Università
Ca'Foscari
Venezia

Master's Degree programme- Second Cycle in Economia e Finanza- Economics and Finance

Final Thesis

–
Ca' Foscari
Dorsoduro 3246
30123 Venezia

N-Tuple patterns

A recent method to forecast market movements, with applications

Supervisor
Ch. Prof. Marco Corazza

Graduand
Ilaria Maragno
Matriculation Number 838249

Academic Year
2015 / 2016

INDEX

INTRODUCTION **6**

A REVIEW OF THE PAST FINANCIAL TECHNIQUES **13**

1.1 THE PAST TECHNIQUES 13

1.1.1 The Efficient Market Hypothesis *14*

1.1.2 The Adaptive Market Hypothesis *19*

1.1.3 The Moving Average Hypothesis *21*

1.1.4 The Econometrics and Macroeconomic Factor approaches *24*

1.1.5 The Positive Feedback Strategy *26*

1.2 CONCLUSIONS 27

MALLIARIS' TECHNIQUE **29**

2.1 THE NOVEL APPROACH 29

2.1.1 Multi-decade patterns *33*

2.1.2 Forecasting *45*

2.2 CONCLUSIONS 54

APPLICATION TO THE ITALIAN INDEX COMIT GENERALE **56**

3.1 INTRODUCTION 56

3.2 MULTI-DECADE PATTERNS 57

3.3 FORECASTING 82

3.3.1 Past pattern strings applied to future data *83*

3.3.2 Decision Tree *92*

3.3.3 Logistic Regression and Neural Networks *97*

3.4 APPLICATION TO THE AMERICAN EXPRESS COMPANY 105

3.5 DIFFERENT TOOLS APPLIED TO THE S&P500 108

CONCLUSIONS **110**

APPENDIX A **113**

APPENDIX B **132**

APPENDIX C **136**

BIBLIOGRAPHY **141**

SITOGRAPHY **143**

INTRODUCTION

In this paper a new decision making strategy to forecast future market movements of stock prices, conditioned to the past patterns, will be presented, specifically patterns from one day up to seven, of a given index. The inspiration of such approach comes from the work of A. G. Malliaris, *"N-tuple S&P patterns across decades, 1950-2011".*

To start, in the first chapter, a review of the major financial techniques for forecasting market movements will be presented, taking as example the works cited by Malliaris; the analysis will be done in a critical way, in order to understand strengths and weaknesses of such theories, taking also examples of inefficiency in the prediction of market movements, like the one of the Dot-com bubble in section 1.1.1. The aim is to demonstrate that those approaches are not effective all the time.

In the second chapter, the novel approach of Malliaris is explained in details: he collected 60 years of data from the S&P500 index, classified each record as an up or a down movement with respect to the previous day, and finally he analysed the dataset decade by decade, in two different ways.
In the first approach, he counted all the different combination of up and down movements, for every string of days from one to six days in a row; for instance, in a two-day string four possible outcomes can happen: Up-Up (i.e. for two days in a row the market outperformed the day before), Up-Down, Down-Up and Down-Down.

In the second approach, Malliaris counted just the number of up days for every string taken into account: considering the previous example, in a two-day string there could be zero, one or two ups in a row.
In both cases the results are considered decade by decade.

After this first part of analysis, Malliaris moved to a forecast activity. The whole sample was split into Training and Validation set to do that.
Two forecasting approaches were used: a frequency approach and a decision tree.
The frequency approach is straightforward: on the training set, the more likely outcome is identified with the respect to what happened, historically, the K day before, with K the number of days taken into account (varying K, different prediction models arise). To start, all the different outcomes for every string of days are counted and collected into a table, which is then used to generate the forecast for the following day, looking at the behaviour of the previous one or ones. For instance, having K = 3, it is possible to predict how the market will move (up or down) looking at all the 3-days strings in the dataset, and counting the most recurrent outcome given the previous two. If, for example, the trend of the previous two days is Up-Up, to forecast whether in the following day it is more probable to have an up or a down, the observer can look at all the three-day strings in the table that have Up-Up in the first two places: between Up-Up-Up and Up-Up-Down, the most recurrent one is chosen to predict an up or a down.
The results of this analysis are applied to the Cross Validation set, to check whether such method generates good forecasts.

In addition to this approach, Malliaris proposed a decision tree, of which a better explanation is given in Appendix B: it is a statistical tool

that, at each step of the process, splits the original data into smaller subsets and then finds the constraints to the input variables (i.e. the features of the dataset) that best fit the data. The result of this analysis is a tree with a set of rules that ranks any input data with the same probability of having a specific target value.

To build the decision tree, data up to December 2009 were used, giving a correct forecast the 67% of the time on the training set. The same approach has been used with the validation set: the correctness of forecast dropped (about 57%) with respect to the training set, but it is still higher than the ones obtained with the simple pattern forecasts. Moreover the decision tree does a better job on predicting an up movement than on predicting a down one.

The whole work proves that certain conditional forecasts outperform the unconditional Random Walk model, demonstrating that prices do not move randomly but they follow a trend based on the past: market movements are not identically, independently distributed. Moreover, both the approaches used to forecast the price of the following day (the frequency method and the Decision Tree) work better in forecasting the up days, with a success of the 58% in the first case, and of the 61% in the second one.

In the third chapter the same approach of Malliaris will be applied on the Italian index Comit Generale, with data collected from the 2nd January 1973 to the 11th November 2016, and on the American Express Company stock price, with data from the 1st June 1972 to the 30th December 2016, in order to test the approach also with a company, that is more subject to suddenly price changes, using the Python programming language.

The first step of the analysis consists of ranking all the samples either as ups or downs, with respect to the closing price in the previous day;

then all the combinations of up and down are counted and displayed with the help of tables and graphs, for every possible product of ups and downs from one up to seven days, splitting the results decade by decade. For example, in the two-day string there are four possible combinations: Up-Up, Up-Down, Down-Up and Down-Down.
After that, a second ranking method is tested, counting just the number of up days for every time window in the dataset from one to seven days. Taking the same example, in the two-day string there could be zero (meaning that the combination for those days is Down-Down), one (meaning that the combinations for those days are Up-Down or Down-Up) or two (meaning that the combination for those days is Up-Up) ups. Those results are also read in a historical sense, in order to check if the Comit Generale adequately reflects the Italian situation.

In the second part of the chapter (paragraph 3.3) these patterns are used to predict future market movements: the aim is to test the features and to understand if they are significant for predicting future movements. The forecasting analysis is done with the same two methods used by Malliaris, so the application of past patterns to future data and the Decision Tree; moreover also Logistic Regressions (explained in details in Appendix B) and Neural Networks models (explained in Appendix C) are employed, in order to try to obtain even better performances.
To test the different models, the sample has been divided into Training set (the first 95% of data) and Validation set (the last 5% of data). For the application of past patterns to future data, all the combinations of up and down from one day to seven days in the training set are collected into a table, without the decade distinction. To select the most recurrent patterns, they have been analysed

couple by couple, where each couple differs just in the last day movement; the reasoning is simple: if *n* previous days are known, looking at past data, which outcome is more probable between up and down, in the *n+1* day? In this study, the maximum value for *n* is 6, while *n+1* will be the prediction for, at maximum, the 7^{th} day. For example, if the prediction for tomorrow is based on the past two days (today and yesterday closing prices), which corresponds to Up-Up, it is necessary to look at the three-day string in the table searching the Up-Up-Up and Up-Up-Down couples: the most recurrent between the two is chosen, and so the prediction for tomorrow will be a third up. After this selection, the results are tested on the validation set, adding also a random forecast: the system randomly generates an up or down forecast to see if the model outperforms the Random Walk Hypothesis. The second approach is a decision tree, which uses the previously analysed patterns (the up-down count in strings from one day to seven, and the count of up days in the same strings) plus the closing price of today, as Malliaris did. The results will be analysed in term of accuracy (the higher it is, the higher the closeness of the predicted values to the real ones) and precision (the closeness of the different prediction to each other, so it can be seen as a measurement of the result consistency).

Since a low performance can be expected because of overfitting (a method using too many parameters can become too complex, and so inefficient, generating low results in the validation set), two different tools are used in addition: logistic regressions and neural networks, of which a more detailed explanation is given in paragraph 3.3.3 and Appendixes B and C. Before applying them, an analysis on the feature importance in the prediction has been performed: the result is that too many variable lead to a poor predictive model, as it was already presumed. The Logistic Regression model is a regression analysis

used to explain the relationship between a dependent variable that is dichotomous (the price can be either an up or a down) and one or more independent variables. Neural networks, instead, are artificial intelligence systems that imitate human brain reasoning; they are structured with one or more input layers as starting point (where the known information are plugged in), and then, those layers communicate, thanks to weighted connections, with one or more hidden layers, which consist of several nodes: this is the network core, where the learning process is done. Finally, with other weighted connections, the hidden layers communicate with the output layer, the last part of the network that gives final result. This tool is defined self-learning because the more it is trained with training set, the more the network adjusts its connections in order to give better results.

A Neural Network model is an extension of the Logistic Regression that basically infers new, artificial features given the one provided to the model.

These two models have been tested several times, using all the features as well as only part of them, in order to check if some independent variables were not necessary.

The following step of the analysis is the application of the same forecasting methods to a dataset of the American Express Company stock price, to investigate the ability of these tools with a more fluctuating sample; a lower performance is expected, given the fact that a company is more subject to sudden changes with respect to an index.

Finally, the new tools, so the logistic regression and the neural networks, are applied to the same sample of the S&P500 Malliaris used, in order to see if they give better performances in predicting market movements.

Chapter 1

A review of the past financial techniques

In the first chapter, the problem of forecasting market movements will be analysed. It will be shown that there is not a defined and universal technique.

1.1 The past techniques

Since when the world of finance has developed, a lot of models were proposed in order to study the behaviour of prices; in facts, in literature, many examples can be found. Among all the papers and researches a dichotomy is persistent during the years.

On one side there are the well-known theory of Market Efficiency developed during the 70s, which will be explained in the following pages, and the Random Walk[1] behaviour of prices; on the other side there is the investigation on the effectiveness of technical analysis[2] and active management

[1] A Random Walk is expressed by the formula $Y_t = Y_{t-1} + u_t$, where Y_t represents the time series whose value is the past values plus the disturbance term u_t; in this process the expected value is 0 and the variance is t. In finance, it means that the better prevision for the price of tomorrow is the value of today. In other words markets work as if a casual law ruled them and there are not any trend or regularity.

[2] Technical analysis is a methodology to forecast the direction of prices through the studies of past market data like supply and demand, price and volume. In other words it uses market indicators to help assess whether an asset is trending, and if it is, the probability of its direction. Two assumptions are done: at any time prices are fair because they reflect all the relevant available information (so the

strategies[3]. Some papers, for example Fama and Blume (1966), Jensen and Benington (1970), Blume et al (1994), state that the Random Walk generates good trading but only in a little time horizon, others assert that transaction costs delete the advantage of trading prediction and finally some give particular meaning to technical indicators[4], the ones used in technical analysis, such as Moving Average (an indicator based on past prices that will be explained better in the following pages).

In the next paragraphs the principal financial techniques will be explained, taking as examples of those disciplines some works mentioned by Malliaris in his study.

1.1.1 The Efficient Market Hypothesis

The Efficient Market Hypothesis (since now EMH) was born in 1970s from the studies of Eugene Fama.

According to this hypothesis, prices instantaneously reflect all available information; to understand which is the information, a distinction among three different versions of the theory will be done. The arrival of new information is unpredictable and so the adjustment of the price will be as well. As a consequence

analysis looks at the security history rather than at external factors such as economic and news event) and market trend can be identified because there are no Random Walks.

[3] The Active Management Strategies are opposed to the Passive ones (according to which investors make an investment that does not involve any forecasting, but simply imitates the performance of an index). They use the human element to manage a portfolio relying on analytical researches, forecasts and personal judgement; the objective is to identify the securities that are mispriced in order to have a profit.

[4] Active traders (the ones who hold securities for little time period to take advantage of short-term movements in prices) use technical indicators to predict future price levels or the price direction, looking at past patterns.

the model that describes better this casual movement is the Random Walk.

Looking at Figure 1, which describes the cumulative abnormal returns of merging companies, the concept is clearer: in the day 0 there is a visible jump when the information about the merge becomes public; by the way after the announcement date of such merge there are no further movements, suggesting that the price contains the new information.

CAR of merging companies in the days of the Announcement date

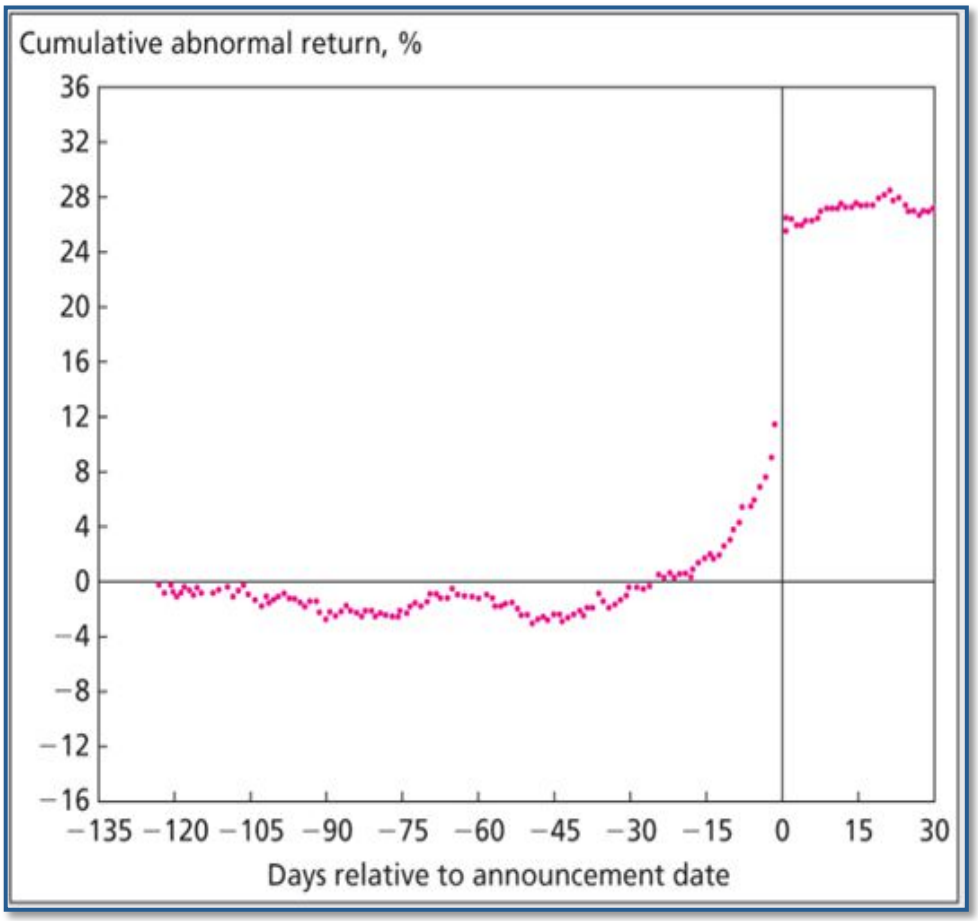


Figure 1: Cumulative abnormal returns surrounding takeover attempts. Source: Arthur Keown and John Pinkerton, "Merger Announcements and Insider Training Activity", Journal of Finance 36, September 1981

As said before, there are three versions of this hypothesis, the *weak-form EMH, the semistrong-form EMH* and *the strong-form EMH,* which differ according to what is meant for "all available information".

In the *weak-form EMH,* the information inside the price is the one deriving from the examination of market trading data, such as past prices and trading volume. In this version, trend analysis is unprofitable because past data are public and costless to obtain; if such data are reliable about the future performance of the security, all investors would have learn enough since to profit by signals. Definitely, the more the signals become known, the more they lose value and the price will adjust almost immediately.

The *semistrong-form EMH* asserts that all of the publicly traded information is available: in addition to past prices, also management quality, the composition of the balance sheet, earnings forecasts, accounting practices and data on firm's product line; in fact if all of this information is open to any investor in the public sources, there will be reflected also in prices.

The third version is the *strong-form EMH*: all of the relevant information for the firm is included in prices, also the one known just by insiders; this is of course the more extreme version.

The EMH involves that Technical Analysis is useless because past history is available at minimum costs and is already inside prices. Moreover Fundamental Analysis[5] adds little value

[5] Fundamental Analysis is used to evaluate a security by determining its intrinsic value exanimating microeconomic factors, such as financial conditions and company management, and macroeconomic factors like the overall economy and

because all the knowledge is public and, as said before, is inside prices; for those reasons it is not enough to find a good firm, but it is necessary to do a better estimate than anyone else. Finally, according to the supporter of EMH, Active Management Strategy is a wasted effort; it is better to use the Passive Investment Strategy that does not try to outwit the market, but simply builds a well-diversified portfolio without searching for overvalued or undervalued stocks. This approach is based on rejection of the buy and hold strategy[6]: it is useless to buy and sell frequently stocks, generating trading costs, since stock prices are at fair levels.

As a consequence of the choice of a Passive Investment Strategy, it is necessary to underline the importance of portfolio selection, where the basic principle is diversification: even if stock prices are fair, each company is still exposed to the firm-specific risk[7], moreover, though the market is efficient, each investor should determine his personal risk-return profile and finally also taxes have to be taken into account.

Professional portfolio managers are not enthusiastic about this hypothesis and they doubt on the real efficiency of markets, underlying their importance in investment decisions: three issues, which imply that the debate between the supporters and non-supporters of the theory will never be solved, can be raised. The first one is the magnitude issue, which derives from the fact that stock prices are very near to the fair value and so only managers of huge portfolios can take advantage of

industrial condition. The goal is to produce a value that can be compared to the current price, to understand if the stock is overvalued or undervalued.

[6] The buy and hold strategy is a passive one, in which the investor keeps the security for a long period of time, being not concerned about short-term movements.

[7] Firm-specific risks correspond to those that are very specific to the company.

little mispricing, namely managers allow a constant evolution of prices to their fair value. The second issue is the one of the selection bias: if a portfolio manager finds a technique that allows huge money return, probably he will take that theory secret because if it becomes public it will be self-destructive; for this reason it is very hard to evaluate the true ability of portfolio managers to generate abnormal returns. The last issue is the one of the lucky event: starting from the assumptions that all stocks have fair prices that contain all of available information, on a large number of investors, who use different investing schemes, there will be for sure someone who will earn more money than the others, so for every big winner there would be many big losers.

To summarize, the EMH states that prices are based on fundamental data and contain all of the information, for those reasons markets are efficient; from this vision emerges the impossibility of outperforming the market in the long period. Inasmuch bubbles[8] prove the inefficiency of markets: as it is explained in the following example, EMH seems to be not always true. It is based on the principle of *Homo Oeconomicus*[9], according to which people are rational, but recent studies proved that investors do not decide only on rational basis but also on emotional ones.

An example of bubble is the dot-com[10] one: it is a stock market bubble that happened between the 1995 and the 2002,

[8] A bubble is an economic cycle in which prices grow rapidly before a great drop, due to the fact that no more investors intend to buy at high prices.

[9] The term Homo Oeconomicus was used for the first time by the philosopher and political economist John Stuart Mill and describes the rational man, interested just in his own affairs.

[10] The Dot-com bubble is the speculative bubble exploded between the late 90s and the 2000: in this period a lot of small companies active in the Internet sector, which was overestimated, born. Moreover the speculation on stocks and the

culminating in the year 2000. In the whole period the prices of technology companies continue to rise, together with the NASDAQ, a well-known American stock exchange, and the Dow Jones Internet Price index. On the 13th of March 2000 a huge sell of dot-com orders let the bubble trigger, leading to a fall of prices: also the NASDAQ (which was standing at 5,132.52 points on the 10th of March 2000) lost 500 points in just three days. This exaggerated rise in shares value is no rational in economic sense, and it is not contained inside the fundamental data that, according to EMH, determines the price; many traders, especially the noise-ones (those who lack in special knowledge or information about trading) invested in the dot-com, not because they thought those shares were undervalued, but because a bullish market[11] were created and they expected a further increase.

To conclude, this example shows that in 2000, at the culmination of the dot-coms, prices were heavily overvalued and after the downfall they were even undervalued: if EMH was always correct, both states could not happen for so long periods.

1.1.2 The Adaptive Market Hypothesis

There are a lot of studies that investigate the effectiveness of technical analysis and active management strategies, such as Fama and Blume in 1996, Jensen and Benington in 1970 and

several venture capitals lead investors to overlook the traditional price evaluation, such as price-to earnings ratio (P/E). All those factors drove to the burst of the bubble, where many companies failed.

[11] A bull market is a market characterized by a great optimism and confidence, in which prices are rising or are expecting to rise.

many others; an example in this category is the *Adaptive Market Hypothesis* (since now AMH) by Andrew Lo (2004).

This hypothesis attempts to joint the EMH to behavioural economics, a social science that includes in economic decisions all the emotional, psychological and social factors an investor has.

According to Lo, investors' behaviours like overconfidence, loss aversion and overreaction [12] are consistent with evolutionary models of human conduct like competition, natural selection and adaptation. All those acts have a consequence on financial and investing decisions and so humans make guesses based on trial and error. It is clear that, under AMH, prices reflect both information (according to EMH) and the environmental conditions that impact on people. With this perspective market efficiency is higher when many species (groups of market participants, e.g. market makers, retail investors and so on) are competing for scarce resources in a single market, and viceversa, the market will be less efficient if a small number of species are competing for abundant resources; the rule that can be derived is: the more the competition, the less the resources. According to Lo, a highly traded financial product, like the treasury bonds, gives an example of this theory, having all, or almost all, relevant information reflected quickly inside the price.

From these theory market efficiency is not a constant condition, but it depends on the context, on the kind of market and again on behavioural biases. The competition of financial

[12] Overconfidence is the overestimation of personal skills; loss aversion is the tendency of people to prefer to avoid a loss than acquire a gain; overreaction is an exaggerated reaction to new information, so the price of a security will change suddenly and will not reflect the real value, this swing doesn't last for long time. All those concepts are notions of behavioural finance.

markets lead to the “survival of the strongest”, as Darwin teaches; in fact a not good trader sooner or later will go out from the market because of the suffered losses. For this reason with this theory it is harder to predict the next movement.

1.1.3 The Moving Average Hypothesis

In 2007 Schulmeister used Moving Averages, as trend indicator, to try to predict S&P 500 futures returns.

The most popular is the Simple Moving Average (SMA), computed as the arithmetic mean of closing prices (the last price of a trading day) over a defined number of days; any day the new mean is calculated again, cutting off the oldest price and adding the new one.

The Simple Moving Average is very popular, but also very criticised: many traders argue that the most recent data is more significant than the older ones, for this reason it should has a greater weight. One of the responses to this critic is the Exponential Moving Average (EMA), also called Exponentially Weighted Moving Average, in which latter prices have more weight, and so it reacts faster to recent price changes.

Figure 2 gives an example of SMA, where the S&P500 (the blue line) is shown together with its 50-day moving average (the red line); the collected data goes from June 2012 to June 2013.

The S&P500 and its moving average

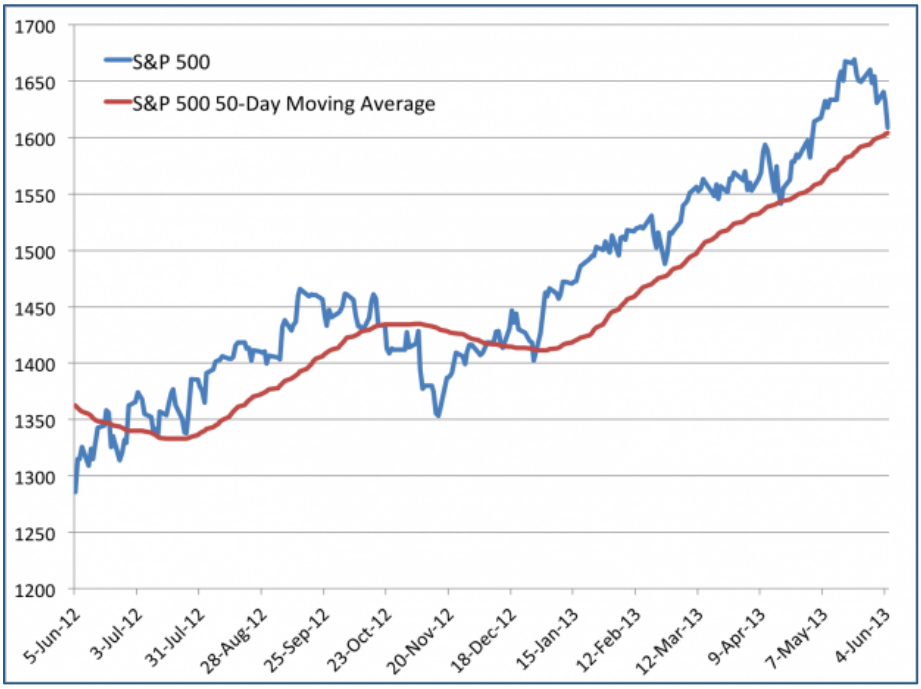


Figure 2: Example of a 50-day moving average computed on the S&P500, source: Business Insider/Matthew Boesler, data from Bloomberg

Since, looking at a stock, a short-term uptrend can be just a bounce in a long or mid-term downtrend, there will be indecision about the real direction of price: the moving average smooth the noise of price waves, pointing out price direction. In fact MA has the same direction of price, but when the trend is up the Moving Average is below the price, as can be seen from the graph in Figure 2 in the period from July 2012 to October 2012, while when the trend is down the Moving Average lies above it (see the period from October 2012 to December 2012, in Figure 2). Another relevant observation to do about Moving Averages is that they cuts the price when the trend changes direction: in simple words when the Moving Average cuts the price from below to above, the trend is going from negative to positive (December 2012, in Figure 2) and so it is time to buy

(this movement is called *golden cross*). Viceversa, when the Moving Average cuts the price from above to below the trend is passing from positive to negative (end of September 2012, in Figure 2) and, therefore, it is time to sell (this movement is called *devil cross*).

Coming back to Schulmeister's work, he analysed the profitability of several moving average models in the S&P spot market and in stock index futures market. He uses the daily stock data to generate a buy-sell decision, employing a short run (from 1 to a maximum of 12 days) and a long run (from 6 to a maximum of 40 days) moving average, with the restriction that the length of the short run moving average differs from the one of the long run by at least 5 days, testing in total 2,124 models. The general rule used is to go long (so to buy) when the short term moving average crosses from below the long run one, and to go short (to sell) when the short run moving average crosses from above the long run one.

This strategy was profitable for the data from 1960s to 1970s, while the same indicators used in the data from 2000 to 2006 have not lead to good results.

Schulmeister tries to explain this difference with the further increase in the speed of trading: Internet and computers financial software facilitate the use of intraday data; on one hand intraday price movements are more persistent, on the other hand price changes of daily data are more inconstant.

1.1.4 The Econometrics and Macroeconomic Factor approaches

Another way to investigate the behaviour of prices is to develop econometric return-forecasting models, like the one created by Doran et Al. in 2009.

They analysed the S&P500 index, using as benchmark to track volatility[13] the VIX[14]. They found out that short term expected returns had higher volatility, which means a higher risk. Moreover they investigated the market risk premium, which will be explained in the following page, and according to the study it went negative on two occasions: in 1987 (before the Black Monday) and in 2000 (during the tech bubble). It is commonly known that higher volatility means higher risk, and in fact many examples can be done about this definition: an investment on treasury bonds will have low volatility because is an almost sure security and its trend is not subject to sudden changes, on the other hand a stock of a private company will have a more fluctuating trend: the complexity of the business management, political decisions that can influence the holding, a parent corporation that owns enough voting stocks to control its management and policies, the discover of new technologies and the evolution of competitors are only some of the influencing factors. The investment on stocks of private companies is riskier than the previous one. For what concerns

[13] In finance volatility is used as a statistical measure to quantify the dispersion of returns of a security or of an index. In other words it quantifies the risk or uncertainty about the size of changes in the value of the security: the higher the volatility, the higher the potential change, in both directions. It is identified with the standard deviation (σ) of the returns of the security or of the market index.

[14] The VIX is the CBOE (Chicago Board Option Exchange) Volatility Index, introduced in the 1993, which shows the 30-day market's expectations of stock market volatility.

the market risk premium, it corresponds to the difference between the expected return of the market and the free risk rate. For the first one it can be used a financial index (for example in America it is used the financial stock market index S&P500), for the second one usually it is taken the Treasury bond rate of return. The historical market risk premium[15] is the one used in this study and so it is equal for every investor, inasmuch as it is based on what happened.

According to the authors, the market risk premium probably went negative because of irrational exuberance, a term used to express an irrational enthusiasm of investors that leads prices to levels that are not supported by fundamentals, which are the information needed to analyse the health and stability of the asset.

On the other side, Avramov and Chordia, in 2005, use the firm macroeconomic variables[16] to investigate the company level predictability return. For example, in a stock market index the important factors for returns are treasury yields and dividend yields; by the way this study works better for small companies and growth stocks. In simple words the authors modelled the company's equity excess return[17] as a function of two parameters, alpha and beta, where alpha varies depending on lagged macroeconomic variables and beta varies on a set of portfolio risk factors. These portfolios risk factors vary

[15] In addition to the historical market risk premium there are also the required and expected risk premium, which change from investor to investor, for example according to the individual risk aversion.

[16] The macroeconomic variables are general economic conditions that affect securities such as business cycle, inflation rate, interest rates and so on. They are difficult to value and predict, and for this reason they are a source of risk.

[17] The excess return is the difference between the return of the portfolio and the risk free rate, or a defined benchmark rate. If the excess return is positive the investment outperforms the risk free rate or the benchmark rate, viceversa if it is negative.

depending on the macroeconomic variables (for example dividend yields, default spread, that is the probability of going bankruptcy, and treasury bill yields) in order to have a time varying equity risk premium. Thanks to the estimation of means, variances and covariances they construct each month two portfolios: a mean-variance portfolio (with no short selling, which is the sale of a borrowed security) and the other one with two parameters alpha and beta and an equity risk premium, all not allowed to vary with the macroeconomic variables. The result is that it has been found a superior performance for the investment strategies that take into account business cycle variables; moreover after this discovery it can be proved that strategies conditioned only on predictable equity premiums are not so reliable. To conclude, firm specific factors, such as treasury and dividend yields, are more useful in predicting returns, with respect to equity risk premium. Unfortunately this theory can be applied only on small companies and so its action field is restricted.

1.1.5 The Positive Feedback Strategy

In 1990 De Long investigated a simple theory: the positive feedback strategy. According to this, the flock mentality[18] of noisy traders (those who trade without using fundamental data) leads them to buy when prices rise, and sell when prices fall, with respect to the previous closing prices. For this reason it can be profitable for a smart investor, the one with the

[18] Flock mentality is a kind of mentality characterized by the lack of individual decision-making, bringing people to act in the same way of the majority of people.

knowledge of fundamental data, to buy ahead the flock and to sell before it.

Rational speculators are responsible for the activation of the positive feedback, because when they receive a signal that noisy traders will buy a certain security tomorrow, they buy those securities today increasing their prices. The next day noisy traders will buy those securities because of the increase of the price; as a consequence prices will increase even further of what was expected by rational speculators and there will be a destabilization due to the movement of prices themselves from the fundamental. When rational investors will buck the trend, prices will again come closer to fundamentals.

This behaviour of noisy traders is also the reason why market declines lead to other declines, and vice versa an increase leads to more increases. Moreover a too long positive feedback strategy, used by noisy traders, can bring to a market crash or to a financial bubble, inasmuch prices rise heavily without a real reason, exceeding valuations. This is a proof that a thoughtless use of this technique destabilizes financial markets.

1.2 Conclusions

To conclude this review of past studies and theories about the forecast of market movements, it is easily notable that there are a lot of researches about stock index patterns.

To summarize, the first theory that has been analysed is the Efficient Market Hypothesis, according to which prices are always fair; the second one is the Adaptive Market Hypothesis, which joins the EMH with behavioural economics, and it states

that investing decisions are based also on trial and error. According to the theories explained in the previous paragraphs, the Econometrics approach uses a benchmark to track volatility while the Macroeconomic approach uses macroeconomic factors, but it works better only with small companies. Finally it has been shown that the Positive Feedback strategy can lead to a market bubble, as well as the EMH: this happens if prices rise consistently in little time.

The purpose now is to determine a decision-making strategy in order to build conditional forecasts; the novel approach used by Malliaris will be explained in the following chapters, and then also a practical application of the theory will be shown.

Chapter 2

Malliaris' technique

In this chapter the approach used by Malliaris in investigating the patterns of the S&P500 index will be explained in full details.

2.1 The novel approach

A. G. Malliaris is a professor of Economics and of Finance at Loyola University of Chicago.
The paper, which will be analysed in the following pages, is a research of year 2013, done with the collaboration of Mary Malliaris.

Malliaris collected the daily movement patterns of the Standard & Poor 500 index[19] for 61 years (from 3/1/1950 to 19/7/2011) in order to find a suitable methodology to set up conditional forecasts. For the second part of the study, the forecasting one, the 15,488 daily observations are split into two groups: the training set, used to determine the most dominant patterns, and the validation set, which is used to test the previously determined dominant patterns.

[19] S&P 500 is the American index made of the 500 higher capitalized American companies.

The first step is to classify the closing prices, which are the final prices at which securities are traded in a trading day, as up movements (higher than the previous closing price) or down movements (lower than the previous closing price). After the classification, all the combinations of closing prices, from two days up to six, are counted and ranked. Specifically, in a two-day string there could be four different combinations of closing prices: Up-Up, Up-Down, Down-Up and Down-Down; the graphs display all the possible outcomes, underlying their recurrence over time. This investigation is done decade by decade and with patterns of several days (strings from 1 to 6 days), in order to find the most recurrent trend.

After this classification, a simpler one is done, counting just the number of up days in the same strings. Taking the previous example, in a two-day string there will be three possible outcomes: zero, one or two ups. Also these results are displayed in the graphs decade per decade, resulting less crowded. This second method of ranking helps to understand the number of positive days in a period, without focusing on the possible combination.

It will be shown in the next pages that some string patterns continue across time with almost the same percentage, and so, in some sense, the "buy and hold strategy"[20] works.

In the second part, the sample has been divided into Training and Validation sets: the first one is used to determine the dominant patterns, while the second is used to determine the direction (up or down) of forecasts, obtained with the dominant patterns previously identified.

[20] The buy and hold strategy is a passive strategy in which the investor buys securities and stocks and keeps them for a long period of time, despite market fluctuations.

The forecasts direction is identified in two ways: by applying the previously determined strings to the validation set and by using a decision tree[21] (a more detailed explication is given in Appendix B).

To better explain, the training set is used to count how many times each one-to-seven-day pattern occurred, ranking all the results into a table, which will be displayed later. For each pair of patterns, which differs just in the last movement, the most recurrent is chosen: for example, to forecast the movement for the forth day knowing the behaviour of the past three, e.g. Up-Up-Down, it is necessary to look at the four-day strings in the Training set and to choose the dominant one between Up-Up-Down-Up and Up-Up-Down-Down.

This method is the basis for the prediction in the Validation set, in which a series of seven forecasts is made. For every pattern of days T-6 to T, a prediction for T+1 was generated, relying on the occurrence in the Training set. To be clear, in the forecast for the first day in the Validation set, an up movement will always be predicted, since in the Training set up dominates down (the result comes from Table 1, which will be displayed and explained in details in the following pages). In the two day predictions, there are two cases to consider: those beginning with down and those with up. The string beginning with up can end with an up or a down, so it is necessary to look at the recurrence in the Training set; since Up-Up is the dominant pattern (4,609 times versus 3,470 of the Up-Down outcome), with a single input of up in the Validation set, another up will be

[21] A decision tree is a tree-shaped diagram, useful in determining the course and consequence of an action, or in showing a statistical probability. Each branch is a possible decision, reaction or outcome: the tree shows how one choice lead to the next one.

forecasted. The same has to be done with the day beginning with a down, specifically, according to the Training set, another down day is more probable to happen, and so, this is the outcome forecasted by the Validation set. To those outcomes, a random one is added, in order to compare to it the performance of the n-tuple forecasts. Finally, all the results are compared to the real ones, in the Validation set, to check the performance of the model.

As said before, the forecasts have been done also with the help of a decision tree methodology. This tool, build with data up to December 2009, checks every input variable at every step of the process, choosing the one with the better performance, thanks to the use of the C5.0 algorithm; the result is a tree that classifies the input data row as likely to have a specific target value, using three inputs: the Up-Down patterns from one to seven days, the number of up days in 1-5 days and the closing value of the current day.

The conclusion is that the forecast is better with either one or six day of information, in particular the first methodology (application of past pattern strings to future data) is successful over 58% of the time in forecasting an up day and about 49% of the time in forecasting a down day; the decision tree performs better, with a correct forecast for a down day of the 51% of the time, and for an up day of the 61% of the time. This results, not so performing, is probably due to the problem of the overfitting: the task of machine learning[22], the discipline

[22] Machine learning is a subfield of computer science, with the aim of studying and constructing algorithms, which can learn from and make predictions on data.

used to complete this project, is to fit in a proper way a model to a set of data; overfitting happens when the model is too complex (e.g. too much parameters compared to the number of observations) and describes random errors or noise, instead of the panel data. The consequence of overfitting is poor predictive performance of the model, because it overreacts to minor fluctuations in the training data; a solution to this problem could be to use more data in the validation set, or to make the model less complex and so with less parameters.

2.1.1 Multi-decade patterns

In his work, Malliaris proposed two ranking methods for the classification of up and down in the patterns from one day to seven; in the first section of the study, he looked at individual string patterns, starting from one day, up to six. To better explain, in the three-day patterns there will be eight possible outcomes (UUU, DDD, UDU, UUD, UDD, DUD, DUU, DDU) associated with three consecutive days of the week (Monday-Tuesday-Wednesday, Tuesday-Wednesday-Thursday, Wednesday-Thursday-Friday, Thursday-Friday-Monday and so on).

In the second section, just the number of up movements in the pattern of interest will be counted, so, in the two-day string, there could be 0,1 or 2 ups, specifically the strings UD and DU, having just one up day inside, will be collected in the same category: with this approach it is possible to focus just on the number of up days.

The following figures show the first approach, from one-day to four-day string patterns; the five-day and six-day are not displayed, as their graphs are too crowded to show. On the x-axis the decades are indicated, from 1950s to 2010s, except for the last one, in which there are only two years, from 2010 to 2011, while on the y-axis the percentage of positive and negative movements is displayed.

One-day string pattern

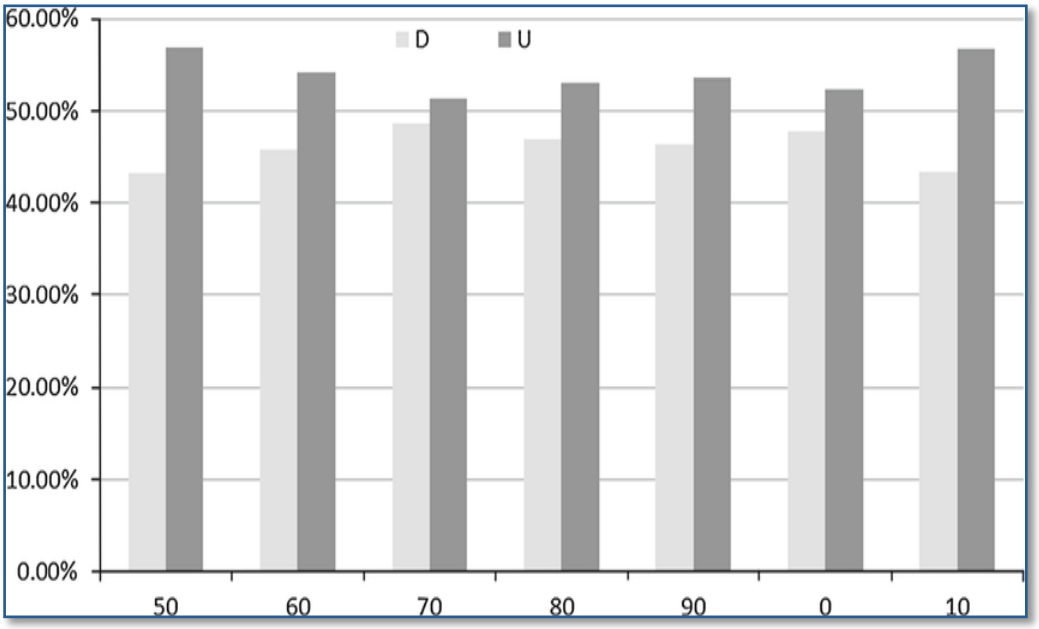


Figure 3: Up (U) and down (D) percentage movements per decade. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Looking at the graph in Figure 3, it is possible to reject the Random Walk hypothesis, according to which prices move without any rule. In the long term, the index is moving up and in every decade the up days are more than the down ones. Only in one decade, the 70s, positive and negative days are almost

close. Except for the years 70s and 00s, the difference between up and down is statistically significant, with a mean of about 15%.

Two-day string pattern

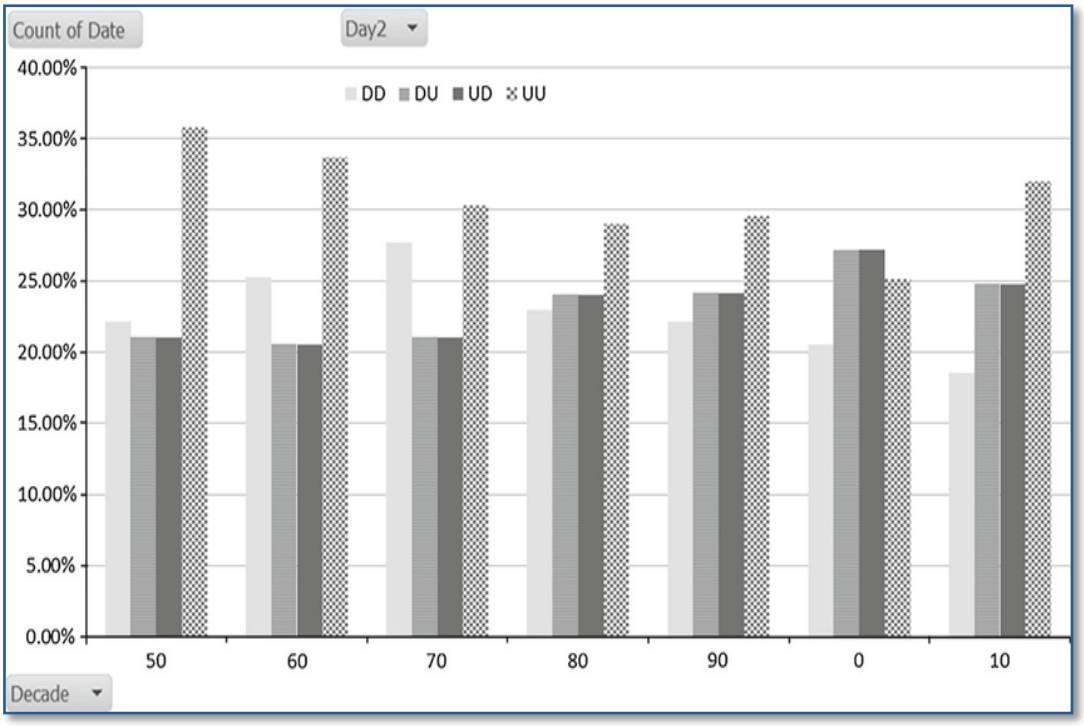


Figure 4: two-day strings of up and down movement. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Figure 4 displays the two-day strings, in which there are four possible outcomes. The most dominant until the decade of the 90s is the Up-Up trend, which indicates that the S&P500 has been positive for two days consequently the majority of time. Just in the decade of 2000-2009 the dominating pattern are Up-Down and Down-Up, a clear sign that in this period the index suffers of uncertainty, probably due to the dot-com bubble and the financial crisis of 2007-09[23]. In the last decade, even if just two years are

[23]The subprime mortgage crisis of December 2007-June 2009 was triggered by a huge decline in home prices. In the early 2000s there was an increase in the

counted, the trend seems to be positive again, with the Up-Up movement dominating the other patters.

Three-day string pattern

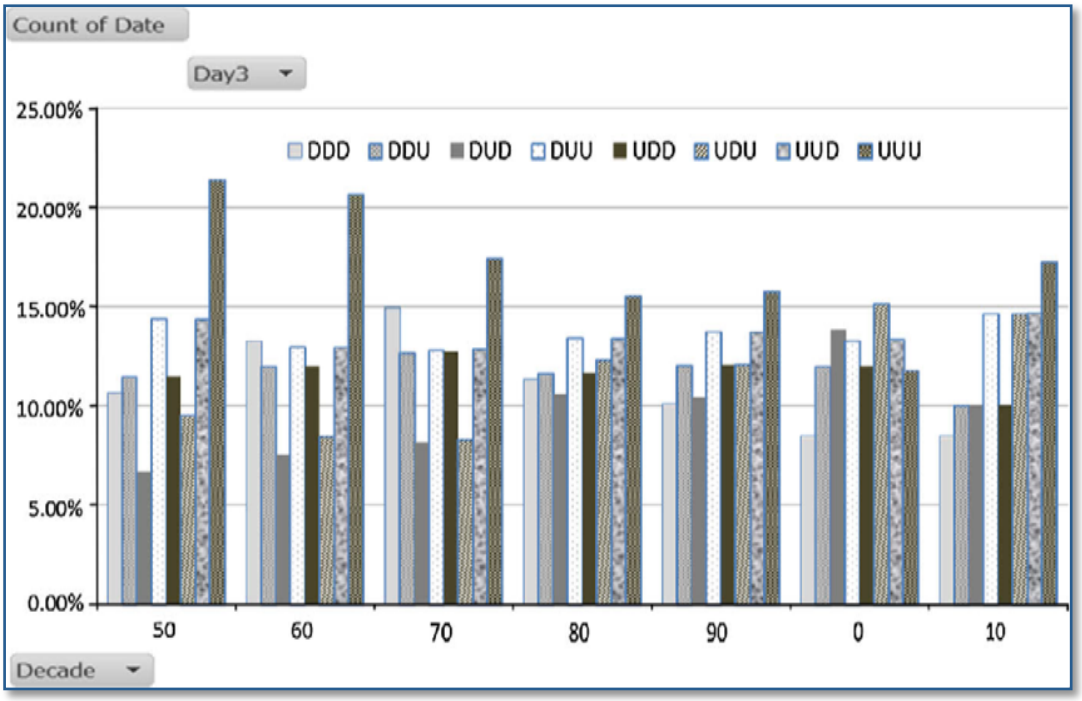


Figure 5: the three-day strings of up and down movement. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Analysing the three-day strings (Figure 5) it is possible to see that the Up-Up-Up pattern has a great dominance for the first decades, decreasing a little until the collapse in the decade of 2000-2009, as seen before in the two-day strings.

Afresh, there is an increase in the last decade, which can be red as an improvement and stabilization in the economy, which is reflected in the index. It is interesting to underline the stability of

issuance of mortgage-backed Securities (MBSs), because of the use of securitization (the pooling of debt and then the issue of assets based upon that debt). The pooling of different mortgages may had reduced risk, so these assets should have been quite safe, but in reality the majority of the mortgages being securitized were of poor quality (sub-prime). The expansion of household debt with low credit quality led to massive defaults (e.g. Lehman Brothers' failure in 2008).

Down-Down-Up and Up-Down-Down strings over decades: these two patterns are almost at the same level in every period of observation, as well as the Down-Up-Up and Up-Up-Down strings, while three consecutive days of down are increasing until the 70s, to decrease again.

Four-day string pattern

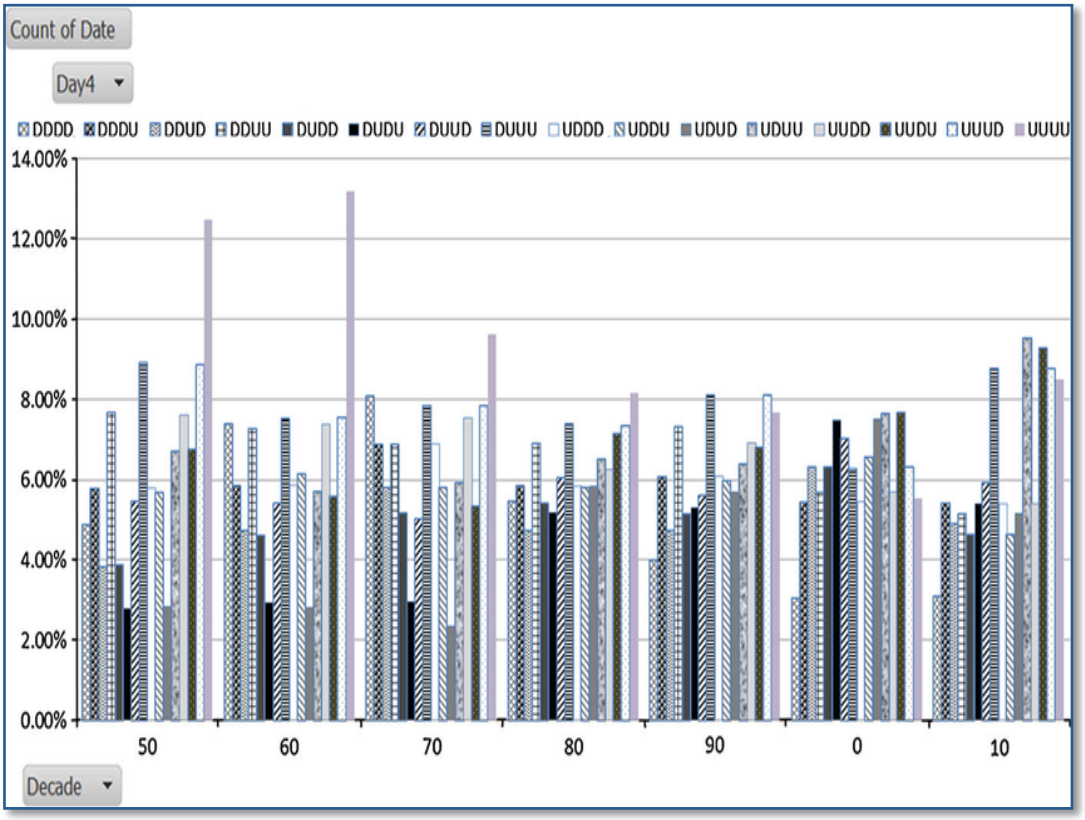


Figure 6: Four-day strings of up and down movement. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Figure 6 represents the four-day strings of movements, so there are sixteen possible outcomes to look at. The first notable thing is the predominance, in the earlier decades, of the UUUU column: it is the highest in the beginning of the investigated period, then it decreases and from the 90s it is not anymore the prevailing

column. For what concerns the most negative pillar, that is DDDD, it increases until the 70s, to decrease during the following decades: from the 90s on, it is the smallest column. For what concerns the UDUU and UUDU columns from the 50s to the 60s and also from 2000 to 2011 they are quite close; just from the 70s to the 90s there is a small difference in values of about 1%, however not a statistically significant data for the index. Looking at DUDU and UDUD columns from the 50s to the 70s they are almost close and with a little percentage of occurrence, instead from the 80s to 2010s they are close again, but they repeats almost double with respects to the other decades. This high occurrence is due to the improvement of daily changes in the index value and the speeding of trade in a single day. At last the Up-Up-Up-Down column, the penultimate on the right side, so the one with three consecutive days of positive closures, it is not so predominant with respect to the column of four positive days until the 70s, but then they are almost close in the 80s, and from the 90s until today the three-ups column predominate over the four-ups one.

For the remaining strings, that are the five and six-day patterns, according to Malliaris, the charts are “crowded” and so they won’t be displayed.

To observe the data from another way, the number of up movements will be counted, in the same strings used before.

To better explain, in the two-day strings there were four possible combinations, instead, looking at the frequency of up movements, there will be just three outcomes (zero, one or two up because UD and DU converge in the “one-up family”).

The following plots will demonstrate the different ranking method and the consequent results. Also in this case, in the y-axis there are the occurrence percentages during the different decades, obviously displayed in the x-axis.

Two-day percentages

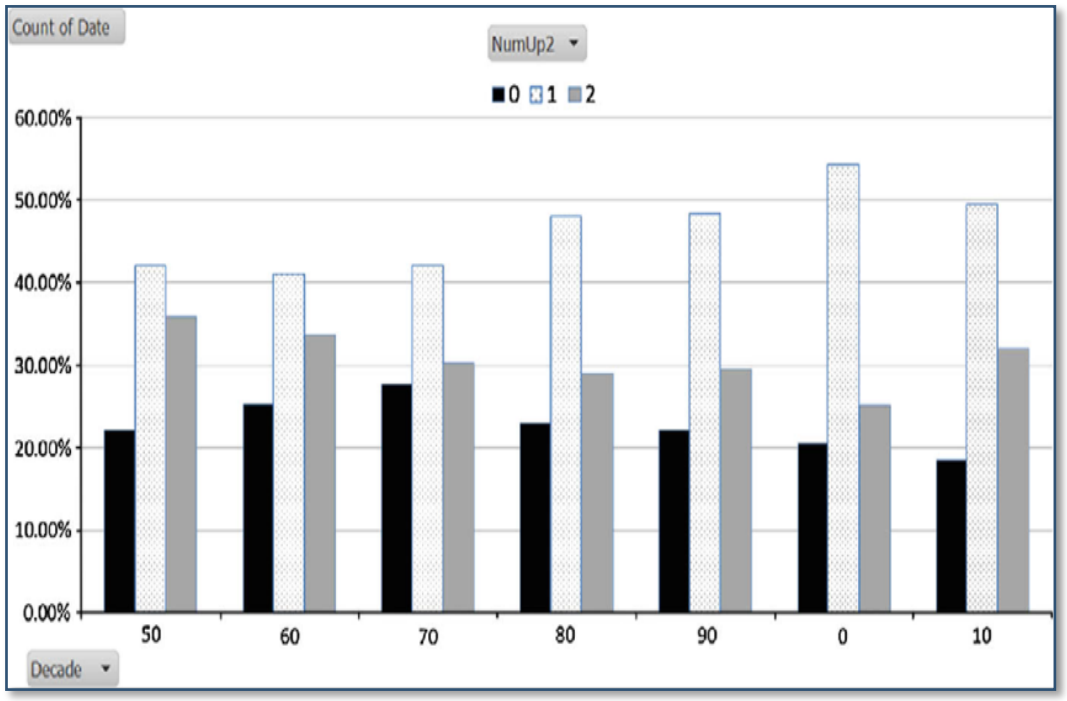


Figure 7: Percentage of 0,1 and 2 up movements in two days. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

At first glance, looking at Figure 7 it is noticeable that the single up in a total of two days is the most frequent outcome in every decade. The column with two consequent up days is anyway more frequent than the zero up column, which is the black one, even if it can be seen that there is a slow decline until the decade of 2000, and then there is again an upward trend in the last decade, which represents the data of about two years, and not ten as all the other past decades.

A second observation can be done thanks to the always greater detachment of the one-day up column from the others: in the earlier decades it is pretty close to the two-up column, with the passing of time it becomes double higher with respect to the others. The great predominance of this column in the decade of the crisis underlines the uncertainty of that period.

Three-day percentages

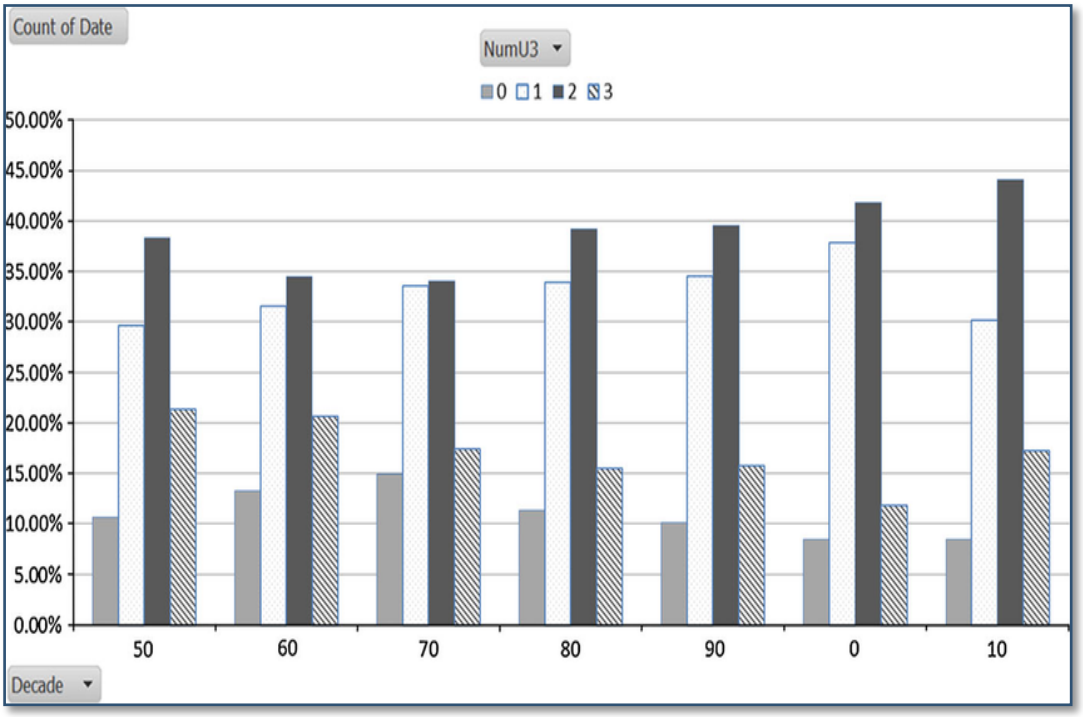


Figure 8: Percentage of 0,1,2 and 3 up movements in 3 days. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Figure 8 displays the result for the three-day strings. The dominant column is, in every decade, the one with two consecutive up days; on the other hand the least is the column with zero up days. There is a great occurrence also for the column of one positive day on the total three, while for what concerns to the column with three up days is not so predominant

and, moreover, the more it approaches to the present the more it declines, with exception for the last column, as it happens in the previous figure.

Four-day percentages

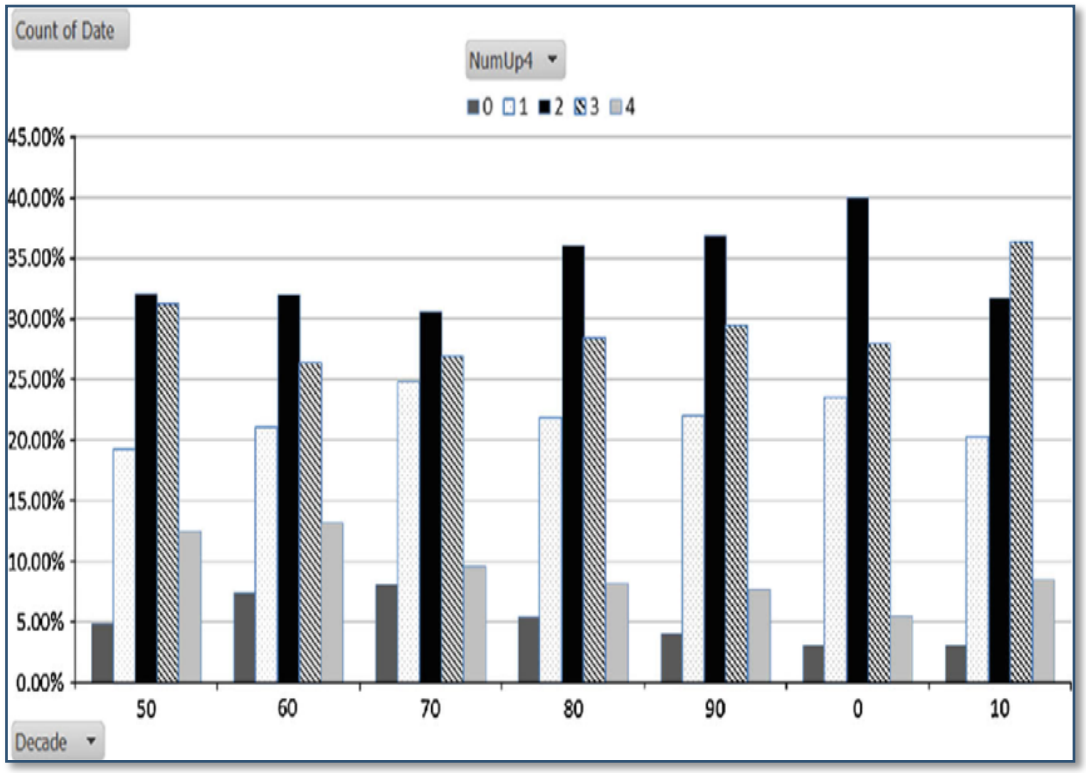


Figure 9: Percentage of 0,1,2,3 and 4 up movements in 4 days. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

In Figure 9 the four-day percentages are displayed: there is a clear predominance of the column with two up days out of the total four, except for the last decade, where the predominance is given by the three up days out of four. The more negative column, that is the one on the left side, with zero positive days over four, is the least recurring. The four consecutive days of up movement, the column on the right side, is not so frequent,

mostly in the last decades, where it decreases more and more, to have a little increase in the last decade.

Five-day percentages

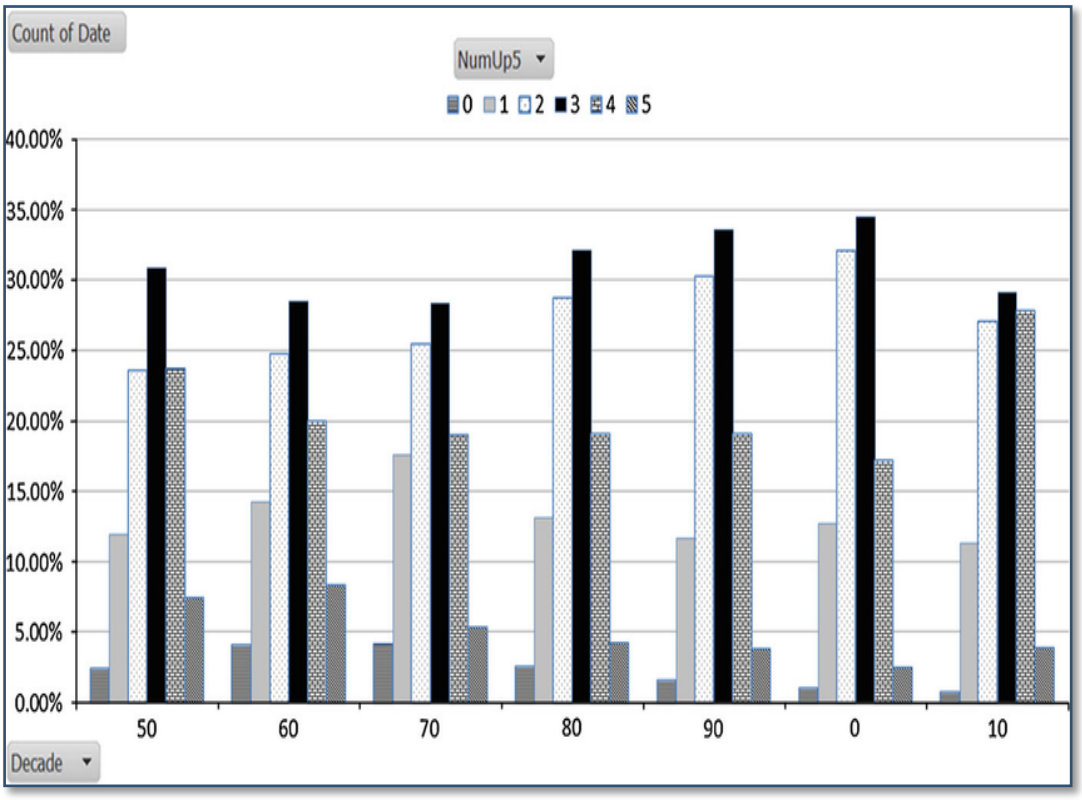


Figure 10: Percentage of 0,1,2,3,4 and 5 up movements in 5 days strings. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Figure 10 is the last chart: the prevailing bar is the black one, which is that of three consecutive up days. Right after, there is the column with two up days and then the one with four up days: these last are at the same level just in the first and the last decades. The column with five consecutive days of up movements is not so frequent, but it is in any case bigger than the one with zero positive days, which is the smallest pillar in every decade. The two-up days out of five-column increases a

little every decade, to have a peak in the second last decennium, while the four-up days column decreases a little decade by decade, to grow again in the last one.

After those examinations of up and down movements of the S&P500, it is interesting to compare the two approaches used to rank the patterns (the count of the different combination of up and down in the strings and the count of the number of up in the same strings, both from one to six days).

Looking at Figure 4 (the two-day string pattern), the Up-Up pattern is the dominant one, except for the period from 2000 to 2009 in which the Down-Up and Up-Down columns are prevailing, at the same level; instead in the plot of two-day up percentages, in which are counted just the number of up days (Figure 7), the dominant column is the one with one up day out of two. This fact is obviously due to the “merger” of the columns with one up day (Down-Up and Up-Down).

The three-day string pattern, plotted in Figure 5, shows a preponderance of the three up-day column, while the graph with the percentage of up days (Figure 8) shows a different result: the prevailing column is the one with just two up days out of three, followed by the column with one up day; the explanation for this difference is the same as before.

Finally the comparison between Figure 6 and Figure 9 shows that in the first one (the four-day string pattern) the four-up day column dominate until the 90s, to leave the step to the three-up day column until the last decade; in the second one (the four-up day percentages), until the decade from 2000 to 2009 prevails the two up-day column, and then the three up-day column predominates.

Those results are due to the fact that, with the second method of classification of data (the count of the number of up days in the strings from one day to seven), the up days are just counted in the string, no matter where they are positioned in the day of interest. Comparing the three-day and the four-day up percentages (Figure 8 and Figure 9) the prevailing column is always the one with two up days. Another interesting fact to notice about the second ranking method, is the behaviour of the external columns, namely the column with zero up days and the column with the maximum number of up days in its relative string; the higher the number of days, the lower the height of those external columns, in fact the strings seem to follow a normal distribution[24], with respect to the first ranking method, with few values in the tails and a higher concentration in the middle. Finally, the first ranking method gives a more optimistic point of view on the movement of the S&P500, in fact it seems that the more positive column (the one with only up days) for every string is the prevailing one, in almost all the decades. This result can be confusing because, in reality, the index is more probable to have daily changes, as it is notable from the second ranking method, as explained previously.

As highlighted before, some results occur more across decades, now it is necessary to understand if those similarities over time can be used to develop forecasts for the following day between up and down.

[24] The normal distribution is a continuous probability distribution, which plot all its values in a symmetrical fashion, with the majority of data clustering around the mean, and the rest weakening toward the extremes in a symmetrical way. The benchmarks of the distribution are mean (μ), equal to zero, and standard deviation (σ), equal to one.

2.1.2 Forecasting

The objective of Malliaris' work is to translate the analysis into a decision-making strategy, in order to predict up and down movements conditioned on the past patterns previously identified, from one to seven days. The method will be explained in the following pages, but first of all it is necessary to make an introduction, since data have been split into two groups:

- Training set: from January 1950 to December 2009, for a total of 15,087 observations,
- Validation set: from January 2010 to July 2011, for a total of 387 observations.

The training set is used to determine the most dominant pattern among all the strings; these dominant patterns are used in the validation set, to determine, for each day, the direction of the forecast, or rather to establish if it is more probable to have an up or a down day, as follows.

First of all, for every single pattern, the number of occurrence during the period of investigation has been counted. Table 1, in the following page, gives a sample: the different outcomes of the training set have been counted and collected in a table for every string of days (from one to seven), no matter for the decade of occurrence. Afterwards, every couple that differs just in the last movement is analysed, and the dominant outcome (the most recurrent between up and down in the last day) within every pair of patterns is the selected one: this is the basis for the validation set.

For example if just one day is known, and the aim is to predict the following day, it is necessary to look at the two-day strings in

Table 1; the pattern can start with an up or a down movement. In the first case the outcome can be an Up-Up string or an Up-Down one: according to Table 1, UU happened 4,609 times, while UD happened 3,470 times so the "winning" pattern is UU. Similarly, when the string started with D, it ended with a positive day 3,469 times and with another negative day 3,539 times, so it is more probable, as outcome, a Down-Up string. Summarizing, in the case of two-day string, it if begins with an up day the forecast will be another up day, while if it starts with a down day, the forecast will be an up day.

Count of patterns

Pattern	Count	Pattern	Count	Pattern	Count
D	7,009	DDDDDD	185	DDDDDDD	83
U	8,078	DDDDDU	215	DDDDDDU	102
DD	3,539	DDDDUD	179	DDDDDUD	91
DU	3,469	DDDDUU	247	DDDDDUU	124
UD	3,470	DDDUDD	220	DDDDUDD	104
UU	4,609	DDDUDU	168	DDDDUDU	75
DDD	1,730	DDDUUD	233	DDDDUUD	116
DDU	1,809	DDDUUU	283	DDDDUUU	131
DUD	1,441	DDUDDD	220	DDDUDDD	111
DUU	2,029	DDUDDU	209	DDDUDDU	109
UDD	1,809	DDUDUD	127	DDDUDUD	61
UDU	1,660	DDUDUU	204	DDDUDUU	107
UUD	2,029	DDUUDD	254	DDDUUDD	133
UUU	2,580	DDUUDU	203	DDDUUDU	100
DDDD	826	DDUUUD	259	DDDUUUD	124
DDDU	904	DDUUUU	334	DDDUUUU	159
DDUD	760	DUDDDD	178	DDUDDDD	103
DDUU	1,050	DUDDDU	197	DDUDDDU	117
DUDD	769	DUDDUD	158	DDUDDUD	75
DUDU	671	DUDDUU	237	DDUDDUU	134
DUUD	871	DUDUDD	133	DDUDUDD	66
DUUU	1,158	DUDUDU	140	DDUDUDU	61
UDDD	904	DUDUUD	174	DDUDUUD	91
UDDU	905	DUDUUU	224	DDUDUUU	113
UDUD	681	DUUDDD	246	DDUUDDD	138
UDUU	979	DUUDDU	212	DDUUDDU	116
UUDD	1,040	DUUDUD	181	DDUUDUD	82
UUDU	989	DUUDUU	232	DDUUDUU	121
UUUD	1,158	DUUUDD	265	DDUUUDD	131
UUUU	1,422	DUUUDU	267	DDUUUDU	128
DDDDD	400	DUUUUD	277	DDUUUUD	139
DDDDU	426	DUUUUU	349	DDUUUUU	195
DDDUD	388	UDDDDD	215	DUDDDDD	90

Table 1: Example patterns and count of occurrence. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Table 2 gives another example of application with a known string of three day, in order to forecast the fourth one. Supposing that the three-day string is DDU, for the forecast of the fourth one it is necessary to look at the training set in order to understand which outcome is more probable between up and down; in this specific example after DDU, a down movement is happened 760 times, while an up is happened 1,050 times. Now the forecast for the validation set can be done: after a string composed by “DDU” days, is more probable an up as fourth day.

Examples of the forecast of the fourth day

Training set 4-day pattern	Training set count	Validation set 3-day pattern	Validation set forecast for the 4th day
DDUD	760	DDU	U
DDUU	1,050	DDU	U

Table 2: Patter examples for forecasting shows DDUU is more likely to occur than DDUD. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

For each string, just the last movement has to be predicted: the most recurrent pattern for the following day in the training set will be chosen as the most likely conduct in the validation set: the results are seven forecasts, from T (the forecast of today, without knowing the past) to T-6+T (the forecast for the 7^{th} day, knowing the previous six) days. Furthermore for each day of the validation set a random number will be generated, to compare its performance with the one of the validation set forecast: if this number is higher than 0.5 the generated forecast is up, while if it is lower than 0.5 the forecast is down.

Those eight forecasts (the seven obtained by the training set and the random forecast) have been compared to the actual value of the day T+1 of the validation set (so the day that the model tries to predict) and the correct and incorrect predictions have been recorded. Table 3 shows the number and the percentage of correct predictions:

Number and percentage of correct forecasts

T-6 to T	T-5 to T	T-4 to T	T-3 to T	T-2 to T	T-1 to T	T	Random
212	218	212	205	205	195	219	178
54.8%	56.3%	54.8%	53.0%	53.0%	50.4%	56.6%	46.0%

Table 3: Percentage of time the validation set forecast was correct using 1-7 day patterns. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

The most correct prediction for tomorrow is the one that does not use past information, but is simply based on what happened more between up and down in the training set, without investigate on previous days, that is T, with a percentage of success of 56.6%; in reality, this is the least difficult prevision to do because every year the number of up days dominates, and so the forecast of a positive day is more correct. Very close to this value with little information, there is the forecast with a six-day string: the percentage of success is 56.3%. The least accurate prevision, instead, is the one with the random number: this is the evidence that prices do not move according to a random walk.

The following step is to understand if the predictions are more accurate in forecasting an up movement rather than a down one; this is done looking at Table 4.

The first row displays the seven strings, plus the random forecast, the notation T-6 to T means that the previous six days are known to predict the seventh and so on, while the second row indicates the number of up and down movements that have been forecasted for each string. The first column, instead, indicates the actual number of down and up movements and their percentages. Finally the bold numbers indicate the correct results, or rather, looking at the table, the intersection between the real down (in the first column) and the forecasted down, or the intersection between the real up and the forecasted up. The not-bold numbers indicate the incorrect forecasts, namely when the real outcome is an up, but the forecast is a down and viceversa, for every pattern of days, from one to seven.

Correct and incorrect forecasts

	Predicted movement for T+1															
Actual T+1	T-6 to T		T-5 to T		T-4 to T		T-3 to T		T-2 to T		T-1 to T		T		Random	
Movement	D	U	D	U	D	U	D	U	D	U	D	U	D	U	D	U
D	**48**	120	**40**	128	**45**	123	**41**	127	**41**	127	**72**	96	**0**	168	**75**	93
U	55	**164**	41	**178**	52	**167**	55	**164**	55	**164**	96	**123**	0	**219**	116	**103**
Data as %	Numbers are percent of column (predicted) total															
D	**46.6**	42.2	**49.3**	41.8	**46.3**	42.4	**42.7**	43.6	**42.7**	43.6	**42.8**	43.8	**0.00**	43.4	**39.2**	47.4
U	53.4	**57.7**	50.6	**58.1**	53.6	**57.5**	57.2	**56.3**	57.2	**56.3**	57.1	**56.1**	0.00	**56.5**	60.7	**52.5**

Table 4: Correct and incorrect up and down forecasts. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

Looking at percentages, an important result is that the correct forecasts of up movements is always higher than the correct forecasts obtained with the random number; once again it is an evidence that confutes the random walk hypothesis described in the first chapter.
The other interesting result is that the better prediction, both for up and for down forecasts, is obtained with a six-day string: it is correct on up days over the 58% of the time, and on down days almost the 50% of the time. This outcome confirms the general result obtained earlier and expressed in Table 3.

In the last step, Malliaris applied another tool in order to obtain a better result in the forecasting of future up and down.
He used a decision tree built with a C5.0 algorithm. A decision tree is a decision-support tool used to determine the probability of a specific outcome. The starting point is called root and corresponds to the first input given to the model, the other inputs are defined in the so-called interior nodes; every input can have different outcomes that are mutually exclusive (a statistical term indicating that two outcomes can not happen at the same time), represented by the branches. The branches end nodes are called leaf-nodes: they represent the class labels, so the specific scenarios that are chosen after the analysis of all the others.
The C5.0 algorithm is used in data mining[25] with a non-numeric target variable (the variable whose values have to be predicted by the other variables) and a set of input variables, which could be either numeric or non-numeric, generally producing accurate

[25] Data mining is a subfield of computer science (the discipline that form the basis for the design and use of computers), whose task is the automatic analysis of a large quantity of data in order to extract information and re-organize them into an understandable structure.

and efficient results in input classification; it divides the sample into smaller and smaller subsets, until possible, and it re-examines the least ones to delete the not significant subsets and to keep the ones that provides more information, using a step-wise process.

In other words, this algorithm checks every input variable in order to get closer to the final result at every step of the process, and picks out the one that would create a split with the most homogeneous groups of subsets. The result is a tree that classifies the input data row as likely to get a defined target value.

Malliaris built the decision tree with data up to December 2009, using three inputs: the Up-Down patterns from one to seven days, the number of up days in 1-5 days and the closing value of the current day. The purpose is to check the prediction ability of such patterns, previously analysed.

The result is a decision tree that makes the right prevision, using the training set, about the 67% of the time, as it can be seen in Table 5:

Training set results

Result	Count	Percentage
Correct	10,165	67.38
Incorrect	4,922	32.62
Total	15,087	

Table 5: Decision tree results on training data. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

For what concerns the outcome on the Validation set, they are displayed in Table 6:

Validation set results

Result	Count	Percentage
Correct	222	57.36
Incorrect	165	42.64
Total	387	

Table 6: Decision tree results on validation data. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

The percentage of correct forecasts in the Validation set is lower than the one in the Training set, but it is still higher than the one obtained with the simple pattern forecasts in Table 3.

Table 7 is a confusion matrix, a useful tool for summarizing an algorithm performance, which allows understanding the correct prediction of the decision tree in the validation set:

Correct and incorrect forecasts in the validation set

	Forecasted down	Forecasted up
Actual down tomorrow	72	96
	51.06 %	39.02 %
Actual up tomorrow	69	150
	48.94 %	60.98 %

Table 7: confusion matrix of up and down forecasts versus actual movement. Source: Malliaris, N-tuple S&P patterns across decades, 1950-2011.

In each row, the actual up and down are displayed, while in each column there are the forecasted values. The intersection between actual down (in the first row) and forecasted down (in the first column) gives the count of the down prediction that really happened in the validation set, viceversa the intersection between the actual down and the forecasted up (in the second column) represents how many up days where wrongly predicted, since the real outcome in the validation set is a down; the same reasoning is done with the second row. According to Table 7, the decision tree makes a correct forecast for down movements 51% of the time, instead, for an up movement makes a correct forecast about 61% of the time. Like the previous model (the application of past strings to the validation set), it makes a better prevision on up days rather than on down days, but with a little bit higher percentages.

This technique gives forecasts, and moreover classifies the input variables, in order of importance for the model itself. Specifically the first four variables ranked for importance are the direction of today's price, the closing value today, the seven-day pattern and the number of up movements in the last three days. Comparing the two models (the application of past patterns to future data and the decision tree), they both took close (few past days) and distant (several past days) information into consideration, since the simple pattern model works better with the one-day and six-day patterns, while the decision tree works better with one-day, three-day and seven day information.

By the way, even if the Random Walk Hypothesis is confuted, the percentage of correct forecasts is not so high; as said before, this is probably due to an overfitting problem: the Validation set could be too small for all the data contained in the Training set or the two models are too complex, since they consider a lot of variables and

parameters. An improvement could be to have a Validation set with a greater period of time or to simplify the models, reducing the number of parameters.

2.2 Conclusions

To summarize, this study is an investigation of daily movement patterns, from 1950 to 2011, in order to analyse the behaviour of ups and downs. First of all, the patterns have been observed in two different ways: as strings of ups and downs over time and as the number of occurring up in the different strings (from one day to seven). Both methods rank the data decade per decade, in order to check if specific patterns happen with similar frequency over time.

The results are interesting: in fact, thanks to the first method, it can be seen that every decade has more up than down days, moreover some specific string patterns continue with almost the same percentage across time; those results support the long-term buy and hold strategy. With the second method, the results are even more evocative, for example in the five-day combination (Figure 10), from 1950s on, there is always the same prices behaviour: more strings with three up than two, more with four up than one and more with five up than zero.

Thanks to those outcomes, future up and down have been predicted using two different methods:

- By applying past pattern strings to future data
- By using a decision tree

The first method, which uses strings of one specific length at a time, works better with both one-day and six-day information. The decision

tree methodology outperforms the previous approach, since it uses information from multiple inputs, and works better with one-day, three-day and seven-day patterns.

Those results underline the ability of this model in overcoming the Random Walk behaviour of prices.

To check if the methodology is successful, the same work will be done on the Italian index Comit General and also on the American Express Company, to understand if the method is performing also on private companies; the results and comments are provided in the next chapter.

Chapter 3

Application to the Italian index Comit Generale

In this chapter the same approach of Malliaris, with some improvements in the forecasting part, will be used in a sample of the Comit Generale closing prices and of the American Express Company.

3.1 Introduction

The same analysis proposed by Malliaris has been performed with a different index, the Comit Generale, in order to see if the results are similar and the methodology is effective.

The most famous Italian index FTSE MIB has not been selected because is too recent, specifically it is working from the 1st June of 2009; in order to have a panel of data long enough, necessary to make a good analysis, data from the Comit Generale have been taken.

The Comit Generale is a daily index processed by Banca Commerciale Italiana (BCI) in the past, and by Banca Intesa nowadays. It considers all the listed equities in the Milano Stock Exchange and it is computed once per day, with the closing prices of such equities.

The sample goes from the 2nd January 1973 to the 11th November 2016, for a total of 11,027 daily observations, which have been displayed decade per decade.

To make the whole analysis, the Python software has been used, a high-level programming language; the code is displayed in Appendix A.

Firstly, the returns have been ranked as up or down, then all the possible combination in the strings from one day to seven are counted and displayed with the help of bar graphs. Data have been analysed also taking into account just the number of up days in the strings from one day to seven. After the presentation of such patterns, a historical reading of the results will be given, with the purpose of demonstrating that these methods reflect what actually happened.

The second part is the forecasting one, in which all the outcomes are ranked into a table, which counts the recurrence of patterns. Those patterns are used to determine the most dominant events among the pairs, which differ only in the last movement, as Malliaris did in his work. As explained in the previous chapter, the results are employed in generating the forecasts for the following day. Finally a decision tree is applied to the same data, to check if, as happened to Malliaris, the results are more performing. To reduce the overfitting problem, also logistic regressions and neural networks are tested. Moreover, the method has been tested also on a company index, to test its ability with a more volatile index. The final step is the application of the new tools to the same sample Malliaris used.

3.2 Multi-decade patterns

The first step of this study is to classify and record all the returns as up (U) or down (D) movements. As said before, the classification as up or down is obtained looking at the value of the previous day. All the values are displayed as percentages; in this way a better comparison is done among the “not complete” decades (the first and the last

ones), and the others. The results of this first analysis are shown in Table 8 and Figure 11:

Percentage of Up and Down movements classified by decade

	1970s	1980s	1990s	2000s	2010s
D	51,2998%	46,6508%	48,3146%	46,4878%	46,9341%
U	48,7002%	53,3492%	51,6854%	53,5122%	53,0659%

Table 8: Up and Down movements' decade per decade

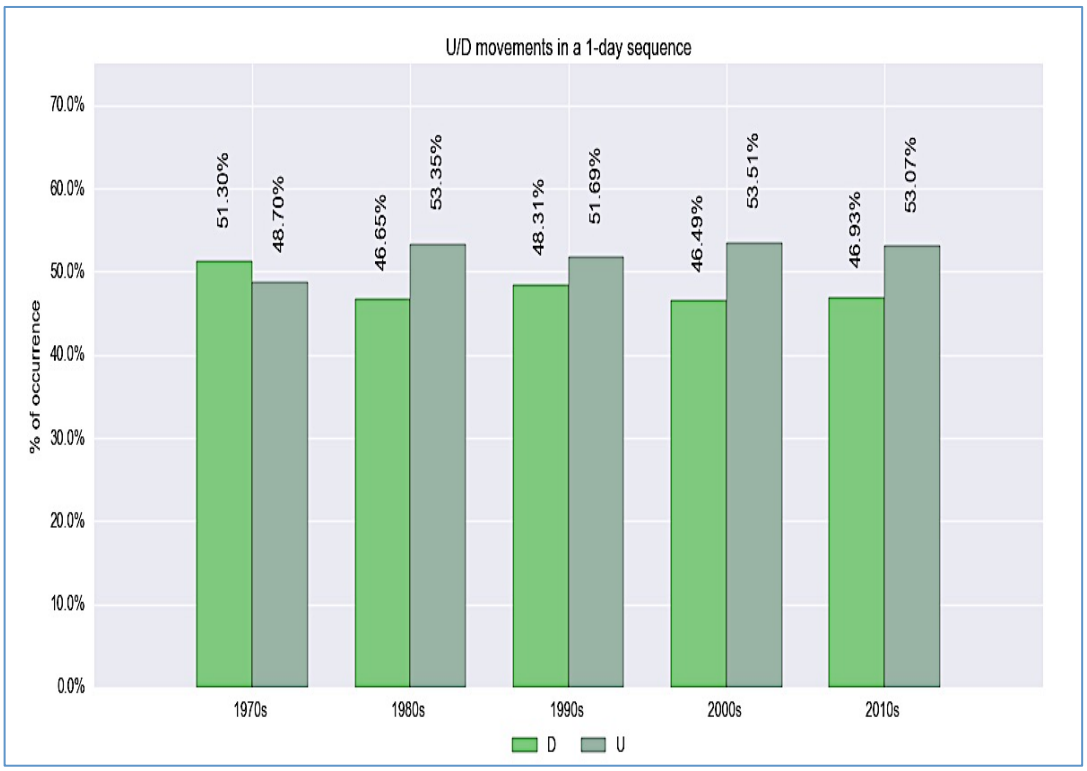


Figure 11: Up and Down movements per decade

On the x-axis are shown all the decades, with the strings displayed as a set of bars, on the y-axis the percentage of up and down is counted. To start, there is a predominance of up days in all the decades, with the exception of the first one, in which, we recall, data start from the

year 1973. Despite the crisis of 2008, the decade of years 2000 is the one with more up days, together with the 80s' decade. All in all, the index behaviour is constant: this proves that prices do not move following the Random Walk hypothesis (according to which prices move randomly), as happened for the S&P500 index.

Now the analysis is done creating string patterns of days, from two to seven; as example, if in a single week there are the following movements:

Monday	Tuesday	Wednesday	Thursday	Friday
U	U	D	U	D

There would be the two-day pattern UU associated with Tuesday, UD associated with Wednesday, DU associated with Thursday, UD with Friday, DU associated with Monday and so on. The same reasoning is done with the other string patterns.

All the results are collected in Table 9 and displayed in Figure 12:

Two-day strings

	1970s	1980s	1990s	2000s	2010s
DD	30,4624%	25,4163%	25,7326%	24,5559%	25,8028%
DU	20,8671%	21,2133%	22,6014%	21,9108%	21,1009%
UD	20,8092%	21,2530%	22,5612%	21,9503%	21,1583%
UU	27,8613%	32,1174%	29,1048%	31,5831%	31,9381%

Table 9: Up and Down movements in two-day strings

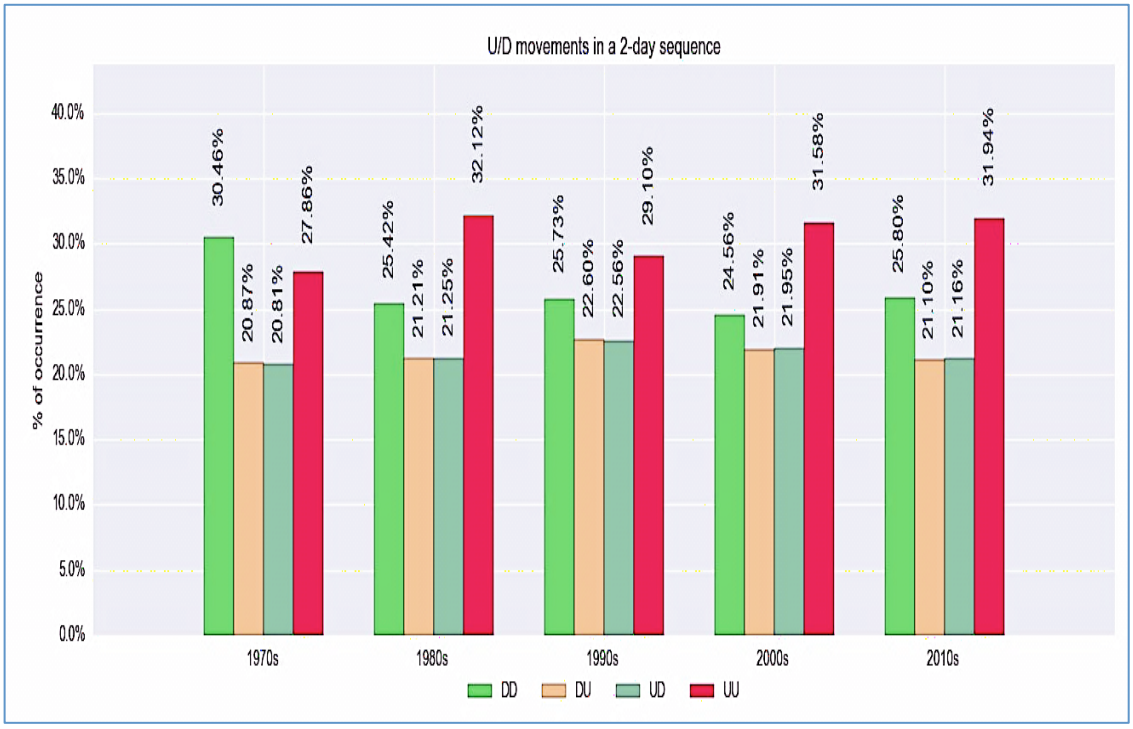


Figure 12: the four two-day strings of up and down across decades

According to Figure 12, the pattern of Up-Up is the most recurrent, except for the first decade, in which the predominant is the one of Down-Down. The Up-Up higher column is in the 80s decade, with a percentage of 32%, period in which Italy, and the whole world, experienced the economic boom (age in which a great recover occurred, thanks to the improvement of world economy). Furthermore, despite the sub-prime 2008 crisis, also the Up-Up column in the 00s decade is very high. Finally, the Up-Down and Down-Up columns are at the same level in all decades and with a discrete presence: this underlines a quite frequent change day to day in the index. There is not a clear increasing or decreasing trend, both with the up and with the down days, even if the two-up days column is predominant from the 80s on.

Now, Table 10 and Figure 13 present the three-day strings:

Three-day strings

	1970s	1980s	1990s	2000s	2010s
DDD	16,3678%	12,4950%	13,2530%	13,1122%	14,2857%
DDU	14,1122%	12,9314%	12,4900%	11,4534%	11,5318%
DUD	8,1550%	7,9730%	10,1205%	8,8468%	8,3190%
DUU	12,6663%	13,2487%	12,4900%	13,0727%	12,7940%
UDD	14,1122%	12,9314%	12,4900%	11,4534%	11,5318%
UDU	6,7091%	8,2904%	10,0803%	10,4660%	9,5812%
UUD	12,6663%	13,2884%	12,4498%	13,0727%	12,8514%
UUU	15,2111%	18,8417%	16,6265%	18,5229%	19,1050%

Table 10: Up and Down movements in three-day strings

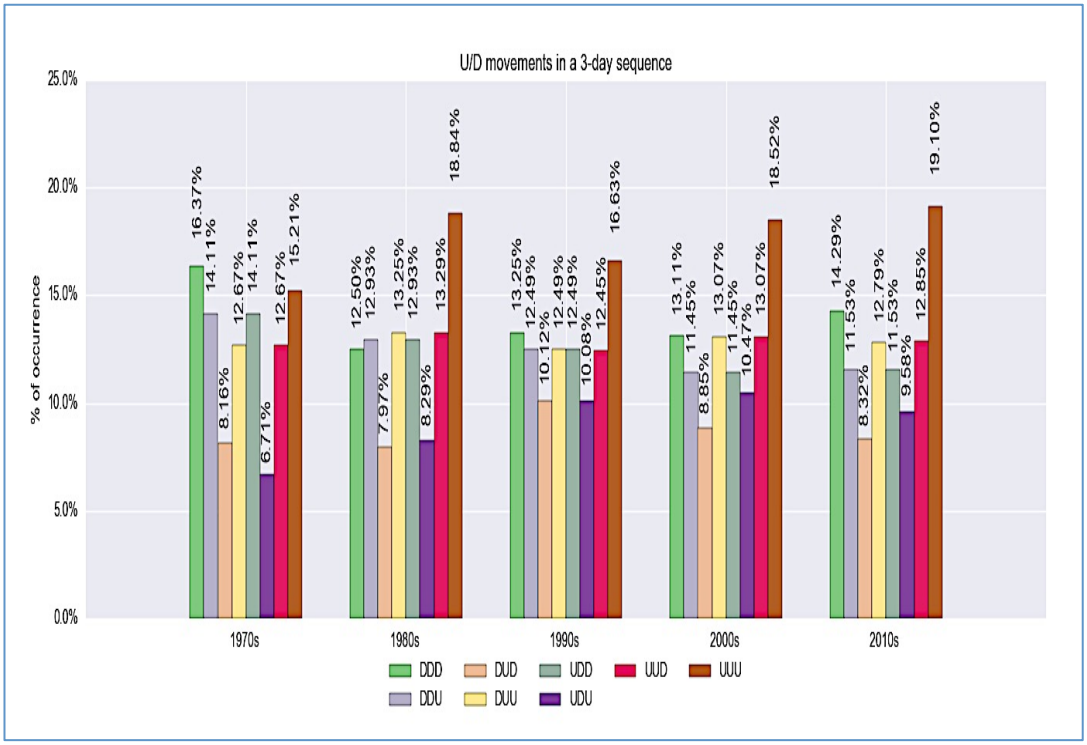


Figure 13: Three-day strings of Up and Down across decades

Looking at the percentages, the UUU column is always predominant, with the highest point in years 00s and 80s, except for the 70s, in which predominates the DDD column (16.4%), as happened in the previous strings. The DDU and UDD columns are always at the same level; similarly DUU and UUD are almost close in every decade. The lower patterns are DUD and UDU, highlighting the fact that day-to-day changes are less frequent. Another interesting fact is the decrease of the DDD column over years, with a corresponding increase in the UUU column decade by decade, with the exception of years 90s, where the UUU pattern is still predominant, but it does not increase with respect to the previous decade.

Four- day strings

	1970s	1980s	1990s	2000s	2010s
DDDD	8,6806%	6,1508%	7,2318%	6,9143%	7,7497%
DDDU	7,6968%	6,3492%	6,0265%	6,2031%	6,5442%
DDUD	6,0764%	5,3175%	6,1470%	4,5832%	4,5350%
DDUU	8,0440%	7,6190%	6,3479%	6,8748%	7,0034%
DUDD	5,9606%	5,3968%	5,8658%	5,1363%	4,6498%
DUDU	2,1991%	2,5397%	4,2587%	3,7139%	3,6165%
DUUD	5,9028%	5,5952%	4,8212%	5,7290%	5,6257%
DUUU	6,7708%	7,6587%	7,6738%	7,3489%	7,1757%
UDDD	7,6968%	6,3492%	6,0265%	6,2031%	6,5442%
UDDU	6,4236%	6,5873%	6,4685%	5,2548%	4,9943%
UDUD	2,0833%	2,6587%	3,9775%	4,2671%	3,7887%
UDUU	4,5718%	5,6349%	6,1069%	6,2031%	5,7979%
UUDD	8,1597%	7,5397%	6,6292%	6,2821%	6,8886%
UUDU	4,5139%	5,7540%	5,8256%	6,7562%	5,9701%
UUUD	6,7708%	7,6984%	7,6336%	7,3489%	7,2331%
UUUU	8,4491%	11,1508%	8,9594%	11,1814%	11,8829%

Table 11: Up and Down movement in four-day strings

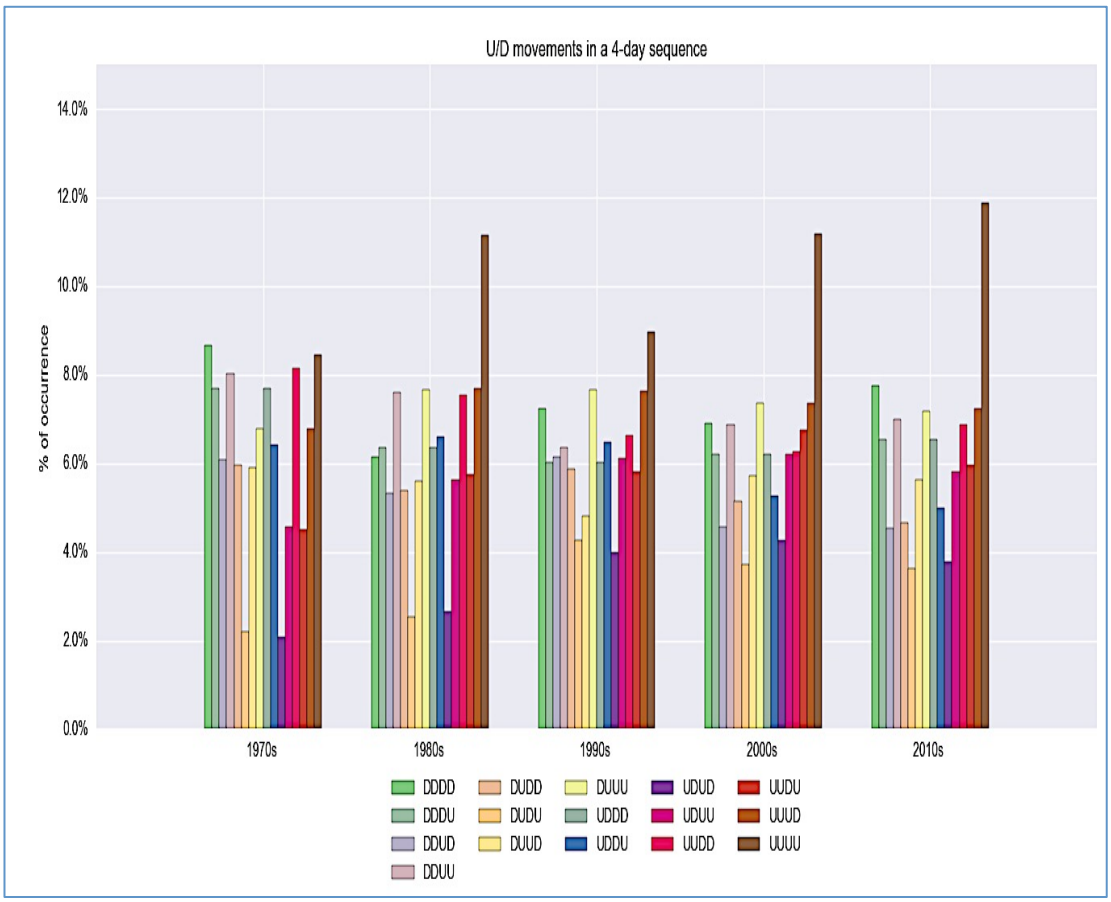


Figure 14: Four-day strings of Up and Down across decades

Table 11 with Figure 14 show the four-day string patterns: the UUUU column is still predominant across decades, except for the first one, in which prevails the DDDD pattern with a percentage of 8.7%. With respect to the previous strings, in the 70s there is a lower gap between the best (four up days) and the worst (zero up days) columns: in a longer period the situation looks better with less presence of many consecutive negative days. The UUUU column is, as seen in the previous figures, higher in the 10s and, right after, in the 80s. DDDU and UDDD columns are always at the same level, while UDUU and UUDU are pretty close across decades. Finally the DUDU and UDUD patterns are at the lowest values, underlying again the poor frequency of daily changes.

Five-day strings

	1970s	1980s	1990s	2000s	2010s
DDDDD	4,4586%	3,1759%	3,7781%	3,6364%	4,6525%
DDDDU	4,2270%	2,9774%	3,4566%	3,2806%	3,1017%
DDDUD	3,4742%	2,4613%	3,2556%	2,6877%	2,4698%
DDDUU	4,2270%	3,8904%	2,7733%	3,5178%	4,0781%
DDUDD	4,2270%	3,6919%	3,5370%	2,6877%	2,5847%
DDUDU	1,8529%	1,6276%	2,6125%	1,8972%	1,9529%
DDUUD	3,8796%	3,3744%	2,7331%	3,3597%	2,6996%
DDUUU	4,1691%	4,2477%	3,6174%	3,5178%	4,3079%
DUDDD	3,1268%	2,9377%	2,4920%	2,6087%	2,6996%
DUDDU	2,8373%	2,4613%	3,3762%	2,5296%	1,9529%
DUDUD	0,6369%	0,9528%	1,6881%	1,4229%	1,2062%
DUDUU	1,5634%	1,5879%	2,5723%	2,2925%	2,4124%
DUUDD	3,8796%	3,7316%	2,3714%	2,9249%	2,8145%
DUUDU	2,0266%	1,8658%	2,4518%	2,8063%	2,8145%
DUUUD	2,7215%	3,4141%	3,6174%	3,1225%	2,4124%
DUUUU	4,0533%	4,2477%	4,0595%	4,2292%	4,7674%
UDDDD	4,2270%	2,9774%	3,4566%	3,2806%	3,1017%
UDDDU	3,4742%	3,3744%	2,5723%	2,9249%	3,4463%
UDDUD	2,6057%	2,8583%	2,8939%	1,8972%	2,0678%
UDDUU	3,8217%	3,7316%	3,5772%	3,3597%	2,9294%
UDUDD	1,7371%	1,7070%	2,3312%	2,4506%	2,0678%
UDUDU	0,3474%	0,9131%	1,6479%	1,8182%	1,6657%
UDUUD	1,9687%	2,2231%	2,0900%	2,3715%	2,9294%
UDUUU	2,6057%	3,4141%	4,0193%	3,8340%	2,8719%
UUDDD	4,5744%	3,4141%	3,5370%	3,5573%	3,8484%
UUDDU	3,5900%	4,1286%	3,0949%	2,7273%	3,0442%
UUDUD	1,4476%	1,7070%	2,2910%	2,8458%	2,5847%
UUDUU	3,0110%	4,0492%	3,5370%	3,9130%	3,3889%
UUUDD	4,2849%	3,8110%	4,2605%	3,3597%	4,0781%
UUUDU	2,4899%	3,8904%	3,3762%	3,9526%	3,1591%
UUUUD	4,0533%	4,2874%	4,0193%	4,2292%	4,8248%
UUUUU	4,4007%	6,8678%	4,9035%	6,9565%	7,0649%

Table 12: Up and Down movement in five-day strings

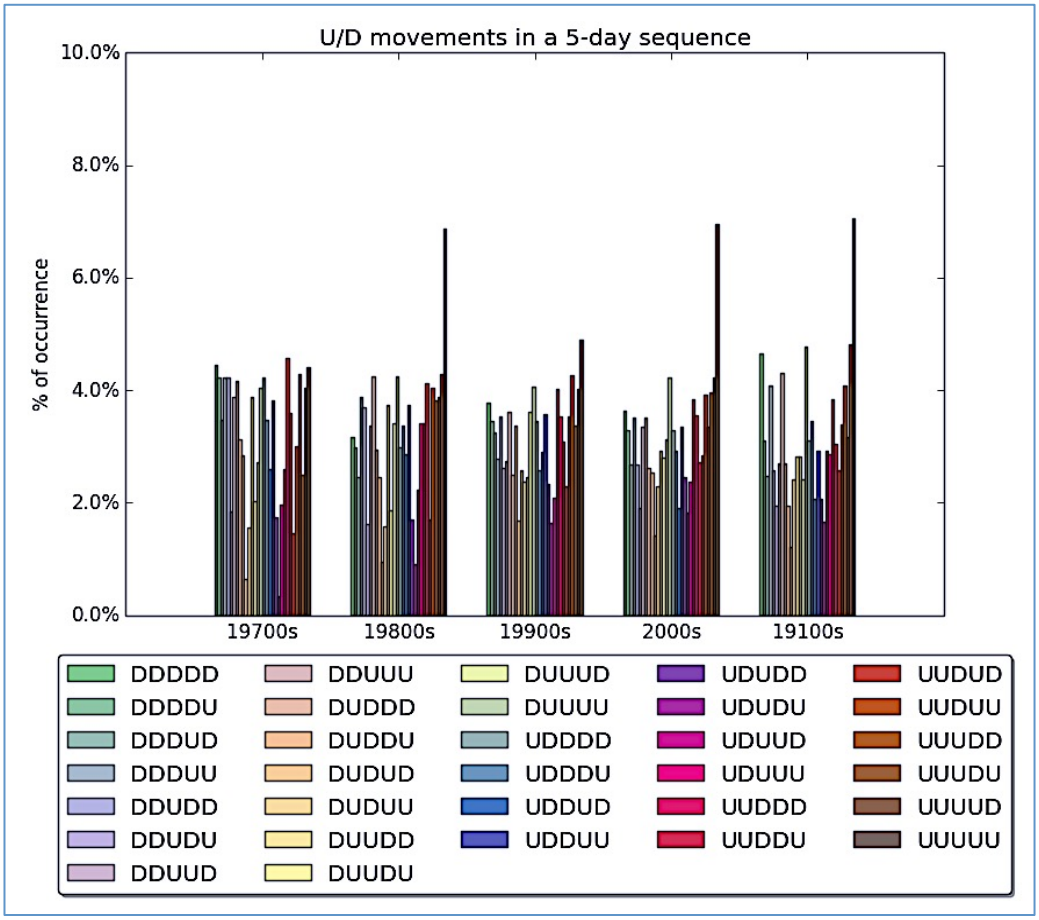


Figure 15: Five-day strings of Up and Down across decades

Table 12 and Figure 15 show again the predominance of the UUUUU column across decades, except for the 70s, in which five consecutive ups happen 4.4% of the time, while five consecutive downs happen 4.45% of the time: the two values are getting even more close with respect to the previous strings. For what concerns the five consecutive up days, the better result is in the years 10s, while the five consecutive down days are lower in the 80s, as it happened in the previous strings. Again, the columns that represent frequent daily changes (UDUDU and DUDUD) have the lower values, among all the other combinations.

Six-day strings

	1970s	1980s	1990s	2000s	2010s
DDDDDD	2,3175%	1,7474%	1,9702%	1,9375%	2,7586%
DDDDDU	2,1437%	1,4297%	1,8094%	1,7003%	1,8966%
DDDDUD	1,8540%	1,2311%	1,7692%	1,4235%	1,1494%
DDDDUU	2,3754%	1,7474%	1,6888%	1,8584%	1,9540%
DDDUDD	2,4913%	1,8666%	1,9300%	1,7003%	1,4368%
DDDUDU	0,9849%	0,5957%	1,3269%	0,9885%	1,0345%
DDDUUD	1,9119%	1,7871%	1,2063%	1,8584%	1,4368%
DDDUUU	2,3175%	2,1048%	1,5682%	1,6607%	2,6437%
DDUDDD	2,3175%	2,0254%	1,6084%	1,4630%	1,5517%
DDUDDU	1,9119%	1,6680%	1,9300%	1,2258%	1,0345%
DDUDUD	0,5214%	0,5957%	0,8444%	0,8304%	0,6897%
DDUDUU	1,3326%	1,0326%	1,7692%	1,0676%	1,2644%
DDUUDD	2,5492%	2,1843%	1,2867%	1,7794%	1,4943%
DDUUDU	1,3326%	1,1914%	1,4475%	1,5817%	1,2069%
DDUUUD	1,5064%	1,5886%	1,9702%	1,5421%	1,4368%
DDUUUU	2,6651%	2,6608%	1,6486%	1,9771%	2,8736%
DUDDDD	1,2746%	1,2708%	1,3671%	1,2653%	1,2644%
DUDDDU	1,8540%	1,6680%	1,1259%	1,3444%	1,4368%
DUDDUD	1,0429%	1,2311%	1,2465%	0,9490%	0,6322%
DUDDUU	1,7961%	1,2311%	2,1311%	1,5817%	1,3218%
DUDUDD	0,5214%	0,5957%	0,9248%	0,7513%	0,6897%
DUDUDU	0,1159%	0,3177%	0,7640%	0,6722%	0,5172%
DUDUUD	0,5794%	0,5957%	0,8444%	0,7513%	1,0920%
DUDUUU	0,9849%	0,9929%	1,7290%	1,5421%	1,3218%
DUUDDD	2,0278%	1,8268%	1,3671%	1,6607%	1,3793%
DUUDDU	1,8540%	1,9063%	1,0052%	1,2653%	1,4368%
DUUDUD	0,6952%	0,7149%	1,1259%	1,1072%	1,2069%
DUUDUU	1,3326%	1,1517%	1,3269%	1,7003%	1,6092%
DUUUDD	1,7961%	1,8268%	1,9300%	1,7398%	1,0920%
DUUUDU	0,9270%	1,5886%	1,6888%	1,3839%	1,3218%
DUUUUD	1,9119%	1,7871%	1,8898%	1,2653%	1,8391%
DUUUUU	2,1437%	2,4623%	2,1713%	2,9656%	2,9310%
UDDDDD	2,1437%	1,4297%	1,8094%	1,7003%	1,8966%
UDDDDU	2,0857%	1,5488%	1,6486%	1,5817%	1,2069%
UDDDUD	1,6222%	1,2311%	1,4877%	1,2653%	1,3218%
UDDDUU	1,8540%	2,1446%	1,0856%	1,6607%	2,1264%
UDDUDD	1,7381%	1,8268%	1,6084%	0,9885%	1,1494%
UDDUDU	0,8691%	1,0326%	1,2867%	0,9095%	0,9195%
UDDUUD	1,9699%	1,5886%	1,5279%	1,5026%	1,2644%
UDDUUU	1,8540%	2,1446%	2,0507%	1,8584%	1,6667%
UDUDDD	0,8111%	0,9134%	0,8846%	1,1467%	1,1494%
UDUDDU	0,9270%	0,7943%	1,4475%	1,3049%	0,9195%
UDUDUD	0,1159%	0,3574%	0,8444%	0,5931%	0,5172%
UDUDUU	0,2317%	0,5560%	0,8042%	1,2258%	1,1494%
UDUUDD	1,2746%	1,5488%	1,0856%	1,1467%	1,3218%
UDUUDU	0,6952%	0,6751%	1,0052%	1,2258%	1,6092%
UDUUUD	1,2167%	1,8268%	1,6486%	1,5817%	0,9770%
UDUUUU	1,3905%	1,5886%	2,3723%	2,2539%	1,8966%
UUDDDD	2,9548%	1,7077%	2,0909%	2,0166%	1,8391%
UUDDDU	1,6222%	1,7077%	1,4475%	1,5421%	2,0115%
UUDDUD	1,5643%	1,6283%	1,6486%	0,9490%	1,4368%
UUDDUU	2,0278%	2,5020%	1,4475%	1,7794%	1,6092%
UUDUDD	1,2167%	1,1120%	1,4073%	1,7003%	1,3793%
UUDUDU	0,2317%	0,5957%	0,8846%	1,1467%	1,1494%
UUDUUD	1,3905%	1,6283%	1,2465%	1,6212%	1,8391%
UUDUUU	1,6222%	2,4226%	2,2919%	2,2934%	1,5517%
UUUDDD	2,5492%	1,5886%	2,1713%	1,8980%	2,4713%
UUUDDU	1,7381%	2,2240%	2,0909%	1,4630%	1,6092%
UUUDUD	0,7532%	0,9929%	1,1661%	1,7398%	1,3793%
UUUDUU	1,6802%	2,8991%	2,2115%	2,2143%	1,7816%
UUUUDD	2,4913%	1,9857%	2,3321%	1,6212%	2,9885%
UUUUDU	1,5643%	2,3034%	1,6888%	2,5702%	1,8391%
UUUUUD	2,1437%	2,4623%	2,1311%	2,9656%	2,9885%
UUUUUU	2,2596%	4,4083%	2,7342%	3,9937%	4,0805%

Table 13: Up and Down movement in six-day strings

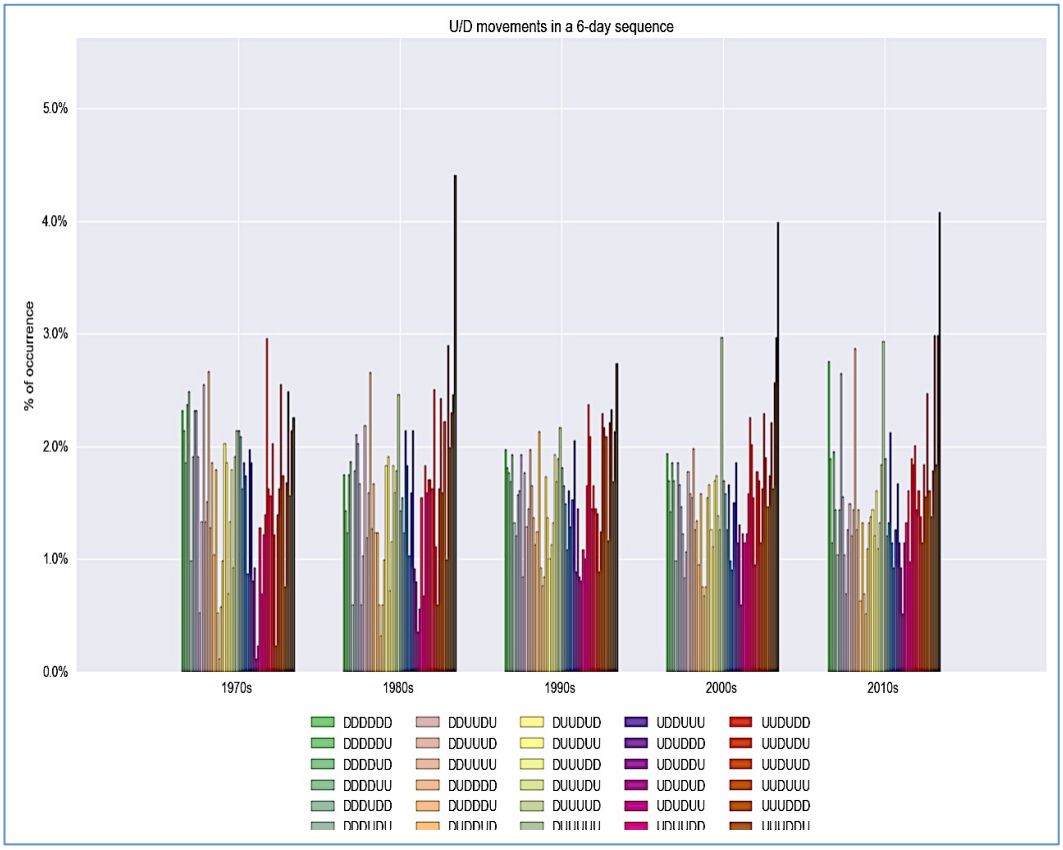


Figure 16: Six-day strings of Up and Down across decades

Table 14 and Figure 16, which show the six-day strings, act similarly to the previous ones. In the first decade, UUDDDD is the predominant column, with an occurrence percentage of the 2.95%. From the 80s on, the predominant column is UUUUUU. In particularly it shows a great increase in the 80s with an almost double percentage with respect to the others patterns; in the 90s it decreases a little, to rise again in every decade. For what concerns the DDDDDD column, it decreases in the 80s, years in which Italy and the whole world experienced the economic boom, but it increases again with a peak of 2.76% in the 10s, contrary to what happens in the previous strings of days, where the all-down column has a slower increase. DUUUUU and UUUUUD columns always have close and very high values, underlying stability in

the index; in fact UDUDUD and DUDUDU patterns are lower than the others, as happened in the previous strings, confirming again that daily changes are not so frequent.

Seven-day strings

	1970s	1980s	1990s	2000s	2010s
DDDDDDD	1,2174%	0,8343%	1,0056%	1,0285%	1,7251%
DDDDDDU	1,1014%	0,9138%	0,9654%	0,9098%	1,0351%
DDDDDUD	0,9275%	0,5562%	1,0459%	0,6725%	0,6901%
DDDDDUU	1,2174%	0,8741%	0,7643%	1,0285%	1,2076%
DDDDUDD	1,3913%	0,8741%	1,0056%	0,9494%	0,6901%
DDDDUDU	0,4638%	0,3576%	0,7643%	0,4747%	0,4600%
DDDDUUD	0,9855%	0,7549%	0,6034%	0,9494%	0,5750%
DDDDUUU	1,3913%	0,9932%	1,0861%	0,9098%	1,3801%
DDDUDDD	1,2754%	1,0330%	0,8045%	1,0285%	0,6901%
DDDUDDU	1,2174%	0,8343%	1,1263%	0,6725%	0,7476%
DDDUDUD	0,2899%	0,2781%	0,4425%	0,3165%	0,3450%
DDDUDUU	0,6957%	0,3178%	0,8850%	0,6725%	0,6901%
DDDUUDD	1,3913%	1,0330%	0,6436%	0,9098%	0,8051%
DDDUUDU	0,5217%	0,7549%	0,5632%	0,9494%	0,6325%
DDDUUUD	0,6957%	0,6754%	0,8850%	0,6329%	1,0926%
DDDUUUU	1,6232%	1,4303%	0,6838%	1,0285%	1,5526%
DDUDDDD	0,9855%	0,8741%	0,9252%	0,6725%	0,7476%
DDUDDDU	1,3333%	1,1522%	0,6838%	0,7911%	0,8051%
DDUDDUD	0,8696%	0,9138%	0,5632%	0,3560%	0,4025%
DDUDDUU	1,0435%	0,7549%	1,3677%	0,8703%	0,6325%
DDUDUDD	0,4058%	0,4370%	0,3620%	0,5538%	0,4600%
DDUDUDU	0,1159%	0,1589%	0,4827%	0,2769%	0,2300%
DDUDUUD	0,5217%	0,4370%	0,6034%	0,2769%	0,8051%
DDUDUUU	0,8116%	0,5959%	1,1665%	0,7911%	0,4600%
DDUUDDD	1,4493%	0,8741%	0,6838%	1,0285%	0,6901%
DDUUDDU	1,1014%	1,3111%	0,6034%	0,7516%	0,8051%
DDUUDUD	0,5217%	0,3576%	0,7241%	0,5934%	0,3450%
DDUUDUU	0,8116%	0,8343%	0,7241%	0,9889%	0,8626%
DDUUUDD	1,1594%	0,8343%	1,1665%	0,7911%	0,6325%
DDUUUDU	0,3478%	0,7549%	0,8045%	0,7516%	0,8051%
DDUUUUD	1,3333%	1,1919%	0,9252%	0,4747%	1,1501%
DDUUUUU	1,3333%	1,4700%	0,7241%	1,5032%	1,7251%
DUDDDDD	0,5797%	0,6754%	0,5229%	0,5934%	0,9776%
DUDDDDU	0,6957%	0,5959%	0,8447%	0,6725%	0,2875%
DUDDDUD	0,9275%	0,5165%	0,6436%	0,7516%	0,6325%
DUDDDUU	0,9275%	1,1522%	0,4827%	0,5934%	0,8051%
DUDDUDD	0,6957%	0,6754%	0,6436%	0,3956%	0,2875%
DUDDUDU	0,3478%	0,5562%	0,6034%	0,5538%	0,3450%
DUDDUUD	0,8696%	0,4768%	0,9252%	0,7120%	0,5750%
DUDDUUU	0,9275%	0,7549%	1,2068%	0,8703%	0,7476%
DUDUDDD	0,2899%	0,3576%	0,3218%	0,3165%	0,4600%
DUDUDDU	0,2319%	0,2384%	0,6034%	0,4351%	0,2300%
DUDUDUD	0,0000%	0,0795%	0,3620%	0,1978%	0,2875%
DUDUDUU	0,1159%	0,2384%	0,4023%	0,4747%	0,2300%
DUDUUDD	0,4058%	0,4370%	0,4425%	0,3560%	0,4025%
DUDUUDU	0,1739%	0,1589%	0,4023%	0,3956%	0,6901%
DUDUUUD	0,3478%	0,5959%	0,7643%	0,8703%	0,5750%
DUDUUUU	0,6377%	0,3973%	0,9654%	0,6725%	0,7476%
DUUDDDD	1,1014%	0,8343%	0,8447%	1,0285%	0,7476%
DUUDDDU	0,9275%	0,9932%	0,5229%	0,6329%	0,6325%
DUUDDUD	0,9275%	0,6754%	0,4827%	0,2373%	0,5175%
DUUDDUU	0,9275%	1,2316%	0,5229%	1,0285%	0,9201%
DUUDUDD	0,5797%	0,4370%	0,6838%	0,6329%	0,5175%
DUUDUDU	0,1159%	0,2781%	0,4425%	0,4747%	0,6325%
DUUDUUD	0,6957%	0,4768%	0,5632%	0,7516%	0,7476%
DUUDUUU	0,6377%	0,6754%	0,7643%	0,9494%	0,8626%

DUUUDDD	1,1014%	0,7549%	1,0861%	1,1076%	0,6901%
DUUUDDU	0,6957%	1,0727%	0,8447%	0,6329%	0,4025%
DUUUDUD	0,2319%	0,4768%	0,6436%	0,7516%	0,6325%
DUUUDUU	0,6377%	1,1124%	1,0459%	0,6329%	0,6901%
DUUUUDD	1,2174%	0,9138%	1,0861%	0,5934%	1,0926%
DUUUUDU	0,6957%	0,8741%	0,8045%	0,6725%	0,7476%
DUUUUUD	0,9275%	0,9138%	0,8447%	1,3054%	1,2651%
DUUUUUU	1,2174%	1,5495%	1,2872%	1,6614%	1,6676%
UDDDDDD	1,1014%	0,9138%	0,9654%	0,9098%	1,0351%
UDDDDDU	1,0435%	0,5165%	0,8447%	0,7911%	0,8626%
UDDDDUD	0,9275%	0,6754%	0,7241%	0,7516%	0,4600%
UDDDDUU	1,1594%	0,8741%	0,9252%	0,8307%	0,7476%
UDDDUDD	1,1014%	0,9932%	0,9252%	0,7516%	0,7476%
UDDDUDU	0,5217%	0,2384%	0,5632%	0,5142%	0,5750%
UDDDUUD	0,9275%	1,0330%	0,6034%	0,9098%	0,8626%
UDDDUUU	0,9275%	1,1124%	0,4827%	0,7516%	1,2651%
UDDUDDD	1,0435%	0,9932%	0,8045%	0,4351%	0,8626%
UDDUDDU	0,6957%	0,8343%	0,8045%	0,5538%	0,2875%
UDDUDUD	0,2319%	0,3178%	0,4023%	0,5142%	0,3450%
UDDUDUU	0,6377%	0,7151%	0,8850%	0,3956%	0,5750%
UDDUUDD	1,1594%	1,1522%	0,6436%	0,8703%	0,6901%
UDDUUDU	0,8116%	0,4370%	0,8850%	0,6329%	0,5750%
UDDUUUD	0,8116%	0,9138%	1,0861%	0,9098%	0,3450%
UDDUUUU	1,0435%	1,2316%	0,9654%	0,9494%	1,3226%
UDUDDDD	0,2899%	0,3973%	0,4425%	0,5934%	0,5175%
UDUDDDU	0,5217%	0,5165%	0,4425%	0,5538%	0,6325%
UDUDDUD	0,1739%	0,3178%	0,6838%	0,5934%	0,2300%
UDUDDUU	0,7536%	0,4768%	0,7643%	0,7120%	0,6901%
UDUDUDD	0,1159%	0,1589%	0,5632%	0,1978%	0,2300%
UDUDUDU	0,0000%	0,1589%	0,2816%	0,3956%	0,2875%
UDUDUUD	0,0580%	0,1589%	0,2414%	0,4747%	0,2875%
UDUDUUU	0,1739%	0,3973%	0,5632%	0,7516%	0,8626%
UDUUDDD	0,5797%	0,9535%	0,6838%	0,6329%	0,6901%
UDUUDDU	0,6957%	0,5959%	0,4023%	0,5142%	0,6325%
UDUUDUD	0,1739%	0,3576%	0,4023%	0,5142%	0,8626%
UDUUDUU	0,5217%	0,3178%	0,6034%	0,7120%	0,7476%
UDUUUDD	0,6377%	0,9932%	0,7643%	0,9494%	0,4600%
UDUUUDU	0,5797%	0,8343%	0,8850%	0,6329%	0,5175%
UDUUUUD	0,5797%	0,5959%	0,9654%	0,7911%	0,6901%
UDUUUUU	0,8116%	0,9932%	1,4079%	1,4636%	1,2076%
UUDDDDD	1,5652%	0,7549%	1,2872%	1,1076%	0,9201%
UUDDDDU	1,3913%	0,9535%	0,8045%	0,9098%	0,9201%
UUDDDUD	0,6957%	0,7151%	0,8447%	0,5142%	0,6901%
UUDDDUU	0,9275%	0,9932%	0,6034%	1,0285%	1,3226%
UUDDUDD	1,0435%	1,1522%	0,9654%	0,5934%	0,8626%
UUDDUDU	0,5217%	0,4768%	0,6838%	0,3560%	0,5750%
UUDDUUD	1,1014%	1,1124%	0,6034%	0,7911%	0,6901%
UUDDUUU	0,9275%	1,3905%	0,8447%	0,9889%	0,9201%
UUDUDDD	0,5217%	0,5562%	0,5632%	0,8307%	0,6901%
UUDUDDU	0,6957%	0,5562%	0,8447%	0,8703%	0,6901%
UUDUDUD	0,1159%	0,2781%	0,4827%	0,3956%	0,2300%
UUDUDUU	0,1159%	0,3178%	0,4023%	0,7516%	0,9201%
UUDUUDD	0,8696%	1,1124%	0,6436%	0,7911%	0,9201%
UUDUUDU	0,5217%	0,5165%	0,6034%	0,8307%	0,9201%
UUDUUUD	0,8696%	1,2316%	0,8850%	0,7120%	0,4025%
UUDUUUU	0,7536%	1,1919%	1,4079%	1,5823%	1,1501%
UUUDDDD	1,8551%	0,8741%	1,2470%	0,9889%	1,0926%
UUUDDDU	0,6957%	0,7151%	0,9252%	0,9098%	1,3801%
UUUDDUD	0,6377%	0,9535%	1,1665%	0,7120%	0,9201%
UUUDDUU	1,1014%	1,2714%	0,9252%	0,7516%	0,6901%
UUUDUDD	0,6377%	0,6754%	0,7241%	1,0680%	0,8626%
UUUDUDU	0,1159%	0,3178%	0,4425%	0,6725%	0,5175%
UUUDUUD	0,6957%	1,1522%	0,6838%	0,8703%	1,0926%
UUUDUUU	0,9855%	1,7481%	1,5286%	1,3449%	0,6901%
UUUUDDD	1,4493%	0,8343%	1,0861%	0,7911%	1,7826%
UUUUDDU	1,0435%	1,1522%	1,2470%	0,8307%	1,2076%
UUUUDUD	0,5217%	0,5165%	0,5229%	0,9889%	0,7476%
UUUUDUU	1,0435%	1,7878%	1,1665%	1,5823%	1,0926%
UUUUUDD	1,2754%	1,0727%	1,2470%	1,0285%	1,8976%
UUUUUDU	0,8696%	1,3905%	0,8850%	1,8987%	1,0926%
UUUUUUD	1,2174%	1,5495%	1,2872%	1,6614%	1,7251%
UUUUUUU	1,0435%	2,8605%	1,4481%	2,3339%	2,3577%

Table 14: Up and Down movement in six-day strings

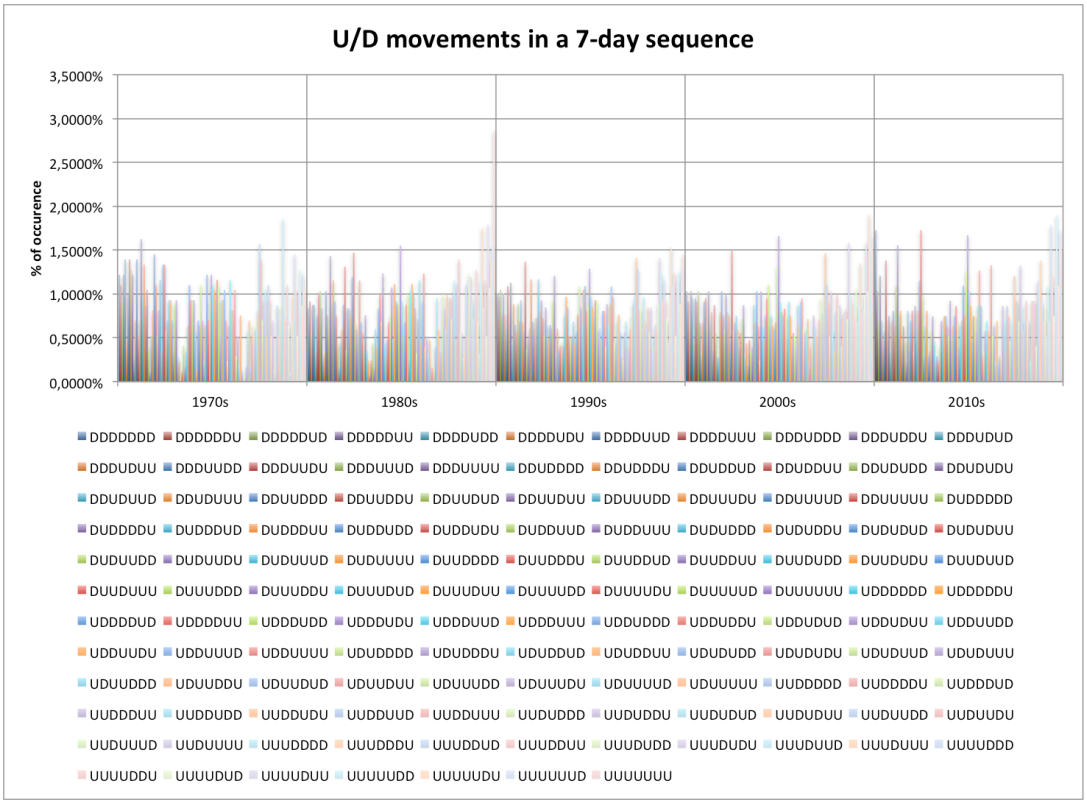


Figure 17: Seven-day strings of Up and Down across decades

Figure 17 and Table 14 display all the possible combinations for the seven-day string. Since the plot is very crowded and all the different combinations are 128 (those displayed in the legend below the plot), Table 15 has been created to underline just the maximum and minimum points for every decade.

Highest and lowest value for the seven-day strings

	70s	80s	90s	00s	10s
min	DUDUDUD	DUDUDUD	UDUDUUD	DUDUDUD	DDUDUDU
	0,0000%	0,0795%	0,2414%	0,1978%	0,2300%
max	UUUDDDD	UUUUUUU	UUUDUUU	UUUUUUU	UUUUUUU
	1,8551%	2,8605%	1,5286%	1,5286%	1,5286%

Table 15: Maximum and minimum points for every decade

According to Table 15, seven consecutive days of up are predominant in the 80s, 00s and 10s, while in the 70s prevails three days of up followed by four days of down. In the 70s, 80s and 00s the lower values correspond to DUDUDUD column, as happened in the previous strings, while, in the 90s the lower column is UDUDDDU and in the 10s is DDUDUDU. The decade of years 80s is again the most positive among the others with the predominance of seven consecutive up days, confirming again the push of the economy in those years; on the other hand the decade of years 70s is again the worst, with the lower value of seven consecutive up days and a quite high percentage of seven consecutive negative days. Looking just at the first column (DDDDDDD), there is a decrease in the percentage passing from the 70s to the 80s, and then an increase, moreover in the last decade. For what concerns the last column (UUUUUUU), it always has an increasing trend, except for the passage from the 80s to the 90s, in which there is a drop (from 2.86% to 1.44%).

Now the data will be displayed in an altered fashion: the focus is just on the number of up days inside the strings. To be clear, looking at the two-day strings, there are four possible outcomes (UU, UD, DU, DD); but if just the amount of up is counted, the outcomes are reduced to three (0 up, 1 up or 2 ups): UD and DU are both examples of 1 up.
As before, all the results, from one to seven days, are displayed both with tables and graphs, with the number expressed in percentages in the y-axis, and the decades in the x-axis.

Count of up days in one-day string

	1970s	1980s	1990s	2000s	2010s
0	51,2998%	46,6508%	48,3146%	46,4878%	46,9341%
1	48,7002%	53,3492%	51,6854%	53,5122%	53,0659%

Table 16: Up movements in one-day string

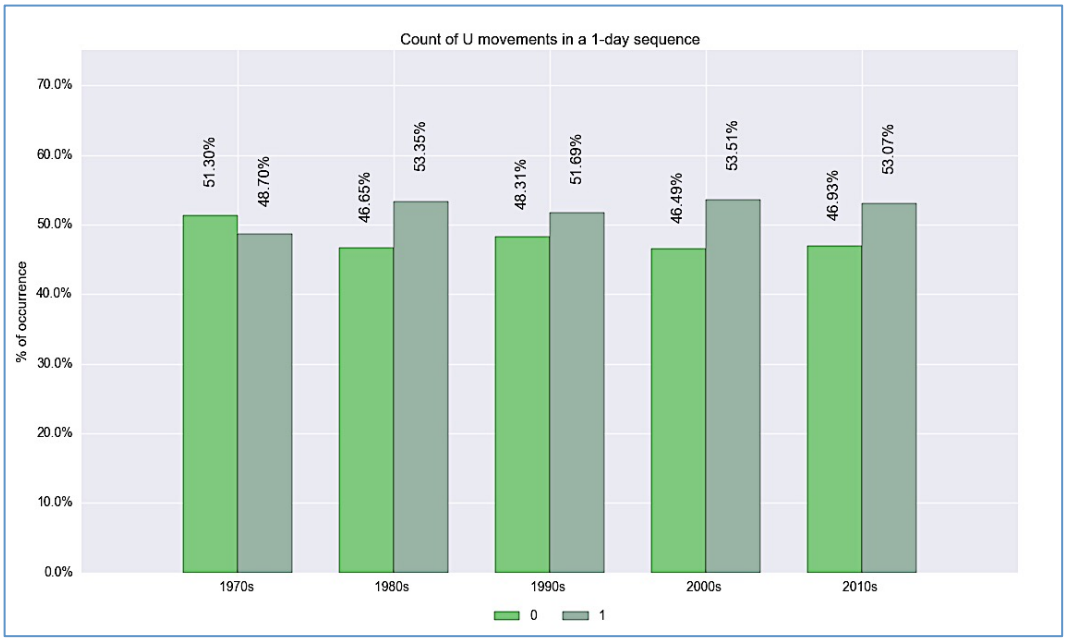


Figure 18: Percentage of up movements in one-day string

Table 16 and Figure 18 represent the one-day string, in which there could be 0 or 1 up movement. This plot is obviously identical to the one in Figure 11, since the string is composed by only one day. It is recalled that one up columns is predominant in every decade, except for the 70s. As stressed before, in the long period the trend is constant: this empirically proves that prices do not move according to a Random Walk. Contrary to the previous graphs with bigger strings, the 80s column is not the more performing: the best result is given by the 00s column, in which there are more up days and least down days.

Count of up days in two-days string

	1970s	1980s	1990s	2000s	2010s
0	30,4624%	25,4163%	25,7326%	24,5559%	25,8028%
1	41,6763%	42,4663%	45,1626%	43,8610%	42,2592%
2	27,8613%	32,1174%	29,1048%	31,5831%	31,9381%

Table 17: Up movements in two-days string

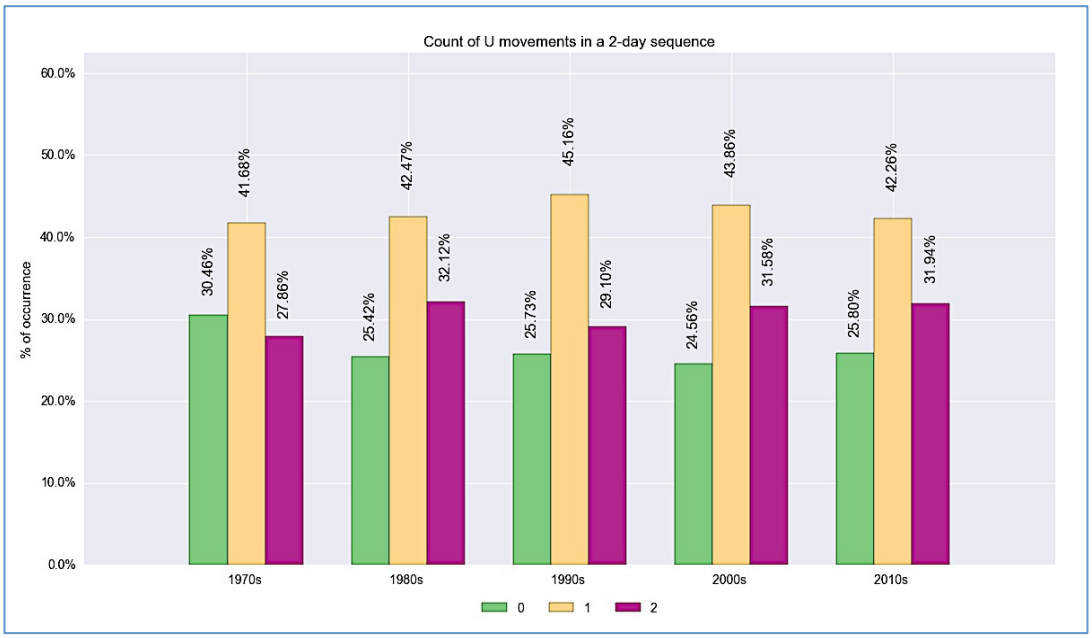


Figure 19: Percentage of up movements in two-days string

Table 17 and Figure 19 represent the percentage of positive movements in the two-days string. The predominant column in every decade is the one with just one up day over two. This result is not surprising, since, with the counting of the up days, the central column (1 up) hold two patterns (UD and DU). The 80s is the decade with the highest value of two positive days over two, while the decade with the lower percentage of no up in two consecutive days is the 00s. The 70s, instead, is the decade with the worst results: there is the higher percentage of zero positive days and the lower of one and two up days.

Count of up days in three-days string

	1970s	1980s	1990s	2000s	2010s
0	16,3678%	12,4950%	13,2530%	13,1122%	14,2857%
1	36,3794%	33,8358%	35,1004%	31,7536%	31,3827%
2	32,0416%	34,8274%	35,0201%	36,6114%	35,2266%
3	15,2111%	18,8417%	16,6265%	18,5229%	19,1050%

Table 18: Up movements in three-days string

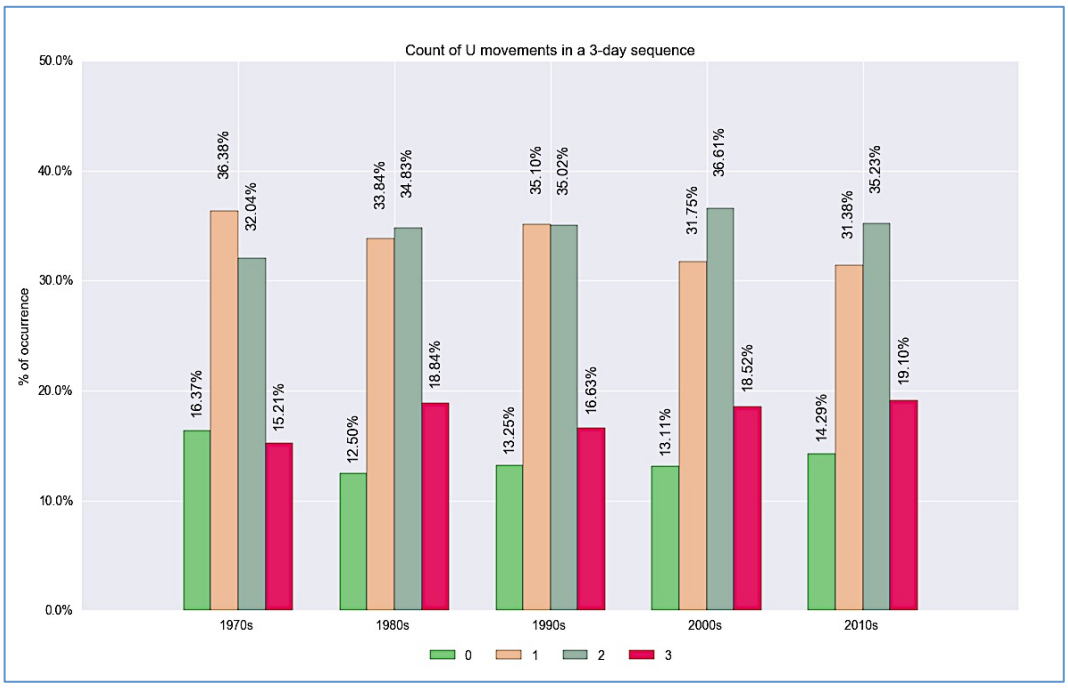


Figure 20: Percentage of up movements in three-days string

Table 18 and Figure 20 display the count of up days in three-day string. The predominant column is the one with two-up days, except for the 70s and the 90s decades, in which the predominant is the one-up day column. The three-up days column is always much higher than the one with zero up on three consecutive days, except for the 70s decade, with an increasing trend until the 80s, a little drop in the 90s and again an increase until the 10s, decade in which it exceed the peak of the three-up days column in the 80s (18.84%).

Count of up days in four-days string

	1970s	1980s	1990s	2000s	2010s
0	8,6806%	6,1508%	7,2318%	6,9143%	7,7497%
1	27,4306%	23,4127%	24,0659%	22,1256%	22,2732%
2	32,8125%	32,5397%	32,5030%	32,1217%	31,9173%
3	22,6273%	26,7460%	27,2399%	27,6571%	26,1768%
4	8,4491%	11,1508%	8,9594%	11,1814%	11,8829%

Table 19: Up movements in four-days string

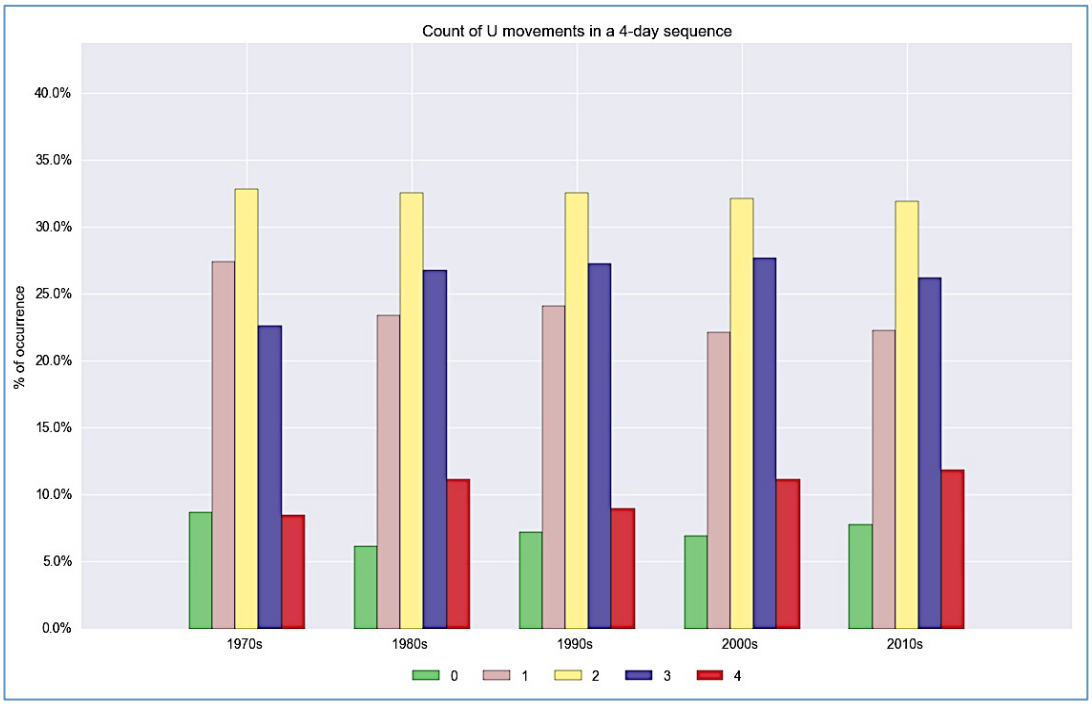


Figure 20: Percentage of up movements in four-days string

Table 19 and Figure 21 display the four-days string. The two-up days column is always the predominant one, with a quite fixed percentage; soon after there is the three-up days column, with a quite stationary presence in every decade (26%-27%), except for the 70s, in which it is much lower, with a percentage of occurrence if 22%. The one-up day column is always lower than the previous mentioned, outside the 70s, in which it is higher than the three-up days one. For what concerns the “extreme”, the

zero-up column is always the lower one, except for the 70s, as happened in the others strings. The four-up days column is quite low all the time, but with 4-5 percentage points more than the zero-up column, except for the 70s, in which they are quite even. As seen before, this last column has an increasing trend until the 80s, a decrease in the 90s, and another increase until the 10s.

Count of up days in five-days string

	1970s	1980s	1990s	2000s	2010s
0	4,4586%	3,1759%	3,7781%	3,6364%	4,6525%
1	19,2820%	15,0457%	16,1977%	14,5455%	13,9575%
2	29,7047%	27,3918%	26,8891%	26,4822%	26,1344%
3	25,9409%	27,6300%	29,2203%	28,2213%	29,1786%
4	16,2131%	19,8888%	19,0113%	20,1581%	19,0121%
5	4,4007%	6,8678%	4,9035%	6,9565%	7,0649%

Table 20: Up movements in five-days string

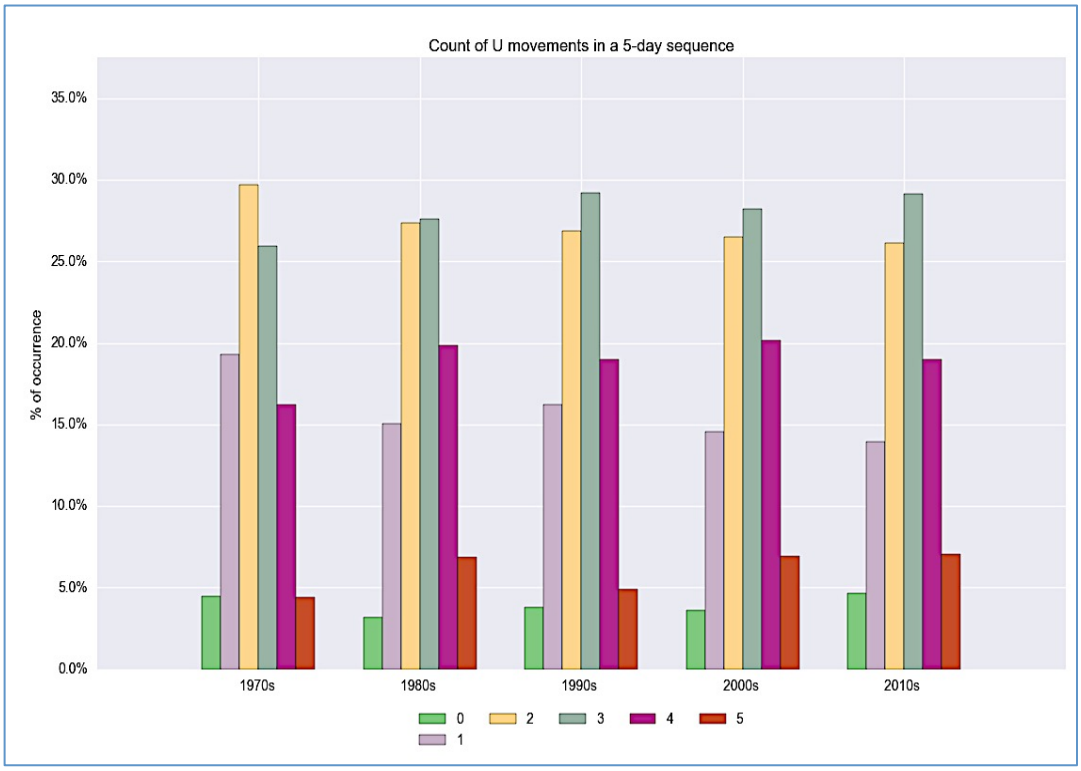


Figure 21: Percentage of up movements in five-days string

Table 20 and Figure 21 represent the count of up movements in a five-days string. Starting from the first bars of every decade, the zero-up column is always the lower one, except for the 70s. The one-up column has a quite low fluctuating behaviour, with the highest point in the 70s, while the two-up column has quite high values (about the 26% of occurrence) with the highest point in the 70s, that is also the highest column for this decade. The three-up column is the predominant pattern in every decade, with a percentage of occurrences of about the 28%, except for the 70s, decade in which this pattern has the lower value (25.9%); the four-up column, instead, has lower values but it is still quite recurrent over time, with a percentage of about 19, and the lower point in the 70s (16%). Finally, the five-up column is always higher than the zero-up one, except for the 70s; it happens about the 7% of the time in every decade, but in the 70s and in the 90s, where the percentage of occurrence is of about the 4.5.

Count of up days in six-days string

	1970s	1980s	1990s	2000s	2010s
0	2,3175%	1,7474%	1,9702%	1,9375%	2,7586%
1	12,2248%	9,2534%	10,2935%	9,2527%	9,1954%
2	24,9131%	21,1279%	20,6675%	20,2452%	18,4483%
3	27,2885%	26,0127%	28,5082%	26,1764%	27,5862%
4	20,4519%	23,3122%	22,9594%	23,1317%	24,9425%
5	10,5446%	14,1382%	12,8669%	15,2629%	12,9885%
6	2,2596%	4,4083%	2,7342%	3,9937%	4,0805%

Table 21: Up movements in six-days string

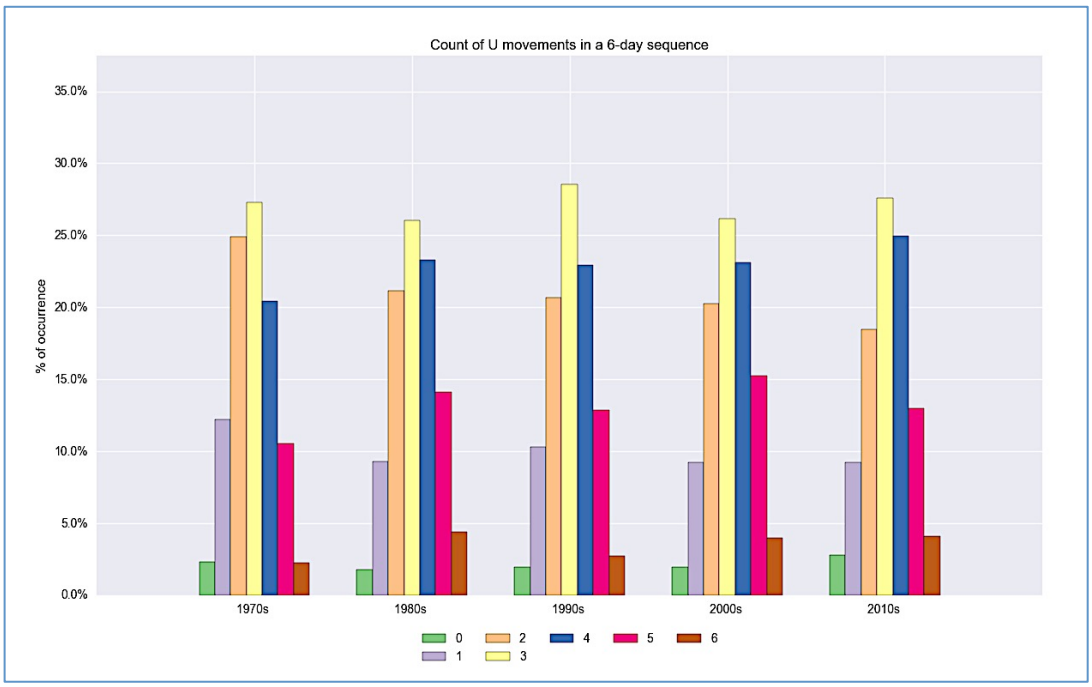


Figure 22: Percentage of up movements in six-days string

Table 21 and Figure 22 show the six-days string; starting from the left, the zero-up columns is always the lower one, but not in the 70s. The one-up column is always at the 9% level, except for the 90s, where it is at a little bit higher level (10%), and the 70s, where is much higher, with a 12% of occurrence. The two-up column has a mean of about 20%, except for the 70s, in which there is a peak (25%), and the 10s, where on the contrary there is a drop (18%). The highest column is the three-up one in every decade, with the highest point in the 90s and a mean value of occurrence of the 27%. The four-up column has an increasing trend until the 80s, drops in the 90s, and then grows again, with the highest point in the 10s (almost 25%). The five-up column has a fluctuating behaviour, with the highest values in the 80s (14%) and in the 00s (15%); the lowest point is in the 70s (10%). Finally, the six-up column has its highest points in the 80s (4.4%) and in the 00s (4%), while the lower point is in the 70s; the behaviour is still fluctuating.

Count of up days in seven-days string

	1970s	1980s	1990s	2000s	2010s
0	1,2174%	0,8343%	1,0056%	1,0285%	1,7251%
1	7,3623%	5,8403%	6,2349%	5,7358%	5,8654%
2	19,3043%	14,8192%	14,9638%	14,6756%	13,2835%
3	25,6812%	23,0433%	25,3017%	22,1915%	21,7941%
4	23,5362%	24,6325%	25,1408%	24,8418%	26,8545%
5	14,9565%	17,7592%	16,9348%	17,9984%	19,4940%
6	6,8986%	10,2106%	8,9702%	11,1946%	8,6256%
7	1,0435%	2,8605%	1,4481%	2,3339%	2,3577%

Table 22: Up movements in seven-days string

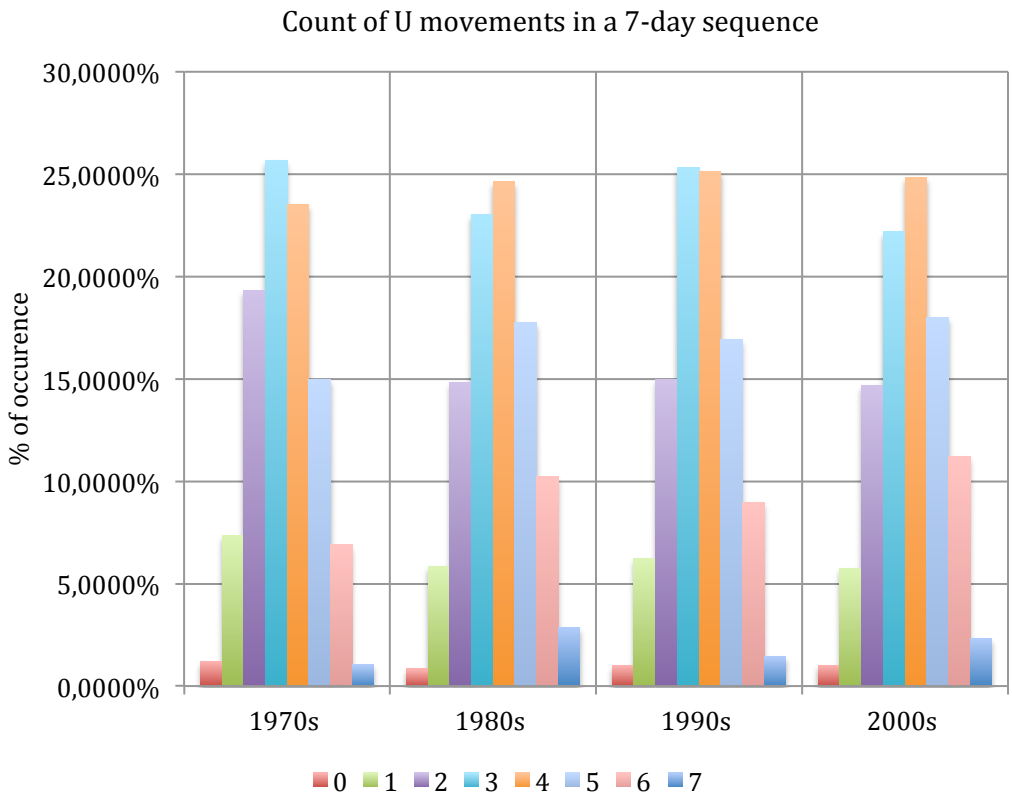


Figure 23: Percentage of up movements in seven-days string

Figure 23 and Table 22 display the seven-day movements.

The zero-up column has a fluctuating behaviour with a peak in the 10s; it is always the lower column, except for the 70s. The one-up column has a peak in the 70s, and then a fluctuating behaviour, with a mean percentage of about the 5.8; also the two-up column has a peak in the 70s, with a high percentage of the 19%, and then a quite fixed trend (about 14%). The three-up column has a fluctuating behaviour, with the higher percentages (about 25 points) in the 70s and in the 90s and the lower point in the 10s, with a value of 21.7%; the four-up column has its highest point in the 10s, with almost 27% of occurrence, and an always increasing trend, except for the 00s, in which there is a little drop. Together with the three-up columns, they are the predominant patterns. The five-up column is always increasing, except for the 90s, and has its highest point in the 10s (19.5%). The six-up column has a floating behaviour, with the highest points in the 80s (10.2%) and in the 00s (11.2%), the lower point is in the 70s, with a value of 6.8%; it is always higher than the two-up column, except for the 70s. The seven-up column has its lowest point in the 70s, followed by a peak in the 80s (2.8%); in the 90s the value is halved (1.4%), to stabilize in the following decades with a percentage of about 2.3.

Looking at the plot generated by the second method for ranking the movements of Comit Generale, it can be notice that, with the increase in the number of days in the strings, there is an always-greater detachment of the tails from the central values. In fact, data appear to follow a bell-shaped distribution, that is, the majority of values tend to concentrate around the mean, while the others weaken towards the tails.

Comparing the two methods, obviously data are more “realistic” and have immediate reading with the second one, because it is underlined

that an all-up string is not so common to have; with the first method, which analyse all the possible combinations of up and down, tails stand out with respect to all the others patterns even if the central columns enclose the same number of up days, just ranked in different combinations, for example for the two-day string Up-Down and Down-Up have both just one up day but they are represented with two different columns.

Looking at this second method, a reading of data in a historical way is given. An assumption must be done: in the decades of the 50s and the 60s, Italy experienced the reconstruction after the First World War and a period of general welfare; the data of Comit Generale start from the year 1973 and the first decade in the analysis is always the worst one. In fact, in 1973 there was the first oil crisis: OPEC[26] stopped the oil furniture toward Occidental countries, because of their support to Israel. This led to a sudden increase in oil price, which joint to the reduced transport and production activities, the increase in goods prices, the currency appreciation and the decrease of the profits, brought to an inflation mechanism. Moreover in the 1979 there was a second oil crisis, similar to the previous one, with an increase in oil price and a consequent inflation; those crises led to a weakening of stock market and a consequent fall in prices. The second decade has an always-positive trend and in fact, despite the poor economy in the first years of the decade, in those years Italy experienced a period of economic boom. The recover was intense, with a strong urbanization, more wealth in families and a lower oil price; the major companies invested in reconstructions and in new products, leading to an

[26] OPEC stands for Organization of the Petroleum Exporting Countries and it is an organisation of 13 countries, which mission is "to coordinate and unify the petroleum policies of its member countries and ensure the stabilization of oil markets, in order to secure an efficient, economic and regular supply of petroleum to consumers, a steady income to producers, and a fair return on capital for those investing in the petroleum industry" ["Our mission", in opec.org].

increase of the service industry and public services. In the 90s the situation was more negative: economic data confirm what can be seen in the plots. The GDP grew less with respect to the European mean, there were fewer workers, the average productivity was falling, salaries were lower and the public debt was huge. This decline is explained by a greater competition from countries with lower costs (imports were higher than exports) and by the introduction of Euro, in addition, on the political side there was the scandal of “Mani Pulite”, a judicial investigation about political corruption in Italy. The first decade of years 2000 seems positive, looking at the plots. Actually, the GDP grew a lot in this period, even if this fact was due to the debt and the easy credit (banks grant mortgages easily): the real estate became much more expensive, taking to the growth of the housing bubble. At the turn of the 00s and the 10s, the trade balance was negative (until 2012); the inflation was very high, with a value of about 8% and the GDP drawn back. In those years there were the economic contraction and the collapse of the industrial sector. In the last decade, the more positive together with the 80s according to the plots, there is a partial economic recovery witnessed by the behaviour of the Italian stock market. This historical interpretation of the results is given to testify the ability of the ranking method used in the process, of portraying the reality, of giving a better explanation of the results and of clearing up why some patterns have repeated over time.

3.3 Forecasting

As previously explained, the purpose of this analysis is to define a decision-making strategy to predict index prices, following two different approaches. The first one applies on future data, the past pattern strings identified before, while in the second approach a

decision tree is used, as Malliaris did in his work; moreover also Neural Networks and a Logistic Regression (see more in Appendix B and C) are employed, to see if different tools can bring to more performing results in prices forecasting; all these instruments will be explained in details in the following pages. The purpose of this section is to identify the best approach and again to check if this method is able to over perform the Random Walk Hypothesis, according to which every possible combination of events has a uniform distribution.
To do this, the sample has been divided into Training set (the first 95% of the sample) and Validation set (the last 5% of the sample):

- Training set: from 2/01/1973 to 10/09/2014;
- Validation set: from 11/09/2014 to 11/11/2016.

3.3.1 Past pattern strings applied to future data

In the first approach, the Training set is employed in identifying the dominant patterns, thanks to the ranking method used before; while the Validation set is used to check if the dominant events forecast correctly the future, and specifically if the model over performs the Random Walk Hypothesis.
Table 23 displays the occurrence of each pattern until 10/09/2014, those patterns are used to identify the most recurrent ones among the pairs that differ just in the last movement:

Count of patterns' occurrence

	Patterns	Total
1 day	D	5007
	U	**5466**
2 days	D D	**2751**
	D U	2253
	U D	2254
	U U	**3210**

3 days	D D D	**1447**
	D D U	1303
	D U D	907
	D U U	**1345**
	U D D	**1304**
	U D U	948
	U U D	1346
	U U U	**1863**
4 days	D D D D	**763**
	D D D U	683
	D D U D	555
	D D U U	**748**
	D U D D	**566**
	D U D U	340
	D U U D	570
	D U U U	**775**
	U D D D	**684**
	U D D U	620
	U D U D	352
	U D U U	**595**
	U U D D	**737**
	U U D U	608
	U U U D	776
	U U U U	**1086**
5 days	D D D D D	**405**
	D D D D U	358
	D D D U D	301
	D D D U U	**382**
	D D U D D	**351**
	D D U D U	204
	D D U U D	336
	D D U U U	**412**
	D U D D D	**290**
	D U D D U	276
	D U D U D	127
	D U D U U	**213**
	D U U D D	**326**
	D U U D U	244
	D U U U D	327
	D U U U U	**448**
	U D D D D	**358**
	U D D D U	325
	U D D U D	254
	U D D U U	**366**
	U D U D D	**215**
	U D U D U	136
	U D U U D	233
	U D U U U	**362**

5 days	U U D D D	**393**
	U U D D U	344
	U U D U D	225
	U U D U U	**382**
	U U U D D	**411**
	U U U D U	364
	U U U U D	449
	U U U U U	**636**
6 days	D D D D D D	**220**
	D D D D D U	185
	D D D D U D	158
	D D D D U U	**200**
	D D D U D D	**200**
	D D D U D U	101
	D D D U U D	173
	D D D U U U	**209**
	D D U D D D	**187**
	D D U D D U	164
	D D U D U D	74
	D D U D U U	**130**
	D D U U D D	**195**
	D D U U D U	141
	D D U U U D	170
	D D U U U U	**242**
	D U D D D D	135
	D U D D D U	**155**
	D U D D U D	108
	D U D D U U	**168**
	D U D U D D	**74**
	D U D U D U	52
	D U D U U D	74
	D U D U U U	**139**
	D U U D D D	**171**
	D U U D D U	155
	D U U D U D	97
	D U U D U U	**147**
	D U U U D D	**180**
	D U U U D U	147
	D U U U U D	182
	D U U U U U	**266**
	U D D D D D	**185**
	U D D D D U	173
	U D D D U D	143
	U D D D U U	**182**
	U D D U D D	**151**
	U D D U D U	103
	U D D U U D	163
	U D D U U U	**203**

6 days	U D U D D D	103
	U D U D D U	**112**
	U D U D U D	53
	U D U D U U	**83**
	U D U U D D	**130**
	U D U U D U	103
	U D U U U D	157
	U D U U U U	**205**
	U U D D D D	**223**
	U U D D D U	169
	U U D D U D	146
	U U D D U U	**198**
	U U D U D D	**141**
	U U D U D U	84
	U U D U U D	159
	U U D U U U	**223**
	U U U D D D	**222**
	U U U D D U	189
	U U U D U D	128
	U U U D U U	**235**
	U U U U D D	**231**
	U U U U D U	217
	U U U U U D	266
	U U U U U U	**369**
7 days	D D D D D D D	**119**
	D D D D D D U	101
	D D D D D U D	82
	D D D D D U U	**103**
	D D D D U D D	**106**
	D D D D U D U	52
	D D D D U U D	82
	D D D D U U U	**118**
	D D D U D D D	**105**
	D D D U D D U	95
	D D D U D U D	36
	D D D U D U U	**65**
	D D D U U D D	**100**
	D D D U U D U	73
	D D D U U U D	79
	D D D U U U U	**130**
	D D U D D D D	86
	D D U D D D U	**101**
	D D U D D U D	64
	D D U D D U U	**100**
	D D U D U D D	**47**
	D D U D U D U	27
	D D U D U U D	48
	D D U D U U U	**82**

7 days	D D U U D D D	**98**
	D D U U D D U	97
	D D U U D U D	54
	D D U U D U U	**87**
	D D U U U D D	**96**
	D D U U U D U	74
	D D U U U U D	104
	D D U U U U U	**138**
	D U D D D D D	66
	D U D D D D U	**69**
	D U D D D U D	72
	D U D D D U U	**83**
	D U D D U D D	**57**
	D U D D U D U	51
	D U D D U U D	73
	D U D D U U U	**95**
	D U D U D D D	36
	D U D U D D U	**38**
	D U D U D U D	19
	D U D U D U U	**33**
	D U D U U D D	**40**
	D U D U U D U	34
	D U D U U U D	67
	D U D U U U U	**72**
	D U U D D D D	**95**
	D U U D D D U	76
	D U U D D U D	56
	D U U D D U U	**99**
	D U U D U D D	**58**
	D U U D U D U	39
	D U U D U U D	66
	D U U D U U U	**81**
	D U U U D D D	**102**
	D U U U D D U	78
	D U U U D U D	57
	D U U U D U U	**89**
	D U U U U D D	**99**
	D U U U U D U	83
	D U U U U U D	111
	D U U U U U U	**154**
	U D D D D D D	**101**
	U D D D D D U	84
	U D D D D U D	76
	U D D D D U U	**97**
	U D D D U D D	**94**
	U D D D U D U	49
	U D D D U U D	91
	U D D D U U U	**91**

7 days	U D D U D D D	**82**
	U D D U D D U	69
	U D D U D U D	38
	U D D U D U U	**65**
	U D D U U D D	**95**
	U D D U U D U	68
	U D D U U U D	91
	U D D U U U U	**112**
	U D U D D D D	49
	U D U D D D U	**54**
	U D U D D U D	44
	U D U D D U U	**68**
	U D U D U D D	**27**
	U D U D U D U	25
	U D U D U U D	26
	U D U D U U U	**57**
	U D U U D D D	**73**
	U D U U D D U	57
	U D U U D U D	43
	U D U U D U U	**60**
	U D U U U D D	**84**
	U D U U U D U	73
	U D U U U U D	78
	U D U U U U U	**127**
	U U D D D D D	**119**
	U U D D D D U	104
	U U D D D U D	71
	U U D D D U U	**98**
	U U D D U D D	**94**
	U U D D U D U	52
	U U D D U U D	90
	U U D D U U U	**108**
	U U D U D D D	67
	U U D U D D U	**74**
	U U D U D U D	34
	U U D U D U U	**50**
	U U D U U D D	**90**
	U U D U U D U	69
	U U D U U U D	90
	U U D U U U U	**133**
	U U U D D D D	**128**
	U U U D D D U	93
	U U U D D U D	90
	U U U D D U U	**99**
	U U U D U D D	**83**
	U U U D U D U	45
	U U U D U U D	93
	U U U D U U U	**142**

7 days	U U U U D D D	**120**
	U U U U D D U	111
	U U U U D U D	71
	U U U U D U U	**146**
	U U U U U D D	132
	U U U U U D U	**133**
	U U U U U U D	155
	U U U U U U U	**214**

Table 23: Count of occurrence of the up and down patterns

Table 23 shows all the different combinations of up and down for the patterns, until seven days. To select the dominant ones that will be used in the validation set, the up and down combinations are analysed couple by couple; specifically, each couple is identical, except for the last day, which can be an up or a down: the most recurrent between the two is chosen, so this method allows reducing the number of combinations, keeping just the ones that occur more often.

For example, to predict the first day, the most recurrent pattern between up and down will be selected, looking at the first two rows of the table: in the whole period an up day occurred 5,466 times, while a down day occurred 5,007 times. The forecast of one day will be always an up, since it dominates down; it is recalled that, in the forecasting of day 1, there is not information about the previously days. For the two-day forecasts, there are two possibilities to consider: a day beginning with up or a day beginning with down and each of them can end with an up or a down. Looking at Table 23, a Down-Down happened 2,751 times, while a Down-Up happened 2,253 times: since the most recurrent has to be selected, if the pattern starts with a down day, the training set will predict another down day. Similarly, the Up-Down pattern happened 2,254 times, while the Up-Up pattern occurred 3,210 times: after an up day, the validation set

will predict another up day. The results of this approach, the same used by Malliaris in his work, are highlighted in Table 23 in bold and are applied to the validation set. For example, if in the validation set a three-day string is known, e.g. UDU, to forecast the fourth one it is necessary to look at the training set.

Table 24 explains this reasoning:

Forecast for the fourth day in the validation set, using the training set

Training set 4-day pattern	Training set count	Validation set 3-day pattern	Validation set forecast for the 4th day
UDUD	353	UDU	U
UDUU	**595**	UDU	U

Table 24: Pattern example for forecasting the fourth day

The UDUU pattern occurs over 200 times more than UDUD, and so the prediction for the forth day in the validation set will always be an up day, as it can be notice in the right side of Table 24.

To summarize, for every couple of string that differs just in the last day, the most recurrent pattern in the training set has been selected as most likely behaviour for the validation set. The dominant patterns are now applied to the whole validation set and all the correct predictions are counted, to understand which pattern works better (one day information or rather more days of information), moreover a random forecast, obviously between up and down, has been generated, in order to check if the model outperforms the Random Walk Hypothesis.

Percentage of correct forecasts on the validation set

Number of days of information	0	1	2	3	4	5	6	Random
Prediction for the -th day	1	2	3	4	5	6	7	Random
% of correct forecasts	53,080%	56,175%	56,774%	56,775%	56,665%	57,174%	57,064%	49,569%

Table 25: percentage of times the validation set forecast was correct with the 1-7 day patterns

Table 25 displays the correct forecasts on the validation set expressed in percentage; the first row of the table expresses how many days are known in order to predict the following one, while the second row represents the day that will be predicted, so the next one. Specifically, if the objective is to predict the following day, without looking at the past, the number of days of information will be equal to 0, if just one day (yesterday) is known, the predicted day will be the number two of the string and so on. In the third row is displayed the value of correct forecasts and, contrary to Malliaris' results, with the application on the Comit Generale, the best predictions are obtained knowing the past five and six days. The random prediction is the least effective, with a percentage of correct forecasts of the 49.57%, confirming that this model over performs the Random Walk Hypothesis. Contrary to Malliaris' work, the forecast of the following day without knowing the previous, is not successful, with a percentage of 53.1%; the forecasts with one, two, three and four days of information have similar percentages of occurrence, almost close to the five and six days. This approach gives importance to past information and states that in order to have a forecast as precise as possible, many days of information are needed. Even if the Random Walk Hypothesis is out-dated, the percentage of success is not so high; this is possibly due to the overfitting problem, according to

which, if the ratio between the model complexity and the training set size is too high, the model has poor predictive performance in the validation set. To try to exceed this issue, different approaches are tested in the following pages.

3.3.2 Decision Tree

In the last section of the forecasting analysis, three different tools have been used in trying to overcome the overfitting problem found with the previous methodology. The first one is the same used by Malliaris in his work: a Decision Tree.

A decision tree is a predictive model, built in order to represent sequentially decisions, according to the occurrence of certain conditions.
The root is the starting point of the decision tree, where the first input is given; then there could be an interior node (with another input) or the leaves, which represent the different class labels or rather the possible outcomes deriving from the interior node input; the conjunctions between root and leaves and then between leaf and leaf are called branches and represent the rule of classification and so the connection of features that lead to those class labels. In other words, after an input there could be another input (so another condition that has to be satisfied) or an outcome. The objective is to split the data set in classes as pure as possible that comply with the previous input, thanks to the help of different algorithms.
In this case, the algorithm C5.0 is used: for every presented input it generates a single prevision, splitting the data set in the subsample that provides the maximum information gain, and so the attribute with the highest importance according to the given input. Every subsample

(leaf) is further divided until possible, and then the lowest subdivisions are reviewed, in order to cut off the not significant ones for the model performance; a more detailed explanation is provided in Appendix B. To measure the performance of a decision tree, the most used metrics are its accuracy, its precision, its recall and its F1-score. These metrics must be evaluated on the Validation Set, in order to test the ability of the model to generalize its predictive power, and they can be computed thanks to the Confusion Matrix, displayed in Figure 24:

Confusion Matrix

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Figure 24: Example of confusion matrix. Source: http://aimotion.blogspot.it/2010/08/tools-for-machine-learning-performance.html

The matrix is a table in which each row represents the actual value and each column represents the predicted value (or viceversa).
If the predicted value corresponds to the actual one, the model has performed well, on the contrary if the prediction is wrong the model is incorrect, causing either a Type I error or a Type II one. The four indicators of the matrix allow evaluating the algorithm performance:

- TP (True Positive): correctly predicted events;
- FP (False Positive): incorrectly predicted events (type I error);
- TN (True Negative): correctly predicted no-event;

- FN (False Negative): incorrectly predicted no-event (type II error);

If the classification uses *n* classes, the Confusion Matrix will have dimension *nxn*. The four parameters obtained with this matrix allow a better understanding of the model performance.
The *Accuracy* is expressed by the formula:

$$a = \frac{TP + TN}{TP + TN + FP + FN}.$$

It is the metric used to summarize the Confusion Matrix, representing how close the predicted outcomes are to the actual values, given the samples.
The *Recall* measures the fraction of positive records correctly classified:

$$r = \frac{TP}{TP + FN}.$$

If the resulting value is high, it means that few records of the negative class have been wrongly classified as positive.
The *Precision* measures the fraction of real positive records among the ones characterized as such:

$$p = \frac{TP}{TP + FP}.$$

A high value means that few records of the positive class have been wrongly classified as negative.

Finally the *F1-score* is the harmonic mean between precision and recall:

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}.$$

If the F1-score is high, it means that precision and recall are high as well, and so there are not so much false negative and false positive.

To build the decision tree, three sets of variables have been used: the up-down strings from one day to seven days, the count of up days from one day to seven and the today closing value, in order to check the prediction ability of such patterns analysed in the previous section.
To be more precise, in the first set of features there where dummy variables (a feature that can assume as values 0 or 1), which represented every possible combination of outcomes from one to seven days in the past. For example, among this set of features there were these four: Up-Up, Up-Down, Down-Up and Down-Down. For every sample, every one could be either 0 or 1, depending on what happened in the two days before that sample: if in the two days before a given sample the market went up, the independent variable Up-Up was 1 (True), while all the others were 0 (False). This, for every K from 1 to 7 days.[27]
The count of up days from one to seven days in the past is just a count of how many times, in the past K days from 1 to 7, the market outperformed the previous closing price.
Figure 25 displays the decision tree performance with past days of information from one to seven, evaluated with the parameters

[27] It could be noted that this creates seven different sparse matrices within the input dataset; from an information point of view, these are actually just seven independent variables.

explained in the previous page; the results are sorted in order of accuracy (from the least precise to the most one).

Decision Tree performance

	accuracy	F1	precision	recall
t_3	0.46558	0.458716	0.473485	0.44484
t_1	0.485507	0.513699	0.479233	0.553506
t_4	0.507246	0.507246	0.536398	0.4811
t_5	0.511797	0.537005	0.513158	0.563177
t_2	0.525362	0.55137	0.566901	0.536667
t_7	0.540835	0.525328	0.557769	0.496454
t_6	0.548094	0.569948	0.574913	0.565068

Figure 25: Decision Tree results in terms of accuracy, precision, recall and F1-score, evaluated on the Cross Validation set

The highest accuracy is obtained with six days of information, with a value of 54.8%: it means that the model forecasted the up and the down movement correctly around 55% of the time, and it did it wrongly about the other 45% of the time. Right after, seven days of information give a similar performance (54.1%) and at the third place, two days of information give a correct forecast the 52.55% of the time. Thus, the model works well with many days of past information (six and seven), but also with just two days of past data.

For what concerns precision, that is the measure of result relevancy, the more it is high, the low the false positive rate is. The higher result is obtained again with six days of past information (57.5%), followed by two days (56.7%) and seven days of information (55.8%).

Recall reflects how many truly relevant results are returned, and so if it is high there will be a low false negative rate. The best result is still

with six days of past information (56.5%), followed by five days of information with a performance of 56.3% and one day (55.3%).
Finally, the F1 score should be as high as possible, since it reflects the absence of false negative and false positive. Six days of past information return again the best value, with a percentage of 57, right after there are two days of information (55.1%) and five days (53.7%).
According to those results, the best performance for the model is always given by six days of past information, even if it works well also with two days of past information. The precision of the model is similar to the one of Malliaris' work, but the percentages are not so high: this is due to an overfitting problem, since the model uses too many variables and so the parameterization is excessive, the predictive ability on the validation set drops. To try to overcome this issue, two different approaches are tested in the following paragraph: a logistic regression and some neural networks.

3.3.3 Logistic Regression and Neural Networks

The additional tools used in this third part of the forecasting process are logistic regressions and artificial neural networks, explained in Appendix B and C respectively.
Both the logistic regression and the neural network models are supervised classification algorithms for data forecast. They lie in the realm of supervised learning because they try to predict outcomes knowing already the outcomes in the training set (opposed to unsupervised learning, where data is not labelled, even in the training set). They are classification algorithms because they try to predict discrete outcomes (limited number of outcomes), opposed to regression models that try to predict continuous (real) dependent variables.

A logistic regression is a discrete choice model, which describes, predicts and explains a dichotomous choice (a choice between two values). Specifically, it explains the relationship between a set of explanatory features (x), and an outcome that is dichotomous (y), in order to predict the probability that such an outcome happens, given a new series of explanatory variables. The probabilities are estimated using a cumulative logistic distribution[28] as loss function. The logistic regression produces a logistic curve with two possible values as an outcome, and for this reason it is specific for this issue, since the tomorrow's price has two possible outcomes.

Another instrument that can be used for this study is an artificial neural network (ANN), a detailed explanation of which is provided in Appendix B.

ANNs are systems of artificial intelligence able to mime a human brain. In these networks the input layers plug the information into the system, then thanks to the weighted connections that contain the learning process, these information are used to test different scenarios in a parallel fashion in one or more hidden layers; finally the hidden layers fire the output layer correspondent to the more probable answer. These systems are defined self-learning because, at each iteration, they adjust the connection weights in order to get closer to the desired output, so the more they are trained, the more they will get closer to the final output. After some iteration, they converge to the minima of the loss function used to track the model performance, and they are ready to be tested on the validation set.

These two approaches have been tested with different independent variables (e.g. the closing price, the up-down patterns, the count of up days in strings from one day to seven) and also compared with the

[28] A cumulative logistic distribution is a continuous probability distribution, resembling the normal one but with heavier tails.

decision tree, as shown in the following pages. It will be displayed that some features are useless in this model, according to their correlation coefficient with the independent variable.

To start the analysis, the weight of every feature has been tested using a neural network, in order to understand if some variables are useless for the prediction and can be deleted, making the model simpler.

Figure 26 shows how the model changes its prediction ability with the exclusion of specific patterns; the results are displayed in order of accuracy, from the lowest to the highest.

Features ability for prediction

	accuracy	F1	precision	recall
only_value	0.471869	0.532905	0.470255	0.614815
all	0.53539	0.584416	0.54878	0.625
no_ups	0.540835	0.587276	0.560748	0.616438
no_updowns	0.548094	0.58843	0.56869	0.609589
only_ups	0.548094	0.584307	0.555556	0.616197
only_updowns	0.571688	0.610561	0.560606	0.67029
no_value	0.578947	0.61204	0.60396	0.620339

Figure 26: Features' prediction valuated with accuracy, F1, precision and recall

The neural network with just the closing price as input layer is the one with the lowest performance: this can be proved also by looking at the correlation coefficient of this feature with the dependent variable (the prediction ability of the model), that is 3.363473%. The best performance is obtained with just the up and down values from one day to seven, and with the whole variables (also the up count from one day to seven), except for the closing value. The plot in Figure 27 represents the different performances of a neural network according

to the number of input layers used to build it, and it confirms those results, underlying that too many variables take to an overfitting problem.

Anova criteria for feature selection

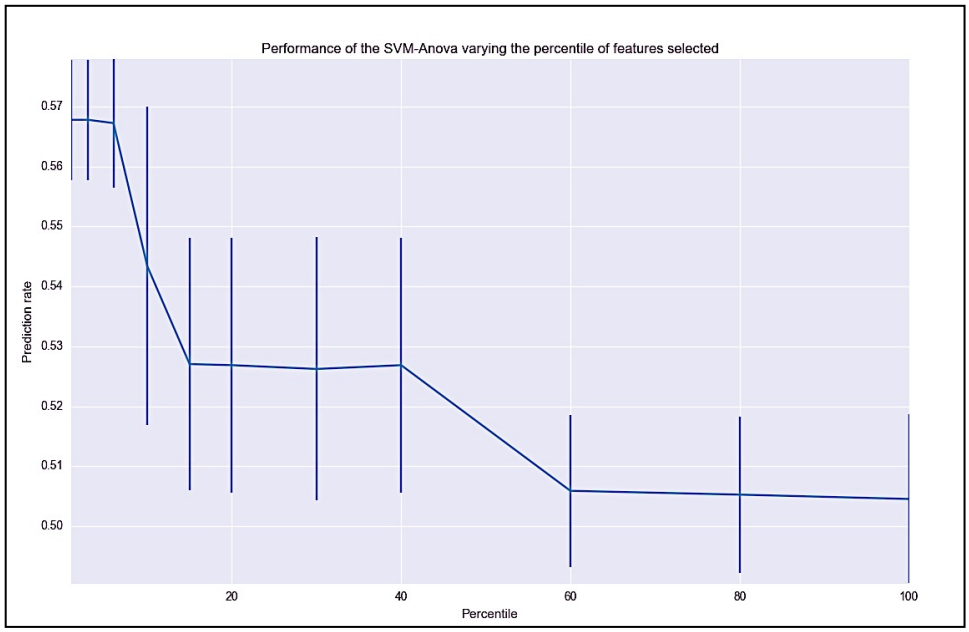


Figure 27: performance of the model with selected features

In fact, Figure 27 represents an ANOVA test for variables selection: it is used to understand if a huge number of input variables are necessary. ANOVA stands for Analysis of Variance, and it is a useful tool for the comparison of the variables in terms of statistical significance. On the y-axis is indicated the prediction rate, which corresponds to the percentage of success of the model in forecasting market movements, while on the x-axis there are the percentiles, which indicate the percentage of selected variables. Specifically, this test orders the features according to their importance for the model, so from the one that bring more information to the one that bring less information. The best performance is on the left side of the plot with the use of few inputs, since the prediction rate of the neural network is the highest, with a value of almost 57%. This plot confirms that too

many variables lead to a poor predictive model; for example with a percentile of 80, so with the use of the majority of input variables among closing price, the up counts and the up-down patterns, the prediction ability of the model is just 51%. As said before, too many variables lead to great performance in the training set, but in the validation set there will be for sure overfitting, since the model will be too specific.

Afterwards the prediction ability of features has been analysed, the forecasting results with the neural network are displayed in Figure 28.

Neural Network performance

```
>>> results
     accuracy        F1 precision    recall
t_5  0.524501  0.688095  0.524501         1
t_6   0.53176  0.525735  0.588477  0.475083
t_4  0.543478  0.584158  0.582237  0.586093
t_7  0.549909  0.586667  0.571429   0.60274
t_1  0.563406  0.586621  0.624088  0.553398
t_3  0.570652  0.617124  0.618123  0.616129
t_2  0.577899  0.598967  0.614841  0.583893
```

Figure 28: Neural Network performance from one to seven days

Figure 28 shows the forecast performance of a MPL Classifier (Multilayer Perceptron Classifier) with a set of features from 1 to 7 days. An MPL Classifier is a classification class used to perform the test that implements the Feedforward ANNs with multiple layers, all fully connected. The performance is sorted in order of accuracy, from the lowest to the highest. Contrary to the results with the decision tree, the prediction with two days is the most efficient, because it has the highest accuracy, and ones of the higher F1 score, precision and recall. The network with five days of information has the lowest

precision and accuracy (a sign of overfitting), but a great performance with the F1 score and the recall. The results are again analogues to those of Malliaris: the network works well both with few days of past information (just two) and with more history (five days), even if with five days of past information it has an higher overfitting risk.

Now, in Figure 29 the performance of a logistic regression and different neural networks has been compared.

Logistic regression and neural networks performance

	accuracy	f1	gmean
LogisticRegression-1K - only_ups_3	0.637681	0.649123	0.643377
NN_(5,2)-1e-3 - all_t_4	0.567029	0.723699	0.640592
NN_(5,2)-1.0 - no_ups_1	0.615942	0.659164	0.637187
NN_(25,2)-1.0 - no_updowns_6	0.557169	0.715618	0.631443
NN_(5,2)-1e-3 - only_ups_3	0.59058	0.664688	0.626539

Figure 29: Performance comparison of model with logistic regression and neural networks.

Figure 29 displays the comparison among a logistic regression and four different neural networks with respect to hyperparametrization (that is, with different number of hidden layers, hidden nodes and learning coefficients). The results are shown in order of geometric mean[29] (computed between accuracy and F1 score, two parameters that allow the evaluate the algorithm performance), from the highest to the lowest. The higher the g-mean, the higher the relationship between accuracy and F1 score.

[29] The geometric mean is a tool used in statistic and social sciences to compare features with different properties. It is the n-th root product of n-number, for example with two parameters *a* and *b*, the g-mean will be: $g - mean = \sqrt[n]{a * b}, n = 2$.

The first model is a logistic regression with a parameter C=1,000, the parameter requested by the function in order to reduce the overfitting risk, and just the count of up days until three days: this is the model with the highest accuracy and geometric mean, so the closer model to the actual results. The proposed neural networks are of two kinds: three are built with two hidden layers, each of them with 5 hidden nodes, and one with two hidden layers, each of them with 25 hidden nodes, all of them with an alpha equal to about 0.001, another requested parameter used to reduce the overfitting risk. The other difference among those networks are the selected features: in the first model there is every input until four days of past information, in the second all the features until one day, except for the up, in the third model there are all the features until six days, except for the up-down movements and the fourth has just the up count until three days of information. Those distinctions have been done in order to understand which model and which features perform better in the prediction of market movements. The first neural network has a great geometric mean and the highest F1 score among all the models, and so, with the logistic regression, is the best approach to the forecasting issue.
Finally, a comparison between the decision tree and those last models is proposed in Figure 30:

Model comparison

	accuracy	F1	precision	recall	train_score
DecisionTree	0.508167	0.53833	0.533784	0.542955	0.999904
LogisticRegression	0.53539	0.581699	0.554517	0.611684	0.578962
NN (25, 2) - 0.1	0.54265	0.620482	0.552279	0.707904	0.587752
NN (5, 2)	0.513612	0.539519	0.539519	0.539519	0.574759

Figure 30: model comparison according to accuracy, F1 score, precision and recall

Figure 30 represents the comparison among four different models: the decision tree, the logistic regression, and two different neural networks: one with 25 hidden nodes per hidden layer and the other with 5 hidden nodes per hidden layer, both of them work with the whole features, so with closing price, the up count from one day to seven and the up-down patterns. The decision tree has the lowest accuracy; moreover its training score is very high. The lowest training score is obtained with the last neural network, even if it has also a low accuracy; precision and recall (and so also the F1 score) are close to those of the decision tree. The best performance is obtained with the logistic regression and the neural network with two hidden layers of 25 hidden nodes each.

This final comparison demonstrates that these features (the up-down count, the up count and the closing value) can be employed with better results using logistic regressions and neural networks; moreover, it has been proved that too many variables lead to an overfitting problem, making the training set prediction easier, but more difficult for the forecasting of market movements in the validation set.

The outcome of these experiments is that past up and down movements allow the forecast for the next day, with a percentage of success higher than the one of the Random Walk Hypothesis, even if a better performance is obtained with just three days of information: it is recalled that the percentage of success is about the 63% for the logistic regression with just the count of up until three days of information. Therefore, it is useless to use too many variables and to make the model too much complex and detailed.

To conclude, this method is applied also to a company in paragraph 3.4, and the new tools are tested on the same sample Malliaris used in paragraph 3.5.

3.4 Application to the American Express Company

The following step of this analysis is the application on a public company. The American Express Company (also called Amex) was founded in 1850 as express mail business. Now it is a multinational financial services corporation, based in New York, and it is one of the 30 components of the Down Jones Industrial Average (USA stock market index created by the Wall Street Journal).
For the forecasting of market movements, data from the 1st June 1972 to the 30th December 2016 have been taken, for a total of 11,247 daily observations; the validation set corresponds to the last 5% of data, so from the 14th August 2014 to the 30th December 2016.
From this experiment a lower forecasting ability can be expected, since a public company has generally greater volatility with respect to a stock market index and prices are affected from many factors, so the prediction of future prices is more difficult.

For this test, just the forecasting part is shown, starting from the application of the past strings to future data, displayed in Table 26; it is recalled that in this step is used the count of the whole movements from one day to seven, with every possible combinations of up and down.

Percentage of correct forecasts on the validation set

Number of days of information	0	1	2	3	4	5	6	Random
Prediction for the -th day	1	2	3	4	5	6	7	Random
% of correct forecasts	52,413%	52,905%	52,914%	53,118%	53,501%	53,785%	54,221%	50,138%

Table 26: Percentage of times the validation set forecast was correct with the 1-7 day patterns

Table 26 displays the percentage of correct forecast for the number of past days of information indicated in the first row. Even if the predictions are made on a public company, this method still exceeds the Random Walk Hypothesis, which has success the 50% of the time.

As happened with the Comit Generale, this approach works better with six and five days of past information, while it is less performing for the prediction of tomorrow without knowing the past. The final results are the same, but the percentages of success are a little smaller than the ones obtained with the Comit Generale, as expected at the beginning of the experiment.

To conclude, also logistic regressions and neural networks have been tested with the public company data and compared with the decision tree performance. Table 27 shows the performance of these different models in predicting market movements of the American Express Company, applying the results of the training set to the validation set.

Model comparison for the forecast of American Express Company prices

	past days=6		past days=3	
	Accuracy mean	Accuracy s.d.	Accuracy mean	Accuracy s.d.
Decision Tree	54,778%	3,111%	56,634%	3,763%
Logistic Regression	56,181%	4,124%	56,207%	4,142%
LogisticRegression - 1K	56,192%	4,125%	56,217%	4,150%
NN (25, 2) - 1e-3	54,030%	3,306%	53,913%	2,534%
NN (25, 2) - 1.0	52,838%	1,730%	52,802%	1,913%
NN (5, 2) - 1e-3	52,849%	1,732%	51,973%	2,834%
NN (5, 2) - 1.0	52,818%	1,711%	51,984%	2,840%

Table 27: Decision Tree, Logistic Regression and neural networks, classified according to accuracy mean and standard deviation

The different models are tested with 6 days of past information and then with 3, to investigate the more performing pattern (few days or a lot of); their results are displayed according to the accuracy mean and

standard deviation. The accuracy has been evaluated with the k-fold cross validation method (here *k*=10): in each iteration, the original sample is divided into *k* subsamples of the same dimension; one of them is randomly kept as validation set, while all the others form the training set. This method for dividing training and validation sets is applied for every tool displayed in Table 27 (decision tree, logistic regressions and neural networks). The iterations happen *k* times, so that every subsample is used as validation set once, giving *k* different accuracies. To produce a single final estimation, for all the tools accuracies' mean and standard deviation have been computed.

With six days of past information, the most performing model is the logistic regression with the C=1,000, with a success of 56.19%; even if neural networks do not give good forecasts, the model performance with the public company data is analogue to the one of the Comit Generale.

With three days of past information, the best result is given by the decision tree, with correct forecasts the 56.63% of the time, followed by the logistic regressions. This is a surprising result because the decision tree has always been the least effective model.

It has been proved that this approach is able in overperforming the Random Walk Hypothesis, both with a stock market index and with a public company stock; of course, with the second kind of data, the results are less satisfactory. Moreover, with Comit Generale, a good way to reduce overfitting is the use of neural networks and most of all logistic regression; further analysis should focus on the best features for these models, in order to reduce again overfitting and improve the predictive ability.

3.5 Different tools applied to the S&P500

Finally, the new tools, which have been introduced in the previous paragraphs, are applied to the same sample of the S&P500 used by Malliaris, to check if the results can be improved. It is recalled that the sample goes from 03/01/1950 to 19/07/2011, with training and validation sets correspondent respectively to the first 95% of the sample and the last 5%. Table 28 displays the results of such analysis in terms of accuracy:

S&P500 performances in forecasting market movements

	Accuracy					
	past days=1	past days=2	past days=3	past days=4	past days=5	past days=6
DecisionTree	55,7195%	54,2820%	54,7960%	55,2129%	55,3135%	54,7545%
LogisticRegression - 1K	64,1325%	64,1300%	64,1274%	64,1250%	64,1230%	64,1214%
NN (5, 2) - 1.0	51,3795%	51,6534%	49,7578%	51,8085%	51,6439%	54,7351%
NN (25, 2) - 1.0	51,2366%	53,1456%	52,9744%	51,6471%	53,0974%	49,9806%

Table 28: S&P500 predictive ability for forecasting market movements with the use of different tools

The tested models are a decision tree, a logistic regression, a neural network with 2 hidden layers and 5 hidden nodes each, and another neural network with 2 hidden layers and 25 hidden nodes each; all of them have been tested with different days of past information, from 1 to 6.

For what concerns the decision tree, the performance is similar to that of Malliaris: the best results are obtained with 1, 4 and 5 days of past information, giving correct forecasts about the 55% of the time; on average the decision tree predictive ability is correct just over half of the time, outperforming the neural networks. Specifically, the first neural network, with 2 hidden layers of 5 hidden nodes each, gives

the best result (analogous to that of the decision tree) with 6 days of past information (54.7%); on average, this tool makes correct predictions about the 51% of the time, with the worst performance with 3 days of past information (49.7%). The second neural network, composed by 2 hidden layers of 25 hidden nodes each, has the best performance with 2 days (53.1%) and 5 days (53.09%) of past information, while the worst result is with 6 days of past information (49.9%). On average, the prediction ability in forecasting market movements of this neural network is about the 52%.

Finally, the logistic regression gives the best results for all the days, exceeding about 10 percentage points the other models; it correctly forecasts up and down days the 64% of the time. By the way, the performance decreases a little with the increase in the number of past days of information.

This test confirms the results obtained with the application on the Comit Generale: the more performing model is a logistic regression and, in addition, too many days of past information lead to a worsening of forecasting.

CONCLUSIONS

With this thesis a different method for forecasting market movements has been explained and analysed.

In the first chapter, many already famous theories in prices prediction are demonstrated to be not effective all the time.

To overcome this problem, in the second chapter Malliaris' method is presented: he used S&P500 index daily observations of 61 years, to analyse up and down movements and check if some specific patterns repeat across time and so can predict future movements; to test the model, the data set is split in training and validation sets. The closing prices are collected in two ways: first, as up and down strings, from one day to seven; secondly, as number of up movements in the same strings as before. These past patterns have been used in forecasting future movements with two methods: the application of the simple past patterns to future data, and a decision tree, which uses as inputs the up-down patterns from one day to seven, the up movements until five days and the closing price of today. Malliaris' work gives positive results, proving that prices do not move according to the Random Walk Hypothesis. Specifically, using past patterns, he obtained a correct forecast the 56.3% of the time with 5 days of past information and the 56.6% of the time with no days of past information; with the decision tree the results improve: it gives the correct forecast for the following day the 57% of the time.

In the third chapter, the same approach is tested on the Italian index Comit Generale, with the addition of new tools (logistic regressions and neural networks) to try to improve the performances obtained by Malliaris.

The first approach is the application of past patterns to future data: it gives positive results using five days (57.2% of correct forecasts) and six days (57.1% percentage of correct forecasts) of past information (so in generating the prediction for the 6^{th} and 7^{th} day respectively), overperforming the random prediction, which generates a correct forecast the 49.6% of the time.

Afterwards, the decision tree is applied: it gives the best result in the forecasting of the 6^{th} day (using five days of past information) with accuracy (the higher it is, the higher the closeness of the predicted values to the real ones) of the 55% and precision (the closeness of the different prediction to each other, so it can be seen as a measurement of the result consistency) of the 57%, followed by the forecast of the 7^{th} day (accuracy of 54% and precision of 56%), with results similar to the first method.

After these applications, logistic regressions and neural networks are used, to improve the already obtained results; but before an investigation on patterns significance has been done, proving that too many variables make the models too complex and specific for the training set, making the prediction on the validation set less performing.

In the first test, the neural network that uses all the information (closing price of today, up-down patterns from one day to seven, up count from one day to seven) gives the better performance in the prediction of the 2^{nd} day of the string (just the first one is known), with accuracy of 57.8% and precision of 61%, and then in the prediction of the 3^{rd} day (accuracy of 57.1% and precision of 61.8%).

The second test is done with a logistic regression and four neural networks; the difference among them is that they use just some features. The higher accuracy (63.8%) is obtained with a logistic

regression that uses just the up-count until day 3, testifying that a lot of variables are not necessary.

Finally, the two new models and also the decision tree have been compared, using all the features; the best performance is obtained with a neural network of 2 hidden layers, each of them with 25 hidden nodes, with accuracy of 54.3%.

Even if a lower performance can be expected from the application of this method on a company index because of its higher volatility, the results obtained with the American Express Company closing prices exceed a random prediction: the application of past patterns to future data leads to correct forecasts the 54.2% of the time with 6 days of past information and the 53.7% of the time with 5 days. The other three tools give even better results: with 6 days of past information, the logistic regression performs better (56.2% of correct forecast), while with 3 days of past information the best outcome is obtained with the decision tree (56.6%), followed by the logistic regression (56.2%).

Finally, the logistic regression and the neural networks have been applied on the same dataset used by Malliaris, to see if also in this case they generate better performances. The logistic regression overperforms the decision tree of about 10 percentage points, with correct forecasts about the 64% of the time with the use of past days from one to six.

This work has proved that prices do not move randomly, but an adequate study on their past features can improve the forecasting ability. Further studies should focus on the selection of appropriate features, since it has been demonstrate that some of them are useless, for example the correlation coefficient between the closing price and the model performance in predicting the correct movement is very low.

APPENDIX A

Python codes that represent the framework of the work:

```
from utils import *
from ml import *
from sklearn import tree, metrics
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier

def data_analysis():
    dataframe = data_import()
    results_up_down = count_up_down(dataframe)
    results_only_up = count_only_up(dataframe)
    results_dominant_event =
forecast_dominant_event(dataframe)

    print "Dominant event: ", results_dominant_event
     yn_save = raw_input("Wanna save the tables?\n")

    for key, df in results_up_down.iteritems():
        if (yn_save == "y"):
            rename_columns(rename_indexes(df)).to_csv("{}-
day-movements.csv".format(key), encoding='utf-8')

        if not isinstance(key, (int, long)): continue
        yn = raw_input("Want to proceed with the k = " +
str(key) + " U/D plot? [y/n]\t")
        if (yn == "y"):
            plot(df, key, 'U/D movements in a ' + str(key) +
'-day sequence')

    for key, df in results_only_up.iteritems():
        if (yn_save == "y"):
            rename_columns(df).to_csv("{}-day-
ups.csv".format(key), encoding='utf-8')

        yn = raw_input("Want to proceed with the k = " +
str(key) + " U plot? [y/n]\t")
        if (yn == "y"):
```

```
            plot(df, key, 'Count of U movements in a ' +
str(key) + '-day sequence')

def custom_forecast_dominant_event(dataframe):

    # With S&P

dataframe = pd.read_csv('sp.csv', parse_dates=["Date"],
    date_parser=lambda x: pd.datetime.strptime(x, '%Y-%m-
    %d'), usecols=["Date","Close"])
    dataframe = dataframe.iloc[::-1]
    dataframe['decade'] = dataframe['Date'].apply(lambda
date: date.to_datetime().year//10*10)
    dataframe['up-down'] = 1 * (dataframe['Close'] >
dataframe['Close'].shift(1))
    training_set = dataframe[dataframe.Date <=
dt.datetime(2009, 12, 31)]
    cv_set = dataframe[((dataframe.Date >= dt.datetime(2010,
1, 1)) & (dataframe.Date <= dt.datetime(2011, 7, 31)))]

    # With INP.CSV

    training_set = dataframe[:int(dataframe.shape[0]*0.95)]
    cv_set = dataframe[int(dataframe.shape[0]*0.95):]

    df = count_up_down(training_set)['all']
    df = df.sum(axis=1)
    dominant_patterns = {}
    for index, combination in enumerate(df.index.values):
        if (index != 0) and (combination[:-1] ==
df.index.values[index-1][:-1]):
            first, second = combination,
df.index.values[index-1]
            dominant = first if (df.ix[[first]].values[0] >
df.ix[[second]].values[0]) else second
            dominant_patterns[combination[:-1]] = dominant[-
1]

    events_for_k = {}
    for k in range(1, 7):
        array_k = generate_list_of_tuples(cv_set, k)
        del array_k[-1]
        for i in range(k):
            array_k.insert(i, None)
        events_for_k[k] = array_k
```

```
    actual_events = pd.DataFrame.from_dict(events_for_k)
    actual_events = actual_events[k:]

    actual_events = actual_events.applymap(lambda x:
dominant_patterns[x])
    actual_events['random_prediction'] =
actual_events.iloc[:, 0].apply(lambda x: random.randint(0,
1))

    results = {}
    for column in actual_events.columns.values:
        results[column] = ((actual_events[column] ==
cv_set['up-down'][k:]) * 1).mean()

    return results

def ml_algo():
    dataframe = data_import()

    X, y, training_set = feature_extraction(dataframe,
past_days = 7)
    a = {}
    a['LogisticRegression'] =
LogisticRegression(verbose=False)
    a['LogisticRegression - 1K'] =
LogisticRegression(C=1000.0, verbose=False)
    a['DecisionTree'] = tree.DecisionTreeClassifier()
    a['NN (5, 2) - 1e-3'] = MLPClassifier(alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1, verbose=False)
    a['NN (5, 2) - 1.0'] = MLPClassifier(alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1, verbose=False)
    a['NN (25, 2) - 1e-3'] = MLPClassifier(alpha=1e-1,
hidden_layer_sizes=(25, 2), verbose=False)
    a['NN (25, 2) - 1.0'] = MLPClassifier(alpha=1.0,
hidden_layer_sizes=(25, 2), verbose=False)

    results = experiment_algorithms(X, y, a, kfold = True)
    print results

def experiment():
    dataframe = data_import('inp.csv')

    a = {}
    a['LogisticRegression'] =
```

```
LogisticRegression(verbose=False)
    a['LogisticRegression - 1K'] =
LogisticRegression(C=1000.0, verbose=False)
    a['DecisionTree'] = tree.DecisionTreeClassifier()
    a['NN (5, 2) - 1e-3'] = MLPClassifier(alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1, verbose=False)
    a['NN (5, 2) - 1.0'] = MLPClassifier(alpha=1e-5,
hidden_layer_sizes=(5, 2), random_state=1, verbose=False)
    a['NN (25, 2) - 1e-3'] = MLPClassifier(alpha=1e-1,
hidden_layer_sizes=(25, 2), verbose=False)
    a['NN (25, 2) - 1.0'] = MLPClassifier(alpha=1.0,
hidden_layer_sizes=(25, 2), verbose=False)

    d = {}
    for k in range(1, 8):
        d['all_t_{}'.format(k)] =
feature_extraction(dataframe, past_days = k)
        d['no_ups_{}'.format(k)] =
feature_extraction(dataframe, use_ups = False, past_days = k)
        d['no_updowns_{}'.format(k)] =
feature_extraction(dataframe, use_updowns = False, past_days
= k)
        d['no_value_{}'.format(k)] =
feature_extraction(dataframe, use_value = False, past_days =
k)
        d['only_updowns_{}'.format(k)] =
feature_extraction(dataframe, use_ups = False, use_value =
False, past_days = k)
        d['only_ups_{}'.format(k)] =
feature_extraction(dataframe, use_value = False, use_updowns
= False, past_days = k)
        d['only_value'] = feature_extraction(dataframe,
use_ups = False, use_updowns = False)

    results = experiment_dataset(MLPClassifier(alpha=0.1,
hidden_layer_sizes=(25, 2), verbose=False), d)
    results = results.sort(['accuracy'])

    results = experiment_dataset(MLPClassifier(alpha=0.1,
hidden_layer_sizes=(25, 2), verbose=False), d)
    results = results.sort(['accuracy'])

    X, y = feature_extraction(dataframe, past_days = 3,
use_value = False)
    algorithms = {'Final': MLPClassifier(alpha=0.1,
```

```
hidden_layer_sizes=(25, 2), verbose=False)}
    results = experiment_algorithms(X, y, algorithms)
    print results

def start():
    if (raw_input("Data analysis [1] or Machine Learning
[2]?\t") == "1"):
        data_analysis()
    else:
        experiment()
    if __name__ == "__main__":
    #setup()
    start()
```

Script with the functions for the up and down count and the forecasting with the first method, defined as “utils”

```
from __future__ import division # if you'll ever need
integer division: c = a // b
import os
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import itertools
import matplotlib
import matplotlib.cm as cm
from matplotlib.ticker import FuncFormatter
import random
import datetime as dt
#import seaborn as sns

def data_import(filename = 'inp.csv', shift_direction = 1):
    dataframe = pd.read_csv(filename)[['yy', 'value']]
    dataframe['decade'] = dataframe['yy'].apply(lambda yy:
yy//10*10) # generates the decade
    dataframe['up-down'] = 1 * (dataframe['value'] >
dataframe['value'].shift(shift_direction)) # defines the up-
down
    return dataframe

def to_percent(y, position):
    # Ignore the passed in position. This has the effect of
scaling the default
    # tick locations.
    s = str(100 * y)
```

```
    # The percent symbol needs escaping in latex
    if matplotlib.rcParams['text.usetex'] is True:
        return s + r'$\%$'
    else:
        return s + '%'

def generate_list_of_tuples(df, k):
    # creates an array of arrays, shifted of value from time
to time
    list_of_values = [df['up-down'][i:].tolist() for i in
range(k)]
    # then it zips them on the base of their position,
creating in this way
    # a list of tuples that represents all the values
sequences that happens
    # in the list of up and down
    list_of_tuples = list(zip(*list_of_values))
    return list_of_tuples

def count_unique_values_sum(df, k):
    list_of_tuples = generate_list_of_tuples(df, k)
    list_of_tuples = map(sum, list_of_tuples)
    # counts the unique values in the arrays
    counts = pd.value_counts(list_of_tuples)
    # gives back the zip correspondent to the results
    return zip(counts.index, counts.values)

def count_unique_values_combinations(df, k):
    list_of_tuples = generate_list_of_tuples(df, k)
    # counts the unique values of combinations
    counts = pd.value_counts(list_of_tuples)
    # gives back the zip correspondent to the results
    return zip(counts.index, counts.values)

def columns_to_percent(df):
    for column in df.columns.values:
        df[column] = df[column]/df[column].sum()
    return df

def count_only_up(dataframe):
    results_sum = {}
    decades = dataframe['decade'].unique()

    for k in range(1, 8): # from 1 to 7 days of possible
combinations
```

```
        results_sum[k] = pd.DataFrame(columns = decades,
index = range(k))
        for decade in decades: # for every decade
            df = dataframe[dataframe['decade'] == decade]
            counts = count_unique_values_sum(df, k)
            for count in counts:
                results_sum[k].ix[count[0], decade] =
count[1]

    # to percent
    for key, value in results_sum.iteritems():
        results_sum[key] =
columns_to_percent(results_sum[key])

    return results_sum

def count_up_down(dataframe):
    combinations = []
    results = {}
    decades = dataframe['decade'].unique()

    for k in range(1, 8): # from 1 to 7 days of possible
combinations
        r_for_k = [item for item in
itertools.product(range(2), repeat=k)]
        combinations.extend(r_for_k)
        results[k] = pd.DataFrame(columns = decades, index =
r_for_k).fillna(0)

    df_counts_all = pd.DataFrame(columns = decades, index =
combinations)
    df_counts_all = df_counts_all.fillna(0)

    for k in range(1, 8):
        for decade in decades: # for every decade
            df = dataframe[dataframe['decade'] == decade]
            counts = count_unique_values_combinations(df, k)
            for count in counts:
                df_counts_all.ix[[count[0]], decade] =
count[1]
                results[k].ix[[count[0]], decade] = count[1]

    # Converts the values from aboslute to percentages
    for key, value in results.iteritems():
        results[key] = columns_to_percent(results[key])
```

```
    results['all'] = df_counts_all
    return results

def forecast_dominant_event(dataframe):

    df = count_up_down(dataframe)['all']
    df = df.sum(axis=1)
    dominant_patterns = {}
    for index, combination in enumerate(df.index.values):
        if (index != 0) and (combination[:-1] ==
df.index.values[index-1][:-1]):
            first, second = combination,
df.index.values[index-1]
            dominant = first if (df.ix[[first]].values[0] >
df.ix[[second]].values[0]) else second
            dominant_patterns[combination[:-1]] = dominant[-
1]

    events_for_k = {}
    for k in range(1, 7):
        array_k = generate_list_of_tuples(dataframe, k)
        del array_k[-1]
        for i in range(k):
            array_k.insert(i, None)
        events_for_k[k] = array_k

    actual_events = pd.DataFrame.from_dict(events_for_k)
    actual_events = actual_events[k:]

    actual_events = actual_events.applymap(lambda x:
dominant_patterns[x])
    actual_events['random_prediction'] =
actual_events.iloc[:, 0].apply(lambda x: random.randint(0,
1))

    results = {}
    for column in actual_events.columns.values:
        results[column] = ((actual_events[column] ==
dataframe['up-down'][k:]) * 1).mean()

    return results

def combination_to_string(comb):
    string = ""
```

```
    for item in comb:
        string += "U" if item == 1 else "D"
    return string

def rename_indexes(df):
    dict = {}
    for indexes in df.index.values:
        if not isinstance(indexes, tuple): break
        dict[indexes] = combination_to_string(indexes)
    return df.rename(index=dict)

def rename_columns(df):
    dict = {}
    for column in df.columns.values:
        dict[column] = ("20" if column < 2 else "19") +
str(column) + "0s "
    return df.rename(columns=dict)

def plot(df, k, title):
    # see: http://emptypipes.org/2013/11/09/matplotlib-
multicategory-barchart/
    fig = plt.figure()
    ax = fig.add_subplot(111)
    space = 0.3

    df = rename_columns(rename_indexes(df))
    conditions = df.index.values # ie. combinations
    categories = df.columns.values # ie. decades
    n = len(conditions)
    width = (1 - space) / (len(conditions))

    for i, combination in enumerate(conditions):
        indeces = range(1, len(categories)+1)
        vals = df.ix[combination].tolist()
        pos = [j - (1 - space) / 2. + i * width for j in
range(1,len(categories)+1)]
        rects = ax.bar(pos, vals, width=width,
label=combination, color=cm.Accent(float(i) / n))
        if (k > 3): continue # skip the % labeling with k
is too big
        for rect in rects:
            height = rect.get_height()
            ax.text(rect.get_x() + rect.get_width()/2.,
1.08*height, '{0:.2f}%'.format(height*100), ha='center',
va='bottom', rotation = 90)
```

```
    ax.set_ylabel("% of occurrence")
    ax.set_title(title)
    ax.set_xticks(indeces)
    ax.set_xticklabels(categories)
    ax.yaxis.set_major_formatter(FuncFormatter(to_percent))
    ax.set_ylim((ax.get_ylim()[0], ax.get_ylim()[1]*1.25))

    box = ax.get_position() # Shrink current axis's height by
10% on the bottom
    ax.set_position([box.x0, box.y0 + box.height * 0.1,
box.width, box.height * 0.9])
    ax.legend(loc='upper center', bbox_to_anchor=(0.5, -
0.05), fancybox=True, shadow=True, ncol=5) # Put a legend
below current axis

    plt.show(block=False)
```

Script with the machine learning functions for the application with decision tree, logistic regressions and neural networks, defined as “ml”:

```
from utils import *
from sklearn import tree, metrics
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import SGDClassifier
from sklearn.cross_validation import train_test_split
from sklearn.cross_validation import cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
import numpy as np
import pandas as pd
import itertools
from scipy.stats import gmean
from sklearn.feature_selection import *
from sklearn.pipeline import Pipeline
from sklearn.ensemble import ExtraTreesClassifier,
RandomForestClassifier
from sklearn import cross_validation

def feature_extraction(dataframe, past_days = 7, use_updowns
= True, use_ups = True, use_value = True):
    """
    Preparares the features, extracting the X matrix and the
```

```
    y target vector from the dataframe provided
    """

    old_columns = dataframe.columns.values.tolist()
    window = dataframe['value'].expanding()
    dataframe = pd.concat([dataframe, window.min(),
window.mean(), window.max()], axis=1)
    dataframe.columns = old_columns + ['min', 'mean', 'max']

    ks = range(1, past_days+1) # number of possible (past)
days took into account: 1 to 7


    # 1st step: defining all the combination of events from
t-1 to t-8 wrt t

    events_for_k = {}
    # n = dataframe.shape[0]
    for k in ks:
        array_k = generate_list_of_tuples(dataframe, k)
        del array_k[-1] # delete the last item: we don't
care about today (since we want to make a prediction)
        for i in range(k):
            array_k.insert(i, None) # and we don't care for
samples we don't have past data for
        events_for_k[k] = array_k
    # matrix of [n * k] size, with all the events *actually*
occurred for each k
    actual_events = pd.DataFrame.from_dict(events_for_k)

    possibile_events = [] # now we generate all the possible
events
    for k in ks: # for each day up to t-k+1
        possibile_events.extend(itertools.product(range(2),
repeat=k)) # all combinations
    # matrix of [n * features] size, with all the events
actually occurred
    features = pd.DataFrame(columns = possibile_events)
    for single_possible_event in possibile_events:
        features[single_possible_event] =
(actual_events[len(single_possible_event)] ==
single_possible_event) * 1


    # 2n step: defining the sum of all the UPs from t-1 to t-
```

```
8

    actual_ups = actual_events.apply(lambda x: map(np.sum,
x) or 0) # just a sum of the tuples

    translation = {} # in order not to merge the wrong
columns (since they share the name)
    for column in actual_ups.columns.values:
        translation[column] = str(column) + "-ups"
    actual_ups = actual_ups.rename(columns = translation)

    # final training set: dataframe + U/D combinations + U
count
    training_set_to_use = [dataframe[(['up-down'] +
(['value', 'min', 'max', 'mean'] if use_value else []))]]
    if use_updowns: training_set_to_use.append(features)
    if use_ups: training_set_to_use.append(actual_ups)
    training_set = pd.concat(training_set_to_use, axis=1)[k:]

    # final column renaming
    translation = {'up-down': 'y'}
    for column in training_set.columns.values:
        if not isinstance(column, tuple): continue
        translation[column] = combination_to_string(column)
    training_set = training_set.rename(columns = translation)

    # remove perfectly correlated variables
    # np.corrcoef(training_set['U'].values,
training_set['D'].values)
    # np.corrcoef(training_set['1-ups'].values,
training_set['U'].values)
    # np.corrcoef(training_set['1-ups'].values,
training_set['y'].values)
    # np.corrcoef(training_set['value'].values,
training_set['y'].values)

    training_set = training_set[[key for key in
training_set.keys() if (key != 'U' and key != 'D')]]

    # done!
    X = training_set[[key for key in training_set.keys() if
"y" not in key]].values
    y = np.asarray(training_set["y"].values)

    return X, y, training_set
```

```
def test_model(model = None, k = 3):
    X, y, training_set = feature_extraction(data_import(),
past_days = k)
    if model == None:
        model = MLPClassifier(alpha=0.01,
hidden_layer_sizes=(1000, 10), verbose=True)
    return
compute_model_score(StandardScaler().fit_transform(X), y,
model)


def feature_preparation(X, y, preprocess = True):
    """
    Generates the training and the test sets, and pre-process
the features (normalization)
    """

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.05)

    if preprocess:
        scaler = StandardScaler()
        # Don't cheat - fit only on training data
        scaler.fit(X_train)
        X_train = scaler.transform(X_train)
        # apply same transformation to test data
        X_test = scaler.transform(X_test)

    return X_train, X_test, y_train, y_test

def experiment_algorithms(X, y, algorithms, kfold = False):
    if (kfold):
        results = pd.DataFrame(columns = ['score'])
        for title, algorithm in algorithms.iteritems():
            results.ix[title, 'score'] =
compute_model_score(X, y, algorithm)

        return results.sort_values(by=['score'],
ascending=False)
    else:
        results = pd.DataFrame(columns = ['accuracy', 'F1',
'precision', 'recall', 'train_score'])

        X_train, X_test, y_train, y_test =
```

```
feature_preparation(X, y, preprocess)

        for title, algorithm in algorithms.iteritems():
            algorithm.fit(X_train, y_train)
            test_prediction = algorithm.predict(X_test)
            results.ix[title, 'accuracy'] =
metrics.accuracy_score(y_test, test_prediction)
            results.ix[title, 'F1'] =
metrics.f1_score(y_test, test_prediction)
            results.ix[title, 'precision'] =
metrics.precision_score(y_test, test_prediction)
            results.ix[title, 'recall'] =
metrics.recall_score(y_test, test_prediction)
            results.ix[title, 'train_score'] =
algorithm.score(X_train, y_train)

        return results

def experiment_dataset(algorithm, datasets, preprocess =
True):
    results = pd.DataFrame(columns = ['accuracy', 'F1',
'precision', 'recall', 'train_score'])

    for title, dataset in datasets.iteritems():
        X_train, X_test, y_train, y_test =
feature_preparation(dataset[0], dataset[1], preprocess)
        algorithm.fit(X_train, y_train)
        test_prediction = algorithm.predict(X_test)
        results.ix[title, 'accuracy'] =
metrics.accuracy_score(y_test, test_prediction)
        results.ix[title, 'F1'] = metrics.f1_score(y_test,
test_prediction)
        results.ix[title, 'precision'] =
metrics.precision_score(y_test, test_prediction)
        results.ix[title, 'recall'] =
metrics.recall_score(y_test, test_prediction)
        results.ix[title, 'train_score'] =
algorithm.score(X_train, y_train)

    return results

def run_experiment(datasets, algorithms, kfold = False):
    if kfold:
        results = pd.DataFrame(columns = ['score'])
        for title_dataset, dataset in datasets.iteritems():
```

```
            for title_algorithm, algorithm in
algorithms.iteritems():
                title = title_algorithm+' - '+title_dataset
                print 'Fitting ' + title + '...'
                results.ix[title, 'score'] =
compute_model_score(dataset[0], dataset[1], algorithm)

        return results.sort_values(by=['score'],
ascending=False)
    else:
        results = pd.DataFrame(columns = ['accuracy', 'f1',
'mean'])

        for title_dataset, dataset in datasets.iteritems():
            for title_algorithm, algorithm in
algorithms.iteritems():
                attempt = 1
                title = title_algorithm+' - '+title_dataset
                print 'Fitting ' + title + '...' +
('[{}]'.format(attempt) if attempt > 1 else '')
                successful = False

                while (not successful) and (attempt < 4):
                    attempt += 1

                    X_train, X_test, y_train, y_test =
feature_preparation(dataset[0], dataset[1], True)
                    algorithm.fit(X_train, y_train)
                    test_prediction =
algorithm.predict(X_test)

                    accuracy = metrics.accuracy_score(y_test,
test_prediction)
                    f1 = metrics.f1_score(y_test,
test_prediction)

                    results.ix[title, 'accuracy'] = accuracy
                    results.ix[title, 'f1'] = f1
                    results.ix[title, 'gmean'] =
gmean((accuracy, f1))

                    successful = metrics.f1_score(y_test,
test_prediction) != 0

        return results.sort(['gmean'],
```

```
ascending=False).head()

def experiment(data = 'inp.csv'):
    dataframe = data_import(data)
    old_columns = dataframe.columns.values.tolist()
    window = dataframe['value'].expanding()
    dataframe = pd.concat([dataframe, window.min(),
window.mean(), window.max()], axis=1)
    dataframe.columns = old_columns + ['min', 'mean', 'max']

    a = {}
    a['LogisticRegression'] =
LogisticRegression(verbose=False)
    a['LogisticRegression-1K'] = LogisticRegression(C=1000.0,
verbose=False)
    a['DecisionTree'] = tree.DecisionTreeClassifier()
    a['NN_(5,2)-1e-2'] = MLPClassifier(alpha=1e-2,
hidden_layer_sizes=(5, 2), verbose=False)
    a['NN_(5,2)-1.0'] = MLPClassifier(alpha=1.0,
hidden_layer_sizes=(5, 2), verbose=False)
    a['NN_(25,2)-1e-2'] = MLPClassifier(alpha=1e-2,
hidden_layer_sizes=(25, 2), verbose=False)
    a['NN_(25,2)-1.0'] = MLPClassifier(alpha=1.0,
hidden_layer_sizes=(25, 2), verbose=False)

    d = {}
    for k in range(1, 8):
        d['all_t_{}'.format(k)] =
feature_extraction(dataframe, past_days = k)
        d['no_ups_{}'.format(k)] =
feature_extraction(dataframe, use_ups = False, past_days = k)
        d['no_updowns_{}'.format(k)] =
feature_extraction(dataframe, use_updowns = False, past_days
= k)
        d['no_value_{}'.format(k)] =
feature_extraction(dataframe, use_value = False, past_days =
k)
        d['only_updowns_{}'.format(k)] =
feature_extraction(dataframe, use_ups = False, use_value =
False, past_days = k)
        d['only_ups_{}'.format(k)] =
feature_extraction(dataframe, use_value = False, use_updowns
= False, past_days = k)
        d['only_value'] = feature_extraction(dataframe,
use_ups = False, use_updowns = False)
```

```
    results = run_experiment(d, a)
    print results
    return results

    # ---

    a = tree.DecisionTreeClassifier()
    a = MLPClassifier(alpha=0.01, hidden_layer_sizes=(1000,
10), verbose=True)
    d = {}
    for k in range(1, 8):
        d['t_{}'.format(k)] =
ml.feature_extraction(dataframe, past_days = k)
    results = ml.experiment_dataset(a, d)

def compute_model_score(X, y, model):
    num_instances = len(X)
    num_folds = 10
    seed = 7
    kfold = cross_validation.KFold(n=num_instances,
n_folds=num_folds, random_state=seed)
    results = cross_validation.cross_val_score(model, X, y,
cv = kfold)
    return results.mean(), results.std()

def feature_filtering_performance(X, y, filter = f_classif,
model = LogisticRegression(), title = 'Performance varying
the percentile of features selected'):
    clf = Pipeline([('filter', SelectPercentile(score_func =
filter)), ('model', model)])
    score_means = list()
    score_stds = list()
    percentiles = (1, 3, 6, 10, 15, 20, 30, 40, 60, 80, 100)
    for percentile in percentiles:
        clf.set_params(filter__percentile=percentile)
        this_scores = cross_val_score(clf, X, y)
        score_means.append(this_scores.mean())
        score_stds.append(this_scores.std())
    plt.errorbar(percentiles, score_means,
np.array(score_stds))
    plt.title(title)
    plt.xlabel('Percentile')
    plt.ylabel('Prediction rate')
    plt.axis('tight')
```

```
    plt.show(block=False)

def feature_analysis(dataframe):
    X, y, training_set = ml.feature_extraction(dataframe,
past_days = 7)

    #np.corrcoef(training_set['U'].values,
training_set['D'].values)
    #np.corrcoef(training_set['max'].values, y)

    # test = SelectKBest(score_func=chi2, k=4).fit(X, y) #
Only for non-negative features
    test = f_classif(X, y)
    test2 =
SelectKBest(score_func=mutual_info_classif).fit(X, y) # see
https://www.cs.utah.edu/~piyush/teaching/22-9-print.pdf
    test3 = ExtraTreesClassifier().fit(X, y)
    test4 = RFE(SGDClassifier(alpha=1.0)).fit(X, y)

    rank = pd.DataFrame(test.scores_, index =
training_set[[key for key in training_set.keys() if "y" not
in key]].columns.values, columns=['f_classif'])
    rank['mutual_info_classif'] = test2.scores_
    rank['ExtraTreesClassifier'] = test3.feature_importances_
    rank['RFE'] = test4.ranking_
    rank['RandomForestClassifier'] =
RandomForestClassifier().feature_importances_
    rank = rank.sort_values(by='f_classif', ascending=False)
    rank = rank.sort_values(by='mutual_info_classif',
ascending=False)
    rank = rank.sort_values(by='ExtraTreesClassifier',
ascending=False)
    rank = rank.sort_values(by='RFE')

    selection = pd.DataFrame(index = training_set[[key for
key in training_set.keys() if "y" not in
key]].columns.values)
    selection['f_classif'] = f_classif(X, y)[0]
    selection['mutual_info_classif'] = mutual_info_classif(X,
y)[0]
    selected_features = [x for x in
selection.sort_values(by='f_classif',
ascending=False).head(selection.shape[0]//5).index.values.tol
ist()\
        if x in
```

```
selection.sort_values(by='mutual_info_classif',
ascending=False).head(selection.shape[0]//4).index.values.tol
ist()]

    X_new = training_set[[feature for feature in
selected_features]].values

    ml.feature_filtering_performance(X, y, model =
LogisticRegression(C=100.0), title = 'Performance of the
LogisticRegression-ANOVA varying the percentile of features
selected')

    transform = SelectPercentile(f_classif)
    transform = SelectPercentile(score_func =
mutual_info_classif)
    clf = Pipeline([('filter', SelectPercentile()), ('svc',
SVC(C=1.0))])
    score_means = list()
    score_stds = list()
    percentiles = (1, 3, 6, 10, 15, 20, 30, 40, 60, 80, 100)
    for percentile in percentiles:
        clf.set_params(filter__percentile=percentile)
        this_scores = cross_val_score(clf, X, y)
        score_means.append(this_scores.mean())
        score_stds.append(this_scores.std())
    plt.errorbar(percentiles, score_means,
np.array(score_stds))
    plt.title('Performance of the SVM-Mutual Information
varying the percentile of features selected')
    plt.title('Performance of the LogisticRegression-ANOVA
varying the percentile of features selected')
    plt.xlabel('Percentile')
    plt.ylabel('Prediction rate')
    plt.axis('tight')
    plt.show()

    X_new = SelectPercentile(f_classif,
percentile=5).fit_transform(X, y)
    scaler = StandardScaler()
    scaler.fit(X_new)
    X_new = scaler.transform(X_new)
```

APPENDIX B

A decision tree is a tree shaped predictive model, employed for mapping observations about an item with the purpose of arriving to a conclusion about the value of that item. Data mining uses decision tree learning with the aim of building a model that, using input variables, is able to predict target variable (the variable that has to be or should be the outcome) values.

The starting point of this diagram is called "root node", with just out-coming branches (not incoming); interior nodes have an incoming edge (input variables) and outgoing edges, used to split the sample into two or more subsets according to the decision rule. The other nodes are called leaves and are also defined as terminal or decision nodes, because they can be allocated to one class with a specific target value or they can have a probability value for the happening of a certain outcome. The paths from root to leaf and from leaf to leaf are called branches and represent the decision rules: they lead to a possible outcome of the tree (leaf) or to another interior node with a different input.

Figure 31 represents this structure:

Decision Tree structure

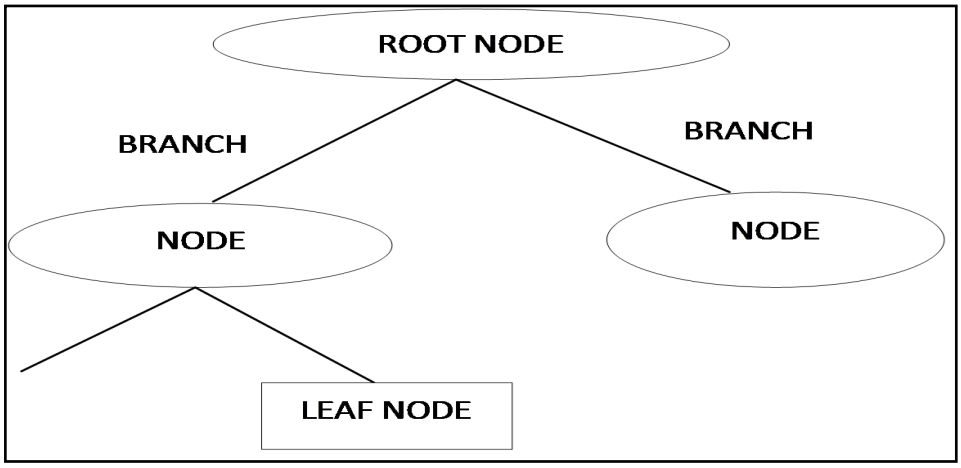


Figure 31: example of decision tree. Source: http://insightfromdata.blogspot.it/2012/06/decision-tree-unembellished.html

There are two types of decision trees: if the values of the model can assume a finite set of outcomes it is a classification tree, where the leaves represent the class labels and the branches are the features connections that lead to those class labels. Instead, if the target variable can assume continuous values (so real numbers), the decision tree is a regression one.

At each passage from node to node, the decision tree splits the dataset into smaller subsets. This process, called partitioning, ends when the subset has the same values of the target or when more subdivisions do not improve the model ability for predictions.

To construct decision trees, data mining uses different algorithms, such as the C5.0 used in this study. The algorithm C5.0 divides the sample according to the outcome that gives the maximum information gain (a metric explained in the following lines). Every subsample is split again and again, until no other subdivisions are possible. Finally, the latter subsamples are reviewed, to cut off the ones that do not contribute in a significant way to the final model value. These algorithms, generally, work top-down: at each passage they choose a variable that splits at the best the sample; to judge the goodness of this "best" different metrics are used, for example information gain. Information gain is employed to establish which is the feature that has to be split at each step of the decision tree; the objective is a small tree, so for every passage the purest (that is the closer to he target variable) subdivision is selected.

Advantages of a decision tree:

- Easy to comprehend, since it can be graphically displayed.
- Employed both with numeral and categorical data.
- It can be used with large datasets.
- It imitates human decision-making.

Limits of decision trees:

- Not so precise with respect to other tools.
- May be non-robust: a little change in the training set can produce big changes in the tree performance.
- Decision tree algorithms can create too intricate trees, causing overfitting because they are not able to properly generalize from training data.

Logistic regression is a kind of regression model characterized by the dependent variable, which is categorical (can take a specified range of values). In economics, logistic regression is a discrete choice model, so a model that describes, unfolds and forecasts choices among two or more discrete alternatives (alternatives that take a finite number of values). Logistic regressions can be:

- Binomial/binary: the dependent variable can assume two different outcomes.
- Multinomial: the outcome can assume three or more values, not ordered.
- Ordinal: regression with ordered values (e.g. a grading scale ordered from the best to the worst grade).

Logistic regression is employed to forecast the odds (that is the probability of a specific outcome) a numerical expression of a specific case, grounded on the independent variable values:

$$odds = \frac{outcome\ probability\ of\ being\ a\ case}{probability\ of\ being\ a\ non-case}.$$

It is recalled that in this kind of regressions one or more predictive variables are employed, and residuals are not normally distributed,

because it is a binomial regression.

The relationship between dependent and one or more independent variables is investigated using a logistic function to estimate probability; it is a s-shaped curve (called sigmoid curve), displayed in Figure 32, whose equation is

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}.$$

Where:

L= curve's maximum value

k= slope of the curve

x_0= x-value of the sigmoid middle point

Sigmoid function

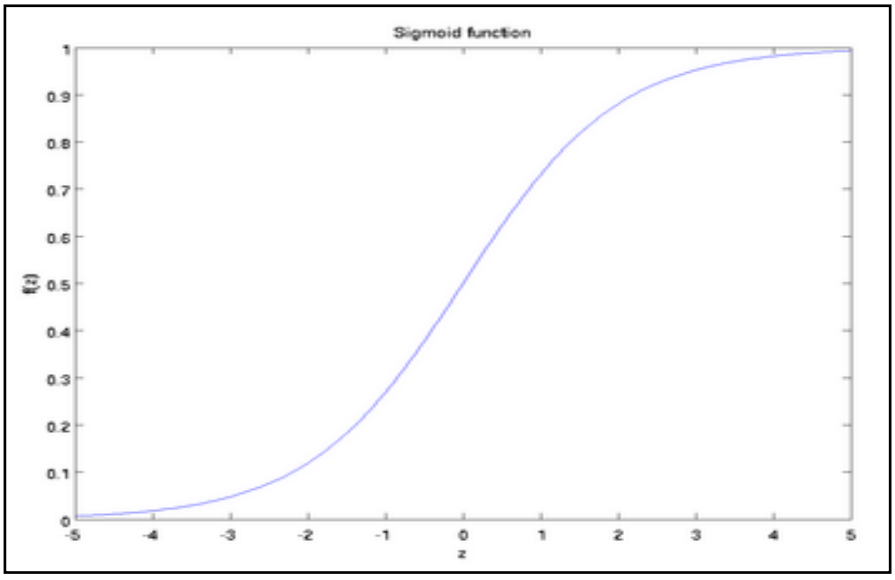


Figure 32: example of sigmoid function. Source: http://ufldl.stanford.edu/wiki/index.php/Neural_Networks

APPENDIX C

Scientists are trying to build a machine that can reason like a human brain, since a long time.

A human brain is composed by billions of cells, called neurons. As it can be seen in Figure 31, each neuron is composed by a central mass, that is the *cell body*, from which different connections release information from the cell input to the cell body (the *dendrites*) and by one *axon*, which is the cell output that carries the information away.

Neuron structure

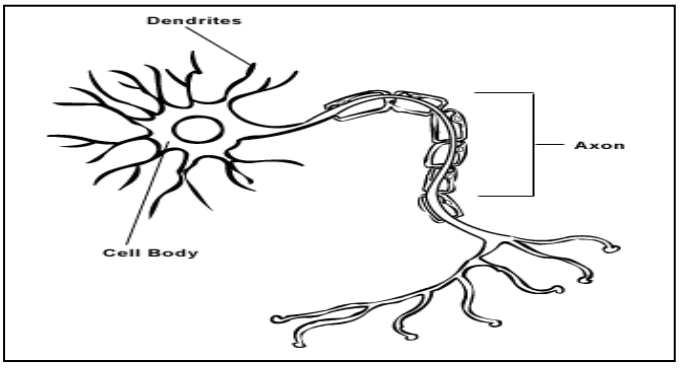


Figure 31: Basic structure of a brain cell. Source: Chris Woodford, Neural Networks

A great achievement in the field of artificial intelligence tools is the creation of Artificial Neural Networks (since now ANNs). Dr. Robert Hecht-Nielsen, the inventor of one of the first neurocomputers, gives a definition of ANNs:

> *"…[A ANN is] a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic stare response to external inputs."*
>
> *(Neural Network Primer: Part I, Maureen Caudill, AI Expert, Feb. 1989)*

In other words, ANNs are computer programs composed by hundreds of artificial brain cells, which are trained to learn and act like a human brain, in order to become a self-learning system able to make decisions. Therefore, the aim of this tools is to solve complex problems like a human brain would, even if ANNs are much more abstract. In this appendix just Feedforward Neural Networks will be described: their principal property is that the connections among the units do not form a cycle, contrary to the Recurrent Neural Networks.
ANN are usually structured in layers, linked by interconnections, as Figure 32 displays:

Structure of a two-layers Neural Network

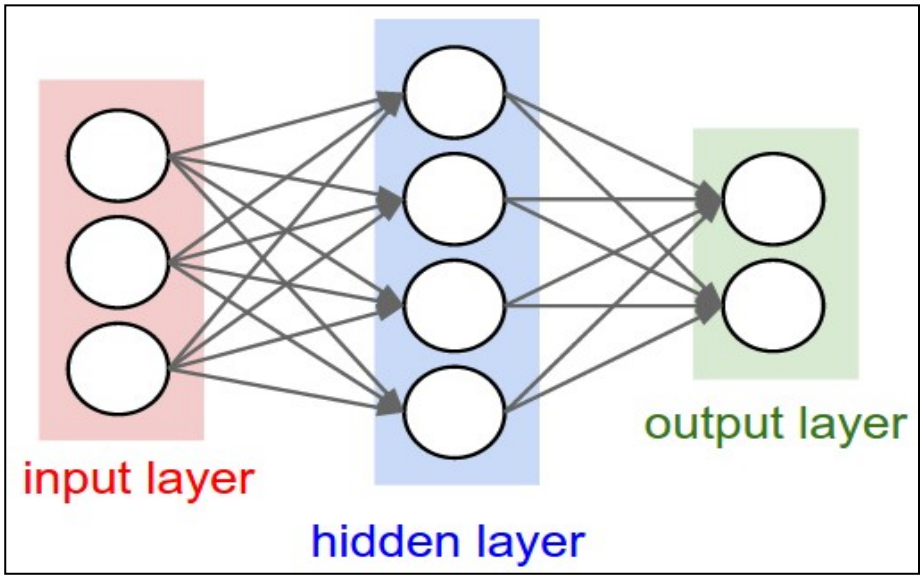


Figure 32: two-layer NN with three inputs, source: http://cs231n.github.io/neural-networks-1/

On the left side there are the input layers, which receive information from the outside; in the middle there are the hidden layers, the core of the artificial brain: these features communicate in a parallel fashion both with the input and output units thanks to weighted connections to improve forecasts. On the right side the output layers receive the information and signal how well the system responds to the learnt

intelligence. The connections among the different layers are also called synapsis and are represented by a number, called weight. Weights can be either positive (one unit triggers another) or negative (one unit represses another): the higher the weight, the higher the influence of the relative unit to another. The majority of ANNs are *fully connected*, like the one in Figure 32: it means that all the units are linked one with another. As previously said, the hidden layers give to the network the capacity of generalizing and make the network able to learn from examples, as a human brain; however the creation of a Multilayered Feedforward Network, so a model with more than one hidden layers, raises the overfitting risk. Figure 33 gives a sample of a Feedforward Network with two hidden layers:

Example of Neural Network with two hidden layers

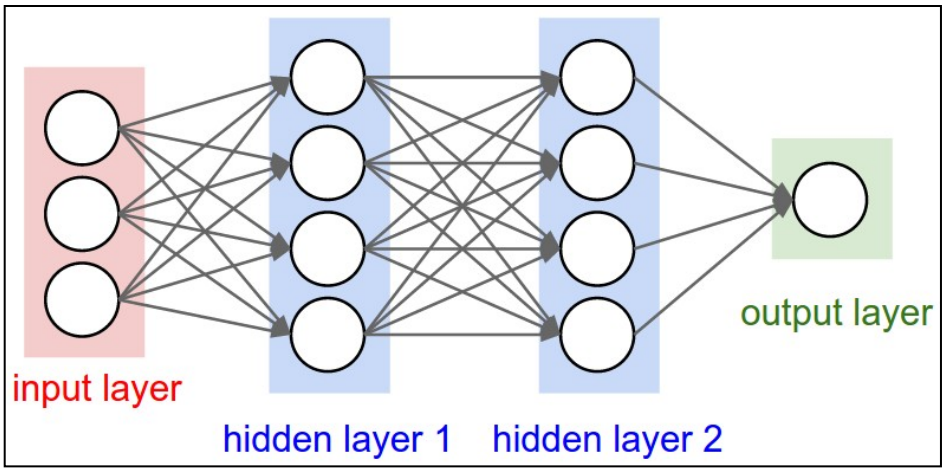


Figure 33: Thee-layer neural network with three inputs and two hidden layers. Source: http://cs231n.github.io/neural-networks-1/

Neural networks are able to learn thanks to different 'learning rules', for example the delta rule, usually used in the Backpropagational Neural Networks (BPNNs). The delta rule allows the system to learn whenever the network receive a new input, so in every cycle, since it initially makes a random guess, which is compared to the actual event

to adjust properly the connection weights and to obtain a better result. This learning process starts from the output and, passing through the hidden layers, arrives to the input, namely it goes backward, as suggested by the name of these ANNs. After the network has been trained with a sufficient number of examples, it can be used with a new set of input.

Neural networks are applied to several fields, such as game playing and decision-making (e.g. in poker), medical diagnosis, pattern recognition like facial identification and in finance to improve trading systems. A very famous application of ANNs is their use to make a computer able to recognise handwritten numbers, like those shown in Figure 33:

Handwritten numbers' sample

Figure 33: Handwritten digits. Source: http://neuralnetworksanddeeplearning.com/chap1.html

To train the neural network is necessary to take a huge number of handwritten numbers (the training examples) and generate a system able to learn from them. Figure 34 displays the three-layer Neural Network used for this experiment, whit the input and hidden layers reduced in order to make the diagram easier to understand:

Three-layer neural network

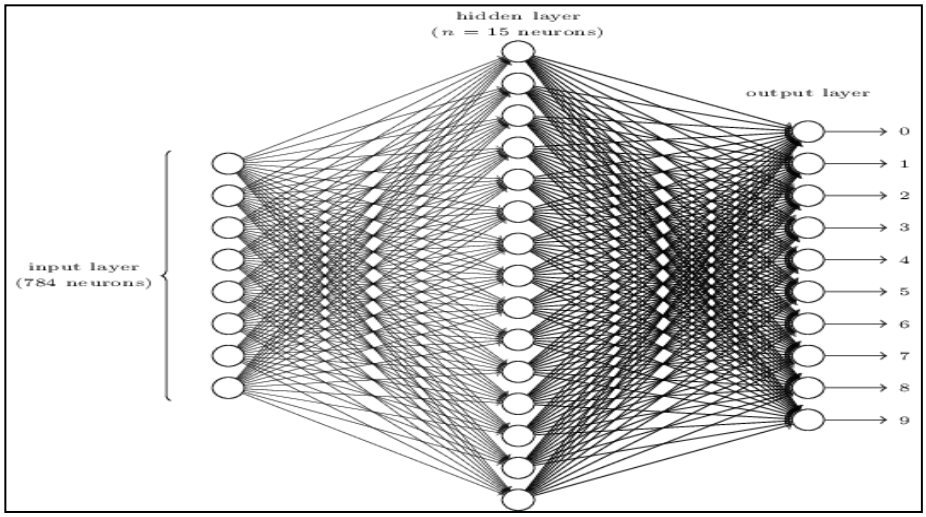
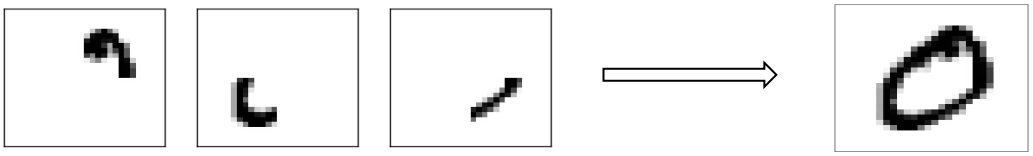


Figure 34: Three-layer neural network for the recognition of handwritten numbers. Source: http://neuralnetworksanddeeplearning.com/chap1.html

The input layers are in total 784 images, representing pixel segments of the different numbers. The hidden layer can assume *n* values, in this example it is composed by 15 neurons, and they have the task of recognise the fragments of pixel received by the input layers. Finally the output layers fire themselves when they are “hit” by the hidden layers that composed the specific number. For example, the hidden layers may have to detect the following fragments, belonging to the same number shown on the right side:



If the relative three layers work in the same moment, the system will fire for sure the output layer that corresponds to a 0.

This experiment underlines the ability of Neural Networks to imitate human brains, thanks to the skill of working in a parallel fashion.

The neural networks used in this thesis have both two hidden layers, but the first time each hidden layer has 5 nodes, the second time it has 25 nodes per hidden layer.

BIBLIOGRAPHY

- Andrew W. Lo (08-15-2004): *The adaptive market hypothesis: market efficiency from an evolutionary perspective:* 1-2, 14-20.
- Avramov D., Chordia T. (2005): *Predicting stock returns.* J Finance Vol. 82, no. 2: 387-415.
- Bodie, Kane and Marcus (2004): *Essentials of Investments*, 5th edition, Chapter 8: 260-270.
- De Long J., Shleifer A., Summers L., Waldmann R. (1990): *Positive feedback investment strategies and destabilizing rational speculation.* J Finance Vol. 45: 379-395.
- Doran J, Ronn E, Goldberg R (2009): *A simple model for time-varying expected returns on the S&P500 index.* J Investment Management.
- Doron Avramov and Tarun Chordia (November 2006): *Predicting Stock Returns.* J Financial Econometrics, vol *82.*
- Fama E., Blume M. (1996): *Filter rules and stock-market trading*. J Bus 39: 226-241.
- J. Bradford de Long; Andrei Shleifer; Lawrence H. Summers; Robert J. Waldmann (June 1990): *Positive Feedback Investment Strategies and Destabilizing Rational Speculation.* J Finance Vol 45, No.2: 374-97.
- Jensen M., Benington G. (1970): *Random walks and technical theories: computational algorithms, statistical inference, and empirical implementation.* J Finance Vol 54: 469-482.

- Jonathan Clarke, Tomas Jandik, Gershon Mandelker (2001): *The efficient market hypothesis, in Robert C. ARFFA, expert financial planning: investment strategies from industry lenders,* chapter 9: 126-141.
- Kleinbaum D., Klein M. (2010): Logistic Regression, Third Edition, Chapter 1: 1-37.
- Malliaris A.G., Malliaris M. (2013): *N-tuple S&P patterns across decades, 1950-2011.*
- Martin T. Bohl, Stefan Reitz (2014): *The Influence of Positive Feedback Trading on Return Autocorrelation: Evidence for the German Stock Market.*
- McNelis P. (2005): Neural Networks in Finance: gaining predictive edge in the market, Chapter 2: 18-75.
- Pandya R., Pandya J. (May 2015): *C5.0 Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning,* International Journal of Computer Application Vol. 117 No. 16: 18-20.
- Patil N., Lathi R., Chitre V. (2012): *Comparison of C5.0 & CART classification algorithms using pruning technique*. Int J Engineering Research and Technology 1(4):1-5.
- Schulmeister S. (2007): *The profitability of technical stock trading has moved from daily to intradaily data.* WIFO Working Paper No. 289.

SITOGRAPHY

- *A Basic Introduction to Neural Networks,* http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html, consulted on 14/11/2016.
- *CBOE Volatility Index (VIX), http://www.cboe.com/micro/VIX/vixintro.aspx, consulted on 17/06/2016.*
- 9/11/2013, *Creating a Grouped Bar Chart in Matplotlib, http://emptypipes.org/2013/11/09/matplotlib-multicategory-barchart/,* consulted on 10/12/2016.
- *Financial Crisis 2007/2008 Overview*, *http://www.wallstreetoasis.com/financial-crisis-overview*, consulted on 25/06/16.
- Florian Agreiter, 5/27/2009: *The dot com bubble* http://agreiter.com/blog/?p=166#_ftn2, consulted on 15/07/2016.
- *Logistic Regression*, http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, consulted on 05/12/2016.
- Marcel Caraciolo (31/08/2010): *Tools for Machine Learning Performance Evaluation: Confusion Matrix*, http://aimotion.blogspot.it/2010/08/tools-for-machine-learning-performance.html, consulted on 10/01/2017.
- Matthew Boesler, 6/6/2013, *ANALYST: Everyone Is Now Paying Attention To The 50-Day Moving Average* http://www.businessinsider.com/sp-500-to-test-50-day-moving-average-2013-6?IR=T, consulted on 10/08/2016.

- Michael A. Nielsen (2015): *Neural Networks and Deep Learning*, Chapter 1, http://neuralnetworksanddeeplearning.com/chap1.html, consulted on 19/01/2017.
- MLP Classifier, http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, consulted on 3/02/2017.
- Model Evaluation: quantifying the quality of predictions, http://scikit-learn.org/stable/modules/model_evaluation.html, consulted on 12/01/2017.
- 22/09/2011, *Model Selection and Feature Selection*, https://www.cs.utah.edu/~piyush/teaching/22-9-print.pdf, consulted on 15/12/2016.
- Morris J. (2008): *Analysis of the Dot-Com Bubble of the 1990s*: 2-6, 25-27 https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1152412, consulted on 12/07/2016.
- Murtuza Morbiwala*: The Decision Tree Unembellished,* http://insightfromdata.blogspot.it/2012/06/decision-tree-unembellished.html, consulted on 18/01/2017.
- *Neural Network Models, http://scikit-learn.org/stable/modules/neural_networks_supervised.html, consulted on 31/01/2017*
- Neural Networks, http://ufldl.stanford.edu/wiki/index.php/Neural_Networks, consulted on 1/02/2017.
- *OPEC: Our mission*, *http://www.opec.org/opec_web/en/about_us/23.htm* consulted on 06/01/2017.

- *SVM Anova: SVM with univariate feature selection, http://scikit-learn.org/ stable/ auto_examples/ svm/plot_svm_anova.html, consulted on 3/01/2017.*
- *Understanding the Decision Tree Structure*, http://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html, consulted on 02/12/2016.
- *Woodford C. (18/03/2016): Neural Networks,* http://www.explainthatstuff.com/introduction-to-neural-networks.html, consulted on 14/01/2017.

ACKNOWLEDGEMENTS

After five years I have finally achieved this important goal. The journey has been difficult sometimes, but with great satisfactions in the while.

My first acknowledgement goes to my academic advisor, Professor Marco Corazza, for his continuous support during these months, for his patience, motivation and knowledge. His direction helped me in my research.

Secondly, I would love to thank my parents, Marisa and Gianfranco, and my grandparents, Arturo and Lauretta; they supported and sustained me during these five years, living with me joyousness as well as difficulties, and loving me constantly.

Another heartfelt acknowledgement goes to Giovanni and my beloved friend Valentina: they encouraged and always believed in me.

Finally, I would like to thank all my friends and fellows for sharing this journey with me.