Politecnico di Milano
School of Industrial and Information Engineering
Course of Studies in Management Engineering
Dipartimento di Ingegneria Gestionale

POLITECNICO
MILANO 1863

Long Short-Term Memory Networks for Stock Market Predictions and Portfolio Construction

Supervisor: Prof. Andrea FLORI

Graduate Thesis by:
Stefano MORANDI ID. 893807

Academic Year 2018-2019

Al mio papà, grazie per avermi insegnato così tanto. Sono fiero di diventare ingegnere come te.

Oh yes, the past can hurt. But the way I see it, you can either run from it or learn from it.

The Lion King

Acknowledgements

I would like to thank all the people without whom it would not have been possible to reach this important milestone. First of all, thank you *Professor Flori* for your immense helpfulness. Your valuable advice were instrumental in the success of this thesis. Thank you *Mom* for all the affection and attention that you show every day to your children. Thank you for your strength and for your ability to love and make yourself loved. Thanks *Benedetta*, your simplicity and attention to people's needs are unique. Don't give up, something big is waiting for you. Thanks *Marta* for always reminding me that I am a nuisance. The taste you have for beauty fascinates me, I think I know who you got it from. Do not waste it! Thank you *grandmother Carmela* for all the pastas you have cooked and for all your reproaches. I admire the way you have always spent on others. Thanks to those who will participate in this degree from a privileged place. Thanks *dad*, I never thought I would not have the chance to see your complacent gaze for this goal of mine. If I become an engineer, I owe it to the deepest respect that I have always had for you. Thank you for all that you have taught me by your example, for your simple faith and for your passion for beautiful things. Thanks *grandpa Bruno*, you were the embodiment of the concept of preference for me. Thanks for raising me. Thank you *grandma Vica* for your immense culture, for trying to teach me Italian grammar and for your breaded cutlets. Thanks *grandpa Carlo*, even if we never had time to get to know each other well. And finally, thanks to all the *friends* who have endured me.

Ringraziamenti

Vorrei ringraziare tutte le persone senza le quali non sarebbe stato possibile arrivare a questo importante traguardo. Innanzitutto, un grandissimo ringraziamento al *professor Flori* per l'immensa disponibilità dimostratami. I suoi preziosi consigli sono stati fondamentali per la riuscita di questa tesi. Grazie *Mamma* per tutto l'affetto e l'attenzione che ogni giorno dimostri di avere verso i tuoi figli. Grazie per la tua forza e per la capacità che hai di volere e di farti volere bene. Grazie *Benedetta*, la tua semplicità e attenzione ai bisogni delle persone penso siano uniche. Non mollare, ti aspetta qualcosa di grande. Grazie *Marta* per ricordarmi sempre che sono un rompiscatole. Mi affascina il gusto che hai per il bello, penso di sapere da chi l'abbia preso. Non perderlo! Aspetto di poter acquistare un tuo capo di abbigliamento. Grazie *nonna Carmela* per tutte le paste che mi hai cucinato e per tutti i tuoi rimproveri. Ammiro il modo in cui ti sei sempre spesa per gli altri. Grazie a quelli che parteciperanno a questa laurea da una posizione privilegiata. Grazie *Papà*, mai avrei potuto pensare di non avere la possibilità di vedere il tuo sguardo compiaciuto per questo mio traguardo. Se divento ingegnere, lo devo alla profondissima stima che ho sempre avuto per te. Grazie per tutto quello che mi hai insegnato con il tuo esempio, per la tua fede semplice e per la tua passione per le cose belle. Grazie *nonno Bruno*, sei stato per me l'incarnazione del concetto di preferenza. Grazie per avermi cresciuto. Grazie *nonna Vica* per la tua immensa cultura, per aver provato ad insegnarmi la grammatica italiana e per le tue cotolette. Grazie *nonno Carlo*, anche se non abbiamo mai avuto tempo per conoscerci bene. E infine, grazie a tutti gli *amici* che mi hanno sopportato.

Abstract

In recent years, long short-term memory networks (LSTM), due to their sequence learning abilities, have moved the performance frontier forward in a large variety of machine learning (ML) application fields.

The objective of this thesis is to test their effectiveness in stock market predictions and in the construction of an illustrative trading strategy. Specifically, the study does not want to benchmark LSTM to show their edge over other ML models; rather, it aims to deliver useful hints regarding managerial decisions for the implementation of LSTM based trading systems.

In doing so, an original approach providing for different LSTM networks predicting out-of-sample directional movements for 44 stocks in the S&P500 from November 2013 to November 2019 at different horizons is employed. It aims both to deal with the bias by ML trading systems to focus only on returns maximization, neglecting risk considerations, and to facilitate the identification of a forecasting horizon range in which an investment strategy can be profitably performed.

From a data science perspective, results prove LSTM networks' ability in recognizing proper recurrent patterns within stock price time-series: they reach peaks in accuracy higher than 40%. From a financial one, the strategy created thanks to the predictions by LSTM networks delivers positive profits of 78.3%.

Moreover, results challenge the standard training approach, followed also in this thesis, regarding the employment of a dataset composed just by most recent and strictly temporally close to the trading period observations; the evidence creates the conditions to think about the possibility to train specific models with observations

coming from periods characterized by market conditions as similar as possible to the one actually playing in the market.

In conclusion, to train all models, technical indicators and macroeconomic variables have been added to daily returns for the first time in LSTM literature, at least to the best of our knowledge. Anyhow, tests conducted to assess the benefits deriving from their introduction in the input features set raise doubts about their massive use in financial literature.

Sommario

Recentemente, le reti di memoria a lungo termine (LSTM), grazie alla capacità di apprendimento sequenziale, hanno portato ad un incremento delle prestazioni in una grande varietà di campi di applicazione machine learning (ML).
L'obiettivo della tesi è quello di testare la loro efficacia per la previsione del mercato azionario e per la costruzione di una strategia di trading illustrativa. Nel fare ciò, lo studio non intende confrontare le loro performance con modelli benchmark; vuole piuttosto fornire indicazioni rispetto alle decisioni operative da prendere per un loro efficace utilizzo.
Nel lavoro empirico, viene adottato un approccio originale che prevede l'utilizzo di reti con differenti orizzonti di previsione per la predizione del movimento del prezzo di 44 azioni appartenenti al S&P500 da Novembre 2013 a Novembre 2019. L'approccio mira sia ad affrontare la tendenza da parte dei sistemi di trading ML a focalizzarsi sulla massimizzazione dei ritorni, tralasciando valutazioni di rischio, sia a facilitare l'identificazione di un intervallo di orizzonte di previsione adatto alla creazione di una strategia di investimento ML.
Da un punto di vista data science, i risultati ottenuti dimostrano la potenzialità delle reti LSTM nel riconoscimento di pattern all'interno di serie temporali finanziarie: le reti raggiungono picchi di accuracy superiore al 40%. In termini di redditività, la strategia creata offre profitti positivi del 78,3%.
Inoltre, i risultati ottenuti sfidano l'usuale approccio di training riguardante l'esclusivo impiego di osservazioni temporalmente precedenti a quelle del periodo di trading; evidenze fanno pensare alla possibilità di allenare i modelli con osservazioni proveni-

enti da periodi caratterizzati da condizioni di mercato il più simili possibile a quelle presenti durante il periodo di investimento.

Infine, per allenare tutti i modelli, sia indicatori tecnici che variabili macroeconomiche sono stati inseriti nel dataset per la prima volta in letteratura, almeno per quella che è la conoscenza dell'autore. Tuttavia, test condotti per valutare i benefici derivanti dalla loro introduzione sollevano dubbi sul loro massiccio utilizzo in letteratura.

Contents

List of Tables

List of Figures

Executive Summary

Stock market prediction is one of the most difficult and challenging problem to solve, not only within financial markets applications, but among all prediction problems generally. This difficulty mainly comes from countless factors that can affect stock price but, more importantly, from investors' behavior. Investors' rationality is indeed considered the discriminant factor regarding the possibility to eventually predict the stock market.

Two important theories have been presented in financial literature regarding financial markets and the related possibility to predict their future movements: Efficient Market Hypothesis (EMH) and Adaptive Market Hypothesis (AMH). EMH, supposing investors' rational expectations, assumes markets to be efficient and consequently stock prices to incorporate all past and current information. According to EMH, past stock price movements cannot be exploited to predict future price movements since future price movements will be exclusively determined by yet unknown information. For this reason, EMH considers wasted all efforts dedicated to the analysis of financial time-series and abnormal returns impossible to be systematically achieved from it. AMH, instead, tries to reconcile EMH with persistent evidence of irrationality of financial markets. AMH considers indeed, under specific circumstances, the possibility to exploit market weaknesses (i.e. identifiable patterns) and past stock price movements so as to obtain positive abnormal returns.

In such context, backed by evidence and afterwards by AMH, professionals in financial markets have always tried to challenge market efficiency applying different techniques in search of arbitrage opportunities. Particularly, in recent years, machine

learning algorithms have emerged as promising tools to do so. Machine learning models, in fact, have many advantages over traditional econometrics models typically applied for the study of financial time-series: they are assumptions independent, they are able to find patterns and irregularities as well as detecting multi-dimensional non-linear connections in data and, finally, they have the ability to extract rules from large sets of data.

This thesis fits into this stream since it deals with the application of a state-of-the-art machine learning algorithm for the purpose of detecting valuable patterns to predict stock market movement and to build a profitable equity investment strategy. The research question of the thesis is indeed the following: *is it possible to achieve abnormal returns in equity market through the use of a recent and powerful Machine learning technique?*

Literature Review

To answer this research question, in the beginning, the thesis analyzes some of the most promising academic researches dealing with machine learning stock market predictions in a non-systematic literature review (**Chapter 2**). Proving the important gap between academic finance and the financial industry, which already makes extensive use of ML systems, the vast majority of such researches belongs to the computer science field.

In any case, at first, from the review the researchers' purpose to focus on different aspect of machine learning applied to financial time-series emerges: a part of the literature is found oriented to look at ML models as supporting tools for more traditional approaches to investment, rather than proper tools able to extract information from a pool of data by their own; a second and most consistent part, instead, is found focused on the development of trading systems exclusively based on ML algorithms. Furthermore, among this second set of approaches, two subsets are recognised: the first one dedicated to the identification of the most appropriate

ML model, among the ones presented in computer science literature, for signal generation, while the second one more oriented on features selection so as to find the most appropriate variables and techniques able to facilitate information extraction by models themselves.

Even more importantly than that, the most significant evidence emerging from the literature regards the temporal evolution of models employed for stock market predictions. Over the last ten years indeed, it is identifiable a precise trend in literature for which researchers have passed from taking advantage of models such as classification trees or support vector machines to extensively employing neural networks, extending through time, thanks to higher computational power availability, their depth and width. The reason of this evolution has to be found in the incredible suitability of neural networks to cope with non-linear dependencies lying among observations within the dataset. In very recent years, moreover, this trend has further evolved starting to focus on the implementation of recurrent neural networks (RNN) for the creation of profitable trading systems. This progress has to be attributed to the RNN advantage, over standard neural networks, of being specifically designed to extract information from correlated observations such as the ones in time-series. Particularly, among the different variations of RNN, long short-term memory neural networks (LSTM), being yet successfully employed in many fields outside finance, have emerged as state-of-the-art algorithms so much so that they have been started to be employed also in academic financial research.

Throughout the review, the reasons why ML financial literature today founds its research upon LSTM networks results evident. If standard recurrent neural networks have represented a great evolution transforming neural networks from static models able to incredibly approach non-linear dependencies to dynamic models able to do the same from a sequence of correlated observations; possibly, LSTM have moved the performance frontier set by RNN further. Fischer and Krauss (2018) find in their empirical study how LSTM networks over-perform "classic" ML models such as RAF and logistic regression in forecasting stock price movement. But even

more importantly, they find LSTM to be better in forecasting and in generating an investment strategy than deep feed-forward neural networks. Sirignano and Cont (2019) help understanding the results uncovering evidence of non-linearity and long memory in financial time-series. Particularly this last feature results to be the key factor to understand LSTM suitability to financial market predictions. To prove that, Liu and Liu (2018) find indeed how two variations of standard RNN specifically created to consider long-term dependencies such as LSTM and GRU manage to perform better than standard RNN in their empirical research.

Machine Learning for Financial Market Predictions

Subsequently, after the literature and before the presentation of the experimental work, the thesis tries to define a theoretical framework regarding the application of machine learning models for financial market predictions (**Chapter 3**). Even if not containing any research novelty, this attempt has to be considered an important contribution since it merges theoretical concepts from both machine learning and finance: until now, it has not been yet emerged a well-defined field of study combining this two domains, even if their union has been reality in the business world for many years.

The framework evidently begins defining machine learning and the related concept of statistical arbitrage (**Section 3.1**). After that, it focuses on a detailed description of the functioning of all the main algorithms emerged from the literature review starting from Classification Trees (**Section 3.2.1**), passing through Support Vector Machines (**Section 3.2.2**) and Feed-forward Neural Networks (**Section 3.2.3**) to finally arrive to Recurrent Neural Networks with an emphasis on Long Short-Term Neural Networks. All these algorithms are treated with the aim to retrace the history of academic approaches so as to give the reader a clear idea of how ML instruments have evolved in time becoming more and more suited to approach financial time-series. Regarding LSTM networks then, the structure of long short-

term memory cells is thoroughly analyzed so as to give reason, from a structural point of view, for their suitability to deal with long-term dependencies.

The third aspect treated by the framework regards features selection with a synthetic taxonomy of the most frequently employed predictors in financial applications (**Section 3.3**). ML models indeed, requires adequate input variables to properly work. More than that, feature selection is found to be particularly important in financial applications since, the relations between input vector X and output Y (i.e. stock price), do not appear always stable in time and in some case neither intuitive as in other application fields. Since financial markets are driven by investors' behavior, in fact, their irrationality and temporary cognitive biases can create strong links between specific variables and securities' prices only in particular market phase or only after particular information availability. Moreover, the definition of a taxonomy aims then at defining a reference context in which new applicants can look at when designing their input feature set and, at the same time, also to stimulate the search for new possible solutions outside it. The categories presented in the taxonomy are the following: *price and volume variables*, *categorical data*, *technical indicators*, *macroeconomic and commodities variables*, *risk variables* and *sentiment variables*.

The last, and probably most important, facet treated by the framework is the difference, peculiar to ML financial applications, between data science and financial performances of ML models (**Section 3.4**). In fact, there is not always a strict relationship between classic ML performance metrics, typically used as objective parameters to be maximised when training a ML model, and financial performance metrics that are the real evaluation metrics for financial applications. Particularly in classification settings, as the one employed in the experimental phase, an accurate forecasted direction for stock market cannot be directly linked to a positive trading performance: high volatile periods, in fact, are more profitable in this sense than low volatility ones. This is the reason why for ML financial applications it is required to split the validation phase into two steps: a first step more oriented to comprehend

the statistical behavior of the model and a second one focused on the effectiveness of predictions from a financial point of view. One of the main drawbacks related to the computer science literature is exactly the missing financial validation of models trained. Such studies clearly present a flaw in their consistence since they do not prove the real advantage in employing computational expensive model for stock market predictions from an operational point of view.

So, starting from these evidences, the framework at first covers data science concepts such as accuracy, sensitivity, precision, F1-score and ROC; after them, it focuses in presenting the most important investment portfolio performance metrics (e.g. return, standard deviation, Sharpe ratio, value at risk, Omega ratio etc.).

Experimental Settings

Afterwards, thanks to the contextualization provided by the literature review and the theoretical framework, the thesis can actually present to a conscious reader the empirical work performed in order to answer the research question (**Chapter 4**).

Methodologically, differently from the majority of ML financial approaches, the objective of the experimental work in this thesis is the definition of an investment strategy with an horizon of five trading days (**Section 4.1**). In doing so, the study further presents two main novelties with respect to the literature.

The first main novelty regards the implementation of a solution that is arbitrarily referred as "modular classification approach". An approach thought to deal with the bias suffered by ML trading systems for which they tend to maximise returns, without any consideration about the risk related to the investment strategy.

To make the investment strategy as sensible as possible to risk factors, "modular classification approach" provides, at first, for the use of ML classification models; classification setting, compared to regression, is indeed less biased towards more volatile stocks: in a regression settings, stocks selected to enter in the portfolio are the ones whose forecasted returns are higher/lower, conversely, in a classification

approach, they are the ones having a higher probability to belong either to a positive or negative price trend regardless of its magnitude. Secondly, the term "modular" refers to the combination of multiple prediction models: the approach consists in training, in addition to the "expected standard model" having a forecasting horizon of five trading days, one additional model for each remained day prior the day in which all positions have to be closed. This translates into different models, where each of them can be seen as a single module of the whole trading system, trained to predict price movements for decreasing time horizons. In our specific case, for one week-ahead forecasting, the whole approach consists in 5 different LSTM models predicting with a forecasting horizon ranging from five to one days-ahead.

The main rationale behind the modular approach is the following one: as the forecast horizon h decreases, models are expected to become increasingly accurate so that their predictions can be used both to re-calibrate positions erroneously taken by previous models (i.e. diminish the expositions on stocks whose predicted trend has reversed compared to the one predicted by previous model) and to exploit new investment opportunities (i.e. take new positions based on predictions from theoretically more accurate models).

Moreover, and even more interestingly in terms of research, such an approach allows to effectively test the hypothesis for which deteriorated performance are expected with increasing forecasting horizons. Such hypothesis has already been found true in the case of many ML models extensively employed in financial research.

Even more, due to the scarcity of information regarding medium-long term investment strategies based on machine learning models, the approach it is intended to explore, from a managerial perspective, the possibility to indicate a suitable forecasting horizon range in which an investment strategy can be profitably performed.

The second main novelty proposed in the thesis regards instead the extension of the input variable set employed to train LSTM models by means of technical indicators and macroeconomic variables. These two types of variable have been already used to train a large variety of ML models but, to the best of our knowledge,

not yet in the case of LSTM networks.

Clearly, the experiment, in the light of the findings from literature, to implement the "modular approach" takes advantage of predictive power by LSTM networks (**Section 4.2**). Precisely, it deals, as said, with five different LSTM classifying observations among three different classes: *positive* and *negative* significant future price movements and *non-significant* future price movements. Regarding the classes, the introduction of the "central" one containing non-significant price movements is to be considered an additional innovation compared to the majority of approaches in literature.

Despite their different forecasting horizon, all five models employ the same database to forecast future price movements (**Section 4.3** and **4.4**). Overall, it is composed of daily returns, weekly returns, technical indicators (moving average, exponential moving average, MACD, parabolic SAR, RSI, Sochastic %K and %D, commodity channel index, average true range, Bollinger bands, Chaikin's oscillator) and macroeconomic variables (VIX, TED rate, US Treasury, Exchange rate US/EU) referred to 44 stocks out of 500 composing the S&P500 index at the end of 2019.

Regarding the training process (**Section 4.5**), the whole dataset is subdivided in 6 different windows each made up of a training set covering two trading years of observations and of a validation and a trading set covering a single trading year of observations each. In every window, for each of the five models, 5 variations differing for the size of hidden nodes in their single hidden (5, 10, 15, 20 and 25) layer are trained. Then, thanks to the validation set, the most appropriate one is selected to effectively make predictions concerning the trading set.

From a financial point of view (**Section 4.6**), instead, the "modular approach" actualizes in a *long-short equity trading strategy*. Long-short strategies are investment strategies that take long positions (i.e. buy) in stocks expected to increase their value and short positions (i.e. sell) in stocks expected to lose value. Moreover, as a particular case of long-short portfolios, the ones built in this thesis are *dollar-neutral portfolios*; they are portfolios in which the dollar amounts of both long

and short positions, at the time of the creation of the portfolio, is equal so as to make the initial investment for the creation of the portfolio equal to zero (excluding transaction costs).

Experimental Results

From a data science perspective (**Section 5.1**), results prove LSTM networks' ability in recognizing proper recurrent patterns within stock price time-series: only 8 models out of 150 during the validation phase and only 4 out of 30 during the trading phase present an accuracy lower than the reference value of 33% (i.e. average result from a random classifier). Moreover, during specific time periods, in terms of accuracy they reach extremely consistent results, all significantly higher than 40%. Considering the difficulty related to the prediction task and the limited means available to train LSTM networks, such results can be considered particularly impressive.

Interestingly enough, data science metrics also provide evidence about the fact that networks trained during periods of high market volatility and successively tested with observations following the turbulent period (or the opposite) present inferior results compared to models trained and tested in similar market conditions. This result challenges the standard training approach, followed also in this thesis, regarding the creation of a dataset composed just by most recent and strictly temporally close to the trading period observations; the evidence, in fact, creates the conditions to think about the possibility to train specific models with observations coming from periods characterised by market conditions as similar as possible to the one actually playing in the market; the approach would lie on the hypothesis that, when occurring such market conditions, market anomalies are usually driven by well-known biases whose effect on stock market could be predicted by well-prepared ML models.

In terms of profitability (**Section 5.2**), the strategy created thanks to the predictions by LSTM networks shows its edge over selected benchmarks: during a trading period of six years it managed to create a profit of 0.783% relative to the amount of

money invested both for the long and short legs of the dollar-neutral portfolios (that in our case was 1$). Other benchmarks, instead, did not manage to create any significant result concluding the six year period with profits respectively of 0.007% by a random guessing strategy, 0.131% by a short-term reversal strategy and -0.001% by a "long-term ML strategy".

Anyhow, if results in terms of profitability are positive, the downside that emerges by all the investment strategies regards their excessive volatility. In terms of Sharpe ratio indeed, even profitable strategies show poor overall performances during the six-years trading period. Such result highlights the need of a specific focus required during the design phase of a ML trading system on actions to reduce the variability created by models' predictions, as the execution of the "modular classification approach" in this thesis tries to be.

In any case, it is important to mention how a general high volatility does not nullify the whole experiment. In fact, it is noted how during limited time periods, the main strategy is able to positively perform even in terms of profits adjusted for the risk. This result is still extremely valuable when compared with evidences in literature that highlight how ML models in recent time period appear unable to spot valuable patterns in stock market time-series.

Regarding the employment of the "modular approach", it does not return the desired result as a possible approach to reduce the risk of the investment strategy created by ML models. The combination of different portfolios, in fact, brings to an increase in the volatility of the overall strategy; such result is mainly due to the very poor performances obtained by both the portfolios $h = 4$ and $h = 5$. In any case, the approach allows to validate the hypothesis for which models' predictions consistently deteriorate with increasing forecasting horizons, both statistically and financially speaking. This evidence, from a managerial perspective, poses consistent doubts about the possibility to use ML models for the creation of a medium/long term investment strategy.

Finally, in spite of such not so positive results, the "modular approach" proves to

be fruitful, as said, to evaluate models' behaviour at different forecasting horizons, but also to provide a reference to follow in future researches regarding the objective of controlling the unavoidable volatility created by ML models.

In conclusion, at the end of the experimental work (**Section 5.3**) the robustness of the whole trading system in relation to some change in decision variables (weighting scheme **5.3.1**, portfolio dimension **5.3.2**, investment horizon **5.3.3** and input features set composition **5.3.4**) is tested. Particularly, the test conducted to assess the benefits deriving from the introduction of technical indicators and macroeconomic variables in the input features set does not provide significant and clear evidences to justify it.

Due to a limited comparison to the case $h = 1$, such test does not intend to be exhaustive so that its results do not have to be taken as overwhelming evidences. Anyhow, it certainly raises some doubt regarding the additional information content provided by technical and macroeconomic variables to the prediction application. This is especially remarkable in the case of technical indicators. In fact, over simple input variables such as price and returns, it is the category of variables most extensively employed in financial literature. It is not clear if this has to be attributed to some objective evidence or simply to a tendency, consolidated through the years among practitioners, to erroneously consider technical variables as functional to the prediction process.

Finally, the second important issue that this robustness check brings to light, crucial from a managerial point of view, is about the trade-off between the dimension of the input features set and the computation time necessary to train the model: how far should the feature selection process go in the search for new input variables? This knowing that the introduction of new variables inevitably leads the model to become more expensive from a computational point of view without providing guarantees about an increase in performance.

Conclusions

In the light of the evidences emerged from the experimental work of this thesis, three promising paths to be possibly pursued emerge in conclusions (**Chapter 6**).

The first one regards a line of research dedicated to the study of the relationship between data science parameters and financial performance of ML models so as to increase the efficiency of the validation process and thus possibly increasing the effectiveness of ML investment strategies. The validation process for ML models is indeed still performed following the ordinary heuristic methods developed by computer scientists; in any case, such methods find little sense when applied to financial predictions whose final objective regards the maximisation of financial performances and not of data science metrics. From our results indeed, there is not evidence of a strong correlation between data science and financial performances.

Secondly, it emerges the necessity to study effective methodologies to reduce the unavoidable variance deriving from ML models' predictions. As shown by results indeed, if the profitability of LSTM investment strategies cannot be doubted, their variability in profits results to be their most important drawback.

Finally, looking at stock market prediction from a financial perspective and not from a computer science one, it appears questionable the approach that indiscriminately employs observations from any type of market condition provided that they are temporally consecutive. It could be interested to create ML trading systems for specific and well-known market flaws in order to make the model able to profit from them when effectively spotted.

Evidently, all these promising paths should be undertaken taking advantage of the analytic power of LSTM networks (or similar evolution of RNN such as GRU) since, to date, they can be considered among the most advanced ML solutions to approach long series of correlated observations.

Clearly, all these results and conclusions are emerged with at least three important limitations that affect all the empirical work.

At first, all the results in terms of profitability are presented without considering transaction costs. This choice is mainly attributable to the difficulty in finding plausible estimates regarding cost-per-trade for the US equity market for the last 10 years.

As already cited, the validation process is relegated to the number of hidden nodes of LSTM networks due to computational constraints. This limitation probably penalises results obtained during the trading phase but, at the same, it corroborates results obtained, leaving a glimpse of the wide room for improvements over already valuable results. So, overall, the parameter selection process could be viewed as too limited and could be considered as a fragility of the whole experimental work.

Lastly, the overall trading period lasts just for six years. During a so limited time frame, it is not possible to verify whether observed under-performance periods have effectively been sporadic or systematic in our case. Anyhow, this decision is somehow forced by the fact that just before 2009 (i.e. the starting year for observations populating the database employed) financial markets experienced an unprecedented crisis, thus during that period they behaved in a really unconventional way. So, to balance out this turbulence period, it would have been required to further extend observations in the database at least until 2000 causing the experiment to become too computationally onerous in relation to the available means.

Chapter 1

Introduction: Pattern Recognition in Finance

Stock market prediction is one of the most difficult and challenging problem to solve, not only within financial markets applications but among all prediction problems generally. Compared to other type of data, time-series in financial markets are affected by countless factors that can cause price changes and that create a high amount of noise in the prediction task. In fact, being fundamental stock value function of investors' expected dividends and of a discount rate in the form

$$P_0 = \sum_{t=1}^{\infty} \frac{\mathbf{E}(Div_t)}{(1+k)^t} \tag{1.1}$$

where k is the discount rate and Div_t is the expected dividend at time t, it is influenced by all factors that impact on these two variables such as interest rates, company's decisions (e.g. dividend rate, investments to be made), political decisions, economic trends etc.

Anyhow, the high number of explanatory variables is just viewed as a secondary issue when considering what really makes stock market predictions so hard to accomplish: investors' behaviour and their reactions to the just mentioned factors are in fact considered the causes of such a challenging task. Investors' behaviour is indeed the effective transmission chain between fundamental stock value (1.1) (i.e. that reflects the intrinsic value of a company) and the effective stock price on the market. This means that upon its rationality the exact coincidence of market value

and fundamental value of a security depends. Clearly, if investors are rational and so securities' market values equal their fundamental values, there are no possibilities for investors to gain from erroneous pricing. This reasoning is fundamental to understand the two most famous theories about financial markets behaviour that will be presented in the following section so as to outline the theoretical framework in which this thesis lays its foundations.

1.1 Stock Market prediction: Efficient Market Hypothesis vs. Adaptive Market Hypothesis

In academic literature, two important theoretical hypotheses have been built attempting to explain financial markets and the related difficulty when dealing with their future expected performance: Efficient Market Hypothesis (EMH) and Adaptive Market Hypothesis (AMH).

According to EMH (Malkiel and Fama, 1970), markets are efficient since security prices fully reflect all available information resulting in past and current information being immediately incorporated into stock prices (i.e. effective stock price always equal to its fundamental value). To state that, EMH assumes inventors' rational expectations and makes different possible assumptions about information availability. Depending on the severity of the assumption on information availability, three different versions of the EMH can be identified: EMH *strong version* assumes information costs and trading costs to be always zero (Grossman and Stiglitz, 1980) and thus no distinction between private and public information. Said differently, in this version investors have perfect information availability. Following this version, the share price is always an accurate projection of future cash flows of the company. In the *semi-strong* version of EMH, only public information is assumed to be immediately incorporated in stock price since it is the only information available for all market participants. Finally, the weaker and economically more sensible version of the efficiency hypothesis states that prices reflect information to the point where

the marginal benefits of acting on information do not exceed their marginal costs (Jensen, 1978). Thus, for the *weak form* of EMH, only public market information is assumed to be incorporated into share price since the marginal costs to obtain other public and private market information result higher than the edge deriving from the awareness of such information.

The direct consequence of this last more viable form of EMH is that past stock price movements cannot be used to predict future price movements: stock market movements will be determined only by unpredictable future public market information, as past movements have been driven by information already completely discounted by rational investors from securities prices. In statistics terms, security prices based on the EMH weak form follow a *random walk* behaviour (Malkiel, 2007) of the form

$$P_{t+1} = P_t + \epsilon_{t+1} \tag{1.2}$$

meaning that changes in price from one period to the next (ϵ_{t+1}) are random and unpredictable. So that the best guess for the next price can only be the current price. The future direction of a stock, in this sense, is no more predictable than the path of a series of cumulative random numbers. In this formulation, ϵ_{t+1} incorporates all the future and still unknown public information about a company and all random fluctuations of the price around the intrinsic value of the stock caused by investors: in the light of the theory, no investor is all knowing, but collectively they know as much as can be known forming as a group the *market*; these individuals constantly update their beliefs about the direction of the market and their disagreement leads to *"a discrepancy between the actual price and the intrinsic price, with the competing market participant causing the stock to wander randomly around its intrinsic value"* (Fama, 1995).

So, ultimately, for EMH is not possible to "beat the market" since stocks are always traded at their fair value and any past information is already reflected in their price. The key conclusion of the theory is thus about the impossibility to obtain *abnormal returns*: any effort that an average investor dedicates to analyse

and trading securities is wasted and, although high rates of returns may be achieved, they are on average proportional to risk.

AMH (Lo, 2004) tries to reconcile the rational EMH with irrational behavioural economics principles (Simon, 1955): within the debate on EMH, in fact, some studies have highlighted persistent evidence of irrationality of the whole market (French, 1980; Keim, 1983; De Bondt and Thaler, 1985; Barberis and Thaler, 2003) difficult to be explained in terms of EMH. According to Lo, AMH can be seen as an updated version of EMH considering the evidence of irrational behaviour observed in financial markets. In the light of the theory, *"prices reflect as much information as dictated by the combination of environmental conditions and the number and nature of "species" in the economy"* (Lo, 2004), where by species it is intended different groups of market participants each behaving in a common manner. Lo states that the efficiency of the market, thus the predictability of securities' price, depends on the number of groups competing for scarce resources in it: the higher the number of species in the market, the higher its efficiency and vice versa. Moreover, AMH asserts that market efficiency is also affected by biases caused by irrational behaviours of investors that ultimately create arbitrage opportunities. Therefore, in the light of such theory it is possible, under specific circumstances, to exploit the weaknesses (i.e. identifiable patterns) of the market, conversely to EMH, obtaining positive abnormal returns from a portfolio of stocks.

In such a context, backed by evidence and afterwards by AMH, professionals in financial markets have always tried to challenge market efficiency applying different techniques in search of arbitrage opportunities. Particularly, in recent years, machine learning algorithms have emerged as promising tools to do so. Due to the rise of computation power indeed, they are gaining momentum within the finance community as valuable tools to automatise and improve the performances of such research.

The empirical work presented in the thesis fits into this framework: it deals with the application of a powerful and state-of-the-art machine learning technique for

the purpose of detecting valuable patterns to predict stock market movement and to build a profitable equity portfolio. Clearly, evidences of positive performance from machine learning algorithms prove market flaws querying its efficiency.

1.2 Fundamental, Technical Analysis and Machine Learning

Before treating machine learning for financial time series prediction, anyhow, it is important to briefly mention the classical juxtaposition between the two classical approaches linked with securities' value forecast in financial markets that are *fundamental analysis* and *technical analysis*. These approaches are important since they express the operative response produced by investors to both AMH and EMH hypothesis; in fact, fundamental analysis can be considered the natural extension of EMH hypotheses, while technical analysis can be better understood if linked to AMH hypotheses. Only in respect to this dichotomy, machine learning can be correctly placed inside the set of possible techniques to face financial market predictions.

Fundamental analysis relies on the examination of business' financial statements, jointly with the overall state of the economy, to spot overvalued and undervalued stocks in the market so that to increase the chances of investors of making higher-than-market average profits. It relies on strict objective valuation methodologies whose results strongly depend on subjective hypotheses: investors have to make assumptions regarding expectations about all factors that will influence the profit of a company in the future to obtain from (1.1) their guess about the *fundamental value* of a company. Fundamental analysis is thus linked to the weak form of EMH hypothesis: it implies that the only way in which investors can obtain positive abnormal returns is thanks to *information asymmetry*, that is a better level of information possessed by one agent compared to the one possessed by other participants in the market so as to produce more accurate assumptions and thus having more precise expectations. As stated by EMH, fundamental analyst do not believe about

the possibility to predict future market movements studying past movements.

Technical analysis, instead, relies on different premises: it assumes that all needed information to predict future stock prices is incorporated into current and past stock prices so that, in order to predict the future, it focuses on the analysis of historical data to find recurrent patterns. In view of AMH theoretical framework, this second approach works due to agents' irrationality that creates recurrent patterns in the market that, if spotted, can provide investors valuable insights to "beat the market". All the knowledge embedded in technical analysis, indeed, *"is to identify a trend reversal at a relatively early stage and ride on that trend until the weight of the evidence shows or proves that the trend has reversed"* (Pring, 2014). From a practical point of view, a technical approach consists in the calculation and interpretation of "technical indicators" that are measures calculated from market data (i.e. primarily price and volume) that do not rely on any specific theoretical framework. They are intended as supporting tools to be used during investors' decision process, helping them deciding when to invest or divest in a specific financial asset. The downside of the approach lies on the interpretation of such indicators that is essentially heuristic: technical investment strategies depend on the experience and acumen of each single investor in understanding what to look at.

Now, machine learning for financial market predictions relies on similar market inefficiency assumptions made by technical analysis: agents irrationality creates recurrent patterns in financial time-series that is possible to spot thanks to mathematical models.

Mathematical models are not new to financial time-series analysis: until some years ago, the largest part of such quantitative approaches in finance were about econometric models as Moving Averages, ARMA, ARIMA, ARCH and GARCH. All these regression models, with some peculiar differences, hypothesise linear dependencies between the price of a securities (the dependent variable) and its lagged prices, lagged errors and some other possible externality. However, since financial market is a complex, evolutionary and non-linear dynamical system which interacts with

political events, general economic conditions and agents' expectations, econometric approaches has not proved to perform very well in predicting securities' prices accurately. Moreover, input features used for predictions in these models have proved to be multicollinear and relationships between factors and returns to be variable and/or contextual. In all these circumstances econometric models have demonstrated not to be flexible enough to adapt. Finally, econometric models require a high effort in the validation of all the assumptions necessary to consider reliable the results. Nonetheless, in less noisy type of prediction applications than price prediction, such as risk measure forecasting, they have shown solid results. Due to all these reasons, in recent years, in addition to traditional econometric approaches, complex models coming from Machine Learning (ML) field have started to be applied for many financial applications, attempting also to predict future movements of securities' prices (Yoo et al., 2005).

Machine learning refers to methods and algorithms that allow machines to discover patterns without explicit programming instructions. In many fields outside finance, machine learning algorithms have proven to be more effective than traditional statistical techniques: one of the most impressive and famous result of ML is related to its application to the game Go, considered for long time to be an Everest for artificial intelligence research, where *AlphaGo*, a Go-playing computer, managed to win against one of the best human players of all times (Silver et al., 2016; Borowiec, 2016). The advantages of such algorithms when applied to financial time series are manifold, among the most important ones it can be certainly identified their assumption independence: ML models indeed, have been created to deal with noisy and non homogeneous dataset in which the validation of all the assumptions required by classical statistical model would not be possible. Moreover, ML models are able to find patterns and irregularities as well as detecting multi-dimensional non-linear connections in data and, finally, they have the ability to extract rules from large sets of data, as the ones containing financial time-series. As stated by Heaton et al. (2017, p. 1) *"applying deep learning methods to these problems [i.e.*

problems in financial prediction and classification] can produce more useful results than standard methods in finance. In particular, deep learning can detect and exploit interactions in the data that are, at least currently, invisible to any existing financial economic theory".

ML techniques have been available for a long time, Frank Rosenblatt invented a neural network that could classify images in 1957 indeed (Rosenblatt, 1958). In any case, the reasons why these techniques have been started to be successfully applied in fields such as genetic, and afterwards also to financial markets, have to be ascribed to the increased computing power, the increased data availability and the creation of new powerful algorithms from disciplines such as computer science and statistics.

In conclusion, to summarise, the objective of the entire empirical research presented in this thesis is to answer the question: *is it possible to achieve abnormal returns in equity market through the use of a recent and powerful Machine learning technique?*. Not only financial industry, but also academia, have struggled to find a way to answer positively to this question. Clearly, a positive answer to the question does not only bring new knowledge about the functioning of financial markets, but it can become source of important earnings for those who decide to keep this knowledge private and use it to invest in the market.

It is important to underline how the answer to this question is not so connected, as it might seem, to advancements in computer science research. Rather, it requires a specific focus. In fact, ML financial applications differ from applications from many other ML fields due to inconsistencies between data science performances and financial performances by ML models. As it will be clear further in the thesis indeed, there is not always a strict relationship between classic ML performance metrics, typically employed as objective parameters to be maximised when training a ML model, and financial performance metrics, real objective parameters of an investment strategy.

Chapter 2

Literature Review

As anticipated, the advancement in computation power and the increasing data availability have naturally introduced Machine Learning in finance/portfolio management in recent years. However, there is still an important gap between academic finance and the financial industry. In the business world big data is reality, while in the financial academic literature *"there is very little published management scholarship that tackles the challenge of using such tools, or better yet, that explores the promise and opportunities for new theories and practices that big data might bring about"* (George et al., 2014). ML methods *"are rarely considered by financial economists who prefer econometric, often linear methods"* (Hsu et al., 2016). As highlighted by Huck (2019, p. 2), to prove this, *"with the leading financial journal "The Journal of Finance", a search for "machine learning" until 2017 produces no reference"*.

Undoubtedly, an important reason regarding this gap has to be attributed to the economic advantage that asymmetric information gives to the investor able to create an effective prediction method and that keeps it private towards the market. Once such information was public, in fact, it would be arbitraged away from the market, preventing the opportunity to earn from it: as found by Huck (2019) in his work *"empirical results confirm a severe drop in performance of trading systems based on machine learning in the most recent years: when these tools became more accessible"* and thus an increasing number of agents started to discount from

prices also this new kind of information. In a way, it is possible to state that the increasing adoption of ML trading models are achieving to make financial markets more efficient discounting information that previously was not possible to posses. The second main reason regarding the gap from academic literature and business applications is related to this increased efficiency: it is requiring more and more powerful method to find new arbitrage opportunities to gain from. Clearly, more powerful methods require high computation power that comes at a cost possible to be suffered exclusively by professional investors able to earn proper returns from their investments and research activities, as opposed to academics.

In this context, almost exclusively computer science researchers have started to deal with the problem of prediction of financial time-series through ML models, especially in the equity market. Many algorithms and methods have been proposed trying to achieve significant results in the prediction of time series and in most cases to implement effective trading strategies based on the outcome of such predictions. From a financial research perspective in any case, the problem of such computer science literature is that in very few applications it looks for valuable business explanations regarding the reasons for which a given set of inputs is used or for some valuable information from models' outputs confuting or verifying common investors knowledge. For instance, a virtuous case regarding the confirmation of common knowledge can be found in Fischer and Krauss (2018): at the end of their empirical study, authors state how their LSTM neural network selects stocks to trade as it was aware of some well-known capital market anomalies (e.g. their LSTM model select stocks with below-mean momentum, short-term reversal characteristics, high volatility and beta), without being explicitly trained to spot them.

As in many other prediction application, financial time-series predictions in literature are tackled as both regression and classification problem. In the regression approach, the objective of the ML model is to make a punctual estimation of the price of the security for a given horizon. This type of approach becomes really difficult when dealing with stock prices and, in the vast majority of cases, ML mod-

els are used to solve classification problems indeed. This is the reason why in the following section, where a non-systematic literature review of valuable approaches presented in literature will be provided, most of applications deal with classification settings. Anyhow, this does not imply that is not possible to find regression attempts in literature. For instance, Chen and Hao (2017) try to forecast the actual closing price of two important Chinese stock market indexes as the Shanghai and Shenzhen ones. They manage to achieve good results, even if working on forecast of composite indexes and not on single stocks. The same attempt for a punctual prediction of a composite index, this time the S&P500, can be found in Cai et al. (2012). Differently, in a classification approach the ML model is trained to predict the trend of the price of the security. In such approach, the ML model is instructed to guess whether the value of a security in a given point of time in the future will have increased its value or decrease it. Most of classifications regards two classes, one for the upward trend of the security's price and the other one for the downward trend.

Regarding the reasons why classification approaches are more frequent than regression ones, at first it can be noted how for the purpose of implementing a successful trading strategy, that in the vast majority of the cases is the actual reason why prediction of financial time series is performed, the output of classification models gives sufficient information for the decision process. But more importantly, the loss in prediction accuracy while performing a regression forecast rather than a classification one can, in many cases, be higher than the marginal gain the former approach would bring thanks to additional information. Finally, selecting stocks through a regression approach would lead to choose stocks with a high level of volatility, namely the stocks with the highest predicted price in absolute terms, while selecting stocks through classification approaches favors stocks more likely to belong to certain trend, resulting most of the times in less volatile stocks selected and consequently in a lower risk of the trading strategy.

2.1 Practical Applications in Literature

In this section, as already explained, a non-systematic literature review is presented. The criteria by which papers have been selected to enter in it concerns exclusively both the willingness to briefly present different research perspective about the interconnection between machine learning and investment strategies and the innovative approach of some of them towards the creation of a profitable ML trading system. Table A.1 in Appendix presents a synthetic view of all the papers that will be presented in this chapter.

Algorithms and application fields regarding the study of financial time series through ML methods are numerous. As previously anticipated, the large majority of articles dealing with stock market predictions are published in journals belonging to the machine learning and operational research community. In some of them, ML algorithm are presented as supporting tools for more "traditional" approaches to investments rather than proper tools able to extract information from a pool of data by their own. For instance, **Pai and Lin (2005)** examine the results produced by the combination of a classical econometric linear model with a non-linear one. The objective of the ML non-linear model, in this case a support vector machine (SVM) with gaussian kernel, is to shape the variability left unexplained by an ARIMA (0,1,0). Specifically, the one-day ahead closing price prediction Z_t of the hybrid model created is given by $Z_t = Y_t + N_t$, where Y_t refers to the ARIMA forecast and N_t to the SVM ones. While ARIMA model is directly trained to predict the one-day ahead closing price of ten stocks with 50 predictor for each stocks, SVM, through a non linear kernel function, is trained to represent the residuals of the ARIMA model using as input lagged residuals of the linear model itself. A comparison between the predictive ability of the two models used singularly and the hybrid model in terms of MAE, MSE, MAPE and RMSE, shows how the hybrid model helps to significantly reduce the overall forecasting errors.

Another interesting example of ML algorithm supporting a "traditional" ap-

proach is provided by **Chandrinos et al. (2018)** that propose ML models as a risk management tool, or better *"as an expert advisor improving the performance of existing trading strategies"* (Chandrinos et al., 2018). In their approach indeed, a decision tree and a deep feed-forward neural network are implemented to classify the produced signals of a technical trading strategy (i.e. channel breakout strategy) proposed in Chandrinos and Lagaros (2018) into "profitable" and "non profitable" ones. The technique is applied to 5 different currency pairs and it aims at reducing the standard deviation of the original trading strategy and at improving the correlation between total returns and the portfolio's standard deviation. The return of the ML strategy is thus determined as the sum of returns of all the original signals transformed into effective trades, that are all the original signals that the algorithm classifies as "profitable". The signals categorised as "non-profitable" are ignored instead. The parameters for the decision tree model and the deep NN are selected on the basis of their performance on a validation set in terms of F1-score. In this specific application the focus is on the ability of the model in recognising profitable signals and a high F1-score would signal the ability of the model in predicting almost all of winning trades indeed. Models are trained separately for each of the 5 different currency pairs (i.e. GBP/USD, USD/JPY, EUR/USD, GBP/JPY, EUR/JPY) thanks to training set of three years and then validated in a validation set containing one year of observations. 14 different technical indicators and 5 features related to the last five entry prices where a signal was produced by the original technical trading strategy are used as input features. The total dataset consists in observations from 2006 to 2016. Financial results are evaluated on a rolling window approach starting from 2010. Annualized returns, standard deviation and Sharpe ratio are finally calculated for each currency pair for two different type of portfolios, an equally weighted and a weighted ones (with weights depending on previous year performance). Both decision tree and NN achieve positive results: decision tree prevents investors form significant drawdowns of the original trading strategy reducing the variance and succeeds to improve total returns for all currency pairs.

NN reduces the standard deviation of 4/5 pairs and increases the total return of 3 of them. The most significant achievement by NN is the elimination of almost all of negative returns that were produced by the original channel strategy. Authors conclude highlighting how *"a noteworthy evidence of these techniques as risk management tools is their ability not only to improve negative returns but to turn most of the losing years into profitable ones and to increase even more highly profitable years"* (Chandrinos et al., 2018, p. 14).

However, differently from a view where ML methods are seen as supporting tools for more conventional approaches to financial time series, a consistent part of the literature focus on the development of a trading system exclusively based on ML algorithms. One of the most noteworthy contribution that is possible to find in literature regarding trading system exclusively based on ML model's outputs is **Huck (2019)**: it manages to combine the financial, operational and computer science perspective in one application. The combination of the three perspective is in fact crucial to take in consideration all of the necessary aspects to build an effective financial trading system based on machine learning. The main objective of the article is thus to tackle concurrently the dimensions of large datasets, machine learning and a discussion of the disagreement between EMH and the evidence reported in the ML literature: large datasets refers to the impressive amount of financial data available that, given recent development in machine learning, can now be implemented to spot recurrent patterns in financial time series and to pose even more evidences against a view of financial markets as totally efficient markets. The natural application field for big data and machine learning in trading, according to the author, is *statistical arbitrage*, a class of short-term financial trading strategies that employ mean reversion models involving broadly diversified portfolios of securities. Huck justifies his research by a lack of empirical works dealing with large datasets and state-of-the-art ML methods. The algorithms used to implement statistical arbitrage strategies in the study are random forests (RAF), deep belief network [1] and elastic net regres-

[1]Deep belief networks (Hinton and Salakhutdinov, 2006) are a particular case of neural networks

sion [2]. Such models are used to forecast the probability of the return of a specific stock to be larger than the median return computed over all stocks in the dataset for one-day and five-days ahead. The experiment is computed dealing with stocks included in S&P900 index from 1990 to 2015. The complete set of all input features includes lagged returns, dummy variables for stock identification, day of the week, month, industry (following ICB classification), implied volatility classes based on VIX, gold and oil price, rates of 10 years treasury and finally Fama and French factors (Fama and French, 1993) plus momentum and short-term reversal (Jegadeesh, 1990; Carhart, 1997), for a total of 592 predictors. Results are calculated among different dollar-neutral portfolios created including each day the 10 top and flop stocks based on output of the different models, trained each time with different combination of input, at two different horizon (1 and 5 days) and with both 100 and 300 stocks, for a total of 40 different combinations tested. For each of the 40 different combinations, within the testing period (1993-2015), each model is re-trained two times per year resulting in 45 re-training per each model. Synthesised results from the experiment are as follows:

- All model generate a positive excess return before transaction costs between 1993 and 2015 with an average annualised return of 35% compared to a market performance of 10%. Including transaction costs, the average becomes 11% with 6 out of 40 combinations of models-input features presenting negative returns.

- RAF results to be the most effective technique, with all significant returns significantly positive.

- On average, the models forecasting one day ahead perform better than the

whose main element are autoencoders. An autoencoder, in turn, is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation for a set of data, typically for dimensionality reduction, by training the network to ignore noisy signals.

[2]Elastic net regression (Zou and Hastie, 2005) is particular case of linear regression specifically designed to manage a large number of features and to perform dimensionality reduction.

ones forecasting at five days.

- A substantial deterioration in performance is seen when all predictors are included in the model compared to the cases with only lagged returns.
- After transaction costs, only 50% of the models still have a positive alpha when evaluating the exposure of returns to common sources of systematic risk.
- In accordance with the evolutionary perspective in AMH, dividing the trading period into 4 different sub-periods, the most recent one (2010-2015) reveals how all techniques are not able to generate positive and economically significant trading signals. *"In some way, this confirms the efficiency of the market given that profit opportunities are arbitraged away with the increasing popularity of these models"*(Huck, 2019, p. 10). In contrast, in the first period (1993-2000), during which algorithms and computation power used were not available, the returns of all model result to be really impressive.
- When including transaction costs, the inertia of each algorithm (i.e. the probability of a given stock to be selected two consecutive days), influences a lot returns. It is computed in 20%, 60% and 40% respectively for RAF, DBN and elastic net.

Within the scenario clearly described by Huck (i.e. ML based approaches for the identification of statistical arbitrage investment strategies) a plethora of experimental work can be found, some of them more focused on the identification of the most appropriate ML model, among the ones presented in computer science literature, for signal generation. For instance, **Krauss et al. (2017)** test the effectiveness of deep NN, gradient boosted tree, RAF and ensemble methods in providing signals to create a profitable trading strategy. Regarding the three ensembles, the first one consists in a equal-weighted ensemble that takes the predictions of the three base models and average them; the second ensemble is based on performance, meaning that the three predictions are weighted as a function of Gini index calculated with

the AUC of each base learner as $Gini = 2 \times (AUC - 0.5)$; finally, the last ensemble weights the different predictions with the inverse of the Gini index creating a scheme less sensitive to outliers. In the database are presented all S&P500 constituents from December 1989 to September 2015. As input features for each different models only lagged returns r_{t-k} are used, where $k \in \{1, ..., 20\} \cup \{40, 60, ..., 240\}$. The entire dataset is divided into 23 study periods where each one is formed by 1000 days of which 750 used for training and 250 for trading (with non-overlapping trading periods). Within the training period, in each day the probability of each stock to outperform the cross-sectional median in $t+1$ is computed; each model thus classifies stocks based on such probability. Regarding the trading strategy, different portfolios are built based on this ranked probabilities. In each of them the system goes long (short) on top (flop) n stocks, with $n = \{10, 50, 100, 150, 200\}$. Main results are as follow:

- Ensembles outperform all base models since they respect *diversity* and *accuracy* conditions. All three ensemble are diverse since the errors of their base learners (i.e. three ML models whose predictions are averaged) show low correlation and they are accurate since they achieve a directional accuracy higher than 50%. Among the three ensembles, however, there are no significant differences.

- RAF achieves better results compared to GBT, probably because it does not suffer from overfitting and because it is robust to noisy feature spaces, as the one of the application.

- Deep feed-forward neural network shows the worst results, in any case authors attribute the result to the difficulty in the fine tuning process of all its hyperparameters. Moreover, it is found to lose efficiency when the number of its layer are diminished.

- Increasing n leads to decreasing returns but at the same time at decreasing

standard deviation. Regarding absolute returns the result is attributable to the fact that within portfolios are included stocks whose prediction have a higher uncertainty; while, regarding their standard deviation, the result is attributable to an increasing diversification of the portfolios.

- In terms of risk, RAF shows the lowest VAR while Deep NN the highest. Moreover, regarding ensembles, positive relation between VIX and their returns are found suggesting ensembles to work better during period of high market turmoils.

- Accounting for common sources of systematic risk, for all models it is found a positive alpha and it is also found how returns partly load on common sources of systematic risk suggesting an investment behavior that incorporates several capital market anomalies from the ML model.

- A sub-division of the entire time period under scrutiny highlights how from December 1992 to March 2001 all models consistently outperform the market, in any case during this period ML models were still not accessible by investors in the market. The period from April 2001 to August 2008, instead, corresponds to a period of moderation in terms of results and, in fact, during it ML models started to be used in financial markets. During the period corresponding to the financial crisis (from September 2008 to December 2009) the long-short term strategy temporarily returns to perform well. Finally, the period from January 2010 to October 2015 corresponds to a period of deterioration with negative annualised returns after transaction costs.

- Focusing on the best results achieved by models, it emerges how they seem able to capture relative mispricing at times of high market turmoil such as the .com bubble in 2000, the global financial crisis in 2008 and the European debt crisis in 2011.

- Most important returns for every model result to be the ones corresponding to the last 5 or 6 days.

- Algorithms are found to prefer high beta (e.g. tech industry) stocks instead of lower beta ones (e.g. financial industry or utilities).

Differently, another type of approaches, more than concentrating on selecting the best ML algorithm, focus on features selection. They aim at finding the most appropriate variables and techniques so as to facilitate the information extraction by ML models. **Picasso et al. (2019)** work on the twenty most capitalised stocks listed in NASDAQ100 to combine technical and sentimental analysis to classify price trends: the complete input vector used for all their tested algorithms is formed by *sentimental variables* extracted from 2 different dictionaries presented in Loughran and McDonald (2011) and in Cambria et al. (2015), 10 different technical indicators and price values. RAF, SVM with gaussian kernel and NN with 4 different layers are tested in a one week-ahead classification so as to select the most appropriate model to be used for the implementation of an effective trading strategy. During training and validation stages, authors adopt the following expedients to deal with typical issues of market time series forecasting:

- To deal with the presence of unbalanced labels, that depends on the tendency of the market to be negative (bearish) or positive (bullish) in a specific time period, *synthetic minority over-sampling technique* (SMOTE) is applied to oversample the minority class creating new synthetic observations.

- Since the main target is to select a model which is able to make correct predictions both on positive and negative samples (so as to create a long-short trading strategy) and not only to achieve a high directional accuracy, geometric score ($Gscore = \sqrt{TPR \times TNR}$) is used as a measure to validate different algorithms. In this way authors aim to remove the bias coming from an unbalanced distribution of the training set.

- Before computing data science metrics, the observations in the dataset are divided into five clusters filled with samples representing increasing percentage changes in market price. The aim of this process is to to understand the behavior of the model with different predicted trends; achieving high performance on the cluster representing large changes in market price, is more valuable than obtaining the same result on less relevant samples indeed.

At the end of the training and validation stages, authors select feed-forward neural networks as reference model to create the trading strategy: the buy signal is generated if $prediction\ NN > 0.5 + threshold$ and the sell one if $prediction\ NN > 0.5 - threshold$. The threshold value is calculated to consider the 75% of the predictions which are considered the most "reliable" according to the softmax output. Data science metric shows how the dataset using sentimental variables from affective space is more effective than the one using variables from Loughran and Mcdonald's dictionary. Moreover, authors, to value the informative content of price and sentimental data, compare results from NN trained with only sentiment variables and technical indicator and finally with both of them: they find that "price&news" NN clearly outperforms "price" one and that "price&news" NN shows only little improvements when compared to "news" NN. However, "Price&news" NN shows higher confidence in prediction than "news" NN. Regarding financial performances, the NN-Lough-McD model shows the highest gains with "price&news" while the NN-affective_space shows the best results with only "news". Authors attribute this result to the inability of the NN to exploit properly the feature combination power. In fact, due to their characteristics, the dataset built with Lough-McD dictionary contains 151 [3] inputs while the one built with affective space 611. In conclusion, the results of this empirical work show how adding sentiment representation to standard price features can lead to important improvements in performance. Authors finally hypothesise how a process of fusion between price and news would allow to exploit even better the representative power of the combined features.

[3]Included technical indicators and price information.

Another example is provided by **Chen and Hao (2017)** that try to improve the robustness and accuracy of a SVM and a K-nearest neighbor in forecasting the closing price of the Shanghai and Shenzhen stock exchange indexes by weighting the contribution of different input features. Their approach considers at first the SVM to classify the trend of the two indexes, after that, depending on the trend predicted, KNN is supposed to perform a punctual estimation of both the index. The foundation of the empirical work relies on the assumption that different input variables do not have all the same impact on the final output; if this is the case, assigning different weights to input features, can improve the results of the algorithm by nudging it to focus more on the most informative inputs within the dataset. As measure to asses the importance of each input feature and to find the weights related to each variable, the *information gain* is used: suppose the training set $T_{train} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ where $y_i \in \{C_{-1}, C_{+1}\} = \{-1, +1\}$. $C_{\{i,D\}}$ denotes the subset of the dataset D belonging to class C_i, where $i = -1, +1$, $\mid D \mid$ denotes the size of the dataset D and $\mid C_{\{i,D\}} \mid$ the size of the sample set $C_{\{i,D\}}$. The probability of a sample belonging to a class C_i can be calculated approximately by $\mid C_{\{i,D\}} \mid / \mid D \mid$. The expected information to classify the dataset can be expressed by entropy as:

$$info(D) = -\sum_{i \in \{-1,+1\}} \frac{\mid C_{\{i,D\}} \mid}{\mid D \mid} \, log(\frac{\mid C_{\{i,D\}} \mid}{\mid D \mid}) \tag{2.1}$$

So, supposing T_{train} to be split on one feature A_{split} into different subset $D_1, D_2, ...D_v$, if the summed entropies of each splits (2.2) are lower than the one in (2.1), then the partition is good and the corresponding feature is considered to have a greater contribution to classification

$$info_A(D) = \sum_{j=1}^{v} \frac{\mid D_j \mid}{\mid D \mid} info(D_j) = -\sum_{j=1}^{v} \frac{\mid D_j \mid}{\mid D \mid} \sum_{i \in \{-1,+1\}} \frac{\mid C_{\{i,D_j\}} \mid}{\mid D_j \mid} \, log(\frac{\mid C_{\{i,D_j\}} \mid}{\mid D_j \mid}) \tag{2.2}$$

$$infoGain(A) = \sqrt{info(D) - info_A(D)} \tag{2.3}$$

The different inputs used in the experiment comprehend index closing prices and 9 different technical indicators. The prediction about the two stock indexes is made

at 1, 5, 10, 15, 20 and 30 days ahead. Results show how feature weighted models perform better than standard models reducing the noise of those variable that are not important for predictions. Nevertheless, as the forecast goes ahead in time, the advantage of the weighted models towards standard ones diminishes.

Finally, **Cai et al. (2012)** start from the assumptions that in the majority of ML applications regarding stock price forecast, models use a small number of features as input possibly causing the model not to have enough information to make accurate prediction due to the complexity of the stock market. At the same time, in any case, a large number of features increases the training time and decrease generalization performances due to a higher probability of overfitting. Therefore, the paper takes advantage of *restricted Boltzman machines* [4] to extract low dimensional features from high dimensional raw input data to be used as input features for SVM in a regression setting. More precisely, 16 technical indicators for the S&P500 index and for other 19 highly correlated stocks are used to train a DBN with the objective of extracting features. Then, the extracted features and the price data of the target stock are used as inputs of a SVM that forecasts the closing price of target stock at next day. Price data are the only input features not transformed by DBN since they are considered the most important ones and, if processed, forecasting accuracy may worsen. Results of a SVM trained only with technical indicators and price data for the S&P500 compared with the ones of a SVM trained with the same data plus the close price of the other 19 correlated stocks, show how the former model outperforms the latter in terms of normalised mean squared error and directional accuracy. This indicates how generalization performance of SVM is deteriorated when dimension of input is increasing, even though more input variables comes with more information content. In any case, the best results are obtained by SVM trained with extracted features from the S&P500 and 19 correlated stocks from a deep belief network with dimension 324-150-80-20. So, results presented by authors confirm

[4]Restricted Boltzman machines (Salakhutdinov et al., 2007) are a two-layered artificial neural network with generative capabilities. They have the ability to learn a probability distribution over their set of input.

how dimensionality reduction, in this case obtained through restricted Boltzman machines, can improve generalization performances in case of a large number of predictors within the dataset.

As emerged so far, looking at the publication time of the different papers and at the ML algorithms employed by them, it can be noted a precise trend in literature over the last ten years: researchers have passed from taking advantage of models such as classification trees or support vector machines to extensively employing neural networks, extending through time, thanks to an increased computational power availability, their depth and width. The reason of this trend has to be found in the incredible suitability of neural networks to cope with non-linear dependencies lying among observations within the dataset. In very recent years, moreover, this trend has further evolved focusing on the implementation of recurrent neural networks (RNN) for the creation of profitable trading systems. This further evolution has to be attributed to the RNN advantage, over standard neural networks and the vast majority of ML models, of being specifically designed to extract information from correlated observations such as the ones in time-series. Specifically, among the different variations of RNN, long short-term memory neural networks (LSTM), being successfully employed in many fields outside finance (e.g. image recognition), have emerged as state-of-the-art algorithms and they have been started to be employed also in academic financial research. The reason of their success is mainly related to the fact that they have been specifically developed to solve the problem affecting standard RNN related to vanishing effects of long-term dependencies. **Sirignano and Cont (2019)** in their empirical work help in understanding the main reason related to such such promising results from LSTM applied to financial predictions: among other conclusions, indeed, they find evidence of long memory in financial time-series. In their paper, authors start with the evidence that thanks to the huge number of data present for market transactions, it is possible to explore the nature of price formation mechanism (i.e. description of how market prices react to fluctuations in supply and demand). According to authors, the reason for which

large scale ML methods have not been extensively deployed in finance yet is that statistical modelling of financial time series has remained asset specific and data used for estimation are often limited to recent time window. This characteristic is in contrast with impressive results obtained in other fields with data sets of order of magnitude larger. If the relation between variables within financial datasets were universal and stationary then one could potentially pool data across different assets and time periods to use a much richer data set to estimate/train models. From such evidences, authors aim at demonstrating effectively the universal and stationary relationship between variables (i.e. parameters varies neither with asset nor with time) in financial time series; to do so, they pool data across different assets and time periods so as to use a much richer data set to train ML models. An example of universality for financial time series is directly provided by the authors: *"For instance, data on a flash crash episode in one asset market could provide insights into how the price of another asset would react to severe imbalances in order flow, whether or not such an episode has occurred in its history"* (Sirignano and Cont, 2019, p. 3). This same idea in fact, known as *transfer learning*, has been applied with great results in image and text recognition. To demonstrate it, in the experiment authors employ a 3-layer network with LSTM units followed by a fully connected layer of ReLu and an output layer with softmax activation function. All the experiment is performed in a regression setting where the objective of the prediction is the next price move within the dataset that can considerably vary from a fraction of a second to seconds. They use a dataset consisting in high-frequency record of all transactions and order cancellations for approximately 100 stocks traded on NASDAQ between 1 Jan 2014 and 31 March 2017. Input features consists in a vector of state variables (i.e. transaction and quotes) encoding the history of the order book over many observation lags. The four main characteristics of financial time-series emerged from the experimental work are the following ones:

- **Non-linearity-** A simple linear vector autoregressive model is compared with

results of the LSTM model. Both models are trained on data coming only from a single stock. A substantial increase in accuracy between 5% and 10% is found when incorporating non linear effects into the LSTM over the VAR model. The same comparison is performed with both models trained on all stocks and, even in this case, LSTM shows a 10% higher accuracy compared to the linear model.

- **Universality-** A deep NN model trained with one specific stock is compared with one identical model trained on all stocks. Results show how the universal model consistently outperforms the stock specific model indicating the presence of common features relevant to forecasting across all stocks. In another test, the universal model also proves to be able to generalise to stocks that are not part of the training dataset. Moreover, the stock specific model proves to suffer from overfitting due to smaller training dataset, while the universal model proves to be able to generalise being less exposed to overfitting.

- **Stationarity-** The model performance in term of price forecasting accuracy is remarkably stable across time showing evidence of a stationary relationship between order flow and price changes.

- **Evidence of long memory-** The out-performance of the LSTM network compared with a standard feed-forward neural network, both trained on all stocks, shows how the history of the input features can provide significant additional information. Even more complex and deep NN are not found to be able to reduce the performance gap between a static model and recurrent model. Finally, the forecast accuracy has found to improve when the LSTM is provided with a longer look-back period as input.

As it will be accurately explained in 3.2.3, within the family of recurrent neural networks, it is possible to find many different variations. All of them have been proposed to deal properly with datasets containing correlated observations. The-

oretically, among all variations, more recent ones should be able to better extract long-term dependencies that standard RNN, affected from vanishing/explodient gradient problems, are not properly able to extract. The work of **Liu and Liu (2018)** is significant in this sense since it compares a standard RNN with more recent LSTM and gated recurrent unit networks (GRU). The study takes advantage of such models to predict the price movements of the HS300 index from Jan. 2005 to Dec. 2017 5-days ahead. Authors do not select the one-day ahead prediction since it is often disturbed by a lot of noise. Moreover, in contrast with the majority of classification approaches that divide the direction of stock movement using a binary classification forecasting model (up or down), this approach creates a third class representing all returns data that lie inside a certain threshold range around zero. This choice is made at first to decrease effective trades that at the end, net of transaction costs, reveal not to be profitable, and secondly because usually models struggle to meaningful distinguish ups and downs when returns are close to zero. Finally, the study focuses on a "movement trend-based data preparation" method, meaning that it focuses on the discretization of technical indicators on the basis of the dynamic relationship between the original trend factor and the stock movement trend (e.g. when $stock\ price > ema_{10}$, $ema_{10} \leftarrow 1$ otherwise $ema_{10} \leftarrow -1$). Results show how the best performance, among all combinations of models and dataset with discretised and non-discretised data, is achieved by GRU trained with discretised data with an accuracy of 68% compared to LSTM with accuracy of 65% and RNN 64%. Moreover, among all models, the deep and narrow structure outperforms the shallow and wide one keeping the total number of neurons in hidden layers constant.

Another big step in machine learning research that has followed the flaws related to standard RNN is the concept of *attention mechanism* (Bahdanau et al., 2015): the theory of attention comes from evidences that many animals focus on only a specific parts of their field of view to take adequate actions; in neural computation this is translated by the fact that it might be sufficient to focus on most relevant piece of information to make further neural processes, rather than using all of it.

Practically, the idea is to let every step of a RNN pick only a part of information to look at, from a larger collection of information. Take as example a RNN aiming at creating a caption to describe an image, for every word it outputs it might pick a part of the image to look at. **Chen and Ge (2019)** are probably the first to apply this mechanism to financial time series through an attention based LSTM with the objective to predict the direction of daily close price movement for stocks in the Hong Kong stock exchange. As input features, the application uses 8 technical indicators plus standard price features. Results from the paper on the accuracy of a two layer LSTM followed by a dense layer and a batch-normalization layer and the same network preceded by an attention layer show how the attention mechanism improves the overall stock movement prediction accuracy. The AttLSTM shows prediction accuracy improvement in 56 stocks out of 72 stocks before parameter tuning. Even after tuning the input layer size, the batch size and the learning rate on a validation set, where the configuration with the highest accuracy is selected, the AttLSTM outperforms LSTM in 22 out of 28 stocks. All accuracy measures for each different stock in the dataset range between 50% and 65%. Authors also create a long-only trading strategy, where for each day they proportionally invest all wealth on those stocks predicted to have positive movements by the two models: during a period of 700 trading days, the attLSTM shows a positive total return of almost 180% compared with 100% of LSTM and 15% of the Hang Seng Index.

To conclude the review, the remarkable application of **Fischer and Krauss (2018)** regarding LSTM networks is presented. The work of the authors is really important for a thesis aiming at examining LSTM networks and financial time-series since, as they state, it provides *"an in-depth guide on data pre-processing, as well as development, training, and deployment of LSTM networks for financial time series prediction tasks"* (Fischer and Krauss, 2018, p. 2). Specifically, authors expand the recent work of Krauss et al. (2017) working on the same dataset; one unexpected finding of the previous experiment was that deep neural networks with returns of 0.33% per day prior to transaction costs under-performed gradient-boosted trees

with 0.37% and random forests with 0.43%. The latter fact is surprising, given the dramatically improvements that deep learning has brought in speech recognition, visual object recognition, object detection and many other domains. One would expect similar improvements in the domain of time series predictions. However, as already cited, authors point out the difficulty in fine-tuning processes for deep learning architectures. This is the reason why Fischer and Krauss focus on LSTM networks, one of the most advanced deep learning architectures for sequence learning tasks in this case. Moreover, differently from any other reserach presented in this section, authors aim at drawing general rules to look into the black box of artificial neural networks thereby trying to unveil their sources of profitability. Authors decides to evaluate the performance of LSTM with a single hidden layer, 240 timesteps, trained only with return sequences, to compare it with random forests since they deliver virtually no tuning and good results, with deep feed forward neural networks (with 31-31-10-5-2 neurons in each different layer) to show the relative advantage of LSTM units and with logistic regression selected as a baseline. All benchmarks, being static models, differently from LSTM, are trained with daily with cumulative returns at m days with $m \in \{1, ..., 20\} \cup \{40, 60, ..., 240\}$. Each model is trained to forecast the probability of the price of a single stock in the dataset to be higher or lower than the cross-sectional median of returns for all stocks at $t + 1$. On the basis of the probability output, different portfolios are created going long (short) on top (flop) k stocks, where $k \in \{10, 50, 100, 150, 200\}$. Main results in terms of performance, within the period Jan.1993-Oct.2015 for stocks included in the S&P500 index, can be summarised as follows:

- In terms of accuracy, it is found a clear advantage for LSTM when $k = 10$ where it reaches the value of 54.3% against RAF reaching 53.8%, deep NN 53.7% and logistic regression with 52.2%. However, the edge is nullified from $k = 100$ on.

- Portfolios created with LSTM show the highest daily returns: daily returns

prior to transaction costs are at 0.46%, compared to 0.43% for RAF, 0.32% for deep NN, and 0.26% for the logistic regression in $k = 10$. Also in case of larger k, LSTM achieves the highest results with a single exception at $k = 200$ where it is tied with RAF. In terms of annualised returns, for $k = 10$, LSTM achieves a performance of 82.29% after transaction costs, RAF of 67.87%, deep NN of 24.6%, logistic regression of 7.11% and the general market of 9.25%.

- In terms of standard deviation, LSTM and RAF present similar results. In any case, both of them exhibit much lower standard deviation than logistic regression and deep NN.

- In terms of Sharpe ratio, LSTM has the highest performance up to $k = 100$ (e.g. at $k = 10$ LSTM has 5.83 and RAF 5). After $k = 100$, RAF perform better than LSTM given low standard deviation of returns.

From this findings, authors evidence how the *"LSTM networks, which are inherently suitable for time series prediction tasks, outperform shallow tree-based models as well as standard deep learning"* (Fischer and Krauss, 2018, p. 8). Moreover, carrying out a critical review of LSTM profitability over different time periods, the following findings are highlighted:

- Financial performance from LSTM and RAF from 1993 to 2000 are strong. Cumulative payouts on 1$ average invest per day reach a level of over 11$ for the LSTM and over 8$ for the RAF until 2000. Anyhow, it is worth mentioning how LSTM has been introduced in late 90s and was feasibly deployed after 2000s. Thus the exceptionally high returns may be driven by the fact that LSTM was unknown or unfeasible for the majority of investors. Similar arguments hold for RAF.

- In the period 2001-2009, authors find still positive returns but lowered compared to the previous period. Authors states how *"it seems reasonable to believe that this period of moderation is caused by an increasing diffusion of*

such strategies among industry professionals, thus gradually eroding profitability" (Fischer and Krauss, 2018, p. 2).

- Surprisingly, RAF performs incredibly well after 2008 compared to LSTM with a Sharpe ratio up to 6. This results is peculiar since for all other algorithms, as said, the return lowered over time given their implementation by an increasing number of financial operators. Authors attribute the result to RAF's robustness to noise and outliers which play out in such volatile times. Moreover, following the literature, they explain the result in light of the particular turmoil situation in which operator may have created relative value arbitrage opportunities and by the fact that, in such periods, limits to arbitrage are exceptionally high, making it hard to capture such opportunities (e.g. short selling costs to borrow stocks may rise or returns may be limited by widening spreads and decreasing liquidity).

- In the last period, 2010-2015, RAF loses its edge destroying more than 1$ based on an average investment of 1$ a day. Conversely, LSTM continues to achieve higher accuracy and to maintain the capital almost constant after transaction costs.

Finally, authors shed a light on common patterns in top and flop stocks selected by LSTM. LSTM is found to be able to spot patterns related to well known market anomalies by its own, none of the identified characteristics has been explicitly coded as feature indeed:

- Flop and top stocks exhibit below-mean momentum, meaning that they perform poorly from day $t - 240$ to day $t - 10$ compared to the cross-sectional mean. At day $t - 9$, top stocks start crashing and, by contrast, flop stocks show increasing returns. Thus. LSTM is found to realise a sort of *short-term reversal strategy*: *"The LSTM network seems to independently find the stock market anomaly, that stocks that sharply fall in the last days then tend to rise in the next period and vice versa."* (Fischer and Krauss, 2018, p. 11).

- High volatility stocks tend to be preferred. Among most frequently selected stocks, it is possible to find those ones with higher Beta. Moreover, they are also more leptokurtic than general market.

- Looking at the industry of stocks selected in the portfolios, it is possible to observe how LSTM focuses on specific industries during specific time period (e.g. tech sector during rising of the .com bubble in late 90s, financial sector during the financial crisis period 2008/2009).

Based on the these common patterns, authors create a simplified rules-based trading strategy. Specifically, they short short-term winners and buy short-term losers, and hold the position for one day. With this transparent and simplified strategy, they manage to obtain returns of 0.23% per day prior to transaction costs (i.e. about 50% of the LSTM returns). In conclusion, Fischer and Krauss find how deep learning, in the form of LSTM networks, seems to effectively constitute an advancement in financial forecasting prediction tasks.

Summarizing, at the end of the review it results clear why financial ML literature today founds its research upon LSTM networks. Standard recurrent neural networks have already represented a great evolution in ML financial research transforming neural networks from static models able to incredibly approach non-linear dependencies among variables, to dynamic models able to extract such non-linear dependencies from a sequence of correlated observations. Possibly, LSTM have moved the performance frontier set by RNN further. We have seen how Fischer and Krauss (2018) found in their empirical study how LSTM networks over-perform "classic" ML models such as RAF and logistic regression in forecasting stock price movement. But even more importantly, they found LSTM to be better in forecasting and in generating an investment strategy than deep feed-forward neural networks. Sirignano and Cont (2019) helped understanding the results uncovering evidence of non-linearity and long memory in financial time-series. Particularly this last feature results to be the key factor to understand LSTM suitability to financial market

predictions. Liu and Liu (2018) found indeed how two variations of standard RNN specifically created to consider long-term dependencies such as LSTM and GRU managed to perform better than standard RNN in their empirical research. Finally, it is worth to remember that the experimentation dealing with LSTM and financial market predictions is quite new and there are not many approaches from the literature about it, especially if compared with the vast amount of researches performed through the years dealing with other ML algorithms. Nonetheless, for instance from Chen and Ge (2019), it appears clear how also in finance, as in other ML fields, the following step in the research dealing with stock market predictions will have to do with the optimization of all the processes related with information content provided to the algorithm whose attention mechanism are among the most prominent example of it.

Chapter 3

Machine Learning for Financial Market Predictions

After the literature and before the presentation of the experimental work, in this chapter the thesis aims to define a theoretical framework regarding the application of machine learning models for financial market predictions. Even if not containing any research novelty, this attempt has to be considered an important contribution since it merges theoretical concepts from both machine learning and finance. Until now, in fact, it has not been yet emerged a well-defined field of study combining this two domains, even if their union has been reality in the business world for many years.

The framework will evidently begin defining machine learning and the related concept of statistical arbitrage (**Section 3.1**). After that, it will focus on a detailed description of the functioning of all the main algorithms emerged from the literature review (**Section 3.2**). As third aspect it will treat features selection with a synthetic taxonomy of the most frequently employed predictors in financial applications (**Section 3.3**). Finally, the last, and probably most important, facet treated by the framework will be the difference, peculiar to ML financial applications, between data science and financial performances of ML models (**Section 3.4**).

3.1 Machine Learning and Statistical Learning

The creation of the term "Machine Learning" is attributed to Arthur L. Samuel, an American computer scientist, that in his paper, working on programming a computer to be better in playing the game of checkers than a person, wrote *"Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort"*(Samuel, 1959). The definition has been then transformed and nowadays Machine Learning is referred as *"the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead"* (Wikipedia, 2020). In particular, the field regarding the research on statistical models used by computers to perform specific actions goes under the name of *statistical learning* and it includes both machine learning algorithms and classic statistical models.

Following the line of reasoning proposed by Hastie et al. (2013, p. 16-30), the problem of Statistical Learning, can be summarised as follows: it is supposed to observe a quantitative response Y and p different predictors $X_1, X_2...X_p$; it is also assumed a relationship between Y and X in the form $Y = f(X) + \epsilon$ where f is some fixed but unknown function of $X_1...X_p$ and ϵ is a random error term, independent of X and with mean zero. In such a formulation f represents the systematic information that X provides about Y and, since it is generally unknown, Statistical Learning refers to a set of approaches for estimating it. Once the estimated function of f, represented as $\hat{f}$, is identified then it can be employed either to make prediction or to make inference.

Focusing on prediction applications, they aim to predict Y through the estimation $\hat{Y} = \hat{f}(x)$. In this formulation, at first, $\hat{f}$ is often treated as a black box since it is not concerned with the exact form of f within the model that creates it. Secondly, $\hat{f}$ is not a perfect estimate for f, thus meaning that an error is always present in predictions. Actually, $\hat{f}$ is not the only source of error in predictions; in fact, even if

it were possible to have a perfect estimate for f, the error would be still introduced by the presence of ϵ. ϵ refers indeed to all the quantity of information remained unexplained by the model due to unmeasured variables or immeasurable variations. Mathematically, the behavior of the prediction error (i.e. the difference between the true value Y and the predicted value $\hat{Y}$) can be understood looking at its variance:

$$E(Y - \hat{Y})^2 = E[f(X) + \epsilon - \hat{f}(X)]^2 = [f(X) - \hat{f}(X)]^2 + var(\epsilon) \tag{3.1}$$

From (3.1) it is possible to see how, even if $f(x)$ was equal to its estimated value $\hat{f}(x)$, the irreducible error given by ϵ would continue to provide an upper bound on the accuracy of the prediction of Y, always quantitatively unknown. By definition, ϵ has a positive variance indeed.

Summarizing, the goal of Statistical Learning methods for predictions is thus to find a function $\hat{f}$ such that $y \approx \hat{f}(x)$ for any observation (x, y) by teaching/training a specific model with the support of a given set of observations (X, Y).

As we have anticipated, statistical learning includes both traditional statistical models and more recent machine learning ones. Despite some similarities, they are really different. The first important distinction between them resides in the dimension of the set of observations (X, Y): ML applications usually deals with a massive sample with high dimensionality (i.e. Big Data), while classical statics with smaller datasets and fewer attributes. The second difference is that classical statistics is "model driven" meaning that the model definition relies on strong assumptions on data such as normality, no multicollinearity, homoscedasticity etc. This type of approach is particularly useful when data are scarce and there is a good knowledge on their underlying structure. At the opposite, Machine Learning is not assumption dependent, its approach is heavily reliant on data itself and makes few assumptions on the problem structure.

In Machine Learning literature different models have been proposed to deal with prediction problems. It is possible to categorised them following three main criteria: their functional structure, their objective and the way in which they are trained.

Parametric and non-parametric models The first important distinction that can be made is the difference between parametric and non-parametric models. Parametric models involve assumptions about the functional form of f, so that the problem of estimating f is reduced to estimate a set of parameters. When dealing with parametric models, it is really important the choice about the number of parameters that influences the flexibility of the model. The trade-off is between a too flexible models (i.e. model with many parameters) that can lead to a phenomenon known as *overfitting*, that will be better explained later, where $\hat{f}$ picks up the noise ϵ instead of the real signal, and a rigid model not able to approximate correctly the true relationships between features and observed outputs. On the contrary, in non-parametric models there are no explicit assumptions about the functional form of f. Their major advantage resides in avoiding the possibility that the functional form used to estimate $\hat{f}$ is very different from the true f, creating a model that will fit poorly the data. On the opposite, non-parametric approaches suffer of the disadvantage of requiring large amount of observations to obtain an accurate estimate for f.

Regression and classification approach Another important difference is the one regarding regression and classification approaches. Problems with a quantitative response are referred as regression problems, while problems with a qualitative response (i.e. a response that takes on value among one of k different classes) are referred as classification problems. Most of the time, the same algorithm can be applied either to a regression problem and to a classification problem with very little tuning.

Supervised, unsupervised and reinforcement learning Finally, the last important distinction in Machine Learning models is the one among supervised, unsupervised and reinforcement learning approaches. Supervised learning approaches are the ones presented so far, in which for each observation of the predictor measurement(s) X_i is present an associated response measurement Y_i. In this approach the aim of prediction is to fit a model that relates the response Y to the predictors X. In

any case, it is also possible to deal with unsupervised learning approaches where for each observation X_i there is no associate response Y_i. The term "unsupervised" thus refers to the lack of a response variable that can supervise the analysis, directing the training process. Finally, reinforcement learning can be considered a particular case of unsupervised approaches since it does not require labelled data. In this approach the learning system observing the environment, select and perform actions. From its action, it gets rewards or penalties based on a well defined policy and, basing on these feed-backs, the system is able to learn the best strategy to interact with the environment maximising its rewards.

Since the application presented in Chapter 4, as the majority of application dealing with financial time-series, refers to a supervised approach, throughout the rest of the chapter only supervised approach will be addressed so that the term ML model will always indicate a supervised approach. Nonetheless, reinforcement learning is proving to produce incredible results in fields such as robotics and it is starting to be used even for financial applications (e.g. Almahdi and Yang, 2019).

Supervised Learning

Every supervised learning approach can be decomposed in three separated phases: learning in-sample, testing out-of-sample and predicting.

In the *learning in-sample* phase, the model selected requires to be trained with a specific *training set*. This learning phase is crucial, since during this process the model is set to reproduce, as accurately as possible, the real function f from the information that it has been able to extract from observations in training set. The training process is a sensible part of any ML application: it must be controlled the trade-off between the construction of a model able to generalise the relations found in the training dataset without losing in accuracy of its forecast. In fact, there is the risk of a model adapting itself to be perfect in delivering a response for the data inside the training set but then found totally unable to do the same for observations never seen before. So, the attention during this phase must be directed towards

the control of the empirical relations learned during the training process to be really general for the whole population and not specific for observations inside the training set.

To verify the universality of the model, during the *testing out-of-sample* phase, the model is usually required to predict on an unseen set of observations called *test set*. Through specific measures of error, in this phase the capability of the model not to be specific to the training data is verified.

Focusing on the trade-off concept, it is usually referred as the *variance-bias trade-off*; through a mathematical proof, it is possible to show that the expected error of a model can be always decomposed into the sum of three fundamental quantities: bias, Variance and an irreducible error as

$$Err(x) = Bias^2 + Variance + Irreducible\ Error \tag{3.2}$$

Bias refers to the difference between the average prediction of our model and the correct value which we are trying to predict (i.e. it refers to the error introduced by approximating a real-life problem by a much simpler model), variance is the variability of model prediction for a given data point (i.e. the amount by which $\hat{f}$ would change if estimated using different training data; theoretically the estimate for f should not vary between different training sets) and irreducible error is the error related to the presence of ϵ. Equation (3.2) tells us that in order to minimise the expected test error (i.e. error obtained during the *testing out-of-sample* phase) it is necessary to select a method that simultaneously achieves low variance and low bias: models with high bias will pay very little attention to training data and will oversimplifies the real relationship between input features and output, while models with high variance will pay a lot of attention to training data and will not generalise on data that they have never seen before. So, this second phase results to be important to verify bias and variance measures.

The usual shape of training and test error related to model complexity (depending on the model, it can be related to many other parameter concerning the training

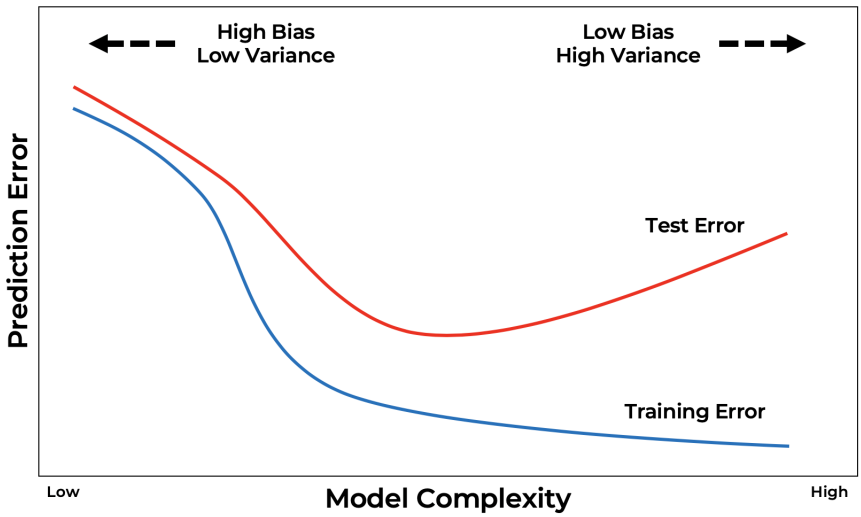

Figure 3.1: Test and training error as a function of model complexity

process) is illustrated in figure 3.1. The key aspect to look at is how the training error decreases whenever the data are tried to be replicated more accurately. At the same time, however, with too much fitting the model adapts itself too closely to the training data and it becomes unable to generalise: we can see how the test sample, over a certain degree of complexity, start increasing while the training sample is always decreasing.

Finally, from bias-variance trade-off that both the concepts of *underfitting*, that happens when a model is unable to capture the underlying pattern of data (model with high bias and low variance), and *overfitting*, that happens when a model captures the noise along with the underlying pattern in data (model with low bias and high variance) derives. Figure 3.2 gives an easy graphical interpretation of the bias-variance trade-off in linear regression.

Measuring the test error requires to leave out a part of the available observations from the training process. The problem in doing so is that machine Learning methods tend to perform worse when trained on fewer observations. So, splitting the entire dataset into two parts, training the model on the first part and measuring a

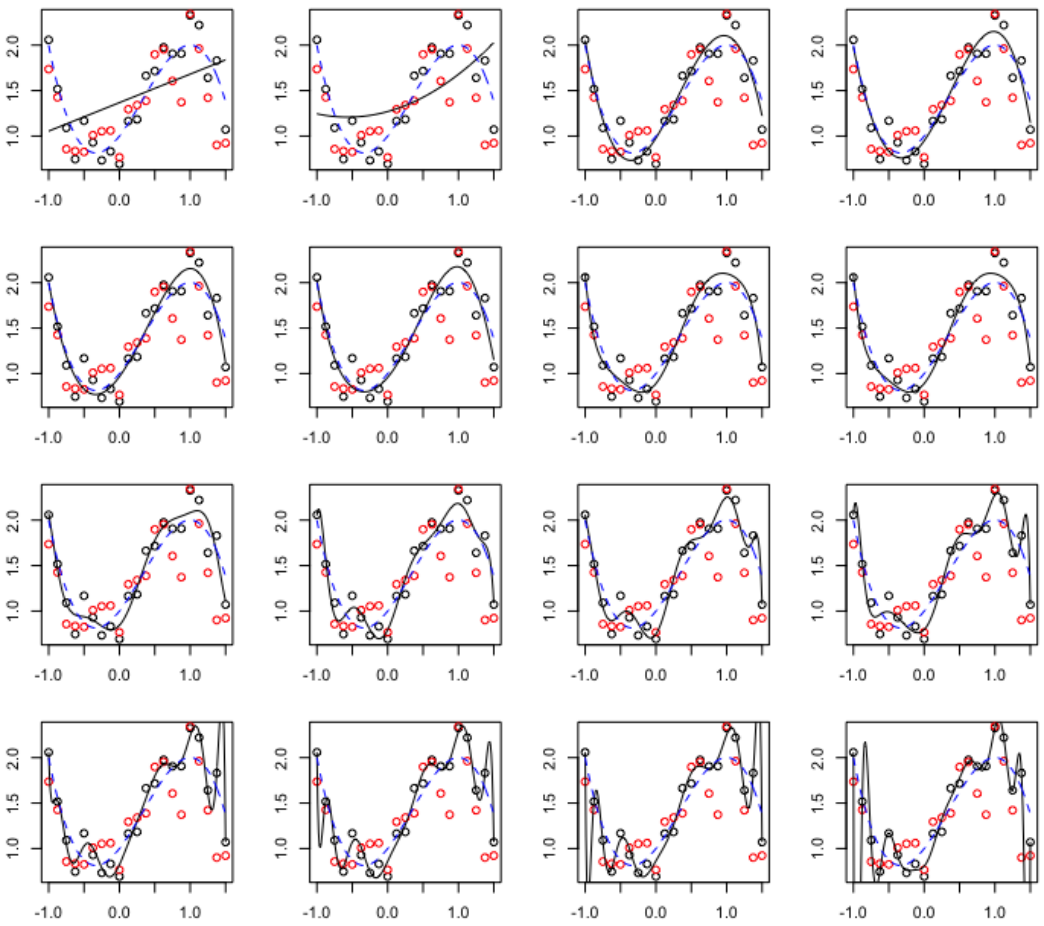

Figure 3.2: Linear regression models with polynomials of increasing order, from 1 to 16. Data are generated from the function represented by the blue line plus a white noise. Black dots represent training set on which the models have been trained and red dots represent the test set. It is possible to see how models of first and second order clearly underfit data and models from the ninth order on overfit data.

test error on the second one might bring to overestimate the test error for the same model fit on the entire dataset. For this reason, methods such as *cross-validation* are used not to lose valuable information in the training phase and at the same time to have a good measure of out-of-sample accuracy/precision. Anyhow, such methodologies are not relevant for time-series: for time-series observations there is the constraint of time that impose to maintain the data in a chronological sequence since they are correlated; methodologies such as cross-validation would require to shuffle observations and use more recent data in training set and older data as test

set. This is the reason why in the application presented in this thesis a training window approach will be applied; anyhow, its functioning will be accurately presented in Chapter 4.

As last step, once a model has been found to have adequate levels of both variance and bias, thus not suffering neither from overfitting not from underfitting, the last phase concerns the *prediction* of values we are effectively interesting to forecast and that will probably influence our decision process.

3.2 Machine Learning algorithms in financial applications

After the briefly introduction about basic statistical learning concepts provided in the previous section, in this section the main ML algorithms applied in literature to deal with financial time series forecasting will be presented. The order in which they will be presented wants to retrace the temporal development of the different solutions applied: over the time, as emerged from literature review, thanks to increasing computer power, more powerful algorithm have been implemented in literature to deal with financial time-series forecasting until reaching solid results in recent years. Among the first applications, the most frequent models implemented have been Classification Trees (3.2.1) and Support Vector Machines (3.2.2). After them, the research has started to shift its focus towards Neural Networks (3.2.3) due to their ability to map non-linear dependencies among data and to their impressive results in other machine learning application fields. However, the real jump in performance regarding financial time-series has occurred with Recurrent Neural networks (3.2.3) specifically designed to deal with correlated data. In particular, in the last couple of years, Neural Networks with Long-Short term memory cells (LSTM), a powerful variation of standard recurrent neural networks, have moved the performance frontier forward. The last part of the section will pay particular attention to LSTM since such algorithm has been selected to create an investment

strategy for the experiment performed in Chapter 4.

All the algorithms will be presented in their classification setting, since this is the same setting decided to adopt for the application presented in Chapter 4.

3.2.1 Classification Trees

Decision trees are among the first ML models used to deal with stock market predictions due to their simplicity. Specifically, among all the different algorithms proposed for the creation of trees, random forests and boosting are the two variations that have had the higher success in financial literature especially due to their robustness to noisy feature space. This is the reason why they will be extensively presented in this subsection.

Decision trees are methods that divide the predictor space (i.e. the set of all possible values for X matrix) into different and non-overlapping regions and, for every observation that falls into the same region, make the same prediction. They can be applied both in regression and classification problems. In both cases, to predict the output given certain features, the portion of the predictor space in which the observation lies must at first be identified; then, in a regression problem, the response the model gives equals the mean of responses in the training set for that particular region, while, in a classification problem, the predicted class refers the most commonly occurring class of training observations within the region.

The search for the optimal predictor space splitting is performed during the training phase through the support of a cost function (e.g. Residual sum of squares for regression and Gini index or cross-entropy for classification): for every step of the process, among all possible splits, the one decreasing the cost the most is selected. Figure 3.3 shows the typical form of a classification tree.

Random Forests

Random forests (Breiman, 2001) are variations of simple decision trees relying on the concept of bootstrap and the related one of bagging.

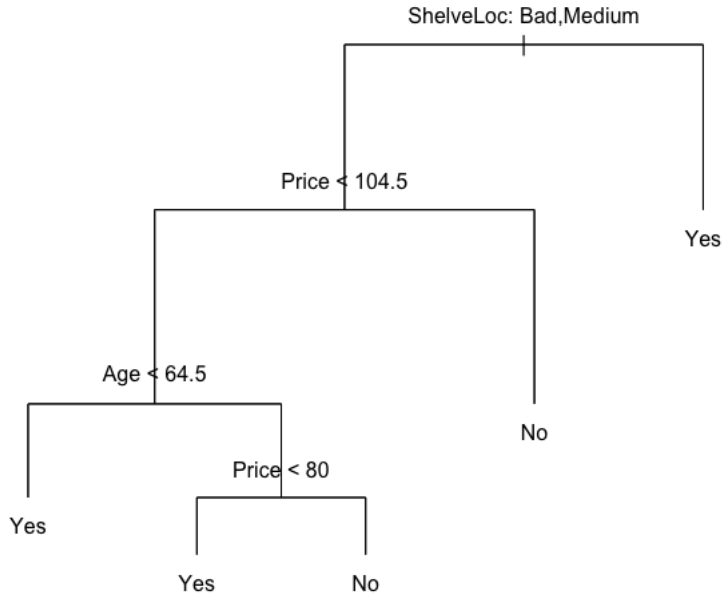

Figure 3.3: Example of a possible classification tree for CARSEATS *dataset. The dataset contains sales of child car seats at 400 different stores. Class "No" refers to stores with less than 8000$ sales, class "Yes" to stores that exceeded 8000$ sales. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch, while the right hand corresponds to $X_j \geq t_k$. The categorical variable "ShelveLoc" refers to the quality of the shelving location for the car seats at each site and takes three values:* `Bad, Medium and Good`*. The variable "Age" refers to the average age of the local population and "prices" to the price the company charges at each location.*

Bootstrap (Tibshirani and Efron, 1993) is a statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning model. The technique is based on the creation of different random samples from an original dataset with replacement (i.e. a given observation in the original dataset can be randomly selected more than once to enter in the n_{th} bootstrapped sample). The method requires each bootstrapped sample to have the same size of the original dataset. The core idea behind the method is that, from the bootstrapped samples, it is possible to estimate any aspect of the distribution of any measure computed in the training set examining, over the N replications of the original sample, its

behaviour.

The idea of bootstrapped samples has been recalled by a method called **Bagging**, algorithm that can be considered the forefather of Random Forests, to reduce the variance of statistical learning method. Bagging (Breiman, 1996) relies on the property for which, given a set of N independent observations each with variance σ^2, the variance of the mean of the observations is σ^2/N: to reduce the variance, and hence increase the accuracy, of a statistical learning method, Bagging builds many training sets from the original dataset (i.e. bootstrapped samples) and trains a separate prediction model for all the different bootstrapped sets; once all models are trained, it averages the predictions of each different model. Averaging the prediction of many weak learners, bagging manages to reduce the variance of the forecasts without affecting the expectations of predicted values. In the light of the bias-variance trade-off 3.2, bagging reduces the expected error related to the prediction by diminishing the variance and keeping at the same time the bias constant [1].

Random forests algorithm, while being very similar to bagging, provides an improvement over it focusing on the creation of decorrelated trees to be averaged. Bagging indeed suffers from the bias associated with the high probability that the same strong predictor is used most of the times by bagged trees in the top split (i.e. predictions of bagged trees are all strongly influenced by a single strong predictors thus being very similar). In a similar situation, the predictions of all trees would result to be highly correlated thus canceling the benefit of the average: the variance would be higher than $\frac{\sigma^2}{N}$ indeed. So, random forests correct this bias by forcing each split to consider a random subset of admissible predictors for each split: on average in RAF $(p-m)/p$ of the splits do not even consider the strong predictor (where m is the number of predictors available for the split a p the total number of predictors).

The strengths of RAF are that the algorithm hardly overfit the training data

[1] The expected value of the mean of all predictions is equal to the mean of expected values of each single prediction indeed, as $E(\frac{1}{N}\sum_{i=1}^{N} x_i) = \frac{1}{N}\sum_{i=1}^{N} E(x_i)$

as the number of uncorrelated trees increases and that it works well in presence of multicollinearity [2]. Moreover, RAF does not require too high processing capabilities and it requires minimum parameters tuning. The only parameters required by the algorithm are the *number of decorrelated trees*, influencing the trade-off between computational costs and marginal improvement in performance after each additional tree, the *max depth of the tree* (i.e. the number of leafs at which each tree must stop its training process), the value m of *admissible parameters* at each split (usually set as $\sqrt{p}$, where p is the total number of parameters in the database, following Hastie et al. (2013)) and the *cost function* used to create the split in the trees.

These characteristics help understand why RAF has been among the first popular algorithm for financial time-series prediction, a notoriously noisy and prone to overfitting application. Even nowadays, RAF is used as challenging benchmark for state-of-the-art ML algorithms since it has proved over the year to deliver impressive results if compared to its inner simplicity.

Boosting

Similarly to RAF, boosting starts from the same assumption of bagging: the predictions of many "weak" classifiers (i.e. classifier whose error rate is only slightly better than random guessing) can be used to create a "strong" predictor. The difference is that, in Boosting, trees are grown sequentially and not simultaneously. In fact, each tree employs information from previously grown trees. Moreover, Boosting does not involve bootstrap sampling since each tree is fit on a modified version of the original dataset.

The two main boosting algorithms found in financial literature are AdaBoost and Gradient Boosting. **AdaBoost** was firstly proposed by Freund et al. (1996), then it has been generalised in **Gradient Boosting** by Friedman et al. (2000). The main difference between the two lies in the way they create weak learner to

[2]Phenomenon in which one predictor variable can be predicted by the other input variables with high degree of accuracy

Algorithm 1 AdaBoost.M1. (Friedman et al., 2001, p. 339)

1: Observation weights w_i are initialised at $\frac{1}{N}$, with N =Number of observations in the dataset.

2: For $m = 1$ to M:

a Fit a classifier $G_m(x)$ to training data using w_i

b Compute

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{3.3}$$

c Compute

$$\alpha_m = log((1 - err_m)/err_m) \tag{3.4}$$

d Set

$$w_i \leftarrow w_i \cdot [\alpha_m \cdot I(y_i \neq G_m(x_i))] \tag{3.5}$$

3: Output

$$G(x) = sign[\sum_{m=1}^{M} \alpha_m G_m(x)] \tag{3.6}$$

be added to form a strong learner: Adaptive Boosting (AdaBoost) changes the sample distribution by modifying the weights attached to each of the instances, while Gradient Boosting trains the weak learner on the remaining errors of the strong learner. Both models, in any case, try to model the weak learners to be able to improve performance in an area of the feature space where they do not perform well.

The AdaBoost algorithm for classification presented in 1996, called *Adaboost.M1.*, is presented in Algorithm 1 while Gradient Boosted one is presented in Algorithm 2.

Starting with AdaBoost, as anticipated, it modifies data at each boosting step applying weights to each training observations. Initially, all weights are set to $1/N$, while for each successive iteration, observations misclassified by the classifier have their weight increased in (3.5), whereas for those that were correctly classified weights are decreased. Each successive classifier is thereby forced to concentrate on observations missed by previous models.

Algorithm 2 Gradient Boosting (Hastie et al., 2013, p. 322)

1: All the predictors $\hat{f}$ are set at $\hat{f} = 0$ for all observations in the dataset. In this way residuals $r_i = y_i$.

2: For $b = 1$ to B:

a Fit a tree $\hat{f}^b$ with d splits to the training data (X_i, r_i)

b Update $\hat{f}$ adding a shrunken version of the new tree

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x_i) \tag{3.7}$$

c Update residuals

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i) \tag{3.8}$$

3: The final output of the boosted model will be

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x) \tag{3.9}$$

Gradient Boosting, instead, fits a tree using current residuals, rather than the effective outcome Y, as response. Each fitted tree is added to a total decision function in (3.7) in order to update residuals in (3.8). The updated residuals, will be then used in (2.a) to fit the following tree. By fitting small trees to residuals, $\hat{f}$ is slowly improved in areas of the feature space where it does not perform well. This is the reason why in Boosting the construction of each tree strongly depends on trees that have already been grown and the reason why, instead of fitting a single large decision trees to data, the boosting approach is said to "learn slowly".

Like RAF, boosting algorithms requires few parameters to be set: the *number of trees* b (with the difference that here, unlike bagging and RAF, a too large value can bring to overfitting), the *shrinkage parameter* λ that controls the rate at which boosting learns and the *number of splits* per each tree d. Boosting, in both AdaBoost and Gradient Boosting version, has proved to posses impressive computational scalability (i.e. it can handle thousands of predictors), to have high capacity to deal with irrelevant inputs and to tend to perform better when handling data of mixed type (i.e. continuous and categorical). Finally, as RAF, generalised boosting

classifiers have a good level of interpretability: it can provide a ranking of variable influences and their marginal effect on the response.

Boosting algorithms, as opposed to RAF, are rarely employed for financial applications nowadays. They have never shown impressive performance when applied to financial time-series prediction indeed, especially in terms of financial performances.

3.2.2 Support Vector Machines

Along with random forests, support vector machines gained a lot of popularity among the first machine learning finance applications in literature. Before the massive implementation of neural networks, in fact, SVM have been selected when dealing with stock price time-series due to their inner ability to look for non-linear decision boundaries within the feature space.

Support Vector Machines (Ben-Hur et al., 2001) algorithm lies on the simple concept of maximal margin classifier and further extend it. The **maximal margin classifier** is a method to classify data through the *maximal margin hyperplane*, that is the separating hyperplane for which the margin (i.e. the minimal perpendicular distance from training observations and a separating hyperplane that divides the feature space in two parts) is largest (Figure 3.4).

Given the definition, it is demonstrable that the functional form of the maximal margin hyperplane depends only on the nearest observations to the plane within the training set, the ones which lies on the maximum margin, and not on the others: looking at figure 3.4, the position of the hyperplane represented depends only on the three squared points lying on the dashed maximum margins. If these points were moved, then the hyperplane would move as well. This is the reason why these points are called *support vectors* since they define the position of the maximal margin hyperplane. Now, the problem with Maximal Marginal Classifier is that it works only if data can be perfectly and linearly separable by a hyperplane. Moreover, the maximal margin classifier is extremely sensitive to change in a single observation, risking to overfit the training data.

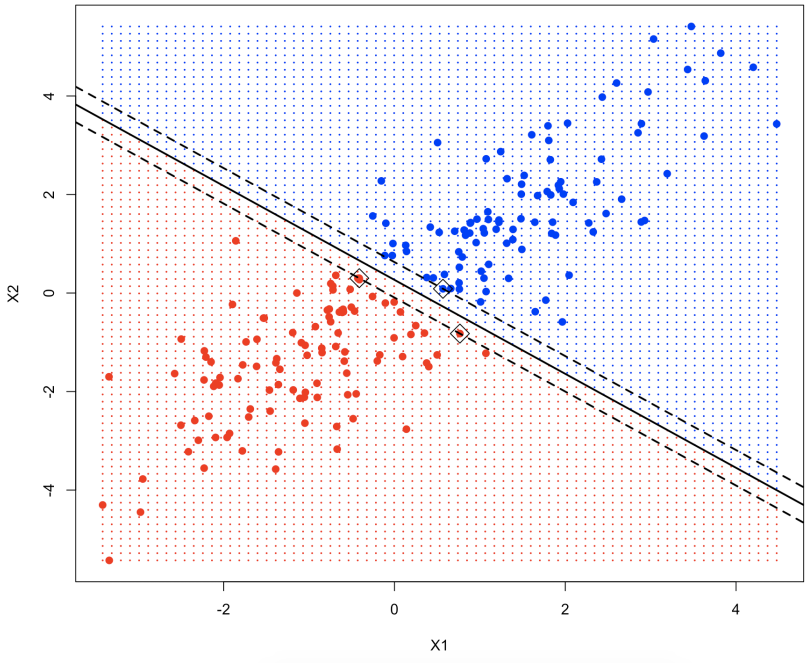

Figure 3.4: Example of a maximal margin classifier

So, the natural extension of maximal margin classifier is called **support vector classifier**. It accounts for the possibility of the two classes in training data not to be perfectly separable and it can be considered the forefather of SVM. In mathematical terms, the support vector classifier can be written as a maximization problem of the form:

$$\max_{\beta_0, \beta_1, ..., \beta_p, \varepsilon_1, ..., \varepsilon_n} \quad M \tag{3.10a}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} \beta_j^2 = 1, \tag{3.10b}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M(1 - \varepsilon_i), \tag{3.10c}$$

$$\varepsilon_i \geq 0, \tag{3.10d}$$

$$\sum_{i=1}^{n} \varepsilon_i \leq C \tag{3.10e}$$

Where C is a non-negative tuning parameter, M is the width of the margin which we want to make as large as possible and ε_i are slack variables which allow observations

to be on the wrong side of the margin or hyper-plane (i.e. "soft classifier" approach).

From its functional form, it is possible to observe how the classifier controls the position of training observations in the feature space on the basis of the sign in the left side of equation (3.10c): it is easy to understand that the classifier correctly classifies the i^{th} observations if $y_i\overline{\beta}\boldsymbol{X_i} > 0$ (while when $y_i\overline{\beta}\boldsymbol{X_i} > M$, it means that the observation concurrently lies on the right side of the plane and at a distance larger than the margin).

In any case, up to this point, the problem results the same of the one of maximal margin classifier. The novelty is introduced by the presence of ε. It accounts for this model for the possibility for observations to lie on the wrong side of the hyperplane. In fact, when $\varepsilon_i \geq 0$, then the i_{th} observation is on wrong side of the margin and when $\varepsilon_i \geq 1$, i_{th} observation is on wrong side of the hyper-plane. However, the possibility for the classifier to wrongly classifies observation is bounded by the tuning parameter C: it bounds the sum of the ε_i and sets the tolerance level for misclassification. Larger values of C indicate larger tolerance. Importantly, C controls the bias-variance trade off: higher C indicates a higher margin (i.e. many observations violate the margin) and therefore a higher number of support vectors, resulting in a more biased but less variable model.

However, also this model presents a limitation: even with the extensions related to ε and C, the linear boundary hypothesis persists. It can be easily understood why this still results a strong limitation for the algorithm from the example in Figure 3.5: suppose to have a dataset as the one in the left figure and to train a support vector classifier from it. From the right image, it can be seen how such a classifier will perform poorly in properly separating the data.

Finally, **Support vector machines** comes to solve the problem of linear boundaries: they enlarge the feature space to deal with nonlinear decision boundaries through the use of the so-called *kernels*. The kernel is a function applied on each data instance to map original observations into a higher-dimensional space so as to make them as much as possible linearly separable. In this enlarged space the

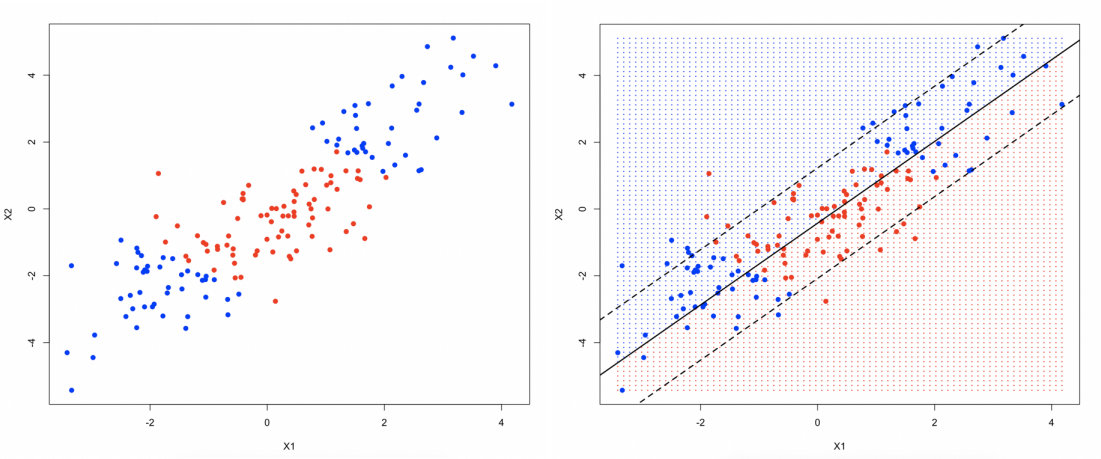

Figure 3.5: Example of a support vector classifier employed to classify observations from a dataset that requires non-linear boundaries.

algorithm looks for a linear boundary that, when re-mapped in the original feature space, will not result clearly linear.

Mathematically, kernels are different type of functions that are used to transform the linear boundary in (3.10c) in a non-linear one, taking advantage of a mathematical property for which the solution of the problem (3.10a) regards only the inner product of the observations. SVM can be written as the same maximization problem of the support vector classifier with the exception of (3.10c) that, with kernel, is transformed in:

$$f(x) = \beta_0 + \sum \beta_i K(x, x_i) \tag{3.11}$$

Where $K(x, x_i)$ represents the selected kernel.

Said that, certainly the most important decision to be taken when dealing with SVM is the type of kernel to be employed. Among the most successful kernels in financial applications, it is possible to find *polynomial kernel* (3.12), *radial basis kernel* (3.13) and *sigmoid kernel* (3.14):

$$K(x, x_i) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{3.12}$$

$$K(x, x_i) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{3.13}$$

$$K(x, x_i) = tanh(\alpha x^T y + c) \tag{3.14}$$

The optimization of parameters such as d in (3.12), γ in (3.13) or α in (3.14) in kernel functions is usually performed through a grid search, where performances achieved by SVM using different values for such parameters are compared and the best one is selected. Figure 3.6 shows how when dealing with the previously presented database (Figure 3.5), SVM with radial kernel is able to correctly classify basically all observations.

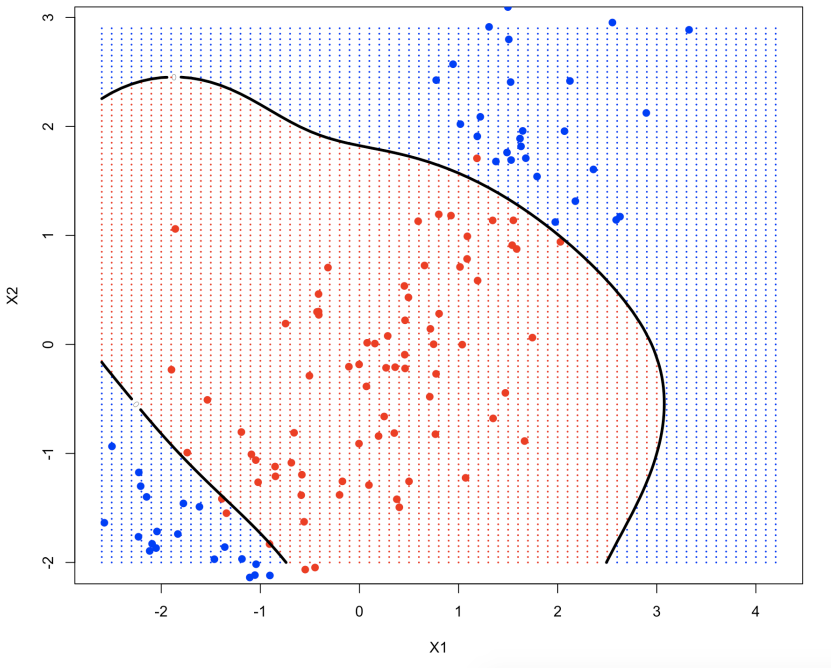

Figure 3.6: SVM with radial kernel applied to the dataset from figure 3.5

Since dynamics of financial time-series are strongly non-linear, it is intuitively understandable why, in theory, SVM with non-linear kernel functions have been extensively employed with financial time-series. Moreover, SVM, compared to other classifiers, are designed to minimise the structural risk, while other techniques are based on the minimization of the empirical risk; that is, SVM seeks to minimise an upper bound of the generalization error rather than minimizing training error. This makes them less vulnerable to overfitting than other methods; the solution of

the optimization problem (3.10a) is unique and absent from local minima indeed. Finally, SVM is easy to modify and it is able to handle high-dimensional data. In any case, it requires long time to be trained since it has a large computational complexity that depends on the kernel function and it suffers in terms of computational scalability, lack of interpretability and ability to handle irrelevant inputs and data of mixed type.

3.2.3 Neural Networks

Briefly, neural networks are a family of ML models intended to stimulate the behavior of biological systems composed of neurons. They are oriented graphs consisting of nodes, which in the biological analogy represent neurons, connected by arcs, which correspond to synapses. The history of neural networks starts around 1950 when *perceptrons* were presented. Perceptrons networks then have evolved into *feed-forward neural networks* and after them, in the last years *recurrent neural networks*, along with one of its famous variant *LSTM*, were proposed.

Even in ML financial applications, as in many other fields, the passage towards the family of neural networks algorithms has been natural after the first implementations especially, as just explained, taking advantage of classification trees and support vector machines. Neural networks have in fact led to a breakthrough in various fields of machine learning applications especially due to their ability to map non-linear dependencies among variables and due to their scalability.

This is the reason why they will be accurately explained throughout the section, in particular we will follow their evolution from static models (NN) to dynamic models (RNN) until presenting one neural networks' state-of-the-art variation that is LSTM networks. The evolution from static to dynamic models has been indeed crucial in terms of accuracy of stock market predictions since it has equipped networks to learn not just from static single observations, as the models presented so far, but from long-term sequences of highly correlated data.

Nowadays, among the most promising financial applications, we can find the

ones achieved through the implementation of long short-term memory networks able to achieve impressive results if compared to the ones obtained by classification trees, support vector machines and even standard feed-forward neural networks. Anyhow, it is possible to understand their functioning only once the functioning and limitations of both feed-forward neural networks and recurrent neural networks are clear.

Perceptrons

To start talking about Neural Network, it is important to start from the concept of perceptron (Rosenblatt, 1958). A perceptron is an operator that takes several binary inputs (i.e. explanatory variables) $x_1, x_2...$ and produces a single binary output. To produce the output, it uses weights w_i that show the importance of the respective inputs to the output. The neuron's output, either 0 or 1, is determined by whether the weighted sum $\sum_j w_j x_j$ is greater or not than a threshold value ϑ, being the threshold a real number. So, perceptrons differentiate one from another on the basis of their weights and threshold value.

Expressing the condition about the output to be 1 in this form $\sum_j w_j x_j - \vartheta \geq 0$, it is possible to understand why the value ϑ (i.e. the treshold value) is referred as "bias": it gives a measure of the easiness to get the perceptron to output a 1.

Figure 3.7 graphically shows how a perceptron works: it takes as input values X and returns as output $f(X)$. Supposing the values of weights and bias already determined, the computation of $f(X)$ is performed through the following steps:

- The weighted linear combination of explanatory variables X is calculated and the bias is subtracted from it:

$$z = w_1 x_1 + w_2 x_2 + ... + w_n x_n - \vartheta \tag{3.15}$$

- The output $f(x)$ is then obtained by applying the activation function $g(z)$ to the value calculated in (3.15):

$$f(x) = g(w_1 x_1 + w_2 x_2 + ... + w_n x_n - \vartheta) \tag{3.16}$$

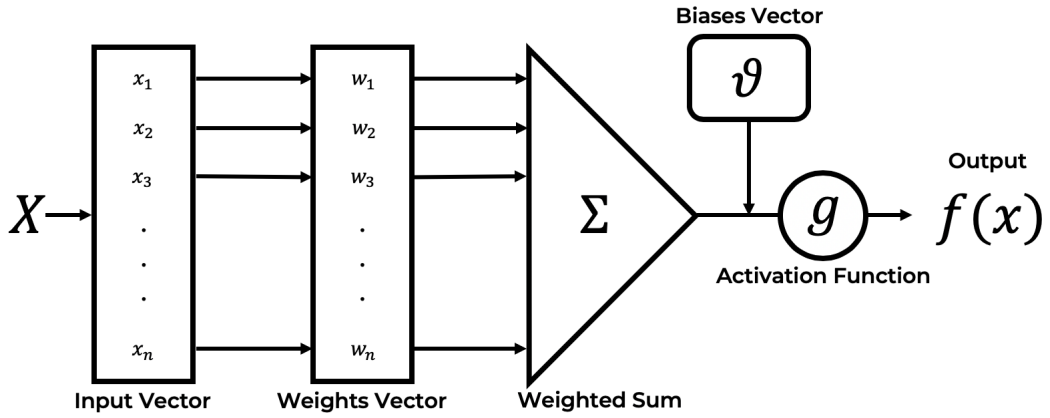

Figure 3.7: Operations performed by a single perceptron to create an output $f(X)$ from X input.

In the case of perceptron, the activation function $g(z)$, as can be easily imagined, is equal to the unit step function returning a value of 1 if (3.15) is positive and 0 otherwise.

Such a functioning will be helpful to understand how a neuron in feed-forward neural networks works.

Feed-forward neural networks

A feed-forward neural network (NN) deals with an evolved type of neurons compared to perceptrons, even if they show many similarities. The significant difference between the two operators lies in the use by neurons in NN of an activation function $g(x)$ that, instead of returning a binary output 0 or 1, returns a continuous value. This difference is important since makes the output $f(x)$ of the operator more sensitive to small changes in weights or bias. For instance, one of the most well-known activation function in NN is the sigmoid function that is defined as:

$$\sigma(z) = \frac{1}{(1 + e^{-z})} \tag{3.17}$$

Where z refers to the quantity in (3.15).

To understand how such a function differs from the behavior of a unit step function, look at Figure 3.8 that graphically shows the difference between the two activation functions. For extreme values the behavior of a sigmoid neuron closely approximates the behaviour of a perceptron: suppose $z = \sum_j w_j x_j - \vartheta$ is a large positive number, then $e^{-z} \approx 0$ and so $\sigma(z) \approx 1$; on the other hand, if z is very negative, then $e^{-z} \to \infty$ and $\sigma(z) \approx 0$. However, when $z = \sum_j w_j x_j - \vartheta$ is of modest size there is a substantial deviation from the perceptron's behaviour; it is precisely in this range that the NN neuron becomes sensible to small changes of weights and bias.

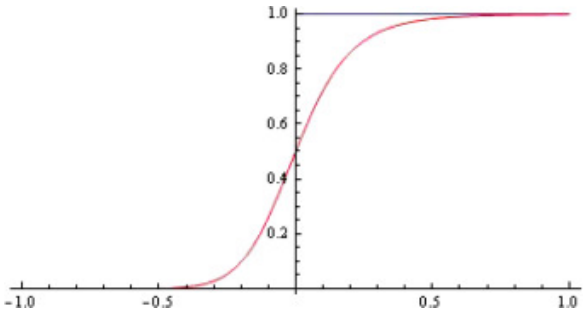

Figure 3.8: Comparison between a sigmoid function (in red) and a unitary step function (in blue).

In any case, sigmoid function, even if certainly among the most frequently used, is not the only activation function that is possible to use to increase the sensibility of the operator. The possibilities for the choice of the activation function are indeed numerous. Among other frequently employed functions we can find *ReLu* and *hyperbolic tangent (tanh)*. The choice of the activation function results to be very important since, as we will see when treating backpropagation, it can heavily influence the performance of a model.

Now, same as perceptrons, such sensible neurons do not seem promising in performing any type of analysis if taken singularly. In fact, they become incredibly powerful when connected one with each other forming networks and passing information one from another. The appearance of a networks of neurons connected between them through different arcs forming a feed-forward neural network is shown

in Figure 3.9. The figure expresses the way in which information is passed trough the networks: arcs identify the output of neurons flowing from input nodes to output node(s) to be finally transformed in $f(x)$. Every arc from the figure can be thus taught as the means by which the information arrives at each node from the previous layer: the X vector expressed in (3.15) for a neuron in such a network is formed by all the outputs of nodes in previous layer directly connected with the neuron itself through an arc; moreover, weights in (3.15) are associated to these arcs signaling the magnitude by which information is amplified or reduced when passed trough them.

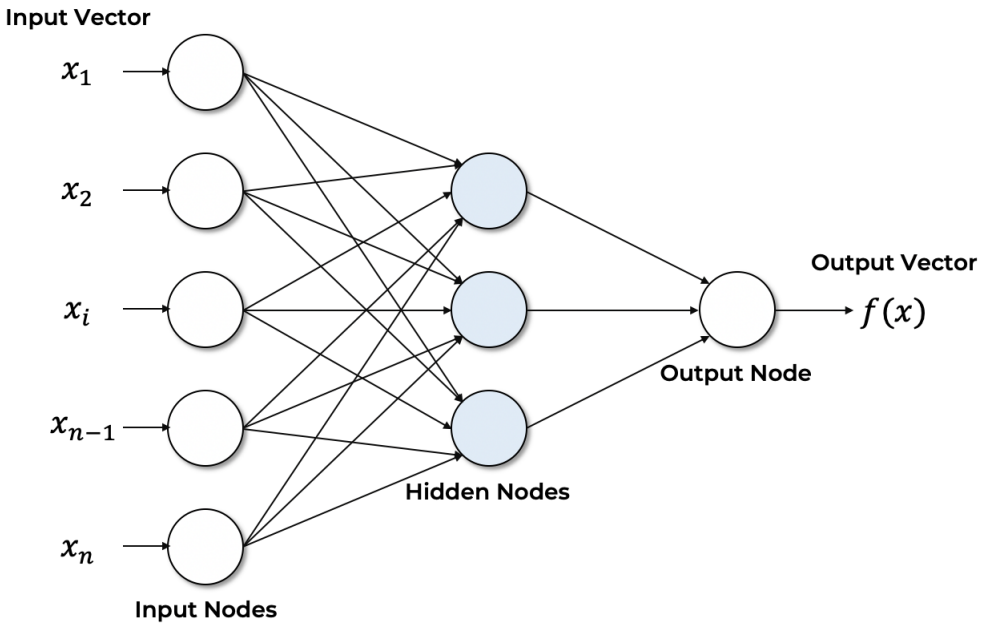

Figure 3.9: Example of neural network. In the network the nodes look like they have multiple output, in fact they are still single output; the multiple output it is just a way to indicate that the output is used as the input to other several nodes. Moreover, weights are associated with the input arcs, while each node is associated with a distortion coefficient (i.e. bias).

Figure 3.9 also clarifies in the identification of the three different type of layers to be possibly found in NN:

- **Input nodes** - Input nodes receive as inputs the values of explanatory attributes for each observation. Usually, the number of input nodes is equal to

the number of explanatory variables.

- **Hidden nodes** - Hidden nodes apply transformations to the input values inside the network. Each node is connected to incoming arcs that can come either from other hidden nodes or from input nodes, and it is connected with outgoing arcs either to output nodes or to another level of hidden nodes.
- **Output nodes** - Output nodes receive connections from hidden nodes and return an output value that corresponds to the prediction of the response variable.

Particularly, among these three layers, another crucial decision regarding the design of a neural network is about the size of hidden layers and the size of neurons in each of them. In fact, the size of input nodes directly depends on the dimension of the input vector, while the size of the output nodes is determined by the predicting application selected. Unfortunately, there is no optimization processes to support such decisions, if anything, during years, many heuristic rules have been created to help the search of a quasi-optimal configurations. Usually, the final decision about hidden layers' numerousness and their density is performed through a grid search of possible values: the NN model is trained and tested on a *validation set* (i.e. a separate set from the training and test one) for different settings of parameters and the final configuration is taken looking for the best performance among such models variations.

Once that the structure of a network (e.g. size of hidden layers and nodes, activation function) has been defined, it becomes crucial to understand how the network can learn to recognise patterns from training data to be then able to replicate them for previously unseen observations. Such a process, also known as *training process*, in the case of NN is related to the sequential adjustment of all the weights and biases within the network.

During the training phase, in fact, all the weights related to each arc and all the biases related to each node are optimised in order to find the best value that better

represent, given input vectors, the associated output. To make so, after weights are randomly initialised, in the beginning of the training process the network start to sequentially examine observations in the training set and, through the current values of weights and biases, it calculates the weighted sum of all signals entering in each neurons until arriving to the form the final output $f(x)$. Then, once the output $f(x)$ is found, an error measure accounting for the difference between the true value of y_i and the values predicted by the network is calculated. After it, values of biases and weights in the network are modified by a local minimization process and, again, the process restart calculating $f(x)$. Such a loop continues for some time until a stop condition is reached. Possibly, thanks to the minimization process, the error measure is decreased at ever step of the process making the networks as capable as possible to reproduce the relations presents in training observations (i.e. so as to make the output of the Neural Network $f(x)$ as similar as possible to the target vector of the training set Y).

Such an error minimization process, just briefly presented, is performed through a descent algorithm, a variant of the *gradient method* that in the case of neural networks it is commonly called **backpropagation algorithm**. In calculus the gradient descent method is an optimization algorithm employed to find the local minimum of a function. Its name derives from the fact that the algorithm defines its search direction by the gradient of the function to be minimised, since it takes advantage of the property for which the gradient of a function calculated in a specific point indicates the direction of fastest increase of the function itself. In the specific case of neural networks is intuitive to understand that the method, at every iterative step, aims at calculating the partial derivative of the cost function (i.e. the error function previously presented) respect to weights and biases in the network to define its search direction.

Since backpropagation algorithm will be crucial to understand the evolution of NN, in the following lines its main passages will be formally explained. At first, to understand its functioning, it is crucial to understand the mathematical passages

to arrive defining the partial derivatives of the cost function in relation to all the biases and weights in the network.

Imagine a NN with L layers and a cost function C (i.e. the function defining the error previously explained) where the index j indicates one specific neuron in the l_{th} layer; with the aim of finding the partial derivatives of the cost function respect to each weight and bias in the network, we start calculating the partial derivatives in the output layer L respect to weighted sum z in each neuron as (3.15):

$$\delta_j^L = \frac{\partial C}{\partial z_j^L} \tag{3.18}$$

Relationship (3.18) expresses the degree at which, changing the specific value of the j^{th} z_j^l in layer l and leaving all other z_j^l the same, it is possible to diminish the cost function C. Taking advantage of the chain rule from calculus, it is then possible to re-express the same partial derivative in respect to the output $f(x)$ defined in (3.16):

$$\delta_j^L = \sum_K \frac{\partial C}{\partial f(x)_k^L} \frac{\partial f(x)_k^L}{\partial z_j^L} \tag{3.19}$$

where the sum is calculated over all neurons k in the output layer. Clearly, depending the output $f(x)_k^L$ only on the weighted input z_j^L for the j^{th} neuron when $k = j$, (3.19) can be simplified to:

$$\delta_j^L = \frac{\partial C}{\partial f(x)_j^L} \frac{\partial f(x)_j^L}{\partial z_j^L} \tag{3.20}$$

Then, assuming the activation function g in (3.16) to be a sigmoid:

$$\delta_j^L = \frac{\partial C}{\partial f(x)_j^L} \sigma'(z_j^L) \tag{3.21}$$

Now, through the chain rule it is also possible to rewrite any partial derivative of any layer $\delta_j^l = \frac{\partial C}{\partial z_j^l}$ in terms of the partial derivative of the following layer $\delta_k^{l+1} = \frac{\partial C}{\partial z_k^{l+1}}$:

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1} \tag{3.22}$$

Where the last term, knowing that $z_k^{l+1} = \sum_j w_{kj}^{l+1} \sigma(z_j^l) - \vartheta_k^{l+1}$, is:

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l) \tag{3.23}$$

Substituting into (3.22), we got:

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l) \tag{3.24}$$

Finally, it is demonstrable how the quantities of interest $\frac{\partial C}{\partial b_j^l}$ and $\frac{\partial C}{\partial w_{jk}^l}$ are equal to:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{3.25}$$

$$\frac{\partial C}{\partial w_{jk}^l} = f(x)_k^{l-1} \delta_j^l \tag{3.26}$$

Once understood how, taking advantage of the chain rule, it is possible to calculate all the partial derivatives of the cost function respect to weights and biases in the network, the steps in which the backpropagation algorithm computes the gradient of the cost function (i.e. the matrix with all the partial derivatives of the cost function respect to every weight and bias in the network) should be easily comprehended in Algorithm 3.

Algorithm 3 Backpropagation algorithm

1: **Input:** it is calculated the corrisponding activation $f(x)^1$ for the input layer.
2: **Feedforward:** for each $l = 2, 3, ..., L$ the quantities z^l and $f(x)^l = \sigma(z^l)$ are computed.
3: **Backpropagation:** From the quantities in (3.21), quantities in (3.24) are computed.
4: **Output:** The gradient ∇C of the cost function is computed through (3.25) and (3.26).

Finally, at the end of the backpropagation algorithm the gradient of the cost function is computed. As said, it identifies the direction of fastest increase of the cost function in a given point thus meaning that, when multiplying its opposite for a small and positive parameter η (known as *learning rate*), it will identify a movement of the point defined by the current values of all weights and biases in the direction which does the most immediately decrease the cost function C. This is exactly the way in which the descent algorithm works in fact: in every iteration sums the vector of changes Δv

$$\Delta v = -\eta \nabla C \tag{3.27}$$

to the vector containing weights and biases possibly moving in the direction that decrease the most the cost function.

As emerged from all the passages related to the functioning of backpropagation, the parameters necessary for the algorithm to properly work are the cost function C and a value for the learning rate η. Regarding C, in regression applications, the measure of fit used to update weights and biases in the network is usually given by sum of squared errors:

$$C(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2 \tag{3.28}$$

While for classification, either the squared error or the cross-entropy functions is used, where cross entropy is defined as:

$$C(\theta) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} log f_k(x_i) \tag{3.29}$$

Even the choice of the cost function, as the one of activation function, can influence the training process of the model. There exist many techniques through which is possible to modify the cost function and improve the learning process (e.g. L1, L2 regularization).

What it has been presented so far, in any case, regards the operations performed by the descent algorithm in a single iteration. As we have said, descent algorithms are iterative processes that in the case of NN tend to adjust weights and biases of a small quantity hopefully moving towards a local minimum of the cost function. So, the reiteration of the whole process comes with the definition of two additional important parameters: number of *epochs* and *batch size*.

The number of epochs refers to the times in which an entire dataset is passed forward and backward throughout the network in Algorithm 3. In fact, the training process can last until the number of epochs has reached a predetermined value given as input parameter or a stopping condition is verified (e.g. as input parameter can be given a specified value of *patience* referring to maximum number of iterations in which the cost function is allowed not to diminish before the training procedure is stopped).

Batch size is, instead, directly related with the idea of *stochastic* and *mini-batch gradient descent*, that are variations of gradient descent employed to speed up learning process. In the standard version of the gradient descent algorithm, also known as *bacth gradient descent*, all the training data are taken into consideration to take a single step. This means that for every observation the cost function and its gradients are calculated and finally the average of the gradients of all the training examples is taken to update parameters as in (3.27). The problem of such approach is that to take one single step, it is necessary to calculate the gradient for all the data in the dataset. This, especially for large datasets, is not efficient. To deal with such issue and to decrease training time, two variants of the gradient descent algorithm are frequently applied when training NN: *stochastic gradient descent* and *mini-batch gradient descent*. Stochastic gradient descent calculates in every step the gradient using just a single observation to update weights and biases while mini-batch gradient descent neither uses all the dataset at once nor a single observation; it uses a batch of a fixed number of training examples called mini-batch and it updates weights and biases in the network using the mean gradient in this mini-batch as:

$$w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k} \tag{3.30}$$

$$b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l} \tag{3.31}$$

where m refers to the number of the training observations X_j within the mini-batch (in batch gradient descent the same calculation is performed, with the only difference that m refers to all observations within the training set). So, returning to the batch-size parameter, it refers exactly to the size m of the mini-batch, in case mini-batch gradient descent would be chosen as method to update networks' variables.

To better understand why stochastic gradient descent and mini-batch gradient descent speed up the learning process: just suppose to have a training set of 1000 observations and a mini-batch size of 50 observations. With a gradient descent algorithm, for every of the 1000 observations, the gradient of the cost function

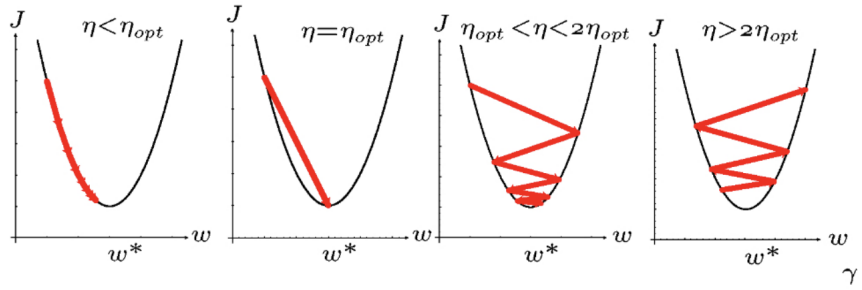

Figure 3.10: Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, the algorithm will certainly converge but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system oscillates around the local minimum but then will slowly converge. If $\eta > 2\eta_{opt}$, the algorithm will diverge.

would be computed 1000 times and then weights updated; while, with mini-batch gradient descent, the process would require only 1000/50=20 gradient calculation to updates weights and biases.

Despite being by far the most employed method to train NN, backpropagation requires to deal with several downsides. The first issue is related to the fact that optimization process heavily depends on the initial parameters choice. The second issue is related to the choice of the learning rate η: it must be at the same time small enough to respect local approximation of the function and, at the same time, it must be large enough to make the gradient descent algorithm to converge (i.e. to reach a local minimum of the cost function) in a reasonable time (Figure 3.10). In practice, η can be chosen to be decreasing in time through a procedure called *adaptive learning rate*. Moreover, the biggest issue related to backpropagation is known as vanishing/exploding gradient but it will be treated when presenting recurrent neural networks.

Concluding with a final consideration about feed-forward neural networks, one major disadvantage of NN is that if not set properly, when having too many nodes and layer, they can easily overfit training data. To deal with such problem a regularization technique called *dropout* is often applied. Ordinarily, as explained, the network is trained forward-propagating X through the network and then backprop-

agating to determine the contribution to the gradient. Applying dropout, this process is modified since for each update of parameters (i.e. for each iteration of the backpropagation algorithm) part of hidden nodes in the network are randomly and temporarily deleted. The number of hidden nodes to be temporarily deleted is an additional parameter to be provided to the model. Heuristically, dropout is similar to training different NN and finally averaging them, and due to this reason the dropout procedure is similar to average the effects of a very large number of different networks: the different networks will overfit data in different ways, but, hopefully, the net effect of dropout will be to reduce overfitting.

To recap, Neural Networks refers to a lot of possible network structures. Thus, a large effort is necessary to setup correctly the parameters that determines the network structure, the so called *Hyperparameters*. Among hyperparameters we can find: the *number of hidden layers and units*, *cost function*, *activation functions*, *learning rate*, *number of epochs or a stop condition* related to the learning process and batch size and eventual *regularization techniques*. The optimization of all these parameters sometimes can result onerous, even because there are no deterministic optimization methods, and this can result to be a limitation for the performances of the model.

Recurrent Neural Networks

Recurrent Neural Networks (RNN) (Rumelhart et al., 1986) are an evolution of feed-forward Neural Network. To understand why they have been created, it is possible to make an analogy with human thinking: people, when reading, understand each word based on the understanding of previous word and not discarding everything away starting thinking from scratch for every term (it is said that thoughts have persistence). Recurrent Neural Networks tries to simulate the same mechanism thanks to loops in them allowing information to persist (as in Figure 3.11). Their introduction have finally evolved machine learning from static to dynamic models able to take in consideration the history of data and not only a picture in a given time

frame. Clearly, such an evolution results to be meaningful for dataset containing correlated observations, as certainly financial time-series are.

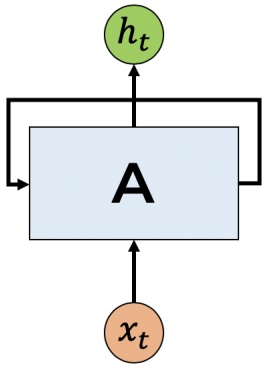

Figure 3.11: Recurrent neural network.

The loops present within the networks make the comprehension of RNN more difficult than NN. However, they are not so different from standard feed-forward neural networks. In fact, RNN can be thought of as a multiple copies of the same network (i.e. same weights and biases), each passing a message to successor as represented in Figure 3.2.3.

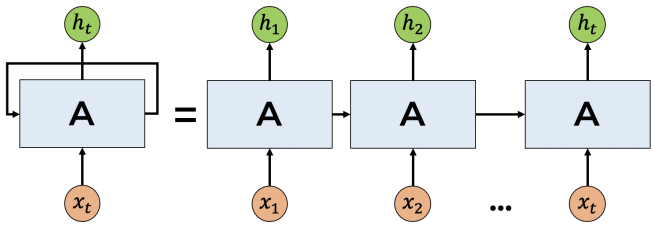

Figure 3.12: Recurrent neural network "unrolled".

Their chain-like nature reveals how RNN are intimately related to sequences. The information passed from previous networks can be viewed as the memory of the model and RNN decisions can be seen as influenced by what the model has already learnt from the past. Standard feed-forward NN remember things too, but only things learnt during each single iteration in training. While RNNs learn similarly

while training and, in addition, they remember things learnt from prior inputs while generating outputs. Outputs produced by RNN are influenced thus not only by weights applied on inputs like in NN, but also by a hidden state vector $h(t-1)$ (Figure 3.13) representing information based on prior inputs and outputs. This means that the same input at time t could produce a different output depending on previous inputs in the series. It is now easy to understand why these type of networks are particularly useful when dealing with time-series data: when there is information in the sequence of data, RNN can use it to perform tasks that feed-forward networks cannot.

Formally, RNN is a NN specialised for processing a sequence of input data $x(t)$ where the time step index ranges between a given interval. The length of this interval depends on the type of application: suppose you want to care about a sequence of three words in a sentence, then $t \in \{1, 2, 3\}$. The length of the interval directly influences the dimension of the model: referring to Figure 3.13, the network is unrolled into n subsequent networks, where n refers to the number of element in t.

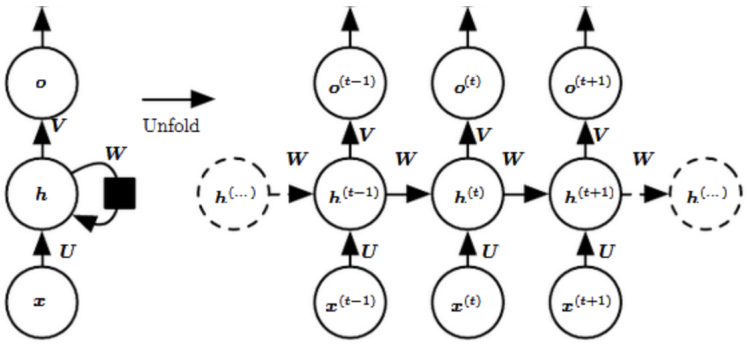

Figure 3.13: Representation of all the different variables and parameters of a RNN model

Looking at Figure 3.13 we can also describe all the components of a recurrent neural network model:

- $x(t)$ is the input vector belonging to a given sequence taken at time step $t \in \{1, ..., n\}$.

- $h(t)$ represents the hidden state at time t and acts as memory of the network. It is calculated as (3.32) from the current input $x(t)$ and the previous time step's hidden state.

- Matrix U parametrises inputs to hidden connections, matrix W hidden-to-hidden recurrent connections and matrix V hidden-to-output connections. All these weights (U, V, W) are shared between all different "unrolled networks" across t.

- $o(t)$ refers to the output of the network at every time step.

So, from all the components emerges how the forward pass, that in NN is represented as (3.16), in RNN can is represented by the following two equations:

$$h(t) = Wh^{(t-1)} + Ux^{(t)} - \vartheta \tag{3.32}$$

$$O(t) = g(Vh^{(t)} - \vartheta) \tag{3.33}$$

Regarding the training process, in such a network, the total loss function for a given sequence of $x(t)$ values paired with a sequence of $y(t)$ values is then just the sum of the losses over all the time steps, over all the "unrolled networks". The gradient computation, that also in the case of RNN is employed to update networks weights and biases, then involves performing a forward propagation pass moving left to right through the graph in Figure 3.13 followed by a backward propagation pass moving right to left through the same graph. The backpropagation algorithm follows the same logic of the one used in ANN; the only difference is given by the fact that parameters are shared by all time steps in the network, so that gradient at each output depends not only on the calculations of the current time steps, but also the previous steps. Thus, in the calculation of the gradient has to be accounted the contribution of the $h(t-1)$ term.

Figure 3.13 shows a so called many-to-many configuration, in which for each time step t the RNN predicts a different output value O_t. However, it is straightforward

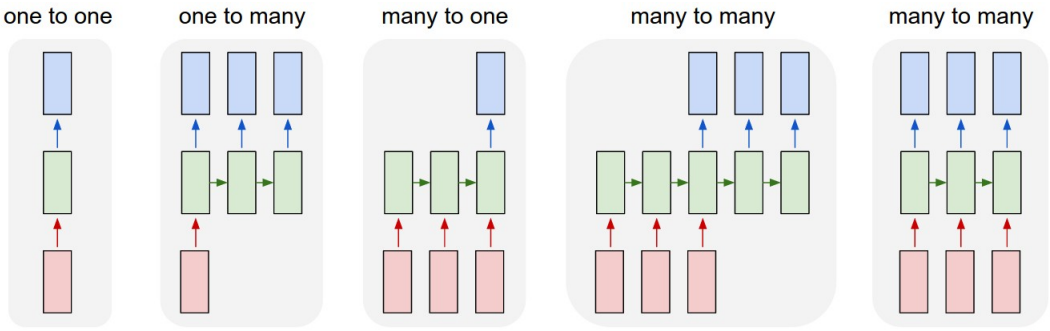

Figure 3.14: Representation of different possible RNN configurations. Rectangles indicate a network layer. Red ones refers to input layers, green ones to hidden layers and blue ones output layers. The sub-figure on the left refers to a standard feed-forward neural networks.

to think at several variations of such configuration, presented in Figure 3.14. They simply derives from different configurations of period in which the model takes observations and predicts outputs.

Anyhow, although RNN have presented in ML history an evolution towards NN as regards the study of correlated observations, they suffer from a major problem: *vanishing/exploding gradient* (Hochreiter, 1998). This problem affects also NN but, as it will be clear, in the case of RNN it becomes even more cumbersome since it affects their ability to effectively deal with long-term dependencies.

As previously explained, in the backpropagation algorithm weights and biases in the network are adjusted in an iterative way calculating the gradient of a cost function. This translates in having each weight and bias adjustment to be proportional to the partial derivative of the cost with respect to the weight/bias itself as $\delta_j^l = \partial C/\partial w_j^l$ or $\delta_j^l = \partial C/\partial b_j^l$. Now, vanishing/exploding gradient problem refers to the fact that the gradient in Neural Network is unstable, meaning that it tends to become too big (i.e. explode) or become too small (i.e. vanish) in earlier layers affecting the learning process: the gradient coming from the deeper layers (i.e. the ones closer to the output) have to go through continuous multiplications as in (3.24), and as it approaches the earlier layers, if it is small (< 1), it shrinks exponentially until it vanishes, oppositely if it too large (> 1) it gets larger and larger.

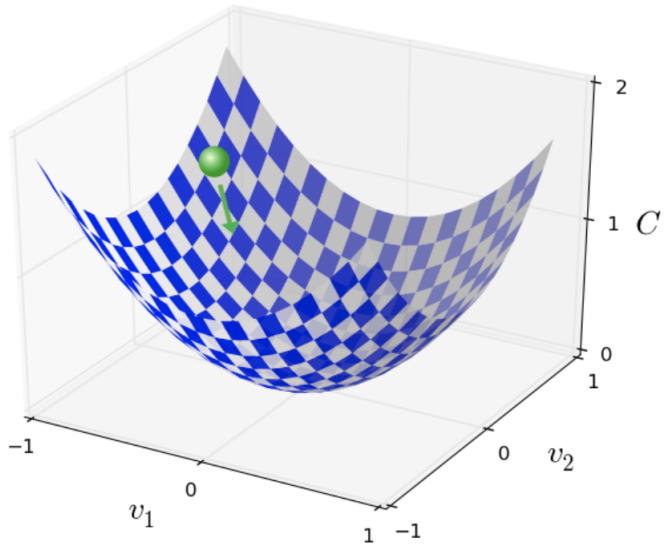

Figure 3.15: Analogy between gradient descent algorithm and the law of motion for a ball that has to move towards the bottom of a valley.

A vanishing gradient would then increase dramatically computation time since the backpropagation algorithm in such situation, at every iteration, would adjust weights and biases of a too little quantity; at the opposite, exploding gradient would cause the algorithm not to reach a local optimum of the cost function since it would adjusts parameters by a too large amount never approaching the local minimum. To make this even more clear, just think at the gradient descent as a method to define the law of motion of a ball; the objective of the gradient descent is to move the ball at the bottom of the valley (i.e. cost function) in which it is found (Figure 3.15). The direction of the next movement of the ball is defined by the gradient δ_j^l and, oversimplifying, the magnitude of its movement by the length $\| \delta_j^l \|$. In a vanishing gradient situation, being $\| \delta_j^l \|$ too small, the ball would require too many movements to reach the bottom of the valley; while, in an exploding gradient problem, where $\| \delta_j^l \|$ is too big, there would be the risk that the ball would continue to oscillate around the minimum never reaching it because of "low sensitivity" in

movements.

As said, the Vanishing and exploding gradient problems is already relevant in Deep NN (i.e. networks with multiple hidden layers). But, it can have an even larger impact in the case of RNN, and this is the reason: let's assume the input sequence of RNN to be a 20-word sentence of the form *"I was born in Italy, ... I speak Italian fluently"*. In the the example, the network, in order to predict the word *"Italian"*, would need information from the word *"Italy"*. The problem is that the term *"Italy"* and *"Italian"* occurs to relatively at the beginning and at the end of the sentence. Unfortunately, when this distance is wide enough, RNNs struggle to learn such dependency (i.e. *Long-term dependencies*) due to vanishing/exploding gradient problems that affect the updating process of W (3.13) matrixes. This problem has been explored deeply by Bengio et al. (1994) and it is solved by LSTM cells.

Long Short-Term Memory

The description of LSTM networks follows Olah (2015). LSTM networks belong to the class of Recurrent Neural Networks. They have been introduced by Hochreiter and Schmidhuber (1997) and have been further refined, e.g. by Gers et al. (2000) and Graves and Schmidhuber (2005). LSTM networks have been specifically designed to learn long-term dependencies and to overcome the vanishing or exploding gradient problem affecting standard RNN: in standard RNN, hidden state activations are influenced by the other local activations closest to them (i.e. Short-term memory), while LSTM networks have been equipped with an activation state that can preserve also information over long distance, hence the name Long Short-Term Memory. Said in other terms, the main difference between LSTM and RNN is that the former has been equipped with cells able to decide whether to keep the existed memory over long time periods, while traditional RNN cells overwrite their content at each time step. In the last few years LSTM networks have been incredibly successful when applied to problems such as speech recognition, translation, image captioning etc.

As explained, all recurrent Neural Networks have the form of a chain of repeating

modules. In standard RNN, the repeating module has a very simple structure, for example a single tanh layer as in Figure 3.16. LSTMs also have this chain structure,

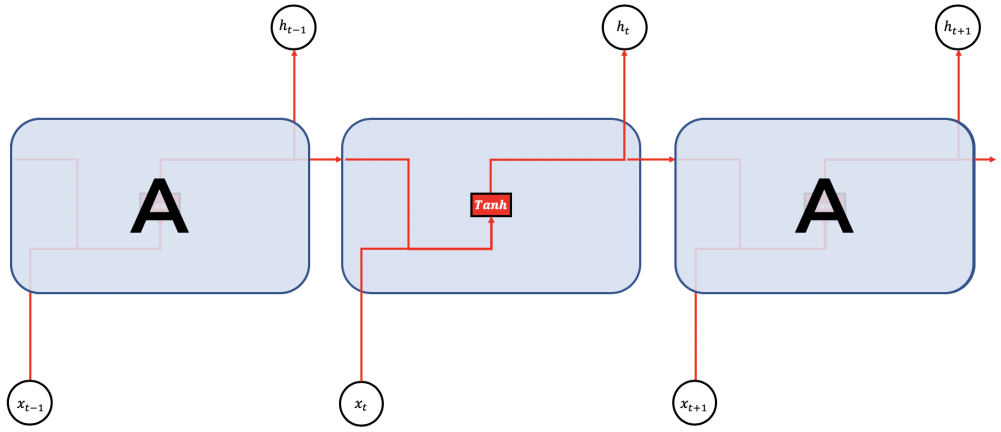

Figure 3.16: Structure of a standard RNN with hyperbolic tangent activation function.

but repeating module, instead of having a single activation function layer, it has four (Figure 3.17). As all Neural Networks, LSTM is composed of an input layer, one

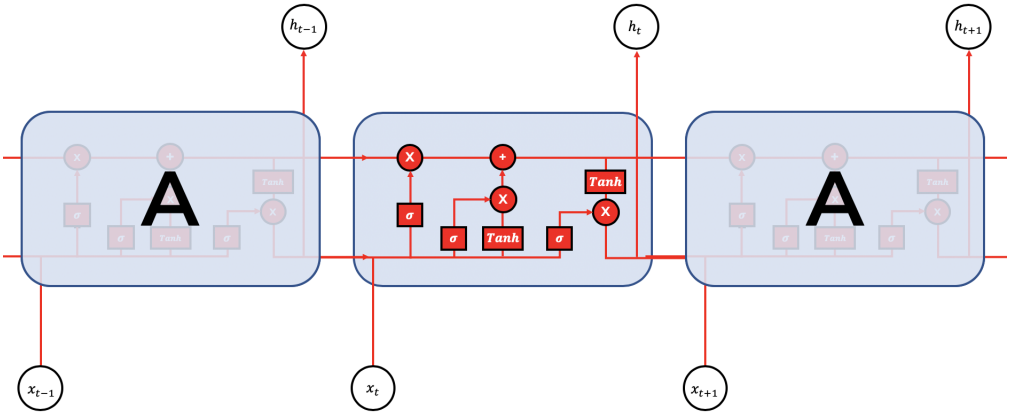

Figure 3.17: Structure of a LSTM network.

or more hidden layers and an output layer. Its peculiarity resides in hidden layers consisting of so-called *memory cells*. These cells are, in fact, the key to understand the suitability of LSTM networks in effectively facing long-term dependencies.

Looking at a single memory cell, the key to the model is the cell state (i.e. horizontal line running through the top of the diagram in Figure 3.17). The memory cell has indeed the ability to either remove or add information to the cell state through structures called *gates*. Gates are composed out of a sigmoid function and a pointwise multiplication operation [3]. Each LSTM cell has three of such gates that uses to control the cell state; at every timestep t, each gate is presented with the input x_t, the output h_{t-1} of the memory cell at the previous time step and the cell state c_{t-1}.

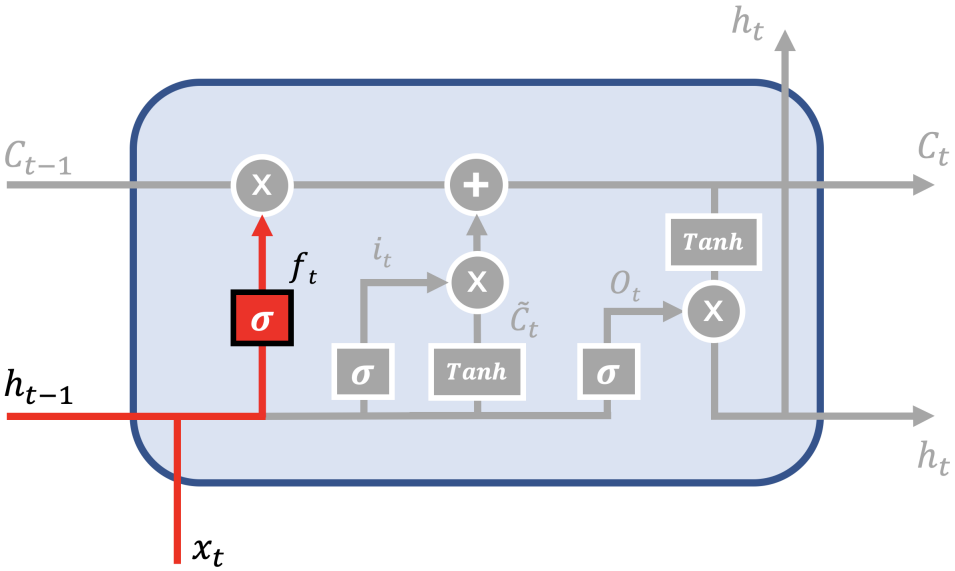

Figure 3.18: Focus on "forget gate" in a LSTM cell.

i. The first operation performed by the cell is to decide what information to remove from the cell state derived from the previous step C_{t-1} (Figure 3.18). This decision is taken by the so called **forget gate**. The forget gate looks at h_{t-1} and x_t and outputs a number between 0 and 1 for each value of the cell state C_{t-1}. A value of 1 represents a "keep this information" while a value of

[3] All red points in figures represent pointwise operations. Pointwise operations refers to operations in which each value in each matrix's cell is considered as a scalar and it is multiplied/summed/divided/subtracted to another cell's value with same index.

0 represents a "get rid of this information". Formally, we have that:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{3.34}$$

where W_f refers to a weight matrix.

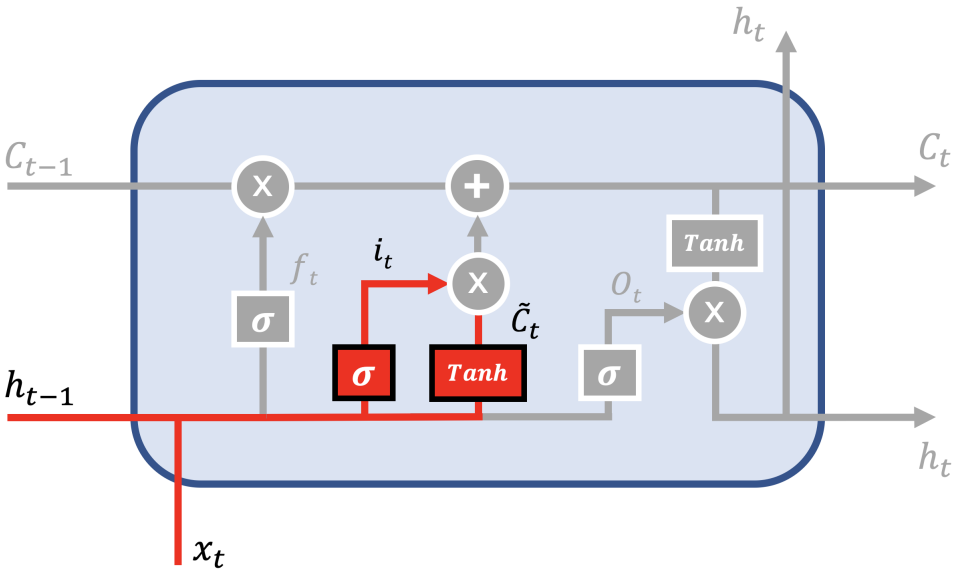

Figure 3.19: Focus on "input gate" in a LSTM cell.

ii. The following operation is related to the **input gate** that decides what new information has to be stored in the cell state (Figure 3.19). The input gate specifically decides which values has to be updated and a concurrent *tanh* layer creates a vector of new candidate values that could be potentially added to the state. The combination of these two thus forms the update to the state cell. Formally:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.35}$$

$$\tilde{c}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{3.36}$$

Then, both f_t and $i_t\tilde{c}_t$ simultaneously update the old cell state C_{t-1}. The update is made by multiplying the old state by f_t, forgetting things decided

to forget earlier, and then adding new information through $i_t\tilde{c}_t$ (Figure 3.20). Formally:

$$C_t = f_t C_{t-1} + i_t \tilde{c}_t \tag{3.37}$$

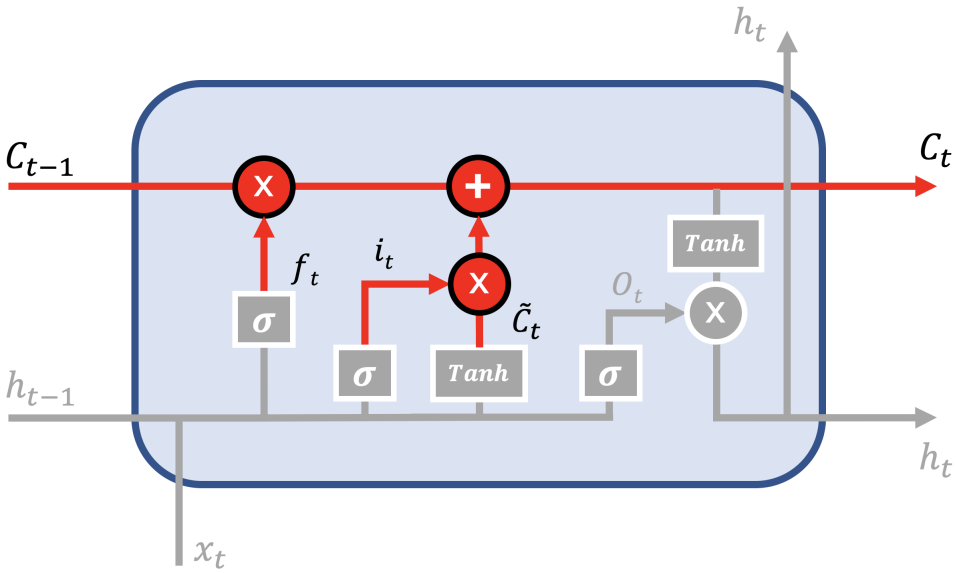

Figure 3.20: Focus on the update of the cell state in a LSTM cell.

iii. Finally, the output of the cell is calculated in **output gate** (Figure 3.21). First, a sigmoid function decides which part of the cell state has to be outputted. The signal exited from the sigmoid is then multiplied by the cell state passed from a *tanh* function. Formally,

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.38}$$

$$h_t = o_t tanh(C_t) \tag{3.39}$$

Taking the example made with RNN regarding the prediction of the word *"Italian"* within the phrase *"I was born in Italy,... I speak Italian fluently"*, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When a new subject is seen, the model wants to forget the gender of the

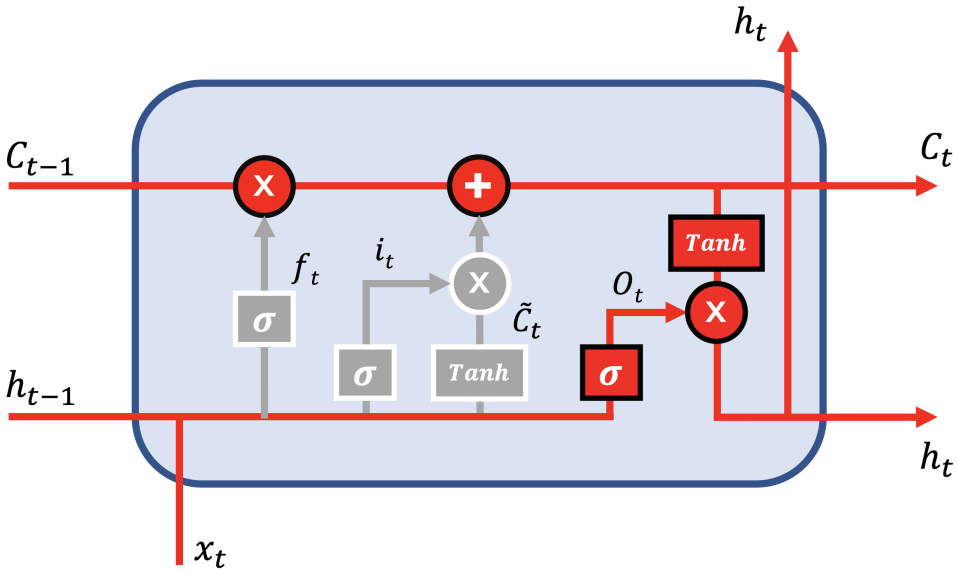

Figure 3.21: Focus on "output gate" in a LSTM cell.

old subject: this is the task of the forget gate, that remove the old gender, and the input gate that actually add the new gender information. Finally, regarding the output gate, it might want to output information relevant to a verb, in case this is what is coming next. For example, whether the subject is singular or plural, so that it is possible to know what form a verb should be conjugated into.

Such structure of memory cells obviously comes with a disadvantage: LSTM networks, compared to all the models presented so far, are the most computationally expensive. Just to have a clue of the the number of weights and biases to be optimized during a training process, they can be calculated as:

$$4ki + 4k + 4k^2 = 4(ki + k + k^2) = 4(k(i + 1) + k^2) \tag{3.40}$$

where k denotes the number of hidden units of the LSTM layer and i the number of input features. $4ki$ refers to the dimension of the four weight matrixes W_o, W_i, W_c, W_f, $4k$ to the dimension of the four bias vector b_o, b_i, b_c, b_f and finally $4k^2$ to the weight matrixes applied to the outputs at previous timesteps.

Finally, it is worth mentioning that the LSTM is not the only successful variant proposed for RNN even if in this thesis it has been extensively treated since it

has been selected to conduct the empirical work presented in Chapters 4 and 5. Among other RNN variants, Gated Recurrent Unit networks (GRU) (Cho et al., 2014) stands out. Greff et al. (2016) offers a nice comparison of most popular RNN variants proposed in computer science literature.

3.3 Feature Selection in Financial Applications

Until now, the mechanisms of the most important models used in financial literature has been presented following their temporal development. However, the selection of the most suitable model is clearly not enough to build a successful ML application. As explained, in fact, ML models require input features to find $\hat{f}$ so as to predict output values: the higher the quality of the information delivered by such features, the higher the performances that the model will be able to deliver. It is thus perceivable why the process to select a subset of the most relevant attributes in the data set for predictive modelling is crucial and it is referred as *feature selection*.

Feature selection becomes even more important in financial applications since, when dealing with them, the relations between input X and output Y, in our case stock price, do not appear always stable in time and in some case neither intuitive. In other fields, feature selection process is found to be more straightforward: take as example a model for the prediction of heart disease in patients that presents a chest pain. Features to be used appear at least understandable even for non-experts; they will certainly be the age and sex of the patients, a cholesterol measurement and probably other heart and lung function measurement. Moreover, with high probability, the relationship between the disease and these features, within a certain range of variability, will remain stable across time; there is no reason to think that a person with specific characteristic and chest pain is diagnosed with an heart disease today and he will not in a month.

Oppositely, since financial markets are driven by investors' behavior, their irrationality and temporary cognitive biases can create strong links between specific

variables and securities' prices only in particular market phase or only after particular information availability. Just think at the so-called *herd instinct bias* defining a situation in which people act like everyone else without considering the reason why. In finance it is referred as the phenomenon in which investors follow what they perceive other investors are doing, rather than their own analysis. An investor exhibiting herd instinct will decide to make the same or similar investments based almost exclusively on the perception that many other investors are doing the same with that precise market conditions (e.g. Lux, 1995; Bikhchandani and Sharma, 2001). Herd instinct has a history of starting large, unfounded market rallies and sell-offs often based on a lack of fundamental support to justify either; for instance, it has been a significant driver of the dotcom bubble of the late 1990s and early 2000s.

It is clear then how, in the moment in which agents are biased, they manages to change standard market relationships for a given time-frame. And this is the actual reason why financial market predictions requires even more focus on feature selection process than many other applications. To summarize, variability and contextuality of features can be considered among the main reasons of the challenges related to feature selection in financial time-series prediction.

Due to all these reasons, the synthetic taxonomy of the most frequently employed predictors in financial applications presented in this section aims then at defining a reference context so as to facilitate new applicants when designing their input feature set and, at the same time, also to stimulate the search for new possible solutions outside it. The following subsections will tackle indeed, without the intention to be exhaustive, the six main clusters of features identified to be typically employed in financial market predictions within academic research: *price and volume variables*, *categorical data*, *technical indicators*, *macroeconomic and commodities variables*, *risk variables* and *sentiment variables*.

3.3.1 Price and Volume related Variables

Evidently, the most intuitive variables to be used in financial market predictions are market price data. Usually they are employed with a daily frequency and they come in this form: *close, high, low and open price*). These prices refer respectively to the price of the security at market closure time, the highest daily price, the lowest daily price and the price at market opening. Then, another measure that usually is related with them is *volume* that refers to the amount of transactions for a specific security occurred in a specific time-frame.

However, carefulness is required when dealing with these type of data: comparing historical stock prices or volumes to those of the present day does not accurately reflect performance indeed. In fact, stock price is mainly affected by supply and demand of market participants but sometimes it changes due to some corporate actions, such as stock splits, dividends or rights offerings. In all these cases, to be comparable, prices in different time periods need to be adjusted.

Some actions (e.g. stock splits or right offering) change the number of outstanding shares in the market, thus affecting the value of the single stock and keeping unchanged the value of the company. Think about a company with 100 outstanding shares whose value is 50$ each that decides to split them by a factor of 2. After the split the company will have $100 \times 2 = 200$ outstanding shares and, remaining the value of the company the same $100 \times 50\$ = 5000\$$, the new value of each share will be $5000\$/200 = 25\$$. Clearly this change in price is not related to a decreased perception of the fundamental value of the company by investors but just to a technical decision. Thus an accurate analysis has to take into consideration an adjusting factor to properly compare price observations before and after the corporate action. Similarly, some others actions like dividends cyclically affect share price in proximity of the day in which they are distributed since investors start buying the stock knowing they will gain an earning soon. Also in this case an appropriate adjustment factor is required.

Once these type of data are adjusted, then it is possible to compare market observations in different time periods. Moreover, the adjustment makes possible the calculation of *lagged returns* that, without any doubt, are the most used types of input features in financial time-series analysis. They are defined as:

$$R_t^{m,s} = \frac{P_t^s}{P_{t-m}^s} - 1 \tag{3.41}$$

where s refers to the single stock and m to the time lag.

3.3.2 Categorical Data

Categorical data are represented in ML models through dummy variables that are binary variables that activate (i.e. take the value 1) if a certain condition occurs and remain inactive (i.e. take value 0) otherwise. Categorical variables are mainly used as input features in financial time series applications to identify peculiar characteristics referring to stock specific behavior or patterns that occur only within a subset of stocks, for example within a specific industry. Think for instance at the already cited .com bubble, the anomaly in that case was confined to the tech industry.

In the case of an introduction of dummy variables for the identification of stock specific behavior, it is required to add a new variable for every different stock within the dataset so as to linearly increase the amount of predictors, proportionally to their amount within the dataset. This is the reason why this choice sometimes can result to be excessively onerous in terms of computation time.

The identification of industry specific behavior is performed with the same logic even if it requires the introduction of way less amount of variables within the dataset. Moreover, regarding the categorization of stocks into different industries, many possibilities exist: the most frequently employed refer to classification benchmarks provided by third parties such as the *Industry Classification Benchmark* (ICB) launched by Dow Jones in 2005 that divides the market into 10 sectors within the macroeconomy or the *Global Industry Classification Standard* (GICS) that consists of 11 industry sectors.

Sometimes dummy variables are also employed to represent days of the week or months. This decision has to be accounted to the intent to spot some particular behavior of the market in specific period of the week or the year caused by particular investors' biases. For instance, DellaVigna and Pollet (2009) use a Friday dummy based on the argument that investors are distracted by the upcoming weekend in their study dealing with post earnings announcement drift (i.e. a particular phenomenon caused by behavioral irrationality of investors).

In any case, there could be infinite possible categorical variables to be used in stock-price prediction and this section do not want to be exhaustive in this sense. Anyhow, it is worth mentioning, how, from a statistical point of view, in a traditional time series framework the introduction of dummy variables would open several issues (e.g. dummy variable trap, multicollinearity, sparse predictors etc.). Such issues are not found as much as risky in the case of ML models, since ML models have demonstrated to be able to deal with such issues efficiently on their own. Rather, the main problem that this type of variables create in ML applications, regards the relation between the number of observation in the dataset and the number of predictors: in case the number of predictors exceeded the number of observations indeed, there would be difficulties during the training procedure. Even so, this is not usually the case with financial time-series.

In conclusion, the focus when taking a decision about the introduction of dummy variables/categorical variables within the set of predictors must be on the trade-off between the increase in prediction accuracy and a proportional increase in computational time due to increasing number of model's parameters.

3.3.3 Technical Indicators

As anticipated in 1.2, *"technical analysis is the study of market action, primarily through the use of charts, for the purpose of forecasting future price trends"* (Murphy, 1999), where the term *market actions* (or "price actions") refers to the two principal sources of information available to technician that are price and volume.

Technical analysis, also known as *charting*, has been a part of financial practice for many decades, but it has not received the same level of academic acceptance as more traditional approaches such as fundamental analysis. The main reason for that is certainly related to the highly subjective nature of the approach. The technical type of analysis bases its effort on three main premises:

1. It is founded on a study of price actions. Price and volume indeed reflect shifts in supply and demand and, as basis of all economic forecasting, if demand exceeds supply, prices should rise; on the opposite, prices should fall. Technicians use this simple statement to arrive at the conclusion that if prices are rising, demand must exceed the supply and fundamental must be bullish while if prices fall, fundamentals must be bearish. The only particularity is that the term fundamentals in this case regards all that factors can affect the price, including psychological factors clearly not taken into consideration by fundamental analysis. In any case, technicians are not interested in understanding the reasons why prices rises or fall; it is not useful since they believe that everything that can affect market price is ultimately reflected in market price. In some way, even if it could sound contradictory, they believe market to be efficient discounting all the fundamentals; but, differently from fundamental analysis, technical analysts does not believe that knowing all the reasons why market can go up or down is necessary for the forecasting process.

2. The concept of "trend" is essential to the technical approach. The whole purpose of charting (i.e. display on graphs all the information relative to) price and volume information of a market is to identify trends in early stages of their development for the purpose of trading in the direction of those trends until they reverse.

3. The last premise is about the value of time: for technical analysis if patterns have worked well in the past, it is assumed that they will continue to work well in the future. Technical analysis is based on the study of human psychology

that ultimately tends not to change. The consequence of such concept, in contrast with EMH, is that the key to understand the future lies in studying the past.

After these premises the main difference between fundamental analysis and technical analysis appears more clear: while technical analysis focuses on the study of market actions, the fundamental approach examines all relevant factors affecting price to determine the intrinsic value of a market; finally, depending on the current intrinsic value, the market is valued overpriced or oversold so as to be be respectively sold or bought. Fundamentalists have always to know the "why", while technician concentrates only on the effects. Moreover, technician states that the problem of the fundamental approach is that, at the beginning of important market moves, it does not explain what the market seems to be doing. This is caused by the fact that when the known fundamentals have already been discounted by the market, the market reacts to other unknown fundamentals.

Now, based on these premises, technical analysis is practically founded on the study of indicators commonly referred as *technical indicators*. Technical indicators refers to the tools used by technical analysts to analyze historical data and to spot trend in the market. They are mathematically based tools and they are built using price and volume information. They can be used either as standalone instruments or in combination either with each other or with other chart information as support for traders' decision-making process. The main criticism that is moved against technical indicators, and technical analysis in general as anticipated, is that their interpretation is highly subjective and, even if in specific situation many technicians may have the same interpretation about market movements, each investor performing technical analysis could realize a different strategy. Each technician indeed creates his own strategy, that is a definitive set of rules that specifies the exact conditions under which trades are established, managed and closed based on the signals provided by technical indicators. The reason of this diversity resides in the fact that the set of rules by technicians is built completely heuristically: each investor, based

on his experience, finds a personal way to objectively determine entry/exit points and other management rules based on his interpretation of information provided by technical indicator.

Summarizing, technical analysis deals with subjective interpretations of objective parameters (i.e. price/volume charts along with technical indicators): technical indicators do not create trading signals on their own since they require specific rules that in turn depend on the experience, the trading style and the risk tolerance of each investor; these rules, when gathered, form the personal investor's trading strategy. One famous anecdote can explain how, however, once the experience has helped a trader to create his own effective rules, the application of such rules becomes really easy to implement: Covel (2009) in his book "The complete turtle trader" describes how in 1983, the famous commodity trader Richard Dennis conducted an experiment to prove he could teach people to become great traders. According to the experiment, the so-called turtles (i.e. the trainees), with a real account to trade and applying a precise strategy, succeeded to earn an aggregate sum of over 100$ million dollars in the following four years.

Among technical indicators, a growing number of them are publicly available for traders; anyhow, many traders also develop their own unique indicators. Indicators can be classified in different ways. As first possibility, they can be clustered according to the speed at with which they react to events in the market. Following this criterion, we can find:

- **Leading indicators -** are used to predict future events, usually they show a signal of a certain movement before the market actually does so. As drawback sometimes they could give false signals. They are normally combined with other leading indicators to increase the speed of signal detection.

- **Coincident indicators -** generate signals at the same time price action has started to move in the direction shown by the indicator.

- **Lagging indicators -** are observable measures that moves or changes direc-

tion after a change is occurred in a target variable of interest. They are useful as confirmation of trends and changes in trends.

The second way in which technical indicators can be clustered is according to the function they exert on charts. Following this criterion we can find find:

- **Trend indicators -** measure the direction and strength of a trend smoothing out a certain measure over a certain period of time. Inside the category of trend indicators, it is worth mentioning *moving averages* since they are considered among the most versatile and widely used technical indicators. A moving average is an average of a certain body of data, where the term "moving" refers to the fact that it is calculated using a fixed amount of data that moves forward with each new trading day. As example, a 10-days moving average is calculated using the total of the last 10 day's closing price/volume/etc. The property of the indicator is to smooth out observations by filtering out the noise related to random short-term fluctuations.

- **Momentum indicators -** momentum measures the velocity of price changes as opposed to the actual price levels themselves. Market momentum is measured by continually taking price differences for a fixed time interval. Momentum indicators thus help identify the speed of price movement by momentum over time. Graphically, they appear as a line below the price chart that oscillates as momentum changes; a divergence between price and momentum indicator usually can signal a change in future prices.

- **Volatility indicators -** measure the rate of price movement, regardless of the direction. They provide useful information about the range of buying and selling points that take place in a given market and help traders determine situations where the market may change direction.

- **Volume indicators -** measure the strength of a trend or confirm a trading direction based on some form of averaging or smoothing of transaction

volumes. They rely on the assumption that strongest trends usually appear while volumes are increasing.

In the following table some of the most frequently used technical indicator will be presented along with their formula and description:

Table 3.1: Table representing most used technical indicators in ML applications along with their formula and description. In formulas H_t, L_t, O_t, P_t refers respectively to high, low, open and close price at time t.

Trend indicators

Indicator	Formula	Description
Simple Moving Average	$SMA_t = \frac{1}{k}\sum_{i=-k+1}^{0} x_{t+i}$	Simple average of x_t over a defined number of periods k.
Exponential Moving average	$EMA_t = x_t \times k + EMA_{t-1} \times (1-k)$	Weighted MA giving more importance to recent data: weights decrease exponentially as observations become more distant in the past.
Moving Average Convergence Divergence	$MACD = EMA_{fast} - EMA_{slow}$	MACD is compared with a EMA of the MACD itself (called "signal line"). The most important signal is generated by the intersection of the two lines: traders may buy when MACD crosses above the signal line and sell when the opposite happens.
Parabolic Stop and Reverse	$SAR_t = SAR_{t-1} + \alpha(EP - SAR_{t-1})$	EP (extreme point) is a record that represent the highest (lowest) value reached by the price during a positive (negative) trend. α represents the acceleration factor, usually initialised at 0.01 for stocks. It is increased by 0.02 each time a new EP is recorded, up to a max value (usually 0.2). On a chart, the indicator appears as a series of dots, if the dot is below the price it is considered a bullish signal and vice versa.

Momentum indicators

Indicator	Formula	Description
Relative Strength Index	$U_t = close_t - close_{t-1}$ $D_t = close_{t-1} - close_t$ $RS_k = mean_k U_t / mean_k D_t$ $RSI = 100 - 100/(1 + RS)$	RSI measures evaluates overbought or oversold conditions of a security returning a value that between 0 and 100. Usually, a value over 70 signals an overvalued condition while a value below 30 indicates an undervalued condition. K indicates the time periods where U (i.e. gain) or D (i.e. losses) are calculated.
Rate of Change	$ROC = (\frac{P_t - P_{t-k}}{P_{t-k}}) \times 100$	It measures the % change in price in a given period of time. It is plotted against zero: a negative value indicates a negative momentum (i.e. decreasing price) and vice versa. The indicator can signal overbought/oversold levels when compared to past extreme values it reached before the price reversed.
William's %R	$\%R = (max_{[t-k;t]}(H_i) - P_t)/(max_{[t-k;t]}(H_i) - min_{[t-k;t]}(L_i))$	The indicator oscillates between 0 and 100 and measures overbought/oversold levels. It indicates where the current price is, relative to the largest range occurred in a past time frame. A value under 20 signals the security to be overbought, while a value over 80 signals an oversold situation.

Stochastic %K	$\%K = \frac{P_t - min_{[t-k;t]}(L_i)}{max_{[t-k;t]}(H_i) - min_{[t-k;t]}(L_i)} \times 100$	The theory behind the indicator states that in a market trending upward, prices will close near the high and vice versa. %K is thus used to predict price turning points by comparing the closing price of a security to its price range. A transaction signal is usually generated when the %K line crosses through the %D one.
Stochastic %D	$fast\%D = SMA_k(\%K)$ $slow\%D = SMA_k(fast\%D)$	%D is uses jointly with %K: the buy signal occurs when the %K line rises above %D and the sell one when %K falls below %D.
Commodity Channel Index	$TP = (H + L + C)/3$ $MeanDev = \sum_{i=t-k}^{t} \frac{\lvert SMATP - TP_i \rvert}{(k+1)}$ $CCI = \frac{TP - SMA_k(TP)}{0.015 \times MeanDev}$	CCI normally should suggests to sell when its value is inferior to -100 and to buy when it is larger than 100; positions should be closed when the value enters within the neutral range [-100, 100]. The interval $[t-k, t]$ has to be set at 1/3 of the length of the past observed cycles.

Volatility indicators

Indicator	Formula	Description
Average True Range	$TR = max[(H - L), \lvert H - P \rvert, \lvert L - P \rvert]$ $ATR = SMA_k(TR_i)$	ATR measures market volatility by decomposing the entire range of an asset price for a specific period. The indicator is usually used as exit method on whatever entry decision.

Bollinger Bands	$TP = (H+L+P)/3$ $Band_{up} = SMA_{t-k,t)}(TP) + m \times \sigma_{(t-k,t)}(TP)$ $Band_{low} = SMA_{t-k,t)}(TP) - m \times \sigma_{(t-k,t)}(TP)$	Bollinger bands consist in two lines plotted m standard deviation away from a SMA. Many traders believe the closer the price move to the upper band, the more overbought the market, and the closer the prices move to the lower band, the more oversold the market. The *squeeze* is another important signal defining a moment in which bands come closer together: it signals low volatility and it is considered a potential sign of future increased volatility and possible opportunities.

Volume indicators

Indicator	Formula	Description
William's A/D	$CMFV = \frac{(P-L)-(H-P)}{H-L} \times Volume$ $A/D_t = A/D_{t-1} + CMFV$	It measures whether a stock is being accumulated or distributed using price and volume info. It aims at finding divergences between the price and the volume flow. As example, if price is rising but the indicator is falling, this indicates how buying (accumulation) volume might not be enough to support the price, so a price decline is expected.
On Balance Volume	$OBV_t = OBVt - 1 + \Delta_t$ $\Delta_t = \begin{cases} Volume & \text{if } close_t > close_{t-1} \\ 0 & \text{if } close_t = close_{t-1} \\ -Volume & \text{if } close_t < close_{t-1} \end{cases}$	The rationale behind the indicator is that when volume increases without a significant change in price, the price will eventually jump upward or downward. Traders are interested in the nature of movements of OBV over time.

Chaikin's Oscillator	$CO_t = EMA_k(A/D) - EMA_{k'}(A/D)$	$EMA_{k'}$ refers to an average over a longer period compared to EMA_k. The indicator oscillates around the zero line. Generally, buying pressure is stronger when the indicator is positive and vice versa.
Volume Rate of Change	$VROC = \frac{Vol_t - Vol_{t-k}}{Vol_{t-k}} \times 100$	The indicator is used with the supposition that almost every significant pattern is accompanied by a sharp increase in volume, and VROC shows exactly the speed at which volume is changing.

In recent years, with the advent of ML models for financial time series prediction, technical indicators have been extensively employed as input features in different models. The expectation behind their use is to let the ML models to find dependencies between technical indicators and behavior of the time-series and trend formation by themselves; the rationale is to use objective instruments that in specific applications have demonstrated to be successful, letting their subjective and profitable interpretation to be found by the algorithm. As analogy, it is similar to think at the ML model as an inexperienced trader that creates his own experience about how to read the evolution of specific technical measures in relation to securities prices by a trial and error approach. Models, in such a view, are thus employed to relate specific behavior of technical measures to the path of specific security's prices without being explicitly trained to do it.

3.3.4 Macroeconomic and Commodities Variables

Stock prices are strongly influenced by the environment in which companies operate; in this sense it is worth to recall from (1.1) how the value of a company depends by all the variables that can affect both future earnings of the company itself and

the discounted rate. Under this rationale, factors such as macroeconomic, risk variables or variables related to commodities are sometimes implemented as ML input features.

Macroeconomic Variables

The macroeconomic environment, in which both investors and companies operate, strongly influences stocks' prices; macroeconomic variables influence both returns of the company, thus expected cash flows, and the discount factor k (Equation (1.1)). Regarding the discount factor, it depends on the *risk premium* that is the excess return that investors expect from the stock market over a risk-free rate. So, for instance, unanticipated changes in the risk-free interest rate influence k that in turn influence stock prices. Moreover, uncertainties on feature macroeconomic scenarios, like the ones caused by the trade war between US and China or by the actual Covid-19 pandemic, can have an influence on the risk perceived by investors affecting the risk premium and indeed the expected returns. Moreover, even the expected rate of inflation, a typical macroeconomic measure, influence nominal rates of interest and nominal expected cash flow. Regarding expected cash flows instead, think for example at the real consumption changes that gives information about changes in indirect marginal utility of real wealth of consumers that in turn can influence first revenues and then earnings of companies. Chen et al. (1986), testing whether innovations in macroeconomic variables are rewarded risks in the stock market, state that *"stock returns are exposed to systematic economic news"*.

Regarding the use of macroeconomic variables in ML financial literature, for sure, the two most frequently used macroeconomic variables are interest rates and exchange rates:

- **Interest rates -** The level of interest rates in different markets is typically reflected in *government bond rates* at different maturities. There are several view that describe how interest rates can affect stock prices. One view asserts that, since stock prices are determined by expected dividends and interest

rates, any surprises in monetary policy are likely to influence stock prices directly via the interest-rate channel or, indirectly, through changes in the determinants of dividends. Another view suggests that an expansionary monetary policy, by raising asset prices, lowers their expected returns and thus depresses the stock market. This occurs because rising equity prices are considered a possible signal of future inflation, which would trigger subsequent counter action. Unfortunately, there is not a single and consistent framework that describes the nature of the interaction(s) between monetary policy and the stock market.

- **Exchange rates -** Hau and Rey (2006) suggest that foreign exchange and equity markets should be negatively correlated because of portfolio rebalancing. To understand why, consider an European portfolio manager with money invested in Europe. When the European stock market rises relative to the US, the manager is overweight with European equities and, to return to a neutral position, sells European positions and then sells the euro proceeds for US dollars. The sale of euro for dollars causes the euro to depreciate at the same time that the European stock market is outperforming. Moreover, many studies seem to state evidence of statistically significant exchange rate impact on stock index returns (e.g. Zarei et al., 2019).

However, they are not the only types of variable employed in financial applications. Even if less frequently, other variables are seldom employed in financial research: Huang et al. (2016), addressing the response of US stock market to fluctuations in oil prices, exchanges rates and US real interest rates, report *"higher correlation coefficients and magnified impulse responses and variance decomposition when real interest rates become negative in early 2009. U.S. stock markets respond, in the more recent period, positively to increases in oil price shocks (expectations of world economic recovery), negatively to an appreciation of the USD against major currencies (trade falls and thus GDP growth, reducing company earnings), and also*

negatively to real interest rates (according to the present value model)" (Huang et al., 2016). Such findings introduce to the second type of variables sometimes employed in ML applications that are commodities variables.

Commodities variables

Commodity and stock returns are usually not intended to have strict and direct relationships. Stock price is determined by discounted future cash flows while commodities are more determined by short-term demand and supply shocks. In any case, in the long run both stock and commodity prices are expected to move in a similar direction as both are linked to future economic performance. Black et al. (2014) argue in favour of a long-run relationship between stock prices and a general commodity price index. If any, however, the relationships between commodities and stocks returns should be negative, in fact, in portfolio setting, commodities are often seen as a preferred investment when equity performance is weak.

Among commodities, the most frequently employed as predictors are certainly *gold* and *oil*. About gold, the usual perception is that it is a safe-haven asset asset and that investors demand it as a hedge against macroeconomic shocks. However, from 2000 to 2013, it can be noted how such hypothesis in not confirmed by a significant drop in the price of gold. Caliskan and Najand (2016) suggest instead how gold is used by short-term investors as a temporary asset during stock market fluctuations, therefore driving the demand for gold even when the market is on the rise. Such theory states that *"short-term contrarian investors sell the winning portfolios (that is, sell high) and herd to gold; conversely, when they find losing portfolios, they liquidate their gold position to buy these portfolios (that is, buy low)"* (Caliskan and Najand, 2016). Authors then find evidence of a correlation between stock market and the price of gold: when the market generate higher negative returns the price of gold decreases.

Regarding oil, the effects of oil shocks on stock markets are likely to vary considerably across different countries depending on their production and consumption

of oil reserves. For instance, with the same conditions, net exporters (importers) of oil are likely to benefit (suffer) from price hikes. Many studies have been conducted on the relationship between stock market and oil price: Jones and Kaul (1996) investigating about the reaction of the U.S. stock market to oil shocks find *"that stock prices rationally reflect the impact of news on current and future real cash flows"* (Jones and Kaul, 1996) but they also find *"no evidence of fads and/or market overreaction"* (Jones and Kaul, 1996); Kilian and Park (2009) find that the response of aggregate stock returns may differ depending on the cause of the oil price shock. *"The negative response of stock prices to oil price shocks is found only when the price of oil rises due to an oil-market specific demand shock such as an increase in precautionary demand driven by concerns about future crude oil supply shortfalls. In contrast, crude oil production disruptions have no significant effect on cumulative stock returns. Finally, higher oil prices driven by an unanticipated global economic expansion have persistent positive effects on cumulative stock returns within the first year of the expansionary shock"* (Kilian and Park, 2009).

3.3.5 Risk Variables

Equity risk refers to the financial risk involved holding equity in companies through the purchase of stocks. The perception of such risk by investors clearly affects stocks price through the above mentioned risk premium.

The typical measure of risk used in finance is the standard deviation of a security's price over a predetermined number of periods. The issue when using standard deviation is that it is calculated on past data while stock price is related to future expectations. In fact, the effective risk affecting stock price depends on expectations of investors in the market more than past level of volatility by a specific stock or by the whole market.

This is the reason why, especially for prediction application, the research about leading risk indicators is crucial. Commonly, as most famous approximation of the expectations about risk perceived by investors in financial markets the *volatitlity*

index (VIX) is employed. VIX is a measure of implied volatility (i.e. expected volatility of underlying asset implicitly assumed in pricing options) provided by the Chicago board options exchange (CBOE), of a wide range of options based on the S&P500 index. It is also probably the most employed as predictor in ML application to model the risk perceived by the market in different periods of time. VIX is founded on the assumption that, conditional on the option-pricing model, implied volatility derived from option prices should represent the market's best prediction of the underlying asset's future volatility accounting for all relevant information. Different researches have demonstrated the positive correlation between VIX index and future market volatility Becker et al. (e.g. 2006); VIX is known as *"investor fear gauge"* (Whaley, 2000) since high levels of VIX coincided with high degrees of market turmoil in the past.

Summarizing, VIX reflects investors' best prediction for the next 30 calendar days about market volatility: theoretically a high VIX should reflect increased investor fear, while a low VIX investors' comfort. During period of market turmoil, the VIX increases reflecting the panic demand for put options as hedge against declines in stock portfolios; while during bullish period, the lower fear is reflected in less need to hedge risks. VIX is also supposed to be a leading indicator for stocks' returns since it is expected to signal future market movements: it has been observed how high levels of VIX often coincide with market bottoms and seems to signal "oversold" markets. Under this light, traders could take long positions in the market anticipating an increase after VIX's peak. For instance, Banerjee et al. (2007) find that *"VIX variables significantly affect returns for most portfolios, with the relationship stronger for high beta portfolios"* (Banerjee et al., 2007) after studying the relationship between future returns and current implied volatility and after having examined portfolios sorted on book-to-market equity, size and beta.

Another frequent indicator used in financial market, directly related to credit risk, is the *Ted spread*. Ted spread measures the difference between the interest

rates on interbank loans and on short-term US government debt:

$$\text{Ted} = \text{3-month LIBOR rate} - \text{3-month T-bill interest rate} \tag{3.42}$$

This indicator signals perceived credit risk (i.e. risk of default of the counterpart) in the general economy. It is calculated, indeed, as the difference between risk free rate (T-bill) and LIBOR that reflects the credit risk of lending to commercial banks.

The linkage between Ted-spread and financial markets resides in liquidity problems caused by an increased perceived credit risk in the market. Liquidity in financial market refers to the possibility for an investor to quickly execute a trade at a price near the fundamental value and it clearly influences the price of stock in the market. However, agents appointed to provide liquidity such as market makers and other traders, to do so, could need to raise capital in primary market against the use of collaterals. So the market value of assets used as collateral, along with the perceived credit risk, plays an important role in funding liquidity (i.e. the willingness of financiers to provide loans). So, an increased credit risk could decrease market liquidity that in turn decreases the value of collaterals in the market decreasing funding liquidity and so on. Now, the Ted spread is a widely employed measure of funding liquidity. Boudt et al. (2017) find how *"the dynamic model linking market liquidity to funding liquidity changes when the TED spread surpasses a 48 bp threshold, whereby the impact of market liquidity on funding liquidity becomes significantly less stabilizing than in the regime with TED spreads below 48 bp"* (Boudt et al., 2017) and they conclude saying that *"the Ted spread should be considered an informative market barometer for liquidity regimes in equity markets"* (Boudt et al., 2017).

3.3.6 Sentiment Variables

Investors' sentiment in finance refers to *"investors' attitude or feeling toward a particular security, which tends to be revealed through an event (such as an earnings announcement) or price movement of the security traded in the market"* (Kim and Kim, 2014). Following rational risk-based asset pricing models, there would not

be a significant impact of investors' sentiment on asset pricing: prices reflect the discounted value of expected future cash flows and, even if some investors were not rational, their irrationality would be quickly offset by arbitrageurs. In contrast, behavioural finance believe that investors' sentiment, as in retail demand, may cause prices to deviate from their underlying fundamental value. Precisely, when sentiment rises, *noise investors* (i.e. impulsive and non completely rational traders) increase their investment allocations to risky assets, and this sentiment-driven demand for assets drives prices above the fundamental values of these assets. After periods of high sentiment, prices revert to the fundamental values. This mechanism implies that investors' sentiment is negatively related to subsequent stock returns.

To assess whether investor-generated content can help predict stock returns, based on the hypothesis just cited, specific procedures to convert unstructured qualitative information into structured quantitative sentiment variables have been created. Sentiment variables thus refer to variables that is possible to use as input in ML models as proxies about investors' sentiments to make guesses about future market movements. In financial literature, the majority of approaches deals with textual variables (e.g. news and social media and Internet message boards) as to extract sentiment information. Renault (2017) distinguishes two ways that can be followed for *Textual sentiment analysis*: *dictionary-based classification* and *machine learning classification*. Dictionary-based classification consists in computing a sentiment variable by counting the number of positive words and the number of negative words in a document using a predefined list of signed words (i.e. dictionaries). For example, imagine *"nice"* and *"smart"* to be considered as positive words and *"ugly"* and *"unpleasant"* as negative. Then, each time in a sentence is found a positive (negative) term a +1 (-1) score is assigned. So, the sentence *"Francesca is a beautiful and smart lady, but sometimes she is really unpleasant"* would receive a score of $1 + 1 - 1 = +1$. There are three main possibilities to create such *lexicons*:

- Pure expert view in which it is created from scratch a list of positive and neg-

ative words, based on the knowledge and expertise of the application domain.

- Two-step procedure in which a list of words can be generated by analyzing a set of unclassified documents and then each word is manually classified by an expert.
- Two-step procedure where from a set of pre-classified document (positive and negative documents) a list of words is extracted and then it is measured each term's frequency in each class of document. Words are finally categorised based on specific frequency thresholds.

Such an approach is clearly easy to implement, but it suffers from the fact that it is necessary to develop field-specific dictionaries for each domain of research since a word can have different meaning based on the context. Moreover, these approaches use a equal weighting-scheme where each word is supposed the same explanatory power. One example of a dictionary specifically created for financial applications is the one presented in Loughran and McDonald (2011) that contains different lists of Constraining, Litigious, Negative, Positive, Uncertainty, Superfluous and interesting words. Picasso et al. (2019) is an example of application that use such lists to extract sentimental features for its ML model. The second method, machine learning classification, consists instead in training a ML model with pre-classified documents and in automatically classifying words into different classes. The advantage of this second method is that ML techniques, as opposed to dictionaries, can also provide answers to the weighting procedure and to the non-independence of words in a sentence. However, this approach requires a sufficiently large training set of manually labelled document that is not always easy to create and its results significantly depends on the ML algorithm used and on its tuning process.

Once a method for the extraction of sentimental variables have been selected, it is necessary to select the types of documents used to extract sentimental variables. Many solutions can be applied: for instance, Picasso et al. (2019) uses news on specific stocks published from selected newspapers, Renault (2017) analyses posts

taken from Stock-Twits, a social network similar to twitter but specifically created for traders where they can post messages.

3.4 Model accuracy and Trading Performances

The evaluation of a data science model might be difficult when applied to the financial domain, especially in classification settings: an accurate forecasted direction for stock market cannot be directly linked to a positive trading performance since in terms of financial results, periods with high volatility are more important than period with low volatility. Even if ML models present high accuracy indeed, portfolio return can be low. Therefore, in ML financial researches, in addition to the validation of standard data science metrics, the validation of the pertinence of the model by simulating effective investment strategies results fundamental.

This is the reason why for ML financial researches a proper validation phase is always divided into two different steps: the first step has to take into account metrics more related to the machine learning domain so as to understand the statistical behavior of the model; however, since promising results in terms of ML metrics are not directly associated to positive financial performances, in a second step, it is necessary to define a trading strategy and to evaluate the model looking at the performance of such a strategy built on model predictions. Assessment measures for this second step clearly regards the traditional measures adopted for portfolio performances. During the first phase, the power of the proposed classifier is proved; during the second one, the real effectiveness of predictions is demonstrated.

In academic literature, as highlighted in Chapter 2, among the studies that employ machine learning techniques to forecast stock price movement, only a few exhibit empirical results of a trading performance obtained through forecasts from models. Such studies highlight a flaw in their consistence: they do not prove the real advantage, from an operational point of view, deriving from the employment of computational expensive model for stock market predictions.

Starting from these assumptions, in the next two sections, the most frequently employed measures to asses the performances both from a data science perspective (Section 3.4.1) and from a business perspective (Section 3.4.2) will be presented. Regarding the data science perspective, it will focus on suitable measures for a classification problems since the empirical work presented in Chapter 4 deals with one of it. The review in these following sections does not aspire to be exhaustive, but rather it aims to sketch a proper methodological framework to validate ML application for financial market predictions from an academic research perspective.

3.4.1 Machine Learning Performance Metrics (classification)

Accuracy The first and most intuitive evaluation metrics for a classification problem is accuracy:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \tag{3.43}$$

It can provide useful information of model's capabilities in case the number of samples in each class is equally distributed (i.e. when the problem is said to be a balanced classification problem). If this is not the case, classification accuracy can give false sense of having achieved great results. Imagine a case in which in our database we have 96% of samples belonging to class 1 and 4% belonging to class 2; a model could achieve a 96% accuracy by simply predicting every training sample belonging to class 1. Such behaviour from a model would cause a lot problems when cost of misclassification for the smaller class is very high: imagine if class 1 refers to patients that do not suffer a recurrence of cancer and class 2 that do. A model with 96% accuracy that classifies all patients as "no recurrence" would send home people incorrectly thinking their cancer is not going to reoccur.

Confusion Matrix Confusion matrix comes into play to solve the bias that accuracy can create in case of unbalanced sample. It is, in fact, a matrix from which it is possible to describe the complete performances of a classification model. Suppose

a two class problem [4], confusion matrix will appear like this:

Table 3.2: Confusion matrix example. The actual size of the database will be $P + N = P' + N'$.

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

From the matrix, it is possible to individuate four different important measures:

- **True positives-** Number of observations correctly predicted as positive ("p") by the model.
- **True negatives-** Number of observations correctly predicted as negative ("n") by the model.
- **False positives-** Number of observations predicted as positive ("p") when in reality they were negative ("n′") (also referred as type 1 error).
- **False negatives-** Number of observations predicted as negative ("n") when in reality they were positive ("p′") (also referred as type 2 error).

From such matrix, it is straightforward to understand that accuracy can be calculated as $Accuracy = \frac{TP+TN}{P+N}$. However, besides accuracy, many other measures containing more detailed information can be measured from confusion matrix. They are, moreover, really useful when dealing with unbalanced classification problems.

[4]It is important to mention that confusion matrix, and all the measures that follow derived from it, can also be calculated in the case of a classification problem presenting more than two classes. In this section for simplicity, the two-class problem is treated.

Sensitivity Referring to the above mentioned case of the classification model for cancer re-occurrence predictions, we would be interested in the ability of the model to spot all the relevant positive cases (i.e. "recurrence") within the dataset. This objective is equivalent to the maximisation of the sensitivity of the model that is calculated as:

$$Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.44}$$

Sensitivity thus can be seen as of a model's ability to recognise all observations of interest within a dataset.

However sensitivity per itself does not solve all problems related to unbalanced classification problem; in fact, from (3.44), if in the example our model labelled all patients as "recurrence", sensitivity would be equal to 1 and one could be tempted to think to have a perfect classifier. Obviously, this would not be true, in fact sensitivity is always in trade-off with precision.

Precision Precision measures the ability of a classification model to identify only relevant data points. It is calculated as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{3.45}$$

So, a cancer model with sensitivity equal to 1 obtained classifying all observations as "recurrence" would have a precision of 0.04 signaling a problem in the construction of the model.

F1 score In some cases, it is desirable to maximise either sensitivity and precision at the expense of other metrics. In our example, it is desirable to have a sensitivity near 1 (i.e. find all patients for follow-up examinations), but, since follow-up examinations are costly, it is desirable to have also high precision. The right measure thus to take into consideration both the measures is F1-score:

$$F_1 = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{3.46}$$

From a mathematical point of view, F1-score is calculated as an harmonic mean of

sensitivity and precision that in comparison to a simple average punishes extreme values of either one of its two components.

Specificity Specificity is the last important measure that can be extracted from confusion matrix. It follows the same idea of sensitivity, but it is focused on correct true negative predictions and not true positive ones:

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} \tag{3.47}$$

Receiver Operating Characteristic Curve Another important instrument to measure the performance of a classification model is ROC curve. It shows the *true positive rate* (i.e. sensitivity) and *false positive rate* (i.e. probability of false alarm or 1-specificity) relationship changes as the threshold of the model for identifying a "positive" in the model is changed.

$$True\ Positive\ Rate = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{3.48}$$

$$False\ Positive\ Rate = \frac{False\ Positives}{False\ Positives + True\ Negatives} \tag{3.49}$$

Two are the important ROC curve characteristics to look at: the top-left corner and the "steepness" of the curve. The the closer the ROC curve to the top-left corner of the graph the better the model. Ideally, in fact, the top left corner of the plot should represent a model with a false positive rate of 0, and a true positive rate of 1. The "steepness" of ROC curves is also important since a steep curve would signal a model able to maximise the true positive rate without increasing the false positive rate indeed.

ROC curve is really helpful especially when comparing different classifiers: a model with a ROC curve more on the left respect to the curve of a benchmark model is considered to be a better classification model for the reasons just explained.

Area Under the ROC Curve From the name of such indicator it is pretty straightforward to understand what AU-ROC represents: AU-ROC refers to the area under the ROC curve of a specific classification model. It is a measure of the ability of a model to distinguish between different classes. An excellent model has a AU-ROC

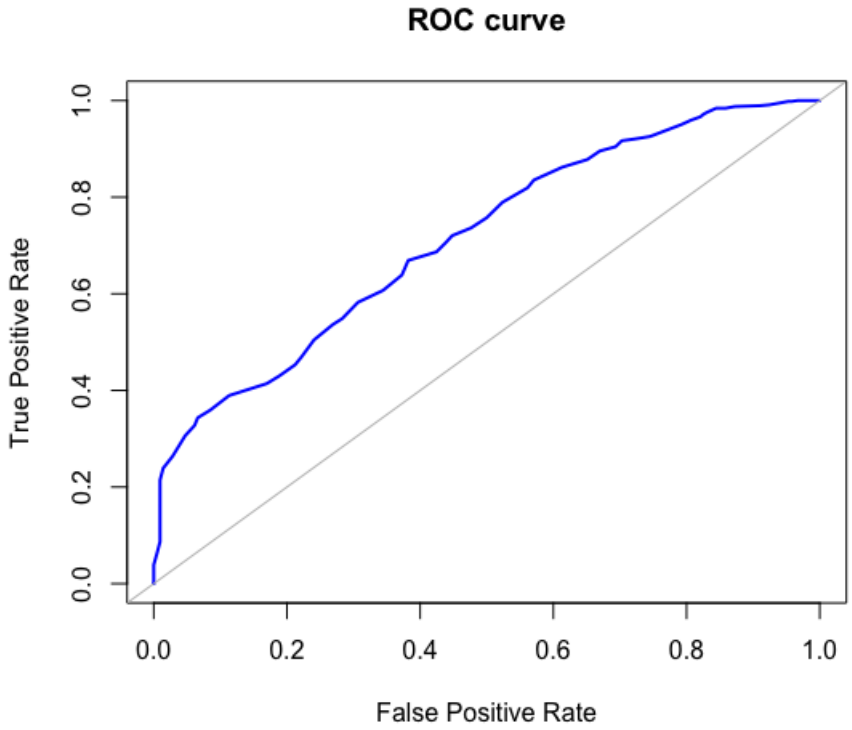

Figure 3.22: ROC curve example. The diagonal grey line represent a random classifier. The blue line represent the result effectively obtained by a classification model.

near 1 which means it has a good measure of separability, while a bad model typically has an AU-ROC close to 0. So the larger the area under the ROC (i.e. the more closed the ROC to the top-left corner of the graph) the better the classifier and viceversa. As from the ROC, also from different AU-ROC it is possible to easily compare the performances of different classifiers.

3.4.2 Financial Performance Metrics

Unfortunately, being the research combining both the financial and computer science domains not so developed as others, proper measures able to provide synthetic information about the performance of a model both from a data science perspective and a financial one have not been developed yet. This forces the validation process to be, as said, literally split in two completely separated parts. As we have already

seen, the first part validates the model following the common measures adopted by computer scientists; anyhow such data science measures are usually not found to be particularly correlated with financial performances. This is the reason why the second part of the validation process result to be crucial, especially from a business viewpoint. In fact, a ML classification model achieving high accuracy and, at the same time, losing money would not be able to justify the research effort necessary to create it.

So, to properly carry out the financial validation of a ML model, it the definition of a precise trading strategy built upon the predictions of the model itself cannot be overlooked. Usually, such strategy has the form of a *statistical arbitrage* strategy consisting in the creation of long-short or long-only portfolios formed by stocks selected on the basis of a ranking of probabilities provided by the model. The definition of the investment strategy is, in fact, preparatory for the calculation of profitability and risk measures related to the strategy itself.

In the remaining part of the section some of these measures will be presented. Due to its more easily quantifiable nature, only two measure of profitability will be exhibited; while regarding the risk perspective, a higher number of measures will be necessary due to the less quantifiable nature of risk itself. Besides, to map all the characteristics of the distribution of returns different risk measures are necessary since, as we will see, each of them focus on single specific aspects of it.

Portfolio Returns Portfolio returns refers to the percentage gain or loss realised by an investment portfolio containing different types of investments. Return of a portfolio can be simply calculated as:

$$Ret_t = \frac{Portfolio\ Value_t}{Initial\ Investment_{t=0}} - 1 \tag{3.50}$$

In case of multi-periods investments, a common practice is to annualised returns so as to make them comparable across other portfolios and potential investments. The annualised return is calculated as the geometric average amount of money earned by the investment over a specified time period: it shows the amount of money that

could have been earned over a year in case of compounded returns.

$$Annualised\ Return = (1 + Cumulative\ return)^{\frac{365}{Hold\ \ period}} - 1 \tag{3.51}$$

Profit & Loss Portfolio returns as expressed in (3.50) cannot be calculated in case of dollar-neutral long-short portfolios. Such type of portfolios are particularly exploited in case of ML classification trading systems since they allow to combine both stocks in a long position whose price is expected to rise and stocks in a short position whose price is expected to fall without investing money at $t = 0$. This is possible thanks to the fact that the portfolio is built so that the amount of money spent for the long positions equals the amount of money received from the sale of short positions. In such a case the denominator of (3.50) is 0 indeed. This is the reason why usually for such portfolios, profit&loss is employed as a measure of profitability and it is calculated as:

$$P\&L_t = Profits\ from\ sale\ short\ positions_t - Costs\ from\ closure\ short\ positions_t \tag{3.52}$$

Standard Deviation Standard deviation of returns is generally employed in finance as a proxy of risk. It express a statistic measure of the dispersion of a dataset in relation to its mean:

$$\sigma_t = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x_t})^2}{n-1}} \tag{3.53}$$

Standard deviation returns information about the historical volatility of a specific investment. The greater this measure for a security or portfolio, the greater the price range in which the value of the security/portfolio moved during the considered time frame and the greater the uncertainty (i.e. risk) suffered by the investor in holding that specific position.

Maximum Drawdown Maximum drawdown is a measure of downside risk over a specified time period expressing the maximum percentage observed losses from a peak to a bottom, before a new peak is reached.

$$MDD = \frac{Trough\ value - Peak\ Value}{Peak\ Value} \tag{3.54}$$

It only measures the size of the largest loss trend without taking into consideration the frequency of losses over the whole time period examined. Moreover, it does not indicate the time to recover from the losses. For these reasons, it can be considered an indicator focused on capital preservation.

Sharpe Ratio The Sharpe ratio measures the profitability of an investment adjusted for its risk.

$$Sharpe\ ratio = \frac{R_t}{\sigma_t} \tag{3.55}$$

The Sharpe ratio gives information about how well the return of an asset remunerate the investor for the risk taken, it is thus a more complete performance measure than simple returns. In financial markets, in fact, returns must be always compared with the risk suffered by the investor since typically they are positive related: high returns and low risk suffered highlights great capabilities by the investor and vice versa. Moreover, Sharpe ratio is particularly useful when comparing the performance of different portfolios: the measure shows whether a portfolio's returns have been due to good investment decisions or a result of a too higher risk suffered; an investment is considered to be a good investment compared to other investments if its returns have not come with excessive additional risk taken indeed.

To better isolate the risk suffered by the investor, as variation of the original formula, at the numerator in (3.55) it is possible to subtract the risk-free rate from returns to better isolate excess returns delivered by the investment.

Sortino Ratio Sortino ratio, similarly to Sharpe ratio, measures the risk adjusted return of an investment. It is a modification of the Sharpe ratio since it penalises only those returns falling below a specified target, while Sharpe ratio penalises upside and downside volatility equally. It is calculated as:

$$Sortino\ ratio = \frac{R_t - Target}{Downside\ Risk} \tag{3.56}$$

$$DR = \sqrt{\lim_{-\infty}^{T}(T-x)^2 f(x)dx} \tag{3.57}$$

Where f refers to the distribution of annuals returns, x to the random variable representing returns and T to the target returns. With respect to the downside risk, as said, it focuses on the negative part of the volatility of the investment: it is observable from (3.57) how it is calculated not taking as reference averages returns but the minimum acceptable return T (usually represented by those of risk-free securities).

From (3.57), it emerges how Downside Risk measures the downward deviations of the yield of the security in relation to a selected minimum acceptable yield, thus expressing that part of volatility not appreciated by the investor: the greater the downward deviation of the yields of the securities compared to the expected return of the investor, the greater DR. It is worth mentioning that in case of a positive deviation the downside risk value would be capped to be 0.

Calmar Ratio Calmar ratio is another possible measure of return adjusted for risk. The risk measure employed to adjust returns for this measure is maximum drawdown. This is the reason why, compared to Sharpe ratio, such measure is more sensitive to extreme losses.

$$Calmar\ ratio = \frac{R_t}{MDD} \tag{3.58}$$

Omega Ratio The omega ratio is another risk-return performance measure frequently employed for portfolios or investment strategies. This measure is employed to asses whether the returns distribution is asymmetric thus signaling, in such a case, a possible ability/inability of the investor in creating the investment strategy. Basically, omega ratio calculates the area under the cumulative distribution function of returns of the portfolio/investment strategy before and after a threshold referred as minimum acceptable return (MAR). The area preceding MAR measures the observed probability of having disappointing results from the investment strategy, while the one following MAR measures the observed probability of having positive returns from it. Finally, Omega simply express the ratio between these two

integrals. Mathematically:

$$Omega(r) = \frac{\int_r^{\infty}(1 - F(x))dx}{\int_{-\infty}^{r} F(x)dx} \tag{3.59}$$

where F refers to the cumulative distribution function of the returns and r refers to the minimum acceptable return (MAR).

Value at Risk VaR is an absolute measure representing the risk of experiencing a loss from an investment. It estimates how much an investment might lose, given normal market conditions, in a specified time period under specified probability thresholds. The measure, as the other risk measure presented, relies on the calculation of the distribution of returns of the investment. Once the distribution has been estimated, VAR is calculated as that value for which a loss greater than it is at most p probable, while a loss smaller than it is at least $1 - p$ probable. For instance, if a portfolio of stocks has a one-week 1% VAR of 5$ million, this means that there is 0.01 probability that the portfolio will lose more than 5$ milion value in one-week with no trading occurring in the meantime.

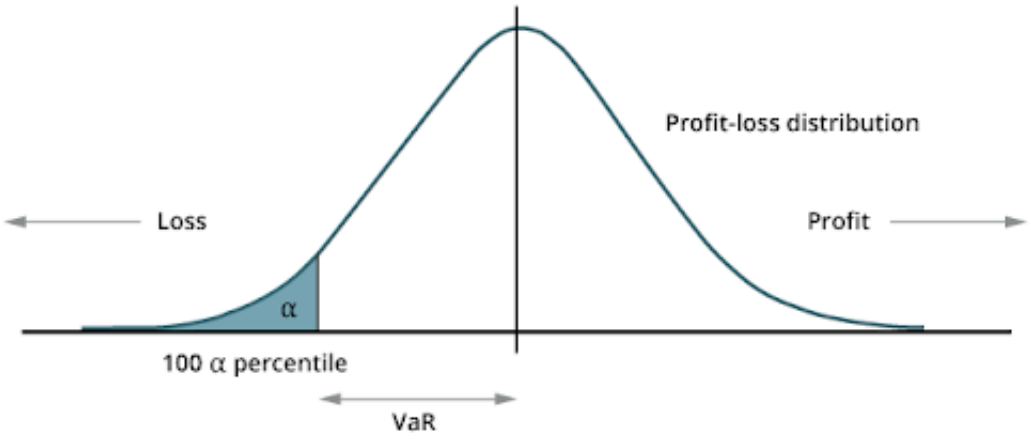

Figure 3.23: Example of value at risk supposing a normal distribution of returns.

From figure 3.23, where distribution of returns is supposed to be normal, the

VaR calculation would be equal to:

$$VaR = \mu + \sigma N^{-1}(\alpha) \tag{3.60}$$

Where μ refers to the mean of returns, σ to their standard deviation and α measures the percentile desired regarding the VAR measure.

Conditional Value at Risk The main problem of VaR is that it does not provide information about the level of financial risk once surpassed the VaR threshold. Conditional VaR (also called expected shortfall) solves this problem since it quantifies the amount of tail risk of a specific investment. Mathematically, cVaR is calculated as the average of the returns values that fall beyond VAR:

$$cVar = \frac{1}{1-c}\int_{-1}^{VAR} xp(x)dx \tag{3.61}$$

where $p(x)dx$ refers to the probability density of getting a return x and c to the cut-off point of VAR. Since cVaR is derived from the calculation of VaR, the assumptions of VaR (e.g. shape of the distribution of returns, cut-off level, periodicity) strongly affect its value.

Benchmark Naive Trading Strategy Finally, trading strategies based on ML model outputs are typically benchmarked with naive strategies. Such benchmark process is performed to check whether eventually positive results of the implemented strategy are due to ML model ability to extract additional information from data or they are due to results attributable to specific positive market conditions.

The most common type of benchmark strategy employed is a *buy and hold strategy* that is a passive investment strategy in which an investor buys stocks and holds them for a predefined period regardless of fluctuations in the market. Another common naive benchmark trading strategy regards the use of market indexes for the markets taken into consideration to build the trading portfolio (e.g. S&P500, DJIA). In any case, possible choices are almost unlimited, and many other possibilities are applicable; it is reported one of them just for example: Fischer and Krauss (2018) asses the financial performance of 100.000 sampled portfolios created

in the sense of Malkiel's monkey throwing darts at the Wall Street Journal's stock page (Malkiel, 2007). They sample 10 "long" stocks and 10 "short" stocks without replacement for each of their main strategy trading day forming a portfolio. Then, they evaluate the mean average daily return of this combined portfolios repeating the process 100.000 times. Finally, they compare the average daily return of such 100.000 random portfolios with their main trading strategy performance.

Chapter 4

Experimental Settings

As presented in Chapter 1, the research question of the thesis is the following one: *is it possible to achieve abnormal returns in equity market through the use of a recent and powerful Machine learning technique?*. In order to contextualise this research question, Chapter 2 presented a review containing different research perspective about the interconnection between machine learning and investment strategies within academic literature and Chapter 3 presented a theoretical framework regarding the application of machine learning models for financial market predictions. From now on, Chapter 4 and 5 will instead present all the details related to the experimental work implemented to effectively answer the research question. Particularly, Chapter 4 will present the methodological structure of the experimental work of this thesis necessary to understand the results presented in Chapter 5.

Differently from the majority of ML approaches presented in Chapter 2, the objective of the experimental work in this thesis is the definition of a suitable ML approach able to create profitable equity portfolios with an investment horizon longer than one day. Specifically, the horizon selected for the experiment is five trading days (i.e. one week).

The study presents two main novelties with respect to the literature. The first main novelty regards the implementation of a solution that in this thesis will be arbitrarily referred as "modular classification approach": the reason why in literature it is not so frequent to find approaches creating investment strategies with

longer than one day horizons has to be attributed to a general degradation of ML model performance when the object of their forecasting is moved forward in time; the objective of this application is thus intended, as it will be clear, to test the truthfulness of such observation even in the case of a state-of-the-art ML algorithm as LSTM networks. At the same time, such solution is intended to improve the financial performance of the whole trading strategy taking advantage of the entire set of information available from data.

The second main novelty proposed regards instead the extension of the input variable set employed to train LSTM models by means of technical indicators and macroeconomic variables. These two types of variable have been already used to train a large variety of ML models but, to the best of our knowledge, not yet in the case of LSTM networks.

4.1 Main Forecasting Idea

One of the biases suffered from trading systems based on supervised ML models is that trading strategies built from the outputs of such models tend to maximise returns, without any consideration about the risk of the investment. Usually, stocks that enter in the portfolio are the ones considered most likely to move in a specific direction given current and past information. The model itself is learned to relate all the input features to the likelihood, and sometimes also to the magnitude, of a specific stock price to move in a specific direction. Within such approaches indeed, risk considerations can be only performed ex-post when computing risk measures. Differently, this is not always true for reinforcement learning approaches. For instance, Almahdi and Yang (2017) propose an unsupervised approach in which the model, through a trials and errors algorithm, is learned to pick stocks (i.e. to generate buy/sell signals) so as to maximise risk adjusted measures as Sharpe ratio or Calmar ratio.

The absence of risk perception by ML models becomes particularly cumbersome

when the forecasting horizon gets larger than hours or days. In such cases, if riskier stocks are frequently selected by models, they could at first make the investor suffer an undesired level of risk, secondly, they could easily erode all the profits previously created by the portfolio. To make matters worse, it has been observed, as said, that models that predict at longer horizons often show disappointing performances in terms of forecasting accuracy.

Since the objective of the empirical work in this thesis is to create a successful ML trading strategy with an horizon longer than one day, to face these problems, it has been decided to adopt the previously mentioned **modular classification approach**. As it sounds evident from the name, it provides for the use of ML classification models; the *classification setting* has been selected since, compared to regression, is less biased towards more volatile stocks: in a regression settings, in fact, stocks selected to enter in the portfolio are the ones whose forecasted returns are higher/lower, conversely, in a classification approach, they are the ones having a higher probability to belong either to a positive or negative price trend regardless of its magnitude.

The term *"modular"*, instead, refers to the combination of multiple prediction models in an attempt to account for risk mitigation: imagine to have a single model forecasting price movements on each Monday for the following one and a trading strategy that, based on such forecasts, creates an equity portfolios to hold for the entire week. A standard approach would not allow the investor to adjust his exposition in the meantime, forcing him not to exploit new information coming from the market in the following days before the closure of all the positions. Conversely, the approach proposed in this thesis aims to create an effective medium-term trading strategy that, through the use of different ML models, takes advantage of the whole information available during the whole week not only to maximise returns but also to lower risks. Practically, continuing with our example, in addition to the first model trained to forecast one week ahead (or any other time period desired by the investor), the "modular approach" consists in training one additional model for each

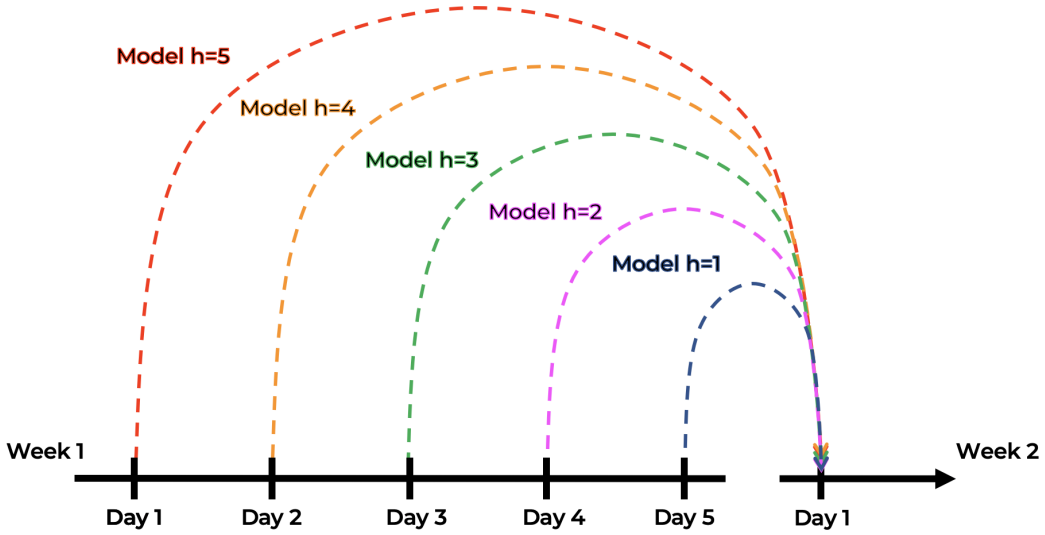

Figure 4.1: Graphic representation of the modular approach in case of one-week ahead forecasting. Each week, for each different day, a different model is used to forecast the price movement of each stock in the database for the first day of the following week.

remained day prior the day in which all positions have to be closed. This translates to different models, where each of them can be seen as a single module of the whole trading system, trained to predict price movements for decreasing time horizons: in our specific case of one week-ahead forecasting, there will be a model $h = 5$ trained to predict with a five-days horizon, a model $h = 4$ trained with a four-days horizon, a model $h = 3$ trained with a three-days horizon, a model $h = 2$ trained with a two-days horizon and a model $h = 1$ trained with a four-days horizon (Figure 4.1).

The main rationale behind the modular approach is the following one: as the forecast horizon h decreases, models are expected to become increasingly accurate so that their predictions can be used both to re-calibrate positions erroneously taken by previous models (i.e. diminish the expositions on stocks whose predicted trend has reversed compared to the one predicted by previous model) and to exploit new investment opportunities (i.e. take new positions based on predictions from theoretically more accurate models).

Moreover, and even more interestingly in terms of research, such an approach allows to effectively test the hypothesis for which deteriorated performance are expected with increasing forecasting horizons. Such hypothesis has already been found true in the case of many ML models extensively employed in financial research.

Even more, due to the scarcity of information regarding medium-long term investment strategies based on machine learning models, such approach is intended to explore from a managerial perspective the possibility to suggest a suitable forecasting horizon range in which an investment strategy can be profitably performed. In this sense, even a disappointing performance from one or more of the models within the experiment would bring valuable information in drawing general conclusion about the effectiveness of specific settings.

Finally, from an investment perspective, it is important to specify that the "modular approach" actualises in a *long-short equity trading strategy*. Long-short strategies are investment strategies that take long positions (i.e. buy) in stocks expected to increase their value and short positions (i.e. sell) in stocks expected to lose value. This decision refers to the possibility, from a research perspective, to evaluate the ability of the various models to recognize not only observations belonging to a single class, as it would have happened in the case of a long-only trading strategy, but two. In any case, details about the constructions of the equity portfolios will be given in the specific section.

4.2 Machine Learning Model Selection

For the construction of the "modular approach", it has been decided to adopt *LSTM networks* to predict stock price movements for all the five different forecasting horizons. In fact, LSTM networks, as explained in 3.2.3, are among the most advanced modification of recurrent neural networks: as RNN, they are specifically designed to take advantage of information included in time series, but, conversely to standard RNN, they are optimised to deal with long-term dependencies. Moreover, applica-

tions regarding the use of LSTM networks for financial time-series predictions are relatively new compared to the vast literature about classification trees, SVM and NN. Finally, compared to other powerful approaches such as GRU, LSTM remains computationally expensive but still feasible with standard hardware configurations.

Specifically, being h the the number of days ahead in which the LSTM network classifies price movements, in this experimental work $h \in \{1, 2, 3, 4, 5\}$. Anyhow, despite different forecasting horizon, for all these models it has been decided to use the same input features set, obviously with the only exception regarding target output. About target output, in fact, depending on the model each observation in the training set has been associated to a given class in relation to its future return at horizon h:

$$Ret_h^s = \frac{P_{t+h}^s}{P_t^s} - 1 \tag{4.1}$$

where s refers to a specific stock and h to the forecasting horizon. However, to prevent models to give a buy or sell signal on non-significant stocks' movements, thus producing a losing trade net of transaction costs, instead of assigning observations to two different classes, it has been decided to assign them to three different ones: the first class containing all positive movements larger than a predefined upper threshold, the second class containing all negative movements lower than a predefined lower threshold and, finally, the third class containing all movements considered not significant as to stay within the upper and lower threshold.

$$\begin{array}{llll} if\ Ret_h^s & > & threshold\_up & s \in \text{class} \quad 1 \\ if\ Ret_h^s & < & threshold\_down & s \in \text{class} \quad -1 \\ Otherwise & & & s \in \text{class} \quad 0 \end{array}$$

Regarding the threshold values for the classes, it has also been decided to make them different depending on the forecasting horizon of the associated model. Starting from an estimation of transaction cost per trade, the threshold values for the $h = 5$ model has been set equal to $5 \times Transaction\ Costs$, the ones for the $h = 4$ model to $4 \times Transaction\ Costs$ and so on. This decision has been taken due in

accordance with the risk management objective of the modular approach previously presented. In fact, it accounts for the return in h days to be higher than the transaction costs that would be originated from an extreme situation in which every day the exposure of a specific security was adjusted. For instance, imagine the situation in which the $h = 5$ LSTM returns a buy signal for the i^{th} stock so that the first day of the current week the i^{th} stock is bought; the second day the same signal is given by the model so that the exposure on the i^{th} stock is increased; the third day the model, at the opposite, outputs a sell signal for the i^{th} stock so that the exposure is decreased and so on. If this happens, each day transaction costs on that particular stock are paid. To prevent losses from a situation like this, it is crucial for the model to signal only stocks that in the following h days can return a profit higher than the maximum possible value of transaction cost to be paid.

Clearly, as it already emerged, such "modular" strategy involves high transaction costs. Anyhow, it is important to remember that it has been created having as main objective a possible way to reduce the variability of returns generated by ML models predictions and, at the same time, as a strategy able to provide information about different behaviours of LSTM models predicting at different horizons. The focus on these two dimensions, more interesting from an academic point of view, then has penalized the optimization of the strategy towards transaction costs.

Regarding the estimation of transaction costs per each trade (bid-ask spread + brokerage fees), the highly conservative measure of 0.2% a day has been used. Considering our database starting in 2009, this choice is even more conservative than the one made by Huck (2019). He deals indeed with observations from 1990 to 2015 and he hypothesises transactions costs of 0.2% per day averaging higher transaction costs of the nineties (i.e. about 0.3%) with lower ones of more recent periods for highly liquid American stocks.

In conclusion, this choice translates into the following thresholds selected per each observation in each of the 5 different models:

$h = 5 \quad threshold\_up = 1.0\% \quad threshold\_down = -1.0\%$

$h = 4 \quad threshold\_up = 0.8\% \quad threshold\_down = -0.8\%$

$h = 3 \quad threshold\_up = 0.6\% \quad threshold\_down = -0.6\%$

$h = 2 \quad threshold\_up = 0.4\% \quad threshold\_down = -0.4\%$

$h = 1 \quad threshold\_up = 0.2\% \quad threshold\_down = -0.2\%$

4.3 Database

Regarding the composition of the database, stocks from the S&P500 index have been selected. This choice is a standard one when testing the effectiveness of ML models in spotting patterns for financial time series: it includes highly liquid and capitalised stocks traded on a market (i.e. the American one) considered remarkably efficient. Due to computational feasibility reasons, among the 500 stocks included in the index at the end of 2019, 4 stocks per each of the 11 industries sectors of the GICS taxonomy have been selected for a total of 44 stocks. Among the same sector, as far as possible, stocks have been selected to be heterogeneous in terms of sub-industry.

This choice, besides being taken to study potential biases from the model towards some peculiar industry, is related with the purpose to create a "universal" model [1] as defined in Sirignano and Cont (2019).

The 44 stocks selected to form the final database are presented in the following table:

[1] Universality refers to the ability of the model to uncover universal features that are common across all different type of stocks.

Table 4.1: Table listing all the 44 stocks presented in the database. Stocks are subdivided following the GICS taxonomy.

Ticker Symbol	Name of the Company	GICS sub-industry
Information Technology		
AAPL	Apple	Technology Hardware, Storage & Peripherals
ACN	Accenture	IT Consulting & Other Services
INTC	Intel Corp.	Semiconductors
MSFT	Microsoft Corp.	Systems Software
Energy		
CVX	Chevron Corp.	Integrated Oil & Gas
XOM	Exxon Mobil Corp.	Integrated Oil & Gas
HAL	Halliburton Co.	Oil & Gas Equipment & Services
OKE	ONEOK	Oil & Gas Storage & Transportation
Materials		
DD	DuPont de Nemours Inc.	Specialty Chemicals
BLL	Ball Corp.	Metal and Glass Containers

	FCX	Freeport-McMoRan Inc.	Copper
	LIN	Linde plc.	Industrial Gases
Industrial Sector			
	MMM	3M Company	Industrial Conglomerates
	AAL	American Airlines Group	Airlines
	CAT	Caterpillar Inc.	Construction Machinery & Heavy Trucks
	UNP	Union Pacific Corp	Railroads
Consumer Discretionary			
	AMZN	Amazon.com Inc.	Internet and Direct Marketing Retail
	F	Ford Motor	Automobile Manufacturers
	MCD	McDonald's Corp.	Restaurants
	NKE	Nike	Apparel, Accessories & Luxury Goods
Consumer Staples			
	WMT	Walmart	Hypermarkets and Super Centers
	KO	Coca-Cola Company	Soft Drinks

	MDLZ	Mondelez International	Packaged Foods and Meats
	PG	Procter & Gamble	Personal Products
	Health Care		
	BSX	Boston Scientific	Health Care Equipment
	HUM	Humana Inc.	Managed Health Care
	JNJ	Johnson & Johnson	Pharmaceuticals
	MRK	Merck & Co.	Pharmaceuticals
	Financials		
	NDAQ	Nasdaq, Inc.	Financial Exchange Data
	AXP	American Express Co	Consumer Finance
	BAC	Bank of America Corp	Diversified Bank
	BRK.B	Berkshire Hathaway	Multi-Sector Holding
	Communications Services		
	VIAC	ViacomCBS	Movies & Entertainment
	DIS	The Walt Disney Company	Movies & Entertainment

	GOOGL	Alphabet Inc Class A	Interactive Media & Services	
	EA	Electronic Arts	Interactive Home Entertainment	
		Utilities		
	ED	Consolidated Edison	Electric Utilities	
	NI	NiSource Inc.	Multi Utilities	
	ATO	Atmos Energy Corp.	Gas Utilities	
	XEL	Xcel Energy Inc.	Multi Utilities	
		Real Estate		
	CBRE	CBRE Group	Real Estate Services	
	WELL	Welltower Inc.	Health Care REITs	
	AMT	American Tower Corp.	Specialised REITs	
	DRE	Duke Realty Corp.	Industrial REITs	

Practically, for each of the 44 stocks, in the time period starting on 4^{th} November 2009 and ending on 15^{th} November 2019 the following daily data have been downloaded from the Yahoo Finance platform [2]:

- Open price (O_t)
- High price (H_t)
- Low price (L_t)
- Close price (P_t)
- Volume of transactions (V_t)
- Close price adjusted for any corporate action (P_t^{adj})

Moreover, as explained in section 3.3, when comparing price information in different periods it is necessary to adjust them for corporate action. For this reason, not only close prices, but also open, high and low prices have been immediately adjusted deriving the adjustment factor directly from P_t^{adj}:

$$k = \frac{P_t^{adj}}{P_t} \tag{4.2}$$

$$O_t^{adj} = k \times O_t \tag{4.3}$$

$$H_t^{adj} = k \times H_t \tag{4.4}$$

$$L_t^{adj} = k \times L_t \tag{4.5}$$

Where k is the adjustment factor directly provided by P_t^{adj}.

Then, all the input features that will be presented in the following section (4.4) have been calculated using these adjusted prices.

[2] https://finance.yahoo.com/

4.4 Input Features

As highlighted in 2.1, the majority of LSTM applications developed so far in literature aim at extracting information only from price features (e.g. returns, high-frequency record of all transaction in the order book). This tendency has to be attributed to the inner ability of LSTM to dynamically recognise patterns in time-series data compared to other static ML model without memory. Theoretically, these latter models, provided with the same data, should not be able to extract even a comparable level of information.

Moreover, another reason is certainly related with the objective of decreasing training time, given that LSTM networks are much more computational expensive compared to static models given the same number of input variables [3], in many applications secondary features have been dropped out from explanatory variables set.

Thus, besides the idea of a modular approach, the second novelty of the experiment here presented is the inclusion in the input features set of technical indicators and macroeconomic variables to test whether they can bring additional information in pattern detection issues or, on the opposite, they only add noise to the prediction application.

Regarding technical indicators, as described in 3.3.3, they are intended as indicator calculated from price features aiming at enhancing the decision process of investors helping them to spot early trends or price reversal. Their effectiveness when included among explanatory variable set of a recurrent network is doubtful since, in some way, they might tend to replicate the task of the LSTM model (i.e. spot trend patterns in time-series data) but forcing the model to focus to some arbitrarily selected measure instead to the whole content of information directly included in prices and volume. Moreover, they are not based on any theoretical

[3] Take as example a NN and a LSTM network both with input layer, a single hidden layer of 10 units each and an output layer of 3 units: in the first case the backpropagation algorithm has to optimise 143 parameters while in the second case 873, 6.1 times more!

framework but refer to heuristic rules sometimes consolidated in beliefs of investors throughout time. However, oppositely, the LSTM model looking at their temporal evolution could spot new patterns that it would not have been able to spot just provided with simple price features: with due propositions, technical indicators could be taught as having the same function of the output of attention mechanisms that aim at helping algorithms to focus on the most pertinent piece of information to improve performance. Results presenting the effectiveness of providing LSTM with synthetic data can be found in Chen and Ge (2019).

Regarding macroeconomic variables, Huck (2019) highlights how in his experimental work random forests, deep belief networks and elastic net regression seem not able to extract valuable information from such data. It is important to specify, however, how even an advance models like DBN, does not have the ability to deal with correlated observations as LSTM. The use of macroeconomic time series in this application, thus, has to do with the hypothesis that from the study of their temporal evolution could come more important insights than the ones coming from static relationships found by the above mentioned models.

Said that, the complete list of 20 input features is here presented:

Table 4.2: Complete list of 20 input features employed for the 5 different LSTM notworks in the experiment. The parameter for technical indicators have been selected using values suggested in literature. VIX index has been downloaded from Yahoo Finance site [6] while TED rate, US treasury rates and US/EU exchange rate from FRED site [7].

Input Feature	**Formula**
Daily Return	$Ret_t^1 = (P_t^{adj}/P_{t-1}^{adj}) - 1$
Weekly Return	$Ret_t^5 = (P_t^{adj}/P_{t-5}^{adj}) - 1$
Simple Weekly Moving Average	$SMA_t^5(P^{adj}) = 1/5 \ \sum_{i=0}^{4} P_{t-i}^{adj}$
Exponential Weekly Moving Average	$EMA_t^5(P^{adj}) = P_t^{adj} \times \frac{2}{6} + EMA_{t-1}^5(P^{adj}) \times \frac{4}{6}$

Moving Average Convergence Divergence	$MACD_t^{12,26}(P^{adj}) = EMA_t^{12}(P^{adj}) - EMA_t^{26}(P^{adj})$
Signal Line	$SL_t^9 = EMA_t^9(MACD_t^{12,26}(P^{adj}))$
Parabolic Stop and Reverse	$SAR_t^{0.01,0.2} = SAR_{t-1}^{0.01,0.2} + \alpha(EP - SAR_{t-1}^{0.01,0.2})$
Relative Strength Index	$RSI_t^{14} = 100 - 100/(1 + RS_t^{14})$
Stochastic %K	$\%K_t^{14} = \frac{P_t^{adj} - min_{[t-13,t]}(L_t^{adj})}{max_{[t-13,t]}(H_t^{adj}) - min_{[t-13,t]}(L_t^{adj})} \times 100$
Stochastic Fast %D	$fast\%D_t^3 = SMA_t^3(\%K)$
Stochastic Slow %D	$slow\%D_t^3 = SMA_t^3(fast\%D)$
Commodity Channel Index	$CCI_t^{20} = \frac{TP_t - SMA_t^{20}(TP)}{0.015 \times MeanDev_t^{20}}$
Average True Range	$ATR_t^{14} = SMA_t^{14}(TR_t)$
%Bollinger Band	$\%BBand_t^{20,2} = (P_t^{adj} - Band_{low}^{20,2})/(Band_{up}^{20,2} - Band_{low}^{20,2})$
Chaikin's Oscillator	$CO_t^{12,26} = EMA_t^{12}(A/D) - EMA_t^{26}(A/D)$
CBOE Volatility Index (VIX)	VIX_t
Daily Return VIX	$Ret_{VIX_t} = (VIX_t - VIX_{t-1}) - 1$
TED Rate	$TED_t = Libor_{3m} - US\ Treasury_{3m}$
US Treasury 10y-3m	$US10y3m = Treasury_{10y} - Treasury_{3m}$
Exchange Rate US/EU	US/EU_t (\$)

At the end of the calculation of all the different features, the final dimension of the database accounts for 2489 observations (from 2526) for each of the 44 different stocks, covering the time period from 22^{th} December 2009 to 12^{th} November 2019. Therefore, overall, the total number of observations observed in the experiment by LSTM networks accounts for $2489 \times 44 = 109.516$.

4.5 Training Process

All five LSTM networks employed in the experiment share the same structure: input layer formed by 20 neurons[8], a single hidden layer populated by a variable number of memory cell units and a dense output layer formed by three neurons[9]. In addition, after the hidden layer, a dropout regularization is applied meaning that a fraction of input units is randomly dropped at each update during training process. Such regularization aims to reduce the risk of overfitting. The selected *dropout value* is 0.1 as in Fischer and Krauss (2018) where authors observe how *"higher dropout values go along with a decline in performance"* (Fischer and Krauss, 2018). As activation function in the last layer, the softmax function [10] is used. The most important characteristic of softmax function is that it outputs values between 0 and 1 that, when summed over the whole layer, equal to 1. This means that, in our specific case, the three values returned by the output layer can be seen as probabilities predicted by the model that the return in h day of the ith stock belongs to the class k. The final predicted class of the model in this way depends on the one among the three output neurons showing the highest probability. Following Fischer and Krauss (2018), the length of the look-back period has been set to 240 days. This choice influences the way in which input matrix for the models is built and its dimension:

[8]The number of neurons in the input layer is always equal to the number of input features.

[9]The number of neurons in the output layer corresponds to the number of classes in which data are divided.

[10]The softmax function is an activation function similar to the sigmoid one. It is frequently used in the output layer since it normalises a K dimensional vector z of arbitrary real values into a K dimensional vector $\sigma(z)$ whose components sum to 1. In other words, the values returned by the output layer can be seen as a probability vector.

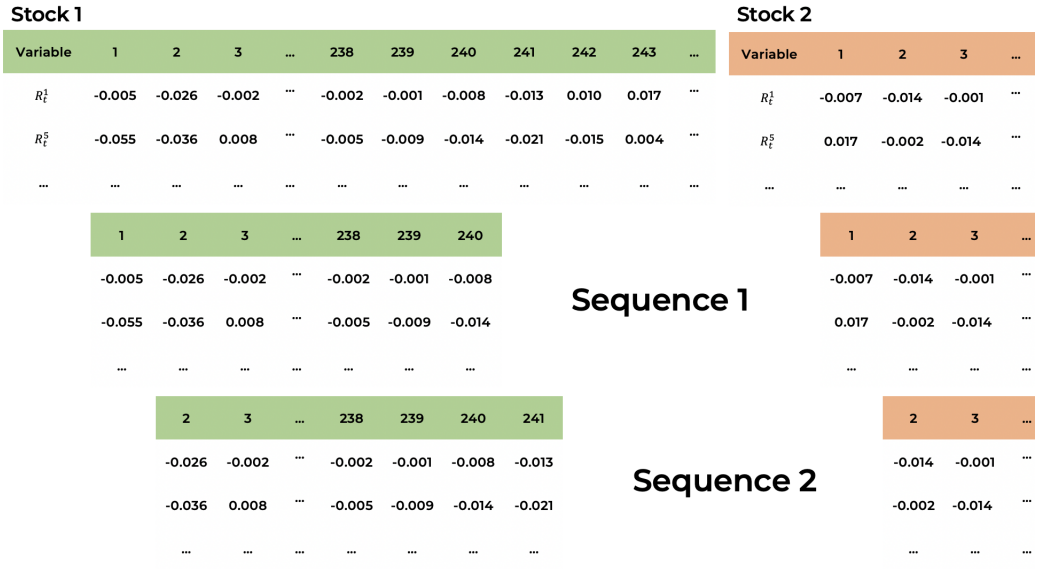

Figure 4.2: Graphic representation of the construction of input sequences for LSTM networks with a look-back period of 240 days (Both feature vector and sequences are shown transposed). In a "many to one configuration" it is clear from the image how the first 239 observations of every stock are not associated with any predicted value. For instance, referring to sequence1, the model predicts the price movement for the observation $240 + h$.

Figure 4.2 shows how the different sequences taken as input by LSTM have been formed. Finally, referring to Figure 3.14, a "many to one configuration" has been chosen, meaning that every model of the experiment returns a single prediction just for the last observation of each input sequence containing 240 days information. The direct consequence of this choice regards the first 239 days of every stock in the dataset for which no value is predicted, but that result to be in any case crucial for the prediction task.

As emerged, the only structural difference among all models trained in the experiment regards the number of hidden nodes composing the single hidden layer, that, due to computational restrictions, has been the only optimized parameter through a validation process. During each training process, indeed, 5 different configurations of the model in terms of number of nodes in the hidden layer have been trained and then validated on a different dataset. As number of neurons in hidden layer, the following values have been tested: 25, 20, 15, 10, 5. The criterion selected to

identify the best parameter among the 5 tested has been the geometric average:

$$G - score = \sqrt{Sensitivity_{(-1)} \times Sensitivity_{(1)}} \tag{4.6}$$

where

$$Sensitivity_{(-1)} = \frac{True\ Positive_{(-1)}}{True\ Positive_{(-1)} + False\ Negative_{(-1)}} \tag{4.7}$$

$$\tag{4.8}$$

$$Sensitivity_{(1)} = \frac{True\ Positive_{(1)}}{True\ Positive_{(1)} + False\ Negative_{(1)}} \tag{4.9}$$

Similarly to Picasso et al. (2019), a geometric average has been preferred over accuracy since, in case of unbalanced training samples, it better takes into account the ability of the model to recognise all significant observations (i.e. those included in both "1" and "-1" classes). As it can be seen, such measure balances the proportion for both the sensitivity of the classes incorporating significant price movements as $Sensitivity_{(-1)}$ and $Sensitivity_{(1)}$. In fact, to achieve a high value of Gscore both sensitivity must be high and assuring to correctly evaluate a model in making correct predictions even when dealing with unbalanced samples.

For each of the 5 different horizons' training processes, a *sliding training window* approach has been adopted: it involved the subdivision of the whole time-series into a series of overlapping training-validation-trading sets so as to simulate real-life trading and to validate the model through frequent out-of-sample datasets. Sliding windows is a standard technique employed when approaching time-series since classic techniques that do not preserve the time order of observations in the data set, such as cross validation, cannot be evidently used to validate models. The term "sliding" refers to the fact that, when passing from one period to another, all the three sets (i.e. training, validation and trading) move forward of the same length of the trading set maintaining their dimension. In this way the temporal relation between observations is preserved and, at the same time, overlapping trading periods are prevented.

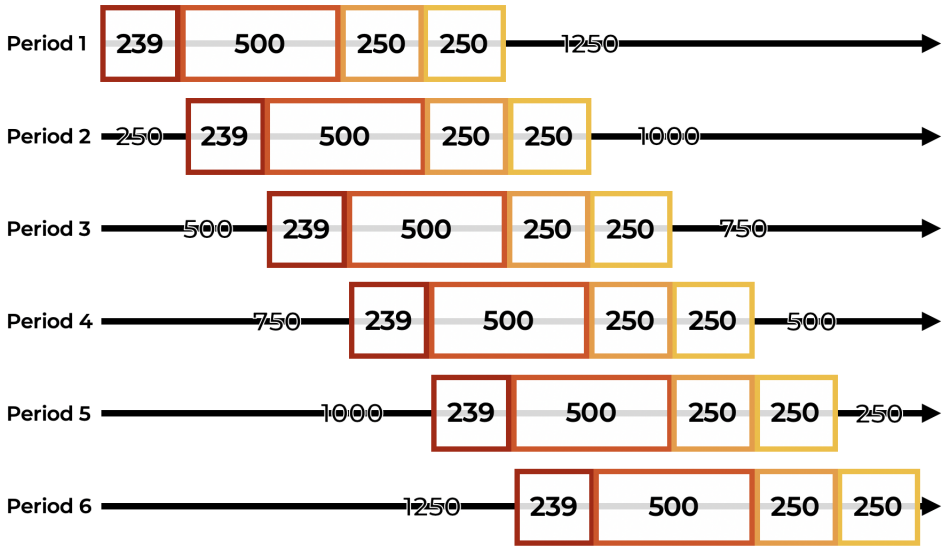

Figure 4.3: Graphic representation of the sliding training window approach. The first box represents the 239 observations per each stock required by the LSTM model and depending from the look-back period selected, the second one refers to the training set, the third one to the validation set in which the hidden layer numerousness is decided and finally the fourth one refers to the trading set.

In our specific case, excluding from the calculation the first 239 observations per each stock necessary for the training process of the first model and dependent on the look-back period selected, the technique has consisted in subdividing the 2250 observations per stocks into 6 different periods/windows containing each a training, a validation and a trading set. Regarding the dimension of these sets, every training set was composed of 500 observations (i.e. approximately two trading years) coming from all the 44 different stocks in the data set plus 239 observations depending on the look-back period selected, accounting for a total of $44 \times (500 + 239) = 32516$ observations. The same for the validation set and trading set composed of approximately one year of trading observations (i.e. 250) plus 239 observations concerning with the look-back period to be multiplied by the 44 different stocks within the dataset as $44 \times (250 + 239) = 21516$.

The whole mechanism adopted is well represented by Figure 4.3 that also shows why, overall, within the dataset 6 different windows/periods are emerged.

Summarizing, accounting for both the modular approach and sliding windows,

the empirical work has required to train overall 5 $Variations \times 6$ $Periods \times 5$ $Horizons =$ 150 different LSTM networks.

About the remaining hyperparameters common to all 150 training processes: the mini-batch size has been set at 250, the number of epochs at 40 per each stock and thus $44 \times 40 = 1760$ for the whole training set; categorical cross-entropy has been used as loss function and, finally, RMSprop as optimiser.

It is worth mentioning how input features within each training set have been standardised as follows:

$$stdX^{s,v} = \frac{x_i^{s,v} - m^{s,v}}{s^{s,v}} \tag{4.10}$$

where the index s refers to the 44 different stocks, the index v to the 20 input features, the index i to the 2489 observations and finally $m^{s,v}$ and $s^{s,v}$ to the mean and standard deviation of observed values for a specific input variable of a stock.

To maintain the same proportions, also the validation set has been standardised in the same way calculating the mean and standard deviation for the same amount of observations of the training set (i.e. 739). For this reason, in the calculation of each mean and standard deviation of each validation set the 250 additional observations preceding the look-back period have been also included.

Both data preparation and handling have been entirely conducted in R (R Development Core Team, 2011), relying on package "TTR" (Ulrich, 2019) for the calculation of technical indicators. The LSTM networks have been developed with Keras (Allaire and Chollet, 2019) on top of Google Tensorflow (Allaire and Tang, 2019).

In conclusion, the pseudo-code representing the process used to train and validate all the six LSTMs in each of the five different horizons is presented in Algorithm 4.

4.6 Trading Simulation

From Figure 4.3 it can be understood how from a database initially containing observations for 10 years, the trading strategy has been effectively implemented

Algorithm 4 Models training over the entire dataset

1: Indexes *start_train*, *end_train*, *start_validation* and *end_validation* are initialised at respectively 240, 739, 740, 989.
2: The vector *first_layer*=$\{25, 20, 15, 10, 5\}$ is created.
3: For p in 1 : 6:
4: For i in 1 : 5
5: For j in 1 : 44

a From the database containing all the 2489 daily observations per each stock, the ones index in the interval [(*start_train*-239):*end_train*] for the jth stock are extracted to form the training set.

b The training set is standardised after the calculation of $m^{s,v}$ and $s^{s,v}$.

c If $j = 1$, the LSTM model is defined having number of neurons in the hidden layer as *first_layer*[i], a *mini-batch size* of 250, a *look-back period* equal to 240, a *dropout value* after the hidden layer of 0.1, a softmax *activation function* for the output layer and categorical cross-entropy as *loss function*. If $j \neq 1$, the step is skipped.

d The model is trained over 40 epochs on the jth stock. At the end of each epoch, network's hidden states are initialised.

A From the database containing all the 2489 daily observations per each stock, the ones index in the interval [(*start_validation*-239):*end_validation*] for the jth stock are extracted to form the validation set.

B The validation set is standardised after the calculation of $m^{s,v}$ and $s^{s,v}$ (m and s are calculated within the set indexed [(*start_validation*-(239+250)):*end_validation*]).

C The following measures are calculated based on the predictions of the trained model on the observations included in the validation set: $Sensitivity_{0,-1,1}$, $Precision_{0,-1,1}$, $Specificity_{0,-1,1}$, $F1_{0,-1,1}$, *Accuracy* and $G - score$.

- *start_train*, *end_train*, *start_validation* and *end_validation* indexes are all increased by 250.

for approximately 6 years (i.e. $250 \times 6 = 1500$ trading days), specifically from 26th November 2013 to 12th November 2019. During this time period, in each window for each horizon, the LSTM that has obtained the highest $G-score$ in the validation set

has been employed to predict price movement for observations in the trading set and thus to effectively implement the proposed trading strategy. In accordance with the validation step and to preserve the experiment from look-ahead bias, observations in the trading set have been standardised using the same mean and standard deviation calculated in the validation set.

As already anticipated, the predictions all models have been used to implement a statistical arbitrage long-short trading strategy (Jacobs and Levy, 1993) with an horizon of one week. Long-short strategies are investing strategies that aim to benefit from both rising and falling prices deriving from inefficient pricing. Such type of strategies, mainly adopted by hedge funds, first of all broaden investment opportunities, since account also short positions, and, moreover, they manage to alter the sensitivity to market movements by shifting the balance of long and short positions. This second characteristics is an important advantage of long-short strategies that may bring to the creation of more diverse and less volatile portfolios when compared to long-only strategies that have 100% exposure to the market. In any case, it must be taken into account when adopting a long-short strategy to face significant losses that, in some cases, can exceed the principal amount invested.

As a particular case of long-short portfolios, the ones built in this experiment have been *dollar-neutral portfolios*: dollar-neutral long-short portfolios are portfolios in which the dollar amounts of both long and short positions, at the time of the creation of the portfolio, is equal so as to make the initial investment for the creation of the portfolio equal to zero (excluding transaction costs). The construction of dollar-neutral portfolios requires the proportion of each stock selected to enter in the equity portfolio to depend on its price at time t; this is made clear in Algorithm 5 where all the exact passages implemented for the calculation of the amount of stock necessary to create such dollar-neutrals portfolios are provided.

Figure 4.4 graphically describes the process of portfolio formation in Algorithm 5.

As it can be additionally noted from the explanation of the portfolio formation

Algorithm 5 Portfolio formation at horizon h at time t

1: Among the stock predicted to belong to class "1" ("-1") at time t by LSTM at horizon h, the 5 ones with the highest probability to belong to such class are selected to enter in the portfolio h. If less then 5 stock are predicted to belong to class "1" ("-1") for that day, all of them will be selected. If for one of the two classes, no predictions are made, on that day no trades will be executed.

2: Assuming the total quantity to be invested in long (short) positions to be equal to 1, this quantity is subdivided among the stocks selected to enter in the portfolio through a weighting scheme proportional to the probability of each to stock to belong to its class as:

$$w_{i,h}^{l(s)} = 1 \times \frac{Prob_{i,h}^{l(s)}}{\sum_i Prob_{i,h}^{l(s)}} \tag{4.11}$$

Where i refers to the ith stock among the set of stocks selected to enter in the portfolio, h to the forecasting horizon, $l(s)$ refers to a parameter for a stock within the long-side (short-side), $Prob_i$ to the output of the softmax function of the LSTM at horizon h referred to the ith stock and w_i to the fraction of the total quantity to be invested in the ith stock.

3: Finally, the quantity of the stock to be bought (sold) is easily found looking at the adjusted closing price of the ith stock at time t as:

$$q_{i,h}^{l(s)} = \frac{w_{i,h}^{l(s)}}{P_i^{adj,l(s)}} \tag{4.12}$$

Where P_i^{adj} refers to the adjusted closing price of ith stock at time t and q_i to the total quantity of the ith stock to be bought (sold).

from Algorithm 5, the weighting scheme selected for the securities within the portfolio has been proportional to the confidence of the ML model regarding the ith stock belonging to a specific class (expressed by the output of the softmax function). Moreover, regarding the size of the portfolios, it has been decided to keep a maximum value of 5 stocks to enter in a long position and 5 in a short position for each day. About this decision, no optimization process or suggested value from literature have been employed; rather, the value has emerged from a heuristic process aiming to solve the trade-off between the selection of a too large value eliminating any discrimination performed by LSTM and a too small value penalising diversification and possibly increasing volatility of the investment strategy. Thus, a maximum

portfolio dimension equal to the 22.7% (i.e. 10 out of 44) of the whole set of stocks in the dataset emerged as a fair value. Nevertheless, due to its subjective selection, the robustness of "maximum portfolio dimension" parameter has been also tested in Chapter 6.

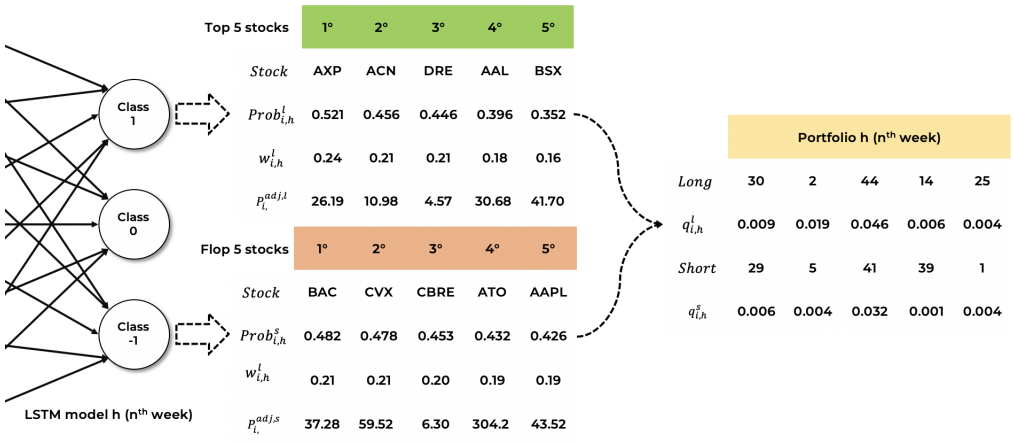

Figure 4.4: Graphic representation of a generic portfolio formation at horizon h at time t

Now, in case the empirical approach had been developed as in most cases in the literature, the explanation regarding the construction of the trading strategy would have ended here. A standard statistical arbitrage approach with an horizon of one week would indeed have required the creation of a long-short dollar-neutral portfolio five trading days before the closure of all positions and nothing else. However, in this particular approach this is not enough. In fact, it results important to understand also how during each week of the trading period, the predictions coming from all the five different models have been gathered to follow the intention of the "modular approach".

In a real world context, the behaviour of the "modular approach", in the days following the first one where the portfolio is initially created, would actualised in the following possible actions:

- If there was already a long position on stock i and its price was expected to

significantly increase in the next h days (with $0 < h < 5$ in our case), the exposition on that stock would be increased.

- If there was already a long position on stock i and its price, differently from previous predictions, was expected to significantly decrease in the next h days, part of the stocks in the portfolio would be sold. In the case the quantity suggested to be sold by the approach exceeds the quantity already in the portfolio, it would be taken a short position exclusively related to the exceeding quantity [11].

- If there was already a short position on stock i and its price was expected to significantly decrease in the next h days, the exposition on that stock would be increased.

- If there was already a short position on stock i and its price, differently from previous expectations, was expected to significantly increase in the next h days, part of short positions would be closed. In the case the quantity suggested to be bought by the approach exceeds the quantity already short in the portfolio, it would be taken a long position exclusively related to the exceeding quantity [12].

- If there was no exposition on stock i yet and its price was expected either to significantly increase or decrease in the next h days, it would be either taken a long or short on that stock depending on the output of the model.

Actually, in this experimental approach to make actionable the process just presented, it has not been created a single portfolio to be adjusted for the following

[11] Imagine a situation in which, from previous days, 5 shares of stocks 1 are present in the portfolio. Imagine, the ML model at time t to suggest to go short on 6 shares. In a real-world situation, all the 5 shares would be sold and it would be taken a short position on a single share of stock 1.

[12] Imagine a situation in which, from previous days, a short position on 5 shares of stocks 1 is present in the portfolio. Imagine, the ML model at time t to suggest to go long on 6 shares. In a real-world situation, the short position would be completely closed and it would be bought a single share of stock 1.

days as just explained, but rather five, one per each day before the closure of all the positions on the first day of the following week. In this way, the decisions taken in every single day have directly influenced the portfolio formation for that day without affecting any previously created portfolio. For instance, if on the first day of the week the model decided to go long on stock i and during the second one to go short on the same stock, within the portfolio corresponding to the first day we fond long positions for the i^{th} stock while within the portfolio corresponding to the second day we found short positions for the i^{th} stock.

Although the decision to create a completely separated portfolio for each day of the week may seem peculiar, assumed no transaction costs it has not influenced the results obtained; even better, it has allowed to simplify calculations and to quantify the partial results related to single forecasting horizons. From a profitability perspective indeed, both the real world example and the approach effectively employed can be considered completely equivalent. In fact, summing all the positions taken by each singular portfolio on every stock i is equivalent to adjust single positions in a unique portfolio. To prove this, imagine a case where a long-short dollar-neutral strategy is implemented in a market composed by two stocks with no transaction costs. Imagine also a ML model suggesting the investor each week in time $t-3$, $t-2$ and $t-1$ (e.g. Monday, Tuesday, Wednesday) to take the following positions to be hold until time t (e.g. Thursday):

	Long	*Short*
$h=3$	$+2$ *stock* 1 (15\$ *each*)	-3 *stock* 2 (10\$ *each*)
$h=2$	$+1$ *stock* 1 (14\$ *each*)	-2 *stock* 2 (7\$ *each*)
$h=1$	$+2$ *stock* 2 (6\$ *each*)	-1 *stock* 1 (12\$ *each*)
$h=0$	*stock* $1 = 15\$$ *each*	*stock* $2 = 10\$$ *each*

The weekly Profit/loss deriving from a strategy accounting for a single portfolio

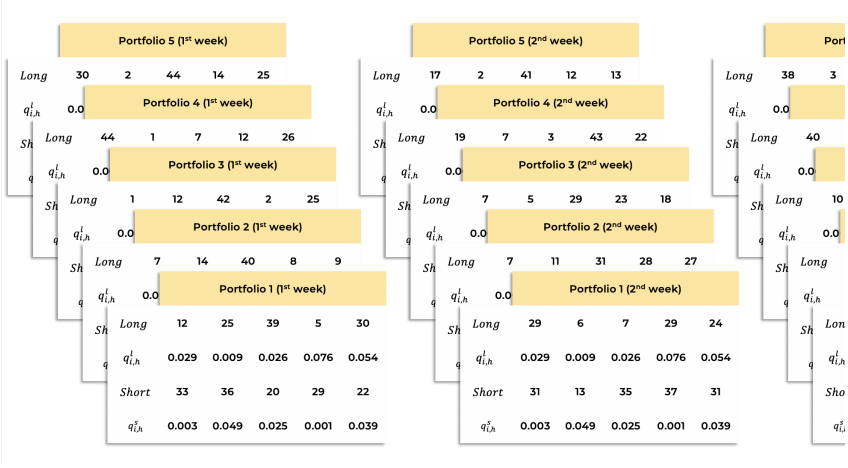

Figure 4.5: Graphic representation of the five portfolio created at different horizons per each week in the "modular approach".

adjusted daily would be calculated as:

$$For\ stock\ 1 \qquad P\&L = -2 \times 15\$ - 1 \times 14\$ + 1 \times 12\$ + 2 \times 15\$ = -2\$$$

$$For\ stock\ 2 \qquad P\&L = +3 \times 10\$ + 2 \times 7\$ - 2 \times 6\$ - 3 \times 10\$ = +2\$$$

$$For\ Portfolio \quad P\&L = -2\$ + 2\$ = 0\$$$

For such a short time period, values are not actualized.

While the weekly P&L deriving from 3 independent portfolios created at $h = 3$, $h = 2$ and $h = 1$, all closed in $h = 0$, would be calculated as:

$$For\ Long\ sides \quad Profits = 3 \times 15\$ + 2 \times 10\$ = 65\$$$

$$For\ Short\ side \quad Losses = -1 \times 15\$ - 5 \times 10\$ = -65\$$$

$$For\ Portfolio \quad P\&L = +65\$ - 65\$ = 0\$$$

For such a short time period, values are not actualized.

As it can be seen, without considering transaction costs (or considering them equal for both long and short trades [13]) the P&L of the two approaches is always the same. This is the reason why the simplification of independent portfolios made in

[13]This assumption is frequently made in applications in academic literature since a precise estimation of all transaction costs related with trades is difficult to be made.

this experiment has not influenced results: in the next chapter financial performance will be presented considering a situation with no transaction costs indeed.

Moreover, the creation of different portfolios has been preferred because it has allowed to easily verify the contribution of the different LSTM predicting at decreasing forecasting horizons to the total profit/loss obtained during the entire week.

In any case, it is worth highlighting how in the example in $h = 1$ the two approaches behaves differently: the first one sells a single unit of *stock* 1 from the 3 present in the portfolio and close 2 of the 5 short positions taken on *stock* 2; the second one, without affecting the positions taken in $h = 3$ and $h = 2$, goes long on two additional units of *stock* 2 and short on a single unit of *stock* 1. Such differences become relevant when considering asymmetric transaction costs; in real transactions, in fact, costs related to short-selling tend to be higher than the ones related to long positions. Under this circumstances, the second approach would generate higher transaction costs.

In conclusion, Figure 4.5 graphically synthesises how the single portfolios created (as in Figure 4.4) in each trading week throughout the 6 years period have been merged to form the final trading strategy. The Figure also helps to finally understand how weekly P&L of the "modular approach" have been calculated:

$$
\begin{aligned}
&Portfolio\ h = 5 && P\&L_5 = \textstyle\sum_i q^l_{i,5} \times P^{adj,l}_i - \sum_i q^s_{i,5} \times P^{adj,s}_i \\
&Portfolio\ h = 4 && P\&L_4 = \textstyle\sum_i q^l_{i,4} \times P^{adj,l}_i - \sum_i q^s_{i,4} \times P^{adj,s}_i \\
&Portfolio\ h = 3 && P\&L_3 = \textstyle\sum_i q^l_{i,3} \times P^{adj,l}_i - \sum_i q^s_{i,3} \times P^{adj,s}_i \\
&Portfolio\ h = 2 && P\&L_2 = \textstyle\sum_i q^l_{i,2} \times P^{adj,l}_i - \sum_i q^s_{i,2} \times P^{adj,s}_i \\
&Portfolio\ h = 1 && P\&L_1 = \textstyle\sum_i q^l_{i,1} \times P^{adj,l}_i - \sum_i q^s_{i,1} \times P^{adj,s}_i \\
&Total\ weekly\ return && P\&L_{week} = P\&L_5 + P\&L_4 + P\&L_3 + P\&L_2 + P\&L_1
\end{aligned}
$$

$q^l_{i,h}$ and $q^s_{i,h}$ refer to Algorithm 5. All P^{adj}_i refers to the first day of the following week in which all the positions taken by the five portfolios are taken.

Chapter 5

Experimental Results

In this chapter, all the results obtained employing the approach presented in Chapter 4 will be provided. Initially the focus will be on data science metrics (Section 5.1), thereafter, the attention will shift towards the financial performance of the different LSTM models (Section 5.2). Finally, with the aim of further providing useful hints regarding managerial decision for an effective implementation of LSTM networks for stock market predictions, in Section 5.3 the robustness of LSTM networks' performance against specific decision variables (i.e. portfolio weighting scheme, portfolio dimension) will be verified.

All the results that will be shown have been achieved thanks to the computation power of a "t2.2xlarge" Ammazon Web Service instance [1]. The LSTM networks provided by the training process and presented in Section 4.5, have been trained indeed on the CPU of such machines. The time occurred to train all the networks is reported in Table C.1.

5.1 Data Science Results

Premise About Future Returns Distribution

As already specified, after every training stage, each model has been validated out-of-sample on a validation dataset so as to select the best network configuration to be used within the following trading phase. As criterion to select the most appropriate

[1] https://aws.amazon.com/it/ec2/instance-types/t2/

model, the G-score measure has been selected instead of accuracy. Among the already mentioned reasons of such choice, one of them was about the presence of a non-homogeneous distribution of observations regarding the three different classes in the dataset. As can be seen from Table B.1, this is actually the case of our database where for all the five horizons there is a prevalence of observations belonging to class "1" (referring to positive and significant price movements). This prevalence can be mainly attributed to the positive trend that has affected the S&P index during the six-years time period considered (Figure 5.1). Always from Table B.1,

Figure 5.1: Uptrend of the S&P index within the time period considered in the analysis.

in any case, this prevalence do not appear an essential issue in terms of training process: the numerousness of the smaller classes can be still considered significant to be identified by LSTM models especially in $h = 5$, $h = 4$ and $h = 3$. In $h = 2$, the size of observations belonging to class "0" (i.e. negative and significant price movements) start decreasing and exclusively in $h = 1$, in the majority of cases, it results to be lower than 16%.

Moreover, when examining the results from the table, a secondary issue can

emerge: the peculiar distribution of observations among classes in $h = 1$ (i.e. most of observations in the two extreme classes and few in the central one) could suggest a bimodal distribution of future returns. To confute such hypothesis and to verify the distribution of future returns to be as normal as possible, a graphical test has been performed. The test immediately shows how the bimodal hypothesis can be rejected for all horizons, especially for $h = 1$ (Figure B.1 and Figure B.6). Rather, from plots of distributions of future returns in Appendix (Figure B.1 to B.10) it is possible to verify how, as expected, returns appear neither to be normally distributed due to fat tails, but rather leptokurtic (e.g. Figure 5.2 shows an example of fat tails distributions that characterizes all returns within the dataset at all different horizons).

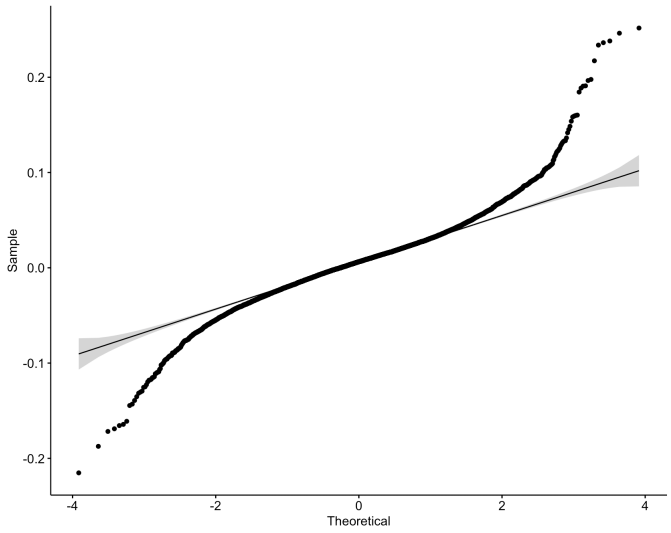

Figure 5.2: Example of Q-Q plot for returns within the experiment. In the specific case the image represent returns distribution belonging to the first window's training set for database $h = 5$.

Evidences

The premises about the numerousness and the distribution of future returns within the three classes is important in order to comment the results achieved by the different models during the validation phase. They have been decomposed depending on the forecasting horizon of the model that has generated them:

- **(h=5)** From Table C.6 in appendix it is possible to observe how the majority of the models, apart partially the ones in the last two training windows, are not sensible to the class "0". This result, for the case $h = 5$, cannot be attributed to the low number of observations within such class. In fact, the number of observations in both classes "-1" and "0" are comparable but, in any case, all models result to be much more sensible to the first one. Instead, the result can be more easily attributed to the inability of the model to recognise clear patterns to distinguish significant from not significant price movements. Moreover, an additional difficulty by the models could come from the fact that class "0" contains observations showing both positive and negative, even if not significant, price movements. On the opposite, the sensitivity for the other two most important classes "-1" and "1" looks satisfactory. It can be further noted how precision for class "-1" is frequently lower than the one for class "1"; such result could be instead attributed to the lower numerousness of class "-1" compared to the one of class "1" (the assumption is that lower numerousness comes with fewer possibilities by the model to train on a specific price movement and ultimately poor performance).
  In terms of accuracy, taking as reference the performance of a random classifier (i.e. 33%), at least one model in each training window, with the exception of the fifth one, reaches a significant result of minimum 37%. In terms of G-score, the worst performances are accounted in the two most recent training windows with values lower than 50%. It is peculiar to note how in the fifth training window, for the model "20/5", the G-score criterion selects a model with an accuracy lower than the reference value of 33% (i.e. 32.5%). In this specific case indeed, in terms of accuracy the model is clearly penalised by its non-sensitiveness to observation belonging to class "0" but at the same time it results to be the best one in recognising observations of both classes "-1" and "1", the ones that effectively maximise profit if correctly identified.

- **(h=4)** Results for $h = 4$ models can be found in Table C.5. From the Table, the same considerations about the sensitivity of the models regarding the class "0" made for $h = 5$ can be replicated for $h = 4$: models are not able to recognise non-significant price movements, especially in the first 4 training windows.
  In terms of accuracy, in each window it is possible to find models showing valuable results ranging from 38.4% to 40.6% with the exception, as in $h = 5$, of the fifth window. Actually, as it becomes clear when examining all the results among all horizons, within the fifth window all models experience a drop in performance. In calendar time, observations belonging to the fifth training set start on 24^{th} November 2014 and end on 16^{th} November 2016. Within such period, more precisely between June 2015 to June 2016, stock prices experienced a decline in their value globally. From Figure 5.1 it is visible how S&P index suffered temporarily losses as a result of some events that frightened investors as the slowing growth in the GDP of China, a fall in petroleum prices, the Greek debt default in June 2015, the effects of the end of quantitative easing in the United States in October 2014 and the 2016 United Kingdom European Union membership referendum, in which Brexit was voted upon. Theoretically, such a period should have been the perfect period for a ML model to spot patterns revealing market inefficiencies due to a higher level of irrational conducts of investors. However, when looking at the time boundaries of the different training windows, it emerges how all models within the fifth window have been trained with observations covering the whole period of market turbulence while being validated on a validation set with totally different market conditions: Table B.1 shows how the numerousness of observations significantly changes (i.e. significant decrease of observations belonging to class "-1") from training to validation set within the fifth training window indeed due to above mentioned change in market condition occurred between the two periods covered by training and validation set. The hypothesis regarding the under-performance of all models within the fifth window is thus

related to the impossibility of models to spot valuable patterns in out-of-sample observations being themselves trained on too dissimilar market conditions. Finally, always regarding the fifth window, sensitivity and precision highlight the increasing bias of models for class "1" along with the decreasing complexity of the networks.

A similar biased behaviour is also observed for models within the third window where, even if the overall level of accuracy reached seems satisfactory, from measures of sensitivity -1 and sensitivity 1, it is possible to prove how results are flawed: as number of nodes in the hidden layer decreases, LSTMs start predicting all observations as belonging to class "1"; sensitivity -1 decreases until 7.1% while sensitivity 1 increases until 94.8% along the window indeed. A similar trend can be noted also for all other 4 horizons thus toughening the hypothesis above mentioned of too dissimilar market conditions between the training and validation set. In calendar time, in fact, the period in which all models of the third window are validated starts on 24^{th} November 2014 and ends on 19^{th} November 2015 thus covering the initial part of the market turbulence of 2015-2016. Also in this case, from the numerousness of the different classes within each dataset, it is possible to note a substantial increase among the number of observations populating the class "-1" from the training to the validation set attributable to the negative trend affecting the equity market from June 2015.

A final evidence to support the hypothesis for which from too dissimilar market conditions among training and out-of-sample data comes poor performances by the models, originates from results within the fourth window. During such time period, LSTM models achieve higher performances compared to other time period, among all horizons, in terms of accuracy and even more importantly in terms of G-score: in the light of the hypothesis, this peak in performance could be explainable by the fact that observations belonging to the 2015-2016 market turbulence are present both in the trading and in

the validation set of the window. The training set, indeed, starts on 26^{th} November 2013 and it ends on 19^{th} November 2015, including the first part of the "anomaly" period, while the validation set starts on 20^{th} November 2015 and it ends on 16^{th} November 2016, including the second part of it. Such over-performance, in the case the hypothesis is proven, would be even in accordance with the expectation of higher performances of ML models during a period of high market turbulence in which the irrationality of investors creates significant arbitrage opportunities, provided that the model has been trained with observations for similar turbulence periods.

Finally, for all the remaining windows, sensitivities of "1" and "-1" classes appear satisfactory and, as in $h = 5$, even in $h = 4$ models result to be more precise for class "1" than for class "-1".

- **(h=3)** From Table C.4 it is possible to see how LSTM models in $h = 3$ achieves higher performances when compared to larger horizons in terms of G-score, particularly in the first two windows. The fourth window, in accordance with other horizons, results to be the one where models achieve higher performances both in terms of accuracy, reaching 42,7%, and G-score, with all 5 LSTMs reaching at least a value of 53.4%.

  The high accuracy values, all above 40%, of the third window are biased by the fact all five models tend to predict most of the observations as belonging to class "1" and by the fact that validation set tends to be more unbalanced than in $h = 5$ and $h = 4$. To prove this, the majority of models in this window achieve poor G-score values: "25/3" 37.7%, "10/3" 35.8% and "5/3" 29.6%.

  Finally, it is worth noting how models in the fifth window, that result to be the only ones sensitive to class "0", achieves the worst performance in terms of both accuracy and G-score.

- **(h=2)** From Table C.3, we see how all models in $h = 2$ are totally unable to recognise observations from class "0" due to the low number of observations

belonging to it. Models in the third window behave as in all other horizons predicting the majority of observations as belonging to class "1". Almost every model has an accuracy around 40% with the exception, as expected by the hypothesis about changing market conditions previously cited, of models in the fifth window that range around 37%. Finally, the same considerations made for all other horizons regarding precision and biased model within the second window can be done.

- **(h=1)** From Table C.2 it is clear how also models in $h = 1$ have no sensitivity regarding observations in class "0". Such low sensitivity in any case was expected due to the particular above mentioned distribution of observations within the three classes. The lowest accuracy among all windows is reached within the fifth one. In particular, the model "10/5", with a performance of 38,9%, result to be the worse for $h = 1$; it is interesting to highlight how this performance if compared to the ones achieved by models predicting at longer horizon looks noteworthy. It appears thus evident how, as expected, better prediction performance seem associated with shorter forecast horizons.

At the end of the training and the following validation process, the parameter selection regarding the number of neurons populating the hidden layer has been performed following as criterion the G-score. Table 5.1 reports the result of such selection process. The first important notation about the selection regards the absence of the prevalence of one preferred configuration over the others. As visible also from Table 5.2, it is present heterogeneity among the selected hidden nodes values overall. However, looking at each singular window (i.e. looking at model predicting at different horizons with exactly the same training input data), always from Table 5.1 it is possible to recognise a precise preference by the third window for more complex configurations; this finding is in line with results previously presented as they showed how simple configurations within the training window were not sensitive towards negative price movement observations. The same consideration can be

Table 5.1: Number of hidden nodes selected per each model in every training window per each different horizons using as criterion the G-score obtained in the validation set.

Window	**h=5**	**h=4**	**h=3**	**h=2**	**h=1**
1	15	5	10	25	5
2	10	5	5	5	20
3	25	20	15	20	15
4	20	10	15	25	10
5	20	25	5	20	15
6	5	15	5	15	25

Table 5.2: Absolute and percentage preference for possible network configurations at the end of the validation process.

Hidden Nodes	**Times Selected**	**Percentage %**
5	8	27%
10	4	13%
15	7	23%
20	6	20%
25	5	17%

made also for the fifth window with the exception of the $h = 3$ horizon. Finally, it is worth mentioning how the use of G-score as selection criteria in place of accuracy has brought 14 times over 30 to a different configuration preferred.

Finally, at the end of the parameter selection phase, LSTM models have been effectively implemented to make prediction for all the observation within various trading sets. Their results are presented in Table C.7 from which it is possible to draw the following considerations:

- As already found within the different validation sets, also in the trading sets all models result absolutely unable to distinguish observations belonging to class "0". The only exception is given by the model "5/6" in $h = 5$, in any case the result cannot be considered statistically significant.

- As in validation sets, precision is higher for class "1" than for class "-1".

- In $h = 1$ and $h = 2$, all models achieve a performance higher than the reference value of 33%. The worst performance among them is achieved by the model "25/4" in $h = 2$ with 35.9%.

- It is found an under-performance of all models within the fourth window of all trading sets. The reason, as previously explained for the validation process, lies in the presence of part of the observations of the 2015 turmoil within the training set and the absence of them within the trading set. In particular, the worst performances are reached by both "10/4" and "20/4" models in $h = 4$ and $h = 5$ that are found to achieve a lower accuracy than a random classifier, respectively 30.5% and 25.3%. It is clear from all sensitivity values how such under-performances are due to a bias in favor of class "-1". The bias intensifies with increasing forecasting horizons: in $h = 5$ the sensitivity for class "1" reaches 6.6% with a precision of 39.7% while having for the class "-1" respectively 76.5% sensitivity and 22.5% precision. In this framework, the model "10/4" predicting at $t+1$ is the only one that is found not to excessively suffer from unbalanced datasets within the fourth window suggesting how, generally, models predicting at lower forecasting horizons are able to spot more solid patterns than all others. This is further confirmed by all high accuracy values reached among all windows in $h = 1$.

- The same under-performance of the fourth window is verified within the fifth one, especially for $h = 5$, $h = 4$ and partially for $h = 3$. Also in this case it has to be attributed to the presence within the training set of all the observations belonging to the market turbulence of 2015-2016 and to the absence in the trading set.

- It is clear from accuracy results how performance deteriorates as forecasting horizon increases. This result was in any case expected and it is the reason for which a "modular approach" has been selected: the hypothesis has been to test whether the possible identification of market movements in advance com-

pared to the most common ML investment strategies could improve financial performance. In terms of data science evidence, results do not seem to support such choice. Even if some model in $h = 5$ managed to reach consistent results (e.g. "10/2" and "25/3").

- In terms of G-score, no clear pattern among windows is recognised. However, as in the case of accuracy, but less clearly, performances deteriorate as forecasting horizon increases.

Overall, the different models have produced results in line with expectations in terms of data science metrics: it has been highlighted a pronounced deterioration in performance with increasing forecasting horizon and a difficulty of the different models in recognising non-significant price movements. Both results were somehow expected in any case.

During specific time period, models trained-tested on too dissimilar market conditions have demonstrated to under-perform in relation with their usual behaviour. In the next section it will be clear if such issue have also affected the financial performance of the trading strategy. Regarding this point, it is worth mentioning how the trading period in the experiment lasts for only six years and how, in such a short period of time, it is possible that a particular condition of the market influences the results of the whole strategy. The focal point in any case must be that, in the long period, the assumption behind the whole experimental work (i.e. market conditions among different sets are similar) is verified most of the times so as to hopefully nullify possible under-performances during peculiar volatile periods. In this thesis, the limited length of the trading period has been dictated by the limited computational capacity and by time constraints. Moreover, there would have been required particular attention to lengthen the trading period as the period immediately preceding the one examined included the 2008 crisis.

As conclusive observation, that possibly opposes to the one referring to too dissimilar market conditions, it is noted a general degradation of results when passing

from the validation set to the trading one. This could be possibly attributed to the long validation period (i.e. one year) that has distanced the live trading period from the training period. From a raw comparison of accuracy and G-score achieved by models within the different trading sets and within validation sets, the benefit of a so long validation process is not clear indeed: accuracy reached in each window of the trading set results higher than the mean of accuracy reached in the same window by the five LSTM variation in the validation set only in 12 cases out of 30; when the same comparison is made with G-score, only in 9 cases out of 30 the performance has been improved. This result questions the real effectiveness of the validation process, or, even better, it gives evidence of how the length of the validation process should be accurately selected so as to be long enough to provide statistically significant results but at the same time not to excessively distance the trading period from the training one. Saying this, it is worth mentioning how in the experiment, due to computational constraint, the grid search applied during the validation process exclusively aimed at finding the optimal number of hidden nodes. Such search could be thus considered limited since it does not include many other parameters related to the training process as learning rate, number of epochs etc.

5.2 Financial Results

As already anticipated, this section presents results from the different LSTM networks trained from a financial perspective. In terms of profitability, all the results will be expressed by Profit&Losses. Dealing with dollar neutral portfolios indeed, the initial investment to create each portfolio accounts for 0$ preventing from the usual calculation of returns. The P&L obtained from every portfolio will be thus referred to the difference between the incomes related to the sale of all the long positions and the costs related to the closure of all the short positions. However, in our experiment such P&L can be considered as similar measures to returns, in fact they can be thought as the percentage gain related to the invest-

ment accounted for the strategy for both the long and short side of the portfolio (that in this experiment amounts to 1$). In addition to profitability, also the risk profile of every strategy/portfolio will be examined through the calculation of the standard deviation of weekly returns, the annualised Sharpe ratio (calculated as $(mean(weekly\ returns)/std.\ dev.(weekly\ returns)) \times \sqrt{50})$, the maximum drawdown, the omega ratio having a loss threshold equal to zero and the value at risk at 99% confidence level.

5.2.1 Global Approach

In terms of profitability, the "modular approach" presented in the previous chapter provides a profit over the 6-years period of 0.783. Figure 5.3 shows graphically the P&L of the strategy over the whole trading period composed by 300 trading weeks. Referring to the six different trading sets, in the graph they correspond to fifty trading weeks each. This means that every fifty trading weeks the result of the trading strategy has been achieved by different LSTM models. As it can be noted, in the periods corresponding to the first, the third, the fourth and the sixth training window models manages to significantly increase the profit of the strategy. On the opposite, in the other two windows, the performances of the models are poor.

If these findings are compared with Table C.7, profitability seem to be partially decorrelated with data science metrics: trading periods with high accuracy or G-score does not have necessarily brought to higher profits and vice versa. For instance, within the second window, the periods in which in terms of accuracy all models highly perform among all horizons, the strategy records profits close to zero; oppositely, the fourth window corresponds to the one with the worst performances in terms of accuracy for all horizons (the LSTMs in $h = 4$ and $h = 5$ even perform worse than a random classifier) and, despite this, the strategy records positive profits during it. Table 5.3 help quantifying more precisely such observation: during the first 50 trading window the profits of the strategy are equal to 0.247, during the third 0.434, during the fourth 0.277 and during the sixth 0.329; oppositely, dur-

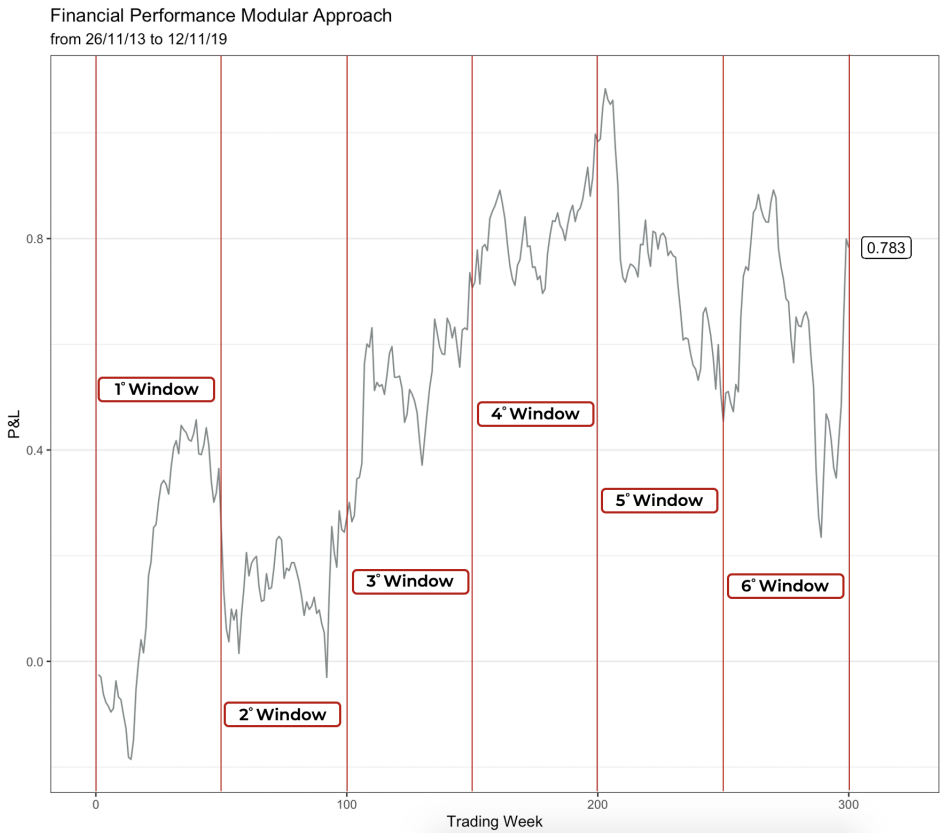

Figure 5.3: Graphic representation of profits and losses deriving from the modular approach described in Chapter 4.

ing the second window they are just 0.025 and during the fifth window the losses amount to -0.530. It is worth mentioning how, in reality, also the profits achieved during the last window would have been negative without the spike obtained during week 299. Oppositely, in the other three positive windows the results appear to be much more consistent and not related to any extraordinary single performance. Such results will be however better examined when the overall performance of the trading strategy will be decomposed into its elementary components, that are the profits obtained by the five portfolios at different h.

From a risk perspective, the results of the strategy are presented in Table 5.4. The first consideration that can be immediately drawn from the table, that is also perceivable from the P&L Figure, is that the strategy records very high levels of

Table 5.3: Table representing half-yearly P&L and cumulative P&L during each of the 12 semesters accounted for the "modular approach" presented in Chapter 4.

Semester	Half-yearly P&L	Cumulated P&L
1	0,302	0,302
2	-0,056	0,247
3	-0,090	0,157
4	0,115	0,272
5	0,242	0,514
6	0,192	0,707
7	0,040	0,746
8	0,237	0,983
9	-0,178	0,806
10	-0,352	0,454
11	0,232	0,686
12	0,097	0,783

Table 5.4: Risk performance of the trading strategy based on the "modular approach" presented in Chapter 4. "Avg.ret" refers to the mean of weekly profits achieved by the strategy, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "Max Drawdown" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

Avg.Ret	Std.Dev	Sharpe Ratio	Max Drawdown	Omega	VaR
0,0026	0,0487	0,3788	0.6161	1.1565	-0.1105

volatility: the performance in terms of annualised Sharpe ratio is indeed very poor. Although during some weeks the strategy manages to create significant positive profits, at the same time it suffers from large drawdowns that, in turn, lower the mean of returns and increase the standard deviation. At this stage of the analysis, however, it is not yet possible to understand by which elementary portfolio, and thus which LSTM, such losses come from. Overall, it can be noted how the omega ratio is higher than 1 meaning that the probability of experiencing a profit during a trading week has been higher than the one related to a loss for the "modular approach" throughout the period examined.

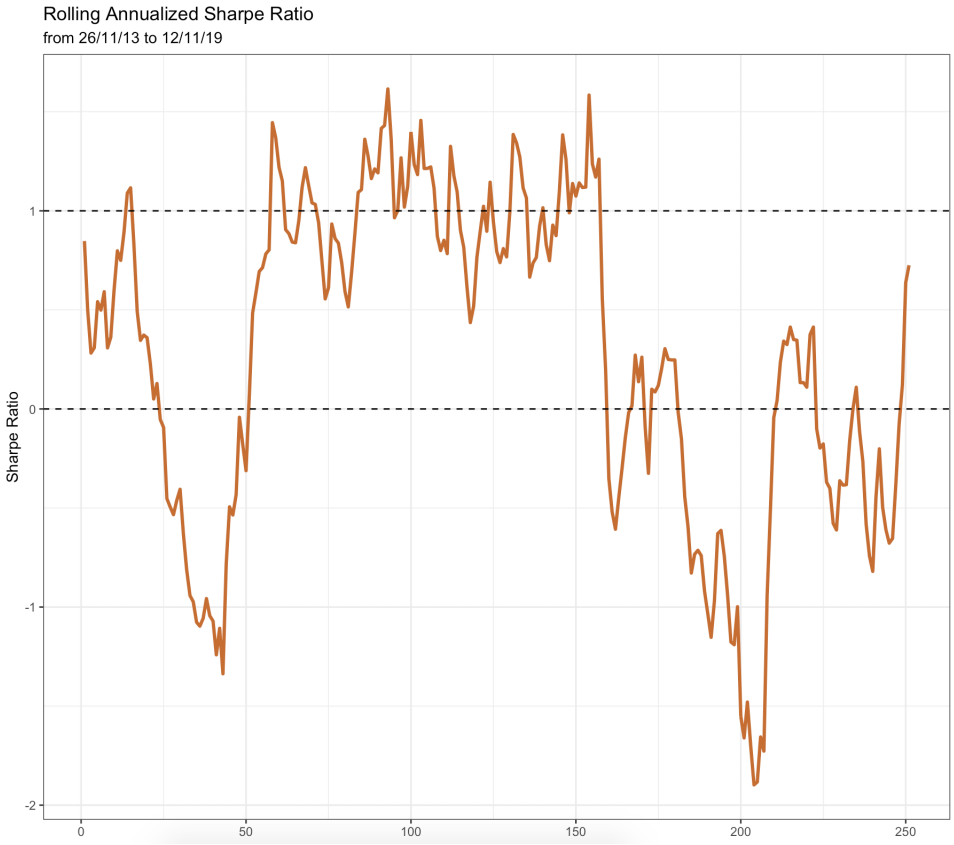

Figure 5.4: Graphic representation of rolling annualised Sharpe ratio calculated for profits obtained from trading strategy originated with the "modular approach" presented in Chapter 4. The rolling window over which the Sharpe ratio has been calculated is formed by observations coming from 50 trading weeks shifting forward one week ahead at every step. In such a framework, in the graph are represented 251 different measures of Sharpe ratio covering the whole trading period (e.g. the first measure obtained from profits from week 1 to week 50, the second one from week 2 to week 51 and so on).

However, being the results presented in Table 5.4 too synthetic, a further analysis has been conducted: the Sharpe ratio has been brake down by calculating it over a rolling window containing 50 weekly profits and moving forward a single week at every step. The aim of this decomposition refers to the aim to check whether the synthetic value of profits adjusted for the risk suffered have been the result of an overall mediocre performance over the whole trading period or the average of positive and negative periods. Figure 5.4 gives credit to the second hypothesis: it is possible to observe superior performance by the strategy in terms of profits adjusted for the risk during a consistent time period (i.e. during the second and

third trading year); during this period indeed, the Sharpe ratio records values higher than 1 most of the times. Such result could not appear impressive by many people, but comparing them with the literature and keeping in mind that they have been achieve in extremely recent time periods, they can be considered extremely valuable. In addition, the graph shows how the real under-performances of the strategy have been registered around week 50 and after week 200 during which the Sharpe ratio has always been negative and where it even dropped up to values lower than -1.

5.2.2 Strategy Decomposition

To investigate the causes of the observed flaws in the strategy, since the final result of the investment approach was achieved merging 5 different portfolios created thanks to the predictions of just as much LSTM models, the final profit of the investment strategy has been decomposed into its elementary components.

From this decomposition (Figure 5.5), in terms of profitability, surprisingly it turns out that more than 50% of the total profits of the strategy are produced by the portfolio $h = 1$. It is less surprising, instead, the ranking among the different models: as in the case of data science metrics, models forecasting at shorter horizon are found to achieve better performances. In fact, $h = 2$ and $h = 3$ models reaches positive profits too, while $h = 4$ and $h = 5$ models end the 6 years trading period with small but still negative profits.

In any case the most important finding, always from Figure 5.5, regards the evidence that the portfolio built upon prediction of LSTM at $t + 1$ results to be most stable among the five portfolios. All portfolios, in fact, are able to create profits in specific time periods but only the $h = 1$ portfolio manages to avoid large losses able to erode all the positive results previously achieved. From Table 5.5, that decomposes the results of the different portfolios forming the modular strategy in each of the 12 different semesters of the trading period, it can be noted how the largest loss recorded during a semester for $h = 1$ model amounts to -0.075 while the ones recorded by all the other four portfolios are all much larger: -0.130 for $h = 2$,

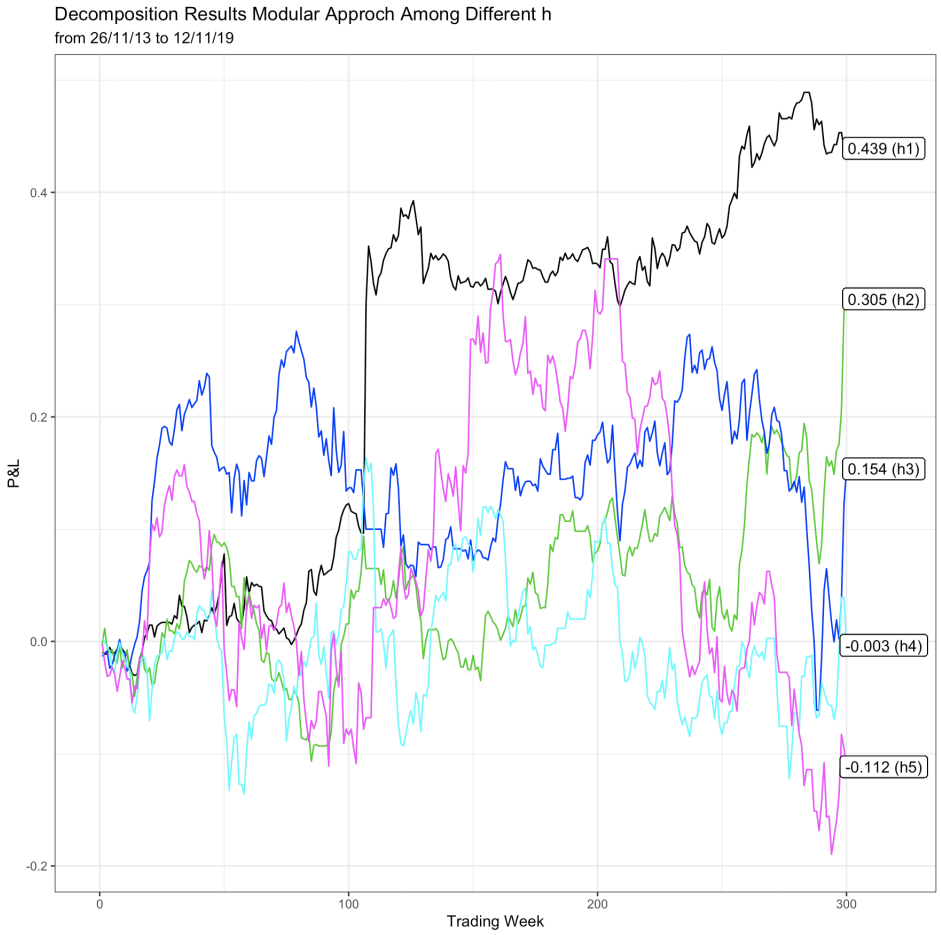

Figure 5.5: Comparison of profits and losses deriving from the modular approach and from the three different benchmarks described in Chapter 5.

-0.121 for $h = 3$, -0.141 for $h = 4$ and -0.295 for $h = 5$.

Regarding the under-performances periods in terms of profitability previously highlighted for the global strategy (i.e. during the second, the fifth and the sixth years of trading), it emerges from both the previous table and graph how they seem particularly correlated with the trend of the $h = 5$ portfolio: it records large losses just before week 50 and during the last two years of trading (i.e. the same period in which the "dynamic Sharpe ratio" showed the worst results). Moreover, even in the case of $h = 5$ portfolio, it is important to observe how profitability does not seem particularly related neither to G-score measure nor to accuracy in Table C.7; such portfolio registered indeed neutral profits both during a positive period where

Table 5.5: Table representing the comparison of half-yearly P&L achieved by all the portfolios forming the "modular strategy" presented in Chapter 4 and by the overall modular trading strategy during the 12 semesters composing whole trading period. The profit of the modular strategy, for construction, is given by the sum of profits from all models at different h.

Semester	h=1	h=2	h=3	h=4	h=5	Modular strategy
1	0,017	0,008	0,190	-0,010	0,098	0,302
2	0,061	0,080	-0,035	-0,058	-0,105	-0,056
3	-0,075	-0,130	0,103	-0,021	0,033	-0,090
4	0,120	0,057	-0,121	0,169	-0,109	0,115
5	0,264	0,041	-0,069	-0,141	0,148	0,242
6	-0,071	-0,069	0,004	0,123	0,204	0,192
7	0,017	0,056	0,068	-0,067	-0,035	0,040
8	0,003	0,037	0,044	0,093	0,060	0,237
9	0,005	0,031	-0,028	-0,133	-0,053	-0,178
10	0,018	-0,086	0,049	-0,038	-0,295	-0,352
11	0,106	0,146	-0,054	0,007	0,026	0,232
12	-0,026	0,134	0,002	0,072	-0,084	0,097

LSTM reached 50.9% and a negative one with 22.4% in terms of G-score.

In addition to such observations, the strategy decomposition also allows to analyze the performance peaks and depressions recorded by portfolios during particular weeks (e.g. the peak by portfolio $h = 1$ during week 107); in fact, they stand out enough to deserve more accurate analyses. At first, during the analysis it has been controlled if they could have been generated by some extraordinary event: for instance, the surprising profit achieved by the portfolio $h = 1$ during the fifth semester is attributable to the profit of 0.2055 obtained during week 107; during such week, portfolio $h = 1$ was indeed short on a single company, Freeport-McMoRan Inc., whose stock price passed from 5.26$ to 4.19$ losing 20.3% of its value in a single day. However, this was the only observed case in which a peak/depression could have been explained by such an abnormal situation; for example, regarding both the profits recorded by $h = 2$ and $h = 3$ during the penultimate week of trading, they are not affected by any particular event but simply by an accuracy of 100% reached by the model for all the 10 stocks within the portfolio. In the same way,

looking for some singular event able to explain all other peaks and depressions registered by the different portfolios (i.e. the initial peak of $h = 3$, the instantaneous crash by $h = 5$ before week 250 etc.), no clear evidence has appeared so much to conclude that these results may depends exclusively on the variability in prediction accuracy of the different models. As further proof supporting such hypothesis, it must be observed how the spikes in profitability registered by the different portfolios are not aligned; if they were aligned for a specific time period, this would have suggested some singular market behaviour or event recognised by models indeed.

Secondly, it has been checked whether such peaks and depressions could have been generated by excessively correlated stocks selected by LSTM during a particular week or by a low numerousness of alternatively the long or short leg of the portfolio [2]. At first, it has been created Table D.1 providing a synthetic view about numerousness and correlation among different portfolios at different h. However, this synthetic view does not highlight any significant abnormal behaviour from any of the examined portfolios: the average number of stocks selected every week and the average correlation do not appear significantly different from all the ones in all other horizons. Regarding industries, it is striking to observe how stocks belonging to the tech sector have been the least chosen by models in all horizon. In any case, it has not been possible to spot any meaningful and evident correlation between industries preference and other explanatory variable able to explain such behaviour from the models.

Therefore, since synthetic measures did not give enough information, the evolution over time of the size and correlation of the decomposed strategy has been examined more in depth for five elementary portfolio. For all horizons indeed, both correlation and numerousness of each single portfolio have been graphically overlapped with its P&L so as to spot potential recurrent patterns. Figure 5.6 shows

[2]In fact, it must be remembered that, despite the limit imposed with respect to the maximum size of the portfolio equal to 10 (5 for the long side and 5 for the short side), if the model classified less than 5 shares for a class during a specific week, the portfolio for that week would have been less numerous.

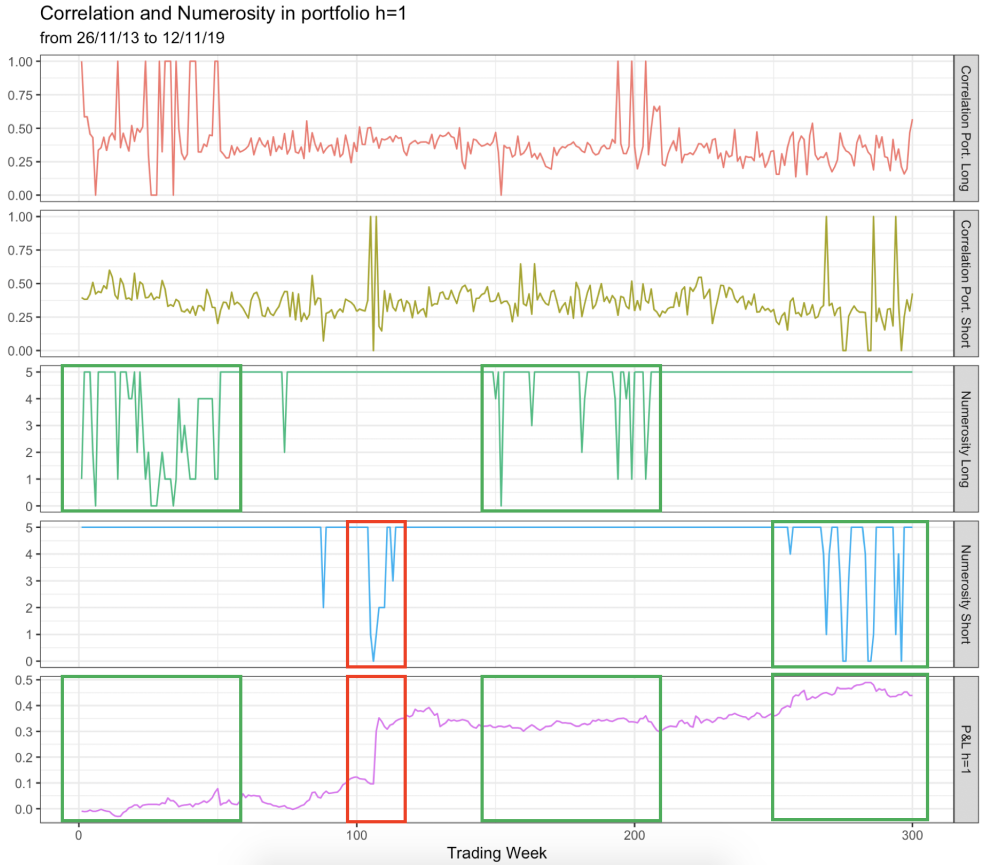

Figure 5.6: Comparison between the evolution of numerosness and correlation of stocks within both the long and short legs of the portfolio $h = 1$ and the profitability of the portfolio itself. Obviously, correlation is equal to 1 when in the portfolios is present a single stock and to 0 when none stock is in it. Red boxes highlight strong results possibly driven by low numerousness of the portfolio while green boxes highlight periods in which low numerousness have clearly not brought to an increase in P&L volatility.

the visual analysis performed for the portfolio $h = 1$ (in appenix it is possible to find the ones for all other h from Figure D.1 to D.4).

As can be noted from the figure, correlation appears constant throughout the whole trading period (excluding weeks in which it is equal to 1 that represent a particular case ascribable to the actual size of the portfolios). This evidence therefore does not support the hypothesis for which particularly positive or negative results in terms of profitability from the strategy could be ascribed to highly correlated stocks present in the portfolio during a particular week. This result can be considered undoubtedly positive since highly correlated portfolios comes with high associated

risk. In fact, imagine a week in which LSTM goes long on 4 stocks belonging to the same industry affected by a particularly positive event in the meantime; evidently, the strategy would incredibly benefit from such investment. Anyhow, such behaviour would not be positive for the model, since in the same way such a highly correlated portfolio could also have lost the same amount in the case in which the industry was affected by a particularly negative event.

Rather, from all graphs it appears how sometimes periods of high volatility in terms of P&L could have been influenced by the size of the portfolios: during periods in which LSTM has not been able to individuate at least 5 different stocks to select for alternatively the long or short side of the portfolio, its P&L registers some spike/depression. It could be objected to this statement that the smaller actual size of the portfolios directly influences their correlation that in turn affects possible spikes or depressions, so that high correlation is the real cause for them. And this would be true. Anyhow, it must be clear that from a managerial perspective these two causes are deeply different. For instance, a problem related to the size of the portfolio could be easily solved increasing the number of the stocks in the database at the expense of longer training time: in this way the model would have higher chances to find at least five stocks belonging to the less populous class for that week. A solution which, instead, would not be so easy to find in the case of a model systematically creating highly correlated portfolio even when fulfilling the maximum allowed dimension.

Despite this, results do not provide a clear evidence of a systematic component in the negative relation between numerousness of portfolios and their volatility: in the image for $h = 1$, red boxes highlight strong results possibly driven by low numerousness, while green boxes highlight periods in which low numerousness does not clearly bring to an increase in P&L volatility; as it can be seen, it is possible to find both of them in it. In conclusion, it cannot be affirmed overall that situations of low numerousness of portfolios have necessarily been the cause of such high levels of volatility of the global strategy. So that, probably, the causes of the high volatility

are way more difficult to be identified.

Looking at Figure D.1, D.2, D.3 and D.4 for all other horizons, the same considerations can be drawn: the correlation is constant throughout the whole trading period and sometimes numerousness is found to be low when the volatility of the portfolio is high.

Finally, looking at risk performance measures (Table 5.6), at first they confirm the stability of $h = 1$ portfolio: the maximum drawdown is really low compared to other strategy and, at the same time, omega is quite high compared to other models. Moreover, such measures highlight how the total result obtained by the

Table 5.6: Risk performance of elementary portfolios forming the trading strategy accounted for the "modular approach" presented in Chapter 4. "Avg.ret" refers to the mean of weekly profits achieved by each portfolio, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "Max Drawdown" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

	Avg.Ret	**Std.Dev**	**Sharpe**	**Max Drawdown**	**Omega**	**VaR**
h=1	0,0015	0,0160	0,6478	0,0932	1,4555	-0,0356
h=2	0,0010	0,0155	0,4636	0,1865	1,2022	-0,0350
h=3	0,0005	0,0192	0,1886	0,3111	1,0775	-0,0441
h=4	0,0000	0,0199	-0,0041	0,2723	0,9983	-0,0462
h=5	-0,0004	0,0254	-0,1036	0,4340	0,9597	-0,0593

global strategy is worse than the results of some of its elementary components. The combination of the five different portfolios, indeed, worsen the performance of the two single best portfolios that are $h = 1$ and $h = 2$; this is particularly clear examining standard deviation and max. drawdown.

Especially from these last observations, the "modular approach" is thus revealed as a useful approach to highlight the ability of LSTM model to forecast stock market movement at different horizons rather than as a proper approach to create profitable investment strategies.

Anyhow, having the same purpose as for the global strategy, the Sharpe ratio has been decomposed and calculated over a rolling window even for all the five

elementary portfolios. As it can be noted from Figure 5.7, the only portfolio that

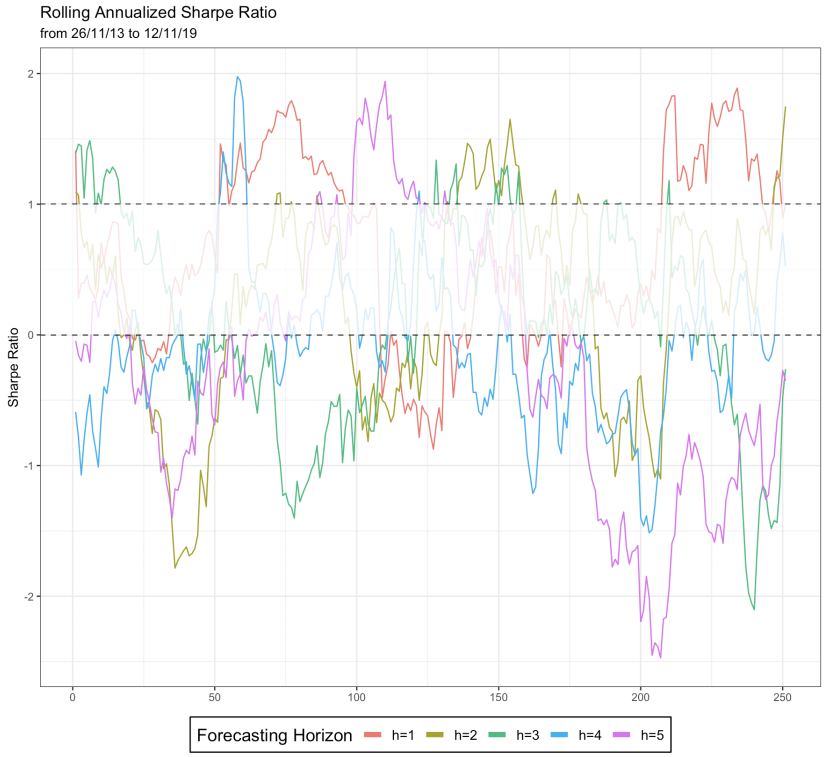

Figure 5.7: Graphic representation of a rolling annualised Sharpe ratio calculated for profits obtained from the five portfolios forming the trading strategy originated with the "modular approach" presented in Chapter 4. The rolling window over which the Sharpe ratio has been calculated is formed by observations coming from 50 trading weeks shifting forward one week ahead at every step. In such a framework, in the graph are represented 251 different measures of Sharpe ratio covering the whole trading period (e.g. the first measure obtained from profits from week 1 to week 50, the second one from week 2 to week 51 and so on). For an increase readability, results between 0 and +1 have been partially darkened.

in two different period consistently reaches a Sharpe ratio higher than 1 is $h = 1$ (i.e. in the second and sixth year of trading). For all other portfolios, their presence in the "positive zone" appears sporadic and not consistent. On the opposite, it can be noted consistent disappointing performances from all the other four portfolios, especially for $h = 5$ during the last two years of training, $h = 4$ during the second half of the trading period and $h = 3$ during the second and last year of trading.

Benchmarking

In conclusion, the global investment strategy has been compared with specific benchmarks so as to effectively understand the ability of LSTM models.

Due to the decision about the construction of dollar-neutral portfolios, the presented strategy could not be compared with simple buy-hold strategy, for instance with a long-only strategy holding all the 44 stocks within the portfolio for the whole trading period or even with the S&P500 index itself. Such strategies, in fact, would have required a positive initial investment and would have implicitly considered a compound interest (i.e. the reinvestment of all the profit made from time $t = 0$ during the whole trading period).

For this reason, to evaluate the results just presented, three different dollar-neutral long-short benchmark strategies have been arbitrarily created as follows:

Benchmark(1) For each of the five different forecasting horizons, 100 random prediction models have been created and the P&L deriving from their forecasts have been averaged. Portfolios at different horizons have been created, exactly in the same way as in the main strategy, by picking maximum 5 stocks long and 5 short. The only difference from the main strategy refers to the weighting scheme of the stocks within the different portfolios that has been selected to subdivide the total investment equally among all the different stocks (referring to (4.11), $w_{i,h}^{l(s)} = \frac{1}{5} = 0.2$).

Benchmark(2) Regarding the second benchmark, it has been adopted a short-term reversal strategy: among all 44 stocks in the database, it has been taken a long position on those ones with a negative return during the past five days and a short position on those ones with a positive return during the past five days. Differently from the original strategy, this benchmark accounts for only a single long-short portfolio with one-week horizon created on the first day of the

week and closed after 5 trading days (i.e. one week). Finally, the weighting scheme has been selected to subdivide the investment equally among shares selected both for the long and short leg of the portfolio.

Benchmark(3) For the third benchmark, as example of a non-frequent re-balancing strategy, the prediction of the $h = 5$ model have been employed semiannually so as to create a dollar-neutral long-short portfolio with an investment horizon of six month. Also in this case, as in Benchmark 2, all the 44 stocks have entered in the portfolio: the ones predicted to belong in the class "0" have been equally subdivided in both classes "-1" and "1". Finally, the weighting scheme has been selected to subdivide the investment equally among stocks both on long and short legs of the portfolio.

Figure 5.8 graphically compares the profits of the original trading strategy with profits of the three different benchmarks. From the image, it is visible that the number of periods in which the main investment strategy performs better than its benchmarks is very high. Such a result is a valuable proof to certify that the performance of our trading strategy does not derive from particular positive market conditions or due to some favourable coincidence, but due to the inner ability of LSTM to recognise specific patterns in price time-series. If it were not so, one would not recognize such a clear over-performance of the strategy in question.

In Table 5.7 it is even possible to directly compare P&L of each strategy during the 12 semesters in which the whole trading period can be subdivided. From both the table and the graph, it is visible how both Benchmark 2 and 3 manages to create profits only in very limited time periods, oscillating for the rest of the time around the neutral line of the x-coordinate; benchmark 1, instead, results neither to create profits nor to create losses during the whole trading period. Given such behaviours, it would have been difficult to extract some particularly useful insights to further

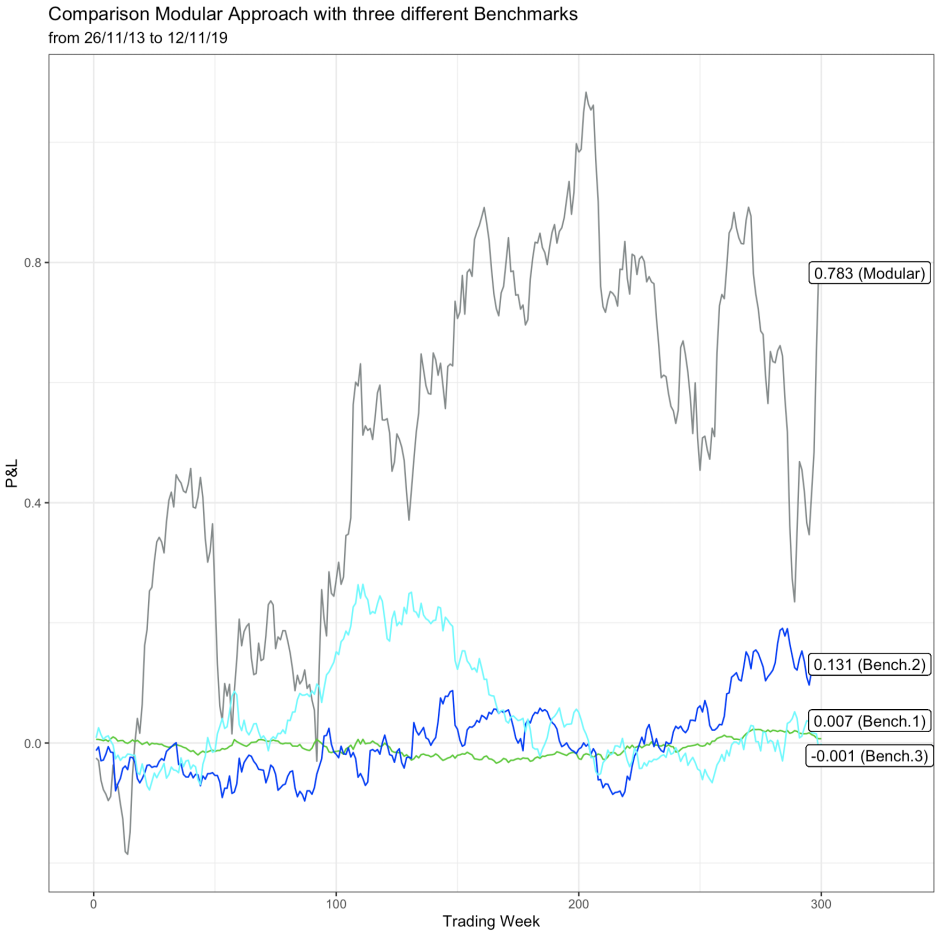

Figure 5.8: Comparison of profits and losses deriving from the modular approach and from the three different benchmarks described in Chapter 5.

comment performances from risk measures calculated over the whole trading period. This is the reason why Table 5.8 provides just the comparisons between the global strategy and benchmark 3 from week 50 to week 150 and between the global strategy and benchmark 2 from week 200 to week 300. These are the only two periods in which benchmarking strategies appear able to provide some positive result indeed.

Both the comparisons highlight once more how the original strategy, even if presenting a clear superiority in terms of profitability over its benchmarks, suffers from very high levels of volatility. Standard deviation, max. drawdown and value at risk for the period under scrutiny clearly highlight it.

Table 5.7: Table representing the comparison between half-yearly P&L achieved by the original strategy presented in Chapter 4 and half-yearly P&L obtained by the three benchmark models created along the 12 semesters composing the whole trading period.

Semester	Modular	Benchmark 1	Benchmark 2	Benchmark 3
1	0,302	-0,001	-0,039	-0,050
2	-0,056	-0,013	-0,015	0,037
3	-0,090	0,018	0,002	0,020
4	0,115	-0,018	0,033	0,145
5	0,242	-0,001	0,005	0,043
6	0,192	-0,005	0,037	-0,073
7	0,040	-0,005	-0,023	-0,085
8	0,237	0,000	-0,010	0,014
9	-0,178	0,022	0,008	-0,093
10	-0,352	0,000	0,064	-0,004
11	0,232	0,027	0,076	0,033
12	0,097	-0,015	-0,007	0,012

5.3 Robustness Analysis

In this section it will be presented the conclusive part of the experimental work where it has been decided to focus the attention on the robustness of the whole trading system in relation to some change in decision variables (i.e. weighting scheme 5.3.1, portfolio dimension 5.3.2, investment horizon 5.3.3 and input features set composition 5.3.4). Such a robustness analysis has been pursued to direct its effort in discovering useful observations from a managerial perspective regarding a decision making process dealing with the construction of an effective LSTM trading system.

5.3.1 Weighting Scheme

As explained in chapter 4, the results obtained by the main strategy have been achieved through the construction of portfolios adopting a weighting scheme proportional to the confidence of each LSTM model in assigning a given stock to a specific class (expressed by the output of the softmax function). To test the validity of this choice, such results have been compared with the results obtained by a similar

Table 5.8: Comparison of risk performance by the original trading strategy based on the "modular approach" presented in Chapter 4 and its benchmarks. "Avg.ret" refers to the mean of weekly profits achieved by the strategy, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "MDD" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

	Avg.Ret	**Std.Dev**	**Sharpe**	**MDD**	**Omega**	**VaR**
Modular (50-150)	0,0033	0,0522	0,4586	0,3570	1,1952	-0,1174
Bench.3 (50-150)	0,0014	0,0176	0,5499	0,1389	1,2253	-0,0392
Modular (200-300)	-0,0021	0,0556	-0,2702	0,6161	0,9000	-0,1307
Bench.2 (200-300)	0,0004	0,0152	0,2033	0,1668	1,0820	-0,0348

strategy selecting the 5 top and flop stocks subdividing the total invested quantity of 1$ equally among them (e.g. with five stocks on each leg of the portfolio, the individual invested quantity $w_{i,h}^{l(s)}$ from Algorithm 5 is 0.2$ per each of them).

The results from both the strategies are plotted in Figure 5.9: it is visible how the strategy providing for equal-weighted portfolios manages to achieve higher profits than the original one providing for portfolios with a weighting scheme proportional to the confidence of LSTM's predictions. Such result, in the beginning, might be unexpected: a system that takes advantage of a partial level of information performs better than a similar one that employs it all.

However, in order to investigate the reasons of this outcome, at first, it is important to remember how the confidence expressed by the softmax output function in classification networks is related to the probability that a given stock on a given day belongs to a specific class according to its expected price trend. There are no considerations from the model regarding the magnitude of such trends indeed, and LSTM networks in this experiment are not an exception: they are not equipped to analyze also the volatility of the different stocks within the dataset.

In sight of this, it is possible to state that, normally, an equal-weighted strategy is

Figure 5.9: Comparison of profits and losses deriving from the modular approach adopting the weighting scheme described in Chapter 5 and the same modular approach creating equal weighted portfolios.

expected to cause more volatile stocks to be less preferred in terms of impact on the total composition of the portfolio compared to a confidence weighted strategy. This due to a LSTM tendency to be less confident on forecasts for more volatile stocks (i.e. lower confidence is equivalent to a smaller output by the softmax function that in turn diminishes $w_{i,h}^{l(s)}$ from Algorithm 5).

Moreover, even if obvious, when a model correctly predicts the price movement for more volatile stocks, this usually results in higher profits than the ones obtained from a correct prediction of less volatile ones.

So, from these observations, it is straightforward to conclude it is possible to face two opposite situations when comparing these different weighting schemes: model's forecasts on more volatile stocks are less accurate than the ones for less volatile stocks

to the point that a confidence weighted strategy performs better than an equally-weighted one in terms of profitability. Anyhow, clearly, it could easily happens the opposite; namely, model's forecasts on more volatile stocks are less accurate than the ones for less volatile stocks thus resulting in higher losses for the confidence weighted strategy. This second case would explain why, in this experiment, the "equally weighted" portfolios reached a higher profitability than the "confidence weighted" ones. However, as anticipated, from such explanation derives that the over-performance of one method over the other in terms of profitability cannot be systematic: it depends on whether the correct predictions by the model have been concentrated more on volatile stocks or not; unfortunately, this aspect cannot be known in advance but it only depends on inner characteristics of the model itself and of the time-series.

The evidence that the over-performance is not systematic clearly emerges also from our comparison: as it can be seen in Figure 5.10, over-performance in terms of profitability by "equally weighted" portfolios is not methodical during the whole trading period. From such observations, it is thus not possible to declare that a strategy providing for "equally weighted" portfolios always return higher profits than the same one providing for "confidence weighted" portfolios.

So, from a managerial point of view, the main conclusion regarding the weighting scheme to be adopted for the investment strategy is that it is not possible to make an optimal choice among the two weighting scheme presented in terms of profitability since profits coming from both of them do no follow a predictable pattern.

Table 5.9: Risk performance of the trading strategy based on the "modular approach" presented in Chapter 4 with "equally weighted" portfolios. "Avg.ret" refers to the mean of weekly profits achieved by the strategy, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "Max Drawdown" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

Avg.Ret	**Std.Dev**	**Sharpe Ratio**	**Max Drawdown**	**Omega**	**VaR**
0,0032	0,0484	0,4620	0,6000	1,1939	-0,1093

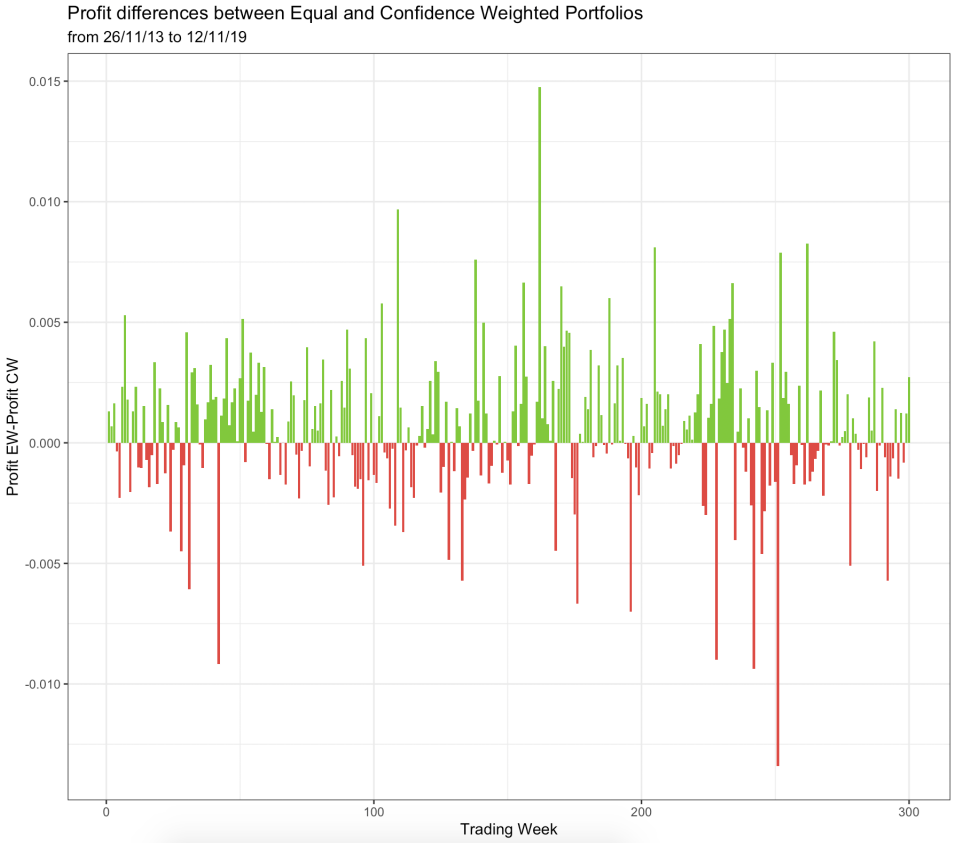

Figure 5.10: Weekly difference in profits between "confidence weighted" portfolio and "equally weighted" portfolio.

The only highly predictable implication to look at, that comes from preferring an "equally weighted" portfolio to a "confidence weighted" one is, as explained, the higher propensity of the former to allocate a higher part of the total investment to more volatile stocks. An example of this is provided by Table 5.10 that shows the composition of the five different portfolios during week 162 (i.e. the one resulting in the largest positive difference between $profitEW$ and $profitCW$ in Figure 5.10) for both the confidence weighted and the equal weighted strategy. In the last column of the table it is presented an average of the weekly volatility from the past 10 weeks of the stocks selected to enter in the portfolio weighted through the percentage composition of the portfolios both on long and short side. Be careful: this measure is not the volatility of the portfolio, but simply a synthetic measure to prove how

Table 5.10: Table representing the distribution of the total invested quantity (1$) among stocks forming both the long and short legs of the 5 portfolios for both the weighting schemes presented during week 162. "Volatitlity$_{10w}$" refers to the weekly standard deviation of the specific stock during the past ten weeks while the column "Load" refers to the weighting average of percentages and volatility. Pay attention: "Load" measure does not express the variance of the portfolios, it wants simply to express the load of both weighting schemes on volatile stocks.

			Stock1	**Stock2**	**Stock3**	**Stock4**	**Stock5**	**Load**
h=1	L	% CW port.	19,16	18,29	17,74	25,41	19,4	1,3938
		% EW port.	20	20	20	20	20	1,4440
		Volatility$_{10w}$	0,85	2,56	1,51	0,81	1,49	
	S	% CW port.	19,66	19,64	19,99	19,92	20,79	1,8538
		% EW port.	20	20	20	20	20	1,8540
		Volatility$_{10w}$	2,31	1,77	1,97	1,29	1,93	
h=2	L	% CW port.	19,8	20,6	19,2	20,6	19,8	7,9953
		% EW port.	20	20	20	20	20	8,0500
		Volatility$_{10w}$	0,82	2,24	1,99	2,48	32,72	
	S	% CW port.	19,64	19,9	20,36	20,5	19,6	1,8591
		% EW port.	20	20	20	20	20	1,8580
		Volatility$_{10w}$	2,78	1,68	0,82	2,87	1,14	
h=3	L	% CW port.	46,25	26,84	26,9			1,7884
		% EW port.	33,3	33,3	33,3			1,6450
		Volatility$_{10w}$	2,38	1,61	0,95			
	S	% CW port.	20,5	19,77	19,81	20,1	19,8	1,9268
		% EW port.	20	20	20	20	20	1,9320
		Volatility$_{10w}$	0,92	3,71	1,37	2,87	0,79	
h=4	L	% CW port.	100					2,3200
		% EW port.	100					2,3200
		Volatility$_{10w}$	2,32					
	S	% CW port.	19,7	19,8	21	19,88	19,62	12,4926
		% EW port.	20	20	20	20	20	12,5740
		Volatility$_{10w}$	1,53	33,1	2,45	24,18	1,61	
h=5	L	% CW port.	26,35	23,55	22,77	7,83	19,5	1,4086
		% EW port.	20	20	20	20	20	1,4320
		Volatility$_{10w}$	1,24	0,83	2,25	1,54	1,3	
	S	% CW port.	20,2	19,8	19,9	19,9	20,2	7,5399
		% EW port.	20	20	20	20	20	7,5700
		Volatility$_{10w}$	2,81	2,43	2,23	29,8	0,58	

"equally weighted" portfolios load more on volatile stocks. From the column it emerges how only in a single case over ten the synthetic measure results to be higher in the case of the "confidence weighted" portfolio.

Finally, in terms of the overall strategy performance, looking at risk metrics, it can be noted how results by "equally weighted" strategy do not differ so much from the original ones (Table 5.9). Only Sharpe ratio appears influenced by higher average returns in the former case.

5.3.2 Portfolio Dimension

As reported in Chapter 4, for each portfolio composing the main strategy, the maximum number of stocks selected to enter both in the long and short side has been set to five: basing on the ranking provided by the softmax output function of each LSTM model, each week the top five stocks for both classes "1" and "-1" have been selected to form the different portfolios. Clearly, in the case in which the model for that specific week had predicted less than five stocks for one of the two classes, they would have been all selected; finally, if no stocks had been predicted to belong to a particular class, for that week no portfolio would have been formed.

This section wants to investigate the robustness of the strategy subject to changes regarding the maximum portfolio dimension parameter. In particular, the comparison that will be presented regards the results provided by the original strategy (i.e. max. portfolio dimension equal to 5) and the ones by four similar strategies having a value for maximum portfolio dimension respectively equal to 1, 3, 10 and 15.

At first, risk performance results for all the possible values tested are presented in Table 5.11. The first important observation that stands out from the different measures in table regards the ability of the original strategy to reach the best trade-off between profitability and risk. Moreover, as might be expected, data also show how diversification in portfolios reduces the risk of the related strategy: standard deviation, max. drawdown and value at risk all diminished with increasing number of stocks within the portfolio. This last observation becomes even more evident when

Table 5.11: Risk performance of the trading strategy based on the "modular approach" presented in Chapter 4 with different values of maximum portfolio dimension ("P.Dim"). "Avg.ret" refers to the mean of weekly profits achieved by the strategy, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "Max Drawdown" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

P.Dim	**Avg.Ret**	**Std.Dev**	**Sharpe**	**Max Drawdown**	**Omega**	**VaR**
1	0,0014	0,0733	0,1336	0,7190	1,0517	-0,1688
3	0,0000	0,0545	-0,0050	0,7547	0,9981	-0,1265
5	0,0026	0,0487	0,3788	0,6161	1,1565	-0,1105
10	0,0013	0,0432	0,2194	0,5986	1,0892	-0,0991
15	0,0014	0,0404	0,2365	0,5627	1,0980	-0,0924

looking at boxplots in Figure 5.11 that present the distribution of weekly returns for each different value of maximum portfolio dimension tested.

Finally, looking at the P&L behaviour over the whole trading period from Figure 5.12 three important focal point for eventual decision processes, even if not particularly unexpected, can be highlighted:

- All the P&L follow the same trend, even if the strategy accounting for a maximum portfolio dimension equal to 1 suffers from a tremendous exposure on the variability in accuracy of LSTM predictions. It can be noted how this last strategy creates notable losses during the first two years of trading if compared to the ones created by all the other strategies; but at the same time it can be noted how it easily recovers from them in very little time. As previously specified, this result highlight how selecting only a single stock associated with the highest softmax output value, choice that in theory should bring to achieve the highest accuracy, does not necessarily brings positive results results from a financial perspective; rather it exacerbates the volatility already observed in previous section.

- Strategies accounting for a high number of stocks within portfolios suffer from the opposite problem. The large number of positions taken for a very high

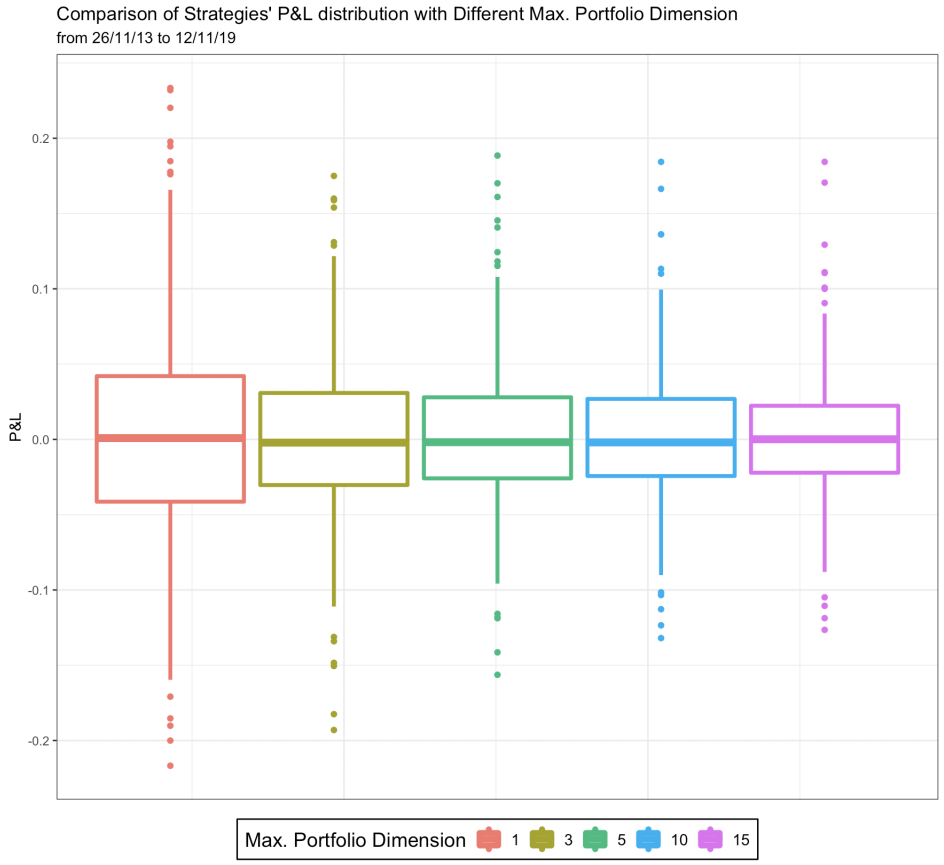

Figure 5.11: Boxplots showing the distribution of weekly P&L from the trading strategy based on the "modular approach" presented in Chapter 4 with different values of maximum portfolio dimension.

portion of the total number of stocks within the dataset causes the inclusion in portfolios of those ones on which the decision by the model has been taken almost randomly (i.e. softmax output just slightly bigger than 0.33). In some days indeed, it is not unusual the algorithm cannot find any recognised pattern for a specific stock so as to make a poor prediction for it. Clearly, the larger the dataset, the lower the impact of this issue on top/flop n stocks selected by the system. At the same time, startegies with high values of "maximum portfolio dimension" highlight how a proper numerousness of portfolios is necessary to lower the unavoidable volatility deriving from LSTMs predictions clearly underlined in the previous section.

Figure 5.12: P&L deriving from "modular approaches" with different values of maximum portfolio dimension.

- The highest profits are provided by the original strategy with a maximum portfolio dimension of 5. This result suggests how, depending on the dimension of the initial database (i.e. in this experiment equals to 44 stocks), it is necessary to balance the maximum portfolio dimension so as to increase the diversification thus lowering volatility but at the same time avoiding the inclusion within portfolios of stocks uncertainly classified by models. For instance, in the case of maximum portfolio dimension equal to 15, every week each portfolio selected the 68% of stocks under scrutiny. In such situation the discriminatory power of the LSTM is particularly limited by the fact that the strategy takes a position on almost all available stocks. In the case of maximum portfolio dimension equal to 5, instead, this percentage becomes

23%.

5.3.3 One-Day Ahead Investment Strategy

Due to the fact that in all the experiments conducted, portfolios $h = 1$ always performed better than all other in terms of financial performance, it has been decided to test the a trading system exclusively predicting at $t + 1$ for all the 1500 days composing the trading period. Thus, on each day each day a portfolio containing the top and flop 5 stocks has been created having an investment horizon of 1 day. As for the original strategy, during the days in which the model did not predict any stock in alternatively class "1" or "-1" no portfolio has been created.

Figure 5.13 shows the P&L of such strategy presenting both a weighting scheme proportional to the confidence of the model's predictions and an equal weighting scheme. In terms of profitability over the whole trading period, the strategy adopting a confidence weighting scheme clearly outperforms the original modular strategy. Moreover, in the case of the strategy adopting an equal weighting scheme, it reaches basically the same results of its modular strategy. This result is really important since it shows the clear superiority of LSTM predicting at shorter horizons and the lack of need to train in every training window 5 different LSTM models each predicting at a different horizon. All the strategies are, in fact, evidently comparable since all of them require the system to take the same number of positions every day; the only difference lies in the time horizon for which they do it.

Looking at risk metrics (Figure 5.12) and comparing them with results obtained by the original modular strategy, it is possible to understand how the most important improvement achieved by the strategies investing one-day ahead regards the lower level of risk suffered by them. The maximum drawdown suffered by the orginal modular strategy result to be three times higher than the same strategy investing only at $t + 1$. Finally, Sharpe ratio increases considerably compared to the modular approach implementing both a confidence weighted scheme and an equal weighted scheme.

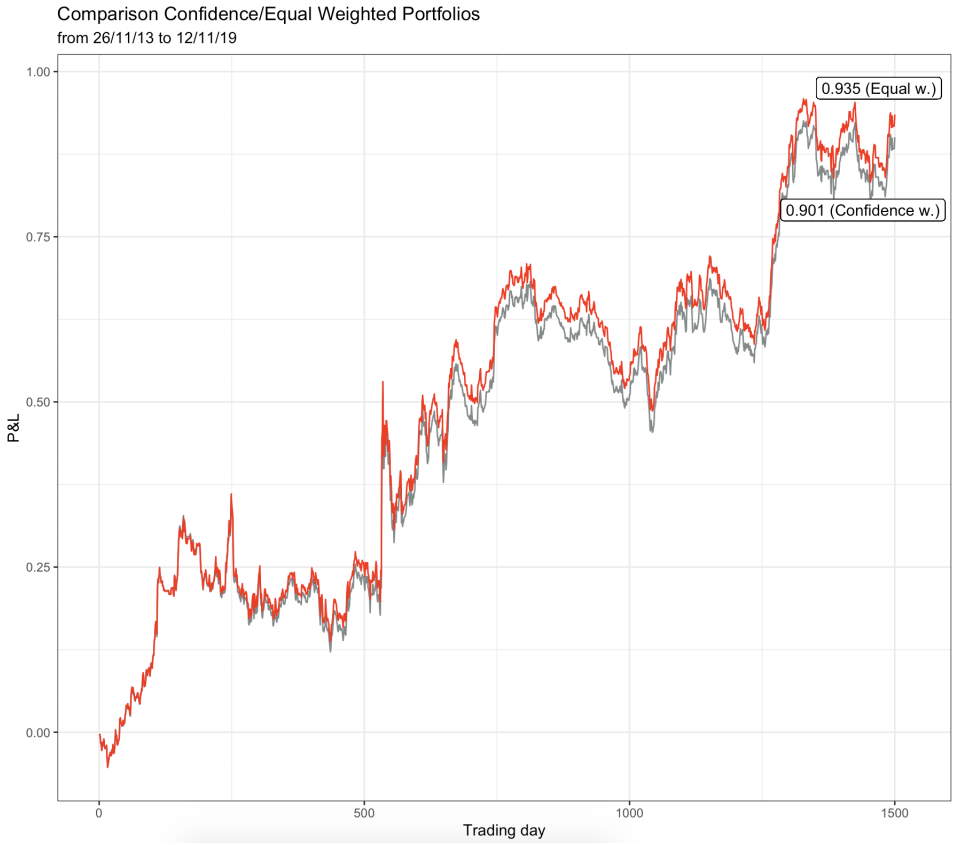

Figure 5.13: Comparison of profits and losses deriving from LSTM predicting at $t+1$ both adopting a weighting scheme proportional to the output of the softmax activation function and a scheme providing for equal weights for all the stocks within each side of the portfolio.

From a managerial perspective thus, evidences seem to recommend the use of LSTM networks for short investment horizons. However, due to the limited comparison performed, it is recommended to make further examinations; many other networks settings, input variables, training parameters can be employed indeed.

5.3.4 Extended Input Feature Set

As conclusive analysis, the advantages derived from the introduction into the input features set of technical indicators and macroeconomic variables have been tested. To make so, due to time constraint and computational limitations, it has just been trained and tested LSTM networks predicting at $t+1$, taking advantage only of daily returns time-series. Technical indicators and macroeconomic variables have

Table 5.12: Risk performance of the "one-day ahead trading strategy". "Avg.ret" refers to the mean of weekly profits achieved by the strategy, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "Max Drawdown" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

W.Scheme	**Avg.Ret**	**Std.Dev**	**Sharpe**	**Max Drawd.**	**Omega**	**VaR**
Confidence	0,0030	0,0319	0,6661	0.2001	1.3674	-0.0710
Equal	0,0031	0,0318	0,6928	0,1960	1,3820	-0,0708

thus been eliminated from the original input feature set. Moreover, due to the same constraints, even the validation phase has been dropped: as number of hidden nodes populating the hidden layer ,the same parameter selected for the models employing as input the entire dataset have been used.

Surprisingly from Table E.1 that presents data science results it can be noted how, especially for the first two training windows, the models trained with the simpler dataset manages to achieve higher accuracy. In any case, when comparing results in terms of G-score, the simpler model is found to over-perform the complex one exclusively for the first two training windows. In all the other ones, it manages to reach very poor performances due to a clear bias towards class "1". This bias is important since it suggests the following hypothesis: the higher level of information present in the original database makes models more robust from biases thus making them less prone to oversimplify the reality and, as it emerges from results, to "over-predict" observations for a certain class. Unfortunately, such hypothesis could not be further verified.

Looking, instead, at the financial performance of the simplified model (Figure 5.14), and comparing it with the one of the LSTM trained with the original dataset, it stands out how data science results are not particularly informative for the under-statement of the ability of the model in generating profits over the time. For instance, it is not possible to observe higher profits for the strategy with the smaller dataset during the first two years as G-score measures would suggest. Besides, during the

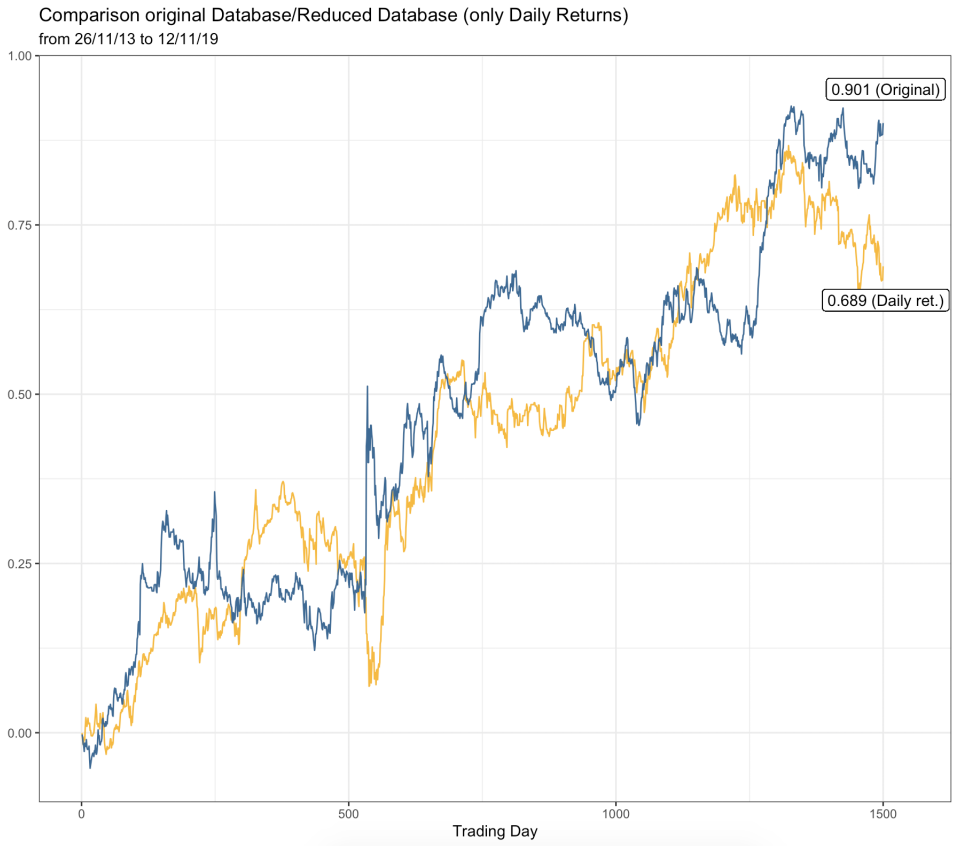

Figure 5.14: Comparison of profits and losses deriving from LSTM predicting at $h = 1$ trained both with the original database and with a database containing only daily returns. Both portfolios adopt a weighting scheme proportional to the output of the softmax activation function of the output layer.

last three years it would be expected a drop in performance for the strategy adopting the simpler dataset that has not occurred. Finally, from the graph the model with the original dataset is observed to obtain higher profits. Thus, from these observations the advantage in economic terms deriving from employing technical indicators and macroeconomic variables does not result clear although in financial literature, especially the former indicators, they are extensively employed.

Even in terms of risk metrics (Table 5.13), the superiority of LSTM trained with the original dataset is not evident. In conclusion, the analysis conducted has not provided strong evidence of over-performance from a financial perspective due to the introduction of technical indicators and macroeconomic variables within input features.

Table 5.13: Risk performance of the "one-day ahead trading strategy" obtained by LSTM networks trained only with daily returns. "Avg.ret" refers to the mean of weekly profits achieved by the strategy, "Std.Dev" to their standard deviation, "Sharpe Ratio" to the annualised ratio of the previous measures, "Max Drawdown" to the maximum drawdown experienced during the whole trading period, "Omega" to the omega ratio calculated having the loss threshold equal to zero and finally "VaR" to the value at risk with confidence level of 99%.

Avg.Ret	**Std.Dev**	**Sharpe**	**Max Drawd.**	**Omega**	**VaR**
0,0023	0,0253	0,6430	0.2552	1.2885	-0.0564

It is important to remember how this comparison does not intend to be exhaustive so that its results do not have to be taken as overwhelming evidences. Anyhow, they certainly raise some doubt regarding the additional information content provided by technical and macroeconomic variables to the prediction application. This is especially true in the case of technical indicators, as said. In fact, over simple input variables such as price and returns, they are the category of variables most extensively employed in ML financial literature. It is not clear if this has to be attributed to some objective evidence or simply to a tendency, consolidated through the years among practitioners, to erroneously consider technical variables as functional to the prediction process. It would be interesting to further study it.

The second important issue that this robustness check brings to light, crucial from a managerial point of view, is about the trade-off between the dimension of the input features set and the computation time necessary to train the model: how far should the feature selection process go in the search for new input variables? This knowing that the introduction of new variables inevitably leads the model to become more expensive from a computational point of view without any strong evidence about an increase in performance.

Chapter 6

Conclusions

Firstly, this thesis has tried to define a theoretical framework regarding the application of machine learning models for financial market predictions. In the thesis indeed, it has been covered the temporal evolution of the main ML algorithms tested in academic financial literature; moreover, the most frequently employed input features and the most important performance metrics frequently employed in financial literature have been presented. This first part, even if not containing any novelty, has to be considered an important contribution since it has merged theoretical concepts from both machine learning and finance: until now, it has not been yet emerged a well-defined field of study combining this two domains, even if their union has been reality in the business world for many years indeed.

After that, in the experimental section, the thesis has tried to understand the ability of machine learning in properly spotting recurrent patterns in equity market for stock market predictions. LSTM networks have been selected as ML reference model to spot such patterns due to their quite recent presentation to the machine learning community and their still limited application in financial market predictions literature, if compared to other ML models. Moreover, their ability to deal with correlated observations and with long-term dependencies makes them perfect for such types of prediction problems. Such models have been tested on the U.S. equity market, more precisely on 44 stocks belonging to the S&P500 at the end of 2019, employing observations from 2009 to 2019.

All the experimental work has been carried out having as final objective the presentation of useful recommendations for managerial decision processes dealing with the construction of a profitable illustrative investing strategy founded on LSTM networks' predictions. The approach has thus been business oriented: the implementation of the so-defined "modular strategy" has allowed to test the effectiveness of LSTM predicting at different forecasting horizons, besides the fact that it has allowed to test a possible approach to reduce the risk of the investment strategy.

The main results from the empirical work can be summarised as follows:

- LSTM networks have shown their ability in recognising proper recurrent patterns within stock price time-series. This ability has been clearly expressed by data science results: only 8 models out of 150 validated (i.e. 3 for $h = 3$, 2 for $h = 4$ and 3 for $h = 5$) and only 4 out of 30 during the trading phase (i.e. 1 for $h = 4$ and 3 for $h = 5$) presented an accuracy lower than the reference value of 33%. Considering the difficulty related to the prediction task and the limited means available to train LSTM networks, such results can be considered particularly impressive. Moreover, during specific time periods, accuracy for LSTM models reached extremely consistent results, all significantly higher than 40%.

- At the same time, always from data science results, it has emerged the inability of LSTM models in recognising stocks presenting non-significant future price movements (i.e. observations belonging to class "0"). The innovative choice to subdivide the entire dataset into three different classes instead of two, therefore, does not immediately appear to be a profitable one; at least taking into consideration the input features set used in this experimental work, LSTM models have not been able to correctly differentiate small price movements, both positive and negative, from large and profitable ones. In any case, this result must not be taken as an absolute evidence of the uselessness of a third "central" class since this class could have been beneficial allowing the model

to identify more clearly the two extreme classes outdistancing them by a clear margin. Particularly, it would be interesting in future researches to possibly validate such hypothesis since in this thesis it has not been possible. Finally, it must be observed how poor performances was expected and how, from a managerial perspective, observations belonging to class "0" are less interesting to be identified.

- In terms of profitability, the strategy created thanks to the predictions by LSTM networks has shown its edge over selected benchmarks: during a trading period of six years it managed to create a profit of 0.783% relative to the amount of money invested both for the long and short legs of the dollar-neutral portfolios (that in our case was 1$). Other benchmarks, instead, did not manage to create any significant result concluding the six year period with profit respectively of 0.007% by a random guessing strategy, 0.131% by a short-term reversal strategy and -0.001% by a "long-term ML strategy". This result have proven the potentiality of LSTM networks also from a financial perspective. Moreover, it has rejected the hypothesis of positive financial results obtained by the main startegy due to favourable market conditions; in fact, if this had been the case, the selected benchmarks would also have achieved comparable financial results.

- If results in terms of profitability were positive, the downside emerged by all the investment strategies based on LSTM predictions regards their excessive volatility. In terms of Sharpe ratio indeed, even profitable strategies have shown poor overall performances during the trading period. This evidence highlights the need of a specific focus during the design phase of a ML trading system on actions to reduce the variability created by models' predictions, as the execution of the "modular classification approach" in this thesis was. In any case, for instance, when examining the evolution of Sharpe ratio through time, it has been noted how during limited time periods the main strategy has

been able to positively perform even in terms of profits adjusted for the risk (i.e. Sharpe ratio consistently higher than 1). This result is still extremely valuable when compared with evidences in literature that highlight how ML models in recent time period appear unable to spot valuable patterns in stock market time-series (e.g. Huck, 2019).

- As it emerged from data science results, LSTM networks trained during periods of high market volatility and successively tested with observations following the turbulent period have shown inferior results compared to models trained and tested in similar market conditions (the same has been observed for models trained during non-turbulent periods and tested on turbulent ones). This result challenges the standard training approach, followed also in this thesis, regarding the creation of a dataset composed just by most recent and strictly temporally close to the trading period observations; such an approach aims indeed to always train models with most recent time-series, lying on the assumptions that newer observations will behave as much as most recent past ones. In any case, the evidence just presented creates the conditions to think about a possible different approach when dealing with financial time series: assuming recurrent patterns in the market created by recurrent investors' behavioural biases, it could be valuable the creation of specific models trained with observations coming only from specific market conditions during which specific biases came into play; the approach would lie on the hypothesis that, during such market conditions, market anomalies are usually driven by well-known biases whose effect on stock market could be predicted by well-prepared ML models.

- The "modular approach", implemented as a possible approach to reduce the risk of the investment strategy created by ML models, has not returned the desired results. The combination of different portfolios taking advantage of new information available from the market, in fact, has brought to an increase

in the volatility of the overall strategy; such result has been mainly due to the very poor performances obtained by both the portfolios $h = 4$ and $h = 5$. In any case, the approach has allowed to validate the hypothesis for which models' predictions consistently deteriorate with increasing forecasting horizons. This evidence, from a managerial perspective, poses consistent doubts about the possibility to use ML models for the creation of a medium/long term investment strategy. Even if, it is worth to remind the limited validation process employed that, if extended, could especially bring models predicting at longer horizons to achieve more consistent results. Finally, in spite of such not so positive results, the "modular approach" has resulted fruitful, as said, to evaluate models' behaviour at different forecasting horizons, but also to provide a reference to follow in future researches regarding the objective of controlling the unavoidable volatility created by ML models caused by some of their structural characteristics.

- Robustness analysis regarding the weighting scheme for portfolios has highlighted the impossibility to select the best configuration possible a priori. The only highly predictable implication emerged from preferring an "equally weighted" portfolio to a "confidence weighted" one is, as explained, the higher propensity of the former to allocate a higher part of the total investment to more volatile stocks. In any case, this evidence does not imply that the final strategy will result more volatile.

- Robustness analysis regarding the portfolio dimension has shown how, depending from the number of stocks presented in the database, it is necessary to validate such parameter in order to find the optimal trade-off between a high value facilitating diversification, thus lowering volatility, and a low value helping to avoid the inclusion within portfolios of stocks uncertainly classified by the ML model, thus lowering profits.

- The introduction of technical indicators and macroeconomic variables in the

input features set has represented a novelty for the literature. Unfortunately, due to time constraint, there was no possibility to test the overall impact on the original strategy of this introduction in relation to an input set containing only daily returns. In any case, the single result obtained for $h = 1$ has not shown significant and clear advantages for the model trained with extended input features set, if not a higher robustness for LSTM models to a bias for class "1" evident from data science metrics. Further tests should be conducted to properly assess the potential benefits deriving from the introduction of these variables within the input features set.

- There have not been evidences of a strong correlation between data science results and financial performances: during periods in which models presented high values of accuracy or G-score not necessarily the trading strategy has been profitable. This missing evidence poses an important issue related to the training and validation process of ML models in financial applications. Models' parameters in this experimental work, as in all others in literature, have been validated taking as decision variable a data science metrics (i.e. G-score), but financial results have later shown how such measure was not so correlated with financial performances. In many other ML application fields, the objective variable can be easily related to one or more data science metrics so that the problem can be reduced to a maximization/minimization of such metrics; in financial applications, the financial performance (i.e. returns and risk) is not so easily associable to any data science metrics instead, as highlighted, causing the validation process to become really difficult to be efficiently performed. This issue is particularly cumbersome for classification settings: imagine a model correctly classifying 4 out of 5 stocks, thus having 80% accuracy; it could easily lose money in the case in which the single wrongly classified stock is the one presenting the largest loss/gain. At the same time, regression settings could partially solve this issue at the expenses of a lower precision that, equally,

could nullify all the gains related to the possession of a an estimate of the magnitude related to the price movement.

- From a comparison of accuracy and G-score achieved by models within the different trading sets and within validation sets, the benefit of the validation process is not clear. The validation phase, in this study as in all other in literature, has been employed for consistency with the ML approach. A plausible reason regarding the difference in performance could be attributed to the fact that, in this experiment, training set and trading set have been separated by one year of observations. This large difference in time among the two could have caused the degradation in performance due to too evident change in market condition. In any case, it is worth mentioning how in the experiment, due to computational constraint, the grid search applied during the validation process exclusively aimed at finding the optimal size of hidden nodes. Such a search could be considered limited since it did not include many other parameters related to the training process such as learning rate, number of epochs etc.

Summarizing, the contribution of the thesis to the literature has been firstly in the formalization of the mixed knowledge from both machine learning and finance necessary to understand this recent approach to financial market predictions possible thank to the development of computer science; secondly, the thesis has highlighted evidence regarding the ability of LSTM networks to provide profitable results even in a context in which the efficiency of financial markets seems increased and arbitrage opportunities diminished due to an extensive use of ML solutions; finally, the thesis has tried to draw useful recommendation from a managerial point of view for an effective application of LSTM networks dealing with stock market predictions.

It is important when looking at the results, in any case, to be aware of all the limitations that have affected the experimental work:

- All the results in terms of profitability have been presented without consider-

ing transaction costs. This choice has been taken considering the difficulty in finding plausible estimates regarding cost-per-trade for the US equity market for the last 10 years. Moreover, it must be added the fact that, dealing with a long-short investing strategy, for a correct estimation it would have been necessary to differentiate cost estimates for both long and short trades; if in the case of former ones it is in any case possible to find some rational estimation, for the latter a realistic estimation is nearly impossible to be performed. It must be taken into consideration moreover how the study focuses on the "modular approach" as a possible approach to reduce the risk associated with the investment strategy and this is the reason why in terms of transactions costs it did not result very efficient, accounting for a lot of portfolio adjustments.

- As already cited, the validation process has been relegated to the number of hidden nodes of LSTM networks due to computational constraints. Many other parameters have been then omitted from a possible grid search having in mind that, for instance, the inclusion of just a single possible value for any other parameter of the model would have doubled the overall training time of the whole approach. This limitation had probably penalised results obtained during the trading phase but, at the same, it has corroborated results obtained, leaving a glimpse of the wide room for improvements over already valuable results. Moreover, also the long time distance (i.e. one year of trading observations) separating the training period from the trading one could have negatively influenced results. So, overall the parameter selection process could be viewed as too limited and could be considered as a fragility of the whole experimental work.

- The length of the overall trading period has just been of six years. During this period, it has been visible how models' performances during limited time periods have presented disappointing results, also in the case of the most profitable investment strategies. However, in general terms, a temporal under-

performance cannot be considered a problem since the underlying idea driving a ML trading system is that under-performing periods are unavoidable but, if limited and less numerous than over-performing periods, they will not consistently affect the final result. Clearly, during a so limited trading period, it has not been possible to extensively verify whether such under-performance periods have effectively been sporadic or systematic in our case. It is important to say that the choice regarding six years trading period has been somehow forced by the fact that just before 2009 (i.e. the starting year for observations populating the database employed) financial markets experienced an unprecedented crisis, thus during that period they behaved in a really unconventional way. So, to balance out this turbulence period, it would have been required to further extend observations in the database at least until 2000 causing the experiment to become too computationally onerous in relation to the available means.

In conclusion, regarding future works dealing with LSTM and financial market predictions, it is possible to highlight three promising paths to be possibly pursued in the light of the results obtained from this thesis:

1. Starting from the evidence about LSTM networks' ability to spot recurrent patterns in financial time-series, it should be further studied the relationship between data science parameters and financial performance of ML models so as to increase the efficiency of the validation process and thus possibly increasing the effectiveness of the investment strategies. The validation process for ML models is indeed still performed following the ordinary heuristic methods developed by computer scientists; in any case, such methods find little sense when applied to financial predictions whose final objective regards the maximisation of financial performances and not of data science metrics. This gap becomes even more cumbersome in the case of classification settings.

2. It should be studied effective methodologies to reduce the unavoidable variance

deriving from ML models' predictions. As shown by results indeed, if the profitability of LSTM investment strategies cannot be doubted, their variability in profits results to be their most important drawback.

3. The vast majority of ML applications in literature are extensively developed for a large quantity of financial data indiscriminately from any characteristic of the market or any particular flaw affecting the market in a specific time period. This approach is mainly attributable to the fact that, basically, all the literature to date comes from computer science field. Approaching the problem from a financial perspective instead, it could be interested creating ML trading systems for specific and well-known market flaws in order to make the model able to profit from them when effectively spotted. In such approach, all the design decisions regarding the structure of the ML models, the training process etc. would be guided by the characteristic of the anomaly itself. This approach would be at the opposite from the one experimented in this thesis, and in the vast majority of the ones in literature: you do not control which well-known market anomaly the model has been able to recognise by itself, but you train your model to become extremely accurate in recognizing the effects of a specific behavior by investors on stock price movements (i.e. you train your model not on temporally closer observations to your trading period, but on observations from most similar market periods to your trading one). Clearly, this way requires as a prerequisite a very good knowledge of financial markets and investors' behaviour.

Evidently, all these promising paths should be undertaken taking advantage of the analytic power of LSTM networks (or similar evolution of RNN such as GRU) since, to date, they can be considered among the most advanced ML solutions to approach long series of correlated observations, as it resulted from evidences presented so far.

Appendix A

Literature Review

Table A.1: Synthetic table of all the papers cited in Chapter 2

Paper	Objective of the experimental work	ML models	Database	Type of prediction (Trading strategy)	Results
Pai and Lin (2005)	Finding a solution for NN training: backprogagtion often converges to local minimum because of the tremendous noise and complex dimensionality of stock market data.	**ARIMA** **SVM** (gaussian)	Daily closing price of Ten stocks and 50 different predictors from each company.	Actual closing price predicted at t+1 in a regression setting.	The hybrid model can significantly reduce the overall forecasting errors made by the each individual model.
Chandrinos et al. (2018)	ML models are implemented as risk management tools with the aim of preventing investor from losses generated by an existing technical trading strategy: classification of the produced signals of a technical trading strategy into profitable and non-profitable ones.	**Decision Trees** **Deep NN**	Five currency pairs during 2006-2016. Input variables: 14 Technical indicators (SMA, RSI, acceleration, MACD, stochastic oscillator, momentum, Bollinger band, ROC) and price related variables.	Classification of every signal generated by the channel breakout trading strategy (The trade is effectively performed only if the signal is classified as "profitable").	DT prevent investors form significant drawndowns of the original trading strategy, reducing also its variance. They succeed in improving total returns for all currency pairs. NN reduce the standard deviation of 4/5 pairs and increase the total return of 3 of them. Particularly, NN manage to improve almost all of negative returns produced by the original strategy.
Huck (2019)	The main objective of the article is to tackle concurrently the dimensions of large datasets, machine learning and a discussion of the disagreement between EMH and the evidence reported in the ML literature. The natural application field for big data and machine learning in trading, according to the author, is statistical arbitrage, a class of short-term financial trading strategies that employ mean reversion models.	**Random Forests** **Deep Belief Networks** **Elastic Net Regression**	All stocks part of the S&P 900 from 1990 to 2015. Input variables: lagged Returns, dummy var. for stock identification, day of the week and month, ICB sectors, VIX, lagged gold and oil returns, Fama and French factors, 10y Treasury.	Classification setting forecasting at both $t+1$ and $t+5$ whether the return of the specific stock will be larger than the cross-sectional mean of returns (Stocks selected to enter in the portfolio each day are the top and flop 10).	Random forest appears to be the most effective forecasting technique. Models forecasting at $t+1$ performs better than the ones forecasting at $t+5$ and models trained only with lagged returns perform better than the ones trained with the whole set of predictors. In most recent periods algorithms are not able to generate significant trading signals confirming the evolutionary perspective of AMH: arbitrage opportunities have been eliminated by increasing popularity of ML methods among investors.

Krauss et al. (2017)	The paper, using only lagged returns, wants to find which of the different algorithms tested gives the best performance. Moreover, through variable importance and regression wants to understand which are the most important variables, patterns and systematic sources of risks that machine learning models are able to extract from data in order to create positive returns.	**Deep NN** **Gradient Boosted Trees** **Random Forests** **Ensembles**	S&P500 index constituents from Dec1989 to Sep2015. Input variables: from prices vectors, returns are computed over different time periods as $r_{t-k} = (P_t/P_{t-k}) - 1$, where $k \in \{1, ..., 20\} U \{40, 60, ..., 240\}$.	Classification setting forecasting at $t+1$ whether the return of the specific stock will be larger than the cross-sectional mean of returns (Stocks selected each day to enter in the portfolio are the top and flop k, with $k \in \{10, 50, 100, 150, 200\}$).	Increasing k leads to decreasing returns and directional accuracy. Ensembles outperform all base learners. Both tree based models perform better than deep NN, in any case authors highlight the difficulty in training such methods. NN lose efficiency when the number of layers is diminished. All models present positive alpha. Moreover, returns partly load on common sources of systematic risk suggesting an investment behaviour that incorporates several capital market anomalies. Algorithms seem to capture relative mispricings at times of high market turmoil.
Picasso et al. (2019)	The paper aims at combining technical and fundamental analysis through a classification model using as inputs both technical indicators and sentiment of news articles.	**Random Forests** **SVM** **NN**	Price-series of the 20 most capitalised company of the NASDAQ100 from 03/07/2017 to 14/06/2018 and features obtained from news related to specific stocks within the dataset obtained from two different dictionaries.	Trend classification with one week horizon (Trading signals on the basis of neural network output: Buy if $prediction > 0.5 + TSH$ and Sell if $prediction < 0.5 - TSH$)	NN trained with both price and news predictors clearly outperform NN trained only with price series. NN trained with both price and news predictors, instead, show only little improvements when compared to the NN trained only with news data.
Chen and Hao (2017)	The paper tries to improve the robustness and accuracy of ML model by weighting the contribution of each different input feature. It assumes that different input variables do not have all the same impact on the final output and it aims to nudge the algorithm to focus more on the most informative inputs within the dataset.	**SVM** **KNN**	Shangai and Shenzen stock exchange composite index from 31 Oct 2008 to 31 Dec 2014. Input variables: price, volume, returns and 9 technical indicators (MA, EMA, MACD, volume ratio, RSI, OBV, momentum index, AR, BR).	Regression setting predicting for both indexes 1, 5 , 10, 15, 20 and 30 days ahead.	The weighting process of features improves the results if compared with a model trained with non-weighted input variables; it manages to reduce the effect of those variable that are not so informative for the final prediction task.
Cai et al. (2012)	The paper wants extract low dimensional features from high dimensional raw input data and use them to make prediction about financial time-series.	**Deep Belief Network** **SVM**	Data related to the S&P500 index from 01/03/2000 to 02/22/2011. Input variables: prices and technical indicators for the index and for other 19 highly correlated stocks.	Regression setting where SVM predicts the actual closing price of S&P500 in $t+1$.	Generalization performance of SVM is deteriorated when dimension of input is increasing, even though more input variables comes with more information content. However, results confirm how dimensionality reduction can improve generalization performances in case of a large number of predictors within the dataset.

Sirignano and Cont (2019)	According to authors, large scale ML methods have not been deployed in finance because statistical modelling of financial time-series has remained asset specific and databases are often limited to recent time windows. They want to prove how, in reality, relations between variables and price time-series are universal and stationary so as to pool data across different assets and time periods and take advantage of a much richer dataset to estimate models.	**LSTM**	High-frequency record of all transactions and order cancellations for approximately 100 stocks traded on NASDAQ between 1 Jan 2014 and 31 March 2017.	Regression settings where the LSTM predicts the next price move; the time interval between two consecutive observations within the dataset is not constant but it varies considerably from a fraction of a second to seconds.	The paper find evidences in financial time series data of: *Universality* the model trained on all stocks outperforms stock-specific models, even for stocks not in the training sample, showing that features captured are not stock-specific. *Stationarity* model performance is stable across time, even for observations outside the sample. *Evidence of long memory in price formation* including order flow history as input, even up to several hours, improves prediction performance. *Generalization* the model extrapolates patterns related also to stocks not included in the training sample.
Liu and Liu (2018)	The study focuses on a movement trend-based data preparation method. It focuses on technical indicators, and it discretises them following specific rules per each indicator based on their financial meaning. Moreover, it compares RNN with their most successful variations.	**RNN** **LSTM** **GRU**	HS300 index from Jan 2005 to Dec 2017. Input variables: technical indicators (EMA, MACD, RSI, stochastic %k and %D, OBV, AD, ATR) converted into discrete values.	The model classifies the 5-days ahead movement of the index (daily Long-short portfolios created based on outputs of the classification performed by different models).	Results show how the best performance, among all combinations of models and datasets, is achieved by GRU trained with discretised data with an accuracy of 68% compared to LSTM with accuracy of 65% and RNN 64%. Moreover, among all models, the deep and narrow structure outperforms the shallow and wide one keeping the total number of neurons in hidden layers constant.
Chen and Ge (2019)	The paper wants to apply attention based LSTM on Honk Kong stock exchange data and compare it with results achieved by standard LSTM.	**LSTM** **Attention Mechanism**	Hong Kong market data from 1/01/2005 to 31/12/2017. Input variables: prices and technical indicators (ROC, EMA, MA, MACD, RSI, VROC, Bollinger bands, volume MA).	Models predict the price movement of each stock in the database in $t + 1$ (Daily long-only trading strategy for all stocks predicted to have a positive trend by the model).	The superiority of the model with the attention mechanism is demonstrated: the LSTM model with attention mechanism shows higher prediction accuracy over the simple LSTM model in 56/72 stocks. In terms of financial performance, this model shows also the higher returns compared to the simple LSTM.

Fischer and Krauss (2018)	The paper wants to compare results from LSTM networks with random forests since it delivers virtually no tuning and usually delivers good results and because it is a powerful benchmark for innovative machine learning algorithm, with deep neural network, to show the relative advantage over a model without memory, with logistic regression that is considered a baseline to derive the incremental value-add of LSTM in comparison to a standard classifier.	**LSTM** **Random forest** **Deep NN** **Logistic Regression**	S&P500 index constituents from Dec1989 to Sep2015. Return Sequences for LSTM: for each stock a sequence of 240 daily return. Input variables for benchmarking algorithms: lagged Returns with lag in {1, ..., 20} U {40, 60, ..., 240}.	Models forecast the probability of each stock to outperform the cross sectional median in $t+1$, making use of info in t (For the portfolio construction, the strategy goes long on top k stocks and short on flop k forming dollar-neutral Long-short portfolios. $k \in \{10, 50, 100, 150, 200\}$.	LSTM shows better performance than other models: it reaches an accuracy of 54.3% when $k = 10$ and daily returns, prior transaction costs, of 0.46%. The second best model is RAF with 53.8% accuracy and 0.43% daily returns. From 2001 to 2009, period in which ML models started effectively to be implemented in the market, the performances start decreasing for all models. Surprisingly, RAF perform incredibly well during the 2008 financial crisis with a sharpe ratio equal to 6. Conversly, from 2010 to 2015 RAF is found to destroy value while LSTM maintains capital constant after transaction costs.

Appendix B

Future Returns Distribution

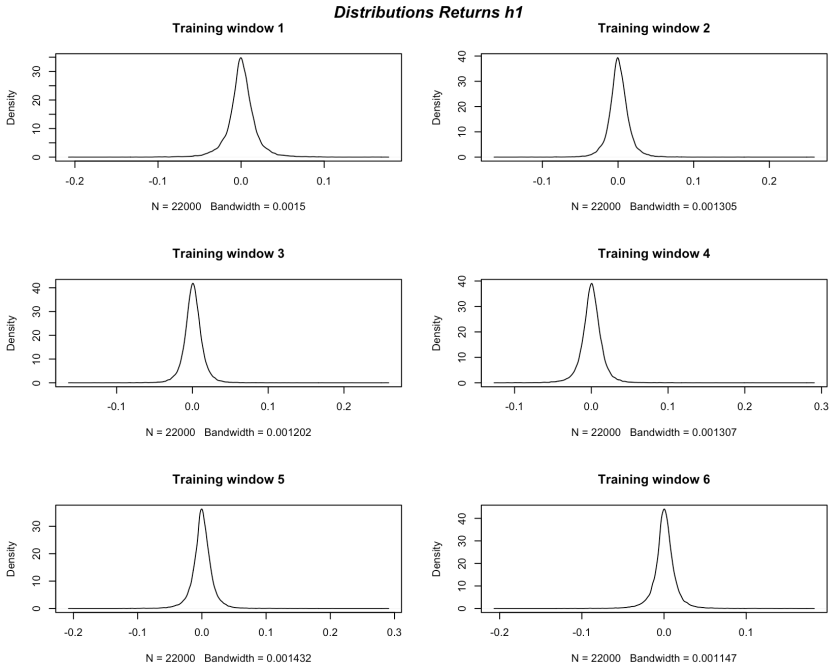

Figure B.1: Graphic representation of distribution of future returns at $t+1$ within all the training sets belonging to different training windows in dataset $h=1$.

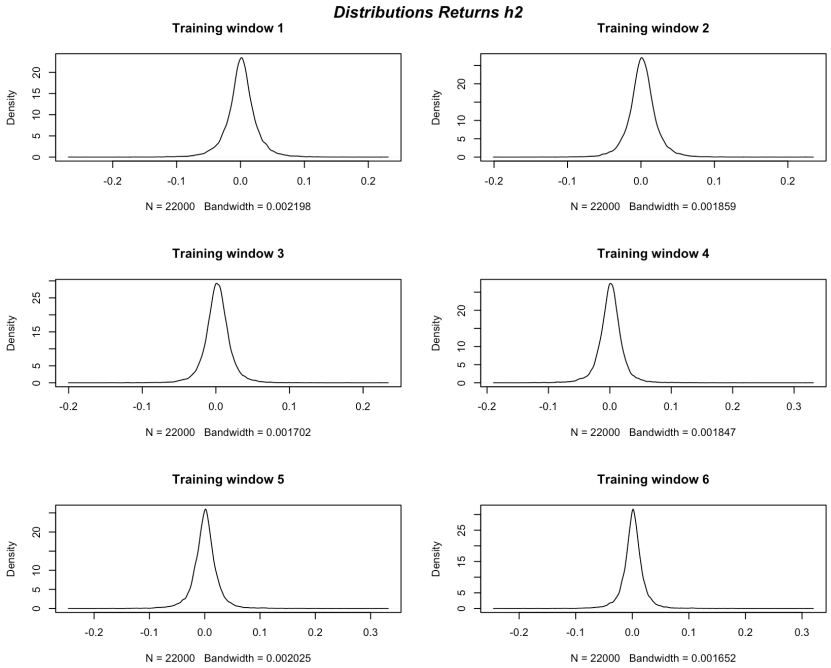

Figure B.2: Graphic representation of distribution of future returns at $t+2$ within all the training sets belonging to different training windows in dataset $h=2$.

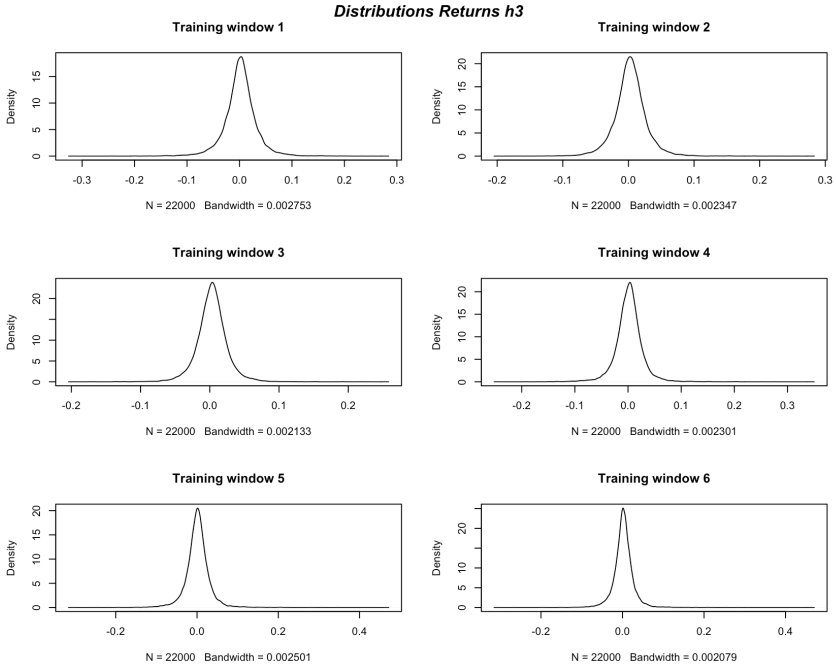

Figure B.3: Graphic representation of distribution of future returns at $t+3$ within all the training sets belonging to different training windows in dataset $h=3$.

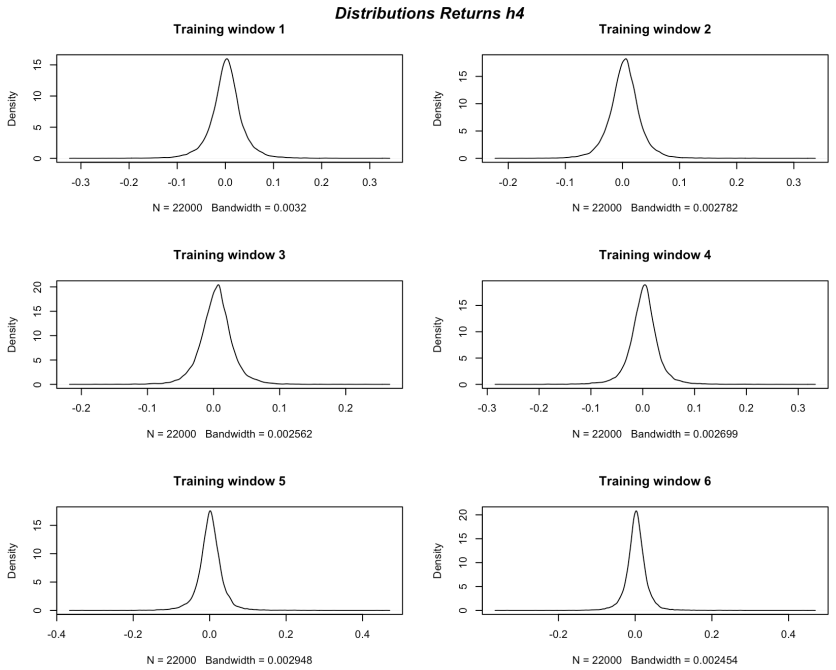

Figure B.4: Graphic representation of distribution of future returns at $t+4$ within all the training sets belonging to different training windows in dataset $h=4$.

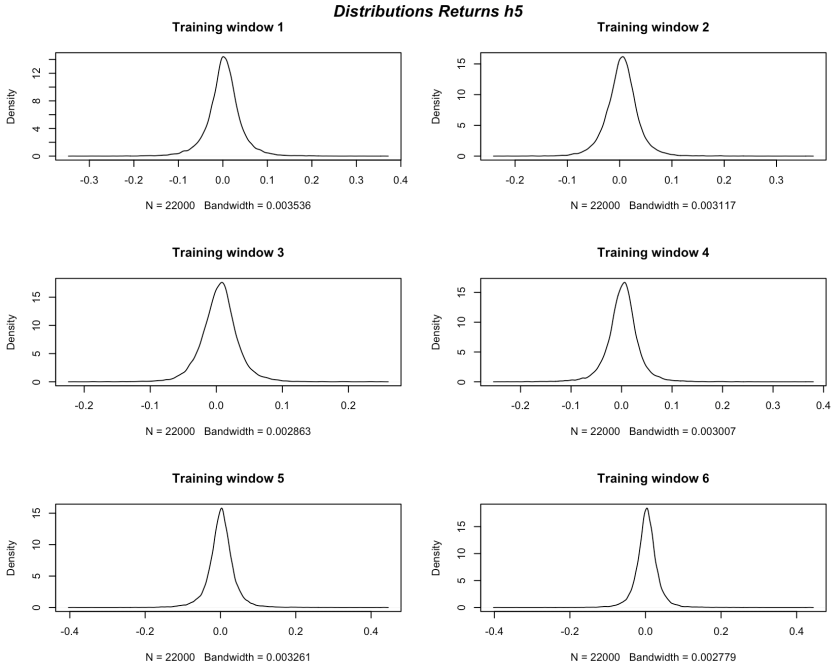

Figure B.5: Graphic representation of distribution of future returns at $t + 5$ within all the training sets belonging to different training windows in dataset $h = 5$.

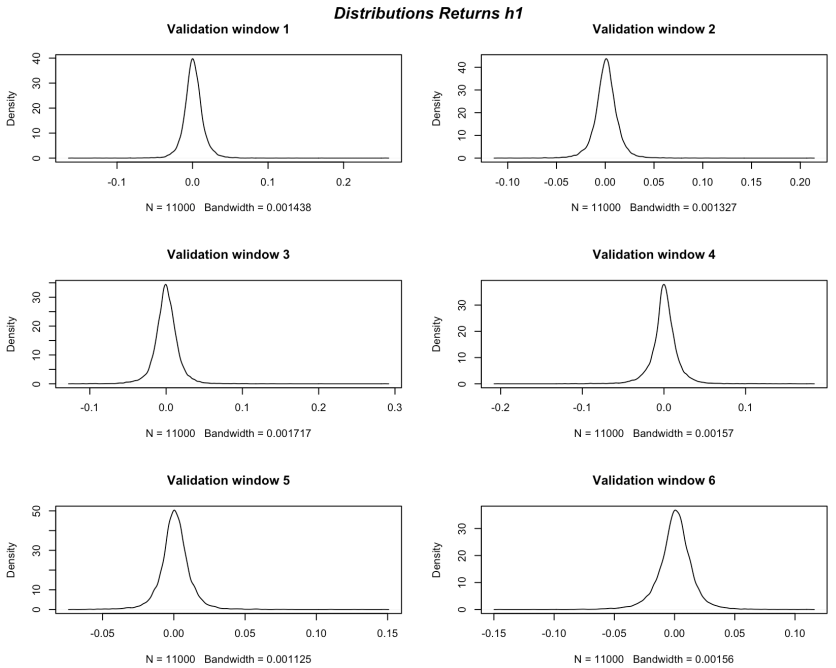

Figure B.6: Graphic representation of distribution of future returns at $t + 1$ within all the validation sets belonging to different training windows in dataset $h = 1$.

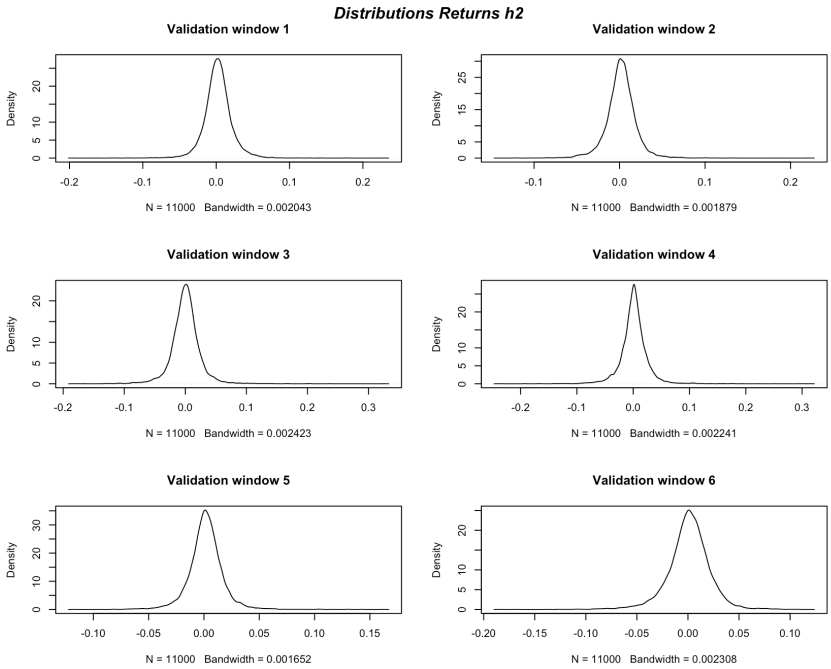

Figure B.7: Graphic representation of distribution of future returns at $t+2$ within all the validation sets belonging to different training windows in dataset $h=2$.

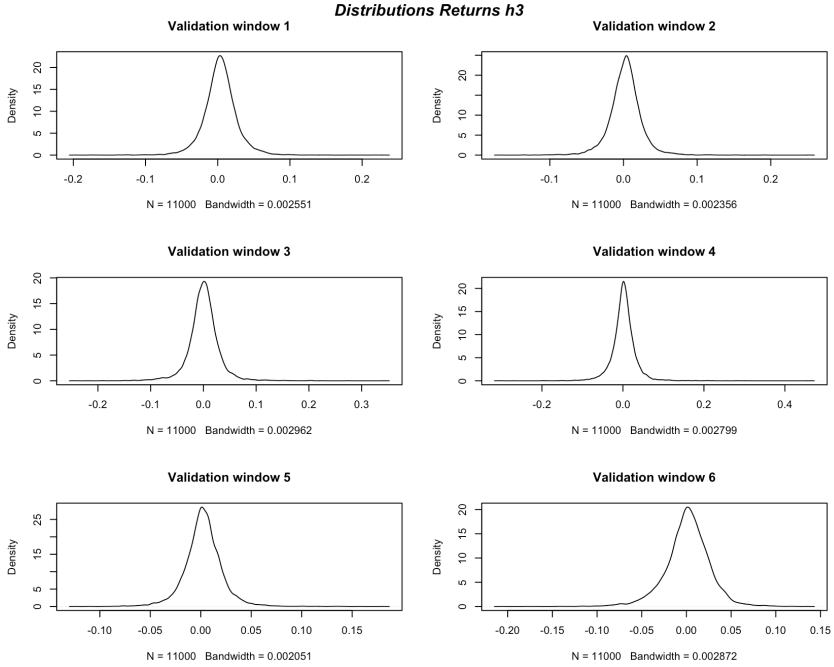

Figure B.8: Graphic representation of distribution of future returns at $t+3$ within all the validation sets belonging to different training windows in dataset $h=3$.

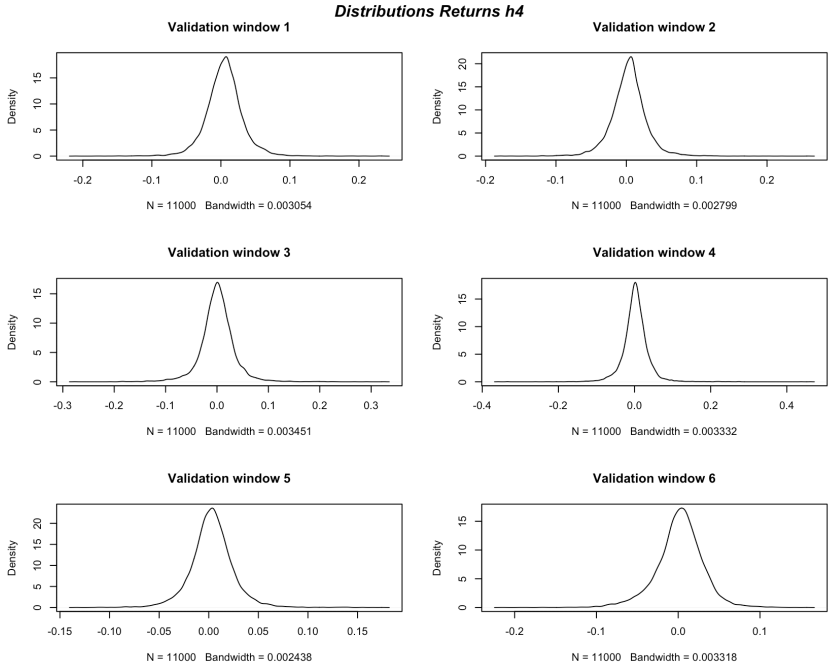

Figure B.9: Graphic representation of distribution of future returns at $t + 4$ within all the validation sets belonging to different training windows in dataset $h = 4$.

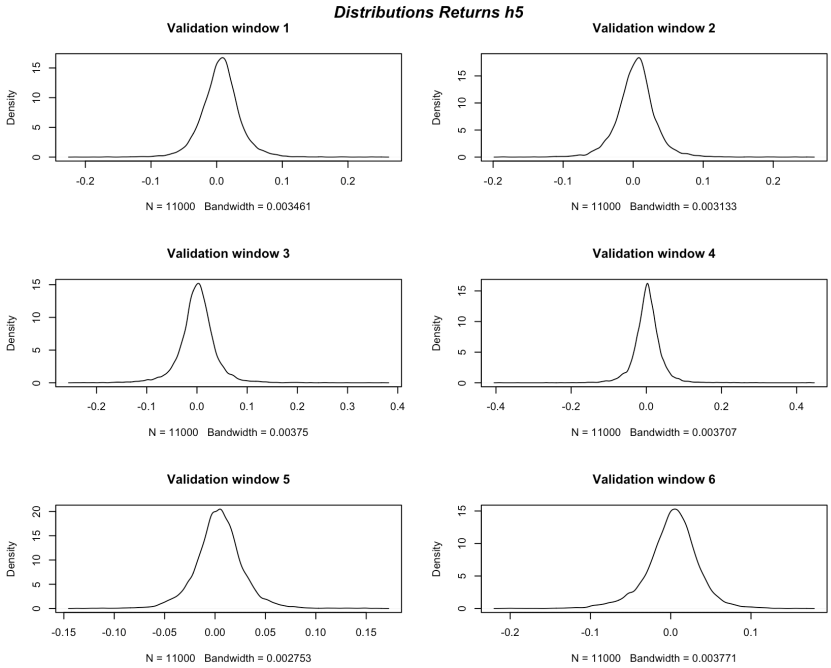

Figure B.10: Graphic representation of distribution of future returns at $t + 5$ within all the validation sets belonging to different training windows in dataset $h = 5$.

Table B.1: Percentage of observations belonging to each of the three classes in the training, validation and trading sets for the different forecasting horizons.

Window	Set	h=5			h=4			h=3			h=2			h=1		
		-1	0	1	-1	0	1	-1	0	1	-1	0	1	-1	0	1
1	*Training*	0,32	0,28	0,40	0,34	0,25	0,41	0,36	0,22	0,42	0,38	0,19	0,43	0,42	0,14	0,45
	Validation	0,26	0,30	0,44	0,28	0,28	0,44	0,30	0,26	0,44	0,33	0,22	0,45	0,38	0,16	0,46
	Trading	0,26	0,34	0,40	0,28	0,31	0,41	0,30	0,28	0,42	0,32	0,24	0,44	0,38	0,17	0,45
2	*Training*	0,28	0,30	0,41	0,30	0,28	0,42	0,32	0,25	0,43	0,35	0,21	0,44	0,40	0,16	0,44
	Validation	0,26	0,34	0,40	0,28	0,31	0,41	0,30	0,28	0,42	0,32	0,24	0,44	0,38	0,17	0,45
	Trading	0,34	0,30	0,36	0,36	0,26	0,38	0,38	0,23	0,40	0,40	0,19	0,41	0,43	0,14	0,43
3	*Training*	0,26	0,32	0,42	0,28	0,29	0,43	0,30	0,27	0,43	0,33	0,23	0,44	0,38	0,17	0,45
	Validation	0,34	0,30	0,36	0,36	0,26	0,38	0,38	0,23	0,40	0,40	0,19	0,41	0,43	0,14	0,43
	Trading	0,31	0,31	0,39	0,32	0,28	0,40	0,34	0,25	0,41	0,36	0,22	0,42	0,40	0,15	0,45
4	*Training*	0,30	0,32	0,38	0,32	0,29	0,39	0,34	0,25	0,41	0,36	0,22	0,42	0,40	0,16	0,44
	Validation	0,31	0,31	0,39	0,32	0,28	0,40	0,34	0,25	0,41	0,36	0,22	0,42	0,40	0,15	0,45
	Trading	0,24	0,39	0,38	0,26	0,36	0,38	0,28	0,32	0,40	0,31	0,27	0,42	0,37	0,20	0,44
5	*Training*	0,32	0,30	0,38	0,34	0,27	0,39	0,36	0,24	0,40	0,38	0,20	0,42	0,42	0,15	0,44
	Validation	0,24	0,39	0,38	0,26	0,36	0,38	0,28	0,32	0,40	0,31	0,27	0,42	0,37	0,20	0,44
	Trading	0,31	0,29	0,40	0,32	0,27	0,41	0,34	0,24	0,42	0,36	0,20	0,44	0,39	0,15	0,46
6	*Training*	0,27	0,35	0,38	0,29	0,32	0,39	0,31	0,29	0,40	0,34	0,24	0,42	0,38	0,18	0,44
	Validation	0,31	0,29	0,40	0,32	0,27	0,41	0,34	0,24	0,42	0,36	0,20	0,44	0,39	0,15	0,46
	Trading	0,30	0,28	0,42	0,32	0,25	0,43	0,34	0,23	0,44	0,36	0,19	0,45	0,39	0,14	0,47

Appendix C

Data Science Results

Table C.1: Hours occurred to train each of the 5 models presents in each training window per each different horizon.

Window	**Time h1 (h)**	**Time h2 (h)**	**Time h3 (h)**	**Time h4 (h)**	**Time h5 (h)**
1	1.12	1.67	1.22	1.32	1.47
2	1.11	2.65	1.23	1.78	2.27
3	1.76	2.54	1.34	2.67	1.46
4	1.99	2.59	2.25	2.63	2.27
5	1.25	2.69	1.08	2.50	2.26
6	1.27	2.71	1.08	2.53	2.30
Total	8.50	14.84	8.20	13.43	12.03

Table C.2: Results of different h=1 LSTM models for all number of neurons in the hidden layer in each training window.

	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.0	Prec.0	Spec.0	F1 0	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
25/1	0,666	0,389	0,353	0,491	0,009	0,176	0,992	0,017	0,360	0,486	0,682	0,414	0,420	0,489
20/1	0,806	0,387	0,209	0,523	0,001	0,333	1,000	0,001	0,219	0,492	0,810	0,303	0,408	0,420
15/1	0,622	0,382	0,379	0,474	0,000	0,000	1,000	0,000	0,389	0,469	0,631	0,425	0,415	0,492
10/1	0,748	0,394	0,291	0,516	0,001	0,100	0,998	0,002	0,297	0,493	0,745	0,370	0,421	0,471
5/1	0,574	0,396	0,458	0,468	0,000	0,000	1,000	0,000	0,461	0,472	0,567	0,466	0,430	0,515
25/2	0,533	0,388	0,492	0,449	0,024	0,182	0,977	0,043	0,491	0,481	0,566	0,486	0,426	0,511
20/2	0,487	0,387	0,534	0,431	0,004	0,259	0,998	0,007	0,549	0,473	0,498	0,508	0,432	0,517
15/2	0,422	0,377	0,579	0,399	0,006	0,183	0,995	0,011	0,594	0,467	0,444	0,523	0,428	0,501
10/2	0,460	0,380	0,546	0,416	0,000	0,000	1,000	0,000	0,569	0,472	0,478	0,516	0,430	0,512
5/2	0,546	0,387	0,478	0,453	0,010	0,150	0,988	0,019	0,480	0,473	0,561	0,477	0,424	0,512
25/3	0,294	0,476	0,756	0,363	0,003	0,250	0,999	0,005	0,771	0,453	0,295	0,570	0,459	0,476
20/3	0,243	0,467	0,791	0,319	0,001	0,143	0,999	0,003	0,805	0,447	0,248	0,575	0,451	0,442
15/3	0,322	0,462	0,717	0,379	0,012	0,123	0,987	0,022	0,722	0,453	0,340	0,556	0,451	0,482
10/3	0,140	0,465	0,878	0,215	0,000	0,000	1,000	0,000	0,884	0,437	0,140	0,585	0,441	0,351
5/3	0,189	0,457	0,830	0,268	0,007	0,192	0,996	0,013	0,837	0,441	0,198	0,578	0,443	0,398
25/4	0,551	0,404	0,456	0,466	0,008	0,241	0,995	0,016	0,483	0,484	0,581	0,483	0,439	0,516
20/4	0,523	0,401	0,478	0,454	0,008	0,182	0,993	0,016	0,498	0,475	0,552	0,486	0,434	0,511
15/4	0,519	0,405	0,491	0,455	0,002	0,231	0,999	0,004	0,517	0,477	0,539	0,496	0,440	0,518
10/4	0,481	0,415	0,547	0,445	0,007	0,267	0,997	0,140	0,564	0,475	0,495	0,516	0,446	0,521
5/4	0,408	0,392	0,577	0,400	0,000	0,000	1,000	0,000	0,596	0,458	0,428	0,518	0,430	0,493
25/5	0,732	0,372	0,287	0,493	0,004	0,167	0,995	0,008	0,287	0,454	0,734	0,352	0,393	0,459
20/5	0,608	0,372	0,408	0,461	0,006	0,292	0,996	0,013	0,405	0,442	0,607	0,422	0,399	0,496
15/5	0,556	0,375	0,466	0,448	0,001	0,333	1,000	0,002	0,479	0,456	0,560	0,467	0,412	0,516
10/5	0,748	0,369	0,264	0,494	0,000	0,000	1,000	0,000	0,266	0,446	0,746	0,333	0,389	0,446
5/5	0,808	0,369	0,205	0,507	0,003	0,162	0,997	0,005	0,199	0,439	0,805	0,274	0,382	0,401
25/6	0,555	0,399	0,456	0,464	0,002	0,250	0,999	0,004	0,462	0,471	0,559	0,466	0,431	0,506
20/6	0,678	0,395	0,326	0,499	0,026	0,181	0,980	0,045	0,313	0,473	0,704	0,377	0,415	0,461
15/6	0,549	0,395	0,452	0,459	0,000	0,000	1,000	0,000	0,455	0,463	0,551	0,459	0,425	0,500
10/6	0,550	0,397	0,457	0,461	0,000	0,000	1,000	0,000	0,463	0,468	0,553	0,465	0,429	0,505
5/6	0,195	0,410	0,818	0,264	0,000	0,000	1,000	0,000	0,821	0,464	0,194	0,592	0,454	0,400

Table C.3: Results of different h=2 LSTM models for all number of neurons in the hidden layer in each training window.

	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.0	Prec.0	Spec.0	F1 0	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
25/1	0,676	0,343	0,355	0,455	0,001	0,133	0,999	0,002	0,377	0,494	0,684	0,427	0,395	0,505
20/1	0,729	0,337	0,285	0,461	0,001	0,273	0,999	0,003	0,305	0,491	0,742	0,377	0,380	0,472
15/1	0,778	0,340	0,248	0,473	0,000	0,091	0,999	0,001	0,264	0,498	0,783	0,345	0,377	0,453
10/1	0,731	0,352	0,328	0,475	0,005	0,213	0,994	0,011	0,337	0,501	0,726	0,403	0,396	0,497
5/1	0,454	0,332	0,545	0,383	0,011	0,193	0,987	0,021	0,547	0,462	0,480	0,501	0,400	0,498
25/2	0,428	0,316	0,554	0,363	0,007	0,300	0,995	0,013	0,561	0,442	0,451	0,494	0,386	0,490
20/2	0,379	0,337	0,642	0,357	0,006	0,238	0,994	0,011	0,669	0,464	0,401	0,548	0,417	0,504
15/2	0,370	0,326	0,632	0,347	0,001	0,400	1,000	0,002	0,638	0,442	0,376	0,523	0,399	0,486
10/2	0,537	0,324	0,462	0,404	0,000	0,000	1,000	0,000	0,483	0,456	0,554	0,469	0,385	0,509
5/2	0,552	0,337	0,480	0,419	0,000	0,000	1,000	0,000	0,494	0,460	0,550	0,477	0,39 5	0,522
25/3	0,283	0,460	0,782	0,350	0,000	0,000	1,000	0,000	0,800	0,437	0,275	0,565	0,442	0,476
20/3	0,284	0,455	0,777	0,350	0,001	0,125	0,999	0,001	0,797	0,438	0,280	0,565	0,442	0,476
15/3	0,185	0,460	0,857	0,264	0,001	0,500	1,000	0,001	0,874	0,429	0,183	0,576	0,434	0,402
10/3	0,190	0,443	0,844	0,265	0,000	0,000	1,000	0,000	0,857	0,426	0,188	0,569	0,429	0,403
5/3	0,191	0,456	0,851	0,269	0,001	0,500	1,000	0,001	0,864	0,428	0,187	0,572	0,432	0,406
25/4	0,518	0,378	0,519	0,437	0,001	0,500	1,000	0,002	0,568	0,475	0,541	0,518	0,427	0,543
20/4	0,618	0,366	0,395	0,460	0,001	0,250	0,999	0,002	0,435	0,471	0,643	0,452	0,407	0,519
15/4	0,565	0,374	0,466	0,450	0,003	0,261	0,998	0,005	0,506	0,472	0,586	0,488	0,418	0,535
10/4	0,432	0,384	0,608	0,406	0,000	0,000	1,000	0,000	0,640	0,455	0,440	0,532	0,426	0,526
5/4	0,351	0,362	0,651	0,356	0,000	0,500	1,000	0,001	0,694	0,451	0,382	0,547	0,420	0,494
25/5	0,558	0,314	0,447	0,402	0,018	0,319	0,986	0,034	0,436	0,422	0,574	0,429	0,361	0,493
20/5	0,629	0,325	0,407	0,429	0,006	0,281	0,994	0,012	0,414	0,443	0,627	0,428	0,371	0,511
15/5	0,628	0,324	0,404	0,427	0,011	0,252	0,988	0,021	0,394	0,430	0,626	0,411	0,363	0,498
10/5	0,608	0,325	0,426	0,423	0,023	0,340	0,984	0,043	0,418	0,439	0,618	0,428	0,370	0,504
5/5	0,873	0,316	0,143	0,464	0,000	0,000	1,000	0,000	0,161	0,485	0,878	0,241	0,340	0,375
25/6	0,505	0,373	0,515	0,429	0,055	0,237	0,956	0,089	0,476	0,450	0,549	0,462	0,402	0,490
20/6	0,481	0,372	0,537	0,419	0,035	0,268	0,976	0,062	0,534	0,462	0,519	0,496	0,415	0,507
15/6	0,585	0,376	0,446	0,458	0,006	0,148	0,992	0,011	0,451	0,461	0,592	0,456	0,410	0,513
10/6	0,356	0,386	0,677	0,371	0,020	0,250	0,985	0,036	0,672	0,452	0,368	0,540	0,427	0,489
5/6	0,404	0,382	0,627	0,393	0,000	0,000	1,000	0,000	0,639	0,453	0,403	0,530	0,426	0,508

Table C.4: Results of different h=3 LSTM models for all number of neurons in the hidden layer in each training window.

	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.1	Prec.1	Spec.1	F1 1	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
25/1	0,645	0,309	0,392	0,418	0,020	0,297	0,983	0,037	0,410	0,499	0,673	0,450	0,378	0,514
20/1	0,469	0,296	0,530	0,363	0,041	0,306	0,968	0,072	0,516	0,461	0,520	0,487	0,378	0,492
15/1	0,611	0,309	0,422	0,410	0,009	0,238	0,990	0,017	0,435	0,478	0,623	0,456	0,376	0,516
10/1	0,676	0,313	0,375	0,428	0,017	0,302	0,986	0,032	0,394	0,506	0,693	0,443	0,380	0,516
5/1	0,783	0,310	0,264	0,444	0,001	0,074	0,997	0,001	0,289	0,516	0,785	0,370	0,360	0,475
25/2	0,584	0,310	0,434	0,405	0,020	0,316	0,983	0,038	0,455	0,465	0,622	0,460	0,373	0,516
20/2	0,669	0,304	0,336	0,418	0,011	0,333	0,992	0,021	0,360	0,464	0,700	0,405	0,357	0,491
15/2	0,464	0,302	0,535	0,366	0,014	0,478	0,994	0,027	0,559	0,445	0,496	0,495	0,379	0,509
10/2	0,514	0,297	0,471	0,376	0,002	0,375	0,999	0,004	0,507	0,449	0,550	0,476	0,369	0,511
5/2	0,525	0,319	0,514	0,397	0,001	0,500	1,000	0,002	0,558	0,466	0,538	0,508	0,39 3	0,541
25/3	0,160	0,456	0,885	0,236	0,018	0,291	0,987	0,035	0,889	0,413	0,169	0,564	0,417	0,377
20/3	0,258	0,439	0,802	0,325	0,001	0,100	0,998	0,002	0,829	0,422	0,256	0,559	0,425	0,462
15/3	0,316	0,389	0,701	0,349	0,002	0,174	0,998	0,003	0,734	0,420	0,335	0,534	0,410	0,482
10/3	0,145	0,411	0,875	0,215	0,000	0,000	1,000	0,000	0,883	0,403	0,144	0,554	0,404	0,358
5/3	0,094	0,413	0,919	0,154	0,003	0,205	0,996	0,006	0,928	0,404	0,101	0,563	0,404	0,296
25/4	0,531	0,353	0,503	0,424	0,009	0,269	0,992	0,018	0,538	0,457	0,555	0,494	0,402	0,534
20/4	0,560	0,353	0,477	0,433	0,003	0,421	0,999	0,006	0,538	0,476	0,589	0,505	0,410	0,549
15/4	0,518	0,373	0,556	0,434	0,004	0,294	0,997	0,007	0,611	0,475	0,530	0,535	0,427	0,563
10/4	0,562	0,357	0,483	0,437	0,000	0,500	1,000	0,001	0,546	0,480	0,588	0,511	0,414	0,554
5/4	0,552	0,354	0,486	0,431	0,000	0,000	1,000	0,000	0,546	0,474	0,578	0,507	0,411	0,549
25/5	0,674	0,278	0,323	0,394	0,069	0,332	0,934	0,114	0,266	0,413	0,749	0,324	0,317	0,424
20/5	0,391	0,280	0,611	0,326	0,280	0,356	0,760	0,314	0,383	0,428	0,660	0,404	0,352	0,387
15/5	0,681	0,284	0,334	0,401	0,178	0,334	0,832	0,232	0,178	0,446	0,854	0,254	0,318	0,348
10/5	0,608	0,277	0,385	0,380	0,216	0,351	0,810	0,268	0,194	0,410	0,815	0,263	0,317	0,343
5/5	0,423	0,303	0,622	0,353	0,198	0,345	0,821	0,252	0,450	0,423	0,593	0,436	0,361	0,436
25/6	0,649	0,353	0,386	0,457	0,113	0,275	0,905	0,160	0,298	0,453	0,742	0,359	0,373	0,440
20/6	0,568	0,342	0,435	0,427	0,089	0,268	0,922	0,134	0,377	0,446	0,664	0,409	0,373	0,463
15/6	0,572	0,341	0,429	0,427	0,132	0,288	0,896	0,181	0,340	0,448	0,699	0,387	0,369	0,441
10/6	0,356	0,343	0,649	0,349	0,144	0,290	0,888	0,192	0,551	0,436	0,489	0,487	0,386	0,442
5/6	0,488	0,372	0,576	0,422	0,018	0,299	0,987	0,033	0,593	0,459	0,498	0,518	0,418	0,538

Table C.5: Results of different h=4 LSTM models for all number of neurons in the hidden layer in each training window.

	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.0	Prec.0	Spec.0	F1 0	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
25/1	0,572	0,292	0,463	0,386	0,042	0,364	0,972	0,076	0,463	0,487	0,611	0,475	0,377	0,514
20/1	0,765	0,301	0,311	0,432	0,013	0,310	0,989	0,025	0,325	0,517	0,758	0,399	0,361	0,498
15/1	0,711	0,300	0,359	0,422	0,000	0,100	0,999	0,001	0,398	0,522	0,709	0,451	0,375	0,532
10/1	0,774	0,282	0,237	0,413	0,001	0,150	0,998	0,002	0,256	0,489	0,787	0,336	0,330	0,445
5/1	0,531	0,303	0,527	0,385	0,003	0,211	0,996	0,005	0,546	0,478	0,524	0,510	0,391	0,538
25/2	0,475	0,280	0,518	0,352	0,063	0,366	0,951	0,108	0,506	0,442	0,561	0,472	0,360	0,490
20/2	0,454	0,281	0,542	0,347	0,011	0,366	0,992	0,021	0,571	0,436	0,492	0,495	0,365	0,509
15/2	0,500	0,296	0,532	0,372	0,026	0,307	0,973	0,049	0,541	0,444	0,534	0,488	0,370	0,520
10/2	0,451	0,288	0,562	0,351	0,009	0,320	0,991	0,018	0,574	0,426	0,468	0,489	0,364	0,509
5/2	0,456	0,302	0,586	0,364	0,003	0,265	0,997	0,005	0,624	0,446	0,466	0,520	0,384	0,534
25/3	0,235	0,413	0,816	0,299	0,025	0,316	0,981	0,046	0,828	0,404	0,254	0,543	0,404	0,441
20/3	0,309	0,382	0,723	0,342	0,008	0,358	0,995	0,016	0,757	0,407	0,326	0,529	0,399	0,484
15/3	0,139	0,430	0,898	0,210	0,019	0,375	0,989	0,035	0,912	0,396	0,152	0,553	0,400	0,356
10/3	0,169	0,384	0,850	0,235	0,001	0,222	0,999	0,001	0,851	0,383	0,163	0,528	0,383	0,379
5/3	0,071	0,411	0,944	0,121	0,000	0,000	1,000	0,000	0,948	0,383	0,067	0,546	0,385	0,260
25/4	0,471	0,357	0,595	0,406	0,047	0,309	0,959	0,081	0,608	0,453	0,519	0,519	0,406	0,535
20/4	0,534	0,338	0,502	0,414	0,023	0,342	0,983	0,043	0,560	0,470	0,586	0,511	0,401	0,547
15/4	0,388	0,348	0,654	0,367	0,029	0,341	0,978	0,054	0,674	0,434	0,422	0,528	0,400	0,511
10/4	0,491	0,341	0,547	0,402	0,009	0,284	0,991	0,017	0,617	0,465	0,534	0,530	0,406	0,551
5/4	0,375	0,321	0,624	0,346	0,020	0,512	0,993	0,039	0,678	0,438	0,429	0,532	0,395	0,504
25/5	0,566	0,282	0,495	0,376	0,194	0,378	0,824	0,257	0,334	0,433	0,726	0,377	0,344	0,434
20/5	0,410	0,261	0,593	0,319	0,369	0,352	0,624	0,360	0,246	0,433	0,798	0,314	0,333	0,318
15/5	0,329	0,265	0,680	0,293	0,163	0,367	0,845	0,225	0,534	0,395	0,488	0,454	0,349	0,419
10/5	0,729	0,256	0,257	0,379	0,105	0,386	0,908	0,165	0,193	0,455	0,855	0,271	0,301	0,376
5/5	0,662	0,257	0,331	0,371	0,234	0,391	0,799	0,293	0,140	0,451	0,893	0,214	0,309	0,305
25/6	0,321	0,311	0,667	0,316	0,171	0,295	0,850	0,216	0,536	0,429	0,498	0,476	0,369	0,415
20/6	0,473	0,316	0,520	0,379	0,173	0,294	0,848	0,218	0,390	0,442	0,654	0,414	0,358	0,430
15/6	0,461	0,343	0,586	0,393	0,168	0,347	0,884	0,226	0,469	0,438	0,578	0,453	0,386	0,465
10/6	0,405	0,333	0,620	0,366	0,237	0,316	0,812	0,271	0,443	0,445	0,612	0,444	0,376	0,424
5/6	0,351	0,339	0,679	0,345	0,126	0,303	0,894	0,178	0,595	0,440	0,468	0,506	0,392	0,457

Table C.6: Results of different h=5 LSTM models for all number of neurons in the hidden layer in each training window.

	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.0	Prec.0	Spec.0	F1 0	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
25/1	0,547	0,265	0,469	0,357	0,045	0,313	0,957	0,079	0,446	0,463	0,599	0,455	0,351	0,494
20/1	0,582	0,258	0,415	0,357	0,046	0,383	0,968	0,082	0,406	0,468	0,642	0,435	0,342	0,486
15/1	0,646	0,283	0,429	0,394	0,041	0,302	0,959	0,072	0,437	0,519	0,685	0,474	0,371	0,531
10/1	0,718	0,275	0,340	0,398	0,016	0,289	0,983	0,031	0,368	0,523	0,739	0,432	0,352	0,514
5/1	0,773	0,274	0,285	0,405	0,042	0,352	0,967	0,074	0,302	0,563	0,818	0,393	0,345	0,483
25/2	0,429	0,244	0,524	0,311	0,058	0,393	0,955	0,101	0,504	0,413	0,524	0,454	0,334	0,465
20/2	0,580	0,262	0,415	0,361	0,037	0,400	0,972	0,068	0,440	0,456	0,651	0,448	0,341	0,505
15/2	0,234	0,272	0,776	0,251	0,104	0,401	0,921	0,165	0,717	0,418	0,335	0,528	0,383	0,409
10/2	0,462	0,302	0,618	0,366	0,073	0,381	0,940	0,122	0,608	0,456	0,518	0,521	0,389	0,530
5/2	0,331	0,259	0,661	0,290	0,042	0,355	0,961	0,075	0,648	0,416	0,394	0,506	0,360	0,463
25/3	0,458	0,382	0,623	0,416	0,024	0,281	0,974	0,044	0,668	0,427	0,485	0,520	0,405	0,553
20/3	0,277	0,404	0,792	0,328	0,059	0,334	0,950	0,100	0,795	0,404	0,329	0,536	0,400	0,469
15/3	0,449	0,382	0,631	0,413	0,013	0,364	0,990	0,026	0,680	0,418	0,457	0,518	0,403	0,553
10/3	0,177	0,394	0,862	0,245	0,013	0,344	0,989	0,026	0,880	0,383	0,188	0,534	0,384	0,395
5/3	0,175	0,408	0,871	0,245	0,012	0,256	0,985	0,023	0,887	0,384	0,185	0,536	0,386	0,394
25/4	0,480	0,342	0,591	0,399	0,074	0,345	0,938	0,121	0,597	0,458	0,556	0,518	0,400	0,536
20/4	0,519	0,330	0,535	0,404	0,045	0,355	0,964	0,080	0,587	0,473	0,589	0,524	0,399	0,552
15/4	0,585	0,326	0,465	0,419	0,039	0,339	0,967	0,070	0,509	0,473	0,644	0,490	0,388	0,546
10/4	0,464	0,317	0,558	0,377	0,038	0,324	0,965	0,068	0,606	0,453	0,541	0,518	0,388	0,530
5/4	0,626	0,329	0,434	0,431	0,039	0,279	0,955	0,068	0,454	0,470	0,679	0,462	0,379	0,533
25/5	0,312	0,244	0,699	0,274	0,301	0,390	0,704	0,340	0,412	0,390	0,610	0,401	0,346	0,359
20/5	0,557	0,253	0,487	0,348	0,089	0,376	0,907	0,144	0,419	0,410	0,635	0,414	0,325	0,483
15/5	0,288	0,233	0,706	0,258	0,262	0,421	0,774	0,323	0,476	0,384	0,538	0,425	0,349	0,370
10/5	0,606	0,260	0,462	0,364	0,098	0,404	0,909	0,158	0,378	0,405	0,664	0,391	0,324	0,479
5/5	0,585	0,250	0,453	0,350	0,197	0,435	0,840	0,271	0,294	0,411	0,746	0,343	0,326	0,415
25/6	0,573	0,305	0,426	0,398	0,155	0,332	0,872	0,212	0,310	0,433	0,724	0,361	0,346	0,422
20/6	0,464	0,315	0,557	0,376	0,233	0,331	0,807	0,274	0,391	0,457	0,686	0,421	0,368	0,426
15/6	0,305	0,346	0,747	0,324	0,240	0,329	0,799	0,278	0,557	0,434	0,508	0,488	0,388	0,412
10/6	0,452	0,305	0,547	0,364	0,250	0,323	0,786	0,282	0,358	0,449	0,701	0,398	0,355	0,403
5/6	0,553	0,327	0,500	0,411	0,210	0,334	0,829	0,258	0,344	0,461	0,727	0,394	0,369	0,436

Table C.7: Results of different configurations selected in the validation process within the trading set.

	Window	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.0	Prec.0	Spec.0	F1 0	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
h=1	*5/1*	0,912	0,351	0,124	0,506	0,000	0,000	1,000	0,000	0,144	0,618	0,918	0,233	0,380	0,362
	20/2	0,427	0,395	0,458	0,410	0,000	0,000	1,000	0,000	0,456	0,370	0,451	0,408	0,382	0,441
	15/3	0,455	0,479	0,595	0,466	0,000	0,000	0,998	0,000	0,610	0,412	0,454	0,492	0,440	0,527
	10/4	0,677	0,382	0,312	0,489	0,005	0,250	0,997	0,009	0,322	0,425	0,695	0,366	0,395	0,467
	15/5	0,665	0,379	0,341	0,483	0,003	0,143	0,997	0,006	0,348	0,500	0,676	0,410	0,419	0,481
	25/6	0,279	0,318	0,672	0,297	0,022	0,280	0,990	0,040	0,673	0,496	0,316	0,571	0,438	0,433
h=2	*25/1*	0,587	0,324	0,438	0,417	0,000	0,000	1,000	0,000	0,462	0,465	0,594	0,463	0,385	0,521
	5/2	0,223	0,360	0,727	0,276	0,000	0,000	1,000	0,000	0,733	0,401	0,243	0,518	0,390	0,405
	20/3	0,286	0,356	0,676	0,317	0,091	0,282	0,938	0,137	0,624	0,405	0,378	0,491	0,381	0,422
	25/4	0,853	0,353	0,151	0,499	0,023	0,171	0,957	0,040	0,142	0,470	0,905	0,218	0,359	0,348
	20/5	0,745	0,365	0,230	0,490	0,012	0,417	0,996	0,023	0,235	0,436	0,767	0,305	0,382	0,418
	15/6	0,695	0,321	0,306	0,439	0,000	0,000	1,000	0,000	0,313	0,507	0,701	0,387	0,378	0,467
h=3	*10/1*	0,452	0,314	0,561	0,370	0,000	0,000	0,999	0,000	0,578	0,434	0,459	0,496	0,380	0,511
	5/2	0,515	0,407	0,479	0,454	0,008	0,571	0,998	0,016	0,528	0,402	0,550	0,457	0,405	0,521
	15/3	0,151	0,407	0,879	0,220	0,056	0,378	0,973	0,098	0,872	0,435	0,192	0,581	0,430	0,362
	15/4	0,704	0,310	0,355	0,430	0,119	0,337	0,883	0,176	0,240	0,407	0,790	0,302	0,335	0,411
	5/5	0,361	0,337	0,626	0,349	0,195	0,242	0,814	0,216	0,448	0,428	0,563	0,438	0,359	0,403
	5/6	0,408	0,314	0,572	0,355	0,023	0,256	0,981	0,043	0,561	0,463	0,443	0,507	0,396	0,478
h=4	*5/1*	0,514	0,270	0,448	0,354	0,120	0,309	0,889	0,172	0,357	0,437	0,663	0,393	0,332	0,428
	5/2	0,455	0,387	0,589	0,418	0,038	0,319	0,969	0,067	0,582	0,386	0,483	0,464	0,384	0,514
	20/3	0,176	0,339	0,824	0,231	0,061	0,285	0,946	0,101	0,795	0,415	0,250	0,545	0,394	0,374
	10/4	0,653	0,274	0,357	0,386	0,186	0,327	0,783	0,237	0,166	0,411	0,862	0,237	0,305	0,329
	25/5	0,703	0,308	0,304	0,428	0,115	0,325	0,911	0,170	0,208	0,427	0,797	0,280	0,334	0,383
	15/6	0,248	0,335	0,783	0,285	0,060	0,239	0,934	0,096	0,711	0,437	0,293	0,541	0,401	0,420
h=5	*15/1*	0,874	0,285	0,160	0,429	0,017	0,387	0,987	0,032	0,153	0,450	0,875	0,229	0,309	0,366
	10/2	0,409	0,336	0,621	0,369	0,041	0,424	0,975	0,075	0,634	0,406	0,450	0,495	0,379	0,509
	25/3	0,311	0,310	0,702	0,310	0,117	0,421	0,927	0,183	0,698	0,442	0,443	0,541	0,400	0,466
	20/4	0,765	0,225	0,193	0,347	0,126	0,350	0,849	0,186	0,066	0,397	0,940	0,113	0,253	0,224
	20/5	0,699	0,303	0,273	0,422	0,137	0,266	0,861	0,181	0,152	0,443	0,862	0,226	0,318	0,326
	5/6	0,314	0,366	0,751	0,338	0,454	0,296	0,594	0,359	0,338	0,448	0,707	0,385	0,362	0,326

Appendix D

Financial Results

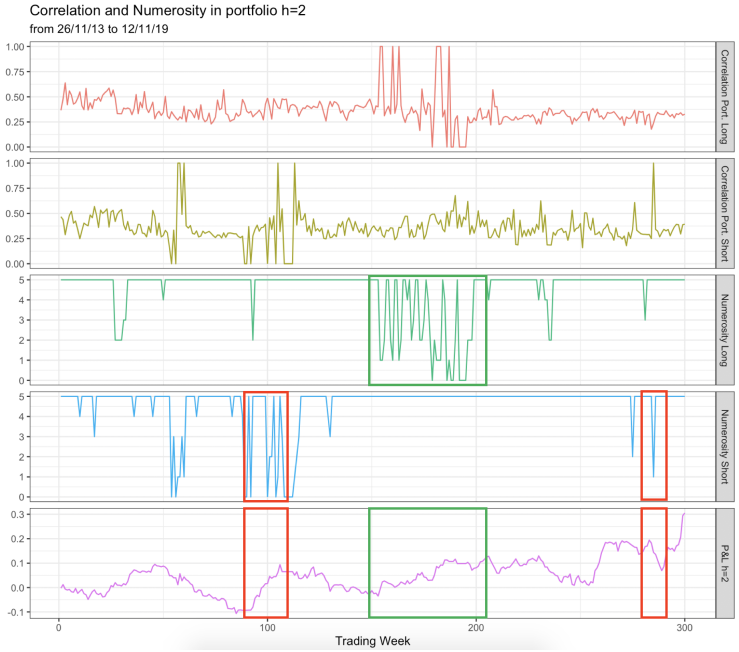

Figure D.1: Comparison between the evolution of numerosness and correlation of stocks within both the long and short legs of the portfolio $h = 2$ and the profitability of the portfolio itself. Obviously, correlation is equal to 1 when in the portfolios is present a single stock and to 0 when none stock is in it. Red boxes highlight strong results possibly driven by low numerousness of the portfolio while green boxes highlight periods in which low numerousness have clearly not brought to an increase in P&L volatility.

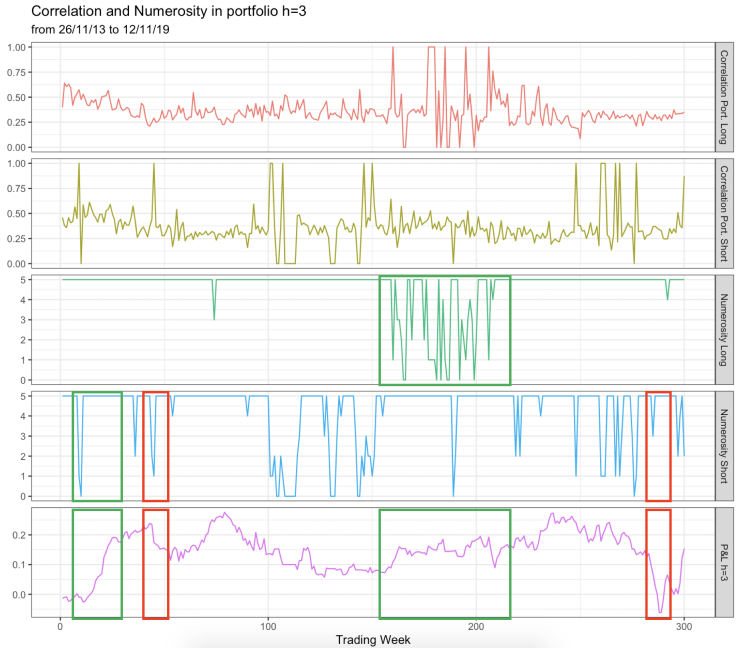

Figure D.2: Comparison between the evolution of numerosness and correlation of stocks within both the long and short legs of the portfolio $h = 3$ and the profitability of the portfolio itself.

Figure D.3: Comparison between the evolution of numerosness and correlation of stocks within both the long and short legs of the portfolio $h = 4$ and the profitability of the portfolio itself.

Figure D.4: Comparison between the evolution of numerosness and correlation of stocks within both the long and short legs of the portfolio $h = 5$ and the profitability of the portfolio itself.

Table D.1: Synthetic table showing the average number of stocks presented in the different portfolios during the whole trading period ("# Stocks"), the average weekly correlation among these stocks ("Weekly Corr") and, finally, the average number of stocks subdivided by industry selected every week by LSTM networks.

		# Stocks	Weekly Corr	TCH	EN	MAT	IND	CD	CS	HC	FIN	COMM	UT	RE
h=1	*L*	4,537	0,384	0,267	0,570	0,387	0,407	0,447	0,523	0,333	0,480	0,397	0,397	0,330
	S	4,757	0,360	0,353	0,273	0,473	0,507	0,413	0,380	0,563	0,480	0,447	0,517	0,350
h=2	*L*	4,510	0,365	0,193	0,700	0,530	0,500	0,490	0,390	0,267	0,373	0,490	0,183	0,393
	S	4,617	0,356	0,380	0,260	0,480	0,390	0,477	0,457	0,470	0,423	0,423	0,493	0,363
h=3	*L*	4,657	0,359	0,197	0,503	0,597	0,323	0,497	0,390	0,317	0,280	0,397	0,720	0,437
	S	4,327	0,364	0,337	0,427	0,350	0,340	0,437	0,507	0,407	0,243	0,477	0,333	0,470
h=4	*L*	4,423	0,362	0,310	0,493	0,353	0,547	0,360	0,490	0,373	0,300	0,527	0,220	0,450
	S	4,460	0,363	0,390	0,337	0,423	0,310	0,390	0,377	0,430	0,447	0,370	0,560	0,427
h=5	*L*	4,083	0,392	0,183	0,667	0,517	0,440	0,477	0,477	0,187	0,253	0,453	0,133	0,297
	S	4,533	0,357	0,387	0,147	0,393	0,437	0,487	0,310	0,480	0,393	0,447	0,610	0,443
			Mean	0,300	0,438	0,450	0,420	0,447	0,430	0,383	0,367	0,443	0,417	0,396

Appendix E

Robustness Analysis

Table E.1: Data science results of $h = 1$ configuration employing only daily returns as input feature.

Window	Sens.-1	Prec.-1	Spec.-1	F1 -1	Sens.0	Prec.0	Spec.0	F1 0	Sens.1	Prec.1	Spec.1	F1 1	Accuracy	G-score
5/1	0,528	0,377	0,473	0,440	0,000	0,000	1,000	0,000	0,491	0,468	0,542	0,479	0,420	0,509
20/2	0,418	0,426	0,577	0,422	0,000	0,000	1,000	0,000	0,585	0,436	0,426	0,500	0,432	0,495
15/3	0,353	0,409	0,659	0,379	0,000	0,000	1,000	0,000	0,665	0,454	0,354	0,540	0,439	0,484
10/4	0,049	0,445	0,965	0,088	0,000	0,000	1,000	0,000	0,962	0,436	0,041	0,600	0,437	0,217
15/5	0,169	0,421	0,849	0,241	0,000	0,000	1,000	0,000	0,850	0,463	0,166	0,600	0,456	0,379
25/6	0,194	0,356	0,778	0,251	0,003	0,128	0,996	0,006	0,782	0,466	0,211	0,584	0,441	0,389

List of Abbreviations

Abbreviation	Description
ML	Machine Learning
kNN	K-Nearest Neighbors
RAF	Random Forests
AdaBoost	Adaptive Boosting
GBT	Gradient Boosted Trees
SVM	Support Vector Machine
DBN	Deep Belief Networks
NN	Feed-Forward Neural Networks
RNN	Recurrent Neural Networks
GRU	Gated Recurrent Unit Networks
LSTM	Long Short-Term Memory Networks
S&P	Standard and Poor
EMH	Efficient Market Hypothesis
AMH	Adaptive Market Hypothesis
GICS	Global Industry Classification Standard
CBOE	Chiacago Board Options Exchange
LIBOR	London Interbank Offered Rate
ICB	Industry Classification Benchmark
VIX	Cboe Volatility Index
MAE	Mean Absolute Error
MSE	Mean Squared Error
MAPE	Mean Absolute Percentage Error
RMSE	Root Mean Square Error
VAR	Value At Risk
TPR	True Positive Rate
TNR	True Negative Rate
AU-ROC	Area under Receiver Operating Characteristic Curve
ROC	Receiver Operating Characteristic Curve
MDD	Maximum Drawdown

Bibliography

Allaire, J. and Chollet, F. (2019), *keras: R Interface to 'Keras'*. R package version 2.2.5.0.
URL: *https://CRAN.R-project.org/package=keras*

Allaire, J. and Tang, Y. (2019), *tensorflow: R Interface to 'TensorFlow'*. R package version 2.0.0.
URL: *https://github.com/rstudio/tensorflow*

Almahdi, S. and Yang, S. Y. (2017), 'An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown', *Expert Systems with Applications* **87**, 267–279.

Almahdi, S. and Yang, S. Y. (2019), 'A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning', *Expert Systems with Applications* **130**, 145–156.

Bahdanau, D., Cho, K. H. and Bengio, Y. (2015), Neural machine translation by jointly learning to align and translate, *in* '3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings'.

Banerjee, P. S., Doran, J. S. and Peterson, D. R. (2007), 'Implied volatility and future portfolio returns', *Journal of Banking & Finance* **31**(10), 3183–3199.

Barberis, N. and Thaler, R. (2003), 'A survey of behavioral finance', *Handbook of the Economics of Finance* **1**, 1053–1128.

Becker, R., Clements, A. E. and White, S. I. (2006), 'On the informational efficiency of s&p500 implied volatility', *The North American Journal of Economics and Finance* **17**(2), 139–153.

Ben-Hur, A., Horn, D., Siegelmann, H. T. and Vapnik, V. (2001), 'Support vector clustering', *Journal of machine learning research* **2**(Dec), 125–137.

Bengio, Y., Simard, P. and Frasconi, P. (1994), 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166.

Bikhchandani, S. and Sharma, S. (2001), 'Herd behavior in financial markets', *IMF Staff Papers* **47**, 1–1.

Black, A. J., Klinkowska, O., McMillan, D. G. and McMillan, F. J. (2014), 'Forecasting stock returns: do commodity prices help?', *Journal of Forecasting* **33**(8), 627–639.

Borowiec, S. (2016), 'Alphago seals 4-1 victory over go grandmaster lee sedol', *The Guardian* **15**.

Boudt, K., Paulus, E. C. and Rosenthal, D. W. (2017), 'Funding liquidity, market liquidity and ted spread: A two-regime model', *Journal of Empirical Finance* **43**, 143–158.

Breiman, L. (1996), 'Bagging predictors', *Machine learning* **24**(2), 123–140.

Breiman, L. (2001), 'Random forests', *Machine learning* **45**(1), 5–32.

Cai, X., Hu, S. and Lin, X. (2012), Feature extraction using restricted boltzmann machine for stock price prediction, *in* '2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)', Vol. 3, IEEE, pp. 80–83.

Caliskan, D. and Najand, M. (2016), 'Stock market returns and the price of gold', *Journal of Asset Management* **17**(1), 10–21.

Cambria, E., Fu, J., Bisio, F. and Poria, S. (2015), Affectivespace 2: Enabling affective intuition for concept-level sentiment analysis, *in* 'Twenty-ninth AAAI conference on artificial intelligence'.

Carhart, M. M. (1997), 'On persistence in mutual fund performance', *The Journal of finance* **52**(1), 57–82.

Chandrinos, S. K. and Lagaros, N. D. (2018), 'Construction of currency portfolios by means of an optimized investment strategy', *Operations Research Perspectives* **5**, 32–44.

Chandrinos, S. K., Sakkas, G. and Lagaros, N. D. (2018), 'Airms: A risk management tool using machine learning', *Expert Systems with Applications* **105**, 34–48.

Chen, N.-F., Roll, R. and Ross, S. A. (1986), 'Economic forces and the stock market', *Journal of business* pp. 383–403.

Chen, S. and Ge, L. (2019), 'Exploring the attention mechanism in lstm-based hong kong stock price movement prediction', *Quantitative Finance* **19**(9), 1507–1515.

Chen, Y. and Hao, Y. (2017), 'A feature weighted support vector machine and k-nearest neighbor algorithm for stock market indices prediction', *Expert Systems with Applications* **80**, 340–355.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), 'Learning phrase representations using rnn encoder-decoder for statistical machine translation', *arXiv preprint arXiv:1406.1078* .

Covel, M. W. (2009), *The complete TurtleTrader: how 23 novice investors became overnight millionaires*, Harper Collins.

De Bondt, W. F. and Thaler, R. (1985), 'Does the stock market overreact?', *The Journal of finance* **40**(3), 793–805.

DellaVigna, S. and Pollet, J. M. (2009), 'Investor inattention and friday earnings announcements', *The Journal of Finance* **64**(2), 709–749.

Fama, E. F. (1995), 'Random walks in stock market prices', *Financial analysts journal* **51**(1), 75–80.

Fama, E. F. and French, K. R. (1993), 'Common risk factors in the returns on stocks and bonds', *Journal of Financial Economics* .

Fischer, T. and Krauss, C. (2018), 'Deep learning with long short-term memory networks for financial market predictions', *European Journal of Operational Research* **270**(2), 654–669.

French, K. R. (1980), 'Stock returns and the weekend effect', *Journal of financial economics* **8**(1), 55–69.

Freund, Y., Schapire, R. E. et al. (1996), Experiments with a new boosting algorithm, *in* 'Proceedings of the 13th International Conference on Machine Learning', Vol. 96, Citeseer, pp. 148–156.

Friedman, J., Hastie, T. and Tibshirani, R. (2000), 'Special invited paper. additive logistic regression: A statistical view of boosting', *Annals of statistics* pp. 337–374.

Friedman, J., Hastie, T. and Tibshirani, R. (2001), *The elements of statistical learning*, Vol. 1, Springer series in statistics New York.

George, G., Haas, M. R. and Pentland, A. (2014), 'Big Data and Management', *Academy of Management Journal* .

Gers, F. A., Schmidhuber, J. and Cummins, F. (2000), 'Learning to forget: Continual prediction with LSTM', *Neural Computation* .

Graves, A. and Schmidhuber, J. (2005), 'Framewise phoneme classification with bidirectional lstm and other neural network architectures', *Neural networks* **18**(5-6), 602–610.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. and Schmidhuber, J. (2016), 'Lstm: A search space odyssey', *IEEE transactions on neural networks and learning systems* **28**(10), 2222–2232.

Grossman, S. J. and Stiglitz, J. E. (1980), 'On the impossibility of informationally efficient markets', *The American economic review* **70**(3), 393–408.

Hastie, T., Tibshirani, R., James, G. and Witten, D. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Hau, H. and Rey, H. (2006), 'Exchange rates, equity prices, and capital flows', *The Review of Financial Studies* **19**(1), 273–317.

Heaton, J., Polson, N. and Witte, J. H. (2017), 'Deep learning for finance: deep portfolios', *Applied Stochastic Models in Business and Industry* **33**(1), 3–12.

Hinton, G. E. and Salakhutdinov, R. R. (2006), 'Reducing the dimensionality of data with neural networks', *science* **313**(5786), 504–507.

Hochreiter, S. (1998), 'The vanishing gradient problem during learning recurrent neural nets and problem solutions', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* **6**(02), 107–116.

Hochreiter, S. and Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hsu, M.-W., Lessmann, S., Sung, M.-C., Ma, T. and Johnson, J. E. (2016), 'Bridging the divide in financial market forecasting: machine learners vs. financial economists', *Expert Systems with Applications* **61**, 215–234.

Huang, W., Mollick, A. V. and Nguyen, K. H. (2016), 'Us stock markets and the role of real interest rates', *The Quarterly Review of Economics and Finance* **59**, 231–242.

Huck, N. (2019), 'Large data sets and machine learning: Applications to statistical arbitrage', *European Journal of Operational Research* **278**(1), 330–342.

Jacobs, B. I. and Levy, K. N. (1993), 'Long/short equity investing', *Journal of Portfolio Management* **20**(1), 52.

Jegadeesh, N. (1990), 'Evidence of predictable behavior of security returns', *The Journal of finance* **45**(3), 881–898.

Jensen, M. C. (1978), 'Some anomalous evidence regarding market efficiency', *Journal of financial economics* **6**(2/3), 95–101.

Jones, C. M. and Kaul, G. (1996), 'Oil and the stock markets', *The journal of Finance* **51**(2), 463–491.

Keim, D. B. (1983), 'Size-related anomalies and stock return seasonality: Further empirical evidence', *Journal of financial economics* **12**(1), 13–32.

Kilian, L. and Park, C. (2009), 'The impact of oil price shocks on the us stock market', *International Economic Review* **50**(4), 1267–1287.

Kim, S.-H. and Kim, D. (2014), 'Investor sentiment from internet message postings and the predictability of stock returns', *Journal of Economic Behavior & Organization* **107**, 708–729.

Krauss, C., Do, X. A. and Huck, N. (2017), 'Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500', *European Journal of Operational Research* **259**(2), 689–702.

Liu, Y. and Liu, X. (2018), A trend-based stock index forecasting model with gated recurrent neural network, *in* '2018 IEEE International Conference on Progress in Informatics and Computing (PIC)', IEEE, pp. 425–429.

Lo, A. W. (2004), 'The adaptive markets hypothesis', *The Journal of Portfolio Management* **30**(5), 15–29.

Loughran, T. and McDonald, B. (2011), 'When is a liability not a liability? textual analysis, dictionaries, and 10-ks', *The Journal of Finance* **66**(1), 35–65.

Lux, T. (1995), 'Herd behaviour, bubbles and crashes', *The economic journal* **105**(431), 881–896.

Malkiel, B. G. (2007), *A random walk down Wall Street: The time-tested strategy for succesful investing*, WW Norton & Company.

Malkiel, B. G. and Fama, E. F. (1970), 'Efficient capital markets: A review of theory and empirical work', *The journal of Finance* **25**(2), 383–417.

Murphy, J. J. (1999), *Study Guide to Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*, Penguin.

Olah, C. (2015), 'Understanding LSTM Networks [Blog]', *Web Page* .

Pai, P.-F. and Lin, C.-S. (2005), 'A hybrid arima and support vector machines model in stock price forecasting', *Omega* **33**(6), 497–505.

Picasso, A., Merello, S., Ma, Y., Oneto, L. and Cambria, E. (2019), 'Technical analysis and sentiment embeddings for market trend prediction', *Expert Systems with Applications* **135**, 60–70.

Pring, M. J. (2014), *Technical Analysis Explained. The Successful Investor's Guide to Spotting Investment Trends and Turning Points*, McGraw-Hill, New York.

R Development Core Team, R. (2011), *R: A Language and Environment for Statistical Computing*.

Renault, T. (2017), 'Intraday online investor sentiment and return patterns in the us stock market', *Journal of Banking & Finance* **84**, 25–40.

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* **65**(6), 386.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), 'Learning representations by back-propagating errors', *nature* **323**(6088), 533–536.

Salakhutdinov, R., Mnih, A. and Hinton, G. (2007), Restricted boltzmann machines for collaborative filtering, *in* 'Proceedings of the 24th international conference on Machine learning', pp. 791–798.

Samuel, A. L. (1959), 'Some studies in machine learning using the game of checkers', *IBM Journal of research and development* **3**(3), 210–229.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al. (2016), 'Mastering the game of go with deep neural networks and tree search', *nature* **529**(7587), 484.

Simon, H. A. (1955), 'A behavioral model of rational choice', *The quarterly journal of economics* **69**(1), 99–118.

Sirignano, J. and Cont, R. (2019), 'Universal features of price formation in financial markets: perspectives from deep learning', *Quantitative Finance* **19**(9), 1449–1459.

Tibshirani, R. J. and Efron, B. (1993), 'An introduction to the bootstrap', *Monographs on statistics and applied probability* **57**, 1–436.

Ulrich, J. (2019), *TTR: Technical Trading Rules*. R package version 0.23-5.
URL: *https://CRAN.R-project.org/package=TTR*

Whaley, R. E. (2000), 'The investor fear gauge', *The Journal of Portfolio Management* **26**(3), 12–17.

Wikipedia (2020), 'Machine learning — Wikipedia, the free encyclopedia', http://en.wikipedia.org/w/index.php?title=Machine%20learningoldid=944198378.

Yoo, P. D., Kim, M. H. and Jan, T. (2005), Machine learning techniques and use of event information for stock market prediction: A survey and evaluation, *in* 'International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)', Vol. 2, IEEE, pp. 835–841.

Zarei, A., Ariff, M. and Bhatti, M. I. (2019), 'The impact of exchange rates on stock market returns: new evidence from seven free-floating currencies', *The European Journal of Finance* **25**(14), 1277–1288.

Zou, H. and Hastie, T. (2005), 'Regularization and variable selection via the elastic net', *Journal of the royal statistical society: series B (statistical methodology)* **67**(2), 301–320.