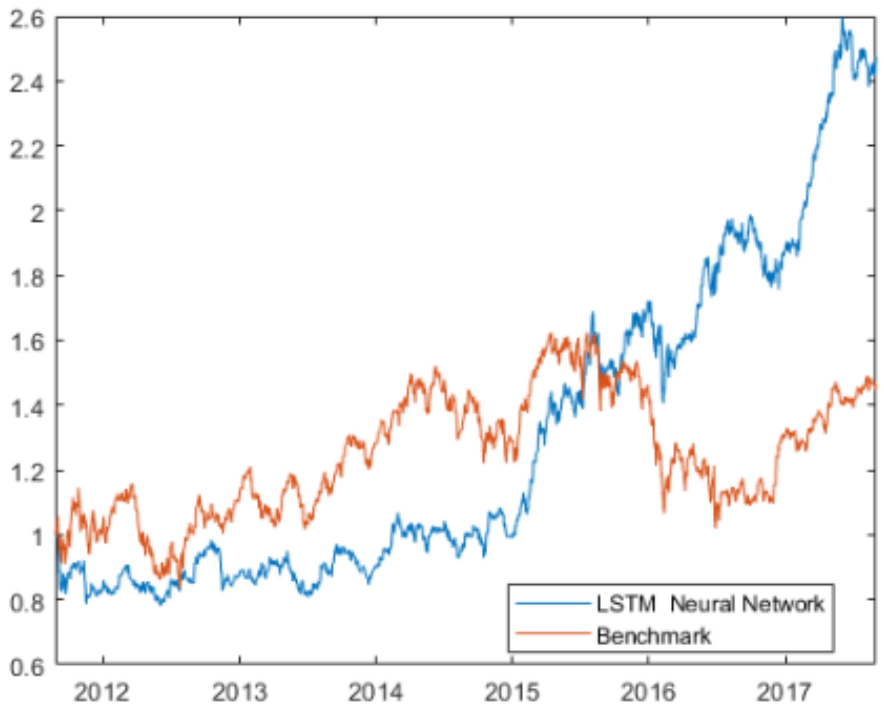


Figure 84: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – FTSE MIB

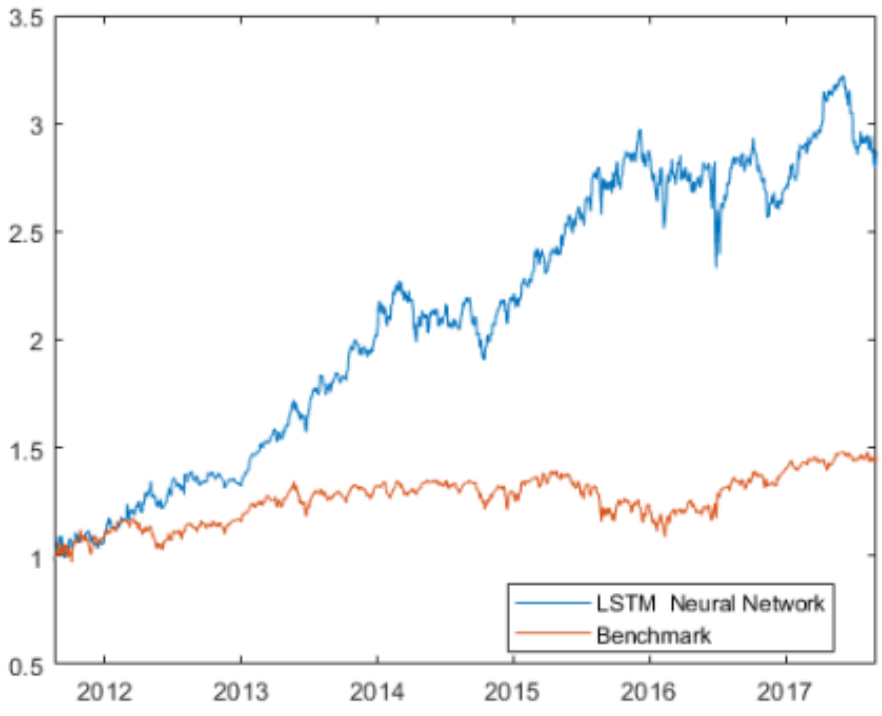


Figure 85: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – FTSE 100

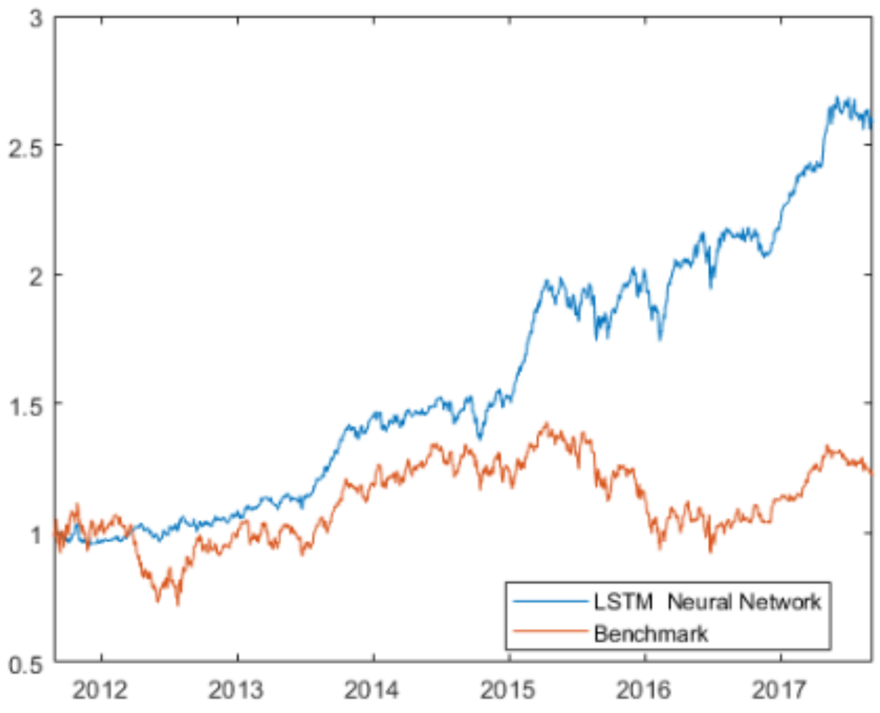


Figure 86: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – Mercado Continuo

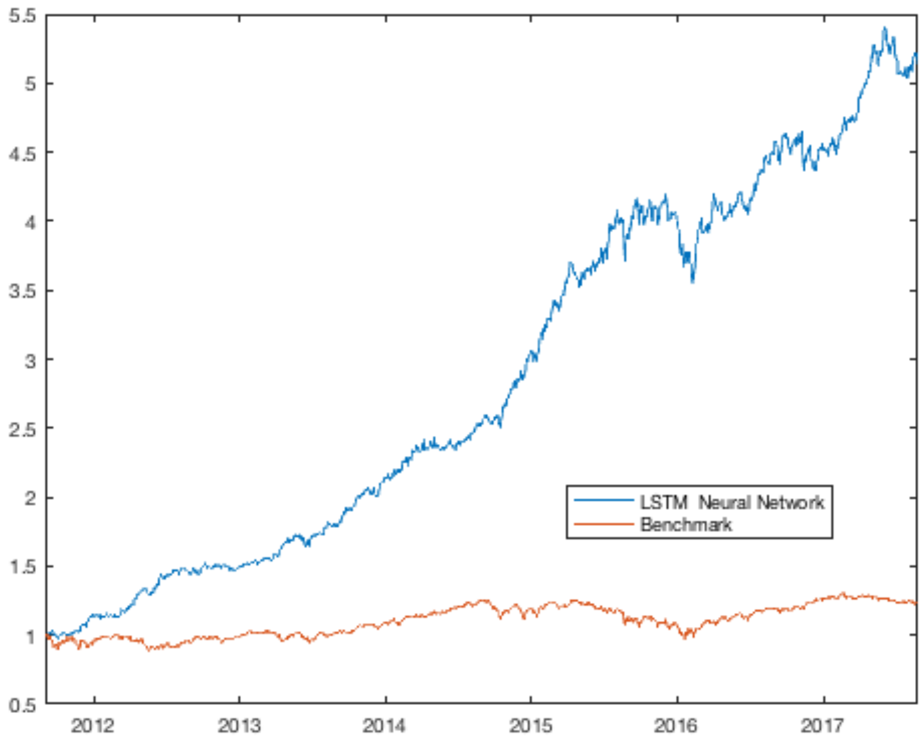


Figure 87: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – S&P TSX 60

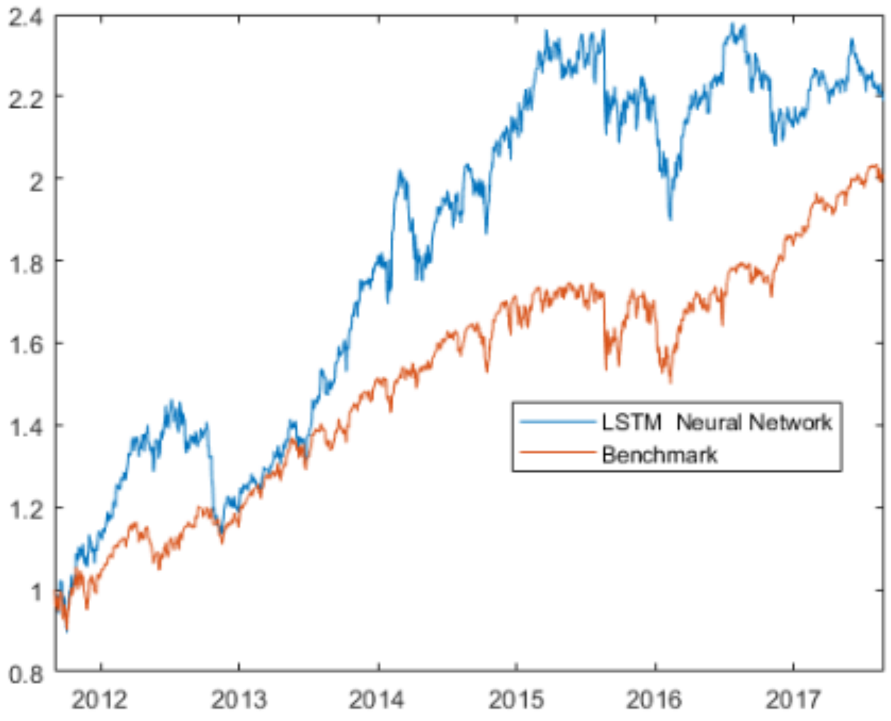


Figure 88: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample – S&P 500

A table with the different calculated indicators is shown below,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
ATX	0,4873	15.484.200	31,33%	28,29%	2,15e-02%	4,58e-02%	1,31%
CAC 40	0,6413	18.287.700	19,05%	26,04%	1,83e-02%	0,56%	0,85%
DAX 30	1,0747	32.006.000	22,38%	29,27%	4,03e-02%	2,88%	0,87%
FTSE MIB	0,7826	24.726.000	24,10%	37,15%	5,15e-02%	2,08%	1,48%
FTSE 100	1,0445	28.693.000	21,54%	22.06%	5,39e-02%	4,87%	0,95%
Mercado Continuo	1,2668	25.981.000	14,13%	35,75%	5,72e-02%	3,71%	1,13%
S&P TSX 60	2,2335	51.373.000	15,41%	22,86%	0,10%	11,80%	0,81%
S&P 500	0,8445	21.870.000	22,42%	14,16%	1,04e-02%	1,12%	0,72%

Table 6: Compilation of Performance indicators – Long Short-Term Memory Neural Network Low Beta Strategy – In Sample

The analysis of the results allows to conclude that the strategy is capable of consistently beating the benchmark indices in all markets in the in-sample period. In most cases this outperformance is very important and during most of the period. Despite the good results, both graphically when looking at the evolution and when looking at the table with the results, it should not be forgotten that this is the In Sample period, that is to say that part of the data from this period has been used to carry out the training of the Long Short-Term Memory neural Network.

In order to confirm the results it is necessary to analyse the behaviour of the strategy also in the Out of Sample period. As mentioned above, this is a new set of data with which the neural network has had no previous contact, hence the importance of the strategy working in this period.

Thereafter, the results of the backtest during the Out of Sample period for the momentum strategy with low beta are shown below.

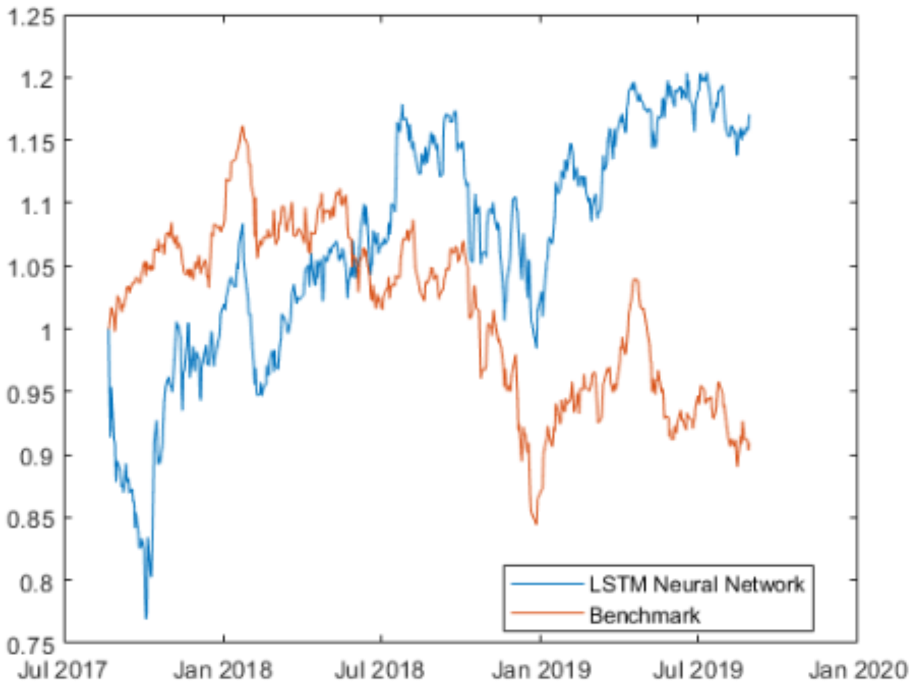


Figure 89: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – ATX

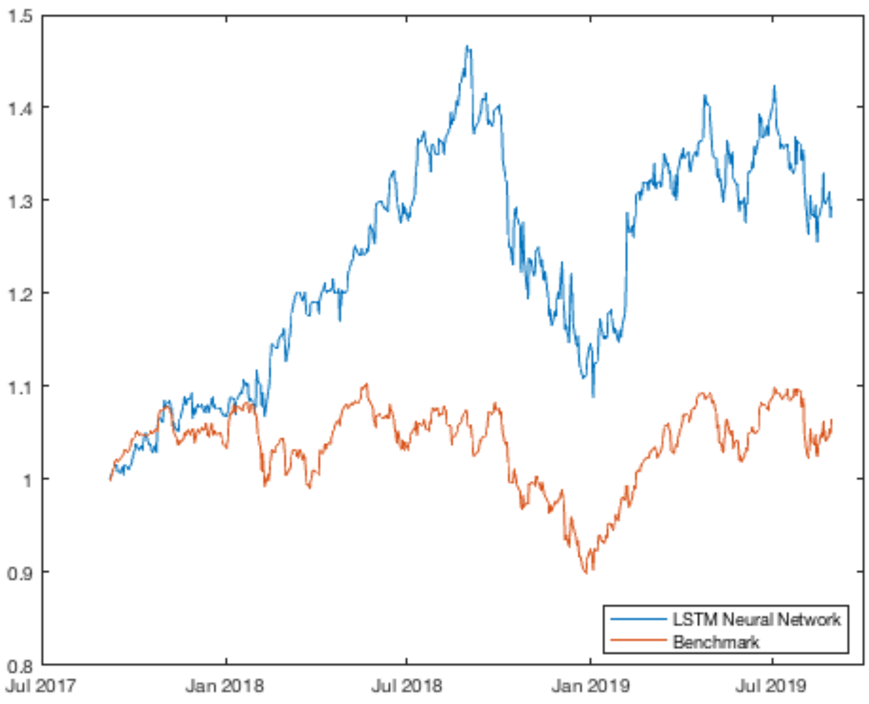


Figure 90: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – CAC 40

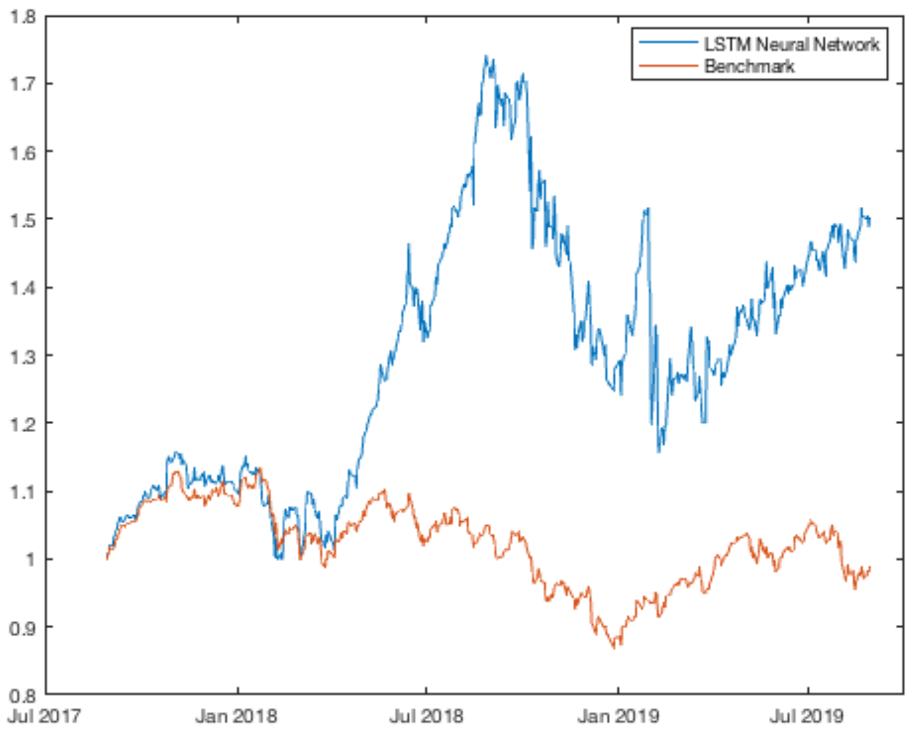


Figure 91: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – DAX 30

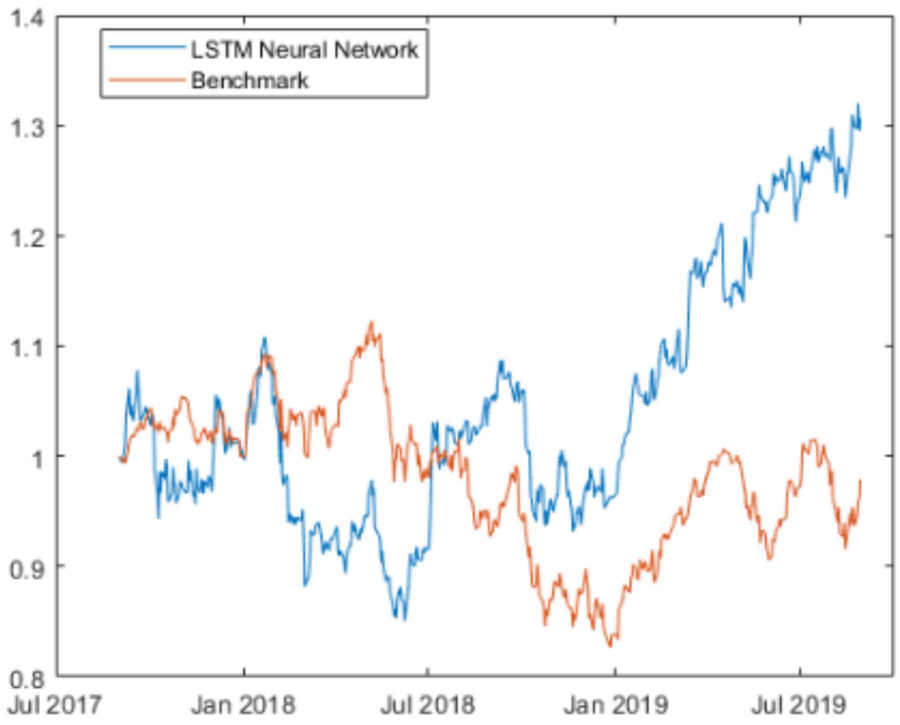


Figure 92: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – FTSE MIB

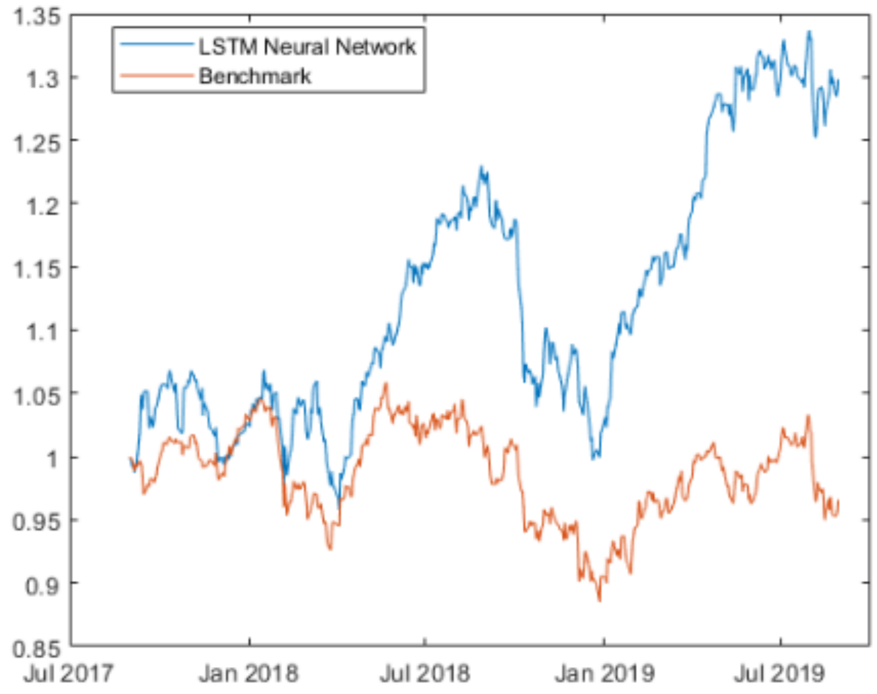


Figure 93: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – FTSE 100

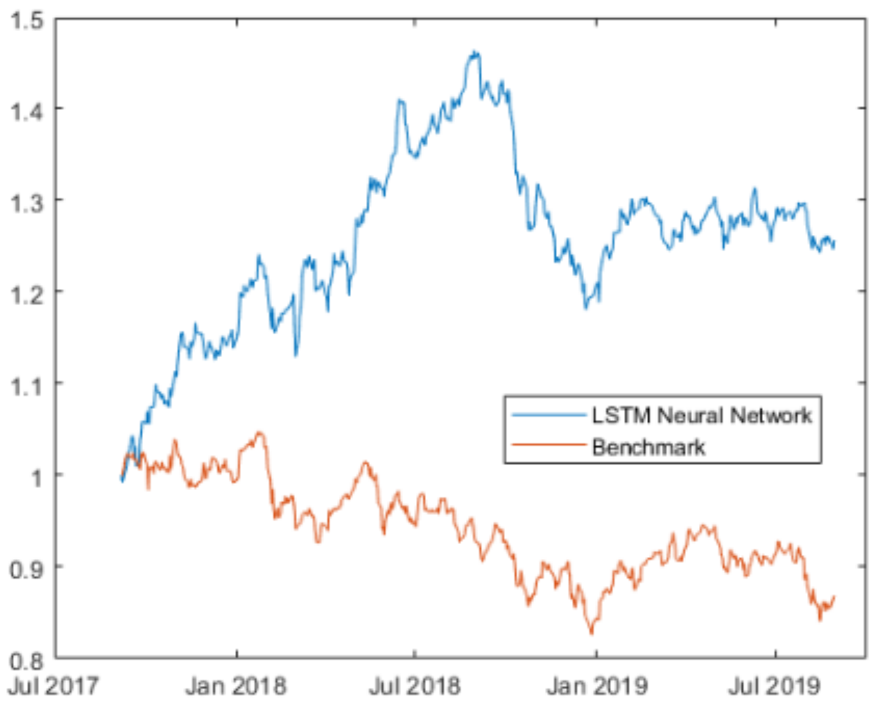


Figure 94: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – Mercado Continuo

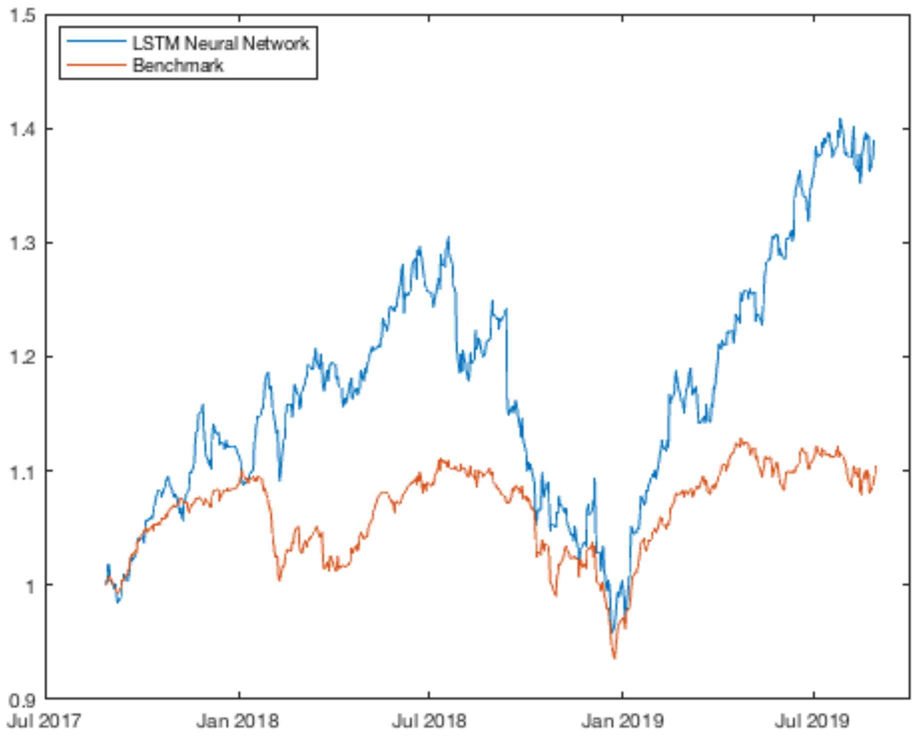


Figure 95: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – S&P TSX 60

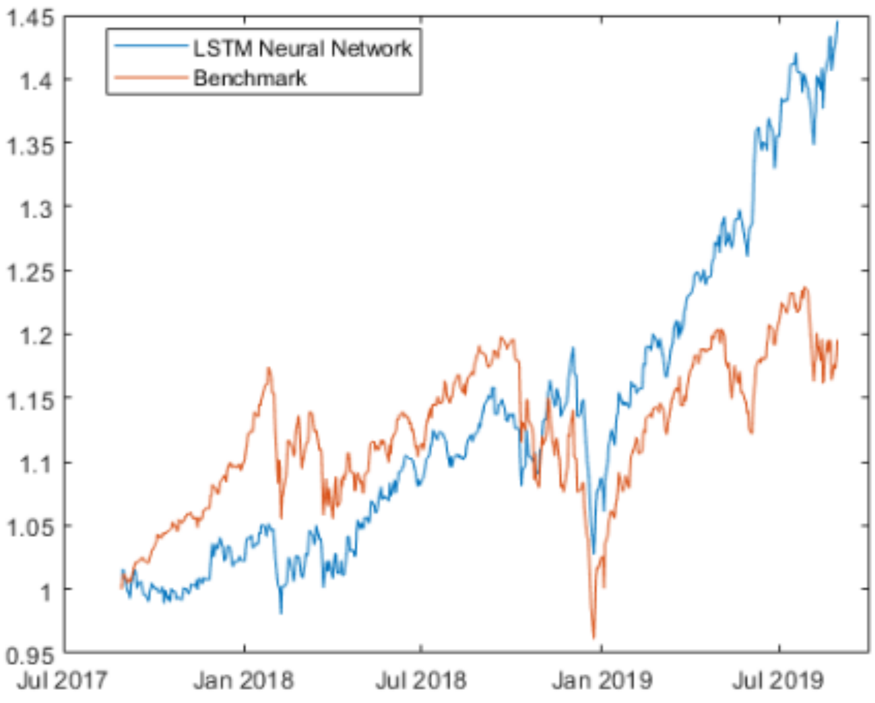


Figure 96: Equity curve – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample – S&P 500

The table with the different calculated indicators for the Out of Sample is shown below,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
ATX	0,4746	11.693.300	23,10%	27,31%	4,92e-02%	4,30%	1,28%
CAC 40	0,7731	12.928.100	25,84%	18,47%	4,34e-02%	4,62%	0,92%
DAX 30	0,8033	15.033.600	33,58%	23,44%	9,88e-02%	5,97%	1,67%
FTSE MIB	0,7271	13.048.300	23,25%	26,40%	6,16e-02%	4,95%	1,24%
FTSE 100	0,9747	12.966.100	18,93%	16.41%	5,95e-02%	8,73%	0,69%
Mercado Continuo	0,8953	12.552.800	19,32%	21,17%	6,22e-02%	8,69%	0,85%
S&P TSX 60	1,0763	13.893.200	26,66%	15,80%	4,87e-02%	5,78%	0,85%
S&P 500	1,5414	14.438.200	13,66%	19,78%	5,04e-02%	6,10%	0,60%

Table 7: Compilation of Performance indicators – Long Short-Term Memory Neural Network Low Beta Strategy – Out of Sample

Thanks to the analysis of both the equity curves and the indicators it seems clear that it can be confirmed with a high level of security that the strategy based on a Long Short - Term Memory neural network works. This strategy has outperformed the benchmarks of all the analysed markets both in the in-sample period and more importantly in the out-of-sample period. The summary tables for both periods show firstly that good Sharpe ratios are achieved, of around 1. There are also very good information ratio values in both periods, i.e. risk adjusted return taking into account the volatilities of the strategy and the benchmark. Finally regarding the maximum drawdowns, in some more than half of the times the strategy achieves lower values than those obtained in the index, in no case reaching excessively high values for which it can be concluded that it is not a very risky strategy.

It is therefore concluded that the strategy based on the Long Short-Term Memory Neural Network complies with the objective by consistently beating the market.

In the following chapter, an investment strategy based on a different type of neural network is analysed.

4.2.2.2.2 Feedforward Neural Network Strategy

The last investment strategy that has been tested is based on a FeedForward type neural network. Unlike the neural network studied in the previous section, it does not have a feedback loop from outputs to inputs. Its architecture is based on an open loop so that no information from the outputs is transferred to the inputs of the next loop. This is the first and simplest of the different neural network architectures that exist. The information that travels through it only goes one way, forward, hence its name. As it will be seen later in the diagram that represents the neural network used, this type of network has input nodes, which form what is called the Input Layer, intermediate nodes that form the Hidden Layers and finally output nodes, forming the Output Layers. In the case where the number of Hidden Layers is 0, the neural network is called a perceptron and it is the simplest case of FeedForward Neural Network. When the number of hidden layers is higher than 0, as it is the case in this work, the resulting neural network is called a Multilayer Perceptron. The advantage of this type of network over the simple perceptron is that it is able to solve problems that are not linearly separable. The universal theorem of approximation states that "A FeedForwad neural network with a single hidden layer containing a finite number of neurons can approximate the continuous functions in compact subsets of Rn, under slight assumptions about the activation function. Thus, the theorem states that simple neural networks can represent a variety of interesting functions when the correct parameters are modelled". It follows therefore that with a single hidden layer and by determining an adequate number of neurons, satisfactory results can be obtained for most problems. That's why in most problems feedforward neural networks with a single Hidden Layer are implemented. This will be the architecture followed in this work.

In this case the definition of the architecture and the parameters of the training are simpler than with respect to the LSTM neural network. In a similar way to the strategy based on a LSTM neural network, the algorithm will try to predict in a numerical way the return of the next day.

First of all, it is necessary to conduct the training of the neural network. It will be done once for each benchmark so that in the end 8 neural networks will be achieved. The procedure followed is the usual one during this work, with 8 modules, one for each index calling the TrainNN function. This function is responsible for generating the neural network. When it is called, a function called MyNeuralNetworkFunction is created, followed by the name of the index, for example MyNeuralNetworkFunctionDAX.

Train Feed Forward Neural Network

FTSE MIB

```
c = 'FTSE_MIB';
[FTSE_MIB_yPredInSample] = TrainNN(FTSE_MIB_Benchmark_xTrainFF,
FTSE_MIB_Benchmark_yTrainFF, FTSE_MIB_Benchmark_Xin, c);
```

```
performance = 0.0011
trainPerformance = 0.0011
valPerformance = 9.7638e-04
testPerformance = 0.0011

MATLAB function generated: myNeuralNetworkFunctionFTSE_MIB.m
To view generated function code: edit myNeuralNetworkFunctionFTSE_MIB
For examples of using function: help myNeuralNetworkFunctionFTSE_MIB
```

FTSE 100

```
c = 'FTSE100' ;
```

```
[FTSE100_yPredInSample] = TrainNN(FTSE100_Benchmark_xTrainFF,
FTSE100_Benchmark_yTrainFF, FTSE100_Benchmark_Xin, c);
performance = 1.8840e-04
trainPerformance = 1.9724e-04
valPerformance = 1.7891e-04
testPerformance = 1.7144e-04

MATLAB function generated: myNeuralNetworkFunctionFTSE100.m
To view generated function code: edit myNeuralNetworkFunctionFTSE100
For examples of using function: help myNeuralNetworkFunctionFTSE100
```

The first step is to define which training function will be used. In this case it has been decided to use the Gradient Descent Backpropagation function, based on a descent gradient. This algorithm enables good results to be obtained quickly. The number of hidden layers and their size are defined below. In this case there will be only one hidden layer with 2 neurons because it is a simple architecture and theoretically it is able to solve with enough precision any complex problem. Then are defined some of the settings for a FeedForward Neural Network generated by MATLAB for this type of neural network.

```
function [yPredInSample] = TrainNN(xTrainFF, yTrainFF, Xin, c)


    x = xTrainFF';
    yT = table2array(yTrainFF);
    t = yT';

    % % Choose a Training Function
    trainFcn = 'traingd';  % Gradient Descent backpropagation.
    %
    % % Create a Fitting Network
    hiddenLayerSize = 2;
    net = feedforwardnet(hiddenLayerSize,trainFcn);
    %
    % % Choose Input and Output Pre/Post-Processing Functions
    net.input.processFcns = {'removeconstantrows','mapminmax'};
    net.output.processFcns = {'removeconstantrows','mapminmax'};
    %
```

The next step enables the division of the training data set. As an additional safety measure to prevent overfitting of the neural network, the data is divided by 60% for neural network training, 20% for validation and the remaining 20% for a test to verify the performance of the algorithm. Finally, the Mean Squared Error is used to measure this performance.

```
% % Setup Division of Data for Training, Validation, Testing
    net.divideFcn = 'dividerand';  % Divide data randomly
    net.divideMode = 'sample';  % Divide up every sample
    net.divideParam.trainRatio = 60/100;
    net.divideParam.valRatio = 20/100;
    net.divideParam.testRatio = 20/100;
    %
    % % Choose a Performance Function
    net.performFcn = 'mse';  % Mean Squared Error
```

Once these parameters are established, the Feedforward neural network can be trained with the next line of code. The net variable is the one that represents the neural network.

```
[net,tr] = train(net,x,t);
```

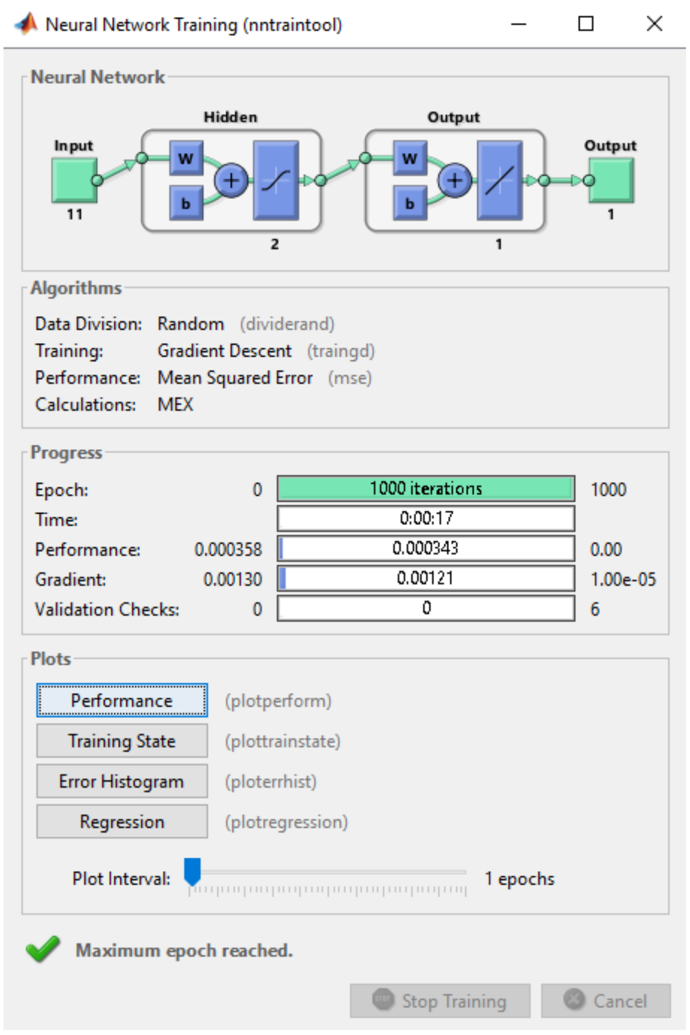
The parameters are then established to control the performance of the algorithm while the training is being conducted and to visualise the evolution of this performance and the characteristics of the network.

```
% % Test the Network
    y = net(x);
    e = gsubtract(t,y);
    performance = perform(net,t,y)

    % Recalculate Training, Validation and Test Performance
    trainTargets = t .* tr.trainMask{1};
    valTargets = t .* tr.valMask{1};
    testTargets = t .* tr.testMask{1};
    trainPerformance = perform(net,trainTargets,y)
    valPerformance = perform(net,valTargets,y)
    testPerformance = perform(net,testTargets,y)

    %   View the Network
    view(net)
```

Finally, by means of a switch structure, depending on which benchmark has been used to train the model, the Feedforward neural network is generated and saved. As shown below.

```
switch c
        case 'ATX'
            genFunction(net,'myNeuralNetworkFunctionATX');
            yPredInSample = myNeuralNetworkFunctionATX(Xin');
        case 'SP500'
            genFunction(net,'myNeuralNetworkFunctionSP500');
            yPredInSample = myNeuralNetworkFunctionSP500(Xin');
        case 'CAC'
            genFunction(net,'myNeuralNetworkFunctionCAC');
            yPredInSample = myNeuralNetworkFunctionCAC(Xin');
        case 'DAX'
            genFunction(net,'myNeuralNetworkFunctionDAX');
            yPredInSample = myNeuralNetworkFunctionDAX(Xin');
        case 'FTSE_MIB'
            genFunction(net,'myNeuralNetworkFunctionFTSE_MIB');
            yPredInSample = myNeuralNetworkFunctionFTSE_MIB(Xin');
        case 'FTSE100'
            genFunction(net,'myNeuralNetworkFunctionFTSE100');
            yPredInSample = myNeuralNetworkFunctionFTSE100(Xin');
        case 'MC'
            genFunction(net,'myNeuralNetworkFunctionMC');
            yPredInSample = myNeuralNetworkFunctionMC(Xin');
        case 'SP_TSX60'
            genFunction(net,'myNeuralNetworkFunctionSP_TSX60');
            yPredInSample = myNeuralNetworkFunctionSP_TSX60(Xin');
    end
```

Below are displayed the plots obtained during the training of the Feedforward neural networks,


Neural Network Training (nntraintool)

Neural Network

Hidden
Output
Input
W
b
11
2
1
Output
1

Algorithms

Data Division: Random (dividerand)
Training: Gradient Descent (traingd)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	1000 iterations	1000
Time:		0:00:17	
Performance:	0.000358	0.000343	0.00
Gradient:	0.00130	0.00121	1.00e-05
Validation Checks:	0	0	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)

Plot Interval: 1 epochs

Maximum epoch reached.

Figure 97: Training of the Feedforward neural network

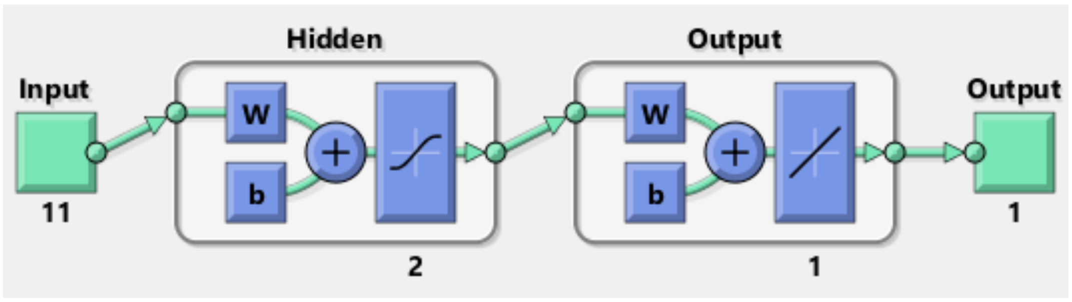


Figure 98: Arquitecture of the Feedforward neural network

Once the trained neural network is available, the SignalCreationFFNN_LowBeta() function is called.

Create FeedForward Neural Network signal and backtest - In-sample

For signal creation, buy and sell depending on the output of the neural network. If the return is in the top 10% we buy, if it is not we do nothing. When a stock we have in portfolio passes to be in the top 10% to not being in the top 10% we sell it. If the stock passes of not being in the top 10% to being in the top 10% we buy it.
An additional clasification is pursued to favor the stocks with a low average beta
We create the signal matrix for each of the index by calling the SignalCreation function for each of them.

Mercado Continuo

```
index = 'MC';
MC_nDay = height(MC_inSample);
[MC_signalinSample] = SignalCreationFFNN_LowBeta(MC_Xin, MC_nStocks,
MC_nDay, MC_inSample, index);
```

S&P TSX 60

```
index = 'SP_TSX60';
SP_TSX60_nDay = height(SP_TSX60_inSample);
[SP_TSX60_signalinSample] = SignalCreationFFNN_LowBeta(SP_TSX60_Xin,
SP_TSX60_nStocks, SP_TSX60_nDay, SP_TSX60_inSample, index);
```

This function follows the same structure as that developed for the LSTM neural network. First, a cell type data structure is generated where each of the cells represents the indicators of each individual share.

Creation of the Signal Matrix

We obtain a list for each in sample day, if signal = 1, we buy, if signal = -1, we sell, if signal = 0, we hold.

Based on the output of the previously trained feedforward neural network and with a low beta filter.

```
function [signal] = SignalCreationFFNN_LowBeta(Xin, nbStocks, nDays, T,
c)


   % We want to build one matrix for each of the stocks, with the 6
first
   % columns + Ret1 + Ret126 + Ret252 + RSI14 + MACD

   Temp_Pred = Xin(:,1:6);

   XinStocks = cell(1, nbStocks);
   yPredInSample = cell(1, nbStocks);

   for i = 1: nbStocks
       ColToKeep = 1;
       ColToDelete = nbStocks;
       Predictors = Xin(:,7:end);
       ColIndex = mod(0:size(Predictors,2)-1, ColToKeep+ColToDelete-1)
== i-1;
       Predictors(:,~ColIndex) = [];
       XinStocks{i} = [Temp_Pred Predictors];
    end
```

A switch structure is then used to determine which of the previously saved Feedforward neural networks to use to generate the prediction of the returns based on the new indicators.

```
      switch c

       case 'SP500'
           for j = 1:nbStocks
               yPredInSample{j} =
myNeuralNetworkFunctionSP500(XinStocks{j}');
           end
       case 'ATX'
           for j = 1:nbStocks
               yPredInSample{j} =
myNeuralNetworkFunctionATX(XinStocks{j}');
           end
       case 'CAC'
           for j = 1:nbStocks
               yPredInSample{j} =
myNeuralNetworkFunctionCAC(XinStocks{j}');
           end
       case 'DAX'
           for j = 1:nbStocks
               yPredInSample{j} =
myNeuralNetworkFunctionDAX(XinStocks{j}');
           end
       case 'FTSE_MIB'
           for j = 1:nbStocks
```

```
                yPredInSample{j} = 
myNeuralNetworkFunctionFTSE_MIB(XinStocks{j}');
            end
        case 'FTSE100'
            for j = 1:nbStocks
                yPredInSample{j} = 
myNeuralNetworkFunctionFTSE100(XinStocks{j}');
            end
        case 'MC'
            for j = 1:nbStocks
                yPredInSample{j} = 
myNeuralNetworkFunctionMC(XinStocks{j}');
            end
        case 'SP_TSX60'
            for j = 1:nbStocks
                yPredInSample{j} = 
myNeuralNetworkFunctionSP_TSX60(XinStocks{j}');
            end
   end

   yPredinSample = cell2mat(yPredInSample()');
   yPredinSample = yPredinSample';
```

The returns obtained are ordered and the stocks that make up the 10% with the best expected return are selected, generating a buy signal for them. For those stocks that are part of the 10% with the worst return a sell signal is generated. With the rest, no operation is undertaken.

```
[~,I] = sort(yPredinSample,2);
   uplimit = round(0.9*nbStocks);
   lowlimit = round(0.1*nbStocks);
   signal2 = yPredinSample;
   for b = 1:nDays
       signal2(b,I(b,uplimit:end)) = 1;
       signal2(b,I(b,1:lowlimit)) = -1;
       signal2(b,I(b,lowlimit+1:uplimit+1)) = 0;
   end
```

Finally, in this case too, the strategy favours those stocks with lower beta, so the filtering is done to generate buy signals for 50% of the stocks with a lower beta.

```
hasbeta = ~cellfun('isempty', regexp(T.Properties.VariableNames, 
'Beta', 'once'));
   beta = T(:,T.Properties.VariableNames(hasbeta));
   beta = table2array(beta);
   [~,I] = sort(beta,2);
   uplimitbeta = round(0.5*nbStocks);
   lowlimitbeta = round(0.5*nbStocks);
   signal3 = beta;
   for b = 1:nDays
       signal3(b,I(b,uplimitbeta:end)) = -1;
       signal3(b,I(b,1:lowlimitbeta)) = 1;
       signal3(b,I(b,lowlimitbeta+1:uplimitbeta+1)) = 0;
   end
```

Finally, with the same for loop used previously, both matrices are combined to generate the final signal matrix.

```
signal1 = zeros(nDays,nbStocks);

  for i = 1:nDays
     for j = 1:nbStocks
           if signal2(i,j) == 1 && signal3(i,j) == 1
            signal1(i,j) = 1;
            elseif signal2(i,j) == -1 && signal3(i,j) == -1
            signal1(i,j) = -1;
           end
     end
  end

   signal = signal1;

   for c = 2:nDays
       for d = 1:nbStocks
           if signal1(c-1,d) == 1 && signal1(c,d) == 1
               signal(c,d) = 0;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == 0
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 1 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == 0 && signal1(c,d) == -1
               signal(c,d) = -1;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == 1
               signal(c,d) = 1;
           elseif signal1(c-1,d)  == -1 && signal1(c,d) == 0
               signal(c,d) = 0;
           elseif signal1(c-1,d) == -1 && signal1(c,d) == -1
               signal(c,d) = 0;
           end
       end
   end
end
```

This concludes the generation of the signal matrix and the backtesting of the strategy can begin. This strategy also uses the portfolio optimization explained above.

The results obtained during the period in Sample for the strategy based on a Feedforward Neural Network and low beta are shown below,

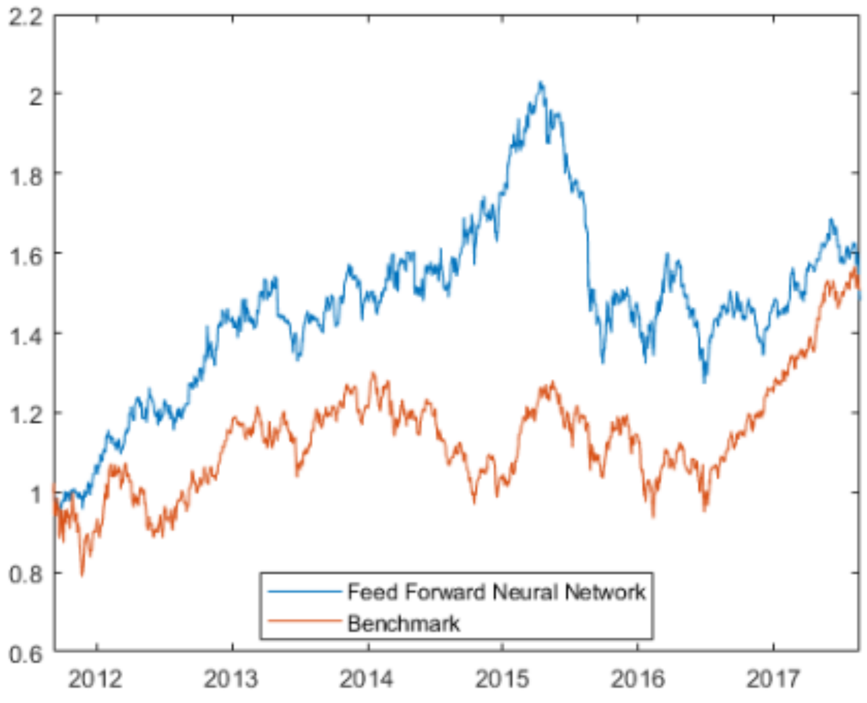


Figure 99: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – ATX

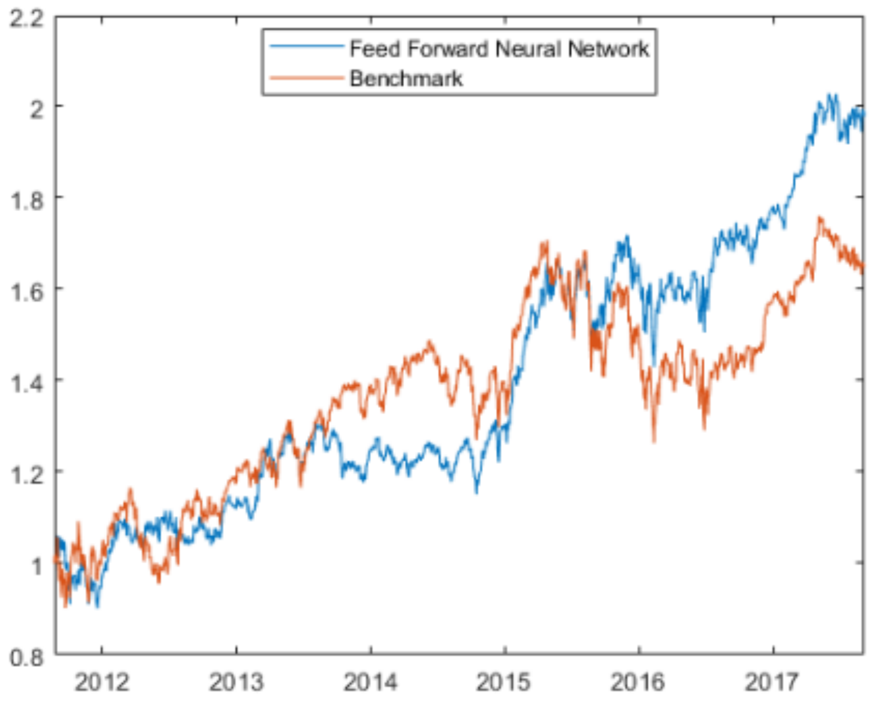


Figure 100: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – CAC 40

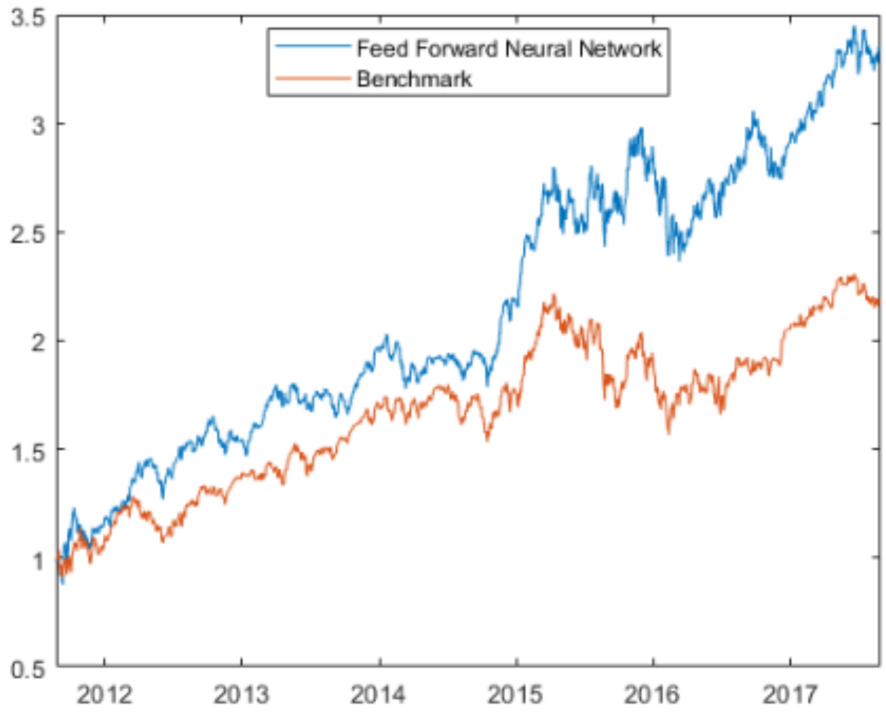


Figure 101: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – DAX 30

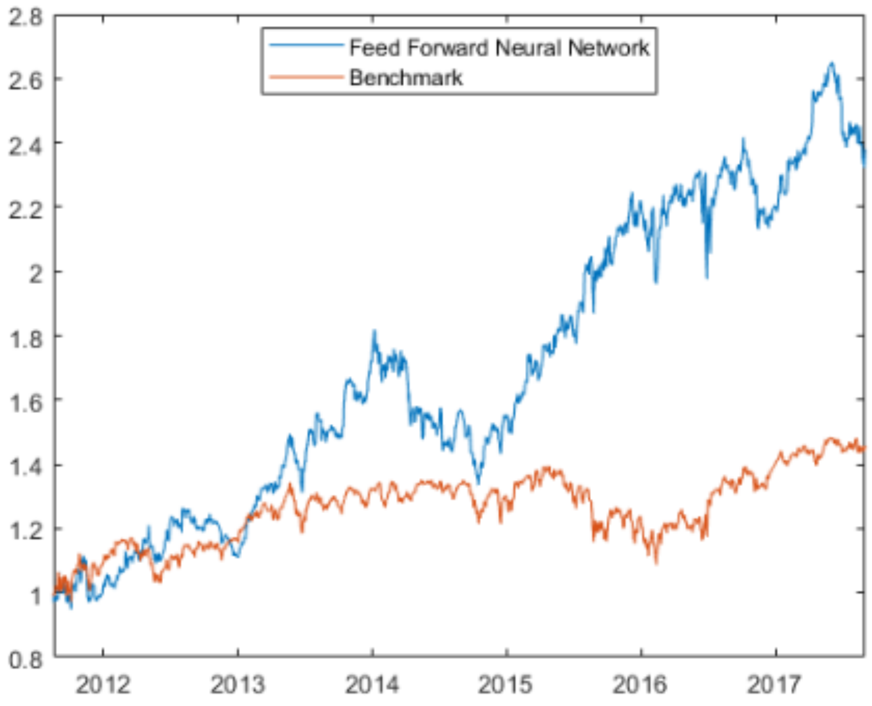


Figure 102: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – FTSE 100

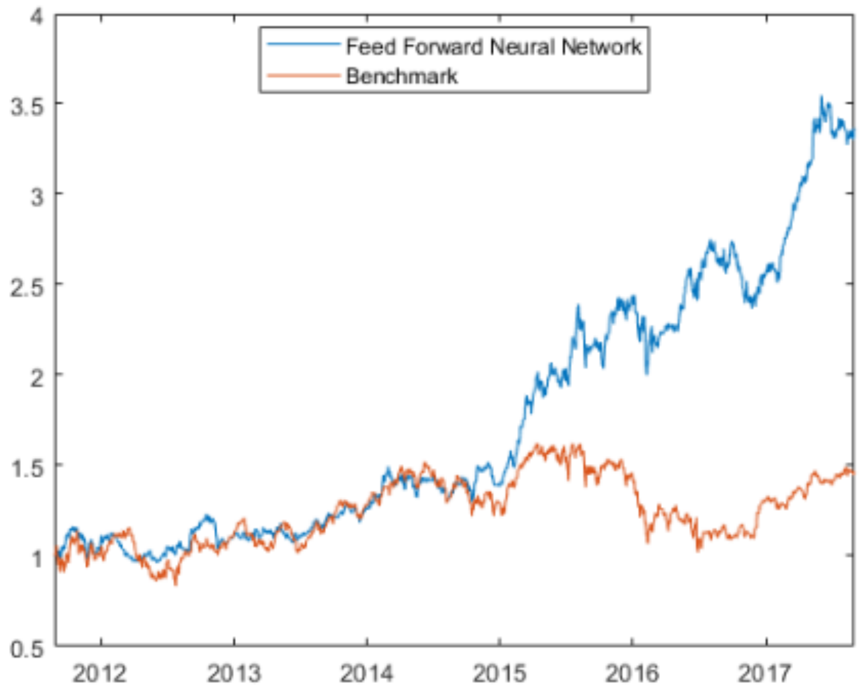


Figure 103: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – FTSE MIB

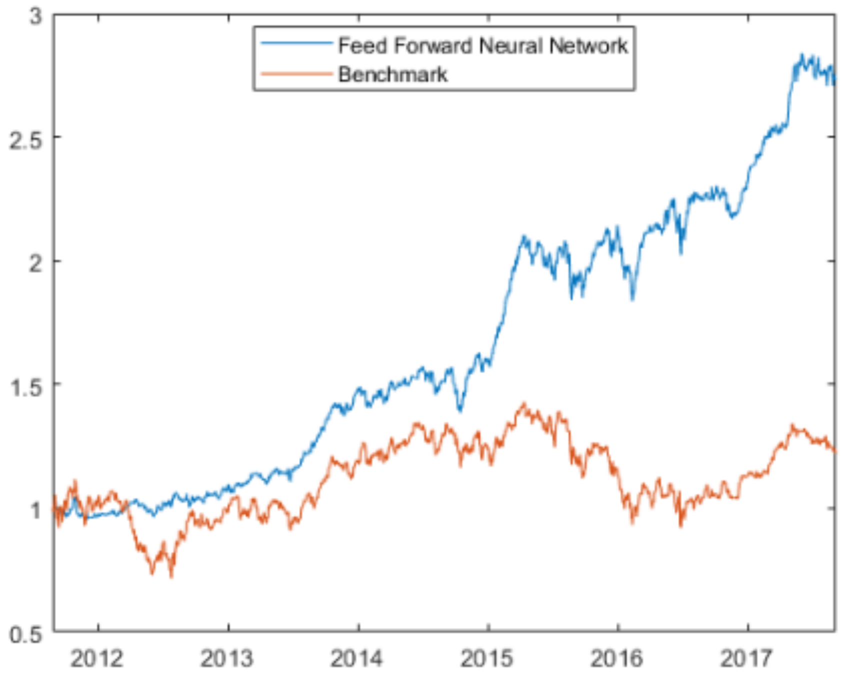


Figure 104: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – Mercado Continuo

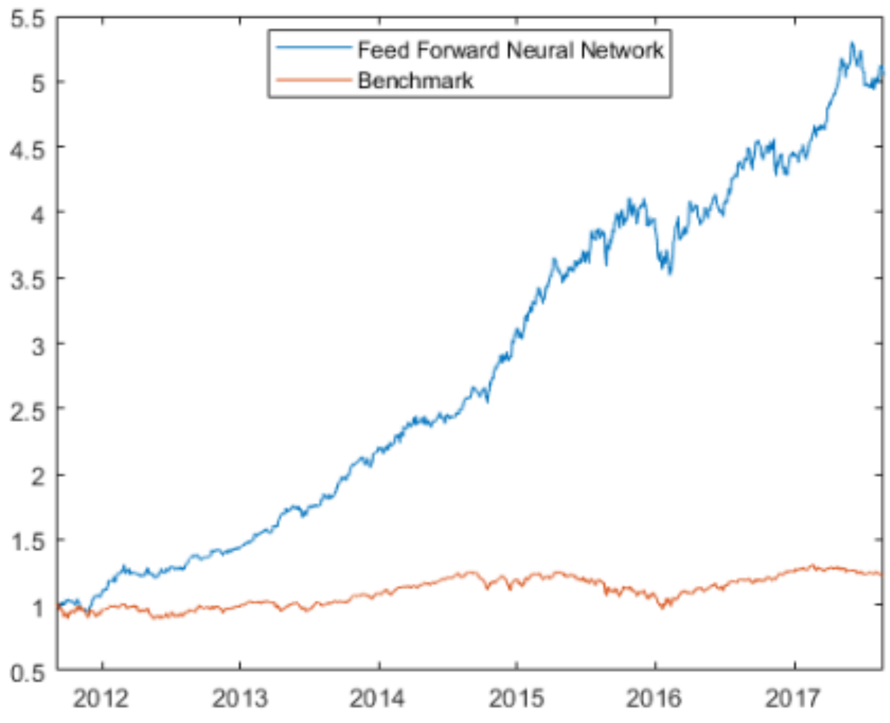


Figure 105: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – S&P TSX 60

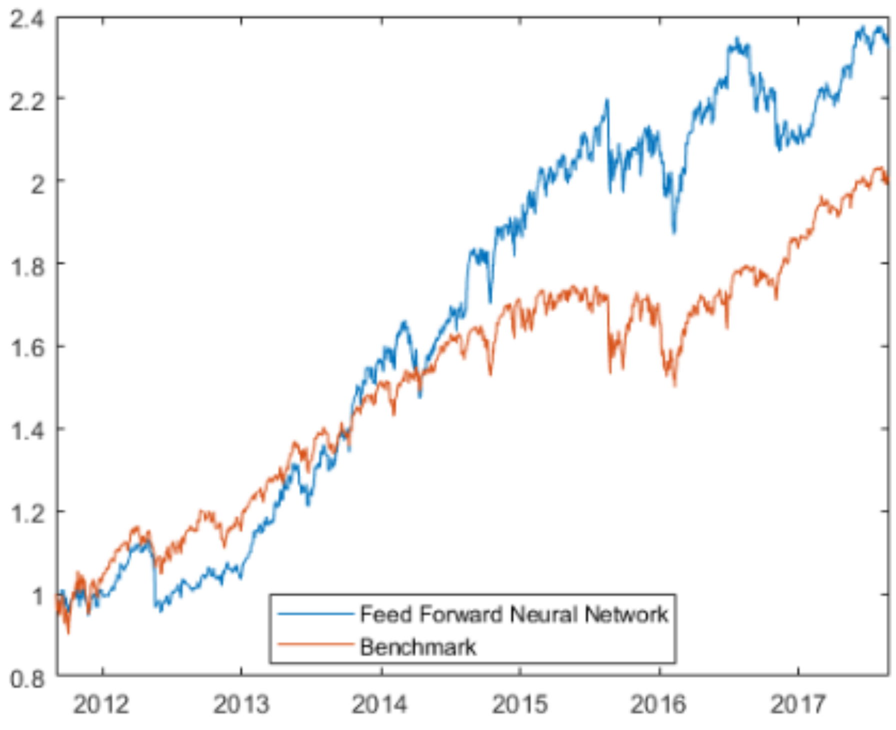


Figure 106: Equity curve – Feed Forward Neural Network Low Beta Strategy – In Sample – S&P 500

Once again, a summary table is shown below with the different indicators obtained,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
ATX	0,4588	14.804.400	37,79%	28,29%	1,97e-02%	-0,22%	1,33%
CAC 40	0,7461	13.787.900	16,85%	26,04%	2,41e-02%	1,09%	0,83%
DAX 30	1,0604	33.443.000	20,65%	29,27%	4,21e-02%	3,17%	0,92%
FTSE MIB	1,0334	16.412.500	18,18%	37,15%	7,11e-02%	3,51%	1,44%
FTSE 100	0,8620	23.787.000	26,59%	22.06%	4,01e-02%	3,74%	0,91%
Mercado Continuo	1,3128	27.381.000	14,32%	35,75%	6,06e-02%	4,03%	1,13%
S&P TSX 60	2,1113	50.384.000	14,39%	22,86%	0,10%	11,01%	0,86%
S&P 500	1,0271	23.189.000	15,71%	14,16%	1,85e-02%	1,70%	0,61%

Table 8: Compilation of Performance indicators – Feedforward Neural Network Low Beta Strategy – In Sample

The analysis of the equity curves leads to the conclusion that during the In Sample period the strategy based on a Feedforward neural network allows to beat the respective benchmarks in a consistent way in all markets.

The various indicators also allow this conclusion to be reached. In practically all cases very good Sharpe ratios are obtained. The information ratios are also particularly good in practically all markets.

A study of the maximum drawdown also shows that in a significant number of cases, 5 out of 8, the objective is achieved with less risk than the benchmark.

Thereafter, the results of the backtest during the Out of Sample period for the momentum strategy with low beta are shown below.

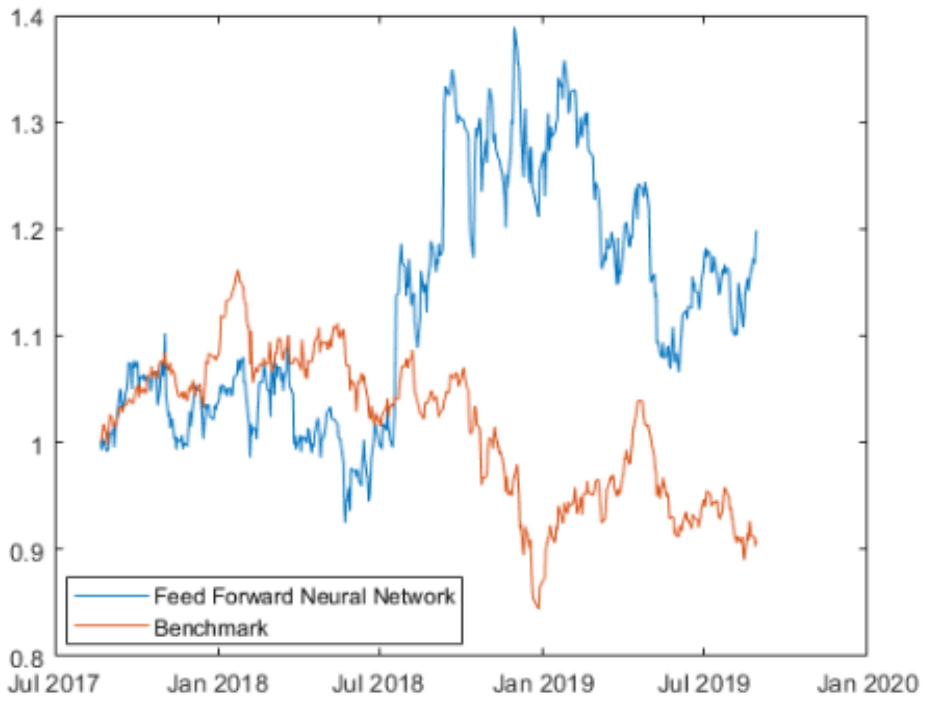


Figure 107: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – ATX

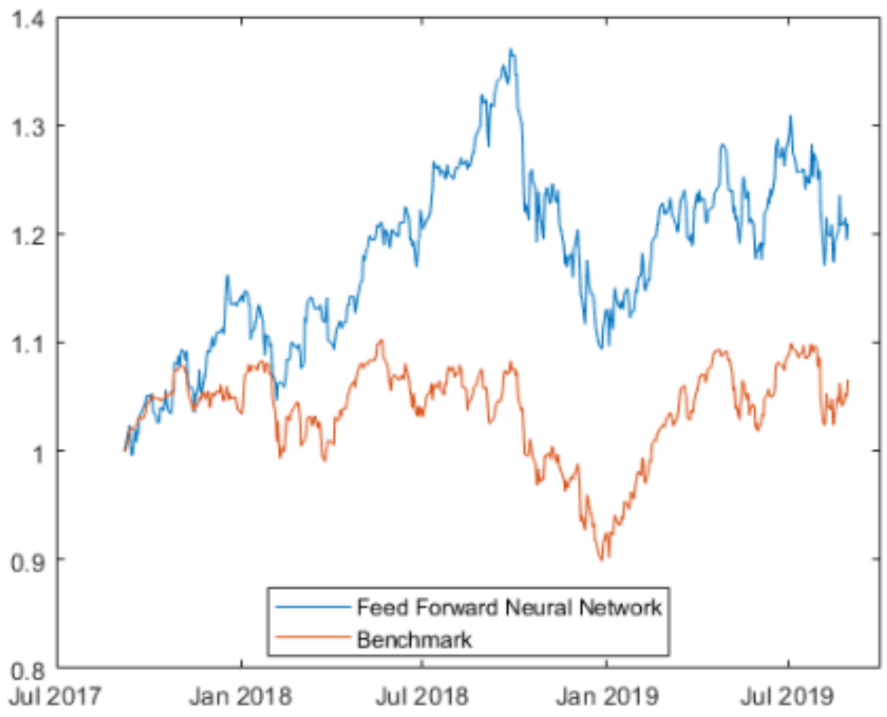


Figure 108: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – CAC 40

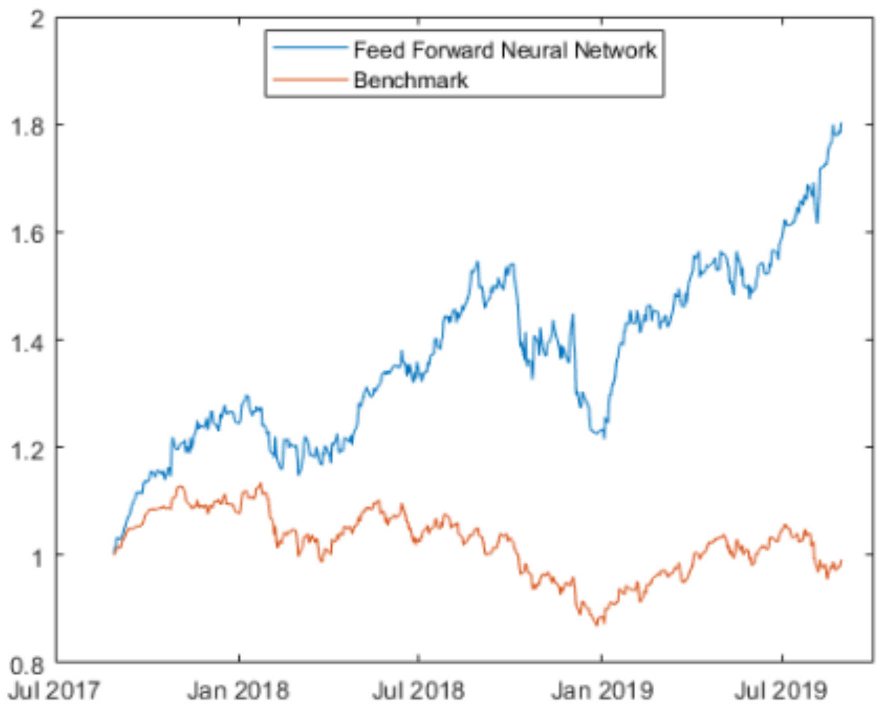


Figure 109: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – DAX 30

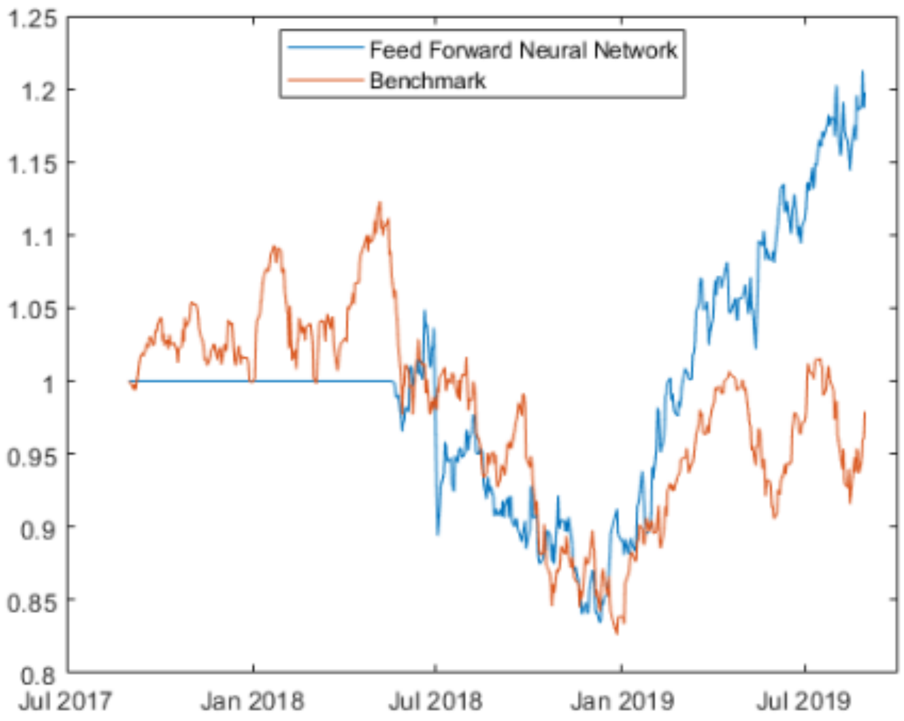


Figure 110: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – FTSE MIB

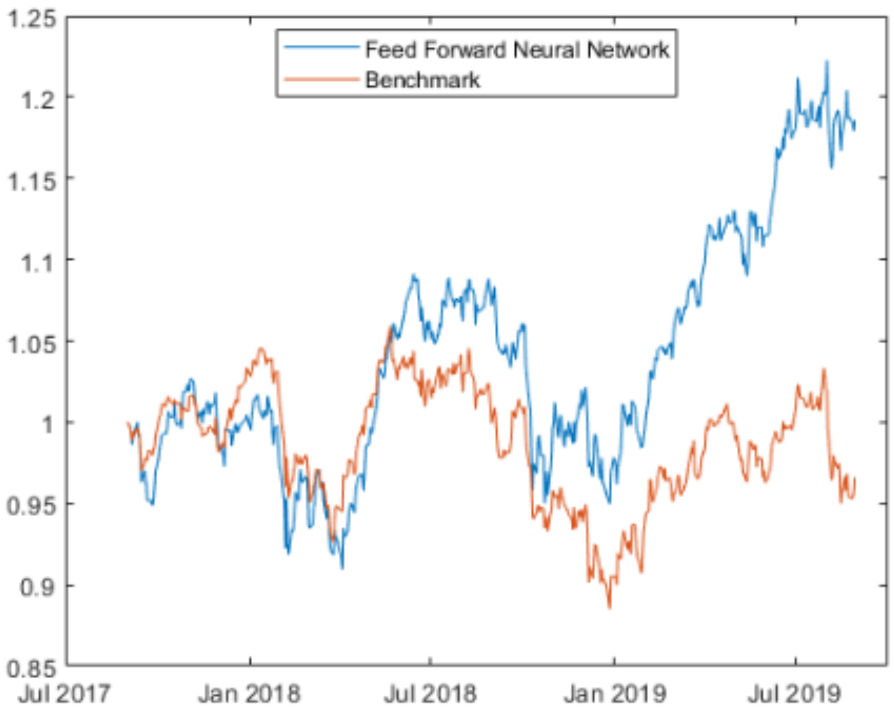


Figure 111: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – FTSE 100

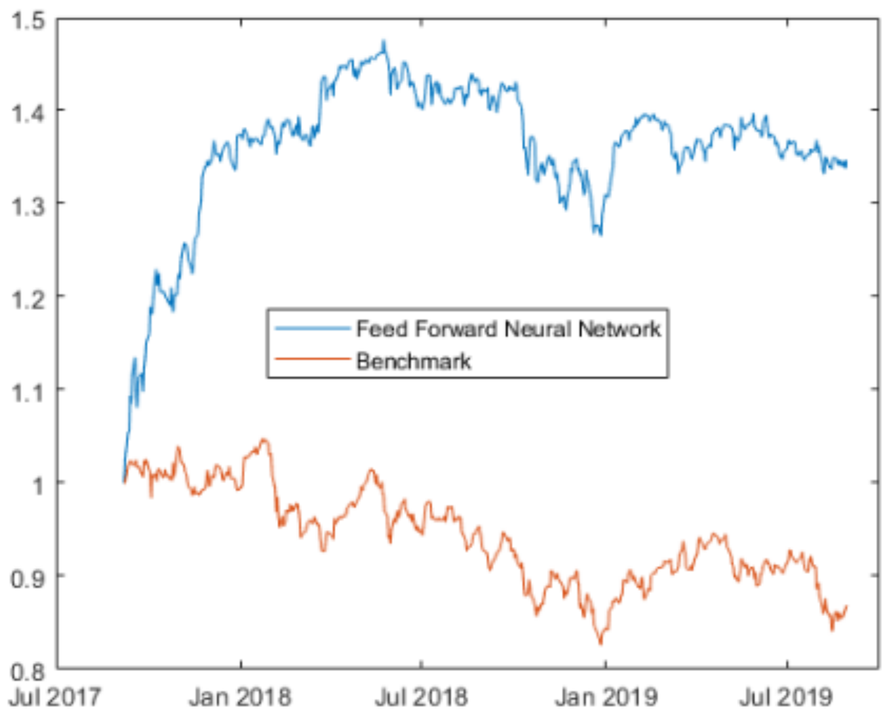


Figure 112: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – Mercado Continuo

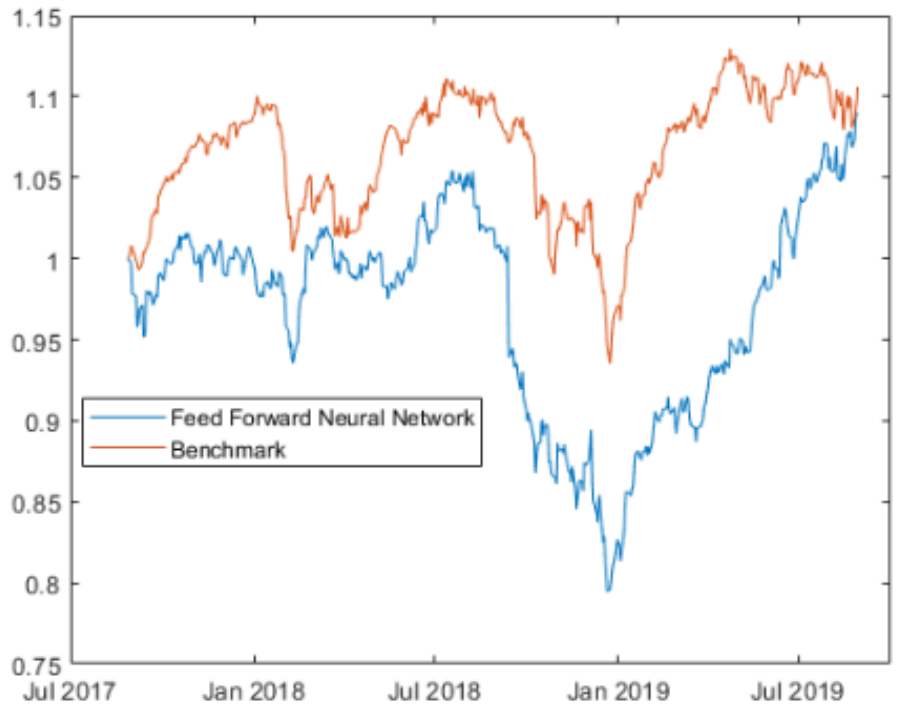


Figure 113: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – S&P TSX 60

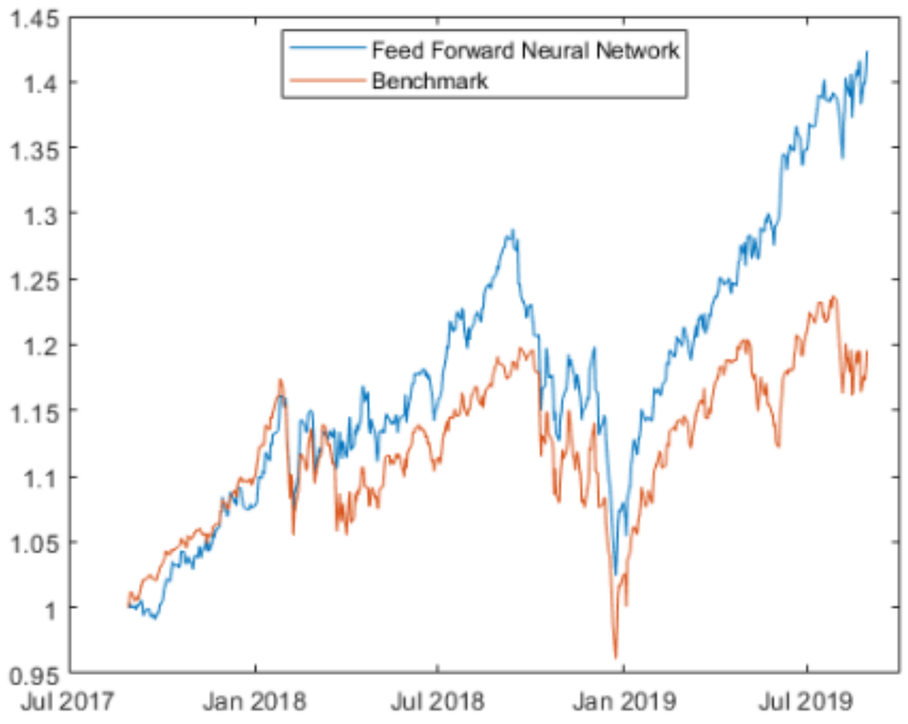


Figure 114: Equity curve – Feed Forward Neural Network Low Beta Strategy – Out of Sample – S&P 500

Thereafter, a summary table is shown below with the different indicators obtained,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
ATX	0,4858	11.974.900	23,25%	27,31%	6,01e-02%	4,45%	1,42%
CAC 40	0,6334	12.081.600	20,22%	18,47%	2,97e-02%	3,35%	0,83%
DAX 30	1,6144	18.007.000	21,40%	23,44%	0,12%	13,14%	0,94%
FTSE MIB	0,6175	11.975.000	20,43%	26,40%	4,12e-02%	3,47%	1,17%
FTSE 100	0,6810	11.844.900	12,95%	16.41%	4,16e-02%	7,31%	0,57%
Mercado Continuo	1,2446	13.451.300	14,28%	21,17%	6,90e-02%	8,92%	0,97%
S&P TSX 60	0,4064	10.883.940	24,63%	15,80%	5,39e-03%	-0,24%	0,70%
S&P 500	1,3604	14.218.000	20,50%	19,78%	4,28e-02%	6,64%	0,51%

Table 9: Compilation of Performance indicators – Feedforward Neural Network Low Beta Strategy – Out of Sample

From the analysis of both the equity curves and the table with the compilation of the different performance indicators, it can be concluded with considerable certainty that the strategy based on Feedforward Neural network with low beta has worked satisfactorily in this backtest. With the exception of the S&P TSX 60, the strategy has outperformed its benchmark in all markets, in some cases significantly so, as can be seen from the charts. In a large number of cases the maximum drawdown is also lower than the index. A glance at the information ratio shows that very good values have been obtained, with the exception of the S&P TSX 60.

It is therefore concluded that the Feedforward neural network strategy manages to beat the market in a consistent way, by beating the index in both the In Sample and Out of Sample periods in the vast majority of tested markets.

The following is a study of how the real-time portfolio management tool has been developed.

5. Real-time Portfolio Management system based on successful strategies

This chapter details how the tool has been created to enable the implementation of the investment strategies seen above and the management of the portfolios in real time. As explained above, a major benefit of using MATLAB for the development of the backtest tool is that the functions and scripts could be reused with a small number of changes for the implementation of the tool in real time.

The main difference of this tool is that it is necessary to obtain the current data every day that an investment is to be made. To do this, the tool must be connected to the Internet via MATLAB and retrieve prices directly, so that the database seen at the beginning of the work is automatically done in MATLAB. The decision was made to use Yahoo - Finance for this, as the connection to their servers for this type of application is made in many academic works. This is because they have the daily prices and other indicators of a very important number of shares. In particular, they have all the necessary indicators for this work and they are free of charge. The fact that the real-time implementation is based on a free financial data provider is particularly interesting as it makes the tool more accessible, no subscription to a financial data provider is required.

The first part of this tool is similar to the backtesting tool. It deals with the recovery and pre-processing of the data. It is carried out on the basis of the first two scripts that have been studied in the first tool. The third script is not necessary since, as it is a real-time implementation, it is not necessary to partition the data but rather it operates with all the information available, downloaded in its entirety from Yahoo - Finance.

First of all it is necessary to have in an Excel file the Yahoo - Finance codes associated with the shares from which the data is to be downloaded. This is an own Yahoo - Finance code and is not used anywhere else. The procedure carried out was to create a file with the codes of the shares of each index in which the investment is wanted. This file only contains the codes in the first column and associated with each code the name of the stock.

The first of the scripts is responsible for retrieving the closing prices, by connecting to Yahoo and saving this data in *Timetable* format, the same format as the one used in the backtesting tool.

This script also follows the modular structure that has been common throughout this work. Each of the modules is responsible for creating the request to Yahoo from the codes saved in the excel files, downloading the data and putting it in the right format, this for each of the indexes in which the user wants to trade.

The modular structure has an additional advantage in this script, it allows in an extremely simple way to add new shares to the investment universe, simply by adding new symbols in the corresponding Excel file, and even add new entire financial markets, in this case the Nasdaq 100 has been added. It has been used as a substitute for the S&P 500 to form a more compact universe, only with the shares of the Nasdaq 100, Ibex 35, CAC 40 and DAX30. When adding a new index it is not possible to use directly the Machine Learning and Deep Learning strategies since the algorithms trained for these actions are not available. In order to be able to implement strategies based on artificial intelligence algorithms, an investment universe has been created with the same indexes used in backtesting, i.e the S&P 500, ATX, CAC40, DAX 30, FTSE MIB, FTSE 100, Ibex 35 and S&P TSX 60.

The first step of this script is as usual to clean up the Command Window and the Workspace, then to define the start and end periods for which the Yahoo data is to be downloaded. Because one of the indicators used in the strategies is based on the annual return, the algorithm needs to have data for at least a whole year before it can work. In this case the start date is set at 3 September 2018. As for the end date, in this case it is not specified, in that case the algorithm downloads the data available until the current date, or the most recent data available on Yahoo - Finance.

Load Data

The objective of this file is to load historical prices into MATLAB work space and store them in TimeTable format.

We downlaod the data directly from Yahoo Finance.

```
clear;
close all;
clc;


startDate = '3-Sep-2018';
%endDate = '27-Aug-2020';
```

The following module is performed for each of the indices contained in the investment universe. In this case it is also done for the shares that make up the index and for the benchmark index itself, i.e. in the case of the DAX 30, it is called once for the 30 shares that compose it and again for the prices of the index itself. This is necessary to calculate the beta of each stock, as the daily returns of the reference index are needed.

This code downloads from the associated Excel document the Yahoo symbols for each stock and its name. It then calls the GetMarketData() function via a for loop. This function carries out the entire process of generating the request to Yahoo and downloading the data. It also manages that the connection to Yahoo's website remains open during the process. This is a complicated function due to the aspects of connection to a web page. To carry out this process the function and explanation provided on the MATLAB website by the developer Artem LensKiy in his post "Yahoo Finance and Quandl data downloader" has been used.

This function has as entries the day from which the data is to be obtained and the symbol of the stock for which the data is to be downloaded. It returns a Timetable with all the daily historical prices available on Yahoo - Finance i.e. date, opening price, high, low, closing price, adjusted closing price and daily volume. In this case only the closing price is needed so in the for loop only the column containing that information is selected.

Once the data has been downloaded the for loop makes the necessary adjustments so that the daily closing prices of each share are stored in a Timetable as shown below.

DAX 30

```
DAX30 = readtable('DAX30 Symbols.xlsx', 'PreserveVariableNames',true);
DAX30Symbols = DAX30(:,"Symbol");
DAX30Symbols = table2array(DAX30Symbols);
DAX30Names = DAX30(:,"Name");


nStocksDAX = length(DAX30Symbols);
```

```
for i = 1: nStocksDAX
    data1DAX = getMarketData(DAX30Symbols{i}, startDate);
    data2DAX = array2timetable(data1DAX.Close,'RowTimes',
data1DAX.Date);
    data2DAX.Properties.DimensionNames{1} = 'Date';
    data2DAX =
data2DAX(~ismember(dateshift(data2DAX.Date,'start','day'),datetime(holi
days,'ConvertFrom','datenum')),:);
    DataDAX(:,i) = data2DAX;
end
allVars = 1:width(DataDAX);
DAX30_Data = renamevars(DataDAX, allVars, DAX30Symbols);
```

DAX30 Index

```
DAX30_Bench = readtable('Bench_DAX30.xlsx',
'PreserveVariableNames',true);
DAX30_Bench_Symbols = DAX30_Bench(:,"Symbol");
DAX30_Bench_Symbols = table2array(DAX30_Bench_Symbols);
DAX30_Bench_Names = DAX30_Bench(:,"Name");


nStocksDAX30_Bench = length(DAX30_Bench_Symbols);


for i = 1: nStocksDAX30_Bench
    data1DAX30_Bench = getMarketData(DAX30_Bench_Symbols{i},
startDate);
    data2DAX30_Bench =
array2timetable(data1DAX30_Bench.Close,'RowTimes',
data1DAX30_Bench.Date);
    data2DAX30_Bench.Properties.DimensionNames{1} = 'Date';
    data2DAX30_Bench =
data2DAX30_Bench(~ismember(dateshift(data2DAX30_Bench.Date,'start','day
'),datetime(holidays,'ConvertFrom','datenum')),:);
    DataDAX30_Bench(:,i) = data2DAX30_Bench;
end
allVars = 1:width(DataDAX30_Bench);
DAX30_Bench_Data = renamevars(DataDAX30_Bench, allVars,
DAX30_Bench_Symbols);
```

Finally, the relevant data is stored in a matrix calleddata1.mat.

This concludes the download of the data from yahoo. The data is then pre-processed using the second script, which is very similar to the one developed for this purpose in the backtesting tool.

The first step is to clean up the Command Window, the Workspace and load the data saved into the data1.mat matrix.

Data Preprocessing

In this file, we are going to perform data preprocessing, including, data cleaning, data factor creation, data reformatting, and data partitioning.

```
clear;
close all;
clc;
load data1.mat
```

The next step is to standardize the data downloaded from each index. One of the requirements to form the investment universe and for the system to be able to generate buy signals in all the shares in a systematic way is to gather all the data and indicators of each share in the same matrix. However, not all the financial markets are open on exactly the same days, so by merging the data it is possible that empty rows are generated. Another possible problem is that the market is actually open but Yahoo does not have this data for some reason. As there is no way to determine in advance what the reason is for the missing data, it has been considered that it is better to eliminate the row. It is preferable that the system does not operate one day and that it does so the next day if the signal is still active, rather than generating a signal when the market is closed. What is done therefore is to synchronize all the Timetables, eliminate those rows that are empty and then synchronize individually the Timetable of each index with the definitive date vector. As shown below.

Standardize

We standardize the timetables so they have the same days.

We choose to remove the rows that have NaN, instead of interpolate the data, as it is better to don't trade one day (even if it was possible), than to make a trade when the market was actually close.

```
SynchMat = synchronize(CAC40_Data, ATX_Data, SP500_Data, FTSE100_Data,
FTSEMIB_Data, Nasdaq100_Data, Ibex35_Data, DAX30_Data);
SynchMat = rmmissing(SynchMat);

DefDays = SynchMat.Date;
CAC40_Data = retime(CAC40_Data, DefDays);
Ibex35_Data = retime(Ibex35_Data, DefDays);
Nasdaq100_Data = retime(Nasdaq100_Data, DefDays);
FTSE100_Data = retime(FTSE100_Data, DefDays);
ATX_Data = retime(ATX_Data, DefDays);
SP500_Data = retime(SP500_Data, DefDays);
FTSEMIB_Data = retime(FTSEMIB_Data, DefDays);
DAX30_Data = retime(DAX30_Data, DefDays);
```

The second step is to call the Preprocessing function, following the same strategy as in backtesting, the function is called once for each index. Its inputs are the Timetable with the closing prices, the number of shares and the Timetable with the closing prices of its reference index. The function returns the updated Timetable with all the necessary indicators calculated, a matrix with the closing prices, another matrix with the extended closing prices, similarly to during the backtesting tool this matrix is more suitable for the calculation of the mean and covariances of the returns during the optimization of the portfolios. It also returns the dates of the Timetable, and the matrix with the standardised indicators and in the format suitable for neural network strategies.

We call the function Preprocessing with each of the timetables

```
[Nasdaq100_Data, Nasdaq_prices, Nasdaq_Days, Nasdaq_x, Nasdaq_y,
Nasdaq_prices_stats] = Preprocessing(Nasdaq100_Data, nStocksNasdaq,
Nasdaq100_Bench_Data);

[Ibex35_Data, Ibex_prices, Ibex_Days, Ibex_x, Ibex_y,
Ibex_prices_stats] = Preprocessing(Ibex35_Data, nStocksIbex,
Ibex35_Bench_Data);

[CAC40_Data, CAC_prices, CAC_Days, CAC_x, CAC_y, CAC_prices_stats] =
Preprocessing(CAC40_Data, nStocksCAC, CAC40_Bench_Data);

[DAX30_Data, DAX_prices, DAX_Days, DAX_x, DAX_y, DAX_prices_stats] =
Preprocessing(DAX30_Data, nStocksDAX, DAX30_Bench_Data);

[ATX_Data, ATX_prices, ATX_Days, ATX_x, ATX_y, ATX_prices_stats] =
Preprocessing(ATX_Data, nStocksATX, ATX_Bench_Data);

[FTSEMIB_Data, FTSEMIB_prices, FTSEMIB_Days, FTSEMIB_x, FTSEMIB_y,
FTSEMIB_prices_stats] = Preprocessing(FTSEMIB_Data, nStocksFTSEMIB,
FTSEMIB_Bench_Data);

[FTSE100_Data, FTSE100_prices, FTSE100_Days, FTSE100_x, FTSE100_y,
FTSE100_prices_stats] = Preprocessing(FTSE100_Data, nStocksFTSE100,
FTSE100_Bench_Data);

[SP500_Data, SP500_prices, SP500_Days, SP500_x, SP500_y,
SP500_prices_stats] = Preprocessing(SP500_Data, nStocksSP500,
SP500_Bench_Data);

save('data2.mat');
```

Within this function, possible missing values are first eliminated, this should not happen because of the step that has been done before, however it is better to do it. Then the prices of the reference index are synchronised with those of the shares which make it up.

```
function [T, prices, Days, x, y, prices_stats] = Preprocessing(T,
nbStocks, T_index)

    %% Data cleaning: missing values could be because the company
wasn't yet listed

    T = fillmissing(T,"constant",0);

    nbDaysIndex = height(T_index);
    nbDays = height(T);
    days = T.Date;
    T_index = retime(T_index, days);
```

We then start with the creation of the indicators, firstly the time indicators. This is done in the same way as in the case of backtesting.

```
    % Factor Creation
    % Temporal factor

    [T.y,T.m,T.d] = ymd(T.Date);

    % Create day of the week where day 1 of the week is Sunday

    T.dayOfWeek = day(T.Date,'dayofweek');

    % Number of days to next trading day and number of day from
previous

    dayDiff = diff(datenum(T.Date));
    T.nNextTD = [dayDiff; NaN];
    if T.dayOfWeek(end) == 6
       T.nNextTD(end) =3:
    elseif T.dayOfWeek(end) == 7
       T.nNextTD(end) = 2;
    else
       T.nNextTD(end) = 1;
    end
    T.nPreviousTD = [NaN; dayDiff];
```

The next indicators computed are the different returns. This is done following the same code as in the backtesting tool.

```
% Generate price related indicators as predictors
    % Generate n-day returns

    nDay1 = [1, 126, 252];
    [~,n] = size(T);
    for i = 1:numel(nDay1)
        for j = 1:(n-6)
            factorName =
append(string(T.Properties.VariableNames(j)),['Ret'
num2str(nDay1(i))]);
            T.(factorName) = nDayReturn(T.(j), nDay1(i));
        end
    end
```

The function nDayReturn is also the same auxiliar function included in the previous chapter.

The next indicator is the RSI. In this case there is no database with the daily RSI of each stock so the MATLAB function, called rsindex() is used. This is one of the functions available in the Financial Toolbox and it only requires the daily stock prices and the time window desired, in this case 14 days. It is calculated with the following for loop, which is also used to add it to the Timetable.

```
% Generate 14-day RSI


    for s = 1:nbStocks
        T.(append(string(T.Properties.VariableNames(s)),'RSI')) =
rsindex(T.(s), 'WindowSize',14);
    end
```

The next indicator is the MACD, which is obtained in the same way as in the backtesting tool mentioned above.

```
% Generate MACD

    for r = 1:nbStocks
        T.(append(string(T.Properties.VariableNames(r)),'MACD')) =
macd(T.(r));
    end
```

The last indicator is the beta of each share. In this case too, this indicator is not available in the database as it was for the backtest, so it has to be calculated. The beta formula mentioned above is used for this purpose, i.e,

$$Beta = \frac{Covariance(Return_{stock}\ , Return_{index})}{Variance(Return_{index})}$$

In this case, a period of 100 sessions is used to calculate the beta. Then, an average is made to have a more stable value, just as it was performed during the backtest. In this case the average is from the last 152 sessions, approximately 7 months. In this way the beta values are available after one year from the beginning of the data. Below is shown how the beta has been obtained for each share.

```
% Generate Beta


   retindex = nDayReturn(T_index.(1), 1);
   hasRet1 = ~cellfun('isempty', regexp(T.Properties.VariableNames,
'Ret1', 'once'));
   Ret1 =  T(:,T.Properties.VariableNames(hasRet1));
   hasRet126 = ~cellfun('isempty',
regexp(Ret1.Properties.VariableNames, 'Ret126', 'once'));
   Ret1(:,Ret1.Properties.VariableNames(hasRet126)) = [];
   Ret1 = table2array(Ret1);
   sizeBeta = height(T);
   Beta = zeros(sizeBeta, nbStocks);
   BetaPeriod = 100;
   for g = 1:sizeBeta-BetaPeriod
       for k = 1:nbStocks
           covar =
cov(retindex(g:BetaPeriod+g,1),Ret1(g:BetaPeriod+g,k),'omitrows');
```

```
           Beta(BetaPeriod+g,k) =
covar(1,2)/var(retindex(1:BetaPeriod+g,1),'omitnan');
       end
   end


   BetaAverage = [152];
   for h = 1:numel(BetaAverage)
       for l = 1:nbStocks
           factornamebeta =
append(string(T.Properties.VariableNames(l)), 'Beta');
           T.(factornamebeta) = averageBeta(Beta(:,l), BetaAverage(h),
BetaPeriod);
       end
   end
```

.

The BetaAvg function is the same function used in the backtesting tool and included previously.

Finally, the matrixes with prices and dates are extracted from the final Timetable and divided into predictors and responses. It is also standardized and put in the appropriate format for strategies based on neural networks, using the same procedure as in the case of backtesting.

```
% Split data into x (predictors) and y (response)

    prices_stats = zeros(height(T), nbStocks);
    for p = 1:nbStocks
        prices_stats(:,p) = T.(p);
    end

    T = rmmissing(T);
    T = timetable2table(T);
    nd = height(T);
    prices = zeros(nd,nbStocks);
    Days = T.Date;
    T.Date = []; % Remove time
    for b = 1:nbStocks
        prices(:,b) = T.(1);
        T.(1) = []; % Remove prices
    end

    % We also remove beta for the Deep Learning Strategies as it won't
be
    % useful for the neural networks

    T2 = T;

    hasbeta = ~cellfun('isempty', regexp(T2.Properties.VariableNames,
'Beta', 'once'));
    T2(:, T2.Properties.VariableNames(hasbeta)) = [];
    x = T2{:,1:end-nbStocks};
    y = T2{:,end-nbStocks+1:end};

    % Standardize training data (predictors) by column

    [nX,nFeatures] = size(x);

    [x, mu, sigma] = standardize(x);

    %Convert from numerical array to cell format for LSTM.

    x = mat2cell(x', nFeatures, ones(nX,1))';

end
```

This concludes the download and pre-processing of the data and the portfolio management tool can be developed.

The structure of the script used for real-time portfolio management follows an architecture similar to that used in the different backtesting scripts. The first step is the Setup, then the matrixes are generated with the buy and sell signals of the selected strategy, then the signal matrixes are merged into one, as well as the different auxiliary variables such as the vector with the names etc. Once a single signal matrix is available the PortfolioManagement() function is called. This function and all the functions it contains are exactly the same as those developed and explained in the backtesting, so they will not be detailed in this chapter. Finally, the analysis of the results is also performed in the same way as during the backtesting.

Firstly, the setup consists of cleaning the Command Window, the Workspace, loading the data and calling the setCost function, which is the same as in the backtesting tool. It also initializes the variables that will contain the portfolio and the different lists with the transactions.

Real - Time Portfolio Management

Management of the portfolios following the succesfull investment strategies in real-time.

Setup

```
clear
clc
load data2.mat


setCost;

Cash = [];
P =[];
T =[];
TOut =[];
TClose =[];
```

```
pnl=[];
```

The same functions developed in backtesting are used in the creation of the matrix of buy and sell signals. It is simply a matter of selecting which one to implement. The following is an example of the implementation of the simple momentum strategy.

Creation of the matrix with the signals

We create the signal matrix for each of the index by calling the choosed function for each of them.

```
Nasdaq_nDay = height(Nasdaq100_Data);
[Nasdaq100_signal] = SignalCreation(Nasdaq100_Data, nStocksNasdaq,
Nasdaq_nDay);


Ibex_nDay = height(Ibex35_Data);
[Ibex35_signal] = SignalCreation(Ibex35_Data, nStocksIbex, Ibex_nDay);


CAC_nDay = height(CAC40_Data);
[CAC40_signal] = SignalCreation(CAC40_Data, nStocksCAC, CAC_nDay);


DAX_nDay = height(DAX30_Data);
[DAX30_signal] = SignalCreation(DAX30_Data, nStocksDAX, DAX_nDay);


ATX_nDay = height(ATX_Data);
[ATX_signal] = SignalCreation(ATX_Data, nStocksATX, ATX_nDay);


FTSEMIB_nDay = height(FTSEMIB_Data);
[FTSEMIB_signal] = SignalCreation(FTSEMIB_Data, nStocksFTSEMIB,
FTSEMIB_nDa);


FTSE100_nDay = height(FTSE100_Data);
[FTSE100_signal] = SignalCreation(FTSE100_Data, nStocksFTSE100,
FTSE100_nDay);


SP500_nDay = height(SP500_Data);
[SP500_signal] = SignalCreation(SP500_Data, nStocksSP500, SP500_nDay);
```

All the individual variables are then merged to form a single investment universe so that the PortfolioManagement() function can manage the investment in all the stocks in the universe.

Merge

We merge the matrices containing the signals, the prices, the final days we are using as not every stock market opens exactly the same days, the vector containing all the names, the number of stocks and the number of days. The final days should be the same for each market as we have already standardized the data.

```
signal = [ Ibex35_signal CAC40_signal DAX30_signal ATX_signal
FTSEMIB_signal FTSE100_signal SP500_signal];
prices = [ Ibex_prices CAC_prices DAX_prices ATX_prices FTSEMIB_prices
FTSE100_prices  SP500_prices];
prices_stats = [ Ibex_prices_stats CAC_prices_stats DAX_prices_stats
ATX_prices_stats FTSEMIB_prices_stats FTSE100_prices_stats
SP500_prices_stats];
Days = DAX_Days;
names = [Ibex35Names; CAC40Names; DAX30Names; ATXNames; FTSEMIBNames;
FTSE100Names; SP500Names];
names = table2cell(names);
names = names';
nStocks = nStocksIbex + nStocksCAC + nStocksDAX + nStocksATX +
nStocksFTSEMIB + nStocksFTSE100 + nStocksSP500;
nDay = DAX_nDay;
```

At this point the PortfolioManagement() function is called and the results are analysed. The exact same code and procedure is used as for backtesting.

Portfolio Management

```
[P, T, TClose, TOut, newPLTable, pnl] = PortfolioManagement(signal,
prices, Days, names, nStocks, nDay, initialCash, prices_stats, P, T,
TClose, TOut, pnl);
```

Performance analytics

```
portValue = table2array(newPLTable(:,2))+initialCash;
portReturn = tick2ret(portValue);
sharpeRatio = round(sharpe(portReturn,riskFree),4)*sqrt(tDay);
pnlFinal = pnl;
```

Visualize the equity curve wtih initial portfolio value of 1.

```
[Benchmark_Data] = Benchmark(MSCI_Data, Days);


equitycurve = ret2tick(portReturn);
BenchReturn = tick2ret(Benchmark_Data);
Benchmarkcurve = ret2tick(BenchReturn);


figure
plot(Days, equitycurve)
```

```
hold on
plot(Days, Benchmarkcurve)
legend('Machine Learning', 'Benchmark','Location','best')
Max_Drawdown_Strategy = maxdrawdown(equitycurve);
Max_Drawdown_Benchmark = maxdrawdown(Benchmarkcurve);


cash = portReturn;
cash(:) = 0;


Portfolio_Alpha = portalpha(portReturn, BenchReturn, cash, 'capm');
[Information_Ratio Tracking_Error] = inforatio(portReturn,
BenchReturn);
```

The only difference is the benchmark used to compare the portfolio's performance. In this case, as it is a global portfolio, the MSCI World has been used. This is a widely used index as a benchmark for global portfolios and is composed in such a way as to reflect the performance of the financial markets in developed countries.

Below are the results of the real-time implementation of the investment strategies up to 28 August 2020.

It was decided to implement all the strategies using the portfolio optimisation tool as satisfactory results were obtained with less volatility.

First, the pure momentum strategy is shown. To implement this strategy it has been decided to try the compact investment universe. It consists of the shares of the Nasdaq 100, the Ibex 35, the CAC 40 and the DAX30.

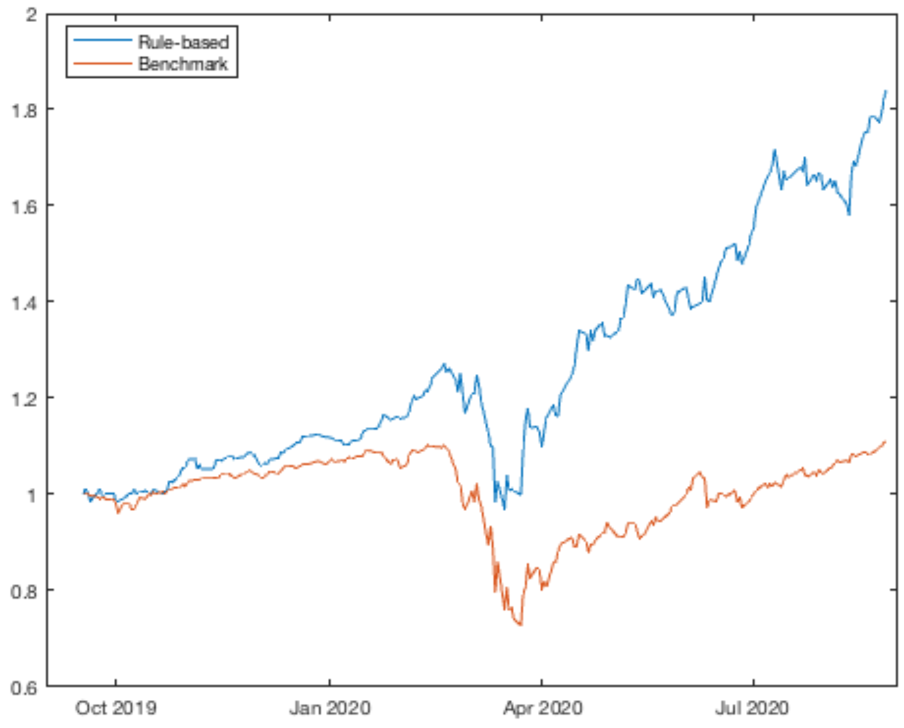


Figure 115: Equity curve Momentum Strategy up to 27 August 2020

The analysis of the results provides the following information,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
Momentum Strategy	2,3018	18.409.400	23,70%	34,01%	0,24%	13,25%	1,64%

Table 10: Compilation of Performance indicators – Momentum Strategy – Up to 27 August 2020

It is clear that the strategy is producing very good results. The equity curve shows that it has beaten the benchmark from the beginning of September to the present. The fall in March corresponds to the severe drop suffered due to the coronavirus. The strategy has been able to obtain very good results by maintaining a lower maximum drawdown, especially during the crisis. Both the Sharpe Ratio and the Information ratio show extraordinary values.

One possible explanation for such positive results is that despite the fall in March, world stock markets have recovered rapidly. In particular momentum-based strategies, which privilege stocks that have done well in the recent past, are doing better than other strategies.

Below are the results of the momentum strategy with low beta also for the smaller investment universe.

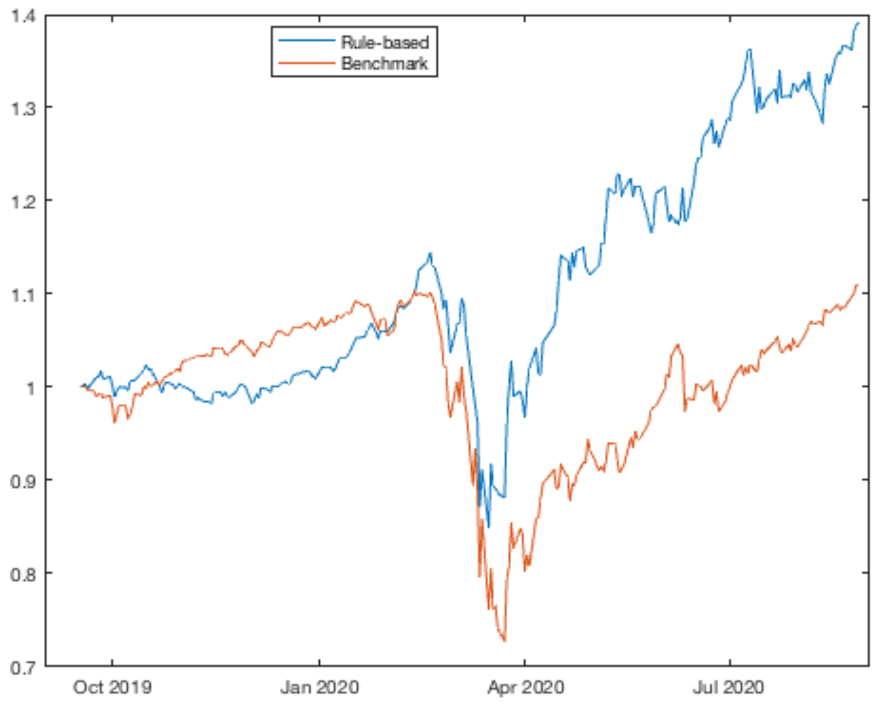


Figure 116: Equity curve Momentum – Low Beta Strategy up to 27 August 2020

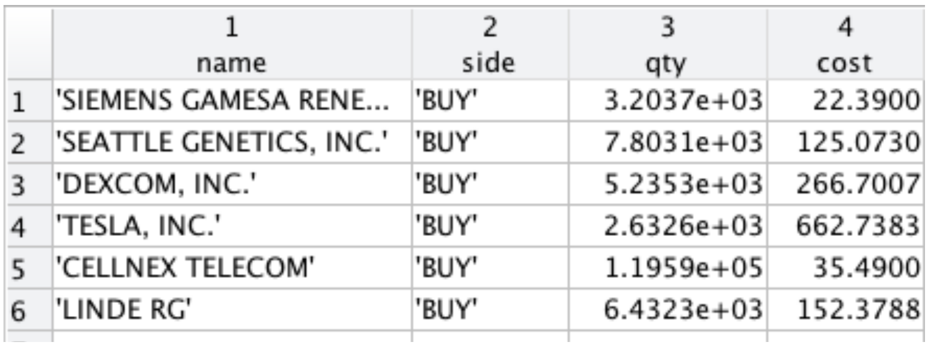
The following table compiles the performance indicators obtained,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
Momentum Strategy – Low Beta	1,4192	13.904.800	25,73%	34,01%	0,12%	6,18%	1,49%

Table 11: Compilation of Performance indicators – Momentum Strategy Low Beta – Up to 27 August 2020

Here too, the strategy has done very well. The equity curve shows how the fall due to the coronavirus was better managed by the strategy than by the index, which has led to a clear outperformance of the strategy from that moment on. This is confirmed by looking at the value of the maximum drawdown and the values of the sharpe ratio and information ratio.

This strategy has achieved results that are not as good as those of the pure momentum strategy. By adding the beta filter, the strategy has been invested in stocks with lower volatility, so it seems that a significant percentage of the good performance of the previous strategy has been driven by those stocks that have seen the greatest increases since the beginning of the coronavirus crisis, such as Tesla for example, which has a significant weight in that portfolio, as can be seen in the Figure 117 representing the portfolio at 27 August 2020 for the pure momentum strategy.



	1 name	2 side	3 qty	4 cost
1	'SIEMENS GAMESA RENE...	'BUY'	3.2037e+03	22.3900
2	'SEATTLE GENETICS, INC.'	'BUY'	7.8031e+03	125.0730
3	'DEXCOM, INC.'	'BUY'	5.2353e+03	266.7007
4	'TESLA, INC.'	'BUY'	2.6326e+03	662.7383
5	'CELLNEX TELECOM'	'BUY'	1.1959e+05	35.4900
6	'LINDE RG'	'BUY'	6.4323e+03	152.3788

Figure 117: Portfolio at 27 August 2020 for the pure momentum strategy

The results of the strategy based on the Naive Bayes classifier are shown below. For this strategy and the following ones based on the neural networks, the expanded investment universe has been chosen, i.e. with the S&P 500, ATX, CAC 40, DAX 30, Ibex 35, FTSE MIB, FTSE 100, as the algorithms are already trained to the stocks belonging to those financial markets.

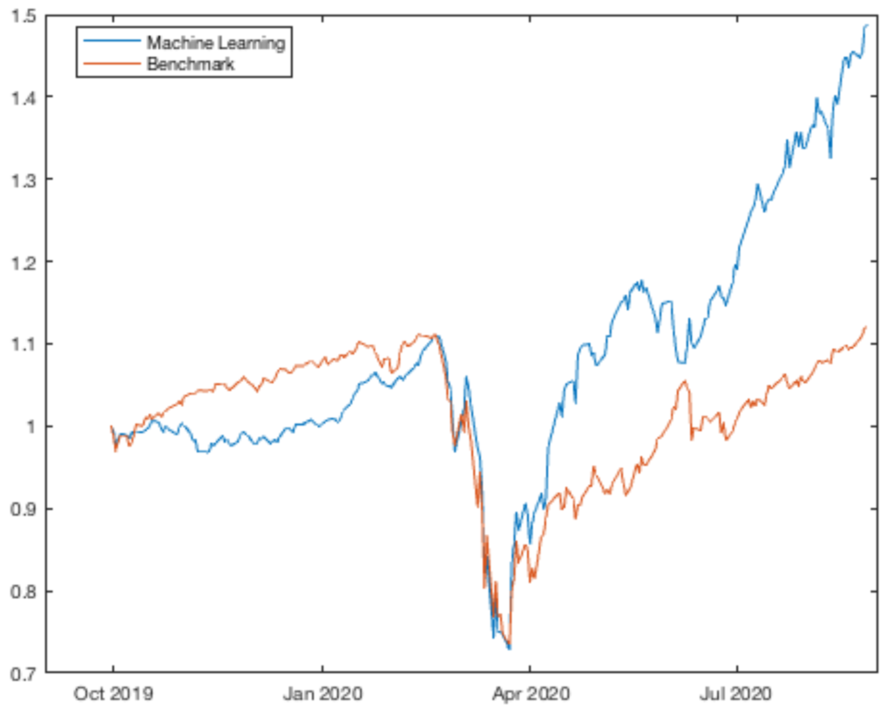


Figure 118: Equity curve Machine Learning Strategy – Up to 27 August 2020

The table with the performance indicators is also shown,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
Machine Learning	1,4097	11.400.800	35,41%	34,01%	0,15%	8,59%	1,55%

Table 12: Compilation of Performance indicators – Machine Learning Strategy– Up to 27 August 2020

In view of the equity curve, it can be concluded that the strategy based on machine learning has adapted very well to the conditions of the financial markets following the coronavirus crisis, as the portfolio has performed very well since the severe fall in March. The Sharpe Ratio and information ratio values are also very good. The strategy, however, did relatively worse than the index in the March fall, reaching a slightly higher maximum drawdown value.

The results obtained with the strategy based on the LSTM neural network with low beta applied to the expanded investment universe are shown below,

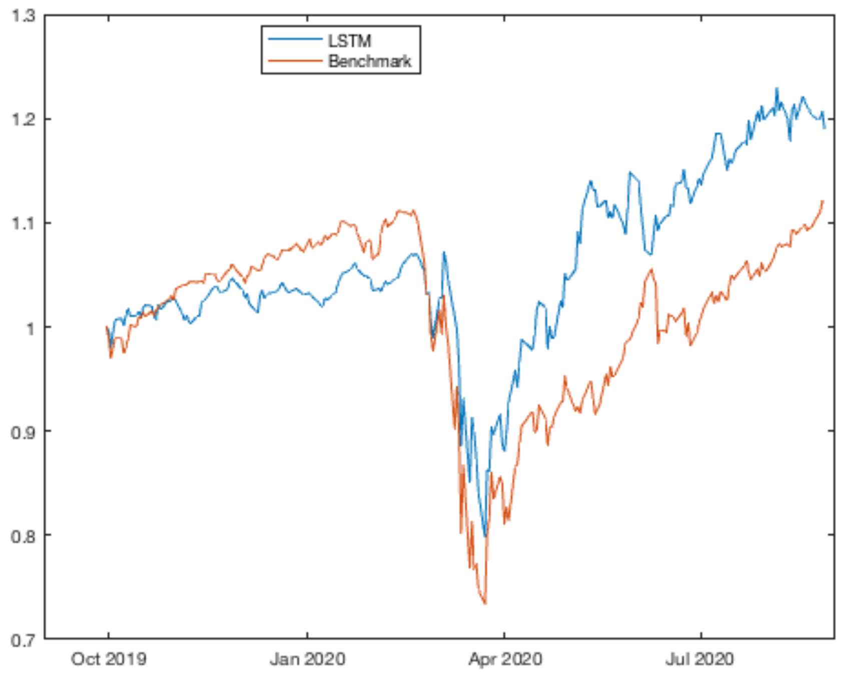


Figure 119: Equity curve – LSTM Low Beta Strategy – Up to 27 August 2020

The analysis of the results provide the following information,

	Sharpe Ratio	P&L Final	Max Drawdown Strategy	Max Drawdown Benchmark	Alpha	Information Ratio	Tracking Error
LSTM – Low Beta	0,8302	11.895.600	25,55%	34,01%	5,02e-02%	1,44%	1,52%

Table 13: Compilation of performance indicators – LSTM Low Beta Strategy – Up to 27 August 2020

The strategy has also adapted well to the new conditions on the financial markets. It managed to outperform the benchmark index and obtained a better maximum drawdown value during the March drop. Thus managing to achieve better returns than the benchmark index by assuming a lower level of risk.

Finally, the results obtained with the strategy based on the FeedForward neural network with low beta applied to the bigger investment universe also are shown below,

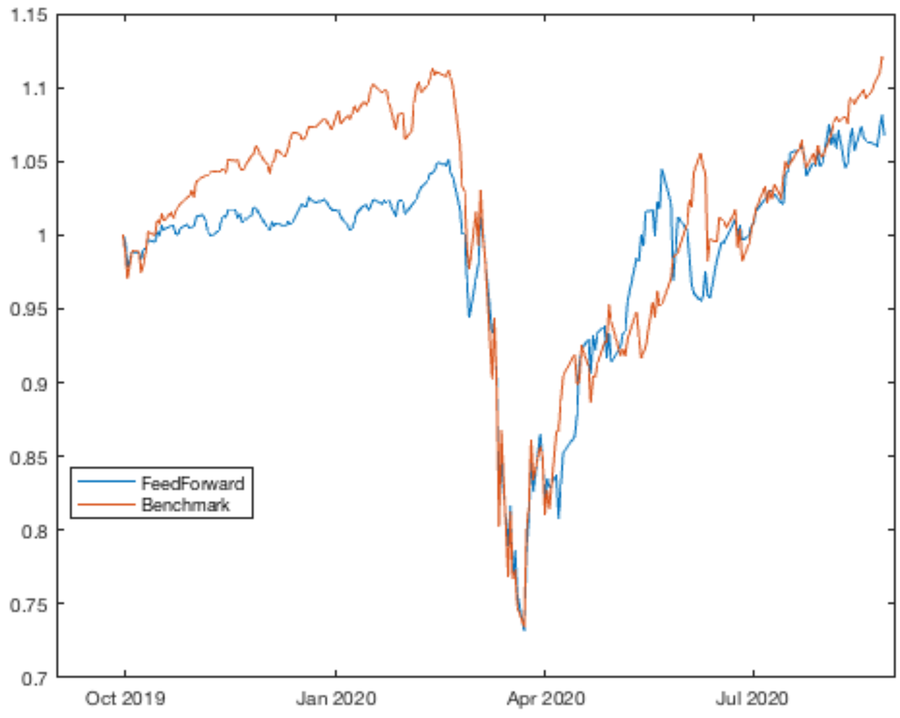


Figure 120: Equity curve . Feedforward Strategy Low beta – Up to 27 August 2020

The table with the performance indicators is also shown,

	Sharpe Ratio	**P&L Final**	**Max Drawdown Strategy**	**Max Drawdown Benchmark**	**Alpha**	**Information Ratio**	**Tracking Error**
Feedforward – Low Beta	0,4080	10.665.860	30,38%	34,01%	3,02e-04%	-1,97%	1,46%

Table 14: Compilation of performance indicators – Feedforward Low Beta Strategy – Up to 27 August 2020

In this case the strategy is not obtaining as good results as in the case of the previous strategies or as during backtesting. Although it is not managing to outperform the benchmark index, it should be noted that the fall suffered in March was not as severe following this strategy as it was for the benchmark. In any case, the results obtained are very similar to those of the index.

Therefore, as of 27 August 2020, of the five strategies that have been implemented, four of them are achieving their initial objective, which is to outperform the benchmark.

6. Conclusions

6.1 General conclusion

The management of a portfolio of equities represents a fundamental part of any investment plan designed to meet an investor's long-term objectives. This is because it is the financial asset that historically provides the highest returns to investors. Due to its importance, a multitude of investment strategies have been developed, covering all possible approaches and of varying complexity. This overwhelming range of possibilities, the apparently growing complexity of the financial markets and the acceleration of the pace at which society is evolving, both on a corporate level and in terms of trends, the development of new technologies, the appearance of new companies and the fall of old ones, etc., make it increasingly difficult to understand and try to develop a vision of what the world may be like in a few decades' time. This phenomenon makes it difficult to commit to a long-term investment strategy for the next 20 or 30 years.

This may be one of the reasons why more and more people are deciding that the smartest solution is to invest passively. Simply buying one or more major indices and holding them no matter what happens over the next few years. In this way the investor avoids having to spend too much time on a problem that seems too complex, predicting what tomorrow's world will be like.

However, this increasing percentage of investment being passively directed, by simply investing according to the size of the companies, acquiring those which are larger, more important and better known, leads to the emergence of additional investment opportunities.

The main objective developed in this work is to develop investment strategies that are capable of obtaining better results than passive investment in a benchmark index, consistently and with less risk.

In this sense, five strategies have been developed and implemented that meet this objective. These are systematic and automatic strategies, in which the investor does not need to make any individual investment decision. He only has to implement the strategy and commit to it in the long term.

Two of the strategies follow a simple approach, where investment decisions are automatically created when certain pre-established parameters are met. The first, called Pure Momentum, generates buy and sell signals solely based on the stock's momentum. The second also adds a filter for investing in those companies with better momentum and low volatility with respect to the benchmark. It is called momentum and low beta strategy. By following these two strategies, based on simple rules, an investor has been able to outperform the market significantly over the last 9 years, regardless of the financial market in which he prefers to invest. Besides that, both strategies are beating a passive strategy, based on a global benchmark since September in a remarkable way.

The other three strategies are based on artificial intelligence algorithms. This type of algorithms, is capable of adapting in a satisfactory way to any new scenario and conditions of the financial markets. These characteristics make them particularly suitable for managing the ever-increasing levels of uncertainty faced by investors. All three strategies are automatic, so the investor does not need to make individual investment decisions either. The first strategy is based on a Machine Learning algorithm, based on the Naive Bayes classifier. The second strategy uses a Long Short-Term Memory neural network in addition to a Low Beta filter. The third strategy is based on a Feedforward neural network and a Low Beta filter. Here again, the investment following these strategies has provided superior results to those obtained with a passive strategy over the past few years, and continue to do it.

Following one of these strategies any investor is able to obtain results above those of the market and with a lower level of volatility. Consequently, fulfilling their long-term financial objectives.

6.2 Future developments

This work has developed a large part of the investment process that any investor should follow. Starting with researching and creating investment strategies, collecting financial data and developing a database to check their performance, through to managing the evolution of a portfolio of shares in real time. The next interesting step that could be developed in this direction is to develop a tool that allows direct connection with a broker. In this way, once the investment strategy generates an investment signal, it is issued directly to the broker.

Following this process, the whole investment process would be automated. It would be the tool that downloads the current data of each company, takes care of the eventual investment decisions and finally executes them.

With this development, a transversal and automatic tool would be obtained, capable of completely assuring that the investor fulfills his investment objectives.

The second future development focuses on the investment strategies. All those developed in this work are based on the stock price. They are founded on the Momentum and Beta factor. However, there are more factors that explain the return that a stock obtains. In particular the Value and Quality factors usually provide good results when combined with the Momentum and Beta factors.

The Value factor is based on the fact that shares whose companies are cheap in terms of the relationship between the share price and some measure of the company's value, such as its earnings, have historically done better than those that are expensive. The Quality factor is based on the fact that historically companies with good earnings levels and low debt have had higher than average returns.

It is therefore reasonable to consider the development of new investment strategies, on the basis of the strategies developed in this work, which incorporate some form of value and quality criteria. With the objective of obtaining higher returns with less volatility.

The incorporation of indicators that allow to discover those companies that are undervalued from a value and quality standpoint could be done both from a rules-based strategy point of view, and based on the artificial intelligence algorithms used in this work.

Finally, the management of the investments has been carried out from a Long - Only perspective. In which a profit is only made by investing in companies that are expected to increase in value in the future. A possible development in this sense would be the incorporation of more sophisticated, although not necessarily better, investment strategies that also incorporate the option of investing in short. This type of investment would have the advantage of obtaining more uncorrelated returns to the market. The objective would therefore be to obtain a similar return to that obtained with traditional stock investment but with lower volatility.

7. References

[1] https://www.researchgate.net/figure/Total-Real-Return-on-10000-Initial-Investment-1802-1997_fig1_334746294

[2] Markowitz, H. (1952). Portfolio selection. Journal of Finance, 7, 77–91.

[3] Treynor, Jack L., Market Value, Time, and Risk (August 8, 1961).

[4] Treynor, Jack L., Jack Treynor's 'Toward a Theory of Market Value of Risky Assets' (Fall 1962).

[5] Sharpe, W.F. (1964), CAPITAL ASSET PRICES: A THEORY OF MARKET EQUILIBRIUM UNDER CONDITIONS OF RISK. The Journal of Finance, 19: 425-442.

[6] Lintner, J. (1965), SECURITY PRICES, RISK, AND MAXIMAL GAINS FROM DIVERSIFICATION*. The Journal of Finance, 20: 587-615.

[7] Mossin, J. (1966) "Equilibrium in Capital Assets Market", Econometrica, Vol. 34, No. 4, pp. 768-783.

[8] Fama, E.F. and K.R. French (1992) "The Cross-section of expected Returns", Journal of Finance, Vol. 47, No. 2 pp. 427-465.

[9] Jegadeesh, N., Titman, S., 1993. Returns to buying winners and selling losers: implications for stock market efficiency. Journal of Finance 48, 65–91

[10] Wilder, W. J. (1978). New Concepts in Technical Trading Systems. Greensboro: Trend Research.

[11] Anson, M. J. P., Chambers, D. R., Black, K. H., and Kazemi, H. (2012). CAIA Level I: An Introduction to Core Topics in Alternative Investments. 2nd Edition. Hoboken, New Jersey: John Wiley & Sons.

[12] Marek, Patrice; Šedivá, Blanka (2017). "Optimization and Testing of RSI". 11th International Scientific Conference on Financial Management of Firms and Financial Institutions.

[13] Tak-chung Fu, Chi-pang Chung, Fu-lai Chung (2013). Adopting genetic algorithms for technical analysis and portfolio management. Computers and Mathematics with Applications, vol. 66, no. 10, pp. 1743-1757, 2013.

[14] Gorgulho, António & Neves, Rui & Horta, Nuno. (2011). Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition. Expert Syst. Appl.. 38. 14072-14085. 10.1016/j.eswa.2011.04.216.

[15] González, Alejandro & Garcia Crespo, Angel & Colomo-Palacios, Ricardo & Guldrís-Iglesias, Fernando & Gómez Berbis, Juan. (2011). CAST: Using neural networks to improve trading systems based on technical analysis by means of the RSI financial indicator. Expert Syst. Appl.. 38. 11489-11500. 10.1016/j.eswa.2011.03.023.

[16] Huang, Yuxuan; Capretz, Luiz Fernando; and Ho, Danny, "Neural Network Models for Stock Selection Based on Fundamental Analysis" (2019). Electrical and Computer Engineering Publications. 170.

[17] Maté, C. (2012). El análisis de intervalos. Aplicaciones en ingeniería. In Anales de mecánica y electricidad (pp. 20-27).

[18] Nassim Nicholas Taleb. Fooled by randomness (2001).

[19] Graham JW. Missing data analysis: making it work in the real world. Annu Rev Psychol. 2009;60:549-576. doi:10.1146/annurev.psych.58.110405.085530

[20] Jilin Zhang Yongzeng Lai Jianghong Lin. The day-of-the-Week effects of stock markets in different countries (2017).

[21] Mark L. Mitchell J. Harold Mulherin. The Impact of Public Information on the Stock Market (1994).

[22] Clifford Asness, Andrea Frazzini, Ronen Israel, and Tobias Moskowitz. Fact, Fiction and Momentum investing (2014).

[23] Juan Pablo Jimeno Moreno. Los mercados financieros y sus matemáticas (2004)

[24] Andrea Frazzini a, Lasse Heje Pedersen. Betting Against Beta (2013).

Alineamiento con los Objetivos de Desarrollo Sostenible de Naciones Unidas

A continuación se realiza una reflexión sobre el alineamiento de los Objetivos de Desarrollo Sostenible de Naciones Unidas con el objeto de estudio de este Trabajo de Fin de Master, cuyo titulo es Portfolio Management System base don technical indicators for crisp and interval-valued data.

En primer lugar se recuerda que dichos objetivos se establecieron el 25 de septiembre de 2015 con el objetivo de hacer frente a los mayores desafíos a los que se enfrenta la humanidad de cara al futuro. Estos desafíos globales se centran en erradicar la pobrezar, proteger el planeta y asegurar la properidad de toda la humanidad. Para que dichos objetivos se cumplan se han especificado metas para cada uno de ellos, las cuales deben cumplirse en un plazo de 15 años.

A continuación se enumeran dichos objetivos,

1. Fin de la pobreza
2. Hambre cero
3. Salud y bienestar
4. Educación de calidad
5. Igualdad de genero
6. Agua limpia y saneamiento
7. Energía asequible y no contaminante
8. Trabajo decente y crecimiento económico
9. Industria, innovación e infraestructura
10. Reducción de las desigualdades
11. Ciudades y comunidades sostenibles
12. Producción y consumo responsables
13. Acción por el clima
14. Vida submarina
15. Vida de ecosistemas terrestres
16. Paz, justicia e instituciones solidas
17. Alianzas para lograr los objetivos

Dos de las áreas más importantes que se buscan con estos objetivos es erradicar la pobreza y disminuir las desigualdades garantizando la prosperidad de todos los habitantes de la Tierra. Se trata en particular de los objetivos 1, 8 y 10. Que el fin de la pobreza este en primera posición no es ninguna sorpresa, se trata de uno de los problemas más importantes y uno de los más antiguos. Si bien es cierto que parece que se están haciendo ciertos progresos en mejorar el nivel de calidad de vida de aquellos que menos tienen, también es cierto que la desigualdad entre aquellos que más tienen y aquellos que menos tienen no para de crecer. Este aumento de las desigualdades ha crecido de forma muy notable a partir de la crisis financiera de 2008 y una de las posibles razones ha sido el importante apoyo que han proporcionado los bancos centrales a los diferentes mercados financieros.

Este fenómeno es relevante ya que la mayoría de los inversores que tienen dinero en acciones o en cualquier mercado financiero suelen ser personas con importantes recursos económicos. Al haber sido las personas más ricas las que se han beneficiado casi exclusivamente de las importantes subidas de los mercados financieros en los últimos años, se ha aumentado notablemente la brecha entre ellos y el resto de ciudadanos que no disponía de dinero invertido en acciones.

Esta desigualdad económica también tiene como consecuencia el aumento de la desigualdad de oportunidades, que pone en peligro directo el correcto funcionamiento de la sociedad, ya que dejan de ser los mejores y más preparados los que tienen acceso a las oportunidades sino aquellos que más tienen.

El objeto de estudio de este trabajo de fin de master es el de desarrollar estrategias de inversión, simples de implementar y sistemáticas que permitan obtener buenos resultados de forma consistente y a largo plazo. El resultado ha sido una serie de estrategias que obtienen muy buenos resultados, relativamente fáciles de implementar y que no requieren de importantes conocimientos, únicamente la voluntad de comprometerse a seguir la estrategia. En este sentido, viendo que los bancos centrales siguen apoyando los mercados financieros, resulta crucial facilitar el acceso a los mercados de acciones no solo a las personas que más recursos tienen sino también a aquellos que más lo necesitan. En los últimos años se han realizado progresos en cuanto a abaratar los costes de la inversión, para que un numero mayor de personas puedan invertir en los mercados financieros. No obstante esta mayor facilidad no siempre se ha traducido en una ganancia económica consistente para los nuevos inversores, debido a que el enfoque de estas plataformas no es el de permitir que sus nuevos clientes mejoren su calidad de vida, si no el que operen más.

Uno de los objetivos de este trabajo de fin de master es por lo tanto proporcionar a los inversores estrategias que les permitan obtener beneficios de forma consistente y en el largo plazo, para que de esta forma todo el mundo pueda aprovecharse de los beneficios de la inversión. Gracias a esto todos los ciudadanos podrán mejorar su nivel de calidad de vida, las desigualdades económicas y de oportunidad se reducirán y la sociedad podrá beneficiarse de las ventajas de un sistema económico y financiero sano con un crecimiento económico sostenible.