POLITECNICO DI TORINO

Master of Science's Degree in Management Engineering

Master of Science's Degree Thesis

Exploring multiple data granularities for multiday stock price forecasting

Supervisors

Prof. Luca CAGLIERO

Prof. Jacopo FIOR

Candidate

Davide BAZZIGOTTI

2019/2020

Abstract

Considering a multiday system based on classification models that analyses historical data from top 100 NASDAQ stocks with daily granularity, the research evaluates the portability of the system to an intraday context, identifies the most profitable data granularity and explore the potential of different classification models. In particular, the goal of the thesis is to understand if it is more convenient, both in terms of predictability and profitability, predicting and trading in intraday or in multi day scenario. In order to do so, several Machine Learning Classification models have been trained on data with different resolution and with different target variables, and eventually evaluated their statistical performance metrics (accuracy, f1-score). Following, the models that have achieved highest predictability score are tested through an automated trading system in order to assess the applicability of the trading system in a real context, through the analysis of the equity line produced by the backtesting of the system. Several variables, in addition to granularity and time to predict ahead, have been tested during the experiments, such as whether including technical indicators, the windows size and whether to predict and trade in the first observations of the day. Experiments have shown that in most cases intraday trading models are able to predict more accurately the movement of the stock price and by using an appropriate trading strategy they also achieve higher profitability results respect day and multi day trading.

Acknowledgements

It has been a long path, but it seems that is finally terminating. It has not always been a smooth and easy path, I encountered several obstacles throughout this journey, but from these I have learnt and grew a lot. However, I had many achievements and satisfactions which I will never forget. Over these 2 year and an half of studies I had the chance to do several different experience where I encountered amazing people that fully deserve a thank. First of all, I want to thank the person who allowed me to do all these experiences: my dad. He has always been supportive and without him I would never done anything, such as moving to another city to study and the Erasmus experience. Although some difficulties, you dad have always been there. I will never stop thanking you for this.
Then I would like to thank my best friends S. and C. with whom I have studied since my bachelor. We have shared many experiences together, both at university and elsewhere. You have always been supportive and company at all times. Without your support I do not know if I would have finished university. Thank you for all my dear friends. I cannot forget the best experience which I did during my Master: the exchange at Delft University of Technology. First thanks POLITO for allowed me this experience, I think it has been the experience where I grew up the most, understood my interests and met amazing people. A big thanks to G., L., M., F. and G.. You guys are amazing people, I did not think it was possible to build such a strong friendship in such a short time.
I have to thanks my neighborhood friends whom I know most of them since kindergarten. It's nice that we've managed to maintain this friendship over the years, even though I've spent little time in Bologna in the last years. Thanks 40138 crew (N, M, I, N, M, F, E, N, I).
In the end, I just want to thanks whoever I missed (sorry), whoever has supported, pushed, hugged but even criticized and teased. You are all part of this experience and helped me grow and fight to reach my goals.
Thank you to all.

Table of Contents

List of Tables

List of Figures

Acronyms

ML
Machine Learning

EMH
Efficient Market Hypothesis

MLP
Multiple Layer Perceptron

DT
Decision Tree

SVM
Support Vector Machine

XGB
eXtreme Gradient Boosting

HO
Hold-out

EW
Expanding Window

WF
Walk Forward Boosting

WS
Window Size

Chapter 1

Introduction to quantitative trading

Since the settlement of the stock market, individual, firms and academic researchers have attempted to predict the price movement of the stock in the near and in the distant future. Several techniques can be used to forecast the price movements of an asset, some more mathematical based and other more expertise knowledge based. Originally most of the trading strategies were generated by an experienced trader who attempted to individuate patterns and trends in the chart by studying the chart and understanding news and micro/macro indicators. These 'manual' techniques are still used in the market, but with the exponential growth of technology and computational power more sophisticated and complicated techniques based on automated trading strategy have spread to all markets. Currently it is estimated that around 60-73% of the volume exchanged in US traded stocks are performed by algorithm in computer machine [1]. Using an automated trading strategy entry and exit points are generated by applying mathematical and statistical models to past data in order to foresee the next movement in the price. The subject which studies and applies advanced mathematical and statistical techniques to the market in order to foresee the movement of the price is called 'quantitative finance'. Designing the model and applying it on traditional trading platforms generates an automated trading system.
In the last decades machine learning algorithms have evolved and spread in all fields and in the financial field have proven compelling results. Indeed, many researches on forecasting stock price prediction through machine learning techniques have flourished in the last few years. Nowadays, most of the researches on forecasting, not only in the financial field, are focused on machine learning techniques. Most of the available researches on stock price forecasting attempt to forecast the movement of the underlying stock in the next day, few days or month. Most research in the field of stock price forecasting are focused on predicting the price in the medium long-term, on the next day or further. Just in the last few years some papers have

been published studying forecasting system in the short-term, from few seconds to hourly predictions. However, it is not simple to carried out researches on short term forecasting because consistent high frequency data is hard to retrieve, this data are usually sold by brokerage firms and more the granularity is high higher is the price. Another obstacle is that the streaming stock data may arrive faster than model's prediction; a model that takes more than one minute to forecast next one minute price is useless. Furthermore there is little incentive to publish working models with good predictability and profitability because they can be easily sold to financial firms or use individually. If published, the competitive advantage can disappear, since the inefficiency spotted from these models would be rapidly exploited by some other traders, thus adjusting the price level. Widespread adoption of a particular trading strategy is enough to drive the price either up or down enough to eliminate the pattern.

The goal of the thesis is to examine and compare the predictability of the stock market return in the short, considered as intra-day, and in the long term, multi-day predictions. In particular the scope is to understand if it would be better, in term of profitability, to use an automatic trading system based on machine learning algorithms in the short term or in the long term. Indeed, the research question of the thesis: *is the market more predictable in the short or in the long term?*

However, the market can be not predictable, completely at random. Indeed, according to the famous Efficient Market Hypothesis (EMH) [2], supported by many studies and researchers, the share price is always the right one because it reflects instantaneously all the information. Future price is independent by the past, and thus any attempt to predict future prices is futile. Nevertheless, several studies have proven profitability, such as [3],[4] and found some form of inefficiency in the market, thus refuting EMH or claiming that weak efficient market hypothesis can be reached on some particular conditions. Still, only few people have shown consistent profitability in a long period of time, and the debate if EMH is valid or not is still open in the academic literature. This research will attempt as well to understand if some inefficiency in the the market can be found and in particular if it is easier to find inefficiency in the short-term or in the long-term.

There are mainly three methods for predicting stock prices and therefore directions:

- fundamental analysis,
- technical analysis
- sentiment analysis.

Fundamental analysis examines the economic factors, both macro and micro, which can somehow influence the price of the stock and reveal the 'fundamental' value of the company. For instance, health of the country economy where the company's

main activities are performed, financial reports, competitor performance, company's balance sheet and many other factors are taken into account. These factors can help to determine if the value of the stock in consideration is 'fair', and thus for example if these factors indicate that the company (stock price) is undervalued the best action would be to buy some shares. Instead, technical analysis focuses on identify past patterns from past data, mainly based on price and volume, to evaluate stock's strength or weakness. Finally, sentiment analysis identifies subjective information and affecting states mainly on social media and specialized websites attempting to understand how people react to certain news. The idea behind sentiment analysis is that the market is driven by the 'sentiment' of the people, this in particular can be very exploitable in the short-term [5].

1.1 Trading Fundamentals

Trading refers to the act of buying, selling, or exchanging stocks, bonds or currency. This work focus on the stock market, which is the market where mainly publicly-held companies' shares can be exchanged. Trader's goal is to maximize his profit by exchanging securities in the market. When a trader opens a position can either go 'long' or 'short', he is basically betting how the price of the stock will move in the future and exploit the difference of the price. Namely, going long means that the trader is speculating that the price of the stock will increase and thus buying it now in order to sell it in the near future. Instead, going short is the opposite, the trader bets that the price will decrease, and so he rents temporarily the underlying stock and sell it to the market. When and if the price drops, the stock is bought back in order to return it and making profit by the sell-buy difference. Traders apply trading strategies which help them to understand which assets to trade, the entry and the exit points and the money management rules. The most important thing of a trading strategy is to generate buy and sell signals that are used to evaluate possible entry and exit points in the market. Starting from how traders create trading strategies over the years have been created automated system based on analytical and statistical techniques which are able to modeled the problem and create a trading strategy. Quantitative trading consists on applying mathematical and statistical based techniques in order to identify trading opportunities in the market. The main advantages of quantitative trading respect a manual human trading is the exploitation of computer power which allows to analyze big amount of information and do complicated mathematical operations in a few thousandths of seconds.
This research will use quantitative trading approach to create a trading system able to generate buy and sell signals which are the most important characteristic of any trading strategies because they give you information about the entry and exit point in the market, that can be exploited to make profit. However, other variables should be taken into account in a complete trading strategy, such as money management. Money management is a strategic technique to maximize the

return based on how much you spend and minimizing the risk. This includes select how much to invest, tracking, budgeting and taxes evaluation. The lack of a money management strategy can make a potentially profitable strategy unprofitable. This research will not dwell too much on money management techniques since the thesis focuses more on applying classification models to stocks data in order to return buy and sell signals. However, the trading system used and implemented includes some parts of money management, such as portfolio construction, trading strategies and risk management, as it will be described in more detail in Ch. 4.5.

1.2 Trading System Validation

A trading system, or automated trading system, is basically a set of operating rules that a trader or investor adopts as a strategy to operate in financial markets using algorithms. The main advantage of the use of trading systems is to eliminate, even if not completely, the influence of the psychological component of the trader, which can negatively affect the process of analysis and implementation of trades.
Trading system performance should be properly evaluated in order to assess if the trading system is able to generate profit consistently and in data never seen before. If the trading system performance are measured in data already seen by the model, i.e in training data, the results are corrupted and very likely higher than the real performance. One of the method applied to properly measure the performance of a trading system is the backtesting. In the context of time series forecasting, the concept of backtesting refers to the process used to establish the accuracy of a forecast method from existing historical data. It is an iterative process, which is repeated for several dates in the historical data. Backtesting is used to predict the accuracy of a forecast method, and is therefore useful in determining which model can be considered the most accurate. Backtesting is generally an intensive process in terms of computing power, as a new forecasting model has to be trained for each test in a different time period. One of the typical mistake is to train the forecast model only once in all the available data and then apply the backtesting method on data that has already been used to train the model. However, if future data is made available to the forecasting model, the model, regardless of the variable to be forecast during the training phase, will inevitably include some information about this future. Consequently, the accuracy measured with backtesting will not reflect the model's generalizing capabilities, but the model's storage capabilities, i.e. the ability to reproduce situations identical to those found in the data set used for training. The abuse of backtest can easily lead to the famous overfitting problem, which indeed occurs when a statistical model has excessively adapted to the data provided to it and thus loses its ability to generalize on new data. Overfitting problem will be discussed in more detail in the next chapter.
Another relevant variable in a trading system and in its performance evaluation is the equity line. It is a chart that shows the trend of the gains and loss to understand in a simple and intuitive way the constancy and reliability of the trading system. In

order to evaluate the quality of the system it is important that the line generated by the chart is a line as constant as possible and that it does not contain inside big negative peaks, in fact the stability of a trading system is one of the characteristics that distinguish the winning systems from the losing ones. Other variables that are fundamental to evaluate the performance of a trading system are [6]:

1. Total Net Profit: total return on capital invested
2. Percent Profitable: number of winning trades divided by the total amount of trades
3. Average Trade Net Profit: total net profit divided by the total number of trades
4. Maximum Drawdown: represents the 'worst-case scenario', the greatest loss from a previous equity peak
5. Total fees: the impact of the fees

All these metrics are essentials to properly evaluate a trading system, we should not focus only on the total net profit generated over a period of time because otherwise we would not be assessing the strategy risk. In particular, the maximum drawdown is a good measure of the reliability and risk of the system.

In the next chapters all the components of a trading system will be described in detail. In particular, in the following chapter, the main element of any trading system, that is the statistical model, in our specific case based on machine learning algorithms, used to generate the forecasting signals.

Chapter 2

Literature studies on Machine Leaning forecasting

In the literature all the researches focused on forecasting the price movement of financial assets are based on discovering information and hidden patterns in the input data. In general, the input data include large amount of information, the so-called 'big data', for instance in the financial field usually include the historical movement of the price of the asset with some other specific information, such as macro/micro economics and news related information. The process of extract information previously unknown from large data sets by applying some mathematical and statistical based techniques such as Machine Learning is called Data Mining [7]. Data Mining is actually included in a broader process, the Knowledge Discovery in Databases (KDD), which has the aim to extract the knowledge from the data and interpret it. KDD also includes the choice of how process and sample the input data. In the following, all the typical steps involved in the KDD are briefly explained [8]:

1. Selection: understanding the application domain and select the most suitable data set.

2. Preprocessing: cleaning the data in order to increases the quality of the data by managing data inconsistency and missing data.

3. Transformation: representation of data in an appropriate way in relation to the objectives of the research. Reduction in size and use of processing methods to reduce the actual number of variables to be fed to the research process.

4. Data Mining: this step includes the choice of the data mining task (classification, regression), then the selection of the most suitable data mining algorithms based on the task and finally the application of the algorithm on the processed and transformed data.

5. Interpretation/evaluation: evaluated whether the objective is achieved, and if it has not been achieved, the previous step and sometimes others are repeated (and possibly modified).

Data mining is the most important step because it selects the statistical model to apply on the data to extract information. Nowadays, the most used models used in the Data Mining field are based on Machine Learning. Machine Learning is a subset of artificial intelligence and it studies algorithms that learn from data and improve automatically through experience. Machine Learning algorithms build a mathematical model from the data provided (the so-called training data) with the aim of building a model that is able to understand the underlying process from which data is derived [9]. In the end the model will be able with a certain degree of error to make predictions on unseen situations and data derived from that process. Nowadays machine learning models are used in a wide variety of applications, the most common and known are email filtering and computer vision tasks.
There are different types of machine learning algorithms which differ on the type of data given as input and their output. The two main popular machine learning problems are Supervised Learning and Unsupervised Learning. Supervised Learning consists on building a mathematical model of the data given as input learning from the actual output that is provided to the model and thus improving over time. On the other hand, Unsupervised Learning algorithms takes a sample of data which contains input and not output and exclusively from the input data they attempt to understand underlying structure on the data, like clustering. This thesis focuses on Supervised Learning algorithms, since the task under studying, forecasting stock price movement, it can be easily designed for a supervised task, because historical data contains both the inputs, stock price and other information, and the output, which represents the price on the next instant.
In the following the most popular and promising Machine Learning Supervised algorithms are explained in detail.

2.1 Supervised Learning Classification Algorithms

Supervised Learning consists on building a mathematical model of the data given as input learning from the actual output that is provided to the model and thus improving over time. Within Supervised Learning algorithms there are mainly two types of problem classification and regression. In classification the goal is the predict the categorical class labels of new instances based on past observations, while regression problems predict continuous value. For instance the stock's price forecasting problem can be approached both with classification and regression algorithm. If a classification algorithm is used the model predicts only the movement of the stock's price, simply by telling if the price in the next timestamp will increase, decrease or remain stable (3 classes). If we think the same task as regression problem, the algorithm will not just tell you in which direction the price will move, as in

classification problem, but it will attempt to give an estimation of the price value in the next observation. In the field of price forecasting, literature has empirically proven that classification algorithm are more effective than regression models. For this reason this research will focus on classification algorithms, as we are going to see afterward.
In the following some of the most popular and effective classification models are briefly explained, in order to have an idea how they work and their inner differences. In literature there are several machine learning classification algorithms available, however here are reported and explained only the ones we have selected and effectively used in our empirical research which it is explained in more detail in Chapter 4.

2.1.1 Decision Tree

A decision tree (DT) is a system with n variables in input and m variables in output. The input variables (attributes) are derived from observation of the environment [10]. Instead, the output variables identify the action to be taken. The decision process is represented with an inverted logical tree where each node is a conditional function. Each node verifies a condition on a particular property of the environment (variable) and has two or more branches downwards in function [11].
The process consists of a sequence of tests. It always starts at the root node, the parent node located higher up in the structure, and proceeds downwards. Depending on the values detected in each node, the flow takes one direction or another and proceeds progressively downwards.
The principle according to which the algorithm divides the various tree nodes into several branches is critical for the accuracy of the algorithm. It is different if is a regression or a classification problem. According to the available criteria the most popular approaches are based on the following metrics: Gini Index, Chi-Square and Information Gain.
Decision trees have the advantage of interpretability. They are easy to understand and execute. Compared to neural networks, the decision tree is easily understood by human beings. Therefore, it is possible to verify how the machine comes to the decision. However decision tree representation is not very suitable for complex problems, because the space of hypotheses becomes too large. The spatial complexity of the algorithm could become prohibitive.

2.1.2 Support Vector Machine

Support Vector Machine (SVM) were first introduced to solve a two class classification problem by looking at the plane which can separate two type of object optimally [12]. Currently, it can work in multiple classes problem finding a nonlinear function that separates the classes by maximizing the margin between this line and the sample points that are closest to the hyperplane. Hence the objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional

space that effectively classifies the data points in the correct classes. There are several possible hyperplane which can separate the classes of data points with a function, however SVM attempts to find the hyperplane in which the distance between data points of different classes is maximized, the so called margin distance. The crucial points that affect the most SVM ability to separate classes are the so-called support vectors, that are the closest data points to the hyperplane and they influence hyperplane orientation and position.

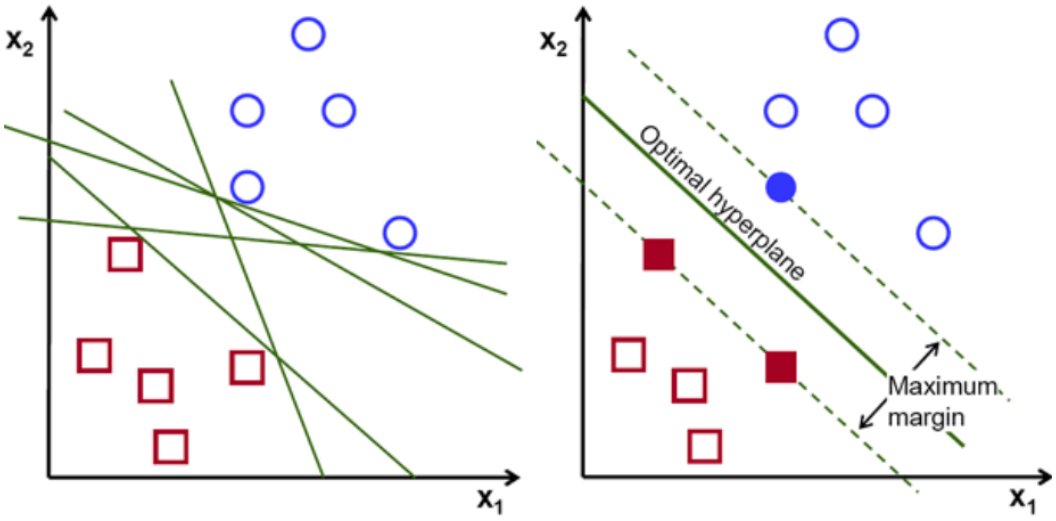


Figure 2.1: SVM example, source: https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f

SVM have shown in several studies that perform well in separating classes, and for this reason are used for different tasks. However, it often happens that SVM builds complex hyperplane in a high dimensional space which help to separate the classes but this increases the model complexity. In these cases SVM algorithms can take a long time to be trained.

2.1.3 Artificial Neural Network

Artificial Neural Network (ANN) are computational systems inspired by biological processes of neural network on human brain. As a real neural network, and ANN is based on an interconnection of neurons, called artificial neurons. The connection are inspired to the synapses in a biological brain which transmits signal to other neurons. When the neuron receives a signal it process it and propagates it to the other interconnected neurons [13].
An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the

synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. In ANN the artificial neuron is a computational unit composed of:

- a set of synapses or connections each of which is characterized by a weight (synaptic efficacy); unlike the human model, the artificial model can have both negative and positive weights;
- a threshold or bias value that has the effect, depending on its positivity or negativity, to increase or decrease the net input to the activation function. Actually this value is optional, but it is used in most of the cases.
- a summation summing the input signals weighed by the respective synapses, producing in output a linear combination of inputs;
- an activation function to limit the output amplitude of a neuron. Typically for convenience the output range [0.1] or [-1,1].

Artificial Neural Network are composed of one or more artificial neuron interconnected and as we have seen, each neuron gets one or multiple input which then combines and finally through an activation function produce an output which will be forwarded to the following artificial neurons.

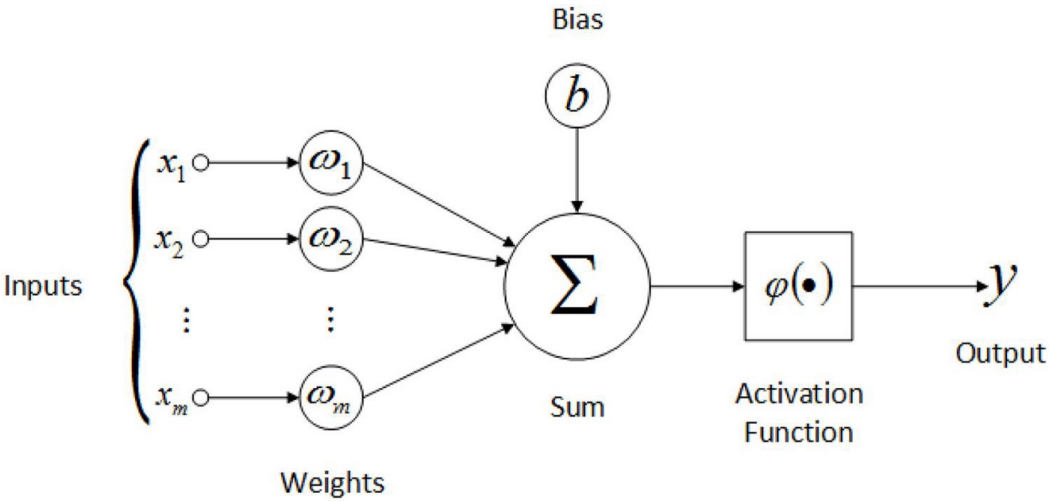


Figure 2.2: Artificial Neural Network structure, source: https://medium.com-/jayeshbahire/

The perceptron is the simplest form of neural network used to classify linearly separable patterns, that are patterns that stand on opposite sides of a hyperplane. It consists of a single neuron with adaptable symptic weights and threshold.
In a typical ANN the neurons are interconnected into multiple layers. Neurons of one layer connect only with neurons in the linked layer, namely the previous and

next layer. A layer between two other layer is called hidden layer.
A multilayer perceptron (MLP) is composed of multiple perceptron with a set of inputs (input layers), one or more hidden layers of neurons (hidden layers) and a set of output neurons (output layers) [13]. The input signal propagates through the network forward from layer to layer, for this reason these types of ANN are called feedforward neural network. Such a network has three distinctive features:

- each neuron includes a differentiable nonlinear activation function (e.g.: sigmoidal, relu, logistic);
- the network contains one or more hidden layers that are neither part of the input nor of the network output;
- the network has high connectivity, meaning that the artificial neuron are connected to multiple neurons.

In contrast of a simple perceptron, a MLP can learn a non-linear function thanks to its non linear activation function and multiple layers of neurons. However, more layers are added in a ANN more the complexity of the model increases and so does the training time. An ANN which includes several layers within the input and output layer is referred as Deep Neural Network (DNN) [14]. DNNs have shown great potential in several fields because the high level of complexity helps to learn complex non-linear relationship. Nevertheless, they present some disadvantages, mainly related to the computational time and overfitting. Overfitting occurs when a model learns too much from the input data (training set), drastically compromising its generalization skills because it is biased on the training data. The thesis will not focus on DNN model because, as it will explained in more detail in Chapter 4 and 5, a short term forecasting problem with the input data available is quite prone to overfitting and furthermore a long time of training cannot be accepted if the goal is to predict in the short term.

Learning is a process on which neural network free parameters are adapted, through a stimulation process, to the environment in which it is placed. The type of learning is determined by the way these adaptations occur. A learning algorithm is a set of well-defined rules that solve a learning problem. In a Neural Network the free parameters are the weights and thresholds which are adjusted during the learning process. The adjustments are done with the goal to approximate the obtained output to the desired. The difference between the generated output and the desired is the error signals which has to be minimized. The learning process continue as long as the error signal keeps reducing and finished only when the error stays constant even after examining additional observations. This process takes place one step at a time minimizing a cost function. The most used optimization method is the gradient descent or Widrow-Hoff method. This is typically used for feedforward neural network, such as MLP, and the optimization method is called Backpropagation. Basically Backpropagation method computes the gradient of the

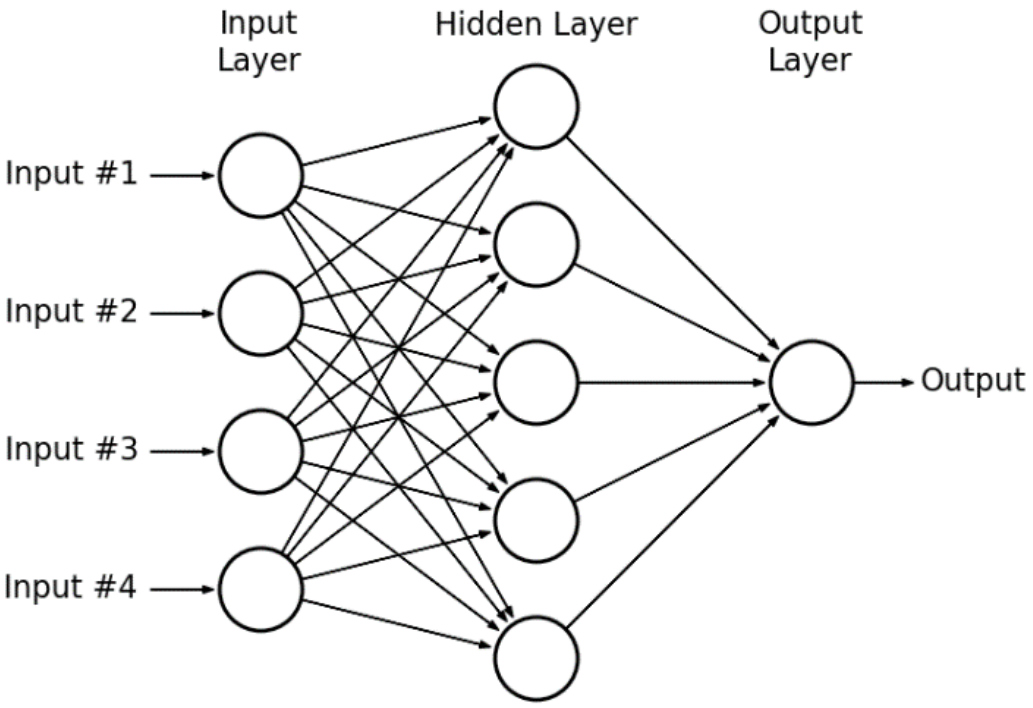


Figure 2.3: A hypothetical example of Multilayer Perceptron Network, source: [15]

defined cost function for a given states and then the weights are updates according to these values.

In the financial field ANN have shown great potentiality, such as in [16], [17], [18] . In these studies ANN are trained in financial time series data with the aim to either predict the price or the movement (up or down) of the next observation.

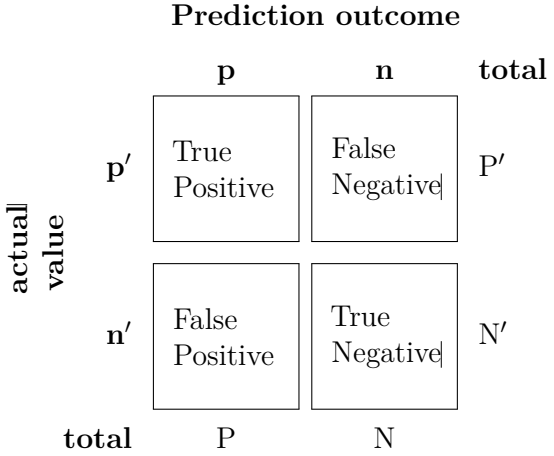
2.1.4 XGBoost

eXtreme Gradient Boosting (XGBoost) is a popular and efficient open source implementation of the gradient boosting tree algorithm. Gradient boosting (GBM) algorithm is based on the research by Friedman in 1999 [19]. It is a supervised learning algorithm that attempts to accurately predict a target variable by combining estimates from a set of simpler and weaker models, typically decision trees.
XGBoost minimizes a regularized objective function that combines a convex loss function (based on the difference between expected and target outputs) and a penalty term of model complexity. The process proceeds in an iterative way, adding new trees that predict the residual values or errors of the previous trees, which are then combined with the previous trees for the final prediction. This technique is called gradient boosting because it uses a gradient descent algorithm to minimize

losses when adding new models. The idea behind this algorithm is that combining (ensembling) simple models, such as decision trees, that individually would perform poorly, leads to better performance, because each decision tree aims to decrease the error of the previous model.

2.2 Evaluation metrics

Following a typical machine learning pipeline, once all the training models have been trained are then tested on an holdout dataset in order to identify which one of the trained models has the highest predictability performance. There are some statistical related metrics, such as accuracy,precision, recall, which measure how good is the model to predict by comparing the predicted results in the hold-out dataset with the real ones [20]. This approach takes into account only the predictability performance of the model, it does not tell us nothing about the profit and losses obtained applying this model on the market. In order to measure the profitability of the model a trading simulation has to be carried out. The trading simulation takes in input the buy and sell signals created by the model and simulate a real trading experience and eventually returns the gain and losses for each transaction.
In statistical classification the most used method to visualize the performance of the model is through a confusion matrix. The confusion matrix is a straightforward table where each row of the matrix represents the occurrences in a predicted class while each column represents the occurrences in an actual class.



The table above shows an example of a Confusion Matrix for a classification problem with two classes either 0 or 1 (which could be though in thesis' case to 'sell' and 'buy'). In this case, as you can see, we have four different measures:

1. True Positive (TP): total number of observation correctly classified as true
2. True Negative (TN): total number of observation correctly classified as false

3. False Positive (FP): total number of observation incorrectly classified with true

4. False Negative (FN): total number of observation incorrectly classified with false

These four measures allow to retrieve more detailed analysis of the performance of the classification algorithm. Two of the most used metrics derived are accuracy and error rate. As we can see in 2.1 Accuracy is the ratio of correct predictions to the total number of instances, while error rate is simply the total number of incorrect prediction over total predictions. Accuracy is a good metric to use when the dataset is balanced, but when the data is imbalanced is not appropriate and it can easily lead to misleading conclusions. For instance, assuming that we have a highly imbalanced dataset, in which 97% of classes are 'not to trade' and the other 3% are 'trade'. In this case, by using a simply model that any time predicts 'not to trade' and taking the accuracy as performance metric we would get a 97% which is considered a high performance. However, if we would use that simply model we would never trade (the model always predicts not to trade) and so we would not be able to make any profit. More the undersampled class is important on the case specific problem, more the accuracy metric is deprecated and misleading. Nevertheless, there are other metrics, such as Precision, Recall and F1-score, that are able to tackle this problem and be used for imbalanced dataset and measure the performance of one specific class.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

$$precision = \frac{TP}{TP + FP} \tag{2.2}$$

$$recall = \frac{TP}{TP + FN} \tag{2.3}$$

Precision 2.2 is the percentage of correct positive predictions over all positive predictions, while Recall or also called Sensitivity 2.3, measures the percentage of correct positive predictions over all actual positive instances and finally F1-score is the harmonic mean of precision and recall. In other words, precision measures how many of the predictions that the model makes are accurate, on the other hand, recall takes into account also the false negative thus measuring how good is a model in predicting the positive case respect all the observed positive case. To fully evaluate the performance of a classification model both precision and recall should be examined. Unfortunately, the two measures are often in contrast: improving recall typically reduces precision and vice-versa. In general if both of the two

measures are essential, a trade-off between the two is done or f1-score metric is used.
In an imbalanced data problem which occurs when one class is over represented in the dataset, both precision and recall would be a better metric to measure the actual predictability of the model, because they are able to measure how good is the model in identifying the minority class, in that case the 'trade' class. An ideal model for this case would have both an high recall and high precision. But in which case is it better to use Precision and when recall? As we have seen, recall takes into account the False Negative, and thus it gives particular importance on all the cases in which the model predicts negative but it is actually positive. Recall is indeed used in all that scenario in which we have to minimize False Negative, for instance for a rare cancer data modelling predicting that a patient does not have cancer when he actually has it is considered a huge mistake. On the other hand, Precision is suitable, for instance, for Video recommendations, where false negatives are less of concern.

2.3 Model Validation

Model validation is a crucial step of statistical modelling used to assess if the outputs of a statistical model are in line with the real values of the process that has been modeled [21]. There are several model validation methods, however they could be simply gathered in two approaches. One approach uses the data that has been used to construct the model, the so called training set, to validate the model itself. The second approach, instead, uses data that has never seen by the model. Both of them usually measures the goodness of fit of the model (through some metrics, such as the ones explained in the previous section), basically comparing predicted values of the model with the observed values. Validation based exclusivity on the first approach, that is validate only on the training set, has many flaws, and has been proven not adequate for statistical modelling. Indeed, it can easily leads to the so-called overfitting problem, which happens when the model has a strong bias on the training set data. Overfitting is considered one of the biggest problem in the machine learning research because it corrupts the performance results making it look like very high, but in reality the performance would be very poorly on new and unseen data. For this reason, the first approach based on validating the model on training set, is not recommended.
In order to properly evaluate the performance of the model the second approach based on unseen data should be followed. In the following subsections three of the most used methods of model evaluation based on unseen data will be described.

2.3.1 Hold-out

Holdout is the easiest and most used method of validation. In the method the data set is splitted in two parts the training set which is used to train the model and the

test set, used to test the performance of the model. The size of the split is arbitrary and depending by the size of the entire dataset, but it usually requires more data points on the training set rather than the test set, because in general more data you have to train the more the model can generalize on unseen data. A typical ratio between training set and test set is around 70/30 or 80/20. The main advantage of using this method is the efficiency. Indeed, it is usually requires shorter time of training and testing, since not all the data or a combination of the data is used for training. It is all done in two step having fixed the instances in the training and testing set. However, this advantage of a simple and not combinatorial split can also be seen as a drawback. Indeed, splitting the dataset in two parts reduces the amount of data available for training, and thus the model can not 'see' several observations in the test set which could be unique. Another flaws of the hold-out method is that the two dataset created may have a different distribution of the classes. This means that one of the classes could be overpopulated in one of the dataset and in the other no. In this case the evaluation of the model performance can not be very useful. Nevertheless, there are some rules that tries to overcome the described issues, such as randomly split the observations of the whole dataset between training and testing set and check if the distribution of the labels in the two sets is comparable.
As we have seen, holdout method is a simple method which can be very powerful and in particular efficient, however is not considered the best validation method, especially in the case where the dataset is small and imbalanced. Another method, based on the hold-out one, the so-called K-fold cross validation, can be used to overcome some hold-out issues.

2.3.2 K-fold Cross-Validation

Cross Validation is a validation technique based on hold-out method, but more robust. K-fold validation consists on partitioning the entire dataset in K subsets of same size and then performing the training on k-1 subsets which correspond the so-called training set, and test the model in the last remaining fold [22]. Cross-validation is performed k times, each time a different subsets is used as validation, it ends only when all the k subsets have been used as validation. In this way the whole dataset is used both as training and validating. Eventually model prediction performance are assessed by averaging the result of each of the k runnings. This is the main advantage of this technique, since it allows to reduce the performance variability and thus get a more accurate estimation of model's performance. As the number of repeating K increases, the variability is reduced, however it extends the training and testing period. The computational time is a variable that has to be taken into account in statistical modelling, in particular when the aim is to predict a variable in a close temporal instant, such as in intraday training modelling.

2.3.3 Expanding Window

One of the most robust way of testing the ability of a trading system is to use the so-called Expanding Window (EW) or Walk forward optimization (WFO), a method first described in the book "Design, Testing, and Optimization of Trading Systems" by Roberto Pardo [23]. The idea behind this approach comes from how real trader optimize a trading strategy in trading, first by applying it in a part of the available data, the so called 'window', to forecast the next period and then rolling out the window, so that it includes the data previous forecast. In this way at any shift of the window, trading strategy parameters are optimized based on previous results.
In other words, walk forward approach optimize on a training set, tests on a period after the training set and then rolls the training set forward and repeats the process until the end of the data. This approach is a specific application of Cross-validation technique, which is based on using several different out of sample data and in-sample data, in order to use a larger out-of-sample data to test the strategy.
The figure below should clarify the explanation. As it shown, in-sample data, which are used to optimize the strategy/model parameters take into account much more data than the out-of-sample. The robustness of this approach is that at the end of its application all the data are used both as training and validation.

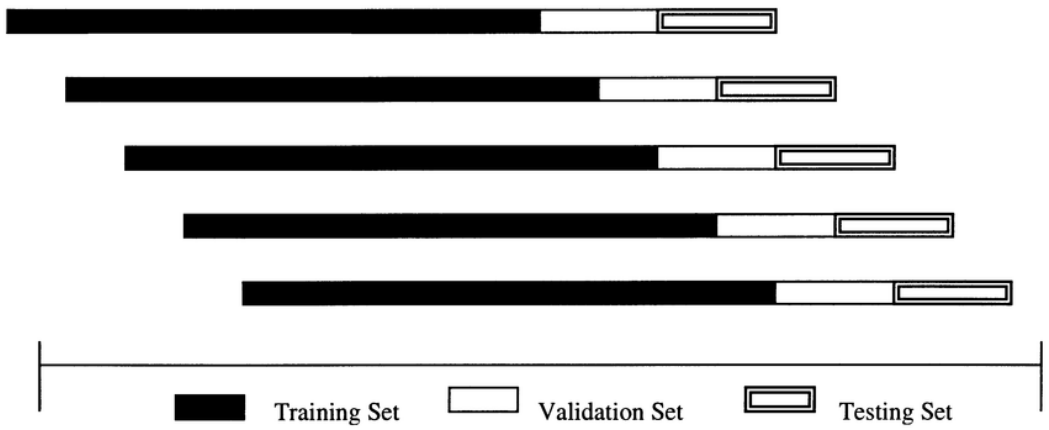


Figure 2.4: Expanding Window/Walk-forward testing example, source: [24]

Chapter 3

Stock price forecasting based on Machine Learning

Trading consists in buying and selling financial instruments (e.g. shares, options, futures, derivatives, currencies) but, depending on the time horizon with which one operates different techniques can be applied. The types of trading respect to the investment time can be divided into four classes of trading types:

1. Scalping
2. Day trading
3. Swing trading
4. Position trading

Scalping can be defined as a technique to operate in intraday and to carry out speculation of very short term exploiting the micro-oscillations of price of the stock. Through this technique, the same stock can be exchanged several time throughout the course of the day, the duration of the trade usually varies from few seconds to some minutes.
Day trading is a technique which executes transactions almost every day and all the operations are closed within the same day, before the closing of the market. Respect the scalping, day trading uses different entry and exit criteria than scalping, for instance based on news, fundamental and technical analysis.
Swing trading is a short term speculation, similar to the day trading, but with a wider time horizon of its trades that can reach up to few weeks. It largely exploits technical analysis based on identifying the trend.
Finally, position trading consists on trade on a long time horizon, in general more than one months to some years. The position trading is not considered speculative trading as the previously explained trading method, but it is perceived as investment. Usually, people uses this approach indirectly through an investment

funds which has the task of building a medium/long-term investment portfolio. This research focuses in particular on the first two trading methods just described: swing trading and day trading.

Nowadays it has become feasible to automate the execution of trading operations in a trading system and brokerage platform. The whole process can be driven by an algorithm that send signals to the Broker's platform so that the trading system will execute the received operation. This system has enabled the development of researches in the intraday trading field. Also this research is mainly focused on intraday trading, indeed its objective is to evaluate the applicability of a trading system in an intraday scenario based on the state-of-the-art of Machine Learning model applied on a daily scenario. The research has the goal to answer the following question: *is it better, in term of predictability and profitability, apply an automated trading system in an intraday or in a multiday scenario? Which one maximize the net final profit?*
In order to understand how to proceed to answer the research question we first need to understand how researches in the literature have approached this problem. These studies can help us to understand which are the best approaches and methods to use and which are the variables that have been less researched. In the following several studies on this field are reviewed and in particular their performances are compared.

3.1 Relevant studies on Intraday and Multi-day Forecasting

One of the most relevant research in Intraday forecasting is 'Forecasting Performance of Nonlinear Models for Intraday Stock Returns' by Matieas et al. [25]. The research focuses on the predictability of intraday stock market applying both linear (simple autoregressive models, smooth transition autoregressive,smooth transition autoregressive with GARCH errors) and nonlinear timeseries models (Markov switching, multilayer perceptron, nonparametric kernel regression and support vector machine models). The peculiarity of this research is that all these models are applied to different time-horizon (of 5, 10, 20, 30 and 60 minutes) and then for each time horizon is evaluated which is the most suitable model to use. As explanatory variable to feed into the models they used the lagged stock logarithmic return variable for the same trading day. All the models used are regressive models so they forecast the one-step-ahead price according to the time temporal horizon considered (5, 10, 20, 30 and 60 minutes). From the trading simulation results was observed that as periods lengthened,the profitability of the intraday investment strategies decreased, indeed ac-cording to the authors, this is due the fact that 'efficiency creation actions of traders expunging any predictability of returns over short periods of time' and then concluding asserting that the weak-form market

efficiency is only reached in a short period of time.
Other studies, such as [26], have studied if integrating financial news has an impact on model's performance, proving that in general if they are added as features, performance increases.
In the day trading scenario, studies such as [4], [27] have been taken as reference and basis, because in these researches are applied the state of the art Machine Learning models in the multday scenario, to understand how to build an intraday system based on methods applied in multiday trading.
Nowadays most of the article in this field apply Deep Learning models, such as Chon et al. in Deep learning networks for stock market analysis and prediction: Methodology, data representations,and case studies [17]. In this studies simple machine learning model, such as linear regression model, are compared to more complicated and state-of-art model, such as Deep Learning model as Multiple Layer Perceptron (MLP) with several layers. From the results they show that most of the times and using different configurations simpler model outperform more complicated ones.

3.2 Intraday versus Multiday techniques

After reviewing and studying in depth several papers concerning multiday and intraday forecasting, some briefly summarized in the previous section, we noticed that there are not significant differences between the methods applied in the two scenarios. Indeed, the same model, input features formatting, data representations and model's evaluations can be applied regardless of the next forecasting period, either short or long. As we have seen in [25] , in which are applied several models with different data representations and features, on stock with different time horizon from 5 minutes to 60 minutes forecasting. What it changes in this case is only the data to feed into the model. While for intraday forecasting the input data in general have an hourly and minute frequency (less than daily up to tick-by-tick data), in multiday forecasting the input is usually represented by daily or weekly information. However, there are some specifics which have to be considered with particular attention when working with short-time forecasting. The three variables individuated that affect the most model's performance in a short term scenario respect a long one are:

1. transaction costs,
2. model's prediction time
3. periodic retraining

For intraday forecasting becomes particular significant evaluate the performance of the model's prediction through a trading simulation and not exclusively through an accuracy metric. It is still important for multiday forecasting, however in intraday trading the variable transaction cost can significantly influence the net profit,

since the market operations for the same time horizon are much greater. In average, we could claim that the transaction costs in brokers platform are around 0.03% per operation. We think that in order to perform a complete study on intraday forecasting, the variable transaction cost and consequently number of operations per day must be measured to evaluate the real performance of the forecasting models.
Furthermore, another variable to take into account is the prediction time of the model, namely how long the model takes to predict the next price. For instance, the model cannot take more than one minute to forecast the next one-minute price, and more the forecasting time is shortened (as in high frequency trading) more this variable becomes relevant. Thus, the model execution time and even its training time should be taken into account. Once all the evaluation metrics are measured, as suggested in [28] , it should be done a trade-off between the performance of the algorithm and its efficiency.
Finally, another variable which has shown to affect the performance of multiday forecasting, but in particular for intraday, is the periodic retraining, which most of the studies do not even take into account or mention. In [29] has shown that model performances are influenced by retraining set size and period of retraining. Indeed, the market conditions change very quickly and it is often important to exclude data which are too far in the past using a rolling window and including more recent information. This is possible by using a walk-forward validation approach as applied in [30]. This validation method incorporates new information as become available, and it can help to have a more realistic view of how effective is the model on this new data.

3.3 Intraday versus Multiday Performances

This subsection attempts to compare the performance of intraday and multiday forecasting of existing studies, in order to understand which one can offer major returns opportunities. Theoretically, short time trading could offer higher annual return respect long-time trading, because if an opportunity occurs in the market traders are ready to exploit it as soon as possible with price quickly adjusted to a new equilibrium level. Thus, we could think that shorter is the time prediction higher are the opportunities to exploit in the market and hence the final profit. However, some studies in multiday trading have shown that are able to consistency create profit, rejecting the Efficient Market Hypothesis. Still, even intraday studies have shown profitability. Despite these opportunities and its economic and financial importance, the analysis of stock return predictability for short forecast horizons is under-represented in the academic literature, although worthy of mention is the studied of [25] . As describer previously, this study has shown that lower is the time horizon forecast higher is the profit in the same period of time. However, this research compares returns obtained in different time horizons, but all in the intraday

scenario. To the best of my knowledge, there are no researches available where is carried out a complete comparison of classification machine learning algorithm on Nasdaq Data with different data granularitiy in order to understand which one offers the best investment opportunity. In the following several studies of intraday and multiday forecasting are compared in term of the techniques, input data, data representations, models applied and finally in particular in performance obtained in the test set. However, it cannot be assumed as a complete and significant statistical comparison of the performances obtained. Indeed, we are comparing performances obtained in different stocks and temporal time, and some of these studies do not perform any trading simulation, only assessing the model's performance through accuracy metrics. However, a scheme comparison can help to understand, in each of the two scenarios, what are the best practices to use and have an indication, not statistically significant, of the returns obtained in the two different trading time prediction horizons. The studies review for intraday is reported in table 3.1, while multi-day studies in table 3.2.

Authors	Data type (Number of input feautures x lagged return)	features type	Target output	Num. Of samples (Training:Validation:Test)	Sampling period	Method	Performance measure	best model and performance in out-of-samples
Matias, Reboredo 2012	US S&P 500 (1 x all the instances of the same trading day)	lagged returns	Stock price in 5, 20, 30, 50 min	6000 5 min ... 350 for 60 min	06/2003 - 09/ 2003	MLP, SVM, KR RW, AR(1), ST(GARCH(1))	MSE, MAE, SIGN, DA trading simulation	KR 30min 5.06%
Geva, Zahavi, 2010	48 S&P500 ~ 395	news data and historical market data	Market direction (up or down)	56000 (2:0:1) 15 min granularity	NA	SLR, CHAID tree, NN	trading simulation avg return above S&P 500	NN 15 min 0.62%
Chong et. Al. 2017	KOSPI market	market data	Market direction (up or down)	58421 (4:0:1), 5 min granularity	04-Jan-2010 to 30-Dec-2014	Feauture selection (PCA, AE, RBM) + AR(10), ANN	NMSE, RMSE, MAE, MI	Raw Data (no feauture selection) + DNN: 70% acc
Khare et al. 2017	10 NYSE stocks	technical indicator, historical prices	Stock price	90000 (5:2:3) 1 min granularity	NA	MLP, LSTM	RMSE	MLP : 2.5*10^-3
Arévalo et al. 2016	AAPL from NYSE	historical prices, technical indicator	Stock price	19110 (20:0:3) 1 min granularity	2/09/08 - 07/11/08	DNN	MSE, DA	MSE: 0.071 DA: 65.2%
Labiad et al. 2016	IAM from CSE	historical prices, technical indicator	Market direction (up or down)	18001 (2:0:1) 10 min	04-03-2008 to 31-12-2015	RF, GBT, SVM	accuracy	RF : 95% GBT: 94% SVM: 90%
Louwerse, Rothkrants, 2014	ING, Fortis and Ahold stocks from the AEX index	historical prices, technical indicator	Market direction (up or down)	NA (7:0:3) NA	02/2008 to 12/2008	ANN	SIGN, MAE and RMSE trading simulation	ING: +21% FORTIS: +53% AHOLD: +25%
Cervellò-Royo et al. (2015)	US Dow Jones index (1 × 10)	technical indicator	Market trend (bull/bear-flag)	91307	to 29-Nov-2013 (15-min)	Template matching	trading simulation	mean of ~ 35%

Figure 3.1: Relevant researches on Intraday Forecasting

In the tables can be observed that in most of the studies analysed shorter is

the time forecasting, higher are the returns. However, as already mentioned, we cannot claim in any way that either multi-day or intraday trading is more profitable. Most of the studies use different metric to evaluate the performance of the trading system, only few of them perform a simulation of the market to get the return on investment. In any case, even if the same techniques of validations and trading simulations were used in all the studies, the results could not be compared because they use different assets and trading periods.

In order to properly evaluate a real and complete comparison and answer the research question of the thesis an empirical research has to be carried out. Trading performances must be evaluated in the same stock, using the same model, and finally tested in the same range of time, but using different time horizons predictions (from few minutes to daily) which is the main point of this research, contrasting intraday and multi-day predictions. In the next chapters the project designs, data pre-processing, evaluation of the performances, and finally the experiments and results are explained in details.

Authors	Data type (Number of input feautures x lagged return)	features type	Target output	Num. Of samples (Training:Validation:Test)	Sampling period	Method	Performance measure	best model and performance in out-of-samples
Enke and Mehdiyev (2013)	US S&P 500 index	historical OHLC	Stock price	361 (2:0:1)	Jan-1980 to Jan-2010 (daily)	Feature selection+ fuzzy clustering + fuzzy NN	RMSE	combination NN + clustering
T.-l. Chen and Chen (2016)	Taiwan TAIEXa and US NASDAQb indice (27 x 20)	historical OHLC	Market trend (bull-flag) in 5, 10, 15 days prediction	3818a, ∗ 3412b, ∗ (7:0:1)	different range time, most recent:02/1997 - 03/2004	Dimension reduction+template matching	trading simulation	NASDAQ 5 day: 168% TAIEX 5 day: 479%
Chiang, Enke, Wu, and Wang (2016)	World 22 stock market indices; ({3~5} × 1)	Technical indicators, lagged return	Trading signal (stock price)	756 (2:0:1)	Jan-2008 to Dec-2010 (daily)	Particle swarm optimization +ANN	trading simulation	mean of ~ 60% in the 22 stocks
Zhong and Enke (2017)	US SPDR S&P 500 ETF (SPY); (60 × 1)	financial indicators, lagged return	Market direction (up or down)	2518 (14:3:3)	to 31-May-2013 (daily)	Dimension reduction+ANN	simulation, statistical tests	daily mean: 8.40E-04
Patel, Shah, Thakkar, Kotecha (2015)	CNX Nifty, S&P Bombay Stock Exchange (BSE); (10 x 1)	Technical indicators	Stock price in 1–10, 15 and 30 days	NA (8:0:2)	Jan 2003 to Dec 2012, daily	2 stages: Support Vector Regression (SVR) + (RF, ANN, SVR)	MAPE, MAE, RMSE, MSE	two stage model: SVR-ANN

Figure 3.2: Relevant researches on Multiday Forecasting

Chapter 4

Design of the ML-based trading system

As mentioned previously, the main purpose of this research is to understand if it is actually more convenient trading in the short term or in the long term. In particular, the thesis wants to find out if a trained machine learning model detects more trading opportunities in an intraday or in a multiday configuration. Since we have not found in literature any studies which effectively compare an automated trading system performance in the long and short term, we have developed an ad-hoc trading system with this scope. The steps that we have followed to properly implement a Trading System are based on the typical stages included in the Knowledge Discovery in Databases (KDD), explained in chapter 2. However, here we have had to adapt the typical stages involved in all machine learning projects to our specific problem, price forecasting movement, which also includes a market simulation system. In the image 4.1 is represented a summary scheme of the components of the implemented trading system.
As it can be observed from the scheme, the input of the system are the stocks data 1-minute granularity from Nasdaq top 100 stocks. The data are passed to a Data Pre-processing step which is used to properly clean and transform the data. Once the data are in the proper formatting they can be fed into the selected Machine Learning Classification models, this step includes hyperparameter tuning, model selection, model testing and finally model validation. In this phase each model is validated, measuring its performance in terms of predictability. Predictability metrics measures how much the predicted value are close to the real ones. These metrics should give an initial insight to understand if the model is able to find more investment opportunities which are actually right in the short or in the long term. However, predictability measures are not enough to make a real conclusion, they do not have any information about profitability. Indeed, in a real trading scenario many others variables affect the whole system. For this reason, following model validation and predictability measuring a trading simulation on the signals

generated by the validated models is carried out. The signals generated by each algorithm, which represents buy and sell signals, are simulated in the market during the test period in order to obtain information about profitability.

In the following sections is explained in detail each phase and steps here briefly described.

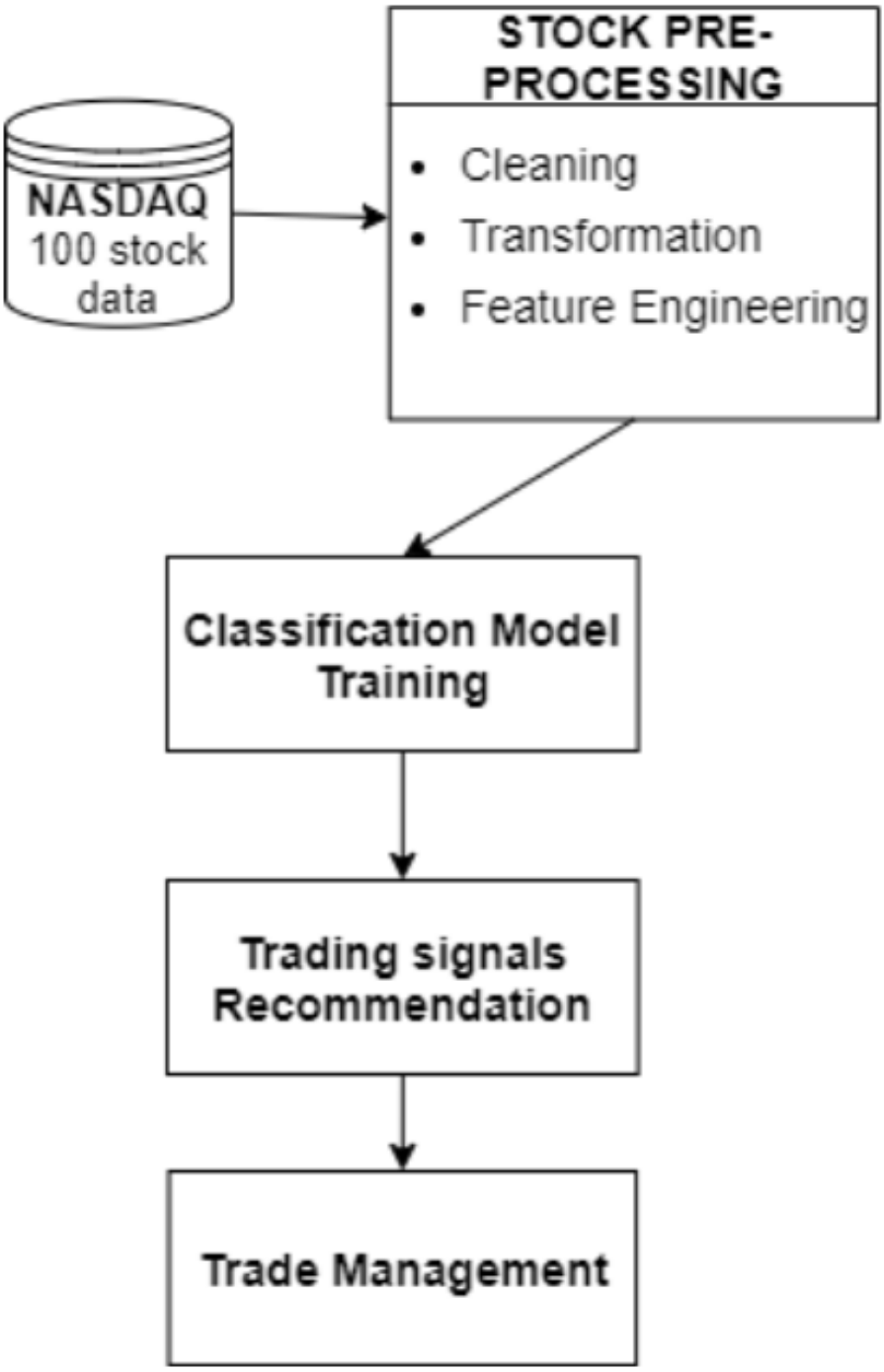


Figure 4.1: Trading system pipeline

4.1 Data description and Engineering

The data used in this research are gathered through Thomson Reuters API [31]. Time series data of 102 equity securities issued by 100 of the largest non-financial companies listed on the NASDAQ, an American stock exchange, the second biggest in the world by market capitalization. From these 103 securities the Nasdaq Index data (.NDX) is included, which represents the average weighted returns of the biggest 100 companies in the NASDAQ . The data collected represent one year of market-based information from 24/10/18 to 24/10/19. Since the objective of this research is to assess the ability of machine learning and statistical models to forecast in the short term, in particular in intraday, the market data gathered are represented with 1-minute granularity. This means that for each stock every one-minute frequency the market information specific to that time interval about volume, open price, high price, low price and close price (the standard open-high-low-close OHLC format) is given, as shown in figure 4.2.

	AAPL_VOLUME	AAPL_OPEN	AAPL_LOW	AAPL_HIGH	AAPL_CLOSE
2018-10-24 13:30:00	4844.0	222.740	222.54	222.74	222.6106
2018-10-24 13:31:00	585649.0	222.520	222.52	223.63	223.3700
2018-10-24 13:32:00	288710.0	223.345	222.70	223.97	223.8015
2018-10-24 13:33:00	278748.0	223.825	223.66	224.20	223.9175
2018-10-24 13:34:00	255362.0	223.920	223.63	224.20	223.8400

Figure 4.2: APPLE stock one minute granularity data

4.1.1 Data Pre-Processing

Data preprocessing is a crucial step before modelling, considered in many applications the most important step of a machine learning projectt. The main steps included in a typical data preprocessing process are: data cleaning, data integration, data transformation, data reduction and data discretization [32]. These steps help to transform the given raw data into an understandable format so that the model can comprehend it more easily and thus become more robust.
Data cleaning step is used to evaluate the quality of the data and identifies the missing values. First the consistent of the data of each stock has been analysed. It has been detected inconsistent data for two securities, namely FOX and FOXA, which represent the same company, i.e. Fox Corporation, but FOXA is a non-voting share class. Since the data provided for these two stocks were completely inconsistent, we just had three months information of the volume replicated in all the other attributes, it has been chosen to drop these two stocks and not consider them in the research. Then, the missing values in the dataset were studied. We

noticed that the data gathered included information even when the market was closed, the so called Extended-hours trading, which is stock trading that occurs either before or after the trading day, i.e., pre-market and after-hour-market. The data provided in the Extended-hours market contained many holes in one-minute frequency and in particular most of the stock exchange do not allow stock trading during the extended-hours, all the data points outside market hours for each stock were filtered out. Evaluating the missing values in the market-hour has shown that the dataset is dense, indeed in average only around 0.3% of the data point is missing in one year of information for each stock. Since the number of missing data are insignificant compared to the total data available, we have not estimated new points to replace the missing ones through interpolation or statistical methods, but the data are kept without these few missing points.
As we have seen almost in all the studies briefly described in 3, market-based information with or without including other information are fed into the model. In most of the researches the market-based features used are based on current and n past information, n arbitrary or experimentally evaluated. Indeed, if we want to predict the price of the stock at the instant t+1, is logical that not just the current instant t influences the price at t+1, but even some information in the previous n instants, i.e. including information from t to t-n+1. For this reason, we have used as feature at each instant t the n lagged return respect the previous instant for each of the five attributes that we originally had. The number of n, represent how far we want to look into the past to predict the next instant. Since we cannot know what is the influence of the past into the future, the number n is adaptable. Indeed, different combinations of the value of n were carried out and then evaluated their respective performance.
The total number of features is thus given by the product of n and the number of initial features, in our case five (volume, open, high, low, close). For instance, if we set n equal to 20 and so look into the n previous instants respect the instant t+1 for each of the initial features, the total number of features will be 100 (5x20). Higher is n and higher will be the number of the features. Thus, more features are presented more the model will take to train and predict. Being this thesis a research which focuses in particular in the short term, long time of training cannot be accepted. Indeed, for instance, if we want to predict the movement of the price in the next minute, the model has to give the result in less than one minute, and we are not even taking into account the time needed to set an order into a broker platform.
The lag return is expressed in % respect the previous instant, as shown in formula 4.1 below.

$$\frac{x_{t-1} - x_t}{x_t}, \frac{x_{t-2} - x_{t-1}}{x_{t-1}}, ..., \frac{x_{t-n} - x_{t-n+1}}{x_{t-n+1}} \tag{4.1}$$

Most of machine learning researches in the data preprocessing step apply some data standardization techniques. Indeed, it has been proved in several studies, such as [33], that applying a normalization on the data often leads to two advantages:

the numerical stability of the model improves and speed-up the training. Actually in the literature is not still clear why normalization improves model performance and speed up the training, however it has shown that is particularly efficient when features are in a different scales. There are several normalization methods, such as min-max normalization, z normalization (standardization) and unit Vector normalization. We have selected the min-max scaler, which simply consists on scaling the values into a selected minimum and maximum value, according to the formula in 4.2. We have set up a range from -1 to 1.

$$\frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.2}$$

4.1.2 Feature Engineering: Technical Indicators

Feature engineering is a technique used to create new features from the available ones with the objective to obtain more information from these new features respect the previous ones. This should help to improve model performance. Often, feature engineering use domain knowledge experts to create new features. In financial forecasting models the most used features are market-based data, technical indicators, macro-economic, micro economic indicators and news-based data, several researches have shown how these data can help to improve model's performances [34]. However, most of the studies use this group of features to forecast in the medium long term, i.e. daily or more. Indeed, it is hard to gather news-based and microeconomic information in the short-term and at each time interval considered. Indeed, there are few cases [35], where news-based information are used as features to predict in the intraday, and to the best of my knowledge there are no studies which use microeconomics or macroeconomics information in the short-term.
Several studies of machine learning forecasting in finance have shown that using technical indicators as features can enhance model performance [36], especially if integrated to other features such as market information. Feature engineering consists on creating new features from a combination of the available features . Computing technical features involves combining the available raw data about opening price, closing price, volume, etc and creating new features which supposedly contain more information than keeping them separated. There are four major types of indicators [37]: trend, momentum, volume and volatility.

1. Trend indicators provide information about the direction of the market is moving, if there is a trend at all. The most used and known one is the Moving Average Convergence Divergence (MACD).

2. Momentum indicators give information about the strength of the trend, often used to understand if a reversal is going to happen .

3. Volume indicators measure how much the volume is fluctuating over time.

4. Volatility indicators are similar to volume indicators, but provide information about the variability of the price and not of the volume.

In our study we have been focused in particular in few indicators within trend and momentum types. Although there are thousands of technical indicators available in literature, most of them are quite repetitive and contain the same information. Furthermore, more features added to the model more the complexity grows and thus the computational expense. Since our resource are scarce (regarding computational power) and our base problem is particularly complex and already contains many features (in general around 20) we opted to add 'only' three technical indicators but each one of them contains a lot of knowledge. Indeed, after a careful studies of most of the technical indicators presented in literature, and considering other similar studies, such as [38], we have selected the following three indicators:Stochastic , Moving Average Convergence Divergence (MACD), Relative Strength Index (RSI).

$$K = \frac{C - L14}{H14 - L14} \tag{4.3}$$

where:
C The most recent closing price
L14 The lowest price traded of the 14 previous trading sessions
H14 The highest price traded during the same 14-day period
K The current value of the stochastic indicator

$$MACD = EMA12 - EMA26 \tag{4.4}$$

where:
EMA12 exponential moving average over last 12 timestamp
EMA24 exponential moving average over last 12 timestamp

$$RSI = \frac{U}{U + D}\% \tag{4.5}$$

where:
U average of upward closing differences of X days D average of the absolute value of the downward closing differences of X days

In the experiments we will empirically try to understand whether including technical indicators to the data to feed into the model can or cannot improve model performance. This will be done for several different configurations of granularity and prediction time. Indeed, for instance, we could have that using technical features in intra-day allows to enhance the performance more than using them in a multi-day configuration.

4.1.3 Imbalanced data and solutions

Imbalanced Data refers to a problem that occurs when the classes of a classification problems are not represented equally [39], namely some of the classes have an higher probability of occurrence. In this situation the distribution of the model's

classes is skewed through the most representative classes. In this case, a machine learning classifier may have a bias with regards to the majority class and thus poorly able to predict the other underrepresented class. For instance, we could have a case in which class-1 is represented 90% of the time and class-2 the other 10%. In this case, the model will have a bias on predicting class-1 most of the time, indeed if it predicts always class-1 it would have an high accuracy of 90%. For imbalanced data set the accuracy metrics does not reflects the real predictability of the model, there are other metrics, which we will discuss in the following section, which can tackle this problem.
In our case problem, i.e stock trading, algorithms and traders are looking in particular in large movements of the price, because these big movements are the ones that potentially can returns higher profits and margins. However, big movements are quite rare, do not happen very frequently and so we could easily have an imbalanced data problem.
There are different way to tackle this problems. If the problem is really severe, and so we have an highly imbalanced data there are two main approaches that can be applied to overcome this issue, namely oversampling and undersampling [40]. Oversampling consists of increase the observations of the minority class with other data points created through some mathematical methods, such as random oversampling and Synthetic Minority Oversampling Technique (SMOTE) [41]. On the other hand, undersampling adjust the class distribution until it finds a balance between the classes. There are several techniques used for removing some sample of the majority class, such as Edited Nearest Neighbours Rule, neighbourhood cleaning rule, one-sided selection method and removing Tomeks links.
Undersampling is used in particular when you have overabundance of data, which does not happen very frequently. Indeed, oversampling techniques are employed much more frequently than undersampling techniques, in general you miss some info and with oversampling you try to create some more information which could increase the model's predictability.

As seen so far, having imbalanced data is a common problem in statistical modeling and in particular in our case problem, because large movements of stock price, the ones we are trying to take advantage of, are rare events. There are several methods such as Oversampling and Undersampling, which attempt to tackle this problem basically by adding or eliminating some instances in order to obtain a balance between classes. However, after a careful analysis of all possible methods within the two major approaches, we concluded that none of the methods is suitable in our case. Undersampling drops many observations and it could easily eliminate some knowledge. Several studies have proved that often more data is fed into machine learning model higher are the performance, this is especially true for machine learning models such as MLP. On the other hand, oversampling, creates new virtual observations, some of them may be inconsistent or repetitive. This could add rumor to the data and thus worse model's performance. Since these

approaches have been excluded, a novel empirical method has been developed to deal with imbalanced data.

In order to tackle imbalanced data problem we have implemented a straightforward empirical method explained in the following. In our case problem the data classes are trading signals which can only assume three values: buy, hold and sell, correspondingly to 1, 0, -1. It should be reminded, as it has been explained in Ch 2, that all the Supervised algorithms are trained by giving them both input and output data. The algorithms, from comparing their predictions to the given output data, will try to build a function able to generalize even to unseen data. In our case the input data represents price stock historical information, we do not have any classes, the classes should be created accordingly to our model. The easiest way to build classes would be to assign +1 or -1 (buy and sell) when the price increases/decreases between one observation and the next. This method has two main disadvantages, the first is that the classes created will most likely be unbalanced, the second is that we are not taking trading fees into account. The empirical method implemented classify 1 when the difference between the close price between two following observations exceeds a fixed threshold, -1 when the price difference will be lower than the negative threshold, and in all the other cases 0. The model predictions and in turns the classes distribution thus strongly depends on this arbitrary chosen threshold. If the threshold assume a high value, the model will rarely predict buy or sold, because big movement of the price are quite rare. In addition, the variability is different in the various granularity and stock, the market is much more volatile in a daily granularity than in the intraday, hence the same threshold value for different granularity but even on different stocks would not be adequate. The implemented system attempts to find an appropriate threshold value to obtain a balanced class distribution. In addition, the threshold has a lower bound constraint which corresponds to the average requested transaction fee in a typical broker platform, which we have set up to 0.003% of the invested capital.

The process that finds the value of the threshold consists in calculating the distribution of the % lag return between two subsequent observations for each granularity and stock and then assign to the threshold the value in the 70th percentile of this distribution. This measure corresponds to the value below which the 70% of the lag returns fall into, or alternatively the value above which 30% of lag-returns fall (i.e observations). Setting this value as threshold ensures to have perfectly 30% of the class buy, 30% or less (because it has been observed that in general the distribution of % lag return is slightly negative skewed) and the remaining 40% or above of the class hold. In this way the classes distribution is almost balanced even if not perfectly because it is not necessary to have a perfect balance among classes. It has been observed that the lag returns distribution is slightly negative skewed, that is, most of the returns between two following observations is positive. This is due to the trend of the market in the period under studying,

from Oct-2018 to Oct-2019. This period has been a particular flourished period, especially for NASDAQ stocks, where most of the stock's price has remarkably grown.

Through this approach the threshold is calculated for each stock and granularity and then used to actually create the classes. As expected, for higher granularity the value is rather low (however always above the configured transaction fee), instead for lower granularity is fairly high (for instance for AAPL in the daily setup the threshold is around 1.2%).
The empirical method implemented leads to several benefits. First it is not needed to arbitrarily and manually find a new and suitable value for every granularity and stocks, this process becomes automated. Furthermore and foremost we can easily control and change the data distribution up to obtain a balanced dataset.

4.1.4 Data granularity

The goal of the thesis is to understand if short-term forecasting could overcome long term-forecasting in terms of predictability, in particular if in intraday there are more investments opportunities than in multi day. For doing so, different experiments varying forecasting time and granularity have been carried out. For intraday forecasting the time selected to forecast are 5, 10, 20, 30, 60, 120 minutes ahead. We have used the same time individuated by [25] described in chapter 2, so we could make a comparison in terms of predictability. There were not carried out any experiments forecasting less than 5 minutes ahead. Even if we had the possibility to do so, since the data provided are in one minute frequency, it has been considered that the stock market is not a particularly volatile market, and thus, in these short granularity is super rare to have a 'big movement,' the ones the model is trying to bust, which could overcome the transaction fee. Therefore modelling a problem in a very short term such as one minute it would have been very hard, very likely it would have led to a highly imbalanced problem ('big movement' very rare), and thus obtaining low performances. Alternatively, it would have even be possible to obtain good performance in terms of predictability, but most likely not in term of %return through a trading simulation, because the movements in a so short period of time cannot overcome the trading fee. It can be argued that the problem still for lower granularity too, such as the ones experiments in this research 5, 10, 20, 30, 60, 120 minutes. This is can be true and these experiments have been carried out even to prove if it is possible to overcome the transaction fees in a short period of time and make profit. However, in higher granularity the market becomes slightly more volatile and we suppose that there may be more trading opportunities in which it is possible to overcome the trading fee. From the results of these experiments, showed in Chapter 5, we could make some conclusion. If the results will show that for these short granularity is not possible to make consistent profit, then the same can also be said for higher granularity, lower than 5 minutes. However there should be made a distinctions between granularity and time to predict ahead. The granularity of the data represents the frequency of occurrences

of the information, for instance if the granularity is 5 minute means that every 5 minute frequency information are provided. Time to predicts ahead in most of the studies correspond to granularity, however they are two different variables. Forecasting time represents how long ahead we want to predict, for instance having a dataset of 5 minute granularity we could predict 5 minutes ahead or more, but not less than 5 minute, such as 1 minute. Actually it is possible to do it, but it would have been very inaccurate. So, as we have said, having a fixed granularity we could predict times ahead greater or equal to the granularity itself. Most of the studies, such as [25], using the same time as granularity and prediction time. In our study we want to understand if granularity can affect the performance keeping fixed the prediction time. For instance, keeping fixed the next time to predict to one hour, we could keep the data to train to 10, 20, 30,60 minutes frequency (it is more accurate if granularity is a multiple of time to predict ahead). To the best of my knowledge, there are no studies in which this variable has be taken into account to assess its effect on performance. Indeed we could for instance observe that having an higher granularity of the data, such as 20 minutes to predict 60 minutes ahead, have an improvement effect in performance, respect using a lower granularity, like the same for the time to predict (60 minute). However this is just an hypothesis, we expect that higher granularity leads to better performance, but to shows the validity of this hypothesis different experiments have to be carried out, keeping fixed the time to predict and varying data granularity. In particular, we have selected a time to predict of 60 minutes ahead and varied the granularity of the fed data, using 30 and 60 minutes. In Chapter 5 the experiments carried out are showed and commented.

4.2 Machine Learning models and hyperparameters

As briefly discussed in Chapter 2, a hyperparameter is a parameter given as input to the model, and it is used to control the learning process of the model itself. It is not to be confused with the parameters or so-called weights of the model, which in contrast are the parameters that are learned and they automatically adjust at each iteration. An example of hyperparameter for a Multiply layer Perception (MLP) is the number of hidden layers of the network.
In this research we have assessed the performance of several machine learning model in a financial task across different time horizons, more in particular we experimented the following algorithms: Decision Tree (DT), Support Vector Machine (SVM), Multiple Layer Perceptron (MLP) and XGBoost (XGB). Their functioning has been been simply explained in Chapter 2 . As we said, each of this model, has one or more intrinsic parameters, the so-called hyperparameter, which influence model performance. A careful choice of these parameters must be made. However, unfortunately, there are no general values which work well in any case, although

there are some hyperparameters configurations which have proved to work well for some kind of problems and data, it is mainly problem-specific. It is thus necessary search for the best hyperparameters for a model from a given set of values and select the one which enhance the performance. This approach is called hyperparameter optimization. There are several hyperparameter optimizations, the standard one and most used is the so-called Grid search. Grid search method simply consists of assess model performance on all available combination of hyperparameters given as input and it returns the hyperparameter combination which has led to the best model performance according to some specified metric (accuracy, precision, recall ..) in the validation data or in the hold-out set. In this research we have implemented and used this method given a list of possible values for the hyperparameters for each model available in appendix. One problem of grid search is the curse of dimensionality, because the research effectuated is exhaustive, which could be both a pro because exploring an higher space a better configuration could be found, but even a contro for the high complexity problem and computational power requested.

4.3 Evaluation metric selected

As we have seen, when the training and testing are terminated the values predicted by the model and the observed values are compared to assess the performance of the model to generalize in new data.
In Chapter 2 several evaluation metrics were described, including their flaws and advantages. After a careful study of each evaluation metric in literature, we opted for f1-score. As described in Ch. 2, f1-score is the harmonic mean of precision and recall. Precision measures how many times our model is correct when predicts true, instead recall take into account even when the model gets wrong. Since the goal of a trading system is to be sure that when the model predicts either to buy or sell is actually a buy or sell (Precision metric), but at the same time, it is important even to maximize the number of transactions and take advantage of as many of opportunities trades as possible (Recall). Measuring only Precision could lead to have a model which predicts only few times buy, but we would trade only those small number of times. Taking into account exclusively recall we maximize the number of trades, however not all the trades are good investments opportunities , especially slight market movements. Therefore, a trade-off between these metrics is the best fit in this case. Fortunately, the F1-score is the best match, since it measures the harmonic mean of precision and recall metrics. However, we want to take track of each metric for each class, because it could help us to identify some pattern in some particular stocks and/or classification algorithms.
For doing so we used the classification_report method in the *sklearn.metrics library*, which returns a matrix within the main classification metrics for each class.

classifier	index	precision	recall	f1-score
MLPClassifier_hidden_layer_sizes=(20,)_max_iter=200_solver=lbfgs	-1	0.353664	0.353985	0.353825
	0	0.440216	0.491984	0.466100
	1	0.334057	0.372963	0.353510
	accuracy	0.421852	0.421852	0.421852
	macro avg	0.375979	0.406310	0.391145
	weighted avg	0.409481	0.421852	0.412889

Figure 4.3: Classification report per class (accuracy, precision, recall and f1-score)

The obtained results includes macro average and weighted average. Macro average is a measure of unweighted mean for label, while weighted average takes into account the weight through the support value to account for label imbalance. Support represents the number of occurrences of each class in the observed data. Since our data are not perfectly balanced we have selected the weighted average of the f1-score. This should address both imbalanced problem and trade-off between precision and recall. However, we would have a table similar to the one in figure x, for each classifications algorithms and stocks. Eventually, we end up with around 2000 classification reports (100 stocks x 20 algorithms), for each trading configurations (granularity, time to predict ahead and validation methods). In order to identify the best classification algorithm within the same setup configuration (granularity, ...) we have grouped by classification algorithms, and averaging on weighted avg for f1-score, and eventually descending sorted by this value. We end up with something like the table in the Figure 4.4:

classifier	precision	recall	f1-score
MLPClassifier_hidden_layer_sizes=(20,)_max_iter=200_solver=lbfgs	0.420422	0.429162	0.423960
MLPClassifier_hidden_layer_sizes=(100, 100)_max_iter=200_solver=lbfgs	0.409527	0.454424	0.418071
DecisionTreeClassifier_criterion=entropy_max_depth=20_min_samples_split=10	0.414511	0.419158	0.416835
MLPClassifier_hidden_layer_sizes=(100,)_max_iter=200_solver=lbfgs	0.403673	0.465649	0.401744
SVC_C=100_gamma=10_kernel=rbf	0.394719	0.430636	0.399936

Figure 4.4: Statistical evaluation metrics per Classification algorithm (best 5 in terms of f1-score)

For instance, in the table reported, the best algorithm (with highest f1-score) is a Multiple Layer Perceptron with the described configurations and hyperparameters (hidden_layer_sizes=(20,), max_iter=200, solver=lbfgs). Once individuated the best algorithm through the method now described, the results of each of 100 stocks

under the best classification algorithm selected are retrieved, as shown in figure 4.5

stock	precision	recall	f1-score	support
GOOG	0.503734	0.541667	0.475951	72.0
CTXS	0.475913	0.472222	0.470528	72.0
TXN	0.461000	0.500000	0.460762	72.0
JD	0.438955	0.458333	0.447973	72.0
SYMC	0.449036	0.444444	0.445493	72.0
VRTX	0.448581	0.444444	0.445004	72.0
AVGO	0.518347	0.444444	0.444587	72.0
NVDA	0.441738	0.458333	0.444086	72.0
DLTR	0.439945	0.472222	0.441892	72.0
COST	0.433749	0.444444	0.438469	72.0

Figure 4.5: Statistical evaluation metrics per stocks (best 10 in terms of f1-score)

In the next chapters, for each setup, the results obtained through this method are reported, compared and commented.

4.4 Validation method selected

As we have seen in Chapter 2, in literature there are several validation method used to assess the performance of a model. Each of them presents some advantages and disadvantages. In the experiments, two different validation methods have been tested: Expanding Window and the standard hold-out. Actually, at the beginning we exclusively selected the Expanding Windows method, but then following some issues related to computational power, which will be discussed in depth in the next

chapter, we tested holdout validation too.
Holdout allows to substantially decrease the training and testing time, because you train the model only in a part of the data and then test on the remaining smaller part.
Expanding Window methodology has been selected because is the most suitable validation method for trading problem, because it works as how real trader optimize a trading strategy in trading, and can be easily applied in online training. EW indeed is applied first in a part of the available data, the so called 'window', to forecast the next period and then rolling out the window until the end of the data. In this way almost all the data are used both for training and testing. Thus, the model is supposed to learn more, because it is fed with more data respect other validation methods and have a more accurate measure of performance. In order to use the Expanding Window methodology some parameters have to be setted, in particular: the minimum number of observations and sliding or expanding windows.
Minimum Number of Observations: At the beginning we cannot test the model for the first few observations, because we do not have enough information. Actually, a model trained exclusively in the first instance, and then tested in the second, can be created, but it would very likely not accurate, since the model has been trained with few information. In any case the first observation can never be tested, because following the rules of an appropriate Validation Method, the training set must be different from the test set.
The model is thus tested only after some n arbitrarily chosen observations. when it is considered that has been trained with enough information. The number of instances which will be excluded for testing and used exclusively for training the model represents the minimum number of observations. After this n number of observations the model will be both trained and tested. In this research the minimum number of observations for training has been set to 20. It has been considered that 20 instances are enough for the model to forecast. It can be argued that 20 is a low number for training the model, this it can be true, indeed we think that the first predictions of the model will not be very accurate, however, following the Expanding Window method, the following prediction should be more accurate since the model have been fed with more data.
The other key variable to set up for this method is to choose whether to use a sliding or expanding window approach. Expanding window approach consists on training in each timestamp on all the previous data available, on the contrary sliding windows in timestamp t select just n arbitrarily chosen previous observations and train on all t-n observation for testing in t. Basically, this method gives more weight to recent observations, and older observations are not taken into account. Even though the market strongly depends more on recent information than past information and so a sliding approach could be suitable, in several studies have been shown that the market movements tends to repeat even after a considerable period of time. For this reason we opted for an expanding window approach. However the expanding method uses much more data to train comparing with the sliding

method, especially the last predictions are performed after the model is trained on all the data. This will substantially lengthen the training and testing time.
After a thorough configuration of this variable is chosen, the minimum number of samples is used to train the model. Then the trained model makes a prediction for the next time step. The prediction is stored and eventually evaluated against the actual value. The window is then expanded by one timestamp and a new model is created and trained on minimum number of samples plus this new observation. The model than predicts the next observation. This method is looped until the last observation is reached. In the end all the predictions made by the models will be evaluated against the actual values to assess the performance. This method has the benefit of providing a much more robust estimation of the performance in a real trading scenario. This improved estimate, however, comes at a high computational time, at every timestamp a new model is created, trained and tested. So, if we have 13000 observations, like the dataset of 5 minute granularity 13000 models will be created and trained. This it can not be a problem if the statistical model is very simply, like a naive bayes or linear regression, however using more complicated models which requires more computational power, suche as Multiple Layer Network (MLP) or XGB it can be infeasible with the the tools at our disposal and extremely computational expensive. Indeed, as we will see in 5, we have not been able for computational constraints to experiment the Expanding Window approach using expanding window for all the granularity and statistical models. For this reason we adopted standard Hold-out method on all the configurations of granularity and predictions selected and discussed above.

4.5 Trading system and Trading Strategy

One of the main goals of this research is to understand if it is more profitable in terms of return of capital over the period considered trading using short or long term predictions generated through a machine learning classification algorithm. In the previous sections we have seen how to assess the performance of a classifier and which metric is more suitable in this case. However, all the evaluation metrics measure exclusively the ability of the model to predict, by comparing the predicted value to the actual value. These statistical evaluation metrics give information about predictability of the model, but not information about profitability. Indeed, it is not ensured that having a good predictability score imply to have higher returns on capital. It is known in the literature that predictability and profitability are related somehow, but this relationship has not been found yet.
In a real trading scenario many other variables influence the whole performance, for instance the strategy, portfolio construction, transaction fee, spread, time required to open/close an order, capital, leverage, money management and many others. Therefore, the only way to understand if the signals generated by machine learning models can generate profit in a real trading scenario, a trading simulation has to be carried out. Only through a trading simulation we could have a measure of

the profitability of different models and configurations and eventually claim which configuration, setup has obtained the best performance. As said, in a real trading scenario several variables could affect the profitability, nevertheless in this research we have focused only on variables considered most relevant and necessary to launch a simulation: portfolio construction, trading strategy, transaction fee, stop loss and operation length.
In the following is described the trading system implemented. At each timestamp the system takes in input all the signals generated by the selected Machine Learning Classifier for all 100 Nasdaq stocks in the corresponding timestamp. However, the system is not designed to trade all the stocks at each time, because otherwise it would open too many operations and therefore the fees would increase. In order to do so, the system select only the signals of top N (arbitrarily chosen, in our case N=20), stocks sorted in descending order by volume. As explained in the previous section, the signals can only assume three values (+1,0,-1), correspondingly to buy, hold, sell. The buy and sell signals are generated when the model predicts that the next price will go up or down above a certain threshold. The trading system thus open and close transactions on the market based on the corresponding input signal. Open transactions are automatically closed only under one of the following 3 conditions:

- Inverse signal: if in input is received an opposite signal, for instance an operation was in long (1) and in the following instant we receive a 'short' signal (-1) the operation is closed.
- Stop Loss: if the maximum loss per operation that has been set is reached the operation is closed
- Max Length: this variable indicates how long a position can remain open. If this time is reached the trade is automatically closed.

One variable that can have a major impact on profitability performance is the fee applied at each transaction. In most of the available researches in the field the transaction fees are not taken into account into the trading system. This has a big influence and compromises the results of the overall % return. This is particular true for intraday trading, because many operations are executed in a short period of time. In order to have a more authentic representation of reality our strategy takes into account transaction fees. Actually, we launched all the simulations with different transaction fee values, to try to measure their impact on final performance. However, the transaction fees depend on different variables, such as capital and brokerages. Therefore it is not easy to assess a fair transaction fee, however in line with other studies in the field in line with other studies, we have experimented with fee values in the range of 0.003% and 0.005%. In the next chapter we will see all the experiments carried out and the results obtained through the trading system implemented here explained.

Chapter 5

Experiments and Results

A typical forecasting stock price movement problem has several variables which can influence the system performance and thus the predictability. However, we could not experiments every single variable which could affect the performance, it would have not been feasible for time and resource constraints. Therefore, we have selected some variables which we believe to be the main drivers of a typical price forecasting system: data granularity, prediction time, windows size, machine learning classification models, validation method, trading strategies and stop loss values . Actually, for some configurations, we tested other variables, such as the impact of adding technical features to the data to feed into the model and whether to take into the training set the first daily observations. The variables experimented have been explained in detail in the previous chapter, however without experimentation it is not possible to know their real impact on performance, whether and how they can actually positively impact the predictability and profitability. The experiments carried out consists on varying one or more of the variables selected and assess the results for any different configurations. This means, for instance, that for each different configuration of granularity or windows size, all machine learning models used need to be trained and tested. This approach can be easily automated, but it still a long task because some models, in particular models with high granularity and/or complex models such as MLP and XGB, take a long time to train. Indeed, higher the granularity and higher will be the training time. The experiments carried out are similar in intraday and in multiday configuration, however this research focuses more on intraday and thus more experiments have been carried out in that scenario. Initially the predictability results are collected, in terms of f1-score, as explained in the previous chapter. Comparing predictability results is the first step to understand which model and/or configuration is the most accurate in understanding the future market movements. Following, other experiments related to the trading system, such as identity the most profitable model, trading strategy and stop loss values are reported.

From the results of all these experiments the goal is to understand the following points:

1. What is the impact of granularity, windows size, technical feature on predictability and profitability?
2. What is the most accurate and profitable Machine Learning Classification algorithm?
3. What is the most profitable trading strategy?
4. Is it more profitable the trading system based on EW or the based on HO validation method?
5. Is it more profitable using a system based on short-term (intraday) or long-term predictions (multi-day) ?

In the following all the experiments carried out will try to answer these research questions.

5.1 Predictability Results

Several experiments in the intraday configurations have been carried out, but unfortunately for time and resources constraints we have not been able to experiments all possible set-up. As we have seen in Chapter 2, the best validation method for a time series prediction problem is the Expanding Window Approach because allows to have a larger test set and has a similar approach to that used by manual traders. In principle, the idea was to experiment different data granularity and models exclusively under this validation method. However, unfortunately we encountered several issues related to computational expense and time. As already discussed, the main flaw of Expanding Window methodology is the training time, because at each observations a new model is created and trained. All the experimentation have been executed on the university server provided by the DIUM department. The computer has this technical features: Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, 4 cores, 4GB RAM. Using the Expanding Window Validation method in intraday configuration for granularity higher than 30 minutes the training approach would be lasted more than one month only for one set-up (fixed hyperparameters) of a MultiLayer Perceptron. We have approximately estimated that the training in the university's computer of a MLP model with the highest number of layers (5) feeding 5 minute granularity data would be lasted around 5 months. Therefore, for technical constraints we had to change our original plan of experimentation and include another validation approach which is much lighter and faster: Hold-Out. Nevertheless, the Expanding Window methodology has not been completely dropped, indeed, where was practically feasible in terms of time and resource, for example for 60 and 30 minutes granularity, we have trained simple models, such as Decision Tree and SVC.
In the multi-day configuration the same approach used in the intraday scenario has been used. In this way we can compare the performance obtained in the two

different results, which it is indeed the goal of this research. As done in intraday, both Expanding Window and Hold out as validation methods are applied for different set-up of data in input. However, the main difference respect the short term forecasting problem, it is that in multi-day setup the number of observations in the dataset are drastically reduced, the granularity is much lower, and this allows to train and test effectively several different models, even model with high complexity using a Expanding Window approach, such as MLP with several layers. Indeed, in this case, we were able to train all the models planned and explained in chapter 2 (with different configurations of hyperparameters, appendix A). For instance, the number of observations using a 5 minute granularity are 19373, while in a daily set-up the dataset is composed of 252 instances, we see that passing from a 5 minute to a daily representation the total number of data is reduce by a factor around one hundred. This should drastically reduce the training time.
As mentioned several time throughout this research, the thesis is mainly focused on studying short term forecasting, because of the lack of research in this scenario than in long term. However, another goal of the thesis is to understand if intraday forecasting could spot more opportunities than multiday. For this reason multiday is studied as well, but not as in detail as intraday forecasting. In the long term configuration some experiments have been carried out, but in this case we did not varied the time to predict ahead, it has been kept to one day, which is the typical prediction time used in most of the literature in this field.

In the following, the results obtained using the different validation approaches are shown and described in detail.

5.1.1 Holdout

The main advantage of using HO validation method is the low computational power and short time of training. This allowed to successfully train and test all the machine learning models in all the possible granularity and windows size setups originally selected and thus perform a complete study. Through the analysis of the results obtained in this configuration, and eventually taking into account multiday performance using a HO validation method, we should be able to answer the research question of this thesis: *'is it better in terms of predictability and profitability to use a machine learning in the short term (intraday) or in the long term (multiday)?'*.
As discussed in Ch. 4.3, f1-score is the metric chosen to evaluate the performance of a model in terms of predictability and make the comparison of different configurations. This metric, according to our observations, is the most robust and suitable in this specific problem: forecasting the movement of stock price. In table 5.1 we can observe all the experiments carried out in this scenario. As discussed, this scenario is the one that allowed us to carried out more experiments, as you can see from the table below, the 11 different data formatting. It should be considered that for any data formatting all the models are trained. In total, varying the hyperparameters, the number of models is 24. Every time the data to feed into the model changes the

training must be re-launched (11 different data formatting, as shown in the table below). Eventually only in this scenario have been trained 264 models (24x11).

Table 5.1: Hold-out intraday experiments

configuration name.	granularity.	prediction.	window size.	validation.	other.	#features.	#instances.	test set
intra_HO-5-5-30	5	5	30	HO		24	19737	5921
intra_HO-10-10-60	10	10	60	HO		24	9988	2996
intra_HO-20-20-90	20	20	80	HO		16	4994	1498
intra_HO-30-60-120	30	30	120	HO			3495	1049
intra_HO-TI-30-30-120	30	30	120	HO	TI	19	3495	1049
intra_HO-TI_WS_30-30-120	30	30	120	HO	TI, WS	19	3221	966
intra_HO-TI-30-60-120	30	60	120	HO	TI	19	3495	1049
intra_HO-60-60-240	60	60	240	HO		16	1745	524
intra_HO-TI_WS_60-60-120	60	60	120	HO	TI, WS	11	1657	497
intra_HO-TI_WS_60-60-180	60	60	180	HO	TI, WS	15	1657	497
intra_HO-WS-60-120-180	60	120	180	HO	WS	12	1657	497

granularity: data frequency (min); *prediction*: next prediction (min); *window size*: look back period (min); *validation*: HO or EW; *other*: tecnhical indicators (TI) or/and slide window (WS); *features*: number of attributes, instances: total observations; *test set*: # of observations in the test set

Studying the f1-score tells us which configurations (data formatting, models, hyperparameters) has performed best in terms of predictability. In table 5.2 is reported for each data formatting the model and the respective f1-score that has performed best respect all the others (1 out of 24) in terms of f1-score in that data configuration. In particular, for prediction of 30 and 60 minutes ahead some experiments have been carried out, trying different setup varying the following variables: technical features, slide windows, windows size and granularity. In the field 'other features' is indicated if other adjustments to the data to feed into the machine learning have been carried out. In particular, it is indicated if Technical Indicators (TI) are present and/or if we have considered the first n observations of the days (WS). If WS is present, this means that the first n daily observations, with n equal to the window size, are filtered out. The idea behind this experiment comes from the hypothesis that the prediction at the first observations of the days are not accurate, since the model uses the information of the previous day. According to this hypothesis if we do not predict in the first daily observation overall performance would increase. Therefore, we experimented this new set-up in which the first n observations are used exclusively as training for the following instances in the same day. In this way we will not trade in the first minutes or hours of the days, we will trade only when we are sure that we have enough data to make a prediction. From the trading system simulation which should effectively prove if this hypothesis holds.
For 30 minutes prediction ahead, we obtained that the best setup is the one which includes technical indicators, does not predict in the first observations of the day and uses a MLP classifier. For predictions 60 minutes ahead, however it resulted the opposite, the best performing configuration is the one which does not include technical indicators and includes all the data, even the first observations of the day. Although, the classifier that performs best is still a MLP and this is true even for predictions longer in the time, such as 120 minutes ahead.
Furthermore, studying how varying data granularity and keeping fixed the prediction time affect the performances, it turned out that if we maintain a granularity of data equal to the time we want to predict in general we achieve better performance. This is true for both 30 and 60 minutes prediction time, the best data formatting is the one with granularity equal to 30 and 60 respectively.
The configuration with the highest f1-score and thus predictability is intra_HO-WS-60-120-180, which uses data of 60 minutes granularity to predict 120 minutes ahead. However the variability of f1-score on different configurations is quite high and there is no clear sign that either increasing or decreasing the time to predict ahead the performance improve or worsen.

Table 5.2 Hold-out intraday predictability performance

configuration name	best classifier	f1-score
intra_HO-5-5-30	DT	0.398
intra_HO-10-10-60	DT	0.394
intra_HO-20-20-90	MLP	0.384
intra_HO-30-60-120	SVC	0.385
intra_HO-TI_WS_30-30-120	**SVC**	**0.397**
intra_HO-60-60-240	**MLP**	**0.388**
intra_HO-TI-30-30-120	DT	0.364
intra_HO-TI-30-60-120	SVC	0.354
intra_HO-TI_WS_60-60-120	MLP	0.375
intra_HO-TI_WS_60-60-180	MLP	0.377
intra_HO-WS-60-120-180	**MLP**	**0.413**

Multi-day Hold-out

In the table 5.3 you can see all the experiments carried out using the Hold Out Validation method on a daily configuration. As highlighted previously, in this configuration the number of training set instances are not a large number of observations, this could be a flaw has shown in several machine learning studies more observations are fed into the model higher are the performance. Indeed, studying the performances obtained in term of predictability, shown in table 5.4, where for each configuration is reported the best model and its performance, the overall performance are not particularly high if compared with the ones obtained in the intraday configuration. Here the f1-score ranges around 0.35 - 0.36, which is slightly better of a random model. Interesting to notice that for all configurations the best performing model is a model with high complexity, such as XGBoost and MLP. As highlighted in bold in the table below, the set up with the highest f1-score is the one that uses 7 days as windows size, does not include technical indicators and it is trained using a MLP classification model.

Table 5.3 Hold-out Multi-day experiments

configuration name	gran	next prediction	window size	other	features	train set	test set
daily_HO-TI-1-1-5	1	1	5	TI	21	252	76
daily_HO-1-1-5	1	1	5		20	252	76
daily_HO-1-1-7	1	1	7		28	252	76

granularity: data frequency (min); *prediction*: next prediction (min); *window size*: look back period (min); *other*: tecnhical indicators (TI) or/and slide window (WS); *features*: number of attributes, *train set*: total observations on training set; *test set*: # of observations in the test set

Table 5.4 Hold-out Multi-day predictability performance

configuration name	best classifier	f1-score
daily_HO-1-1-5	XGB	0.363
daily_HO-1-1-7	MLP	**0.364**
daily_HO-TI-1-1-5	XGB	0.350

5.1.2 Expanding Window

As previously mentioned, due to computational and time constraints, we could not run all experiments originally planned on high granularity, such as for 5, 10 ,20 minutes granularity. In this scenario, it would have taken up to one year to effectively training and test all the models original planned. However we manged to train all the models selected for medium granularity, from 30 minutes to 1-day . In Table 5.5 we can see all the experiments carried out using the Expanding Window Validation Method, indicating the model which performed best in terms of f1-score in that set-up.

Table 5.5: Expanding Window intraday experiments

configuration name.	granularity.	prediction.	window size	other.	#feature	#instances.	test set
intra_WF-TI_WS_30-30-120	30	30	120	TI, WS	19	3375	3355
intra_WF-30-60-120	30	60	120		16	3495	3475
intra_WF-TI_WS_60-60-120	60	60	120	TI, WS	11	1745	1725
intra_WF-TI_WS_60-60-180	60	60	180	TI, WS	15	1745	1725
intra_WF-WS-60-120-180	60	120	180	TI, WS	15	1745	1725
intra_WF-60-120-240	60	120	240		16	1847	1827

granularity: data frequency (min); *prediction*: next prediction (min); *window size*: look back period (min); *validation*: HO or EW; *other*: technical indicators (TI) or/and slide window (WS); *features*: number of attributes, instances: total observations; *test set*: # of observations in the test set

In the table 5.6, we can see for each configuration described in the previous table, which classifier has registered the best f1-score and its respective value. From the results we can observe that for almost all the models f1-score ranges around 0.38 to 0.4. This shows that the model's predictions are better than random. Being a multi-classes problem, and having three classes, a random model would have an accuracy and an f1-score around 33%. Our accuracy, in particular for the most performing models is 21% higher. It still not an astonishing performance and at this point we can not know if our model is able to generate profit in a real market simulation, however the accuracy results obtained are comparable to the state-of-the-art performance in intraday trading [25]. If we exclusively look into the configurations in which the granularity is 30 minute, we observe that the most performing configuration is intra_WF-TI_WS_30-30-120, which is the configurations where the technical features and the slide window (WS) method are applied. In this set-up the best performing model is the MLP (hyperparameters: hidden_layer_sizes=(20,)_max_iter=200_solver=lbfgs) , obtaining a f1-score of 0.404. For 60 minutes prediction the best performing configuration is intra_WF-TI_WS_60-60-120 trained on MLP model (hidden_layer_sizes=(20,)_max_iter=200_solver=lbfgs), that is a set-up with granularity equal to the prediction time, technical indicators are included, the first observations of the days are only used for training, and the windows size is the double of granularity and prediction time. Instead, for 120 minute prediction ahead the best configuration is intra_WF-60-120-240, applying a SVC model (C=0.01_gamma=10_kernel=rbf) obtains 0.401 as f1-score. For 60 minutes granularity seems that even applying TI and slide window method do not improve performance.
Overall, performance are quite variable and it is difficult to identify which variable has the greatest impact. However, we can observe that for 30 and 60 minutes predictions the best models are the ones with data granularity equal to the time to predict, respectively 30 and 60. This is the same result that occurred in intraday Hold-out experiments, shown in the previous section. Furthermore, it is interesting to notice that the best performing models apply the so-called slide window (WS), that is the method that does not predict in the first n observations of the day, these are used exclusively for training.

Table 5.6 Expanding Window intraday predictability performance

configuration-name	best classifier	f1-score
intra_WF-30-60-120	SVC	0.355
intra_WF-60-120-240	**SVC**	**0.402**
intra_WF-TI_WS_30-30-120	**MLP**	**0.404**
intra_WF-TI_WS_60-60-120	**MLP**	**0.392**
intra_WF-TI_WS_60-60-180	MLP	0.384
intra_WF-WS-60-120-180	MLP	0.399

However, as discussed previously, the experiments in intraday configuration are not complete, only few granularity and predictions times have been experimented (30, 60 and 120 minutes) mainly in few simple classification models (Decions Trees and some simple configurations of Support Vector Machine and MultiLayer Perceptron). Hence, from the results seen so far we cannot imply any conclusions about which model performs better in intraday and different setup of prediction time, because we do not enough data to make a significant comparison. However, it is interesting to notice that for the set-up where the prediction is higher than 20 minutes the best performing classifier is in most of the cases a MultiLayer Perceptron.

Multi-day Expanding Window

The same experiments carried out for Hold out approach are experimented using the Expanding Window method as well. However, as highlighted several time throughout the thesis, in this case more instances are used both for training and testing. This is supposed to improve the ability of the model to predict, since it uses more data and the results obtained in the test set should be more significant and robust.
In table 5.7 for each experiments is reported the best performing model and its f1-score. For all these configurations the f1-sore does not vary significantly, in all three setups the f1-score is around 0.37, which is slightly better than performances obtained using Hold out as validation, although this difference is not significant. In any case, it is interesting to point out that while for Hold Out approach the best performing models have an high complexity, in this case the models with highest performances are fairly straightforward, such as Decision Trees. Furthermore, we observe that the configuration having the highest ability to predict uses the

technical indicator in input and a Support Vector Machine model, which in general has an higher level of complexity respect Decision tree classifiers. As shown in all the different validation methods and for intraday, it is still true for multi day that using a wider window size does not improve and affect significantly the f1-score.

Table 5.7 Experiments Multi-day Expanding Window

configuration name	gran	prediction	window size	other features		train set	test set
daily_WF-TI-1-1-5	1	1	5	TI	21	328	298
daily_WF-1-1-5	1	1	5		20	328	298
daily_WF-1-1-7	1	1	7		28	328	298

granularity: data frequency (min); *prediction*: next prediction (min); *window size*: look back period (min); *features*: number of attributes; *train set*: # observations on training set; *test set*: # observations in the test set

Table 5.8 Predictability results Expanding Window Multi-day

configuration name	best classifier	f1-score
daily_WF-1-1-5	DT	0.371
daily_WF-1-1-7	DT	0.373
daily_WF-TI-1-1-5	**SVC**	**0.375**

5.2 Machine Learning model experiments

In Ch 2, several Machine Learning Classification algorithms have been explained in detail. In Ch 4 sec 4.3, are listed the Classification models applied in this research and the importance of hyperparameters tuning. After following the typical process used to select the most performing configuration of hyperparameters for each algorithm, briefly explained in the previous sections, a trading system has been tested on the signals generated by each of them. The macro classification models used to forecast the movement of the price of NASDAQ 100 stocks are the following four: Decision Tree, Support Vector Machine, Multiple Layer Perceptron and XGBoost. For each of them has been experimented different configuration of hyperparameters, all the setup tested for each algorithm can be found in Appendix. The predictability results for each configuration of each model have been gathered and compared against each other in order to select within the same algorithm the

most performing configuration in the Validation set in terms of f1-score, the metric selected to assess the ability of the model to predict the movement of the price.
Following the process of measuring profitability results, trading simulation on the same period of time, for different prediction time (from minute to daily) have been launched, in order to evaluate which Machine Learning model reach the highest performance in terms of profitability (percentage of return over a period of time) in the different predictions setup. The aim of this approach is to select only one over many (in this case we have started with 4 macro-models, but considering the variation of hyperparameters the model trained and tested are more than 20) machine learning models and configurations. The advantage of having a single model is to make a valid comparison of the performance of this model on the variable under studying. In this specific case, as described previously, the variable of most interested in this research is the prediction time, i.e using the same machine learning model on different prediction time (5, 10, 20, 30, 60, 120 minutes and 1 day) how profitability performance change?
In order to make a valid comparison we have thus first select the same model for each prediction time and only once it has been found it becomes possible comparing performance on different prediction time. For instance, a comparison between a MLP model for 5 minutes prediction against a SVC model for a daily prediction can not return results that can be generalized, if we had higher return on the MLP on 5 minute predictions we could not conclude that it is more profitable make trading at short period, because we do not know how are the performance of the same MLP model in a daily configuration.
As said, in order to select one model over several, we have compared each of them in the different configurations, i.e prediction time and validation method. In the images 5.1 and 5.2 below, it is reported the evolution of the equity line for each classification model, in the first image in a configuration with 30 minutes prediction, while in the 2nd image 60 minute predictions. These are two examples over several in which Multiple Layer Perceptron is the classification model with the highest ability to predict and reach positive return. This is true practically for all the prediction time tested, expect few rare cases where XGBoost algorithm seems to slightly overcome MLP.

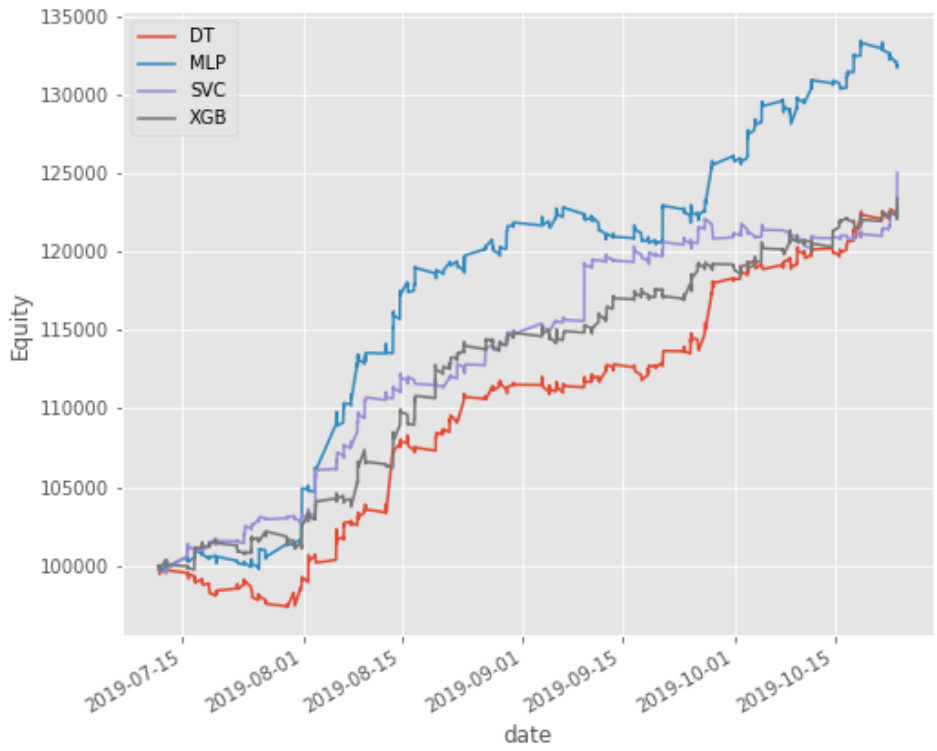


Figure 5.1: Evolution equity lines of 4 Machine Learning Classification model from 11/09/2019 to 23/10/2019, model configuration: 30-30-120 HO

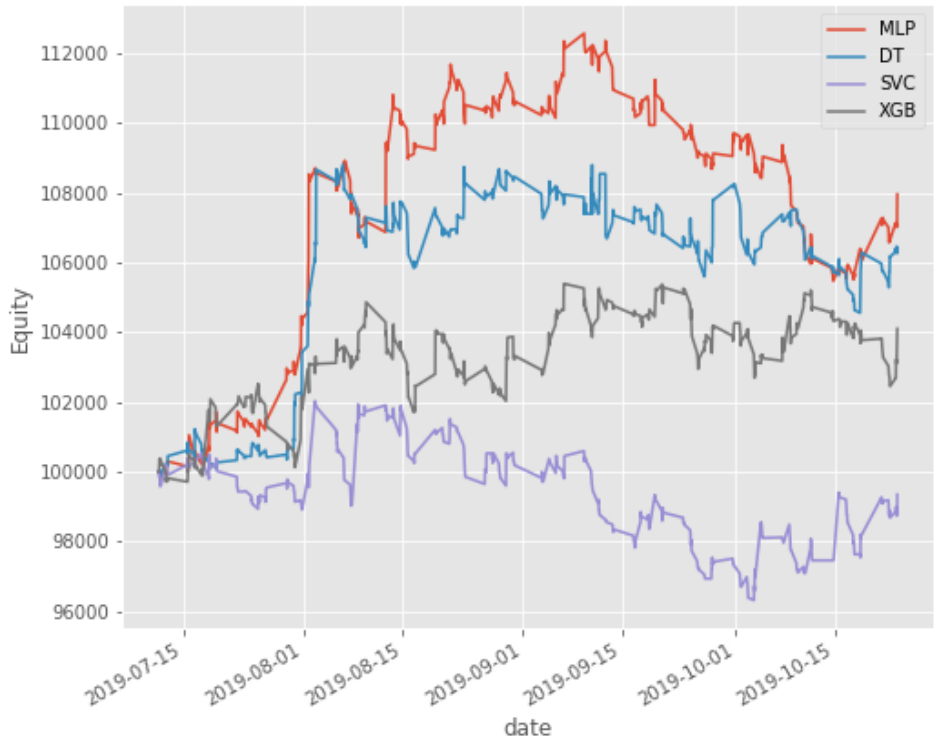


Figure 5.2: Evolution equity lines of 4 Machine Learning Classification model from 11/09/2019 to 23/10/2019, model configuration: 60-60-240 HO

5.3 Trading strategies experiments

The trading strategies applied has an enormous influence on profitability performance. Even though the model trained has registered good performance in terms of predictability, it could be possible that if this model is simply experimented thorough a trading simulation with a simple or no strategy the same model which performed very good in terms of statistic performance performance very poorly in terms of profitability. A suitable trading strategy can then turn a negative return of the same model in a positive return. This has also been observed by our tests. Indeed, at first, we have not implemented a proper trading strategy and thus we have simply fed the signals generated by the model to the trading system. the simple trading strategy that we have used in this case was a simple long and short, without any consideration and studies of several other variables that often are taken under studying in a trading system, for instance the stop loss, take profit, capital invested, profit risk ratio, and portfolio building. The initial trading system was solely trade on the Nasdaq index and did not take into account all the other signals generated for the other 100 stocks in the NASDAQ 100.
In image 5.3, the equity line obtained in the Nasdaq index is shown. As can be seen, the performance are positive, around 4%, but quite disappointing after training such complex models for so long. After gathering several similar negative and disappointing results, we have realized that we had to implement a proper trading system and trading strategy which takes into account several variable and do not trade exclusively in only one stock (previously only in the NASDAQ index) but also using the signals generated for all the others stocks.

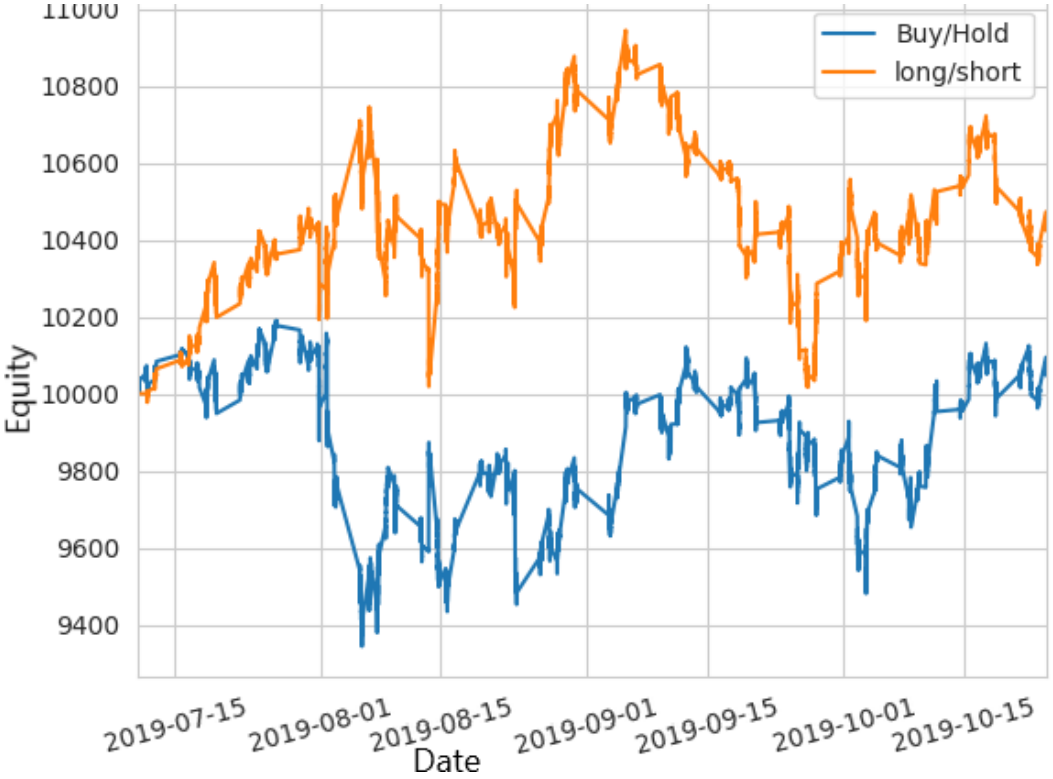


Figure 5.3: Evolution %return 120 min trading in the NASDAQ index from 2019-07-10 to 2019-10-24, model configuration: 60-120-240 HO

The trading system used is described in more detail in Ch 4.5, here we focus in particularly in the trading strategies implemented. After reviewing and studying in the literature several trading strategies [42], the following four trading strategies have been selected and experimented:

1. long-short standard
2. fixed fractional
3. max opened
4. close end of the day

The long-short standard trading strategy is the one we have used in the image 5.3, but in that case using a much simpler trading system. It simply consists on investing long if the signal in input is but and instead, goes short when the signal is the opposite.
The fixed-fractional trading strategy consists on limit each trade to a fixed portion of the equity, usually the portion ranges between 1 and 10 percent. So, at each

trade the trading system compute the amount to invest (equity*fixed fractional percentage) respect the equity at that specific instant, and then checks if in the available capital (the capital not yet invested) the amount obtained is disposable, if yes the operation is opened otherwise we go to the next instant.
The Max opened is similar to the fixed fractional strategy. It simply consists on indicating the maximum number of operations which can be opened and the capital is evenly distributed across the available positions. When the limit of opened operations we cannot invest until one of the opened positions is closed.
Close end of the day is a widespread strategy, especially for manual scalping traders, it is basically a standard long-short trading strategy, but all the positions opened within a day should be closed before the end of the day, if a position is still opened at the last instant of the trading time, it is automatically closed by the system. In all the trading strategies applied the trading system apply the same Stop Loss.
Here, we have followed the same approach used to select the most performing model explained in the previous section, thus we had launch different trading simulation on different prediction time and compared the results in order to select the trading strategy that in most of the cases obtains higher returns.
In image 5.4, it is shown the equity line of the four different strategies in the same period of time. As it can be observed, Max opened and Fixed Fractional strategies are by far the best approaches, and this is true in all the experiments done in others prediction and validation methods too. Interesting to notice that the evolution of the equity line on the two strategy is quite similar, almost the same, indeed, as previously said , the two strategies are very similar. In particular in the case reported the fixed fractional amount is equal to 10% of the capital (several experiments varying the fixed fractional have been tested, and 10% has resulted the best in most of the cases) and max opened is equal to 10. The signals in input are the same in both strategies and they invest almost the same amount, since max opened is set to 10 it will invest 10% at each stock, however the main difference in this case it is that with fixed fractional we can have opened more than 10 positions if we have enough capital while with max opened this it will never be possible. This is the main reason of practically the same evolution and results of the equity line in the two cases. Then we can see that the standard long short strategy is able to obtain positive results, however far away from the ones obtained by the first two strategies (44% vs 95% and 86% respectively max opened and fixed fractional). The worst strategy is resulted to be Close end of the Day, registering a negative return of -24%.This result is particularly negative for short-term predictions, as the case in the image 30 minute predictions, because by closing at the end of the day much more operations are opened and closed and thus the fees drastically increases, it should indeed it should be remembered that the fees are paid both when the position is opened but even when is closed. For higher predictions, for instance for 1 day trading, close end of the day strategy has obtained much higher results, however Max opened is still the best strategy in all the configurations experimented.

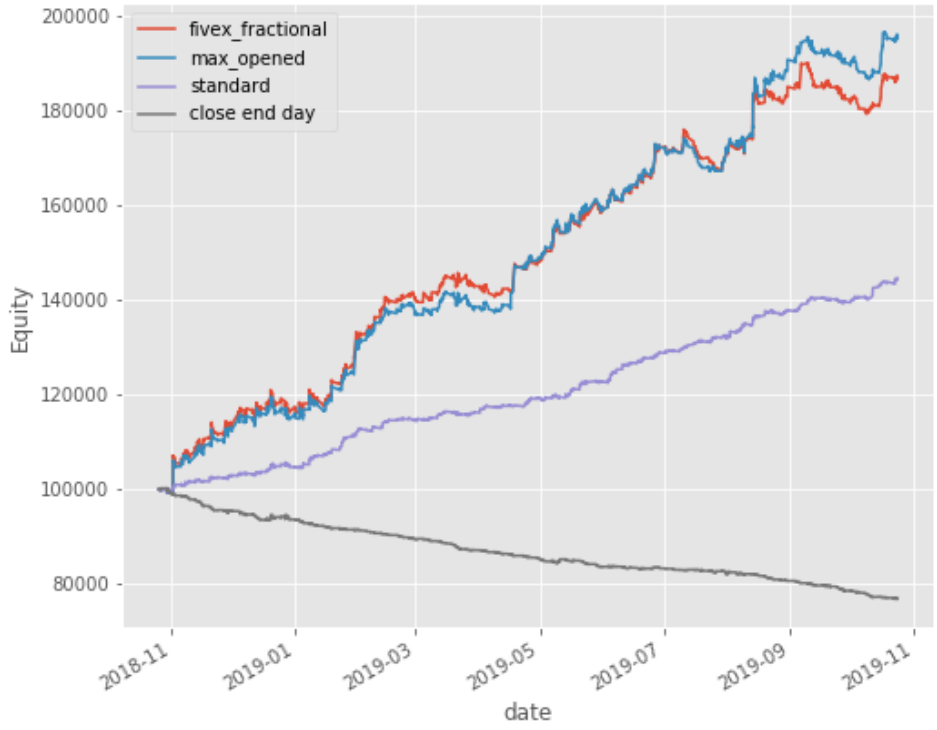


Figure 5.4: Evolution %return 30 min trading in the NASDAQ for 4 different trading strategies from 2018-11-12 to 2019-10-24, model configuration: 30-30-120 EW

So all the experiments that we are going to see from this point forward have been done using an MLP model and a trading strategy max opened equal to 10.

5.4 Stop Loss variation impact

Stop loss is a risk management tool that is used by traders and automatic trading system to limit losses in every single trade. From a technical point of view, a stop loss is a limit order, buy or sell, sent to the intermediary (brokerage platform) at the same time as the entry level of the trade, the broker will automatically close the position when the losses indicated by the stop loss is reached. Stop loss allows to establish before trading the maximum amount of capital that a person is willing to lose with a specific operation.

Therefore, whether the stop loss is applied or not, and its level, has a great impact on overall performance. Applying the stop loss can change the returns from negative to positive. Also the level of stop loss is particularly significant, a stop loss of 5% has a completely different effect than a stop loss of 1%. Furthermore, its level strongly depend on the granularity we are intend to invest. If it is invested in the short time, prediction intraday, the variation of the price is much lower than over a longer period of time such as daily. For instance, if we have a stop loss of 1% and using a trading system based on 5 minute signals, it is very unlikely that a position would close for stop loss, in short minute trading (5,10 minutes trading for example) a movement of the price of 1% is very rare. On the other hand, in a trading system with one day signal trading a stop loss of 1% is much more suitable, because within a day it is very likely that the price will change by 1%.
Since this research experiments trading on different prediction times, from minutes to day, different values of stop loss should be identified depending on the prediction time, in order to obtain better performance. Therefore, we have implemented a method to identify the most suitable stop loss for each prediction time. This is based on the distribution of the variation of the price of the Nasdaq Index on the different granularities examined.
In the image 5.5 below is shown the distribution of the variation of the price of the Nasdaq index using a 30 minute granularity. For each granularity and thus prediction time, the standard deviation (sigma) and 1.5sigma has been computed. If the distribution follows a typical Normal distribution, the value between the average and +/- sigma are around 68%, and for 1.5 sigma includes around 82%. Making the assumptions that the variation of the price distribution on different granularity can be approximated to a Normal Distribution, if we take sigma and 1.5 sigma as Stop Loss values we should be able to include most of the variation of the price without closing each operation for stop loss cause.

Actually, empirically we have shown that the distribution of the price for different granularity is not perfectly approximated with a Normal Distribution, as shown in the image 5.6, where it is fitted a normal distribution to the empirical data. As can be observed, the empirical distribution has a shape similar to the Normal Distribution, however, it is much denser in the values around the average. For this reason, we have experimented that the values of sigma and 1.5 sigma include around 90% and 95% of the values. Although the approximation of the Normal Distribution is not perfect, we have considered sigma and 1.5 sigma good stop loss values, because it should be remembered that the implemented system does not close the position at the following time t+1 and it is opened (t), but it can keep the position opened for longer time.

In the table 5.9 are shown all the obtained values of stop loss for different prediction times applying the method just described. For each of these values and prediction experiments have been carried out to identify what is the most suitable

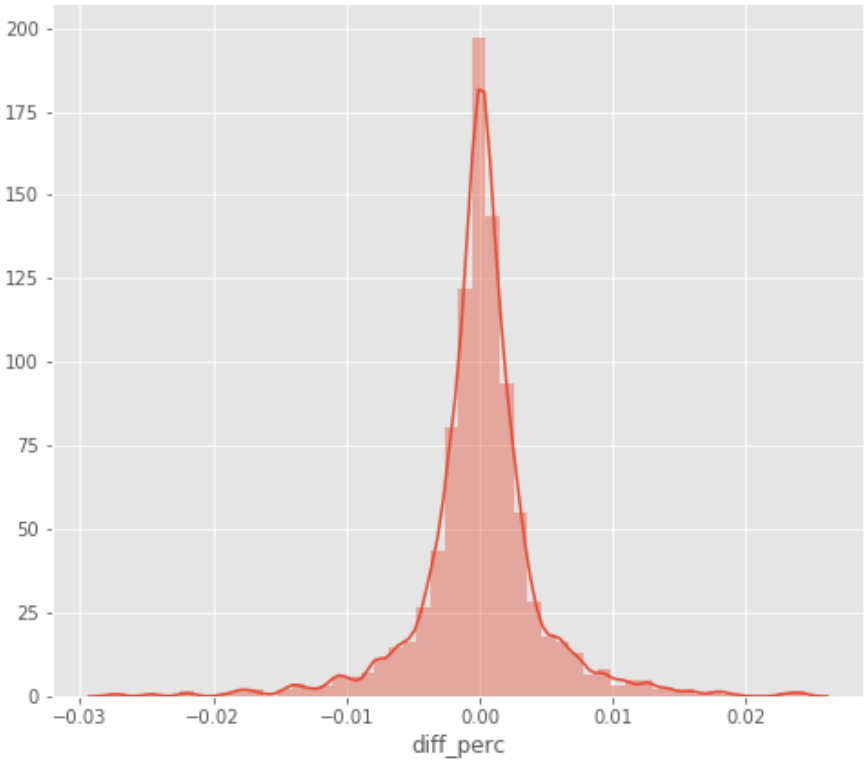


Figure 5.5: Evolution %return 30 min trading in the NASDAQ using different Stop Loss values 2019-07-11 to 2019-10-23, model configuration: 30-30-120 HO

value of Stop Loss for each configuration. As we could expect and see in table longer is the prediction time and higher is the variability of the price.

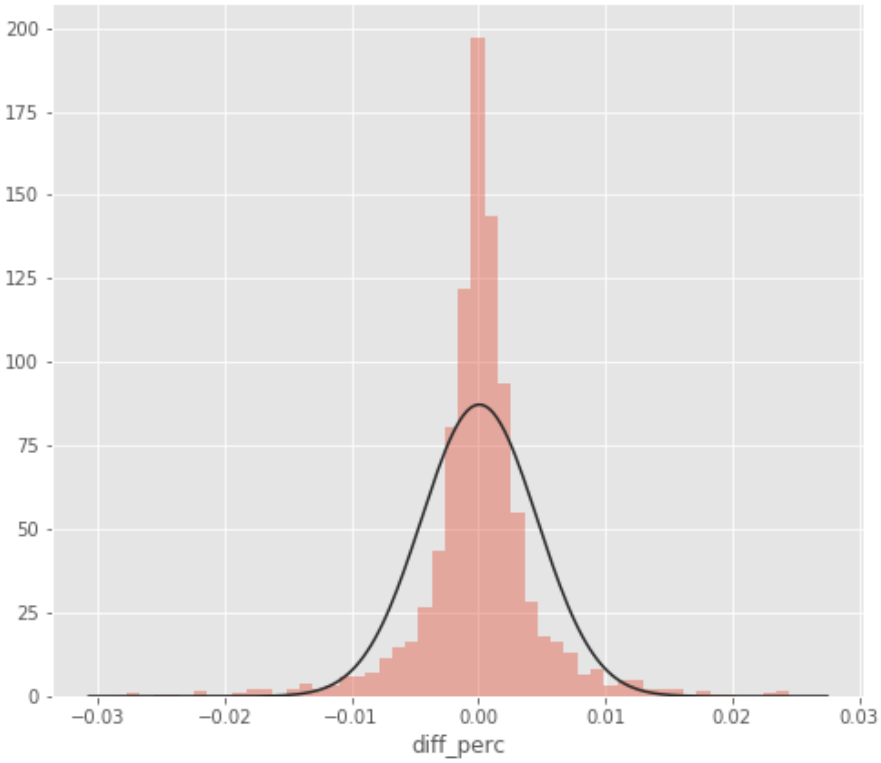


Figure 5.6: Empirical Distribution of the % price difference of Nasdaq with 30 minute granularity compered with a Normal Distribution

Table 5.9 Stop Loss values experimented on the trading system

predictions(min)	sigma	1.5 sigma
5	0.142%	0.213%
10	0.197%	0.296%
20	0.272%	0.408%
30	0.336%	0.504%
60	0.457%	0.686%
120	0.548%	0.822%

For daily prediction the stop loss has been kept to 1%, following other similar

studies [4].
In image 5.7 are shown the different equity lines for different values of stop loss, all the other variables are kept fixed, i.e MLP model and strategy Max Opened as we have identified and explained in previous sections. As it can be observed from the image, the return is higher when the stop loss is equal to sigma, and it shown that higher is the Stop Loss lower is the profitability. This has resulted true in all the prediction times experimented.

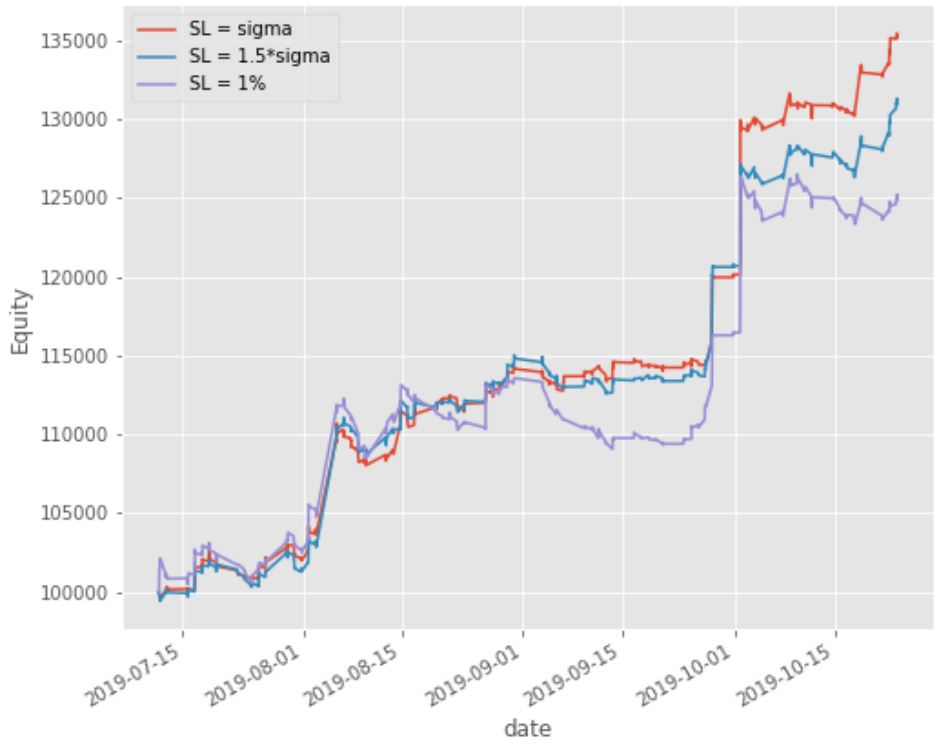


Figure 5.7: Empirical Distribution of the % price difference of Nasdaq with 30 minute granularity compered with a Normal Distribution

5.5 Expanding Window versus Holdout

As we have mentioned several time throughout the thesis, in particular in Ch 4, the two validation methods applied, expanding windows and holdout, are trained and tested on different period of time. Expanding window is tested on most of the data

(from 11/12/2018 to 23/10/2019, almost 11 months of price information), while the Hold-out approach is tested only on the last 30% of the data, from 11/07/2019 to 23/10/2019, 3 month and a half. This chapter is not intended to explain the difference between the two methods, because this has already been seen in chapter 4, here we want to show which of the two methods is more effective in the same period of time. The only way to compare them is to 'restrict' the period of trading for the Expanding Window in the same period the Holdout has been testes, i.e from 11/07/2019 to 23/10/2019. Doing so allows to make a valid comparison in order to understand which validation method performs better. Logically it can be thought that Expanding Window will get better results, since its trained on more information and a new model is created at every instant. However, there are many other variables that can affect performance, particularly Expanding Window method may suffer from overfitting problem in the initial training period, as it is trained on limited information. Furthermore it could have some bias on the most recent data (the instant immediately before the next prediction) that are not included in the Hold-Out. The only way to verify which of the two obtains better results in the different setups is through an empirical experiment, which we have carried out and the results are gathered in table 5.10.

Table 5.10 Expanding Window versus Holdout profitability results on Nasdaq from 11/07/2019 to 23/10/2019

prediction	**HO**	**EW**
30min	**35.23%**	24.47%
60min	8.62%	**20.98%**
120min	19.91%	**31.24%**
1day	**4.86%**	3.47%

As can it be observed from the table above, there is no a clear 'winner' for some configurations EW performs better (120 and 60 minutes), while for others (30 and 1 day) HO obtains better returns. Nevertheless, Expanding Windows seems in general to obtain slightly better results, even when HO has higher returns, the difference between the two is not as great as when EW is better. In image 5.9 is shown the configuration with 30 minute prediction where HO performs better, while in image below 5.8 prediction of 120 minutes where EW obtains much higher results. However here we do not want to make any conclusion about which of the two validation methods is better in terms of profitability and furthermore we do not have enough data to support either validation methods. We will resume this comparison only once a complete analysis of the performance achieved in the different granularity and evaluation methods is carried out, as we will show in the next section.

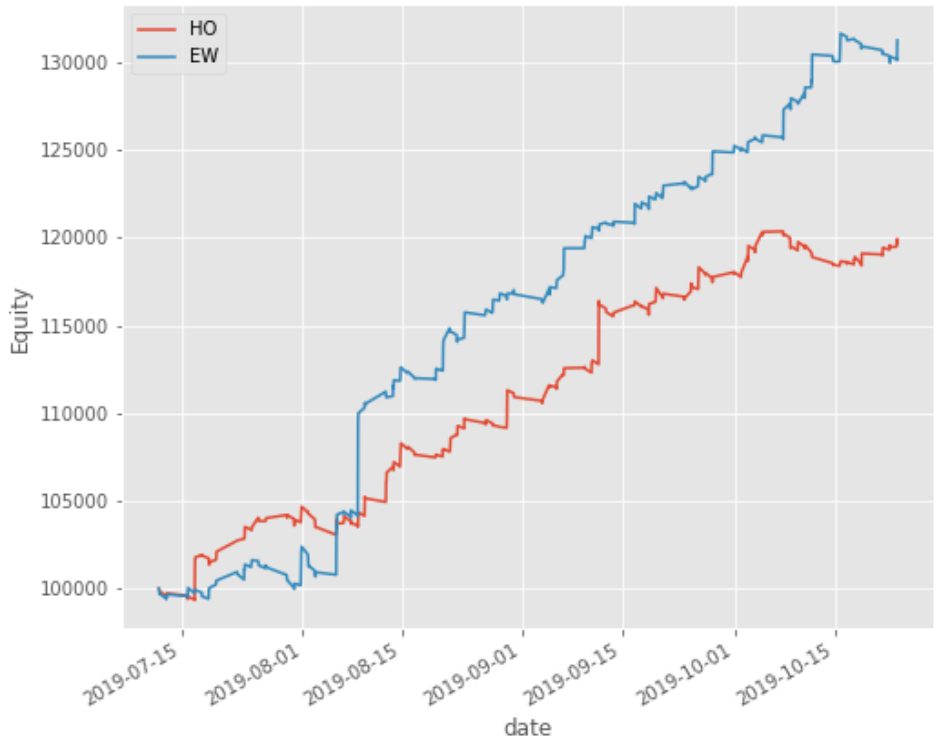


Figure 5.8: Evolution equity line for Holdout vs Expanding Window trading system, 120 min trading in the NASDAQ from 2019-07-11 to 2019-10-23

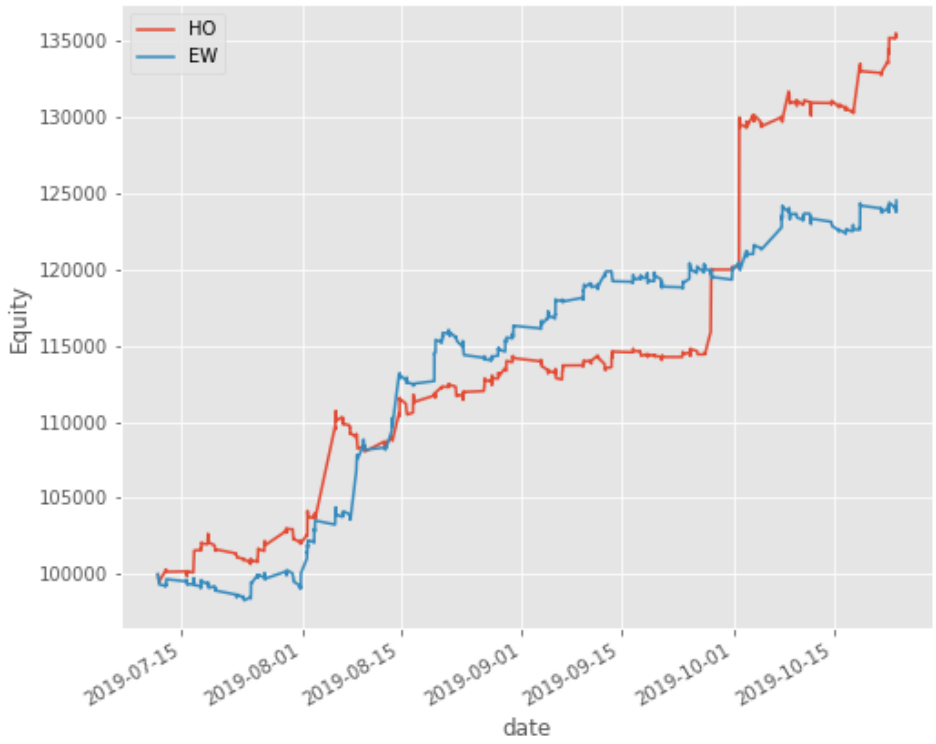


Figure 5.9: Evolution equity line for Holdout vs Expanding Window trading system, 30 min trading in the NASDAQ from 2019-07-11 to 2019-10-23

5.6 Intraday versus Multi-day trading

One of the goals of this thesis is to understand if it is more profitable use a trading system with predictions in the short term, here considered intraday, or in the long term, i.e daily prediction. With the experiments carried out on the same trading period is possible to compare the percentage return for daily and intraday prediction systems implemented. However, it should be reminded, that HO and WF (or Expanding Window) use two different test set, Expanding Window 11/12/2018 to 23/10/2019 while Holdout from 11/07/2019 to 23/10/2019. For this reason, unfortunately, it is not possible to compare intraday versus daily predictions results obtained in the two Validation approached together, they must be divided. In the previous section we have already compared the two Validation methods predictions against each other, but we had to narrow down the time of the Expanding Window method, taking into account only the prediction over three months out of eleven. Here we do want to include all the available data because the main scope is not to compare the two validation approaches, because this has already been done in the previous chapter, but to compare the results obtained using a daily prediction respect the results obtained using intraday predictions. In the following the results obtained applying Holdout and Expanding Windows are shown and described separately in detail.

5.6.1 Holdout

In table 5.11 below are shown the profitability return obtained over 11/07/2019 to 23/10/2019, signals generated through a Multiple Layer Perceptron, using a strategy max opened with maximum number operations opened equal to 10, and the Stop Loss is equal to the standard deviation of the price difference respect the granularity taken into account, whereas in the table below 5.12 are reported some metrics about the opened trades, such as number of operations in profit and loss, operations in short and long and the total feed paid. As it can be observed from the results on both table, for short prediction times (5, 10, 20 minutes) event the most performing model and configurations of the trading system are not able to generate profit. All of them have a negative return, the shorter is the prediction period the performance get worse. For instance, for 5 minutes predictions, we can see that the return is around -26%, which is really a poor result considering that trading time is over a short period, around three month and a half. According to our analysis this is due to the fact that at short predictions time the number operations opened and closed within a day are considerably higher respect trading system based on prediction over 30 or more minutes. For instance, the trading system based on prediction of 5 minutes opens 13693 transactions, while if we use 30 minutes predictions the number of operation opened in the same period of time are almost a ninth of this (1839). Nevertheless, looking at the return obtained on longer prediction times, the results get better, and for 30 minutes granularity upwards all get positive performance. The highest return is obtained with 30

minutes predictions, reaching a 35% return on capital and a max drawdown of only -2.41%, which it is a quite astonishing results over only three months of trading. Even 60, 120 minutes predictions achieve good performance. Regarding daily prediction, instead, the return is around 5%, which it is still a good results, but not comparable respect the results obtained using lower prediction times. Furthermore, it is interesting to notice that although daily prediction achieve a positive return the max drawdown overcomes it, this is a clear sign that the trading system based on daily prediction is risky and not completely reliable.
Furthermore, looking at operation statistics metrics in table 5.12 it is interesting to notice that the number of operations in profit respect the operations in loss is pretty low, their ratio is around 25-35%. This means that only around 25-35% of the operations opened in the market are actually profitable. This is explained by the fact that in general the SL is set at a low level and therefore many operations are closed due to the SL, but this does not seem to particularly affect the overall performance, because the operations in which we are profitable have a much higher average return than the loss operations, so the system is able to exploit the so-called 'big movements' of the market.
In conclusion, trading system based on intraday predictions for not too short predictions (30, 60, 120 minutes) outperforms the returns achieved by a trading system based on daily predictions, as shown in image 5.10.

Table 5.11 Profitability performance Hold-out on Nasdaq from 2019-07-11 to 2019-10-23

prediction	**% return.**	**max drawdown**	**daily-operations**
5min	-25.98%	-26.48%	132
10min	-13.08%	-13.95%	84
20min	-9.94%	-14.37%	44
30min	**35.23%**	-2.41%	18
60min	8.62%	-4.599%	16
120min	19.91%	-1.63%	9
1day	4.86%	-5.11%	6

prediction: the underlying model frequency prediction (min); *% return*: percentage return on the capital invested in the trading period; *max-drawdown:* computed as the highest loss from a previous equity peak in the period considered; *daily-operations:* average number of transactions per day

Table 5.12 Operation stats metrics - Holdout

pred	operations	%profitable	fees	short	long
5min	13693	25.3%	42951.46	6507	7186
10min	8685	26.3%	23937.70	4747	2286
20min	4599	26.6%	13354.54	2627	1225
30min	1826	31.4%	6295.41	687	574
60min	1702	31.1%	5470.99	974	529
120min	960	34.5%	3203.18	538	331
1day	726	36.4%	2313.96	406	264

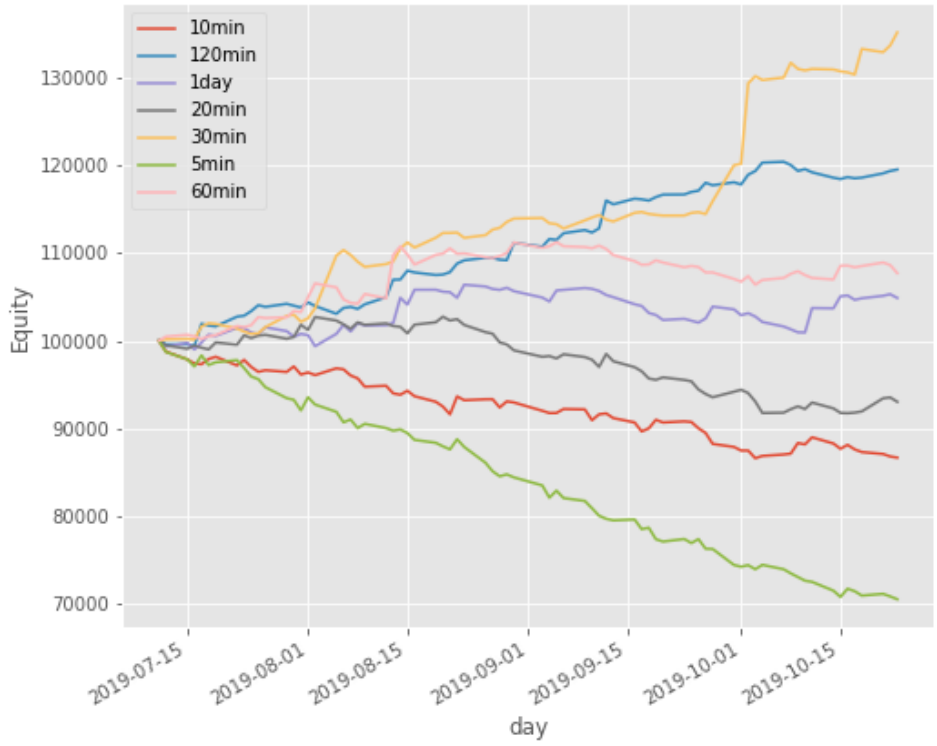


Figure 5.10: Evolution equity lines for different predictions time, trading on NASDAQ 100 from 2019-07-11 to 2019-10-23, HO

5.6.2 Expanding Window

The advantage of applying Expanding Window as training and validation method is the size of the test period which allows to obtain more robust results, in particular respect the Hold Out method. Here, all the trading simulation have been executed from 11/12/2018 to 23/10/2019, on more than 11 months of data. As it can be observed on table 5.13, the return on capital is strongly positive for all prediction configurations, the highest performance is achieved on 120 minutes trading system prediction (149%). The difference among 30 and 60 minutes predictions are not significant to claim either to shorter or longer predictions the performance gets better. Here again, similarly to what we have observed using the Holdout method, the daily trading system has performance significantly lower respect the ones obtained in shorter predictions system, for instance for 30 minutes 87% versus a miserably 4.9% return obtained on daily prediction. The metrics on open market

operations, shown in 5.14, open trades that are profitable are about 35% in all configurations. This is a slightly better result than in the HO method, but still not astonishing, many operations are closed due to a low SL set. Although the number of profitable trades is not so high the final profits are positive, this means that the system when it is 'right' opens very profitable trades.
To sum up, intraday trading returns far exceed daily returns, meaning that in terms of profitability it would be more convenient and less risky using a trading system based on intraday prediction rather than one based on daily predictions. In image 5.11 the equity lines obtained in the different periods of predictions are shown and compared using the Expanding Window method. As can be seen from the image, 120-minute predictions outperforms all other configurations.

Table 5.13 Profitability performance Expanding Window on Nasdaq from 2018-12-11 to 2019-10-23

prediction.	**% return.**	**max drawdown.**	**daily-operations**
30min	86.96%	-2.93%	23
60min	83.47%	-3.92%	12
120min	**148.97%**	-1.27%	10
1day	4.93%	-6.84%	6

prediction: the underlying model frequency prediction (min); *% return*: percentage return on the capital invested in the trading period; *max-drawdown:* computed as the highest loss from a previous equity peak in the period considered; *daily-operations:* average number of transactions per day

Table 5.14 Operation stats metrics - Expanding Window

pred.	**operations**	**%profitable**	**fees**	**short**	**long**
30min	7192	31.8%	30356.3	3549	3643
60min	3769	32.3%	15130.3	1823	1946
120min	3272	37.3%	16343.8	1473	1799
1d	2180	37.6%	7284.6	1023	1157

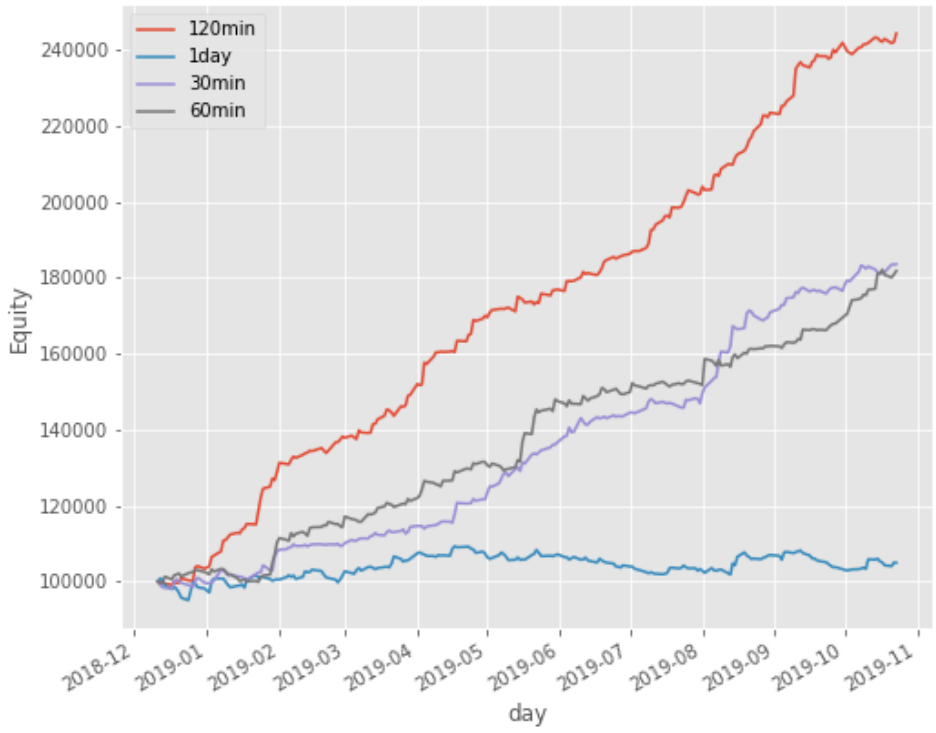


Figure 5.11: Evolution equity lines for different predictions time, trading on NASDAQ 100 from 2018-12-11 to 2019-10-23

Chapter 6

Conclusion

In this research have been studied how machine learning classification models perform on predicting the movement of the stock's price in the short and long term (intraday vs multi day). In particular, predictability and profitability performance are studied in detail and compared in the two different time. In order to carried out a complete empirical study in this field we had first to study in depth the literature in this field. Indeed, in Chapter 2 are explained in detailed four statistical machine learning models that have been selected to use in the empirical study. For each of them is described its basic functioning, how the model effectively work in a classification task and its most important parameters. Following are explained the evaluation metrics used to assess model's performance and finally the most used validation methods with their advantages and disadvantages. This chapter aims to show the pros and cons of each model, evaluation metrics and evaluation method. This study has allowed us to select the best metrics and evaluation methods to use in our particular time series prediction problem. Furthermore, it gives the basic steps to follow in a typical machine learning problem: selection of the model, training, validation method and finally evaluation of performance through some statistical metrics.
In Chapter 3, our literature study continues on studying in more detail the price forecasting problem by reviewing several studies of intraday and multi day forecasting present in the literature. The main steps involved in the researches are reported and described. In this chapter we have attempted to compare the different techniques used for predicting in the short and long term. This study has surprisingly pointed out that the techniques used in the two time scenario, that includes models, features, evaluation methods and metrics, are generally the same. The main difference between the two is simply the data formatting fed into the model. In the end of the chapter we tried to compare performance of different studies in short and long term. However it is not possible to make a real and significant comparison, because each studies use different period of time, market and models. There are too many variables to be taken into account and in order to make a valid comparison these variables should be keep fixed, especially for the trading window,

stock and machine learning model used, in both scenario, short vs long term. The unsuccessful comparison of performance from similar studies in the literature gives more strength to our study and research question. In fact, it has allowed us to understand that a complete study comparing short and long term performance does not exist in the literature (to the best of our knowledge) and therefore to answer our research question an empirical system must be implemented accordingly.
Following the conclusion of chapter 3, in Chapter 4 is explained in depth how our forecasting and market simulation system has been developed, describing in detail all the steps involved. In this chapter the typical steps followed in a machine learning classification problem are deployed. First the collection of the data and its pre-processing, which includes cleaning, check for inconsistency in the data and normalization. Also, the technical indicators selected and added to the dataset are explained in detail. Following we go into more detail about data manipulation showing how to change data granularity that is essential to experiments on different time frame. Here we explain why we have chosen f1-score validation metric to measure the profitability performance and why we selected hold out and walk forward evaluation methods. In the end, it is described how the trading simulation deployed works and which trading strategy we have used to evaluate the % of return over a period of time.

Finally, in Chapter 5 all the experiments carried out and their respective results are shown and described in detail. First, the performance in terms of predictability results, measured through f1-score are studied, in order to identify before starting experiments with the trading system, whether the signals generated by the machine learning models and their ad-hoc configurations are accurate or not. This has helped us to identify which machine learning model has in general achieved the highest performance in the different configurations. In general, we have found that a Multiple Layer Perceptron has higher ability to predict respect all the others machine learning classification models tested (DT, XGB, SVC). Instead, regarding the experiments carried out varying granularities and prediction time, it has been observed that in general shorter is the prediction time higher are the performance in terms of predictability. The relationship between prediction period and f1-score appears to be indirect, when prediction time increases (passing from 30 minutes to 60 minutes for example) f1-score decreases, this is true almost for all except few cases.
Following, all the experiments and results carried out through the trading system implemented are reported. Several variables have been tested, starting from selecting which machine learning classification model has achieved highest return. In line with the predictability results obtained through f1-score, also profitability results shown that the Multiple Layer Perceptron is the most suitable algorithm, in terms of return of capital. Then, keeping fixed the model, several popular trading strategies, including max opened, fixed fractional, close end of the day and simple long-short, have been compared. Our results shown that in almost all time

prediction configurations the max opened trading strategy, with 10 max opened operations, outperforms all the other strategies.
Moving on, keeping fixed the machine learning model and now the strategy, we have experimented what is the most suitable stop loss value on different granularity. Here, through an empirical method ad-hoc implemented based on the distribution of the price difference of the price some feasible stop loss values in the different granularity have been tested. Comparing the results we found for each prediction period the value of the stop loss that gets the highest returns, which turned out to be equal to the standard deviation of the distribution of the price difference between two following instats.
Knowing the best value of the stop loss, machine learning model and trading strategy, we compared the profitability results achieved using the two different validation methods applied: Expanding Window versus Holdout. In order to make a valid comparison, we had to narrow down the Expanding Window test set and take only the signals within the same period of the Holdout test set. From the results obtained in the two approaches, it is not simple to draw a conclusion, for some granularity configuration EW outperforms HO, for other is the opposite. However, we have pointed out, that when EW gets better results, the difference with HO is much higher than when HO achieves higher returns.
Finally, the main research question, i.e identify whether it is more convenient in terms of profitability apply a trading system based on intraday predictions rather than using daily predictions, we have compared the results obtained over the same period of time for daily trading versus intraday trading. Here the results are divided based on the Validation method applied, since the trading time considered is different. However, the results obtained in the two approaches are in line, and show that intraday trading outperforms daily trading by far. This, however, is not always true, for very short predictions (5,10, 20 minutes) the returns are negative and thus daily predictions outperforms very short trading system. This is explained by the fact that trading in such a short time, the market transactions exponentially increase and therefore also the fees.

To summarize, the main takeaways of this research are:

- Identified the major drawbacks of the EW method applied to high granularity data;
- Predictability and profitability results are in line showing that Multiple Layer Perceptron model achieved the highest performance in practically all the data granularity tested;
- Empirically experiments has shown that the Max opened trading strategy outperform all the other strategies tested;
- Impact of stop loss on different granularity and how to select a stop loss value that maximizes profit;

- Comparing the performance of EW versus HO on different granularity, showing that in the granularity where are achieved better results EW outperform HO;
- Trading system based on short-term predictions are more profitable than trading system based on long-term predictions

However, it should be said that this research present some limitations and drawbacks which could be interesting to analyze for future works. The main drawbacks is that we were not able to finish to train all the models for time and technical constraints of all the configurations using the Expanding Window approach. Therefore, unfortunately in this validation method we do not have a complete study and in particular we have not collected the performance for high granularity (from 20 to 5 minutes). Future works could extend this study by training machine learning models for so short predictions through an EW approach. This, however, as we have shown, it is possible only if you have enough computing capacity, using multiple parallel computing systems or you could rely on online Cloud Computing systems. Another possible opportunity for future works is to study more deeply the relation between the granularity of the data and the prediction time. This is a relationship that we have studied throughout the thesis, but always due to technical problems, we have not a complete study on all the granularity. Our results show that maintaining a granularity of information equal to the frequency of predictions is in general more accurate, but this is true only for few granularity and future studies could show the opposite. One last aspect that would be interesting to study in future works is the application of Deep Learning Classification models, such as RNN and LSTM, on different data granularity.

Although the research limitations, this research has been effectively able to compare how machine learning models perform forecasting in the short and in the long term and if it is better in terms of profitability trade in intraday or in a daily scenario. Furthermore, through several experiments we have been able to identify the most accurate machine learning model and the trading strategy and stop loss values that achieve the highest return on capital on different predictions time.

Appendix A

Model's Hyperparameters

```
    "SVC": [{
          'kernel': ['rbf'],
          'C': [0.01, 1, 100],
          'gamma': [0.01, 1, 10]
     },
          {
               'kernel': ['linear'],
               'C': [0.01, 1, 10]
          }],
     "SVC-reduced": {
          'kernel': ['linear'],
          'C': [10]
     },
     "DT": {
          'criterion': ['gini', 'entropy'],
          'max_depth': [5, 20],
          'min_samples_split': [2, 10]
     },
     "MLP": {
          'hidden_layer_sizes': [(20,), (100,), (100, 100)],
          'solver': ['lbfgs'],
          'max_iter': [200]
     },
     "XGB":{}
```

Bibliography

[1] Michael McGowan. «The Rise of Computerized High Frequency Trading: Use and Controversy». In: *Duke Law and Technology Review* 16 (Nov. 2010) (cit. on p. 1).

[2] Meredith Beechey, David Gruen, and James Vickery. «The Efficient Market Hypothesis: A Survey». In: (Feb. 2000) (cit. on p. 2).

[3] Alexandra Ţiţan. «The Efficient Market Hypothesis: Review of Specialized Literature and Empirical Research». In: *Procedia Economics and Finance* 32 (Dec. 2015), pp. 442–449. DOI: 10.1016/S2212-5671(15)01416-1 (cit. on p. 2).

[4] Luca Cagliero, Paolo Garza, Giuseppe Attanasio, and Elena Baralis. «Training ensembles of faceted classification models for quantitative stock trading». In: *Computing* (Jan. 2020), pp. 1–13. DOI: 10.1007/s00607-019-00776-7 (cit. on pp. 2, 20, 64).

[5] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok. «A Novel Ultrathin Elevated Channel Low-temperature Poly-Si TFT». In: 20 (Nov. 1999) (cit. on p. 3).

[6] *Interpreting a Strategy Performance Report*. 2019. URL: https://www.investopedia.com/articles/fundamental-analysis/10/strategy-performance-reports.asp (cit. on p. 5).

[7] M. Bharati and Bharati Ramageri. «Data mining techniques and applications». In: *Indian Journal of Computer Science and Engineering* 1 (Dec. 2010) (cit. on p. 6).

[8] Christopher Matheus, Philip Chan, and Gregory Piatetsky-Shapiro. «Knowledge Discovery in Databases». In: *IEEE Transactions on Knowledge and Data Engineering* (Dec. 1998) (cit. on p. 6).

[9] J E T Akinsola. «Supervised Machine Learning Algorithms: Classification and Comparison». In: *International Journal of Computer Trends and Technology (IJCTT)* 48 (June 2017), pp. 128–138. DOI: 10.14445/22312803/IJCTT-V48P126 (cit. on p. 7).

[10] Lior Rokach and Oded Maimon. «Decision Trees». In: vol. 6. Jan. 2005, pp. 165–192. DOI: 10.1007/0-387-25465-X_9 (cit. on p. 8).

[11] Abbas Alharan, Radhwan Alsagheer, and Ali Al-Haboobi. «Popular Decision Tree Algorithms of Data Mining Techniques: A Review». In: *International Journal of Computer Science and Mobile Computing* 6 (June 2017), pp. 133–142 (cit. on p. 8).

[12] Theodoros Evgeniou and Massimiliano Pontil. «Support Vector Machines: Theory and Applications». In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12 (cit. on p. 8).

[13] Enzo Grossi and Massimo Buscema. «Introduction to artificial neural networks». In: *European journal of gastroenterology hepatology* 19 (Jan. 2008), pp. 1046–54. DOI: 10.1097/MEG.0b013e3282f198a0 (cit. on pp. 9, 11).

[14] Juergen Schmidhuber. «Deep Learning in Neural Networks: An Overview». In: *Neural Networks* 61 (Apr. 2014). DOI: 10.1016/j.neunet.2014.09.003 (cit. on p. 11).

[15] Hassan Hassan, Abdelazim Negm, Mohamed Zahran, and Oliver Saavedra. «ASSESSMENT OF ARTIFICIAL NEURAL NETWORK FOR BATHYMETRY ESTIMATION USING HIGH RESOLUTION SATELLITE IMAGERY IN SHALLOW LAKES: CASE STUDY EL BURULLUS LAKE.» In: *International Water Technology Journal* 5 (Dec. 2015) (cit. on p. 12).

[16] Yakup Kara, Melek Boyacioglu, and Omer Baykan. «Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange». In: *Expert Systems with Applications* 38 (May 2011), pp. 5311–5319. DOI: 10.1016/j.eswa.2010.10.027 (cit. on p. 12).

[17] Eunsuk Chong, Chulwoo Han, and Frank Park. «Deep Learning Networks for Stock Market Analysis and Prediction: Methodology, Data Representations, and Case Studies». In: *Expert Systems with Applications* 83 (Apr. 2017). DOI: 10.1016/j.eswa.2017.04.030 (cit. on pp. 12, 20).

[18] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. «Predicting stock market index using fusion of machine learning techniques». In: *Expert Systems with Applications* 42 (Oct. 2014). DOI: 10.1016/j.eswa.2014.10.031 (cit. on p. 12).

[19] Jerome Friedman. «Greedy Function Approximation: A Gradient Boosting Machine». In: *The Annals of Statistics* 29 (Nov. 2000). DOI: 10.1214/aos/1013203451 (cit. on p. 12).

[20] David Powers and Ailab. «Evaluation: From precision, recall and F-measure to ROC, informedness, markedness correlation». In: *J. Mach. Learn. Technol* 2 (Jan. 2011), pp. 2229–3981. DOI: 10.9735/2229-3981 (cit. on p. 13).

[21] Nathalie Japkowicz and Mohak Shah. «Performance Evaluation in Machine Learning». In: Jan. 2015, pp. 41–56. ISBN: 978-3-319-18304-6. DOI: 10.1007/978-3-319-18305-3_4 (cit. on p. 15).

[22] Daniel Berrar. «Cross-Validation». In: Jan. 2018. ISBN: 9780128096338. DOI: 10.1016/B978-0-12-809633-8.20349-X (cit. on p. 16).

[23] Roberto Parado. *Design, Testing, and Optimization of Trading Systems*. 1992 (cit. on p. 17).

[24] Lv Dongdong, Zhenhua Huang, Meizi Li, and Yang Xiang. «Selection of the optimal trading model for stock investment in different industries». In: *PLOS ONE* 14 (Feb. 2019), e0212137. DOI: 10.1371/journal.pone.0212137 (cit. on p. 17).

[25] Jose Matias and Juan Reboredo. «Forecasting Performance of Nonlinear Models for Intraday Stock Returns». In: *Journal of Forecasting* 31 (Mar. 2012). DOI: 10.1002/for.1218 (cit. on pp. 19–21, 33, 34, 51).

[26] Tomer Geva and Jacob Zahavi. «Predicting Intraday Stock returns by Integrating Market Data and Financial News Reports.» In: Jan. 2010, p. 39 (cit. on p. 20).

[27] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. «Evaluating multiple classifiers for stock price direction prediction». In: *Expert Systems with Applications* 42 (May 2015). DOI: 10.1016/j.eswa.2015.05.013 (cit. on p. 20).

[28] Mohamed Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. «A Survey of Classification Methods in Data Streams». In: Apr. 2007, pp. 39–59. DOI: 10.1007/978-0-387-47534-9_3 (cit. on p. 21).

[29] Eduardo A. Gerlein, T.M. Mcginnity, Ammar Belatreche, and Sonya Coleman. «Evaluating machine learning classification for financial trading: An empirical approach». In: *Expert Systems with Applications* 54 (Feb. 2016). DOI: 10.1016/j.eswa.2016.01.018 (cit. on p. 21).

[30] Matthew Dixon, Diego Klabjan, and Jin Bang. «Classification-Based Financial Markets Prediction Using Deep Neural Networks». In: *Algorithmic Finance* (Dec. 2017). DOI: 10.2139/ssrn.2756331 (cit. on p. 21).

[31] *Thomson Reuters API*. URL: https://developerportal.thomsonreuters.com/home (cit. on p. 27).

[32] Sotiris Kotsiantis, Dimitris Kanellopoulos, and P. Pintelas. «Data Preprocessing for Supervised Learning». In: *International Journal of Computer Science* 1 (Jan. 2006), pp. 111–117 (cit. on p. 27).

[33] Hubert Anysz, Artur Zbiciak, and Nabi Ibadov. «The Influence of Input Data Standardization Method on Prediction Accuracy of Artificial Neural Networks». In: *Proceedia Engineering* 153 (Jan. 2016). DOI: 10.1016/j.proeng.2016.08.081 (cit. on p. 28).

[34] Raymond Chiong, Zongwen Fan, Zhongyi Hu, Marc Adam, Bernhard Lutz, and Dirk Neumann. «A sentiment analysis-based machine learning approach for financial market prediction via news disclosures». In: (July 2018), pp. 278–279. DOI: 10.1145/3205651.3205682 (cit. on p. 29).

[35] Tomer Geva and Jacob Zahavi. «Predicting Intraday Stock returns by Integrating Market Data and Financial News Reports.» In: (Jan. 2010), p. 39 (cit. on p. 29).

[36] Felipe Oriani and Guilherme Coelho. «Evaluating the impact of technical indicators on stock forecasting». In: (Dec. 2016), pp. 1–8. DOI: 10.1109/SSCI.2016.7850017 (cit. on p. 29).

[37] John J Murph. *Technical Analysis Of The Financial Markets*. 1998. URL: https://www.academia.edu/4075580/John_J_Murphy_Technical_Analysis_Of_The_Financial_Markets (cit. on p. 29).

[38] Yauheniya Shynkevich, T.M. Mcginnity, Sonya Coleman, Ammar Belatreche, and Yuhua Li. «Forecasting Price Movements using Technical Indicators: Investigating the Impact of Varying Input Window Length». In: *Neurocomputing* (June 2017). DOI: 10.1016/j.neucom.2016.11.095 (cit. on p. 30).

[39] Bartosz Krawczyk. «Learning from imbalanced data: Open challenges and future directions». In: *Progress in Artificial Intelligence* 5 (Apr. 2016). DOI: 10.1007/s13748-016-0094-0 (cit. on p. 30).

[40] Prabhjot Kaur and Anjana Gosain. «Comparing the Behavior of Oversampling and Undersampling Approach of Class Imbalance Learning by Combining Class Imbalance Problem with Noise». In: Jan. 2018, pp. 23–30. ISBN: 978-981-10-6601-6. DOI: 10.1007/978-981-10-6602-3_3 (cit. on p. 31).

[41] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. «SMOTE: Synthetic Minority Over-sampling Technique». In: *J. Artif. Intell. Res. (JAIR)* 16 (Jan. 2002), pp. 321–357. DOI: 10.1613/jair.953 (cit. on p. 31).

[42] Charalampos Stasinakis and Georgios Sermpinis. «Financial Forecasting and Trading Strategies: A Survey». In: Jan. 2014, pp. 22–39. ISBN: 978-0415636803. DOI: 10.4324/9780203084984 (cit. on p. 58).