CHALMERS

Price distribution over a set of events
μ(t)+σ(t)
μ(t)
μ(t)-σ(t)
Price difference from event day [%]
t [days] (Day of the event: t=0)
25
20
15
10
5
0
-5
-20
-10
0
10
20
30

Developing components for a data-driven trading system

Bachelor's thesis in Computer Science and Engineering

FRED HEDENBERG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

DEGREE PROJECT REPORT

Developing components for a data-driven trading system

FRED HEDENBERG

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Developing components for a data-driven trading system
FRED HEDENBERG

Examiner: Peter Lundin

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover: This visualization is meant to communicate a summarized view of how the price develops over a set of events. It shows both mean price and standard deviation, at every discrete timestep t.

Department of Computer Science and Engineering
Gothenburg 2018

ABSTRACT

Data is an important resource, which through data science can yield many interesting insights and predictions. To utilize data, it needs to be collected, stored, preprocessed and modelled. The purpose of this project was to develop components that together form a simple data-driven stock trading system. Firstly, software to web scrape publicly available news and pricing data from the Swedish stock exchange AktieTorget, is developed. Secondly, this data is then used to test whether it is possible to create models and tools able of aiding/performing trading decisions. To support the development of the above-mentioned software and models, some theory is provided about stock markets, together with a walkthrough of the workflow and components of data science. The result is a system that utilizes a Python module, Scrapy, to automatically collect news and pricing data and then pass it on to a MongoDB database. The characteristics of the collected pricing data made it hard to work with, which was partly solved by manually collecting data from Nasdaq OMX. In addition to the system to collect and store data, three experiments were conducted to test the model and tool developed. All three experiments gave interesting insights, even though the results weren't assertive. The single most interesting result was the model's predicting performance for clustered signals. As a notice of potential future work, an API connection to a broker (e.g. Nordnet) could be developed. This would enable models to be used for real time trading. Moreover, the news and pricing data can be used together with natural language processing to create more sophisticated models.

Keywords: web scraping, data science, machine learning, stock market

SAMMANFATTNING

Data har blivit en viktig resurs och med hjälp av data science kan den ge intressanta insikter. För att kunna utnyttja data bör den inhämtas, lagras, processeras och modelleras. Syftet med detta projekt var att utveckla delkomponenter till ett aktiehandelssystem, där data står i fokus. Första delmålet var att utveckla en komponent som ska utföra automatisk inhämtning av nyhets- och prisdata, genom s.k. “web scraping”. Det andra delmålet var att utveckla en komponent för att kunna analysera den inhämtade datan. Detta i form av en modell och ett visualiseringsverktyg som ska kunna användas som beslutsunderlag för aktiehandel. För att underlätta utvecklandet av ovannämnt system gås teori om aktiemarknaden samt data science igenom. Det resulterande systemet använder en Pythonmodul, Scrapy, för att utföra den automatiska inhämtningen av nyhets- och prisdata. Datan skickas sedan vidare från Scrapy till en MongoDB-databas. Prisdatan som inhämtades visade sig vara svåranalyserad, vilket löstes genom att istället analysera manuellt inhämtad prisdata från Nasdaq OMX. När första komponenten var utvecklad utfördes tre experiment, alla tänkta att testa den modell och det verktyg som utvecklats. Alla tre experiment gav intressanta insikter, även om de inte gav definitiva resultat. Det mest intressanta resultatet var utan tvekan hur bra prisutvecklingen blev då den utvecklade modellen gav köpsignal i flera aktier under samma dag. Framtida arbete skulle kunna vara att utveckla komponenten i systemet som ska koppla upp det mot en mäklare (t.ex. Nordnet), vilket möjliggör aktiehandel i realtid. Utöver en uppkoppling mot mäklare kan natural language processing (NLP) utnyttjas för att klassifiera och bestämma vilka nyheter som ska tolkas som köpsignal.

Nyckelord: webbskrapning, data science, maskinlärning, aktier

TABLE OF CONTENTS

NOMENCLATURE AND ABBREVIATIONS

Dividends - A way of distributing company earnings to the shareholders of a company.
Efficient market hypothesis (EMH) - A hypothesis about the efficiency of stock pricing.
Indicator (stock market) - A way of representing some part of the dynamics of stock pricing/volume data.
Jupyter notebook - A browser based interactive IDE.
Kaggle - An online site for machine learning competitions.
Leverage (stock market) - Makes it possible to buy x worth of stocks with less than x amount invested.
Machine learning (ML) - Enables computers to recognize patterns through computational learning.
Moving average (SMA or MA) - Indicator which mathematically represents an unweighted rolling mean.
Natural language processing (NLP) - A way of processing human language, to make it understandable for computers.
Numpy - Python module for fast numerical computations.
Pandas - Python module that can organize data in data structures. It allows for easy data manipulation.
Price - will be used interchangeably with closing price, if nothing else is explicitly stated (such as opening price).
Risk-adjusted return - Metric that accounts for both risk and return. Sharpe ratio is an example of a risk-adjusted return metric.
Schema (database schema) - A way of defining the structure of data stored in a database.
Regular expression - A standardized way of finding patterns and certain substrings in a string of characters.
Scikit-learn - Python module for machine learning.
Simulated out of sample test - In this project, simulated out of sample test means that the test set used “is yet to come”, from the training set point of view.
TAlib - Python module for technical analysis which has a great variety of different indicators.

1. INTRODUCTION

Data science in combination with the stock market provides an interesting learning platform. By creating a framework for working with data related to the stock market, ideas and theories can easily be tested and implemented. As the title suggests, focus will be on making data the driving force of a trading system.

1.1 Background

This thesis is done as an in-house project at Wibelius Consulting AB and is supervised by Niklas Wibelius. Wibelius Consulting AB is a consultancy company specialized in software-, hardware- and business development. The company is currently in the process of expanding its area of expertise to be able to provide services related to financial technologies.

Trading platforms can be costly and often lack flexibility. Tying the data analysis to a widely used programming language (e.g., Python), rather than a trading platform, can increase the flexibility. Creating a flexible framework can make it easier to produce trading strategies that differentiates from the norm, which in turn can give an edge. Additionally, utilizing uncommon data sources can increase differentiation from the norm even more.

1.2 Purpose

The purpose of this project is to develop components that together form a simple data-driven stock trading system. Figure 1A illustrates the different components in question (excepting the live trading, which will be omitted and left as a possible future component to develop).

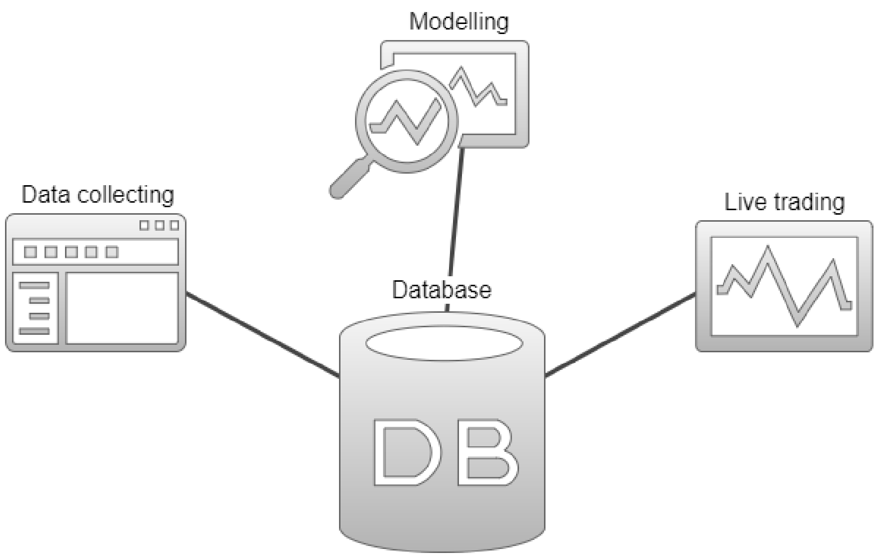


Figure 1A*. An illustration of the system to be developed. Note that the component handling live trading is not a part of the purpose nor goal.*

1.3 Goal

1. Develop software to automatically collect and store news and pricing data.
2. Using the collected data, develop models and tools to aid/perform stock trading decisions.

1.4 Limitation

- To develop the part of the system that connects it to a broker API will not be a part of the goal. Thus, no live trading will take place.
- Existing libraries and frameworks will be used as much as possible.
- The approach to different modelling techniques will be on an applied level. Hence, how a model works mathematically will not be discussed in depth.
- The correctness of the collected data cannot be guaranteed.

2. DATA SCIENCE

Data science is a field that tries to derive insights or predictions by analyzing data. In his work “The future of data analysis” from 1962, John W. Tukey wrote:

> All in all, I have come to feel that my central interest is in data analysis, which I take to include, among other things: procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data [1, p. 2].

He discussed this further by stating “I would regard it as a science” [1, p. 6], referring to the above-mentioned interest. This is the first time someone in an academic paper tried to consolidate and bring together data-driven analysis techniques and the surrounding work into a single field or concept. What Tukey was trying to articulate was probably what the term data science today stands for.

The term (and field) data science is not widely accepted, nor well defined. There are different ideas about what it is, or if it even should exist as its own field. However, academic institutions have now started adopting it, for instance as master programs [2] and other initiatives [3]. To bring clarity throughout the coming chapters, data science will hereafter be defined as “to gain insights into data through computation, statistics, and visualization” [4, p. 4].

2.1 Data science workflow and components

The need for a correct methodology and workflow cannot be stressed enough when working with data. To mention a few possible fallacies when working on a data science problem: Data leakage, overfitting, underfitting and wrong assumptions made regarding the data. By falling into one of the many traps, faulty conclusions can be made.

There are numerous ways to approach a data science project. However, the key components stay the same. Without getting specific about any order, below are some of these components. This approach is inspired by Harvard’s data science class CS109 online material [4, p. 39].

2.1.1 Asking a question or stating a hypothesis

One of the core components of a data science project is the question (or hypothesis). It will guide the rest of the work. However, asking the right question can be hard. E.g., a potential question can be “What’s the strategy that yields the best return?” Depending on how “return” is defined, this might not be an optimal question, since it doesn’t account for risk. A better question might be “What’s the strategy that yield the best *risk-adjusted* return?” Moreover, to take it even further, some trading strategies can be stressful and consequently reduce the quality of one's life. If quality of life is the reason why a trading strategy is developed and used in the first place, taking stress into account might be needed. From this example, a good lesson learned is that the quality of the question will unambiguously determine the value of the answer. In other words, a high-quality answer cannot outweigh the poorly asked question it answers.

2.1.2 Collecting the data

The complexity of collecting data varies a lot. Sometimes downloading an already existing file of a few kilobytes will suffice. In some instances, text from hundreds of millions of webpages needs to be collected and preprocessed. To give easy access to data, organizations provide APIs (Application Programming Interface). Not all organizations provide APIs, which means that the data needs to be collected by other means. This is where web scraping and web crawling enters the picture.

According to [5], web scraping means gathering data programmatically, without the use of any API (Application Programming Interface). Usually, this is done by writing automated programs that interact with a web server. Content available on the web, which people are meant to read using a web browser, can be gathered by utilizing web scraping. The targeted data can either be structured (e.g. tables of containing today's weather data) or unstructured (e.g. blog posts). When data gathering involves software based and systematic browsing of several websites, this is referred to as web crawling. Search engines use web crawling to find index content from the Internet [6].

Scrapy is a Python module (software library) with a framework in place for web scraping and web crawling. It consists of different components, such as spider, item, pipeline and scheduler. The scheduler sends requests to a website which the spider later receives the responses for and parses them. The spider is meant to create items containing the data of interest and pass it on to the pipeline. Items gets passed on to the pipeline, which among other things can be used to filter out items. The final purpose of the pipeline is to display data or store it.

Web scraping/crawling should nevertheless not be brought up without a word of caution. Firstly, some websites explicitly state scraping/crawling as disallowed. This is done by making rules accessible on www.exampledomain.com/*robots.txt*. Secondly, the requests made to a website can in different ways cause harm. Further reading about precautions to take is needed, before one starts web scraping/crawling.

2.1.3 Storing the data

When working with data, storing it in files can be convenient and simple. With the python module pandas, it's possible to import data from a file with a single line of code, namely *pandas.read_csv("data.csv")*. With regards to scalability and some other factors, this might not be the best option in the long run. Nevertheless, in a phase of exploration, it will probably suffice (and possibly even keep a project lightweight and agile).

When designing a long-term solution, with data being added continuously, a database might be needed. A relational database management system (RDBMS), such as PostgreSQL, can be useful when working with data. The strength of a RDBMS is also its weakness; it enforces data to adapt to a schema. Sometimes enforcing schemas is good, because you expect the data to be structured in a certain way. On the other hand, sometimes the data collected is unstructured. NoSQL (Not only SQL) is another type of database system, which utilizes a non-relational architecture. MongoDB is a NoSQL database system, which doesn't enforce any schema. This makes it flexible to design and work with.

2.1.4 Understanding the data

You are provided with an array of numbers, without getting any context as to what it represents. The goal is now to predict which the next number will be. This problem can be approached in many ways and the prediction made might even be correct. But what if you were told that the array is a time series, containing price data for a stock? Now, you know for certain that the number cannot be negative. Additionally, you now know that the array is ordered. If you also happen to know something about the dynamics of stock price movements, this will most definitely help as well. Depending on the situation, understanding the data might help, or even be essential to successfully analyze it.

Understanding how data was collected and sampled is also important. It sets the foundation for a properly made analysis. E.g., when testing trading strategies with historical data, a common way of choosing the historical data is from the current OMXS30 index list. The list consists of the 30 most actively traded stocks (by turnover) on Nasdaq OMX Stockholm. This will however not be a good representation of how well OMXS30 will perform in the future. For example, "Fingerprint Cards" is currently on the list, but wasn't a few years ago. The total return from five years ago until today (as of 2017-08-08) is over 5000% [7]. If a strategy was made for OMXS30 stocks, Fingerprint Cards wouldn't have been traded at that time, since it didn't enter the list until January 4th, 2016 [8]. A better choice of dataset would be to only include historical data for stocks when they were included in the OMXS30 list. Even though it sounds trivial that the current OMXS30 list is not fully representative of OMXS30 past performance, it can be hard to spot these type of assumptions. Especially without domain knowledge about stock markets.

2.1.5 Preparing the data

Once the data is collected, it is to be preprocessed so that it's easier to store and work with. One of the parts of preparing data is cleaning it. This can consist of anything from decoding/encoding text, converting strings to integers, changing short forms to its original form, to filtering out insignificant parts of the data. Sometimes there's missing data which needs to be handled somehow.

Even though data should mostly be cleaned before storing it, it's a good practice to also store the raw format of it as well, since information might otherwise be lost. If for instance a new machine learning model is developed that depends on raw data rather than cleaned data, it should be possible to access the raw data easily.

Normalizing is another part of the data preparation. Some models work better with unit variance and zero mean. Some work better when data is rescaled to fit in the range from 0 to 1. Other models work well without any type of normalization. In the case of stock pricing, normalization can be a good practice. It's otherwise hard to compare stock A with a price of 5 units to stock B with a price of 200 units. An interesting metric is rather how the stocks performed (percentage increase) from day X to day Y.

Feature engineering prepares data in a way that a machine learning algorithm easily can find patterns. In statistics, features are called synthetic variables. If a dataset includes titles of people, such as "Mr Peter Anderson", "Mrs Olivia Robertson" and "Ms Tina Roper", one idea for a feature would be "male" or "female".

Using features is a common practice in stock trading. Features are instead called indicators and can be used for visualization purposes as well as a mean to prepare the data for modelling. An example of an indicator is the simple moving average (SMA or MA), which shows the rolling mean of the close price over a certain time period. It can be a way of reducing "noise", which in this case is fast price fluctuations.

Usually, MA is used together with current close price. Comparing close price with a MA is a well-known practice within the world of stock trading [9]. Let $CSMA = \frac{close\ price}{moving\ average}$. Depending on the characteristics of CSMA, it is possibly ready to be used as a data input for a model, or it might need further preprocessing.

Feature engineering has shown to be an effective way of improving the performance of a model. "For most Kaggle competitions the most important part is feature engineering, which is pretty easy to learn how to do" [10] - Tim Salimans, two time Kaggle competition winner. This quote makes a compelling argument about how important feature engineering is. However, it's important to point out that the machine learning competitions held by Kaggle are given an already well defined problem domain. The wanted output is already determined and sometimes domain insight is given or otherwise open for discussion among participants [11].

2.1.6 Exploring the data

Exploring data is a good way to get familiar with it. This can be done by looking at the raw data, plot it and make histograms of the distribution. Are there any patterns? Are there any anomalies? Sometimes, it might be difficult to come up with a question or hypothesis about a dataset. By exploring the dataset, interesting questions might arise.

The histogram below (Figure 2A) shows the distribution of the close-to-close price development for the stock SCA B (Svenska Cellulosa AB). Close-to-close price development is calculated by the close price of a given day divided by the close price of the day before it. The distribution looks somewhat Gaussian. Notice that there is an anomaly close to the center of the distribution. It is much more probable that the close-to-close price development is slightly higher than 0% as compared to slightly lower than 0%.

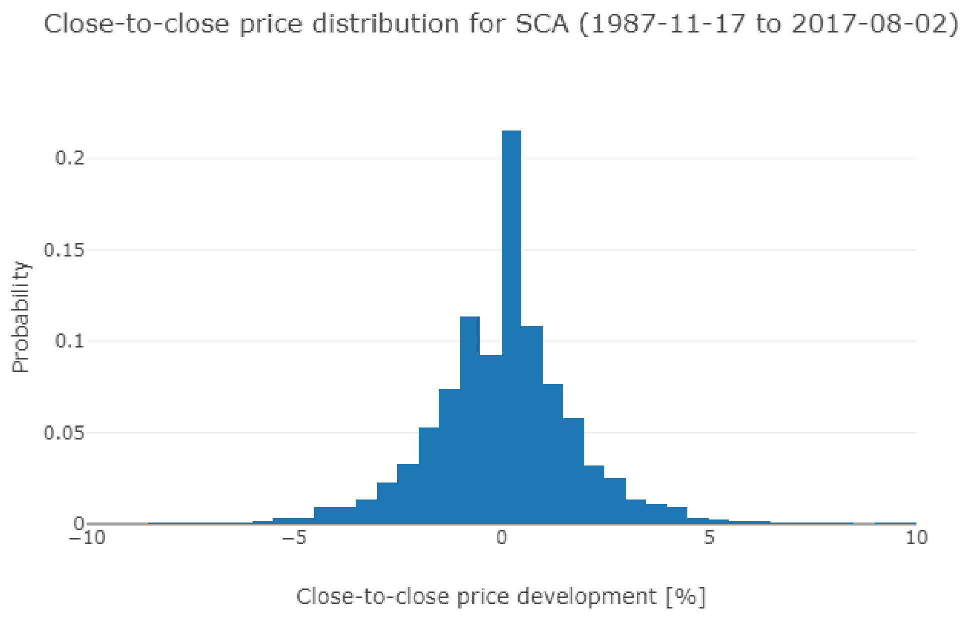


Figure 2A. *Shows a distribution histogram of the close-to-close price development in the stock SCA (Svenska Cellulosa AB). The data source is Nasdaq OMX, with a time period chosen to be all available data (1987-11-17 to 2017-08-02).*

2.1.7 Modelling the data

According to [12], a mathematical model is “a representation in mathematical terms of the behavior of real devices and objects.” In this case, the object (or system) to be described is the stock market and stock price movements in particular. The model doesn’t have to describe the complete behavior of the system. George E.P. Box, a British statistician, famously wrote a paper containing a section with the title “All models are wrong but some are useful” [13, p. 2]. He ends the section with:

> Now it would be very remarkable if any system existing in the real world could be exactly represented by any simple model. However, cunningly chosen parsimonious models often do provide remarkably useful approximations. For example, the law PV = RT relating pressure P, volume V and temperature T of an "ideal" gas via a constant R is not exactly true for any real gas, but it frequently provides a useful approximation and furthermore its structure is informative since it springs from a physical view of the behavior of gas molecules.
> For such a model there is no need to ask the question "Is the model true?". If "truth" is to be the "whole truth" the answer must be "No". The only question of interest is "Is the model illuminating and useful?" [13, pp. 2-3]

This section provides insight on how a model can be useful, even though it cannot describe the complete behavior of the system in question. The real question is if a model can provide value. As a comparison to theoretical models, such as the one described above, models can be created using empirical data. This can be done using a variety of tools/methods, for instance machine learning.

Closely related to asking the right question is stating a correct optimization problem. An optimization problem or metric can be defined by a cost function. It can be fairly simple to define the loss function. Beware though, the phrase “Be careful of what you wish for, you might just get it” has never been more appropriate. Nick Bostrom, a Swedish philosopher, wrote in 2003 about a seemingly harmless “paperclip maximizer” problem. The moral of the story is that if an objective is described, for example “produce as many paperclips as possible,” a possible way of pursuing the given objective could be to transform all land on Earth to factories and to use all humans as raw material [14]. This example is of course an extreme one, but it does point out the importance of defining the optimization problem. It also assumes that the optimizer isn’t limited in what possible actions it can take to achieve its goal. In machine learning, the possible actions to take and available resources are usually limited.

One example of a machine learning model is the so called random forest. It utilizes a learning method called supervised learning. Supervised learning effectively means that the training of a model is done by giving it both the wanted output (answer) and the corresponding inputs. Exactly how the model learns from this information varies.

2.1.8 Privacy and dangers of data science

The example given earlier about feature engineering gender/sex (male or female) can be relevant in some instances and irrelevant in others. For instance, sex is highly relevant when looking for breast cancer and prostate cancer. On the other hand, it might be a good idea to make gender/sex irrelevant when looking through resumes for good job candidates. In some cases, sex/gender can be inferred by looking at other parts of the data. E.g., if the sex is removed from a dataset with medical journals, there might still exist data regarding how many prostate cancer and breast cancer checkups an individual previously had. These are clear examples that highlight the possibility of discrimination. However, the discrimination or harmful biases might be subtler and more hidden than this. With a more complex and comprehensive dataset, such as millions of news articles, biases can stay well hidden in the data and influence the insights derived from the dataset. Asking the right question is the first step to getting a high-quality answer, but the question is to be answered using data. Therefore, if empirical data containing biases is used to train a model, the answer will most likely contain biases. E.g., a model is created to predict (or even decide) who the next president of the U.S. will be. By training the model using data about former presidents, and having gender among the feature inputs, the model will most likely be highly biased against women. In the case where models make decisions, the biases might get reinforced. However, if done right, these biases can be found and accounted for [15].

Some data can be determined as safe to share (with people or software applications). Smartphones are usually equipped with accelerometers, which captures orientation (tilting) and movements. A research team at Georgia Tech used an iPhone 4 to record accelerometer readings while it was placed next to a computer keyboard. Using the data readings, they were able to decipher sentences written on the computer keyboard with up to 80% accuracy [16]. This discovery shows how a seemingly harmless sensor can become a privacy and security issue.

3. STOCK MARKET

A *stock market* is a marketplace where investors (individuals and organizations) can trade (buy and sell) shares of companies. To be able to trade shares, a connection to a stock exchange API is needed. This can be done directly to the exchange, but is usually done through a broker, such as Avanza or Nordnet. There are a few different stock exchanges in Sweden; Nasdaq OMX Nordic and AktieTorget among others. Both Nasdaq OMX and AktieTorget opens at 09.00 and closes 17.30, apart from weekends and certain holidays. Every day, the closing price is recorded, as well as the highest/lowest price of the day. Nasdaq OMX stores opening price data, while AktieTorget doesn't.

The companies listed on the Swedish exchanges ranges from multinational corporations valued at hundreds of billions of SEK [17] to small companies valued at less than 50 million SEK [18]. Liquidity is a term that describes how easy it is to buy or sell a stock, without affecting the stock price [19]. Generally, stocks with higher valuation will inherently have better liquidity.

The price of a stock is essentially determined by supply and demand. When buyers dominate, the price goes up and vice versa [20]. A stock price is dependent on the information available for the investors. When new information is made available, such as news that influences the outlook of a stock, the stock price will most likely adjust. According to "Efficient capital markets: A review of theory and empirical work" by Eugene F. Fama, the stock price always reflects all current available information [21, p. 383]. The paper proposes three different levels of market efficiency. One of them being the weak form efficient market hypothesis (henceforth weak form EMH). It argues that future asset (e.g. stock) pricing cannot be predicted using historical pricing data [21, p. 388].

Another level of the hypothesis is the semi-strong form EMH. It proposes that all publicly available information, such as historical pricing data, news and other information regarding the asset, will be fully reflected in the asset pricing [21, p. 388]. Notice that this hypothesis is a superset of the weak form EMH. Thus, if the weak form EMH is nullified, so is the semi-strong form EMH.

3.1 Technical analysis

Technical analysis (hereafter TA) is a common practice in stock trading [22]. It utilizes historical pricing data to predict future pricing. Note that this is exactly what the weak form EMH proposes to be unfeasible. Even though it is difficult to completely nullify the weak form EMH, the fact that TA is a common practice casts doubt upon it.

To give an insight into what TA is, let's take the example of moving average (MA). MA is a simple so called indicator, commonly used in TA [23]. Indicators are meant to capture some of the dynamics of price and volume change over time. In Figure 3A, price and MA200 (MA using the previous most 200 days) is visualized. Notice that when price is trending either upwards or downwards, MA200 lags. The stock can be defined as in a “positive trend” when price is higher than the MA200 and as in a “negative trend” otherwise. A simple strategy can be developed, which buys this stock shown below when a positive trend is started and sells it when entering a negative trend. By excluding trading fees and dividends from the equation, this simple strategy would approximately yield a 380% return, meaning 1000 SEK invested would end up as 4800 SEK. Meanwhile, if a buy-and-hold strategy was used instead, the corresponding result would be approximately -6% and 940 SEK. These results should however not be presented without clearly warning the reader; There are many, many possible fallacies when drawing conclusions from looking at historical prices. A robust way of testing strategies is needed.

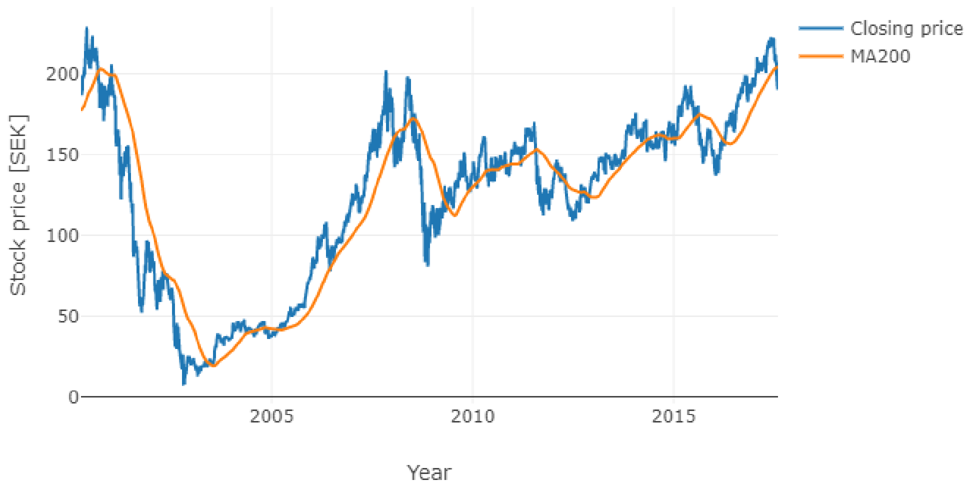


Figure 3A. *The closing price each day is shown together with MA200. A trend can be defined as positive or negative when the MA200 lags. This can be used to create a simple trading strategy.*

In the example above, price together with an indicator was used to make a strategy. Sometimes, several indicators can be used in combination to create strategies. There are indicators to measure price volatility (e.g., Bollinger Band® [24]), price range (e.g., Stochastics [25]) and many more aspects of the price and volume.

3.2 Fundamental analysis

Accompanying the technical analysis is the *fundamental analysis* (hereafter FA). According to [26], FA is “a method of evaluating a security in an attempt to measure its intrinsic value, by examining related economic, financial and other qualitative and quantitative factors”. The analysis can be as simple as looking at the price-earnings ratio [27]. A complex analysis can be performed by taking hundreds of different aspects into account, such as macroeconomic events, competitor outlook and news releases regarding the company in question. As mentioned earlier, the semi-strong form EMH proposes that publicly available information will be fully reflected in the stock pricing. This goes against FA, if the FA is assumed to be performed using only publicly available information.

4. SYSTEM AND METHOD

The system to be developed will consist of a few components. The components will together create the foundation for a data-driven trading system. As stated in the limitations, the live trading part will be left for future work. Focus will instead be on developing tools for data collection, storage and modelling. First, a tool for automatically collecting and storing news data from AktieTorget is developed. Second, using knowledge from the first tool, another tool for collecting and storing stock pricing data is developed. Third, a tool for general analysis of price movements before and after a set of events is developed. To test this tool, an experiment is conducted, which utilizes the previously collected and stored pricing and news data from AktieTorget. Since the experiment turned out to contain faulty assumptions about the data, a modified version of the experiment was conducted. Fourth, a tool for analysis and modelling of stock data is developed. To test it, an experiment is conducted by developing a model and testing if it can predict future pricing.

4.1 News data collection and storage

The objective is to collect news data from AktieTorget and store it in a database. There were several reasons as to why AkieTorget was chosen. Firstly, there are no policies against automatic scraping and the robots.txt allow bots to request news articles (as of 2017-08-13). Secondly, all deals made on AktieTorget are recorded and made publicly available, which among other things, makes experiment replication easier. Lastly, greater stock liquidity is presumed to roughly translate into more interest in the stock. Subsequently, obvious patterns to trade on are rapidly found and exploited.

As a way of collecting the data, Scrapy is used together with MongoDB. One of the core components of Scrapy is the “spider”. The spider, which is implemented as a Python class, holds information about what links to follow as well as parsing the incoming http response html. In this case, the link following rule is defined using the regular expression “NewsItem\.aspx\?ID=\d+”. The website domain is omitted since the spider later adds it. The “\d+” is the regular expression for the integer mentioned above, which in this case is the unique identifier for every news article.

The link following rule will only tell the spider what links to follow while already on a page. What it doesn’t tell the spider is where to begin scraping. That’s where the start url(s) enters the picture. The only start url needed here is “http://www.aktietorget.se/News.aspx?Page=1&Show=25000”. Since there are currently about 21000 news available on AktieTorget, this will expose all news article links at the same time, which is a mere 15MB of html data. It could, for instance, be divided into 10 start urls of 2500 news articles to show on each. However, if there are news released in between the http requests of these pages, some news might jump from one page to another. Thus, it cannot be assured that all news are scraped.

Every time the spider follows a link and get an http response, it will pass the response to a method as an argument. The method's purpose is to pass on an "item" to the item pipeline. A *Scrapy item* is a class with untyped fields. In this case, every item is populated with a headline, url, time of release, time of scrape, news body and raw html. The raw data can come in handy if some other information is needed later, such as html keyword tags etc. In that case, no http requests are needed and the work can be done without involving Scrapy.

Before populating the fields, the http response needs to be parsed. An easy, yet not very efficient, way of extracting the wanted data is to use regular expressions. Online tools like regextester.com/ together with the raw html will make the construction of regular expressions rather simple. To extract news articles from AktieTorget, the regular expression can look like this:

```
(<h1 style[\s\S]+>)([\s\S]+)(<\/h1>[\s\S]+class="ingress">)([\s\S]+<\/h2>)([\s\S]+<\/div>)([\s\S]+)(Publicerat: )([\d\- :]+)
```

A regular expression can be subdivided into groups using parentheses, where some groups holds the data of a field. If there is a matching substring to the regular expression, group 2 will hold headline, group 4 and 5 will hold the body and group 8 will hold the time of release.

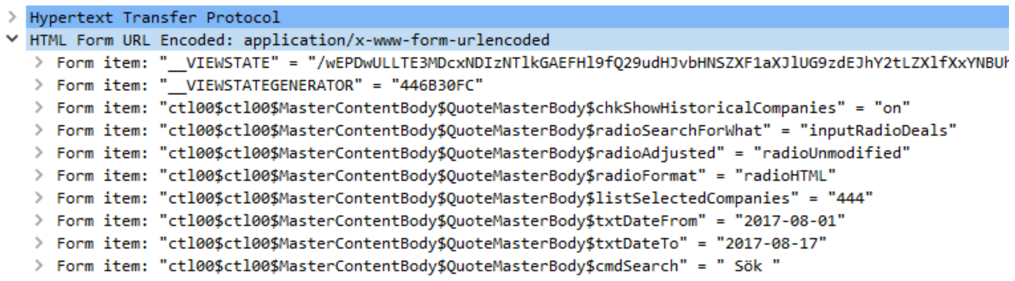
The time of release is at this point a string with the format "YYYY-MM-DD HH:MM:SS". This can either be stored as a string or a datetime object. MongoDB can interpret and store Python datetime objects. To convert the time of release string to a datetime object, Python module ciso8601 can be used. It's a fast way of converting datetime strings of various formats to a Python datetime object [28]. Because of other dependencies in the overall system, the current version of the software is implemented by using the original datetime string.

BeautifulSoup is another scraping module for Python. In this case it is only used to remove html tags from the text (in the different groups). Once html tags are removed, the data is clean and ready to be stored. As mentioned earlier, this is done by creating an item, populating its fields and then simply letting the method return the item. By returning the item it gets passed on to the item pipeline. The Scrapy item pipeline connects to MongoDB using the Python module pymongo. Scrapy documentation includes code for the interface between Scrapy and MongoDB at [29]. Every item passed to the item pipeline will be inserted to the wanted collection in the database. Now, the Scrapy spider can be started and the news articles will be collected and then stored in the MongoDB.

4.2 Pricing data collection and storage

The next objective is to collect *pricing data* related to the collected news. AktieTorget provides full order book publicly. Every time a stock is traded (changes owner), the details about the deal can be found on AktieTorget's webpage at [30]. The deal data is hereafter referred to as *intraday data*.

The intraday data is normally collected or viewed manually. The data is however not as easily scraped as the news data, since there are a few user interactions needed before it can be displayed. The important user inputs are: choose stocks from a list, choose the wanted time frame and if it should reply with end of day or intraday data. To be able to collect the intraday data using scraping, one needs to find out how and what user input data is sent from the web browser client to the web server. The first try to find this data was done by analyzing the html and JavaScript code for the web page. It gave some initial insights, but was unfortunately not enough to solve the problem. Next attempt was made using a network protocol analyzer called *Wireshark*. By logging the packets sent by the web browser, the user input data gets exposed. Figure 4A below shows an example of how a packet with user input data can look like.



```
> Hypertext Transfer Protocol
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "__VIEWSTATE" = "/wEPDwULLTE3MDcxNDIzNTlkGAEFHl9fQ29udHJvbHNSZXF1aXJlUG9zdEJhY2tLZXlfXxYNBUl
  > Form item: "__VIEWSTATEGENERATOR" = "446B30FC"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$chkShowHistoricalCompanies" = "on"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$radioSearchForWhat" = "inputRadioDeals"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$radioAdjusted" = "radioUnmodified"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$radioFormat" = "radioHTML"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$listSelectedCompanies" = "444"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$txtDateFrom" = "2017-08-01"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$txtDateTo" = "2017-08-17"
  > Form item: "ctl00$ctl00$MasterContentBody$QuoteMasterBody$cmdSearch" = " Sök "
```

Figure 4A. *The figure shows the packet of interest, which was found using Wireshark. The packet contains an html form which holds user input information.*

Scrapy has native support for sending html form requests. Therefore, the mimicked user input can be incorporated in a spider, in a similar way as with the spider mentioned earlier. Scraping news meant parsing a single page for links and then following them. The intraday data spider needs to choose stocks and time period to show. There are roughly 500 stocks in the list and for every stock there's a time period for which it's been traded. The intraday data is, for some of the stocks, to comprehensive to show in a single web page. By showing the intraday data a month at a time, it will fit on one page. The intraday data is shown from the year 2000 and onward. To brute force all pages for all stocks, it would take roughly 100 000 http requests ($500\ stocks \times 18\ years \times 12\ months = 108\ 000\ http\ requests$). With a time delay of two seconds between requests, it would take 60 hours to complete the scraping. Since stocks gets listed and delisted, many of the requests will return no intraday data. To reduce load on the web server as well as scrape time, a better way needs to be found. This can be done by dividing the scraping task into the three following steps:

1. Send an http request for the web page containing the form at http://www.aktietorget.se/QuoteSearch.aspx?Language=2. This is to obtain the list of stocks, which is included in the http response message. By parsing the http response's html code with regular expression, the list is extracted. The previous regular expression found was meant to find one match. This regular expression is meant to find all matches within the html code.

2. All the end of day data for a stock fits in a single page. For every stock in the list, a form request is sent with a time period enclosing all dates available (2000-2017). Moreover, end of day data is asked for, instead of intraday data. This will yield all dates that are available for that stock.
3. Brute force is now avoided, since the first and last dates from part 2 determine the first and last month to scrape intraday data for. Form requests are sent to iteratively collect intraday data, stock and month wise. The http response's html code is then parsed and values are extracted with regular expression. Every deal made is then inserted to the MongoDB server as a document. The document includes ticker (a short security identifier, such as "SLOTT A"), time of the deal, amount of securities traded and price.

The pricing data was collected and stored using the above-mentioned steps. MongoDB document IDs will be set as the primary key and is the only field to be initially indexed. When querying the intraday data, ticker together with time period is useful for sorting and filtering. The database can therefore be optimized by indexing ticker and datetime. A simple query to get the intraday data for a stock on a single day takes 19.04 seconds to perform. By indexing ticker and datetime, the same query takes 0.58 seconds. This should of course be seen as a rough comparison.

4.3 Price distribution over a set of events

There are many ways to come up with a set of events that are of interest. Trading strategies and news releases are among them. To visualize a price movement from a single event can be insightful. It is, however, not statistically significant. The stock market has a lot of variance. Let's instead take a set of events and calculate the mean (μ) and standard deviation (σ) at each time step. The mean and standard deviation is visualized with timestep t=0 being the time of the event.

To provide a set of events, the model developed during experiment 3 (see chapter 4.4.1) is used. The number of events in the set is 17, which is low and should thus not be used to draw too many conclusions. Furthermore, using standard deviation as a measurement of variation might not be fully optimal, since it assumes the price distribution to be Gaussian. Below is Figure 4B, which illustrate the how the visualization works. Notice that there's a sudden price spike from t=0 to t=1.

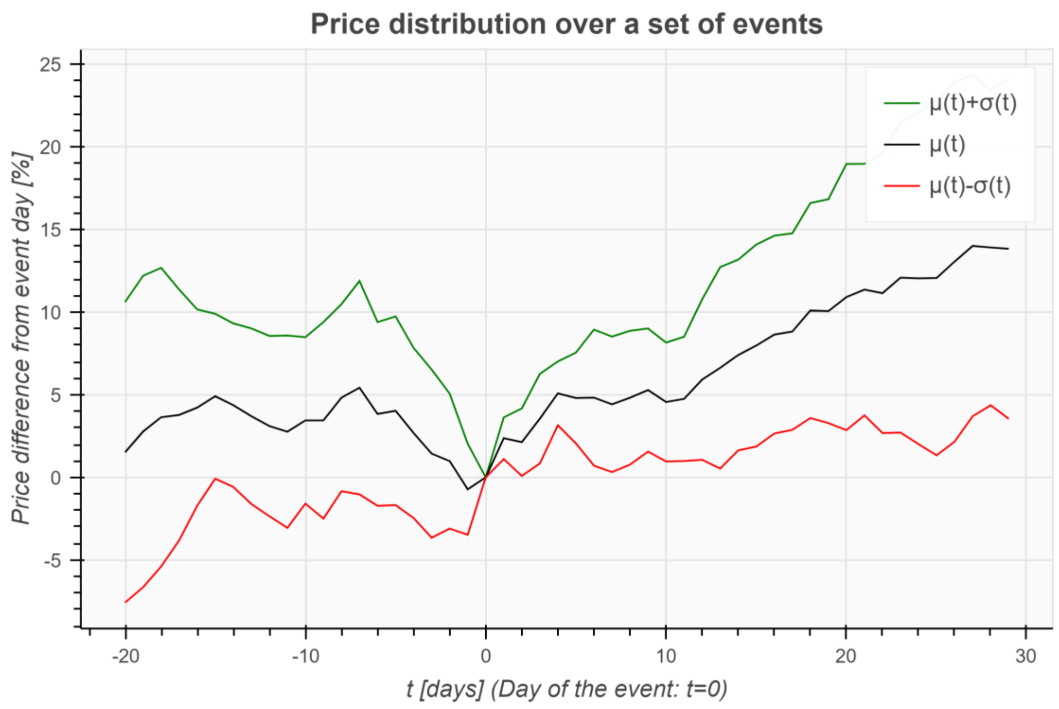


Figure 4B. *This visualization is meant to communicate a summarized view of how the price develops over a set of events. It shows both mean price and standard deviation, at every discrete timestep t.*

4.3.1 Experiment 1 - Testing impact of news on stock pricing

Observation: Events, such as news releases (e.g. a news article stating a patent was approved for company X), seem to trigger stock price movements. A part of stock price adjustments seems to be delayed, instead of fully adjusting for the newly added information immediately. This suggests that the semi-strong form EMH does not apply.

Hypothesis: The return as well as the risk-adjusted return can be above average when trading a stock after a news release. Even if the stock is bought t_{entry} time units after it was released.

Method: The hypothesis is tested by visualizing the price movement around news releases, from t_{before} time units to t_{after} after. The data used in the experiment will be obtained from the MongoDB-database. The stock is "bought" t_{entry} after the news release. t_{exit} time after the news release, the stock is "sold". As mentioned earlier, price movement after a single event is not statistically significant. Therefore, mean price μ(t) is calculated using several news releases that contain the same keyword (e.g. "patent"). In addition to mean price, standard deviation σ(t) is also calculated. Risk-adjusted return is determined by Sharpe ratio with the following formula: $SR = \frac{AR-RF}{SD}$, where $SR = Sharpe\ Ratio$, $AR = Average\ Return$, $RF = Risk\ Free\ interest\ rate$ and $SD = Standard\ Deviation$. Since the time horizon is short, risk-free interest rate can be approximated to be zero. The risk-adjusted return is therefore calculated using $SR = \frac{AR}{SD}$. An important note is that the average return and standard deviation should be calculated from t_{entry}, not t=0. In addition to the risk-adjusted return, average return is also included in the hypothesis. To compare both return and risk-adjusted return to a benchmark, random samples of “non-events” will be taken from the same stock.

Limitations:

- The pricing used is taken from the deals made. This does not fully reflect if the fictive buying and selling done at t_{entry} and t_{exit} could have been performed at those price levels. On the other hand, if the liquidity is high enough, fictive buying and selling price levels can be approximated by the price levels of deals made.
- By binning the deals made, time at a given timestep does not reflect the price at that exact moment. Instead, it reflects the price of the deals made inside that particular bin. To avoid data leakage, price at time t is determined by the price within the period $t - binsize$ to t. This can be put in contrast with binning made using $t - \frac{binsize}{2}$ to $t + \frac{binsize}{2}$ or t to $t + binsize$, which will both leak information backwards in time.
- Related to the binning problem described above is that deals made on t_{event}+1 micro second will be displayed on timestep $t + binsize$. Thus, it’s a good idea to avoid using $t + binsize$ as entry.
- The resolution of news release timestamps is on second level. However, the resolution of deals made is on minute level. The timestamp of a deal is determined by simply removing the seconds (i.e., deal made at 12:52:42 becomes 12:52:00).

4.3.2 Experiment 2 - Testing impact of news on stock pricing (modified)

Observation: Same as in experiment 1.

Hypothesis: Same as in experiment 1, except that risk-adjusted return won’t be covered. Instead, *return* will be the metric of interest.

Method: To tackle the problem with sparse pricing data (see subsection 5.2.1.1), a second experiment concerning the impact of news on stock pricing is conducted. This time, mean price and standard deviation for each timestep will be neglected. Instead, only release time price, “buy” price and “sell” price will be of interest. The first deal after t_{entry} will be used as the “buy” price and the first deal made after t_{exit} will be used as the “sell” price. The release time price is determined by the deal assured to be before the news release.

A plot is made, where lines are drawn between every entry point and its corresponding exit point. The points are determined by price level and time of buying/selling. The plot will only give an idea of price development. The actual result is determined by average return, which in turn is determined by the entry and exit prices.

Limitations:

- The pricing used is taken from the deals made. This does not fully reflect if the fictive buying and selling done at t_{entry} and t_{exit} could have been performed at those price levels. On the other hand, if the liquidity is high enough, fictive buying and selling price levels can be approximated by the price levels of deals made.
- The first deal after t_{entry} will be used as the buy price and the first deal made after t_{exit} will be used as the sell price. These deals are assumed to be representative of the current price level. Since AktieTorget stocks doesn't provide a lot of liquidity, this might be false premises.

4.4 Developing a model to aid/perform trading decisions

By using historical pricing data from some of the OMXS30 stocks, a model to predict the next day's closing price will be developed. Further details will be explained in the experiment below.

4.4.1 Experiment 3 - Testing a trading model using historical stock pricing data

The database (the one populated by scraping news and pricing data from AktieTorget) is not publicly available for querying. Therefore, it's hard to reproduce the experiment made with the data. To contrast the previous experiment, this experiment will be based on data that can be easily collected manually; end of day data from the OMXS30 list. In addition to reproducibility, OMXS30 stocks generally provide better liquidity (as mentioned earlier).

Observation: It seems possible to use historical stock pricing data to develop a trading model that outperforms the market in general, in terms of risk-adjusted return. Consequently, weak form EMH should not apply.

Hypothesis: Using end of day pricing data from OMXS30 historical data, a model can be created that yields a better *risk-adjusted return* per day than average.

Method: To test the hypothesis, a *random forest model* with a binary classifier will be developed. The output is to simply predict if a day's closing price will be 0.5% higher than the previous day's closing price $\left(\frac{Close(t+1)}{Close(t)} > 1.005\right)$. Input for the model will be features created from pricing data. Random forest is chosen because of its simplicity and flexibility.

1. End of day data for OMXS30 stocks is *downloaded manually* from nasdaqomxnordic.com/aktier/historiskakurser. The full list of stocks can be found in "Ticker symbols for the stocks used in experiment 3" (appendix B). The period chosen when downloading data for all stocks is 1900-01-01 to 2017-08-03 (note that the data doesn't actually start at 1900-01-01, it's just a way of catching all data available).

2. *Features are engineered* using pricing data only. These will be used as the inputs for the model. The random forest model will, as a byproduct of the trained model, provide results of how important the different features are. The engineered features are described in the table “Features” (appendix A).
3. To test the model, the dataset is divided into three subsets (training set, test set 1 and test set 2). The last year of the dataset (2016-08-03 to 2017-08-03) is used as test set 2 and will be used for simulated out of sample testing. The remaining data will be used to randomly sample training set and test set 1.
4. When dividing the data into training set and test set 1, price and volume for stock A on 2015-05-02 could end up in the training set, while the corresponding data for stock B on the very same date ends up in the test set. Since stock prices correlate, this might cause the model to overfit. This is solved by putting all the data from a particular date in either the training or test set.
5. A random forest model is developed. Hyperparameters are tuned by trial and error, while examining the performance of the model.
6. The first observable result is available when the model has been trained. To produce the result, test set 1 is used. The real result is obtained from using the model to make predictions for test set 2. The average close-to-close risk-adjusted return for the whole test set 2 will be compared to the average close-to-close risk-adjusted return when the model output is above a certain threshold. To calculate risk-adjusted return, the previously mentioned Sharpe ratio will be used. Just as in experiment 1, risk free interest rate is approximated to be 0. This results in the following formula: $SR = \frac{AR}{SD}$.

Further details on how the experiment was performed is explained by the resulting source code, available as a Jupyter notebook at [31].

Limitations:

- Data is taken from current (as of 2017-08-03) OMXS30 list. This was earlier mentioned as not fully representative of the historical performance of OMXS30. In this case, it’s assumed to not affect the outcome too much.
- Training and test data is randomly sampled from the dataset. If data from day x is used to train the model, data from day x-1 might be used to test the model. To counter this, another test set will be made using the last 365 days of the dataset. The last 365 days is unfortunately a relatively short period of time. Neither of the two types of test set sampling methods are optimal, but combining the results will have to suffice for this experiment.
- Hyperparameters are tuned by trial and error. This might cause overfitting.

5. RESULT

The goal was split into two sub-goals. Both should be considered achieved.

5.1 Develop software to automatically collect and store news and pricing data

The first of the two sub-goals were *to develop software to automatically collect and store news and pricing data*. The software/tools described in subchapters 4.2 and 4.3 is/are functioning according to the set goal. It utilizes Python together with the Python module Scrapy to automatically collect news and pricing data. It then passes the data to be stored on a MongoDB database. Currently, scraping is not performed using any scheduling. Instead, commands need to be entered to start the scraping. This means that the level of autonomy is limited. Since there was a second sub-goal to consider, automatic scheduling was left as a future feature to develop.

5.2 Develop models and tools to aid/perform stock trading decisions

The second of the two sub-goals was to *use the collected data, develop models and tools to aid/perform stock trading decisions*. To achieve the sub-goal, a visualization tool and a machine learning model was developed. The software developed for the tool and model is considered a part of the result. Moreover, the model and tool have experiments tied to them, with results that are relevant for testing if the sub-goals were achieved.

5.2.1 Price distribution over a set of events

The resulting visualization can neatly communicate a summarized view of several price movements, in a single graph. Mean price μ(t) can be used to find a good exit point, for instance when it’s visibly clear that the price movement upwards has started decaying. E.g., in Figure 4B, the price movement plateaus between t=4 and t=12. A good exit point might therefore be at t=4. Furthermore, the standard deviation “bands” $\mu(t) + \sigma(t)$ and $\mu(t) - \sigma(t)$ gives a notion of how wide the distribution is at a given timestep. It gives an impression of how risky it is to buy a stock using the event as trigger. In addition to looking at what happens to the price after a set of events, it’s also interesting to look at what happens before.

The resulting software to visualization price distribution over a set of events mostly prepares the data in different ways. Once the data is prepared, it’s easy to visualize it.

5.2.1.1 Experiment 1 - Testing impact of news on stock pricing

The impact of news on AktieTorget was hard to quantify using the described experimental hypothesis and method. The method contained faulty assumptions about the pricing and news data. It was assumed that there were deals made during every timestep. This assumption was important to the calculation of mean price and its standard deviation at every timestep. There might be ways of neatly dealing with this sparse data problem. However, with regards to the other experiments to perform, the sparse data problem won’t be investigated further. A new and short experiment will instead be performed, which is supposed to be more tolerant to sparse data.

5.2.1.2 Experiment 2 - Testing impact of news on stock pricing (modified)

After trying a few different keywords in combination with different stocks, no patterns of significance were found. The Swedish keywords *avtal, patent, order, uppköp and förvärv* (translation: deal/contract, patent, order, buyout and acquisition) were queried for on news related to the company *Cherry AB*. The key metrics related to these keywords can be seen in the table below (see Figure 5A).

Summary of results for price development (stock: *Cherry AB*)

Keyword (Swedish)	Keyword (English)	Average return (AR)	Standard deviation (SD)	Number of news
Avtal	Deal/Contract	0.28%	1.91%	17
Patent	Patent	-0.34%	-	1
Order	Order	0.05%	0.47%	3
Uppköp	Buyout	-	-	0
Förvärv	Acquisition	0.24%	1.37%	35

Figure 5A. *The database is queried for the Swedish keywords. "Number of news" show how many news were found for each of the queries. For every news article, the underlying stock was fictively bought at t_{entry} and fictively sold at t_{exit}. This yielded an average return as well as standard deviation for the fictive trades performed.*

As seen in the figure above, the number of news are relatively low. Since news are released both during and outside the open hours of the stock exchange, not all them can be used as buy signals. What's more, sometimes there were no deals made in between t_{entry} and t_{exit}. These instances were neglected and left out of the result. With all things considered, the outcome of this experiment shows that a price movement caused by a news release might partly be delayed. Even if a stock is bought t_{entry} minutes after the news release. The experiment is nonetheless minuscule, causing the result only to be slightly indicative whether there's any truth to the hypothesis or not. A more extensive experiment is therefore required for the result to carry weight. The tool made to visualize the result did however serve its purpose (see Figure 5B below).

The news was firstly filtered using the keyword "förvärv". Using only keywords turned out to be a bad way of classifying good and bad news. Therefore, to simulate that a human or computer (that utilizes NLP) was classifying news, they were filtered by simply glancing at the headline for no more than five seconds to determine them to be positive or not. The result was slightly better than simply using the keyword. As a final notice, the figure looks messy (which speaks in favor of the “Price distribution over a set of events” visualizer). After a certain amount of news to plot price development for, it becomes hard to interpret. But it serves a few purposes. First, it surfaces any possible outliers. There seems to be two positive outliers. The clustered part is somewhat hard to make sense of, though. Second, it gives a hint on how long it takes before the first deal is made after t_{entry} and t_{exit}. If the liquidity is low, it will probably show on the visualization.

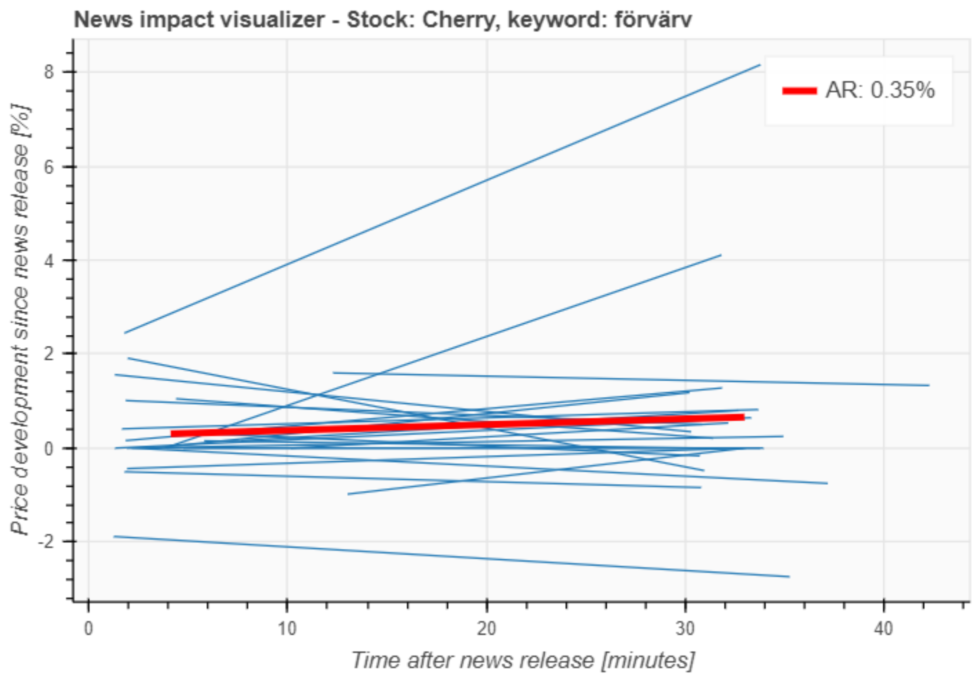


Figure 5B. *A visualization that is meant to give a summarized view of how company news impact the company's underlying stock price. When showing price development for 15 or more news simultaneously, it fails to communicate the summarization in a clear manner.*

5.2.2 Developing a model to aid/perform trading decisions

The model developed seems to have found patterns. However, the biggest gain from developing the modelling is not the model itself. The resulting software that reads data from csv files and prepares it (mostly consisting of feature engineering) can be used together with other types of models. Likewise, adding more features is easily done. The resulting code is available at [31].

5.2.2.1 Experiment 3 – Testing a trading model using historical stock pricing data

The random forest was initially fitted (trained) using 200 decision trees and 10 samples per leaf. By looking at the performance (close-to-close return) when model output was above the threshold, these were changed to 2000 and 15. Using these parameters, and only trading when model output is above the threshold of 0.5 units, the following results were obtained:

Test set 1:
Percentage of days where model output was above threshold: 5.70%
Benchmark Average Return (BAR): 0.12%
Benchmark Standard Deviation (BSD): 2.23%
Benchmark Sharpe Ratio (BSR): 0.05
Average Return (AR): 0.81%
Standard Deviation (SD): 3.38%
Sharpe Ratio (SR): 0.24
Number of buy signals: 1609
Number of days with a buy signal: 720
Total number of days: 28232

Test set 1 provides a first impression whether the model found any patterns. Notice that AR is significantly higher than BAR. On the other hand, SD is slightly higher than BSD. All in all, it adds up to SR being 380% higher than BSR. The training and test data used was sampled from the same period. The result will probably not reflect how well the model would perform on current data. The real question is if the found patterns from this period still applies to test set 2.

Test set 2 (simulated out of sample test):
Percentage of days where model output was above threshold: 1.02%
Benchmark Average Return (BAR): 0.06%
Benchmark Standard Deviation (BSD): 1.66%
Benchmark Sharpe Ratio (BSR): 0.04
Average Return (AR): 0.28%
Standard Deviation (SD): 1.57%
Sharpe Ratio (SR): 0.18
Number of buy signals: 67
Number of days with a buy signal: 55
Total number of days: 6578

Using test set 2, the resulting performance metrics changed drastically. Only 1% of the model outputs were above the threshold, in comparison to test set 1 which had a corresponding figure of over 5%. This meant that the model only flagged 67 out of the 6578 data points as buy signals. The AR dropped from 0.81% to 0.28%. 0.28% can still be seen as a positive result, since it outperformed the BAR. BAR and BSD also dropped from test 1 to test 2, but BSR more or less stayed the same. Test set 1 was expected to provide a better result than test set 2. However, the resulting SR only dropped from 0.24 to 0.18.

The 67 fictive trades made using the buy signals were compounded and a total return of 19.6% was achieved. This was unfortunately based on a faulty assumption that the trades were made sequentially. As seen in test 2, there were more buy signals made than the number of days with trade signals. To tackle this, when there were several buy signals on the same day, the fictive portfolio was equally allocated to the different stocks that produced a buy signal. E.g., if ABB and Autoliv produces a buy signal on the very same date, 50% of the current balance will be allocated to each of them. Surprisingly, this made the total return drop to a mere 2.4%. At a first glance, this might look like a bad result. But it raised an interesting question: If the return drops when clustered signals are made less significant, what does that imply? This question lead to a new discovery about the result.

The 67 buy signals are divided into two sets. One with the clustered signals (signals occurring on the same date) and one with non-clustered signals. The mean outcome of the non-clustered signals was in fact negative (-0.26%). In contrast, the mean outcome of the clustered signals was 1.39%! To better exemplify the difference between the two sets just described, a histogram is displayed below (Figure 5C).

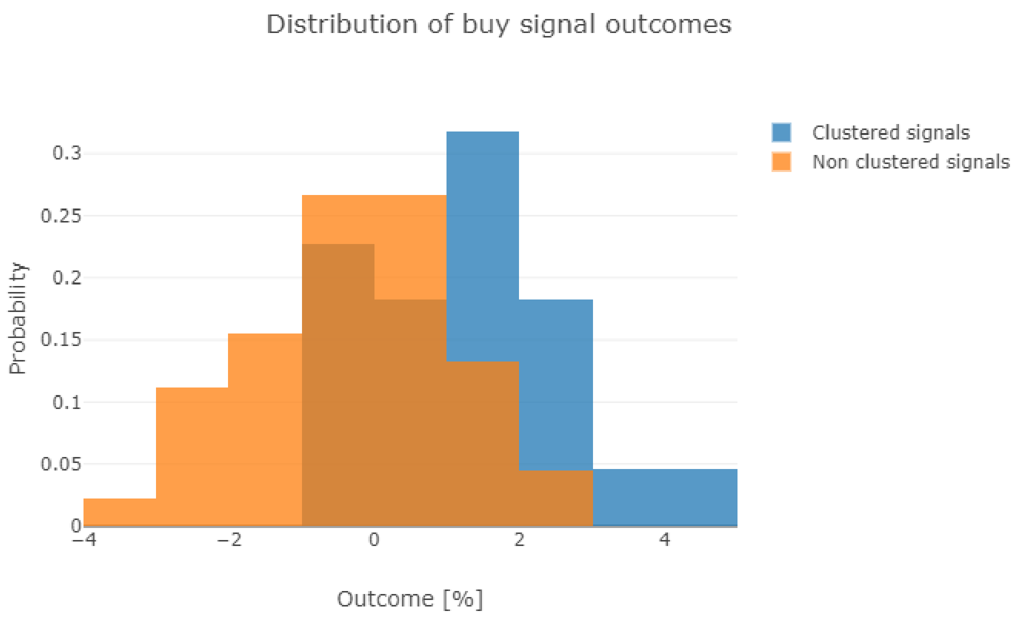


Figure 5C. *The signals produced with test set 2 were divided into clustered and non-clustered signals. The figure clearly shows that the outcome for clustered signals outperform the non-clustered signals.*

As seen in the figure above, the clustered signals are clearly shifted towards positive outcomes. Furthermore, it can also be inferred that less than 25% of the trades had a negative outcome.

6. CONCLUSION AND DISCUSSION

The problem at hand, to develop components for a data-driven stock trading system, presented many challenges. To begin with, the different learning curves for new things: Data science, Python, Scrapy, MongoDB, Pandas, Numpy, etc. At the beginning of the project, having multiple things to learn at the same time made the progress slow going. Towards the end of the project, an acceptable proficiency level for the different things was reached. This meant moving forward in a significantly higher pace. Furthermore, data related to the stock market has a lot of complex dependencies and correlations. During the development of the tool and model, assumptions made about the data were found mid-process. E.g., the model was originally meant to predict 5 days ahead, instead of 1 day ahead. This would cause information leakage, if not properly addressed. The list goes on. In conclusion, the experience from developing components for a data-driven trading system is definitely both fun and valuable.

During the process of developing components, some dead ends were reached. Visualizing mean and standard deviation over time around a set of news releases was one of them. *Price distribution over a set of events* implemented with daily closing prices, did however provide a neat and informative summarization of price development. This was the second best resulting component. The single best component was, without question, the model developed. It was made without much emphasis towards perfecting it. It’s rather comparable to a minimum viable product. This implicates that a better model is within grasp. Because of the recently gained proficiency and already developed software related to the model, it would now take only a few hours to add and test new features. Since the results already hinted that clustered buy signals are strong ones, developing features that detect clustering can be a good start for future work.

The resulting database system and Scrapy software worked as expected. It automatically collects and stores news and pricing data. Querying the database to get insights into the data is easy, especially when software developed for the experiments is reused. The system does on the other hand not automatically collect real time data. What this means is that it needs to be upgraded to be able to make trading decisions in real time.

Experiment number one and two can be seen as inconclusive. They still give the impression that there might be a delayed price development after news releases. To make an assertive conclusion, a more extensive experiment needs to be conducted.

Experiment number three, where a model was developed, did provide an interesting result. The test carried out was made on “new” data, with a positive outcome. The average return as well as Sharpe ratio was above the corresponding benchmark values. However, there were only 67 buy signals, which is a bit low to make an assertive conclusion. It does however imply that the weak form efficient market hypothesis might be faulty. The existence and extent of the TA community is another hint that the hypothesis is faulty. Disproving the EMH is nonetheless out of the scope of the purpose and goal.

As mentioned in section 2.1.8, there are a few ethical aspects to consider when using a data-driven approach towards getting insights. The example where the US president is to be chosen using empirical data about previous presidents sheds light onto one of the problems. It teaches us that a data-driven approaches should be performed with caution, since making decisions based upon it can create dangerous reinforcement loops. In the case of stock trading, the problems are slightly different from the ones mentioned above. For instance, the underlying system (stock market or stock price) can be affected by trades made by an algorithm (or trading model). Therefore, it might in fact be performing trading in an unlawful way. Since the developed model only provides buy signals during the closing of Nasdaq OMX, which is a time of high liquidity, it is highly unlikely that it would be able to perform unlawful trading.

REFERENCES

[1] J. W. Tukey, "The Future of Data Analysis," *The Annals of Mathematical Statistics,* vol. 33, no. 1, pp. 2-6, 1962.
[2] "Applied Data Science Master's Programme - University of Gothenburg", *Utbildning.gu.se/education*, 2017. [Online]. Available: http://utbildning.gu.se/education/courses-and-programmes/program_detail/?programid=N2ADS. [Accessed: Sept. 13, 2017].
[3] "About the NYU Center for Data Science - NYU Center for Data Science", *NYU Center for Data Science*. [Online]. Available: https://cds.nyu.edu/about/ [Accessed: Sept. 13, 2017].
[4] H. Pfister, J. Blitzstein and V. Kaynig, "www.github.com/cs109", *GitHub.com*, 2015. [Online]. Available: https://github.com/cs109/2015/raw/master/Lectures/01-Introduction.pdf. [Accessed: Sept. 13, 2017].
[5] R. Mitchell, *Web Scraping with Python*, 1st ed. Sebastopol, CA: O'Reilly Media, 2015, p. viii.
[6] "How Google Search Works | Crawling & Indexing", *Google.com*. [Online]. Available: https://www.google.com/search/howsearchworks/crawling-indexing/. [Accessed: Sept. 13, 2017].
[7] "Fingerprint Cards AB ser. B (FING B) - Köp aktien på Nasdaq Stockholm AB - Nordnet", *Nordnet.se*. [Online]. Available: http://www.nordnet.se/mux/web/marknaden/aktiehemsidan/index.html?identifier=4870&marketid=11. [Accessed: Aug. 8, 2017].
[8] Affärsvärlden SIX, "Fingerprint tas med i OMXS30," *Affärsvärlden*, Dec. 4, 2015. [Online]. Available: https://www.affarsvarlden.se/bors-ekonominyheter/fingerprint-tas-med-i-omxs30-6752551. [Accessed: Aug. 8, 2017].
[9] "Hjälpmanual | Hitta kursvinnare", *Hittakursvinnare.info*. [Online]. Available: http://www.hittakursvinnare.info/hjalpmanual/?hmhelp=ta_lista_medelvarden.html. [Accessed: Sept. 13, 2017].
[10] D. K. Wind, "CONCEPTS IN PREDICTIVE MACHINE LEARNING," M.S. thesis, DTU, Lyngby, 2014.
[11] Kaggle. "Kaggle: How it Works," *YouTube*, Oct. 28, 2011 [Video file]. Available: https://www.youtube.com/watch?v=PoD84TVdD-4. [Accessed: Aug. 30, 2017].
[12] C. L. Dym, "What Is Mathematical Modeling?" in *Principles of Mathematical Modeling*, 2nd ed. San Diego, CA: Academic, 2004, p. 4.
[13] G. Box, "Robustness in the Strategy of Scientific Model Building", Wisconsin Univ-Madison Mathematics Research Center, Madison, 1979.
[14] "Ethical Issues In Advanced Artificial Intelligence", *Nickbostrom.com*. [Online]. Available: http://www.nickbostrom.com/ethics/ai.html. [Accessed: 23. Aug, 2017].
[15] "Bias in machine learning, and how to stop it", *Techrepublic.com*. [Online]. Available: http://www.techrepublic.com/article/bias-in-machine-learning-and-how-to-stop-it/. [Accessed: 23. Aug, 2017].
[16] "Georgia Tech Turns iPhone Into spiPhone", *Innovations-report.com*. [Online]. Available: http://www.innovations-report.com//html/reports/information-technology/georgia-tech-turns-iphone-spiphone-184116.html. [Accessed: 23. Aug, 2017].
[17] "Hennes & Mauritz B (HM B) | Information om bolaget | Avanza", *Avanza.se*. [Online]. Available: www.avanza.se/aktier/om-bolaget.html/5364/hennes---mauritz-b. [Accessed: 9. Sep, 2017].

[18] "ECOMB (ECOM) | Information om bolaget | Avanza", *Avanza.se*. [Online]. Available: www.avanza.se/aktier/om-bolaget.html/272628/ecomb. [Accessed: 9. Sep, 2017].
[19] "Liquidity", *Investopedia.com.* [Online]. Available: www.investopedia.com/terms/l/liquidity.asp. [Accessed: 9. Sep, 2017].
[20] "Supply and Demand—How Stock Prices Are Set", *Thebalance.com.* [Online]. Available: https://www.thebalance.com/how-stock-prices-are-set-3141289. [Accessed: 9. Sep, 2017].
[21] E. F. Fama. (1970, May). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance* [Online]. *25(2)*. Available: http://www.jstor.org/stable/2325486
[22] "Technical Analysis Drags Down Performance", *Forbes.com.* [Online]. Available: https://www.forbes.com/sites/rickferri/2014/06/02/technical-analysis-drags-down-performance. [Accessed: 8. Aug, 2017].
[23] "Moving Average (MA)", *Investopedia.com.* [Online]. Available: https://www.investopedia.com/terms/m/movingaverage.asp. [Accessed: 9. Sep, 2017].
[24] "Bollinger Band®", *Investopedia.com.* [Online]. Available: https://www.investopedia.com/terms/b/bollingerbands.asp. [Accessed: 9. Sep, 2017].
[25] "Stochastics: An Accurate Buy And Sell Indicator", *Investopedia.com.* [Online]. Available: https://www.investopedia.com/articles/technical/073001.asp. [Accessed: 9. Sep, 2017].
[26] "Fundamental Analysis", *Investopedia.com.* [Online]. Available: https://www.investopedia.com/terms/f/fundamentalanalysis.asp. [Accessed: 9. Sep, 2017].
[27] "Price-Earnings Ratio (P/E Ratio) | Investopedia", *Investopedia.com.* [Online]. Available: http://www.investopedia.com/terms/p/price-earningsratio.asp. [Accessed: 9. Sep, 2017].
[28] "ciso8601 1.0.1 : Python Package Index", *Python.org.* [Online]. Available: https://pypi.python.org/pypi/ciso8601/1.0.1. [Accessed: 16. Aug, 2017].
[29] "Item Pipeline — Scrapy 1.4.0 documentation", *Scrapy.org.* [Online]. Available: https://doc.scrapy.org/en/latest/topics/item-pipeline.html#write-items-to-mongodb. [Accessed: 16. Aug, 2017].
[30] "AktieTorget", *AktieTorget.se.* [Online]. Available: http://www.aktietorget.se/QuoteSearch.aspx?Language=2. [Accessed: 16. Aug, 2017].
[31] "fredrudolf/nasdaq_omx_stock_predicting", *Github.com.* [Online]. Available: https://github.com/fredrudolf/nasdaq_omx_stock_predicting. [Accessed: 19. Sep, 2017].

APPENDIX

Features - Appendix A

Feature name	Formula	Explanation
CC-1	$\frac{Close(t)}{Close(t-1)}$	Price development from yesterday's close till today's close
CBBL1	$\frac{Close(t)}{BollingerBandLower(20,2)}$	This feature is meant to capture if the current price deviates from the previous 20 days' price levels. See Bollinger Band [SOURCE]. The second parameter make any difference for the random forest model
CBBL2	$\frac{Close(t)}{BollingerBand(5,2)}$	See above description
BBW1	$\frac{BollingerBandUpper(20,2)}{BollingerBandLower(20,2)}$	Bollinger band width is meant to capture how volatile the price has been during the previous 20 days
BBW2	$\frac{BollingerBandUpper(5,2)}{BollingerBandLower(5,2)}$	See above description
CSMA1	$\frac{Close(t)}{MA(200)}$	Current price is compared to the moving average of the previous 200 days.
CSMA2	$\frac{Close(t)}{MA(3)}$	See above description
HL	$\frac{High(t)}{Low(t)}$	Highest price of the day divided by lowest price of the day
CL	$\frac{Close(t)}{Low(t)}$	Close price divided by lowest price of the day
HC	$\frac{High(t)}{Close(t)}$	Highest price of the day divided by close price

Ticker symbols for the stocks used in experiment 3 - Appendix B

ABB, ALFA, ALIV, ASSA, ATCO, AZN, BOL, ELUX, ERIC, GETI, HM, INVE, KINV, LUPE, NDA, SAND, SCA, SEB, SECU, SKF, SSAB, SWED, SWMA, TEL2, TELIA and VOLV.