POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Integrating news sentiment analysis into quantitative stock trading systems

Advisor
prof. Luca Cagliero

Co-advisor:
prof. Paolo Garza

Candidate
Vincenzo Savarese

Academic Year 2018-2019

Abstract

Wouldn't it be great if we could teach a computer to perceive the underlying feeling of a text and exploit that information in stock price forecasting? Are there words that have a more significant influence on the daily stock price change?

This research study addresses these questions in the context of stock price forecasting based on news sentiment analysis, using a quantitative trading system which relies on Machine Learning techniques. Many previous studies focused on studying the impact of news on trading system performance by applying them to limited period or to a limited number of stocks. Hence, there is the need for extensive empirical studies to evaluate the potential of news sentiment analysis in quantitative trading system. More specifically, the most significant challenge is to understand how to combine and exploit news information into Machine Learning models.
A quantitative trading model implements trading strategies relying on quantitative analysis, which applies both mathematical computations and statistical indicators to identify patterns in market stocks. Sentiment analysis, on the other hand, is a text mining process that tries to extract subjective information from text. A text is considered "positive" if, for instance, the sentiment analysis process detected more positive than negative words.
The quantitative trading system, together with the sentiment analysis enhancement, exploits *Supervised Classification* techniques that take decisions based on some *independent characterising* variables. The core of this research is the choice of *news-based* variables, the ones deriving from the news sentiment analysis phase. The more those variables are descriptive, the higher the probability of making a profit. This choice is divided into two phases: *a priori variable* definition and *final news-based features* selection. In the first phase, we define some variables that we expect to reflect best news sentiment. Later, in the second phase, statistical measures are applied to select most representative and predictive variables in order to maximise their influence on the forecasting model. To infer the effectiveness of the news sentiment addition, we did experiments with and without the sentiment. Results with news sentiment addition significantly outperformed the ones without the addition, with an average total profit peak of increment of nearly 30%.

Acknowledgements

Eccomi giunto alla fine di questa tesi e di questi anni di Università, che mi hanno insegnato tanto sia dal punto di vista umano che professionale. Questi anni da studente presso il Politecnico di Torino hanno richiesto da parte mia un grande sacrifico ed impegno. Il trasferimento in un'altra città, a differenza di quanto si possa pensare, è sì un'ulteriore difficoltà da affrontare, ma mi ha dato la possibilità di maturare come persona e di conoscere degli amici, non colleghi, che mi hanno sostenuto ed aiutato a raggiungere questo tanto desiderato traguardo.
Vorrei quindi dedicare questa pagina a tutte quelle persone che, in un modo o nell'altro, mi hanno supportato e sopportato in questi ultimi anni.
I miei più sinceri ringraziamenti vanno alla mia famiglia, che è riuscita a sostenermi nei momenti di maggiore difficoltà e senza la quale non sarei dove sono. Il loro aiuto e supporto è stato costante e pertanto meritano di gran lunga questi ringraziamenti.
Un grazie va anche ai miei amici storici ed ai miei cugini e parenti più stretti, che hanno saputo trasmettermi il loro affetto anche a distanza di centinaia di chilometri.
Infine, un ringraziamento speciale, va a tutto il gruppo con il quale ho condiviso il giorno e la notte di questi ultimi due anni. Grazie perché siete stati la mia vita sociale, il mio gruppo di uscite ed il mio supporto psicologico nelle varie difficoltà degli esami. Grazie perché mi avete dimostrato che anche delle semplici serate attorno ad un tavolo di casa possono rimanere nella storia per la loro memorabilità. Grazie per le tante notti insonni a girovagare senza meta per la città, insegnandomi che non conta il come o il dove, ma il "con chi" si trascorre il tempo. Grazie, inoltre, a chi mi ha accolto in casa sua ogni volta che ne avessi bisogno, senza mai far problemi.
Spero vivamente che, qualunque sia il luogo in cui vivremo in futuro, riusciremo sempre a trovare il tempo per continuare a coltivare questa bellissima amicizia.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This research work aims to study both the influence and the impact of online web articles on a quantitative stock trading system. Thanks to increasing availability of electronic information of last years, several types of research were carried out in this field. Many previous studies focused on studying the impact of news on trading system performance by applying them to limited period or to a limited number of stocks. Hence, there is the need for extensive empirical studies to evaluate the potential of news sentiment analysis in quantitative trading system. More specifically, the most significant challenge is to understand how to combine and exploit news information into Machine Learning models. We analysed all American companies belonging the *Standard & Poor's index* (little more than 500) between the years 2007 and 2017, exploiting a news corpus of about 9 millions of distinct web articles. On the one hand, this vast number of articles allows great companies coverage, but on the other hand, could make prices to be more unpredictable, thus forecasting to be more difficult.
As might be guessed from this research title, two macro-topics were addressed: *sentiment analysis* and *quantitative trading*.
Sentiment analysis, as the words itself says, is a text mining process that tries to extract subjective information from text. This process, also known as opinion mining, tries to identify and label a text according to its tone. An article is considered "positive" if, for instance, the sentiment analysis process detected more positive than negative words.
Sentiment analysis is a gill of the Natural Language Processing (NLP). Natural language processing aims to train computers to understand and correctly process human language. In the last years, artificial intelligence (AI) and deep learning techniques have done giant strides and, as a consequence, sentiment analysis is being used in more and more applications and sectors. Furthermore, when the amount of textual data is enormous, manually reading and extracting useful information

from a text cannot be considered a feasible option and automatic techniques, as it is sentiment analysis, must be used.
When speaking about forecasting stock prices, two schools of thought exist: *technical analysis* and *fundamental analysis*. Technical analysis tries to predict stock price change according to some technical indicators that focus on stock's price and volume patterns. Fundamental analysis, instead, believes that its intrinsic value influences stock's price and therefore, events such as either financial news or macroeconomic factors contributes to stock price variations. Our research has a mixed approach since we exploit both *technical indicators*, that recognise price and volumes patterns, and *news-based indicators*, that comes from a sentiment analysis carried out on web articles.
However, forecasting price variations, regardless of the school of thought, has always been a challenging task. Indeed, the *efficient-market-hypothesis* (EHM)[1], one of the most famous financial economics theory, states that the market prices reflect all available information, meaning that a trader cannot in any way "beat" the market and make substantial gains. This theory, however, was found not to be wholly accurate and, later, the author proposed a revised less restrictive version.
In this scenario, the quantitative analysis applies both mathematical computations and statistical indicators to identify patterns in market stocks. A quantitative trading model applies trading strategies relying on quantitative analysis. This model carries out forecasting decisions based on some characterising variables. This model generates an output which is then used in the trading phase. The generated output carries out a prediction for each stock market date analysed and labels each day as *buy, sell* or *hold*. *Buy* suggests to the virtual trader to buy the analysed stock; *sell*, as it can be imagined, suggests to sell the current stock and *hold*, instead, means that the model did not detect any important changes in the stock price and thus suggests to do nothing.
The choice of the model's characterising variables is crucial because the resulting model could not be able to make profitable decisions. Moreover, applying some trading strategies to a "wrong" model output could result in a significant loss of invested money.
In our case, the choice of those variables is the result of both the news sentiment analysis and quantitative analysis phases. The latter was already implemented in this quantitative stock trading system. Thus we will not go into the details. The core of this research is precisely the choice of those *news-based* variables, resulting from the news sentiment analysis. This choice is divided into two phases: *a priori variable* definition and *final news-based features* selection. In the first phase, we define some variables that we expect to reflect best news sentiment. Later, in the second phase, statistical measures are applied to select most representative and predictive variables in order to maximise their influence on the forecasting model.

What also distinguishes this study from past researches, is the criterion chose for assessing results. Past researches mainly focused on some criteria, such as recall or accuracy rates. Those criteria, however, do not indicate if the model contributed to making a profit, which is one of the major points for investors. Our methodology, instead, evaluates the trading strategy with the *total profit* measure.
To infer the effectiveness of the *news sentiment analysis* module, we did experiments with and without the module. Results with news sentiment addition outperformed the ones without the news sentiment addition, reaching a peak of increase in the average total profit of about 30%.

Overview. This document is organized as follows. Chapter 2 gives a survey on the *state-of-the-art*. The structure of the sentiment analysis module is depicted in Chapter 3, together with sentiment analysis techniques description. It follows a comprehensive description of correlation analysis in Chapter 4.
Chapter 5 contains the quantitative trading system description and it illustrates the experiments we did. In Chapter 6 are shown experiments' results. Finally, in Chapter 7 conclusions and future works are set out.

Chapter 2

Related works

In the last decades, sentiment analysis and, more generally, text mining approaches were deeply explored in the context of stock price prediction. In this section, we will briefly describe both related works key points and their differences and similarities with our research study. The main aspects that can be used to make comparisons between the various works are:

- *Dataset*: textual and stock price data used in the analysis.
- *Period and time-frame*: time interval to which the analysis was extended and prediction periodicity (daily, intra-day, long-term).
- *Machine learning algorithm*: algorithms chose to perform stock market predictions.
- *Forecast type*: the most used forecast types are categorical and regression. Categorical approaches try to understand price change and to categorize it with labels such as *UP, DOWN* and *STAY*. In regression problems, instead, the objective is to try guessing the stock close price.
- *Evaluation method*: methods and criteria used to evaluate the model.

Our work exploits a huge news corpus of web articles (little more than 9 millions of distinct news), collected between years 2007 and 2017. Several machine learning algorithms are used (*Support Vector Machine, Naive Bayes, Multilayer Perceptron and Random Forest classifier*), with a categorical forecast type. Results are assessed applying trading strategies and checking measures such as average total profit and average profit per operation. None of the following related works is entirely comparable with our research because of several factors: unavailable datasets, different

periods, time-frames and evaluation methods.
We mainly focused on closely related works which exploit sentiment analysis techniques as an enhancement for a short-term stock market prediction, by using machine learning algorithms.
Our work is pretty similar to the one proposed by *Ichinose et al.*[2]. They proposed a method to predict the *Nikkei Stock Average* (one of the most popular stock market index), exploiting news found on the web. There are two substantial differences between their and our method: dataset and machine learning algorithm used. Their dataset consisted of 44164 web articles for 212 days, while with regards to machine learning algorithms, they used only *SVM* and *linear perceptron*.
Schumaker et al.[3] used an *SVM* algorithm to predict *intraday* stock price. Although they extended the analysis on all companies belonging to the *SP500* index, they picked a minimal period (*26-Oct-2005* to *25-Nov-2005*). Their results in terms of *Accuracy* were good (nearly 59%) considering that this work was one of the first approaching textual analysis with stock market prediction.
Hagenau et al.[4] did an interesting research in the sentiment analysis field. They investigated the impact of "*news momentum*" in predicting stock prices at medium-term (one week, two weeks, four weeks and eight weeks). They also used two different trading strategy to evaluate their results. Unfortunately, even this work is not comparable to ours. They collected news from *DGAP* (Deutsche Gesellschaft für Adhoc-Publizität) and *Reuters*. *DGAP* collection of news is no longer available, while *Reuters* corpus was collected between the years 2003 and 2009. Moreover, their time-frame was long-term while our is daily. Always *Hagenau et al.*[5], in the same year, published another research work, investigating daily price change using *market feedback*. They reached a very high *Accuracy* ($\sim$ 76,3%), even though their approach is tough to replicate and compare. Their news base comprised corporate announcements from Germany and the UK. For each announcement, they traced the company and the corresponding stock price.
A different approach was followed by [6]. *Ding et al.* tried to extract *events* from data. An event has an *Actor*, an *Action* and an *Object*. For example, let us consider this news title "*Apple sues Microsoft*". Using a term-level representation could make it difficult to predict stock movements of both companies, while with an *event-based* representation, the actor and the object can be better captured. Like our work, they extended the analysis to all individual company belonging to the *SP500* index. They achieved a good accuracy score ($\sim$ 65%), but there was no mention if the whole process leads to a positive profit or not.
Shynkevich et al.[7] faced a different problem. They investigated possible advantages that could derive from different levels of news relevance. They reached *Accuracy* peaks of 80%, but their news corpus is no more available. One of the most recent works [8], it is also one of the most comparable except for the fact that they

analysed a single stock company (Apple Inc) and used *Accuracy* measure to asses the quality of their work.
A lot of other researches were examined, but those works differed too much from our intentions and objectives and we took them only as point of reflections. *Kawabata et al.*[9] studied news impact on foreign exchange market, which is vastly different from the stock market. Both *Shou et al.*[10] and *Geva et al.*[12] used an *intra-day* time-frame in the forecasting process, making their works completely not comparable with ours. *Nguyen et al.*[13] objective is similar to our research in terms of objectives but differs for many aspects: news coming from social media (Yahoo message boards, no longer existing), very limited number of stock examined (five stocks) and different evaluation method (accuracy measure).

Chapter 3

The sentiment analysis module

3.1 Introduction

Many research studies applied sentiment analysis techniques in quantitative trading, some of which with really encouraging results in terms of prediction accuracy. This work tries both to take inspiration from these research papers and to generalise their experiments, in order to be less dependant on the circumstances of each company. For this reason, we analysed all companies belonging to the *Standard & Poor's* index (also known as S&P500 or simply S&P) and their stock trends in the last 11 years.
The main goal of this work is to develop a *news sentiment analysis module* to be integrated into an already developed quantitative trading system. A quantitative trading system exploit Machine Learning techniques to make stock price forecasting. More specifically, the trading system uses *Supervised Learning*[29], which is divided into two categories: Classification and Regression. The former forecasts the class (category) the data belongs to, while the latter predicts a numerical value relying on previously observed data. We chose *Classification* techniques since we are interested in labelling new portions of data based on some *independent characterising* variables. Those variables, also known as *features*, help the classification algorithm build a model which is used for predictions. The core of following two chapters is to choose and select features reflecting the news sentiment analysis (*news-based features*).
Both the architecture and fundamental modules of the quantitative trading system are shown in Chapter 5. Since the *news sentiment analysis module* can be seen as a standalone module, in this and the next chapter we will describe its functioning, while its integration in the quantitative trading system is explained in Chapter 5.
The news sentiment analysis module, as shown in Figure 3.1, can be logically subdivided into two phases:

- *Sentiment Analysis phase*: this phase is explained in this chapter, and it consists of preparing, filtering and evaluating the sentiment of a collection of news. At the end of this process, a subset of *a priori* features is computed and provided to the next phase: the *Correlation Analysis* phase.

- *Correlation Analysis phase*: statistical measures are used to check the correlation between features provided by the previous phase and stock price change. The main goal of this phase is to select the best *news-based* features to give to the *Classification model training* stage.

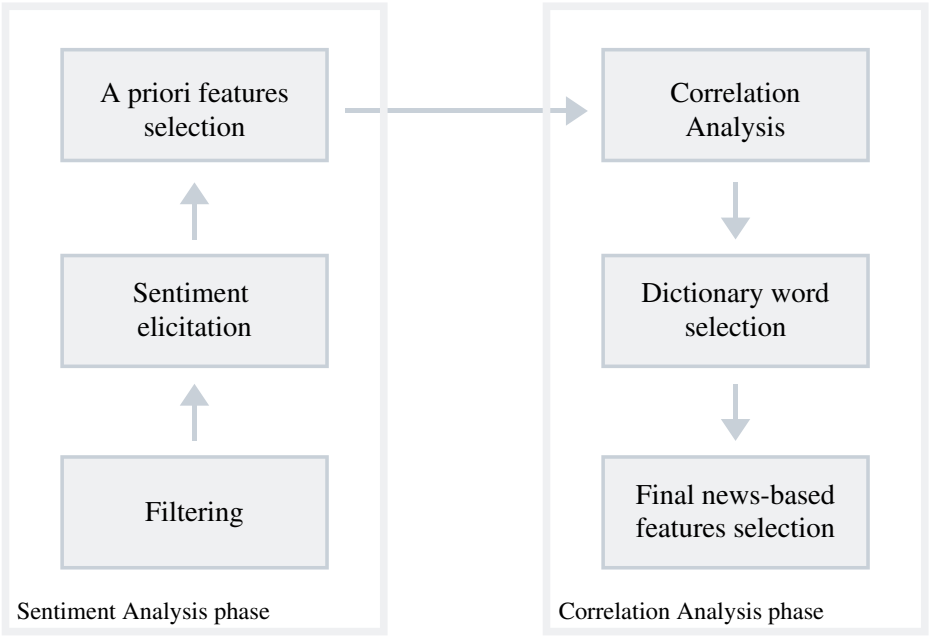

Figure 3.1: Logical pipeline describing the sentiment analysis module.

Section 3.2 is subdivided in two parts, corresponding to the company and news collection part. First of all, *SP500* companies data were collected, together with their historical prices. Subsequently, it is explained how the news corpus was fetched and prepared for the sentiment elicitation phase.

3.2 Data identification and collection

3.2.1 S&P500 companies historical prices collection

The complete list of companies belonging to the *S&P500* index was fetched from Wikipedia [15]. This list counts a few more than 500 American companies. In addition to the company name, we also downloaded many other data:

- *Ticker Id*: a ticker symbol that identifies the company.
- *Ticker Name*: the company name.
- *Sector*: the sector in which the company operates.
- *Sub Industry*: the subindustry in which the company operates.
- *Foundation*: company foundation year.

These data were essentials in the filtering phase (Section 3.3.1).
Table 3.1 shows an extract of the file containing *SP500* companies information.

Table 3.1: S&P500 company information extract.

TickerID	Ticker Name	Sector	Subindustry	Founded
MMM	3M Company	Industrials	Industrial Conglomerates	1902
ABT	Abbott Laboratories	Health Care	Health Care Equipment	1888
ABBV	AbbVie Inc.	Health Care	Pharmaceuticals	2013
ALLE	Allegion	Industrials	Building Products	1908
...	...	...	...	...

Quantitative trading makes excellent use of statistical indicators to identify patterns in historical stock prices and exchanged volumes. For this reason, it is necessary to retrieve *daily* stock prices for all companies belonging to the S&P500 index, from 2007 to 2017.
Yahoo Finance [16] provides these historical data for free by using their API. Unfortunately, not all company historical prices were available and so the number of companies to analyse decreased from 506 to 501. For each company, we used the following measures to describe daily price variations:

- *Date*: stock price date
- *Open*: stock price at the market's opening.
- *High*: the highest stock price registered on that day.
- *Low*: the lowest stock price registered on that day.
- *Close*: stock price at the market's closing.
- *Adj Close*: "Adjusted closing price amends a stock's closing price to accurately reflect that stock's value after accounting for any corporate actions." [18].
- *Volume*: exchanged volumes registered on that day.

Each company has its file containing historical data. Table 3.2 shows how *Apple Inc* file is organised. Figure 3.2, also known as the *candlestick* chart, shows *Apple Inc* price oscillation between 2014 and 2017.

Table 3.2: Apple Inc company historical prices extract.

Date	Open	High	Low	Close	Adj Close	Volume
2007-01-03	12.327	13.368	11.700	11.971	8.016	309579900
2007-01-04	12.007	12.278	11.974	12.237	8.194	211815100
2007-01-05	12.252	12.314	12.057	12.150	8.136	208685400
2007-01-08	12.280	12.361	12.182	12.210	8.176	199276700
...	...	...	...	...	...	...

3.2.2 News Corpus download and organisation

Our news corpus is composed of more than 9 billion of web text news between 2007 and 2017, nearly 9 Gb of text only. This corpus includes major key events news articles of almost all American companies. We collected this data from one of the leading American news aggregators, *Reuters.com* [14].
The *Reuters* website [14] grants public access to its enormous web news archive, containing online web articles from 2007 to 2018. The year 2018 was not included in our downloading process since, at downloading time, was not concluded yet. The archive was organized in a list of indexes, divided per year, each of them representing a day. Each day was a URL to the *day archive page*. Once landed in the *day*

Figure 3.2: Candlestick chart of Apple Inc (APPL). [19].

archive page, there were listed all articles published in that day; each article was a URL to the news page, containing the actual news content.
The downloading process was very long and took almost a month of h24 downloading. We used several computers, each one with several processes making HTTP requests. To prevent IP-blocks by the server, each HTTP request was interspersed by a random waiting time and by a random user-agent selection. Moreover, each computer was "virtually" located in a different part of the world, thanks to a VPN. This downloading process was split into three phases:

- *Indexes downloading*: an URL identified each "final resource" (the article content). This URL was copied and saved in a file, in order to reach the "final resource" without going through the Reuters indexes archive every time. The URL file was structured straightforwardly: two columns, corresponding to the timestamp/date and the news URL, respectively.

- *News preprocessing*: "final resource" is an HTML web page and, in order to have the text free from unnecessary HTML tags, it must be preprocessed.
  All HTML tags, such as <p> or <td>, were removed. Moreover, the BeautifulSoup Python library [17] was used to exactly localize and download the article body, excluding additional text.
  Finally, any carriage return character (\r or \n) is deleted from the text.

- *News saving*: article text is saved in a *csv* file. This file will contain the whole collection of online web articles.

Each row of the news corpus file represents an online web article. At most, four categories of information are stored. Sensitive information, such as author name, location or users comments, were discarded. The four categories are organized in 4 columns:

- *Timestamp*: the exact moment in which the article was released.
  The timestamp is in the format *YYYYMMDD HH:MM AM/PM EST/EDT*.
  If the timestamp is not available, it is replaced by the date. Date format is the following: *YYYYMMDD*.

- *Title*: News title.

- *URL*: the web address of the news.

- *Body*: News body. It could be empty if the online web article were not available at downloading time

In the Table 3.3 there is an example of how the news corpus is organized.

Table 3.3: News Corpus extract.

Timestamp	Title	URL	Body
20070215 06:53 PM EST	Agilient earnings...	reuters.com/...	Electronics testing...
20070215 10:44 PM EST	Predators acquire...	reuters.com/...	The Nashville Predators...
...	...	...	...
20071128	Alpha Bank...	reuters.com/...	–
20071128	US Stock Market...	reuters.com/...	U.S. stocks rallied...
...	...	...	...

Next section details the "first phase" of the *sentiment analysis module*.

3.3 Sentiment elicitation phase

This section is structured as follows: each subsection will describe both its functioning and which kind of files are generated in output, ready to be used in the next stage.

3.3.1 News filtering stage

Due to the considerable news corpus's size and weight, it was decided to split it in one news file for each company. In this way, each company will have its news file containing only the articles strictly related to the company.
The rigidity or less of the filtering process could lead to imprecise results.
To better get into the problem, let us define two variables:

- *Target*: the string to find.
- *Phrase*: the phrase in which the *target* must be looked for.

Let us consider the Apple company filtering procedure. Apple data are obtained from the file downloaded in Section 3.2.1. Now let us assign to the *target* variable the Apple "Ticker Name" value ('Apple Inc'), and to the *phrase* variable the first body article to analyse. Several issues could arise:

- The *target* variable could contain more than one word. How to proceed? Look for matching the whole string or every single word? Single words might either not make sense without the others or, worse, be equal to widely used words, resulting in a continuous *word match* (i.e. the word "Company" in the "3M Company" name)
- Look for an exact match or even for a partial match?
- Case sensitive or insensitive research?

The *first filtering strategy* adopted was the following: take the *target* string as it is, look for a partial match with a case insensitive research. Results were inaccurate and contained many unrelated articles due to the partial match assumption.
The *second filtering strategy* consisted of choosing the exact match and the case sensitive assumption. The exact match was achieved defining a simple regular expression, using Python library *re* [20]. The function below shows how second filtering assumptions are fulfilled.

```
import re

def exact_match(phrase, target):
return re.search(r"\b{}\b".format(target), phrase) is not None
```

Code 3.1: Regex function used for second filtering strategy

The Wikidata improvement

Once defined how accurate comparison should be made, let us describe which information to use in the *filtering strategy*. In addition to stock information downloaded in Section 3.2.1, we used *Wikidata* to enhance our information quality.
"*Wikidata* is a free and open knowledge base. Wikidata acts as central storage for the structured data of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wiktionary, Wikisource, and others." [21].
Thanks to their API's, is it possible to query their database and get much other useful information about the query.
In particular, for each successful query (some companies could not be present inside the Wikidata database) we collected:

- *Aliases*: company name aliases
- *Industry*: the industry in which the company works

This extra information, in addition to the one collected in Section 3.2.1, will be combined to obtain different levels of news relevance. We defined three different configurations:

- *Targeted configuration*: the *target* was a list of words composed by only the company name (*Ticker Name*) and possible aliases found on Wikidata. This configuration, along with the *second filtering strategy*, is suitable for focused research.
- *Medium configuration*: the *target* was a list of words composed by the company name, sector and subindustry.
- *Full configuration*: this configuration adds to the medium one data coming from Wikidata (aliases and industry).

Filtering Methodology

We chose the *second filtering strategy* with the *targeted configuration*, in order to maximize pertinence between companies and articles in which a company is mentioned.
The algorithm works as follows: for each company in the *SP500* company list, Wikidata data are fetched. Then, according to the configuration chosen, a *target_list* is prepared, containing all keywords that identify the company. For each news contained in news collection downloaded, if the article contains one "keyword" among those present in the *target_list*, that news is saved in a file corresponding to the company considered. At the end, each company will have a file containing articles that talk about that company.

3.3.2 Sentiment extraction stage

To identify the sentiment of each news, we used a Dictionary based approach. We relied on Loughran-McDonald's research [22]. They used a master dictionary containing Finance specific words. Each word has its *polarity* score. *Polarity*, in sentiment analysis, is the emotion (sentiment orientation) expressed in the text. According to its *polarity* score, each word was assigned to a different dictionary, and seven distinct dictionaries were created.
Among those defined by their research, we took the positive and the negative dictionaries; the negative contains 2353 words, while the positive only 353.
The sentiment evaluation process is pretty simple and straightforward. Basically, for each article of each news company file (the ones coming from previous *news filtering stage*), two lists of words were created: the *positive list* and the *negative list*. The former contains positive words found in that article while the latter stores the negative ones. These lists will be saved in a new file which objective is to store the sentiment of each article of the examined company. Each row of the company news file corresponds to the relative row in the sentiment file. An example of the company sentiment file is shown in Table 3.4.

3.3.3 A priori features design stage

Once completed both the filtering and the sentiment evaluation process, it was time to start defining some *a priori* features. The definition of these features is a very crucial point since from a meaningless feature definition nothing more than a weak and insignificant output can be expected. Furthermore, it is essential to monitor *news-based features* contribution in order to understand if they bring to positive or

Table 3.4: Apple sentiment file extract.

Date	Positive List	Negative List
2007-01-03	Despite	Argue, Deny, Wrong
2007-01-03	Able, Despite, Easy, Better	Break
2007-01-04	Gains	Abruptly, Concern, Deny
2007-01-04	Despite, Better	Lowest
2007-01-05	Gains	Break
...	...	...

negative results. *News-based features* contribution is shown in depth in Chapter 6. As the "a priori" adjective suggests, this is a preliminary definition phase since, in Chapter 4, we will adopt statistical measures to study the significance of these features on *next-day stock price change*.

Features definition

We defined nine *apriori* features, which refer to each article:

- TPW: the total number of positive words found in that article.
- TNW: the total number of negative words found in that article.
- $DBTPNW$: the difference between the total number of positive and negative words found in that article ($TPW - TNW$).
- $TSPN$: the total number of *single positive* news found on that day. An article is *single positive* if it contains only positive words. This feature has a daily scope since many articles could be found in a day.
- $TSNN$: the total number of *single negative* news found on that day. This feature is exactly the opposite of *TSPN*.
- $DBTSPNN$: the difference between the total number of single positive and negative news ($TSPN - TSNN$).
- $TSPP$: the total number of *single positive* phrases found in that article. The sentiment calculated in Section 3.3.2 is at article level, not at phrase level. For this reason, we subdivided the article in phrases and, for each phrase, the sentiment was dynamically calculated. A phrase is *single positive* if it contains

only positive words. This feature has a daily scope since many articles could be found in a day.

- *TSNP*: the total number of *single negative* phrases found in that article. This feature is exactly the opposite of *TSPP*.
- *DBTSPNP*: the difference between the total number of single positive and negative phrases ($TSPP - TSNP$).

In addition to these features, one last measure was taken into account, named *next-day variable*. This measure is the stock next-day price variation, in percentage. To better understand this essential measure, let us consider Apple stock (Table 3.2). Assuming we are analysing the 2007-01-03 market date, let us define two variables:

- *nd_close*: next day close price. In our example is: 12.237143
- *nd_open*: next day open price. In our example is: 12.007143

Next-day measure is calculated in the following way:

$$Next - day = \frac{nd\_close - nd\_open}{nd\_open} * 100 \tag{3.1}$$

$$Next - day = \frac{12.237143 - 12.007143}{12.007143} * 100 \tag{3.2}$$

$$Next - day = 1.92 \tag{3.3}$$

General considerations

Before deepening into how these *a priori* features were calculated, there are some important considerations to do.

Article publication date *vs* market date. First of all, these *a priori* features should be calculated only when the market is open. On the other side, online web articles might have been published on any calendar date, and so we have to decide what to do with articles published when the market is closed. Furthermore, an even more peculiar situation may occur. Let us imagine a company that becomes increasingly important and news aggregators starts speaking about this event. However, our fictitious company could not be listed yet on the *S&P500* index.
In this scenario, what should we do? We could either accumulate those articles

(and their particular features) until the first company market date is reached or we could ignore them and start collecting *a priori* features from the first company market date onwards.
We chose to define a *temporal window* of 7 days. For each market date, we will consider only articles published at most seven days before. In this way we have a trade-off between being positively influenced by articles published someday before each market date, that could influence stock price variation, and not being adversely affected by articles published several months or even years before.

Duplicated news. We noticed that, sometimes, several updates of the same news were published on the same day. This behaviour could result in an inaccurate features evaluation. For this reason, subsequent updates of the same news, expressing the same sentiment of the first one (already counted), will be discarded. On the contrary, if an article update brings some new information, it contributes to the calculation of the features.

Algorithm implementation

Here is described at a very high level how *a priori* features are calculated. The algorithm generates a file for each company, containing a row for each market date. Each row represents a *snapshot* of both how the price changes in the following day ($Next - day$ variable) and whether and how sentiment features captured this information.
Among the comprehensive set of *Python* libraries used, a pair of them deserve a special mention and brief explanation:

- *Pandas*: this library [23] provides several high-performance data analysis tools and methods for dealing with very large *csv* files.
- *NLTK*: Natural Language Toolkit helps *Python* programs to work with human language data [24]. This library and in particular its "tokenize" package, was used for the *TSPP* and *TSNP* features calculation. Subdividing a text in phrases may be more complicated than it might expect. For this reason, *tokenize* package uses an unsupervised algorithm to build a model for abbreviation words, collocations, and words that start sentences; and then uses that model to find sentence boundaries.

The algorithm takes as input each file generated so far. For each company that has to be analysed, and for each article contained in the company news file, the

article date is compared with the *previous* date considered. Until the article date is equal to the *previous date*, the algorithm computes and accumulates *a priori* features. When the article date is different from the previous one, both the date and *a priori* features are saved in the output file if and only if both the market is open and articles are not flagged as "to skip". A bunch of articles can be flagged as "to skip" if their publication date is not inside *temporal market windows* (described in Section 3.3.3). In both cases, features and other data structures are reset, ready for the next iteration. At the end of the algorithm execution, each company will have a file similar to the one shown in Table 3.5
For a matter of space, features' names are shortened in: *I + the number* representing the order in which they have been defined (*TPW* corresponds to *I_1*, *TNW* corresponds to *I_2* and so forth).

Table 3.5: Apple a priori features extract.

Date	Next-day	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
2007-01-03	-2,89	48	96	-48	0	0	0	36	61	-25
2007-01-04	1,92	47	62	-15	0	0	0	42	41	1
2007-01-05	-0,84	39	41	-2	1	0	1	37	28	9
2007-01-08	-0,57	56	47	9	2	0	2	47	36	11
2007-01-09	7,08	57	64	-7	3	1	2	43	40	3
...	...	...	...	...	...	...	...	...	...	...

Chapter 4

Correlation analysis and final news-based features selection

In this chapter, we will deepen the second phase of the *Sentiment Analysis* module, Figure 3.1. This module's phase has a critical importance on the outcome of the quantitative stock trading system. Indeed, if we could figure out which are the *news-based* features statistically correlated with the *next-day stock price* variation, we could imagine that if, for instance, our *TPW* feature grows (as a consequence of much positive news), so it will do the company stock price in the next market date.
In the next sections, are explained in depth which correlation index we used, which experiments we made and which are their results.

4.1 Preliminaries

Correlation is a statistical measure that indicates the extent to which two or more variables fluctuate together [25], meaning that a change in the first variable reflects a change in the second variable. Let us assume we are evaluating the correlation between two variables.
When to the increase (decrease) of the first variable correspond the increase (decrease) of the second one, we have a *positive correlation*. When to the increase (decrease) of the first variable correspond the decrease (increase) of the second one, we have a *negative correlation*. Correlation, however, must not be confused with causation. The reason why two variables increase or decrease in parallel could be attributed to a third unknown factor.
A correlation coefficient is a numerical measure of some types of correlation [26]. We used the *Pearson Product-Moment Correlation Coefficient* to calculate the correlation of our data.

4.1.1 Pearson Product-Moment Correlation coefficient

The Pearson product-moment correlation coefficient (or Pearson correlation coefficient, for short) is a measure of the strength of a linear association between two variables and is denoted by r [27]. Let us consider two arrays of data points, called $a1$ and $a2$. The Pearson product-moment algorithm tries to draw a line of the best fit through the values of $a1$ and $a2$.
Pearson correlation coefficient is a measure of how far away all $a1$ and $a2$ data points are from the best fit line previously drawn.

Pearson range values The *Pearson correlation coefficient*, r, can assume values between -1 and +1.

$$\text{Pearson correlation} = \begin{cases} \text{positive}, & \text{for } 0 \leq r \leq 1 \\ \text{none}, & \text{for } r = 0 \\ \text{negative}, & \text{for } -1 \leq r \leq 0 \end{cases}$$

Figure 4.1 shows how positive, negative and not correlated dataset could be, respectively.

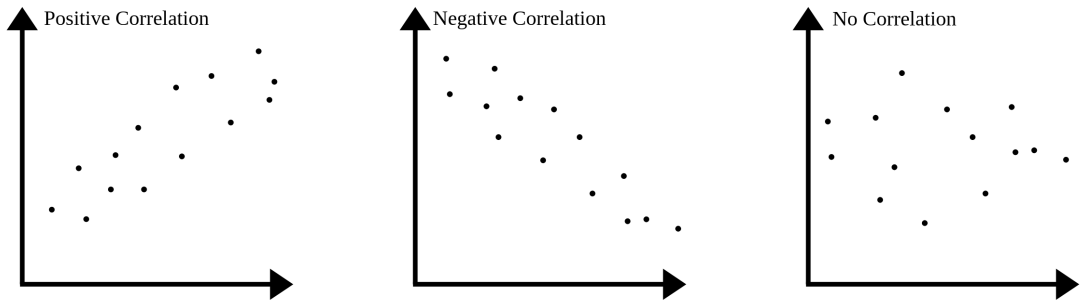

Figure 4.1: Pearson Product-Moment Correlation examples

Closer is the Pearson correlation coefficient to the upper bound and the lower bound (+1 and -1), higher is the association strength between the two arrays of values. Table 4.1 can be used to estimate the strength of the association.

Pearson Correlation mathematical definition. Given a pair of variables (X, Y), the Pearson correlation coefficient is obtained with the following formula:

$$\rho_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} \tag{4.1}$$

Table 4.1: How to interpret the Pearson correlation coefficient.

Strength of Association	Positive coefficient r	Negative coefficient r
Low	0.1 to 0.3	-0.1 to -0.3
Medium	0.3 to 0.5	-0.3 to -0.5
High	0.5 to 1	-0.5 to -1

where $cov(X, Y)$ is the covariance of the two variables (X, Y) and σ_x, σ_y are their standard deviations.
Covariance between two variables (X, Y) is calculated as:

$$\text{cov}(X, Y) = \text{E}[(X - \text{E}[X])(Y - \text{E}[Y])] \tag{4.2}$$

where $\text{E}[X]$ is the *expected value* (mean) of X and $\text{E}[Y]$ is the *expected value* (mean) of Y.
The standard deviation of a variable is the square root of its variance. Let μ be the mean of X:

$$\mu = \text{E}[X] \tag{4.3}$$

Standard deviation of variable X is:

$$\sigma_x = \sqrt{\text{E}[(X - \mu)^2]} \tag{4.4}$$

4.2 News correlation analysis and variable definition

As already said, the main goal of this analysis is to find a possible statistical correlation between *news-based* features, calculated in Chapter 3, and *next-day stock price variation*. To exploit the Pearson correlation, we must define two variables, upon which the correlation will be evaluated. Variable values change according to the experiments performed, but their semantic field will always remain the same.

- *Sentiment variable*: it refers to the *a priori* features and assumes a normalized value according to the *a priori* feature value.
- *Next_Day_price_variation variable*: it refers to the *next-day stock price variation* (*Next-day* variable described in Equation (3.1)) and assumes a normalized value according to *Next-day*.

The experiments done can be divided into two categories: *feature-based* and *dictionary-based*. All *feature-based* experiments are conducted on *a priori* features. The objective of these tests is trying to find if companies price oscillation is "registered" by some features. If so, we can easily conclude that those features bring excellent information and can positively contribute to the classification process.
Instead, concerning *dictionary-based* experiments, a novel and really interesting approach were used. We tried to answer the following question: *can a single word be correlated to the price change?* Alternatively, *are there words more representative than others?* Those questions will be answered in Section 4.3.4
In Section 4.3, we will report and describe the experiments in the same order we made when we were approaching this analysis so that assumptions and choices can be better understood.

4.3 Experiments definition and results

The whole set of experiments was made on each company belonging to the *S&P500* index. Later, correlation results coming from each company are aggregated in a single file for studying how was the average result.

4.3.1 Correlation with all news-based features

Our first experiment belongs to the *feature-based* category, and its description is quite simple. We wanted to study the correlation between each single *a priori* feature and the *next-day price variation*. In this first attempt, we did not use the *Sentiment* and *Next_Day_price_variation* variables (those described in Section 4.2), the reason why our results were poor.
Keeping in mind that the Pearson correlation coefficient works with two lists of values, let us consider the output of Section 3.3.3, in particular Table 3.5. Each feature is coupled with the *Next-day* column, forming nine couples of values' lists. Correlations coming from each company file are aggregated by couples in order to have a medium correlation between the *Next-day* and each feature variable.
Table 4.2 shows how *Next-day* and each *a priori feature* are coupled for the Apple company.
As previously said, results were not satisfactory, showing a medium correlation, for each couple, close to 0. These results mean a "No-Correlation" between the two variables. These not exactly exciting results are due to a wrong definition phase because both the *Next-day* and *feature* variable can assume an infinite range of

Table 4.2: Pearson couple matching, referring to Apple *a priori* features.

$\rho(Next-day, I\_1)$		$\rho(Next-day, I\_2)$		$\rho(Next-day, I\_3)$		...	
Next-day	I_1	Next-day	I_2	Next-day	I_3	...	...
-2,80	48	−2.80	96	−2.80	−48	...	...
1,92	47	1.92	62	1.92	−15	...	...
-0,84	39	−0.84	41	−0.84	−2	...	...
...	...	...	...	...	...	...	...

values.
Following experiments take into account this issue by normalizing variables in a given range. Moreover, we are not interested in *how much* companies price goes up or down, but in understanding if some feature grows/decreases in parallel with the *Next-day* variable. Let us assume we have an *a priori* feature that, for positive values, we expect will reflect a positive variation in *Next-day* variable. Although *next-day* grows of 10% or 50%, it does not matter because the feature helped us in the correct choice.

4.3.2 Correlation with TSPN and TSNN

As stated in the title, in this experiment, we wanted to analyse the correlation between *next-day price variation* and *TSPN/TSNN* features.
This time *Next_Day_price_variation* and *Sentiment variables* are used. Their definition is the following:

- *Next_Day_price_variation variable*: it can assume three different values:

$$\text{Next\_Day\_price\_variation} = \begin{cases} 1, & \text{for } nd > 1 \\ 0, & \text{for } -1 \leq nd \leq 1 \\ \text{-1}, & \text{for } nd < -1 \end{cases}$$

  where nd is the *Next-Day* price variable.

- *Sentiment variable*: it can assume 2 different values: $+1$ or -1. *Sentiment* is equal to $+1$ if and only if $TSPN > 0$ and $TSNN = 0$. On the other hand, *Sentiment* is equal to -1 if and only if $TSPN = 0$ and $TSNN > 0$

Now that we have defined the two variables, we can extract their *Pearson correlation coefficient*. Once again, results coming from each company file are aggregated in a single file, containing as columns only the company name and computed correlation

between *Next_Day_price_variation* and *Sentiment* variable, in descending order. An extract of the result can be seen in Table 4.3, where *Next_Day_price_variation* name is shortened in "NDPV".

Table 4.3: Second experiment results extract.

Ranking	Company ticker ID	NDPV_Sentiment
1	EMR	0,6455
2	WELL	0,3273
3	RHI	0,3170
4	FMC	0,2704
5	GRMN	0,2467
...	...	...

Even in this case, results were not exciting, but a hint of medium-low correlation started to appear. In the next section, we will continue analysing correlation with other *a priori* features.

4.3.3 Correlation with difference features

As shown by many past works Chapter 2, the difference between TPW and TNW was proved to be very predictive. In this experiment, we got our best result among the experiments belonging to the *feature-based* category. As usual, let us define the *Next_Day_price_variation* and *Sentiment* variable:

- *Next_Day_price_variation variable*: the definition is identical to one described in the second experiment. For a matter of convenience, its definition is reported below:

$$\text{Next\_Day\_price\_variation} = \begin{cases} 1, & \text{for } nd > 1 \\ 0, & \text{for } -1 \leq nd \leq 1 \\ \text{-1}, & \text{for } nd < -1 \end{cases}$$

  where nd is the *Next-Day* price variable.

- *Sentiment variable*: it can assume 2 different values: +1 or 0. *Sentiment* is equal to +1 if at least one between $DBTPNW$, $DBTSPNN$ and $DBTSPNP$ is positive. On the other hand, *Sentiment* is equal to 0 if both $DBTPNW$,

$DBTSPNN$ and $DBTSPNP$ are negative.

We also experimented with a slight variation in the *Next_Day_price_variation* variable definition, but final results were comparable (even a little bit lower) with the ones described below. For this reason, we chose not to show this variation so as not to confuse the reader.
Results were acceptable and showed a medium correlation. A little extract of the aggregated file can be seen in Table 4.4, where *Next_Day_price_variation* name is shortened in "NDPV".

Table 4.4: Third experiment results extract.

Ranking	Company ticker ID	NDPV_Sentiment
1	EMR	0,7303
2	LW	0,5774
3	DWDP	0,5714
4	CLX	0,4714
5	FTI	0,4485
6	BHF	0,4082
...	...	...

However, companies with a correlation higher than 0,3 were 20 out of 501. For that reason, we decided to completely change our approaches, moving the attention not on feature values but the *dictionary words*.

4.3.4 Word correlation

This experiment is the first belonging to the *dictionary-based* category. As we did for the first experiment, we started with a very general approach, for later trying to extract useful information from results.
The experiment target is to study if there is a correlation at dictionary word level. As always, *Next_Day_price_variation* and *Sentiment* variable are defined as:

- *Next_Day_price_variation variable*: classic definition, already seen in experiment two and three.

$$\text{Next\_Day\_price\_variation} = \begin{cases} 1, & \text{for } nd > 1 \\ 0, & \text{for } -1 \leq nd \leq 1 \\ \text{-1}, & \text{for } nd < -1 \end{cases}$$

  where *nd* is the *Next-Day* price variable.

- *Sentiment variable*: it can assume 2 different values: +1 or 0. *Sentiment* is equal to +1 if the word found is positive. On the other hand, *Sentiment* is equal to 0 if the word found is negative.

Since we are working at word level, we need company sentiment files calculated in Section 3.3.2. The following example should clarify the selected methodology. Let us refer to Table 3.4, showing first rows of Apple sentiment file.

For each word found, a pair of values is generated; *Next_Day_price_variation* variable inferred from the *Next-Day* variable and the *Sentiment* variable deducted from the word polarity (positive or negative). Considering Table 3.4 first date and assuming that *Next-Day* variable for "2007-01-03" is 1,30%, the couple *NDPV-Sentiment* is shown in Table 4.5. Once the full sentiment file has been processed, the correlation is evaluated on these variables (*NDPV-Sentiment*).

Table 4.5: Example of sentiment for the Apple dataset.

NDPV	Sentiment
1	1
1	0
1	0
1	0
1	1
1	1
1	1
1	1
1	0

However, even in this case, correlation seemed to be inexistent, giving results between $[-0{,}1, 0{,}1]$. However, the negativity of those results gave us the possibility

to reflect and think about how to extract underlying information from our data.

4.3.5 Word-by-word correlation

This last experiment gave us the desired results. It belongs to the *dictionary-based* category, but technically it is a mixture between the *feature* and the *dictionary* one. Our experiment goal was a little bit sophisticated but, at the same time, very specific.
We wanted to study the correlation between each dictionary word and the *Next-Day* variable. From this brief and incomplete experiment goal description, three kinds of issues arise that deserves a short explanation. These descriptions are necessary to understand the definition of the experiment and its results better.

Issue n°1 This problem takes over in the following situation: let us assume we start analysing Apple company and the positive word examined is "Despite". *Next_Day_price_variation* and *Sentiment* variables definition can be the one described in Section 4.3.4. However, what is different from the previous experiment is that we are no more taking into account each word found, but the analysis is performed one word at a time. Moreover and most importantly, for each distinct pair (*date, word*), only one couple of values (*NDPV, Sentiment*) is generated, where "NDPV" is the *next day price variation* variable. Indeed, in the example below, the word "Despite" is found two times in "2007-01-03", but there is only one entry in the couple (*NDPV, Sentiment*). Taking as example the Table 3.4 and assuming that *Next-Day* variable for the date "2007-01-03" is +1,30% and for the date "2007-01-04" is −0,50%, the couple (*NDPV, Sentiment*) for the word "Despite" is shown in Table 4.6.

Table 4.6: "Despite" word analysis, for Apple sentiment file in Table 3.4.

NDPV	Sentiment
1	1
0	1

Assuming that we will continue until the file is finished, the *NDPV* column in Table 4.6 will range between $[-1, +1]$ without control, while the *Sentiment* column will be made up of an array of 1. The issue is all here. It is not possible to extract

the correlation of a couple of variables thus constituted, because the *Sentiment* standard deviation $\sigma_{Sentiment}$ is equal to 0.
It does not matter if the word is positive or negative; the standard deviation of an array of numbers equal to each other will always be 0.

Issue n°2 As proved in the *Issue n°1* description, we cannot calculate correlation with those assumptions. We need to introduce a variation in the *Sentiment* definition, in order to make the standard deviation calculation possible again. Under such conditions, let us consider the negative word "Argue" and let us refer to the usual Apple sentiment file (Table 3.4). Assuming that *Next_Day_price_variation* variable definition remains the same, let us change the *Sentiment* variable definition as follows: "the *Sentiment* variable will be equal to 1 if the considered word is found in the analysed article, 0 otherwise". This variation implies that if the word "Argue" was found 30 times out of 58334 articles (the realistic Apple news number), its correlation cannot fail to be adversely influenced by the 58304 articles in which the word was not present. Furthermore, from the conceptual point of view, it is wrong to penalize a word correlation when this word is not even mentioned in the article.

Issue n°3 One last scenario has to be analysed. Let us consider that, on a given date, ten different articles were published. From the Sentiment elicitation phase, we deducted that only 1 of them had a positive sentiment, while the remaining nine were considered negative (from the sentiment point of view). We expect that the *Next-Day* variable should go down, considering the number of negative news released. Let us assume that the *Next-Day* variable goes down and that the experiment aim is to analyse positive word correlation with *Next-Day* variable. Words correlation would be surely penalized by the number of negative news published on that day. Perhaps the positive article had a persuasive tone, and its words were very descriptive, but the number of negative news sway the analysis.

Definition of the experiment and results. This a little bit verbose introduction was necessary to understand better why we chose a definition as described below. Insights derived from issues description helped us in re-defining experiment goal:

1. Try a way to study the word correlation without being adversely influenced

by other words.

2. Each word must not be negatively influenced when not found in the text.

3. Positive (negative) word correlation with *Next_Day_price_variation* should not be influenced by the number of negative (positive) news.

To reach our goals, we exploited *news-based* features, laid down in Section 3.3.3. For each distinct article date, the lists of positive and negative words are saved and only when the date changes, *Next_Day_price_variation* and *Sentiment* variable are calculated. If, for instance, the word "Despite" is found three times in the same date (due to several positive articles), it will have only one entry in the couple (*Next_Day_price_variation, Sentiment*).
Next_Day_price_variation and *Sentiment* variables are defined as follows:

- *Next_Day_price_variation variable*: classic definition, already seen in experiment two, three and four.

$$\text{Next\_Day\_price\_variation} = \begin{cases} 1, & \text{for } nd > 1 \\ 0, & \text{for } -1 \leq nd \leq 1 \\ \text{-1}, & \text{for } nd < -1 \end{cases}$$

  where *nd* is the *Next-Day* price variable.

- *Sentiment variable*:

  1. if the word is positive and *DBTPNW* is greater or equal to 0, the *Sentiment* variable is 1, 0 otherwise.
  2. if the word is negative and *DBTPNW* is lower than 0, the *Sentiment* variable is 1, 0 otherwise.

Let us show an example in order to clarify any possible doubt. Let us consider both Table 3.5, showing Apple *news-based* features, and Apple sentiment file (Table 3.4). Let us assume we are analysing three words (clearly one at a time): "Despite, Deny and Gains". Their couples (*NDPV, Sentiment*) are shown in Table 4.7, in the same order (Despite, Deny and Gains). Remember that, for each date, only one couple (*NDPV, Sentiment*) is present. In table Table 4.7, "NDPV" column represent the *Next_Day_price_variation* variable.

The analysis, as always, is done for each company belonging to the *S&P500* index and for each dictionary word. What is different from the previous *filtering,*

Table 4.7: "Despite, Deny and Gains" word analysis, for dates between 2007-01-03 and 2007-01-05.

NDPV	Sentiment
1	0
1	0

NDPV	Sentiment
1	1
1	0

NDPV	Sentiment
1	1
0	0

sentiment and *a priori features selection* stages is that we made this analysis one year at a time. The reason will be explained in Section 4.4. For each company and each year we will have an output file containing "word-by-word" correlation for that company in that year.
Of course, another parameter must be taken into account: *word occurrence*. For this reason, each output file has those columns: "Word", "Word_Occurrence", "NDPV_Sentiment", where "NDPV" is *Next_Day_price_variation*. Results were really encouraging, showing a medium-high correlation of hundreds of words. Results are shown in Section 4.4, in which it will also be presented the methodology for choosing best and worst words.

4.4 Dictionary word selection

This section is devolved to the analysis of our best results, those achieved with the "Fifth Experiment" (Section 4.3.5). The positive dictionary contains 353, while the negative 2353. A large portion of both dictionaries has never been found in any article contained in the *Reuters news corpus*, precisely 1147. Among the ones found at least one time, we calculated the correlation as done in the *Fifth Experiment*. The complete list cannot be shown, but Table 4.8 summarises how many words belong to each correlation interval.
Little more than half of the analysed words do not show a significant correlation,

Table 4.8: Fifth experiment summary.

Correlation interval	Number of words
$0{,}3 < \rho \leq 1$	214
$-0{,}3 \leq \rho \leq 0{,}3$	1100
$-1 \leq \rho < -0{,}3$	245

but for the other half, we have a medium-strong correlation. Obviously, between

them, there are also some words, with very few occurrences, with a strong correlation. For this reason, these results must be aggregated and filtered. We defined two thresholds, one on the *word correlation* and the *total word occurrence*. The former is used right after, while the latter will be used in the final *news-based* features selection Section 4.5.
By setting a threshold on the correlation, we are automatically selecting best and worst words. Words with a very negative correlation must not be discarded because they bring useful information too. The word attribution to a given dictionary was left to the Loughran-McDonald's research [22] and cannot be considered an *error-free* procedure. Perhaps, a positive word was misclassified as a negative one or vice versa.
Therefore, words with a correlation higher than 0,5 were considered as *best words*, while the ones with a correlation lower than $-0{,}5$ were considered as *worst words*.
Word selection process it is not finished here because there is another very important issue to manage properly. As the reader will remember, the fifth experiment produced a file for each company and each year. The reason is that the framework to which the sentiment module was included performs both classification and trading phase one year at a time.
Let us assume we start with the year 2007, and we have a *word-by-word correlation* file as a result of an analysis performed between 2007 and 2017. Self-evidently, we would like to use the just discovered dictionary features in order to give the classifier more possible elements to make correct decisions. However, here is the trouble: from the conceptual point of view, can we use information related to the years after 2007?
The answer is clearly no because we are using information derived from years that are in the future concerning the year considered in the example (2007). For this reason, we split the *word-by-word correlation* process per year. In Section 4.5 we will explain how to transform those promising results into *features* for the classifiers.

4.4.1 Algorithm implementation

In this subsection we will describe the *best* word selection algorithm procedure. The method for selecting *worst* words is dual to one described below and will be omitted. For each year to be examined (between 2007-2017), given a correlation threshold, each *word-by-word correlation* file is analysed. For each of them and for each word contained in those files, if the word correlation is higher than the threshold we store both its correlation and how many times has occurred, otherwise the analysis continue. The same procedure is repeated for each *word-by-word correlation* file. When every file is examined, for each word found it is computed its

medium correlation score, its percentage as best and how many times was found. Before continuing with the successive year, those data are stored in a file representing *best* words found in a year.

An extract of the best words selection algorithm for the year 2007 can be consulted in Table 4.9. This file, in particular, counts 513 unique words, with an average correlation that oscillates between [0,5, 1]. Words are ordered according to the "%_As_Best" column.

Table 4.9: Best words for the year 2007.

Word	%_As_Best	AVG_Correlation	TOT_Occurrences
Benefit	4,7904	0,8438	88
Gains	4,7904	0,7758	102
Boost	4,5908	0,7428	107
Gained	4,1916	0,7544	81
Best	4,1916	0,7406	80
...	...	...	...

An extract of the worst selection algorithm results, always for the year 2007, can be consulted in Table 4.10. This file, in particular, counts 509 unique words, with an average correlation that oscillates between $[-1, -0{,}5]$. Even in this case, words are ordered according to the "%_As_Worst" column.

Table 4.10: Worst words for the year 2007.

Word	%_As_Worst	AVG_Correlation	TOT_Occurrences
Positive	5,7884	-0,748	137
Better	5,7884	-0,7546	150
Strength	5,5888	-0,7628	127
Strong	5,5888	-0,7694	129
Boosted	4,99	-0,7415	95
...	...	...	...

To give a general idea of which are the first five best words between the years 2007 and 2017, refer to Table 4.11. Table 4.11 is reported only for a matter of

curiosity because it cannot be used as it is, for the problem explained at the beginning of this section.

Table 4.11: Best words for years between 2007 and 2017.

Word	%_As_Best	AVG_Correlation	TOT_Occurrences
Improvements	8,3832	0,7012	271
Favorable	7,984	0,7133	196
Underperform	7,3852	0,788	128
Enhance	7,3852	0,7096	152
Strongest	7,1856	0,71	162
...	...	...	...

4.5 Final news-based features selection

At this stage, the correlation analysis can be considered concluded. It took us a lot of time and energies but brought remarkable results. In this section, we will show how to transform those results into *features* ready for the classification phase. Among the features defined in Section 3.3.3, we decided to remove *difference* and *phrase-level* features. In particular, features removed are: *DBTPNW, DBTSPNN, TSPP, TSNP, DBTSPNP*. The reasons are two:

- difference features showed a medium correlation with *Next_Day_price_variation* variable, but they can assume negative values that, depending on the classification algorithm used, could lead to problems. Nevertheless, since *DBTPNW, DBTSPNN* and *DBTSPNP* are calculated as the difference between two other features, their information is still present in a different form.
- phrase-level features showed a poor correlation in other experiments performed and not mentioned in this work.

In addition to the features coming from the correlation analysis, we introduced three new features that take into account general considerations about the number of articles/words found on that day. The complete list of selected features is:

- *TPW*: total number of positive words found in that article.
- *TNW*: total number of negative words found in that article.

- *TSPN*: total number of *single positive* news found on that day. An article is *single positive* if it contains only positive words. This feature has a daily scope since many articles could be found in a day.
- *TSNN*: total number of *single negative* news found on that day. This feature is exactly the opposite of *TSPN*.
- *TNN*: total number of news found on that day.
- *TWN*: total number of words found on that day.
- *TNWS*: total number of news expressing a sentiment. If either a positive or negative word is found in an article, this last has a sentiment and is counted by this feature.
- *TNPWWHC*: total number of positive words with the higher correlation found on that day.
- *TNNWWHC*: total number of negative words with the higher correlation found on that day.
- *TNPWWLC*: total number of positive words with the lowest correlation found on that day.
- *TNNWWLC*: total number of negative words with the lowest correlation found on that day.

Each company will have a file for each year. Each file has as first column the *market date*, and the following columns are the *final features* defined earlier. Table 4.12 shows first five rows of Apple company in the year 2009.
Unlike what one may think, the first row of the year 2009 it is not the first market date available in 2009, but it is a date belonging to 2008. In general, for each year considered (excluding 2007), there are exactly 50 market dates of the previous year that precede the first market date of the year in question. This design choice was made to maximise results coming from other framework indicators. In Section 5.1.1, this choice is adequately explained. For a matter of available space, we used the same naming convention already seen in Section 3.3.3.

Table 4.12: *Final news-based features* for the Apple company file.

Date	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_10	I_11
2008-10-21	35	50	0	0	15	4264	6	19	12	5	3
2008-10-22	72	287	1	1	29	11624	22	46	76	11	36
2008-10-23	78	257	3	0	24	10619	20	42	63	20	38
2008-10-24	2	23	0	0	1	637	1	1	4	0	3
2008-10-27	20	108	0	2	14	4222	9	13	37	4	23
...	...	...	...	...	...	...	...	...	...	...	...

Chapter 5

Quantitative trading system design and experiments

5.1 System design

The previous chapter ended with the selection of most representative *news-based features*, the ones that better describe textual data used as input. The *news sentiment analysis module* can be seen as a standalone module, which can be enabled and disabled at will. This module was integrated into a quantitative stock trading system, which is an automated trading system that exploits statistical indicators to identify patterns in historical stock prices. The core of the framework is forecasting stock price change using machine learning algorithm. The system can choose from several algorithms: *Support Vector Machine classifier, Random Forest classifier, Multinomial Naive Bayes classifier and Multilayer Perceptron*[29]. A general overview of the logical pipeline composing the quantitative trading system is described in Figure 5.1. As shown in Figure 5.1, the framework is composed of four distinct parts:

- *Sentiment analysis module*: the module described so far;
- *Stock pre-processing*: used to generate an input dataset ready for the *classification stage*;
- *Classification model training*: depending on the algorithm chosen, it creates a model used to predict stock price variation;
- *Trading signal generation*: it applies a trading strategy relying on the output of the *classification stage*.

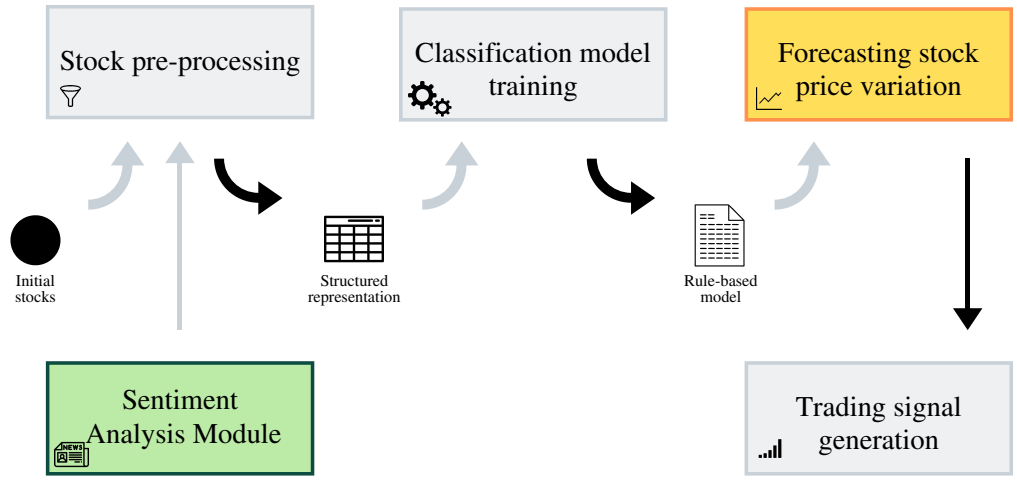

Figure 5.1: Logical pipeline describing the quantitative trading system.

The framework takes as input the following parameters:

- *candlestick_ YYYY.csv*: stocks file containing *Open, Low, High* and *Close* prices for each company belonging to the *S&P500* index. "YYYY" is the year to which the prices refer;
- *volume_ YYYY.csv*: stocks file containing exchanges *Volumes* for each company belonging to the *S&P500* index. This file must contain volumes of the same year "YYYY" of *Candlestick file*;
- *Classification algorithm*: acronym of the classification algorithm to use. Possibilities are: *SVM, MNB, MLP* or *RFC*;
- *News-based features directory*: location containing already computed *news-based* features (output of the *Sentiment Analysis module*). Sentiment Analysis must be done before the pre-processing phase because it comprises several stages and computations to do. This last parameter is optional, giving the possibility to disable sentiment analysis enhancement at will.

5.1.1 Stock pre-processing

This stage is critical because it provides to the classification phase the *input dataset* on which to perform calculations and predictions. This *input dataset* is constituted by at most four different classes of features, and each of these needs the pre-processing phase. Their combinations are described in Section 5.3.
Following operations are necessary to create the *input dataset*, independently from the feature's combination:

1. *Company filtration*
2. *Indicator evaluation*
3. *Class label evaluation*
4. *Normalisation*
5. *Indicator filtration*

Company filtration

During the years 2007 - 2017, several companies either left or entered the *S&P500* index. For this reason, it is possible that a company could have *null* price values during the year considered. To avoid this kind of problems, each company that has at least one *null* price value, in the year specified in the framework input parameters, is discarded from the analysis. Let us consider the case of the *Abbvie* company, founded in 2013. *Abbvie* historical stock prices before the year 2013 do not exist and, consequently, the *Yahoo finance* API (seen in Section 3.2.1) returned to us *null* when trying to fetch prices before 2013.
This means that the company *Abbvie* was discarded between years the 2008 and 2012, while it was considered for the remaining years.

Indicator evaluation

As already mentioned, the quantitative trading system makes significant use of statistical and mathematical indicators. Those indicators are evaluated in this stage. Their definitions are briefly described in Section 5.2. An important consideration has to be done on indicators belonging to the *technical* category. Most of the *technical indicators* are lagged, meaning that require several past prices to be evaluated. This behaviour could significantly reduce the number of market dates in which it is possible to make predictions. In order to clarify the problem and to better understand our solution, let us consider the following example.

The year 2008 contains 253 market date available for forecasting. Since some technical indicators require up to 50 past days to produce an "output value", it follows that the first 50 market dates should be discarded in order to be sure that all *technical indicators* have completed their computations. For this reason, both the *candlestick* and *volume* files of years between 2008 and 2017, have exactly 50 market date before the first available date of the year considered. In this way, the number of available market dates for the *candlestick* and *volume* 2008 files grows to 303, giving the possibility to the classifiers to make predictions starting from the first date of the year, without discarding any suitable date. Table 5.1 shows a possible representation of the dataset constituted of technical indicators.

Table 5.1: Input dataset extract at *indicator evaluation* process.

Day	A014	ADX14	WD14	...
0	null	null	null	...
1	-35,71	31,25	-4,20	...
2	-35,71	36,68	-5,91	...
3	-50,00	38,48	-5,28	...
...	...	...	...	...

Class label evaluation

In order to correctly train our classification algorithms, a *class label* for each entry of the *input dataset* is necessary. *Class labels* are evaluated according to *next-day* price variation variable, defined in 3.1 and used in all correlation experiments seen in Chapter 4. We used a discrete version of the *next-day* variable, calculated as:

$$L = \begin{cases} \text{BUY}, & \text{for } nd > 1 \\ \text{HOLD}, & \text{for } -1 \leq nd \leq 1 \\ \text{SELL}, & \text{for } nd < -1 \end{cases} \tag{5.1}$$

where nd is *next-day* variable and L is the class label. The input dataset (Table 5.1) now becomes the one shown in Table 5.2.

Normalisation

Due to the great difference between feature range values, a normalisation stage is necessary. We used two different normalisation techniques:

Table 5.2: Input dataset extract at *class label evaluation* process.

Day	A014	ADX14	WD14	...	Label
0	null	null	null	...	BUY
1	-35,71	31,25	-4,20	...	BUY
2	-35,71	36,68	-5,91	...	SELL
3	-50,00	38,48	-5,28	...	HOLD
...	...	...	...	...	...

- *Standardization*: this method, also known as (*Z-score Normalization*), is widely used in machine learning algorithms. For each feature, it is necessary to determine its *mean* ($\bar{x}$) and *standard deviation* (σ). Then, each value of the feature is normalised with the formula:

$$x' = \frac{x - \bar{x}}{\sigma} \tag{5.2}$$

  where x' is the feature's element normalised.

- *Rescaling*: with the previous normalisation strategy, feature elements can assume any value. This strategy, also known as (*min-max normalization*) scales feature's elements to a range of values, in our case is [0,1]. Assuming $\min(x)$ is the minimum value of the feature vector and $\max(x)$ is its maximum value, each feature's element is normalised with the formula:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{5.3}$$

  where x' is the feature's element normalised.

Standardization was applied when using *SVM, RFC* and *MLP* classification algorithms, while *Rescaling* was used for *MNB*, due to its need to have features with positive values. Our *input dataset* example, once *Standardization* was applied, becomes the one shown in Table 5.3.

Indicator filtration

As already seen in the *indicator evaluation* process, some indicators could have *null* values in specific market dates. Here, those values are identified and, whenever a

Table 5.3: Input dataset extract after *Standardization*.

Day	A014	ADX14	WD14	...	Label
0	null	null	null	...	BUY
1	-0,57	0,20	-0,59	...	BUY
2	-0,57	0,78	-0,95	...	SELL
3	-0,81	0,98	-0,82	...	HOLD
...	...	...	...	...	...

null indicator value is found, the corresponding market date is discarded. Table 5.4 shows this process on our *input dataset* example.

Table 5.4: Input dataset extract after the *indicator filtration* process.

Day	A014	ADX14	WD14	...	Label
0	-0,57	0,20	-0,59	...	BUY
1	-0,57	0,78	-0,95	...	SELL
2	-0,81	0,98	-0,82	...	HOLD
...	...	...	...	...	...

5.1.2 Classification model training

Since the classification model depends on the classification algorithm selected in input parameters, we will not go into too much detail of this phase. What is important to know are the following design choices: the *validation strategy* used and how to interpret the classification *output*.
We analysed two different *validation strategies*, both widely used on Machine learning algorithms:

- *Hold-out*: this strategy assigns to the training set a fixed portion of data. We set the training set to be the 66% of the *input dataset* while the remaining part is assigned to the test set. In this strategy, the model is built only once and, in the testing phase, class labels are assigned according to that model.

- *Expanding window*: this strategy, instead, is more dynamic since it uses an increasing expanding training set. The initial training set is set to 33% of the *input dataset*. The model is generated M times where M is evaluated with

the formula:

$$M = \mid D \mid -\mathrm{i} \tag{5.4}$$

where i is the initial training set and $\mid D \mid$ is the *input dataset* size (number of days). For each model generated in this way, the prediction is done only on the next day, the one that immediately follows the training set.

With regards to the *classification output*, it can be seen as an array of labels. To each date assessed in the *testing* set, it is assigned a label corresponding to the classifier's prediction. Figure 5.2 depicts an example of the *classification output* array.

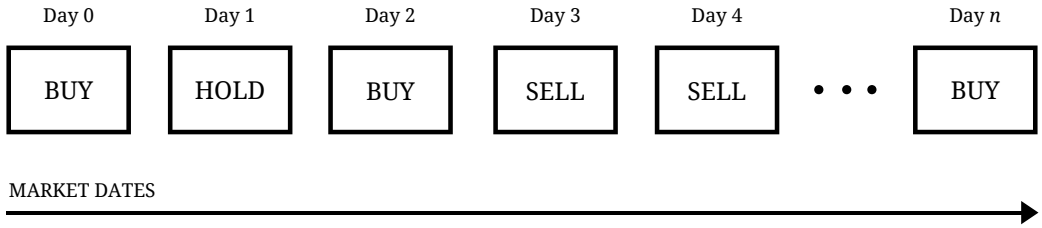

Figure 5.2: Classification output array example.

5.1.3 Trading signal generation

The output array, provided by the classification stage, is used to generate trading signals by applying a suitable trading strategy. The trading process takes decisions day after day, starting from the first element of the *classification output array* (corresponds to the first day of the test set). Three possible decisions are accepted:

- *BUY*: buy a given stock;
- *SELL*: sell a given stock;
- *HOLD*: no relevant change detected, do nothing.

Correct predictions could result in a *profit*. The framework does not take into account quantities, but only the percentage profit.
However, given an *initial investment* quantity, the absolute gross profit is easy to calculate: *profit* * *initial_ investment*.
The trading strategy relies on the same assumptions defined and listed in [28], and reported here:

- two possible trading operations exist: *SHORT* and *LONG*. The former is used when stock prices fall, the latter when stock prices grow. A trading operation cannot be activated if another is already active.
- the overall income generated by a SHORT operation is positive if the stock closing price is lower than the opening one, while it is negative if higher. For LONG operations, it applies the opposite rule.
- BUY label suggests that the stock price will grow. A LONG position is opened; if a SHORT position is already opened, at the end of the day is closed.
- SELL label suggests that the stock price will decrease. A SHORT position is opened; if a LONG position is already opened, at the end of the day is closed.

The trading strategy used is called: *multiple days* strategy. Thanks to this strategy, active operations can remain active for multiple days. In particular, a LONG active operation remains so if successive labels are either BUY or HOLD; a SHORT active operation remains so if successive labels are either SELL or HOLD. Figure 5.3 shows two operations, one of which lasts for three days.

This stage provides as output a summary file. For each distinct classification algorithm, configuration, year and validation strategy, the framework computes the *total profit* as the arithmetic sum of each single company *profit*. Table 5.5 shows an example of the trading summary file.

5.2 Quantitative analysis indicators

The existing framework uses three indicators' classes: *technical*, *temporal* and *price indicators*. Since those indicators were already developed in the quantitative trading system, we will limit to a simple description, without deepening those topics. Those indicators, together with the *final news-based features*, will be combined in several experiments. Next section shows all experiments carried out in order to asses if the *news sentiment module* leads to an improvement or not.

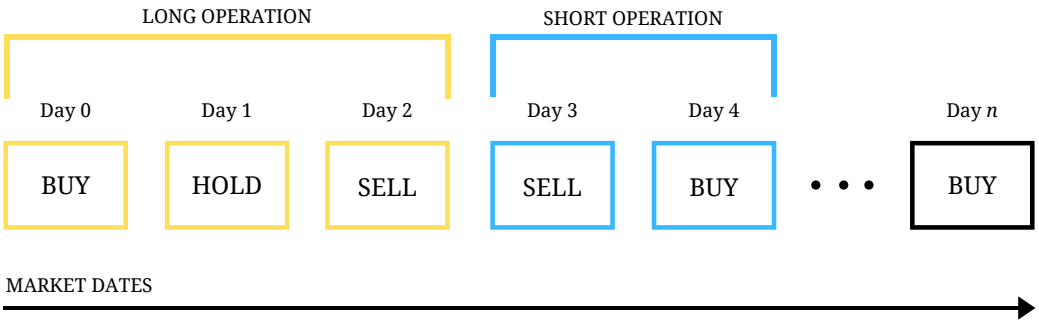

Figure 5.3: Multiple days strategy applied to classification array example.

Table 5.5: Trading summary file example.

Classifier	Year	Config	Validation	Total profit
RFC	8	(10, entropy)	exp	49,09%
RFC	8	(10, entropy)	hold	41,65%
RFC	8	(10, gini)	exp	42,36%
...	...	...	...	...
SVC	8	rbf	exp	36,26%
SVC	8	rbf	hold	43,34%
SVC	8	linear	exp	34,58%
...	...	...	...	...

Technical indicators

Technical features are either heuristic or mathematical calculations performed both on stock prices and exchanged volumes. The framework exploits 22 different technical features, belonging to different categories. Table 5.6 shows which technical features are embedded in the framework. For additional information on their definitions, meanings and usages refer to [28].

Temporal indicators

These features refer to stock close prices. A temporal window W is defined. W represents not only the temporal window, but also the number of temporal indicators that will be calculated. According to the framework design choices, W was set to 10, meaning that 10 temporal features are created. Their evaluation follows

Table 5.6: List of technical indicators used in the framework.

Technical indicators		
SMA5-20	EMA20-50	PPO12_26
SMA8-15	MACD12-26	RSI14
SMA20-50	AO14	MFI14
EMA5-20	ADX14	TSI
EMA8-15	WD14	SO14
CM014	ATRP14	PVO14
FI13	FI50	ADL
OBV		

those steps:

1. discard first W market date.
2. for each market date i, collect a *window* of $W+1$ close prices, called wcp_array.
3. for each *window* collected, evaluate W temporal indicator using the following formula:

$$\sum_{i=W}^{D}\sum_{j=1}^{W+1} t_{i,j} = \frac{(\text{wcp\_array}[j+1] - \text{wcp\_array}[j]) * 100}{\text{wcp\_array}[j]} \tag{5.5}$$

where i the *i-th* dataset row, j if the *j-th* temporal indicator and wcp_array[j] is the close price at position j of the window wcp_array.

Price indicators

These indicators reflect stock price components and are the simplest to understand. The framework defines five *price indicators* that exactly corresponds to *Open, High, Low, Close* and *Volume* stock price series. Using as example Table 3.2, they correspond to the second, third, fourth, fifth and seventh column, respectively.

5.3 Experiments

The experiments' objective is twofold:

- analyse news impact on Machine Learning models trained with different features
- analyse effectiveness of "combined models" which exploit both news sentiment features and quantitative analysis indicators, those described in Section 5.2.

Experiments were carried out one year at a time, between 2008 and 2017. Four sets of experiments were conducted; each set includes experiments with and without the addition of the *news sentiment analysis module*. Set of experiments are:

- *Technical* **vs** *Technical and News*
- *Temporal* **vs** *Temporal and News*
- *Price* **vs** *Price and News*
- *Price, Temporal and Technical* **vs** *Price, Temporal, Technical and News*

For each experiment set, four machine learning algorithms were used:

- *Support Vector Machine*
- *Random Forest Classifier*
- *Naive Bayes*
- *Multilayer Perceptron*

For each machine learning algorithm, different algorithm configurations were explored:

- *SVM*:
  - *Kernel*: rbf, linear and polynomial.
- *RFC*:
  - *Number of trees*: 10, 50, 100.
  - *Criterion*: *gini* and *entropy*.
- *MLP*:
  - *Hidden layers*: 20.
  - *Solver*: lbfgs.

- *MNB*:
  - *Alpha*: 0.2, 0.4, 0.6, 0.8, 1.0.

For each algorithm configuration, both *hold-out* and *expanding window* validation strategies are used.
The operation type is set to *LONG-SHORT* and *multiple days* trading strategy is used.

Chapter 6

Results

In this chapter, we will introduce the experimental results. The most precise and understandable way to show results is using structured tables. However, due to the massive amount of collected experiments, we will show result aggregating algorithm configurations and validation strategies. In this way, we can focus better on the purpose of this work: assess *sentiment analysis module* effectiveness in the quantitative trading system.
Results are organised as follows: each section shows the experiment set made. Each section depicts four different tables corresponding to the classification algorithm used to build the model. Each structured table has the following columns:

- *Year*: the year analysed;
- *Average total profit* (ATP_nosent): average total profit without the sentiment module;
- *Average total profit* (ATP_sent): average total profit with the sentiment module enabled;
- *Delta percentage*: percentage variation calculated with the formula:

$$\delta = \frac{ATP_{sent} - ATP_{nosent}}{ATP_{nosent}} * 100 \tag{6.1}$$

  where ATP_sent stands for "Average Total profit" with the sentiment module enabled, while ATP_nosent is the "Average Total profit" without sentiment module.

6.1 Technical indicators *vs* technical indicators and news-based features

Average Total Profit tech column represent the average total profit coming from experiments using only technical indicators. *Average Total Profit tech-sent*, instead, represents the average total profit coming from experiments using both technical and news-based features as input dataset for the classification phase. Table 6.1 shows results using *RFC* model, Table 6.2 shows results using *SVC* model, Table 6.3 shows results using *MLP* model and Table 6.4 shows results using *MNB* model. We reach a peak of increase of 530,99% with *MNB* model and the sentiment enabled.

Table 6.1: Average total profits per year with RFC model, "tech *vs* tech-news" experiment.

Year	Average Total Profit tech	Average Total Profit tech-sent	δ
8	45,27%	46,75%	+4,13%
9	22,02%	23,30%	+5,86%
10	2,18%	2,30%	+1,08%
11	16,09%	13,56%	-14,94%
12	5,94%	7,27%	+22,39%
13	14,69%	15,12%	+4,02%
14	13,75%	14,20%	+4,54%
15	10,57%	10,27%	-6,27%
16	10,01%	10,68%	+5,89%
17	11,36%	10,44%	-8,09%

6.2 Price indicators *vs* price indicators and news-based features

Average Total Profit price column represent the average total profit coming from experiments using only price indicators. *Average Total Profit price-sent*, instead, represents the average total profit coming from experiments using both price and news-based features as input dataset for the classification phase. Table 6.5 shows results using *RFC* model, Table 6.6 shows results using *SVC* model, Table 6.7 shows results using *MLP* model and Table 6.8 shows results using *MNB* model. In Table 6.7 there is a configuration (year 2012) that performs meaningfully worse

Table 6.2: Average total profits per year with SVC model, "tech *vs* tech-news" experiment.

Year	Average Total Profit tech	Average Total Profit tech-sent	δ
8	41,25%	46,14%	+10,83%
9	25,67%	25,72%	+2,47%
10	5,72%	5,41%	-5,41%
11	2,36%	3,32%	+40,67%
12	5,92%	5,76%	+12,41%
13	14,79%	13,03%	-10,25%
14	9,24%	8,32%	-4,45%
15	4,34%	3,26%	-5,16%
16	11,83%	10,35%	-3,70%
17	6,74%	5,89%	-12,61%

Table 6.3: Average total profits per year with MLP model, "tech *vs* tech-news" experiment.

Year	Average Total Profit tech	Average Total Profit tech-sent	δ
8	56,61%	52,49%	-7.28%
9	20,89%	19,56%	-3,45%
10	-3,67%	0,16%	+32,85%
11	26,73%	28,27%	+6,80%
12	8,84%	9,93%	+15,23%
13	11,84%	14,37%	+39,99%
14	14,66%	16,83%	+15,40%
15	17,44%	13,44%	-12,33%
16	15,30%	16,54%	+6,34%
17	12,16%	12,72%	+1,40%

(-199,30%), while the *MNB* model in 2011 (Table 6.8) registers the higher *delta percentage* (+176,25%), even if the is slightly negative.

6.3 Temporal indicators *vs* temporal indicators and news-based features

Average Total Profit temp column represent the average total profit coming from experiments using only temporal indicators. *Average Total Profit temp-sent*, instead, represents the average total profit coming from experiments using both temporal

Table 6.4: Average total profits per year with MNB model, "tech *vs* tech-news" experiment.

Year	Average Total Profit tech	Average Total Profit tech-sent	δ
8	45,61%	57,42%	+29,10%
9	25,32%	21,90%	-13,40%
10	0,61%	3,03%	+324,77%
11	-0,92%	3,96%	+530,99%
12	1,22%	3,85%	+258,81%
13	6,45%	11,52%	+76,02%
14	1,21%	6,05%	+393,27%
15	2,34%	2,29%	-10,12%
16	6,74%	8,07%	+27,18%
17	1,67%	6,70%	+358,60%

Table 6.5: Average total profits per year with RFC model, "price *vs* price-news" experiment.

Year	Average Total Profit price	Average Total Profit price-sent	δ
8	54,64%	53,67%	-5,45%
9	35,61%	40,94%	+14,96%
10	15,21%	14,58%	-4,07%
11	31,99%	32,58%	+2,00%
12	11,69%	10,00%	-14,45%
13	11,32%	11,68%	+3,18%
14	9,32%	9,27%	-2,17%
15	21,25%	21,20%	-1,03%
16	12,85%	14,04%	+20,11%
17	12,79%	11,35%	-8,16%

and news-based features as input dataset for the classification phase. Table 6.9 shows results using *RFC* model, Table 6.10 shows results using *SVC* model, Table 6.11 shows results using *MLP* model and Table 6.12 shows results using *MNB* model. *RFC* model in 2008 produced the higher average total profit (69,96%, in Table 6.9) among all experiments set done. Higher and lower *delta percentage* obtained in this experiment belong both to MNB model in year 2012 and 2016, respectively.

Table 6.6: Average total profits per year with SVC model, "price *vs* price-news" experiment.

Year	Average Total Profit price	Average Total Profit price-sent	δ
8	36,73%	41,38%	+19,39%
9	7,57%	13,47%	+87,85%
10	10,63%	11,09%	+5,36%
11	7,86%	8,71%	+16,46%
12	6,68%	5,60%	-4,43%
13	5,79%	9,24%	+62,57%
14	2,67%	5,35%	+76,89%
15	4,83%	3,66%	-12,24%
16	9,83%	10,07%	+10,05%
17	3,36%	5,90%	+75,89%

Table 6.7: Average total profits per year with MLP model, "price *vs* price-news" experiment.

Year	Average Total Profit price	Average Total Profit price-sent	δ
8	52,21%	56,81%	+3,72%
9	26,71%	24,22%	-16,84%
10	13,43%	9,92%	-35,12%
11	33,27%	36,82%	+16,74%
12	16,72%	8,83%	-199,30%
13	11,92%	13,40%	+7,64%
14	13,02%	10,96%	-8,37%
15	20,03%	19,55%	-2,64%
16	12,20%	13,58%	+2,78%
17	11,52%	9,91%	-5,79%

6.4 Temporal, technical and price indicators *vs* temporal, technical, prices and news-based features

Average Total Profit ptmth column represent the average total profit coming from experiments using prices, temporal and technical indicators. *Average Total Profit ptmth-sent*, instead, represents the average total profit coming from experiments using all framework features as input dataset for the classification phase. "ptmth" abbreviation stands for "price-temporal-technical" Table 6.13 shows results using

Table 6.8: Average total profits per year with MNB model, "price *vs* price-news" experiment.

Year	Average Total Profit price	Average Total Profit price-sent	δ
8	28,16%	47,58%	+77,10%
9	28,01%	15,00%	-56,21%
10	7,45%	11,56%	+61,16%
11	-2,03%	-0,87%	+176,25%
12	2,38%	2,85%	+13,04%
13	4,57%	8,68%	+140,44%
14	1,25%	3,32%	+165,73%
15	1,94%	0,26%	-69,91%
16	3,26%	8,79%	+169,64%
17	2,95%	5,99%	+159,90%

Table 6.9: Average total profits per year with RFC model, "temp *vs* temp-news" experiment.

Year	Average Total Profit temp	Average Total Profit temp-sent	δ
8	69,95%	69,96%	+0,98%
9	42,13%	42,14%	+1,00%
10	24,55%	24,16%	-1,89%
11	23,09%	21,24%	-6,62%
12	6,80%	6,71%	-1,97%
13	26,05%	25,62%	-2,46%
14	10,91%	11,13%	+3,17%
15	12,92%	12,09%	-1,03%
16	12,32%	11,71%	-2,98%
17	17,16%	16,27%	-2,60%

RFC model, Table 6.14 shows results using *SVC* model, Table 6.15 shows results using *MLP* model and Table 6.16 shows results using *MNB* model. Even in this set of experiments, *MNB* model reaches peak of increase of 197,71%.

6.5 Results summary

Due to the high number of experiments done, a structured table that summarises results aggregated per year can be useful to draw general conclusions. Table 6.17

Table 6.10: Average total profits per year with SVC model, "temp *vs* temp-news" experiment.

Year	Average Total Profit temp	Average Total Profit temp-sent	δ
8	60,58%	65,85%	+12,49%
9	36,52%	35,17%	-5,47%
10	19,95%	17,89%	-9,62%
11	11,97%	10,34%	-1,04%
12	6,43%	3,43%	-41,53%
13	14,45%	13,36%	-3,00%
14	4,17%	4,36%	+39,00%
15	7,11%	5,02%	+20,54%
16	7,79%	7,74%	-0,46%
17	7,05%	5,08%	-8,46%

Table 6.11: Average total profits per year with MLP model, "temp *vs* temp-news" experiment.

Year	Average Total Profit temp	Average Total Profit temp-sent	δ
8	69,53%	67,31%	-3,66%
9	36,12%	31,86%	-15,91%
10	17,46%	13,45%	-38,07%
11	29,39%	29,84%	+5,91%
12	5,50%	7,70%	+54,80%
13	16,96%	17,60%	+8,14%
14	9,47%	13,30%	+42,74%
15	18,21%	20,30%	+15,08%
16	15,92%	15,94%	+1,11%
17	16,04%	18,20%	+5,36%

shows those aggregated results, where "Average Total Profit" words is shortened in "ATP" due to lack of space. Concerning the news sentiment impact into the trading system, results are particularly evident when using the *Multinomial Nayve Bayes* classifier, resulting in an average total profit 30,35% higher. *SVC, RFC* and *MLP*, always with the *news sentiment module* enabled, performs on average 4,4%, -0,51% and 0,71%, respectively.
Best result in terms of *Average Total Profit* (69,96% in Table 6.9) was achieved with *Random Forest Classifier* in the year 2008 when using as input dataset for the

Table 6.12: Average total profits per year with MNB model, "temp *vs* temp-news" experiment.

Year	Average Total Profit temp	Average Total Profit temp-sent	δ
8	33,03%	45,28%	+41,97%
9	32,76%	25,68%	-23,57%
10	8,20%	10,47%	+31,72%
11	-1,34%	-1,56%	-2,11%
12	1,67%	-2,05%	-222,79%
13	4,56%	10,26%	+131,10%
14	0,49%	3,30%	+496,97%
15	0,52%	1,86%	+81,85%
16	1,70%	7,71%	+659,22%
17	2,63%	7,94%	+209,43%

Table 6.13: Average total profits per year with RFC model, "pricetemptech *vs* pricetemptech-news" experiment.

Year	Average Total Profit ptmth	Average Total Profit ptmth-sent	δ
8	48,96%	46,03%	-5,57%
9	22,59%	22,73%	+3,72%
10	4,39%	2,67%	-73,30%
11	26,78%	26,18%	-1,87%
12	7,75%	7,91%	+1,08%
13	11,92%	13,33%	+8,76%
14	11,12%	11,02%	+1,04%
15	12,28%	12,31%	+0,09%
16	11,30%	11,49%	+0,79%
17	11,36%	10,37%	-2,16%

classification phase both *temporal indicators* and *news-based* features.
If we aggregate results at experiment set level there are two important conclusions that we can draw. First of all, Table 6.18 *delta* column underlines the positive impact of the sentiment module, in all the experiments done, symptom of the importance of news in the classification process. Contrary to what might be thought, there is not a direct correspondence between the number of features exploited in the model training phase and the quantitative trading performance. Indeed, the lowest result in terms of Average Total Profit was registered by the experiment set "tech" (without the sentiment module integration). This configuration counts 22 technical

Table 6.14: Average total profits per year with SVC model, "pricetemptech *vs* pricetemptech-news" experiment.

Year	Average Total Profit ptmth	Average Total Profit ptmth-sent	δ
8	46,40%	50,14%	+6,79%
9	23,41%	23,70%	+,1.5%
10	7,44%	8,68%	+13,13%
11	14,30%	15,07%	+11,03%
12	8,18%	8,66%	+0,46%
13	13,95%	12,45%	-1,50%
14	8,75%	9,29%	+3,50%
15	7,83%	7,50%	-0,74%
16	10,74%	11,40%	+0,76%
17	8,50%	9,46%	+11,04%

Table 6.15: Average total profits per year with MLP model, "pricetemptech *vs* pricetemptech-news" experiment.

Year	Average Total Profit ptmth	Average Total Profit ptmth-sent	δ
8	55,58%	61,74%	+11,26%
9	19,55%	22,75%	+19,46%
10	2,45%	3,75%	-3,99%
11	38,72%	35,86%	-3,76%
12	10,94%	10,87%	-0,42%
13	9,26%	9,23%	-0,13%
14	15,69%	16,42%	+1,49%
15	13,53%	17,11%	+5,68%
16	15,75%	16,09%	+6,67%
17	11,48%	11,45%	-0,17%

indicators, and it performs worse with respect to the "temp" experiment set that exploits only 10 temporal features. The latter configuration obtained an average total profit score even higher that the configuration "price-temp-tech", which uses 37 features to build the model. This problem, widely known as *curse of dimensionality* [29], must be taken into account in order to maximise performances and classification quality. In Chapter 7 we propose a solution to this issue that will be integrated in future works.

Table 6.16: Average total profits per year with MNB model, "pricetemptech *vs* pricetemptech-news" experiment.

Year	Average Total Profit ptmth	Average Total Profit ptmth-sent	δ
8	55,50%	59,30%	+7,56%
9	11,10%	16,44%	-16,35%
10	9,18%	6,64%	-14,36%
11	15,85%	18,24%	-8,64%
12	8,38%	4,69%	-167,67%
13	13,50%	11,57%	-13,91%
14	4,18%	8,29%	+179,61%
15	3,34%	3,55%	+1,75%
16	11,68%	9,98%	-6,89%
17	4,34%	8,24%	+197,71%

Table 6.17: Experiments summary where Average Total Profits are aggregated per years.

Experiment set	Classifier	ATP_nosent	ATP_sent	δ
tech *vs* tech-news	RFC	15,19%	15,39%	+1,32%
tech *vs* tech-news	SVC	12,79%	12,72%	-0,49%
tech *vs* tech-news	MLP	18,08%	18,13%	+0,27%
tech *vs* tech-news	MNB	9,02%	12,48%	+38,28%
price *vs* price-news	RFC	21,67%	21,96%	+1,35%
price *vs* price-news	SVC	9,60%	11,45%	+19,28%
price *vs* price-news	MLP	21,10%	20,40%	-3,33%
price *vs* price-news	MNB	7,79%	11,43%	+46,66%
temp *vs* temp-news	RFC	24,59%	24,10%	-1,97%
temp *vs* temp-news	SVC	17,61%	16,83%	-4,43%
temp *vs* temp-news	MLP	23,46%	23,35%	-0,46%
temp *vs* temp-news	MNB	8,42%	10,89%	+29,29%
ptemtech *vs* ptemtech-news	RFC	16,84%	16,38%	-2,75%
ptemtech *vs* ptemtech-news	SVC	14,95%	15,59%	+4,28%
ptemtech *vs* ptemtech-news	MLP	19,30%	20,53%	+6,36%
ptemtech *vs* ptemtech-news	MNB	13,71%	14,70%	+7,20%

Table 6.18: Results aggregated at experiment set level.

Experiment set	ATP_nosent	ATP_sent	δ
tech *vs* tech-news	13,70%	14,68%	+7,15%
price *vs* price-news	15,04%	16,31%	+8,44%
temp *vs* temp-news	18,52%	18,79%	+1,47%
ptemtech *vs* ptemtech-news	16,02%	16,80%	+4,86%

Chapter 7

Conclusions and future works

The objective of this work was to determine whether, and to what extent, financial news could influence stock price change.
Results highlighted that the quantitative trading system performed on average better in any simulation with the *news sentiment module* enabled. Summarised results, depicted in Table 6.17, showed that in most cases, experiments with the sentiment module enabled brought to a higher profit. In particular, when using the *Multinomial Nayve Bayes* classifier, average total profit, among all experiment sets done, with the *news sentiment module* enabled, is 30,35% higher than the one without the sentiment module. From Table 6.18, it is clearer the sentiment analysis contribution, which is on average 5,48% higher with respect to the experiments without the sentiment module.
Those findings can be considered incredibly robust as they are the result of several aggregations done on four different classification algorithms, with several configurations tried, analysing nearly 500 stock companies between years 2008 and 2017. If we had presented results considering only a few stocks on a limited period, *Average Total Profit* would have even assumed values up to 1000%. Extending the analysis hundreds of companies in 10 different years, instead, drastically reduce the possibility to be dependent on the circumstances of each company.
Results not only confirmed sentiment analysis importance in stock forecasting but have still pointed out the possibility for investors of making a profit exploiting this kind of quantitative trading systems.

Many aspects could be explored in future implementations. First of all, in the news filtration phase, we made very restrictive choices in order to be sure that each article was strictly related to the company. As described in [7], different level of news relevance could enhance prediction's quality, since articles involving companies' sectors could influence their stock prices. However, the computational load

in the *filtration phase* increases a lot, and the abundance of articles might result in inaccurate incomes. With this in mind, also different dictionaries could be assessed, containing words with intermediate polarity scores. With regards to the classification phase, other algorithms (other than those already used) could lead to better/worst results as well as other trading strategies might lead to different findings. Finally, we plan to integrate feature selection steps into the trading system in order to select a subset of most relevant features for use in model training and to mitigate *curse of dimensionality* problem described in Section 6.5.

Bibliography

[1] Malkiel, Burton G.; Fama, Eugene F. (1970). *"Efficient capital markets: a review of theory and empirical work"*. The Journal of Finance.

[2] Ko Ichinose, Kazutaka Shimada. *Stock market prediction from news on the Web and a new evaluation approach in trading*, 2016

[3] R.P. Schumaker, H. Chen. *Textual analysis of stockmarket prediction using breaking financial news: the AZFin text system.* ACMTransactions on Information Systems (2) (2009).

[4] Michael Hagenau, Matthias Hauser, Michael Liebmann, Dirk Neumann. *Reading all the news at the same time: Predicting mid-term stock price developments based on news momentum.*

[5] Michael Hagenau, Michael Liebmann, Dirk Neumann. *Automated news reading: Stock price prediction based on financial news using context-capturing features*

[6] Xiao Ding, Yue Zhang, Ting Liu, Junwen Duan. *Deep Learning for Event-Driven Stock Prediction.*

[7] Yauheniya Shynkevich, T.M. McGinnity, Sonya Colemanl, Ammar Belatrechel. *Stock Price Prediction based on Stock-Specific and Sub-Industry-Specific News Articles.*

[8] Kalyani Joshi, Prof. Bharathi H. N., Prof. Jyothi Rao. *STOCK TREND PREDICTION USING NEWS SENTIMENT ANALYSIS.*

[9] Kimihisa Kawabata, Hitoshi Takata, Yoshihiro Fukunaga. *Forecasting of FX-Trading with Consideration for the Impact of News.*

[10] SHOU-HSIUNG CHENG. *FORECASTING THE CHANGE OF INTRADAY STOCK PRICE BY USING TEXT MINING NEWS OF STOCK.*

[11] Heeyoung Lee, Mihai Surdeanu, Bill MacCartney, Dan Jurafsky. *On the Importance of Text Analysis for Stock Price Prediction.*

[12] Tomer Geva, Jacob Zahavi. *Empirical evaluation of an automated intraday stock recommendation system incorporating both market data and textual news.*

[13] Thien Hai Nguyen, Kiyoaki Shirai. *Topic Modeling based Sentiment Analysis on Social Media for Stock Market Prediction*

[14] `https://www.reuters.com/`

[15] `http://en.wikipedia.org/wiki/List_of_S%26P_500_companies`

[16] https://finance.yahoo.com/
[17] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[18] https://www.investopedia.com/terms/a/adjusted_closing_price.asp
[19] https://www.tradingview.com/
[20] https://docs.python.org/3/library/re.html
[21] https://www.wikidata.org/wiki/Wikidata:Main_Page
[22] Tim Loughran and Bill McDonald, 2016, *Textual Analysis in Accounting and Finance: A Survey.* https://sraf.nd.edu/textual-analysis/
[23] https://pandas.pydata.org/pandas-docs/stable/
[24] https://www.nltk.org/
[25] https://whatis.techtarget.com/definition/correlation
[26] https://en.wikipedia.org/wiki/Correlation_coefficient
[27] https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php
[28] Giuseppe Attanasio, *Comparing time series and associative classification approaches to quantitative stock trading.* https://webthesis.biblio.polito.it/9061/
[29] Bertrand Clarke, Ernest Fokouè, Hao Helen Zhang. *Principles and Theory for Data Mining and Machine learning.*