Seeking Alpha

An Analysis on Stock Market Forecasting Techniques

Tim Conze

2024-06-12

Table of Contents

rmarkdown::render("Seeking-Alpha_Capstone-Project.Rmd", envir = my_env)

1 Introduction

Since the establishment of the first stock exchange, the Amsterdam stock exchange, in 1602, people have sought to predict value and develop strategies to identify unseen opportunities.
Since then, the world economy has grown exponentially with the advent of the industrial era, followed by the information age and increasing globalization. This has naturally led to the growth of financial markets and their global importance. Today more than ever, we see financial markets playing a key role not only in the global economy, but also in the geopolitical and socioeconomic landscape.
As such, the financial markets do not just impact investors and market participants, nor do they exclusively affect the general public passively through their implied influence on global politics. As we've seen in the past, the volatile and delicate nature of these markets can lead to significant direct and sudden impacts on uninvolved bystanders when paired with market crazes. Moreover, the performance of a country's financial market can be a crucial determinant in their overall economic health. Since a country's economic situation hugely impacts various sectors, the trickle-down effect can be felt in almost all aspects of life.

In order to grasp why financial markets have an impact on a country's economy and serve as an indicator of its economic health, it's beneficial to examine the structure of the country's economy and the fundamental mechanisms at play in the financial markets.
While financial markets encompass various types of markets, including stock markets, derivatives markets, bond markets, and commodity markets in which this interconnectivity can be observed; the stock market stands out as a key driving force and best exemplifies the aforementioned characteristics.
A country's economy consists of various sectors such as finance, agriculture, and metal, among others. These sectors collectively represent the overall economic power of a country. The economic situation of a country directly or indirectly impacts these sectors, and vice versa. They are closely interconnected, with each one influencing the other.
The growth of these sectors depends on their volatility, which follows the fundamental economic principle of supply and demand. Demand for a specific sector directly affects the stock market, with increased supply prompting traders and financial institutions to invest in that sector or stock, leading to higher prices. Furthermore, regular dividend payments contribute to the generation of profits and returns on invested capital.
Each sector is a part of the country's overall economy, and as such, the sector's growth is reflected in the health of the economy. Conversely, a country's economic situation also impacts supply and demand, as stable financial times lead to the increase of both. A better financial landscape results in the founding and success of more companies, as well as a greater inclination for individuals to invest.
The stock market provides a platform for investors to buy and own a part of a company. As companies expand, they often need more funds to support their future plans. With the approval of existing shareholders, who see a reduction in their ownership due to the creation of new shares, companies can sell these shares to investors to raise money. Successful outcomes lead to an increase in the stock market value for the shares.

In this context, there are different strategies that investors can use, and various types of specialized traders. Fundamentally, there are long-term investment strategies and short-term investment strategies. Investors using a long-term strategy hold on to shares for an extended period, while short-term investments involve buying and selling shares in a shorter timeframe; aiming for profits within days or weeks. Some specialized types of traders include scalpers, swing traders, day traders, and position traders. It is crucial for any investor, regardless of their strategy, to find the most opportune moments to buy and sell shares in order to achieve their desired returns. This is a very challenging task, as the above explanation is an immense oversimplification. In practice, numerous factors are involved in determining a security's price, and investors have to consider various aspects.

Given the difficulty of predicting the stock market, investors are constantly seeking tools and methods to assist them. Since the stock markets present significant opportunity for financial gain and are notoriously challenging to predict, they are widely popular among both investors and data scientists. While investors are primarily drawn to potential monetary gain, data scientists are primarily interested in the analytical challenges. This has led to the development and application of various different methods and techniques in

this field.
Throughout the years of their existence, as financial markets evolved, so have the strategies and techniques utilized by market entities. Traditionally, fundamental and technical analysis are popular approaches to understand market trends, but they possess inherent limitations due to the involvement of lagging indicators and prediction inaccuracy.
In the present day, the financial markets are being redefined by advanced algorithms that make use of machine learning and deep learning models. Although traditional time series forecasting methods, such as autoregressive algorithms continue to be used successfully, the continuous improvement of these new algorithms is pushing the boundaries of what is possible.
That being said, the use of these systems is not universal as many investors continue to rely on more traditional strategies, such as chart analysis and news trading, thus proving that such strategies are still viable and profitable. While it is possible for individuals who do not possess extensive knowledge of the financial markets or utilize advanced strategies to be profitable, for instance, either by sheer luck or by investing in low volatility securities during stable economic times; the true objective of any investor is risk adjusted returns. In financial circles, this concept is referred to as “Alpha”.

Throughout this project, several popular techniques that can be employed to forecast the stock market will be analyzed and evaluated. Methods will be described and applied in order to identify if they can be successfully utilized to seek alpha.

2 Abstract

2.1 Prior Work and Research

The field of stock market research is a popular field of study. Investors are on a continuous search for methods that help them better quantify hidden information, tools that simplify the search for good investment opportunities, or systems that are simply better at forecasting a security’s value. New and improved systems and methods can be utilized by investors to automate their processes with sufficient prediction accuracy or serve as another reference point to make better informed investment decisions. While a significant part of the present research in the field is undoubtedly motivated by investors and their interest in monetary gain, much of the current research is also motivated by data scientists that are invested in this field for its notorious unpredictability, and challenging nature. Paired with the fact that the stock market enjoys a wide appeal in the general public for the same reasons, it is not surprising that a plethora of research papers, articles, and open source resources on this matter are published and made openly available.

Existing research into the field encompasses all areas and disciplines of stock market investing. The stock market is a broad field that includes many niches. As such, scientific papers conducted on a wide range of diverse focal points can be found. This variety encompasses the many specific strategies scientific papers choose to discuss, the various different exchanges that analyses are conducted on, and all types of investing; including, but not limited to, the general buying and selling of stocks. While the specific topics of analyses can differ greatly, most commonly, scientific research is aimed at predicting future stock prices of predominantly American securities that can be found in the S&P 500 or the NASDAQ stock exchange. Less variety can be found in the chosen forecast horizon, as the majority of papers aim to predict the next day price. This often leads to a common pitfall in stock market forecasting, discussed later. Other popular forecast periods include predicting two weeks into the future, which is useful for swing trading, and periods used for long term investment strategies. Most commonly, predictions are acquired recursively, while predictions for either a single end date value, or all individual values within the forecast horizon are obtained. For the target value, a wide majority of scientific papers choose the daily closing price of a stock. While a limited feature selection including the volume, the high price, the low price, and the open of a security is often utilized, more sophisticated papers include various different technical, fundamental and sentiment indicators, either in combination or in a vacuum. While occurring the least, sentiment indicators generally provide the most predictive power. However, the validity of each indicator highly depends on the particular method that is utilized, and thus features can have varying degrees of success. When it comes to the evaluation of results, the wide majority of papers make use of the MSE, RMSE, and MAE error metrics. Scientific papers utilizing error metrics that relate more specifically to the stock market, for instance the Sharpe ratio, are a rare occurrence. Similarly, papers contextualizing their results, simulating practical environments and evaluating results based on practical applications are an exception.

This variety pertaining the focal point of analyses extends to the types of forecasting methods utilized. Over the years, researchers have proposed several solutions to the task of stock market prediction. Some of the most popular approaches include machine learning algorithms, deep learning algorithms, and algorithms specialized on time series forecasting. However, traditional approaches used in prediction tasks as well as novel approaches can still be seen applied. In addition to this, various papers propose ensemble algorithms, in an attempt to combine the strengths of multiple individual algorithms. If those systems are carefully deliberated and applied successfully, they usually show the most promising results.

While traditional methods have not lost their relevance and can still be seen applied with success, deep learning algorithms have surged in popularity in recent years for their superior accuracy and capability to deal with large quantities of data. Generally speaking, improvements of deep learning algorithms and advancements in computational power have let to clear improvements in prediction accuracy. Due to that, state of the art deep learning algorithms and ensembles including various deep learning algorithms are the most discussed methods for stock market forecasting in the field today. It is yet to be seen if transformer models can further improve prediction accuracy in stock market forecasting. While transformers have successfully been applied in many fields, scientific papers applying these systems on stock market forecasting tasks are sparse. However, most papers which apply transformers to predict stock seem to suggest good results. The same can be said for the closely related theory of "attention", first seen applied in the paper "Attention is everything". It showed particular notoriety for its impact on natural language processors and inspired the transformer models that followed.

While prior research is comprised of a lot of quantity, quality on the other hand can be hard to come by. Research papers discussing stock market prediction, for instance, are oftentimes conducted by data scientists who do not possess intimate knowledge of the financial markets. This can lead to the loss of nuance, important aspects and terminology being misrepresented, crucial financial mechanisms being ignored, the misuse of features which in financial forecasting are oftentimes technical indicators, oversimplifications, the reliance on traditional evaluation metrics and difficulties in conceptualizing results, among others. This is often manifested in the use of similar features, that do not add combined value, the utilization of suboptimal scaling methods, an uncertainty in how many previous values should be considered, and model applications that show little practical value. Additionally, as is the case in many other scientific fields, results are often glorified and misrepresented. This can be observed more commonly in a popular and challenging field such as stock market forecasting. It can be thus difficult to trust results, as such papers muddy the waters. This is only intensified by the fact that most research papers do not include complete or even partial code of their models, nor do they include the utilized hyper-parameters. This is especially the case for papers that allege good results. Similar fallacies can be observed in a wide range of available articles describing various different techniques and models in the field. Most articles do not give detailed explanations but instead resolve to surface-level descriptions. Articles that supply code, and attempt to showcase how certain methods can be utilized for stock market forecasting in practice, usually do not explain the supplied source code in sufficient detail, while oftentimes including mistakes and false information. Articles describing the theory of certain methods can be similarly depthless. In addition, it is common for such articles to contradict one another while the used terminology is highly inconsistent.

These problems become only more prominent when asserting the fact, that credible and detailed scientific resources on theories and methods are generally close to impossible to find. Informative literature on these topics must often be purchased or otherwise be pieced together, as easily accessible summations of information do not exist. Hence, it can be difficult to gain intricate knowledge of these topics, that goes beyond a fundamental understanding of the subject. Conversely, to confidently utilize certain methods, apply them to specific problems and aim to add value that goes beyond plagiarism, it is essential to be intimately aware of the subject matter.

While good resources can be found, it can be a challenging task due to the reasons mentioned above. Finding resources often involves relying on sources that apply the pertinent methods in other fields. They must thus be adapted to the specific use case. This tailoring is not always straight forward, as different fields can have different problem statements, priorities and characteristics. A similar challenge concerns the utilized programming language, as the wide majority of deep learning algorithms are written and applied in Python. While general processes can be easily understood and applied in R, not all tools are available in both languages.

Lastly, it is important to note that successful models and algorithms achieving high prediction accuracy do not get published. Although the general concepts of a successful model might get published, the exact tunes and specifications never are. While this is generally the case for most fields, it is especially prevalent in the field of stock market forecasting due to the highly competitive nature of financial markets, and since a successful model is synonymous with monetary gain. An individual able to solve the mentioned problem with any success would be advised to be careful with their findings. Moreover, while it is not impossible for an individual to create a sufficiently accurate model, the most powerful models are built by collectives and held by financial institutions who possess the necessary computing power to make them viable, and naturally, do not make them public.

2.2 Hurdles and Pitfalls

Throughout this sub-section, some of the most noteworthy pitfalls and hurdles will be described. It is essential to be aware of them so that pitfalls can be avoided and hurdles can be overcome.

As is the case for all data science projects, it is important to be intricately familiar with the data that is used for analysis. This is especially the case for projects concerning the stock market, as values can be stored in various different currencies. Wrongful assumptions can thus lead to incorrect comparisons between stocks and skewed results when multiple untransformed currencies are used in the same analysis.

Another false assumption that often occurs, is that all stocks behave the same. A model showing high prediction accuracy when applied to one security should therefore also have high predictive power when applied to other securities. In practice, however, there are many different types of securities with vastly different characteristics. High priced stocks of successful companies for instance, are usually less volatile than those of companies with less market capitalization. Those stocks show less movement while the price of the stock is less prone to rapid changes. Counterintuitively, these stock prices are more challenging to predict for some methods.

An often occurring pitfall in stock market forecasting is the prediction of the previous day value for the unknown next day value. This fallacy occurs in various publications, wrongfully assuming a high prediction accuracy. While results appear highly accurate when graphically represented and when their error is calculated on one of several error metrics, their true nature is revealed when the results are more closely observed. In truth, forecasting a point to be the exact value of its preceding point, knowingly or unknowingly, naturally deems the model completely unviable. In cases in which only the next day is predicted, there is no value in predicting an already known value since it is known with certainty that the price of the security will

change. In the more dangerous case that this fallacy occurs when multiple values are predicted into the future, predictions are made recursively. The model utilizes information it should not have, namely future values that should be exclusive to the test set on which the results are evaluated, to predict each future day value. Even if no future values pertaining to a particular point are utilized in forecasting that particular point, in a prediction task stating a multiple day forecast horizon, any value in the range of the period that is being forecasted must be excluded, or a fallacious statement of a multiple day forecast is made. While, in the case of a multiple day prediction task, this can be seen as a standalone pitfall, it usually occurs in combination with improperly built models.
The model accurately presumes that the last value preceding the value to be forecasted, has the most predictive power, but erroneously completely disregards additional information. Therefore, the model greatly simplifies the prediction task and is incapable of discerning any real structure in the data. This pitfall is most commonly observed in LSTM networks. For more information on this specific pitfall we refer to the resource in section 10.

Many more pitfalls pertain to LSTM networks as they are complex systems and require a lot of pre-processing, comparatively to other methods. Many pitfalls occur during the pre-processing step. Sequencing the input data is a necessary step for many LSTM network applications. Correctly sequencing the data requires attentiveness, as incorrectly sequenced data can often go unnoticed.

Since stock market analysis requires problem-specific knowledge, the learning curve can be steep for data scientists unfamiliar with the stock market's mechanics and its terminology. It can thus be challenging to properly assert the correct objective of analysis, make decisions on what features should be utilized, what key characteristics are important to be aware of, or how to properly evaluate results.
As the stock market is a complex field with many moving parts this can be a major hurdle for data scientists who do not have pre-existing knowledge or interest in the field. That being said, this hurdle can be fairly easily overcome as many resources on the subject matter are openly accessible.

Good sources for data, on the other hand, are not always easy to find and freely accessible. Good data is essential to be able to obtain good results. Previously the popularity of the stock market, hugely motivated by monetary incentives, was mentioned. Consequently, clean datasets containing useful information are exceedingly valuable. For that reason, many sources offering stock market data require payment. Some free alternatives will be mentioned later.

Another hurdle pertains to the fact, that good resources on the most promising forecasting techniques are difficult to find. The reasons and ramifications of this are described in much detail in the previous subsection. The majority of deep learning algorithms were first applied and made available in Python backends. Due to this, most primary and secondary resources providing detailed information on their utilization are also written in Python, more specifically PyTorch. For some deep learning algorithms, applications and resources are exclusively found written in Python. It can hence be difficult for a data scientists, not familiar in Python, to apply these models. In addition to this, applications in other coding languages are often limited and do not include all options and specifications available.

Many other possible hurdles relate to deep learning algorithms specifically. For instance, machine learning algorithms in general are non-deterministic. Due to the stochastic nature of the algorithms or the evaluation procedure, and since weights in neural networks are randomly initialized, results can have high variance. Furthermore, deep learning algorithms are computationally expensive, turning repeated model runs into a tedious process. Therefore, it can be challenging to evaluate the results properly. However, this can be somewhat counteracted by setting a seed.

2.3 Mission Statement

The objective of this analysis is to describe and compare several different forecasting methods in an attempt

to build a useful and viable model to predict future stock prices. Throughout this process, this paper will simultaneously aim to answer several fundamental questions.
A useful and viable model will be defined as either achieving sufficient prediction accuracy or otherwise showing value in forecasting. As such, the viability and usefulness of models will be evaluated on either their prediction performance measured on the RMSE, or in their ability to add overall value in forecasting trends and market movements. Additionally, computational efficiency will be considered in the evaluation, as available computational power might be limited in practice, and decreases the time it takes to obtain predictions; which can be highly beneficial in stock market prediction tasks.

The focus of this analysis is to answer a few fundamental questions while attempting to build a useful and viable model to predict stocks. The viability and usefulness of said model can either be evaluated on its performance in predicting stocks measured on the RMSE, or in its ability to add value in its trend and market movement forecasting ability. A model can be deemed viable and useful if either sufficient prediction accuracy was achieved or if it otherwise shows forecasting value.
The overarching question this paper aims to answer is, if it is possible for an individual to build a model that is sufficiently capable of predicting a stock's price or is able to add value in stock market investment decisions, given a reasonable amount of time and effort invested. Additionally, this paper will explore how accurate such a model can be, and if deep learning algorithms can simplify the process of building a good model.
Throughout this process, more specific questions will be answered. Do other new techniques exist to make stock forecasting easier for individuals? Various papers can be found utilizing sentiment indicators with success, while "news trading" is a popular trading strategy. Can models refraining from utilizing sentiment indicators be viable and accurate? While most financial institutions and many investors utilize some form of stock market prediction algorithm, rarely are they utilized in isolation. Can models provide sufficient value to be used as lone prediction tool?

An often discussed topic in the field of investing is the efficient-market theory. This hypothesis states that share prices reflect all available information, and that generating consistent alpha is impossible. According to the efficient market hypothesis, stocks always trade at their fair value, making it impossible for investors to purchase undervalued stocks, or sell stocks for inflated prices. Therefore, even if stocks are selected expertly and the ideal market timing is chosen, it should be impossible to outperform the overall market. The only way for an investor to obtain higher returns is by making riskier investments.
Proponents of this hypothesis state that the financial markets become continuously more efficient and are close to reaching their peak efficiency. Information on the financial markets is more available than ever before, and computational power is increasing exponentially while advanced machine learning and deep learning algorithms improve at a rapid rate. In particular, institutional investors benefit from those factors the most, as they have vast resources available to them. Consequently, it is impossible for retail investors to find profitable investment opportunities as they are put at a major disadvantage.
Although, the efficient-market theory is a cornerstone of modern financial theory, it is highly controversial and often disputed. While academics point to a large body of evidence in support of the hypothesis, equal amount of dissension also exists. Examples of investors consistently beating the market over long periods can be found, which by definition, should be impossible according to the efficient market theory.
Since a lot of academic research including empirical and theoretical research was conducted on this topic, and no clear consensus could yet be reached, this paper does not presume to find irrefutable evidence for or against the theory. Regardless, it will be put into question and discussed.

3 Methodology

In this section we will outline the general approaches considered to tackle the problem of stock market prediction, and describe the approaches and techniques that were ultimately used throughout this paper in more detail.

Since the stock market is a very popular research topic, the literature includes various techniques from other fields that were adapted to fit this specific use case. For that reason, this section will not be an exhaustive representation of all approaches researched and adapted in this field but instead, will focus on the most promising and widely used techniques.
This overview will start from the ground up, describing the format and properties of the data we will be working with. Afterwards, this paper will outline the most popular and widely used techniques in the field, of which the ones chosen for further examination will be described later in our analysis. Additionally, possible features will be assessed, given that they are an important facet in any data science project and are an essential avenue to consider in the attempt to improve model performance. Due to the fact, that most features in stock market prediction require some knowledge of the underlying field and their interconnectivity with the stock price we are trying to predict, their role and value in the wider picture will be explained with the goal of understanding the opportunities they could present.
This is especially vital, since most of the features considered in this paper are classified as "Technical Indicators", derivative values that do not have a clear intuitive meaning. In those cases their respective formulas will be given and put into perspective.
Lastly, this section will aim to highlight error metrics used to evaluate the results of our forecasts and model performance.

3.1 Stock Market Data

The most crucial part in any analytic endeavor and data science project is to understand the subject matter of the analysis and hence the underlying data.

As previously stated, the task of predicting the stock market might not necessarily focus on the close price of a given stock or any price for that matter. Some projects might instead find more value in predicting the volatility of any given stock or the entire market, thus focusing on predicting the trading volume with accuracy. Other investors might choose to focus on future dividend yields or another aspect entirely. Aside from all other facets of the financial markets not mentioned here, such as ETF's, bonds, FOREX, the derivative markets among many others, which can be considered their own discipline. There are many different objectives stock market analyses can set out to achieve, each with their own respective value.
The focal point of this analysis will be the natural choice of predicting a stocks price after a given period, and over a given forecast horizon. While this approach might seem straightforward, there are in fact many ways the price of a stock can be represented, depending on the information it aims to convey. Each used for a specific purpose and relaying information important to unique use cases.
An example for price quantifications that are particularly case specific are intraday data. Intraday data describes the price in a short time frame throughout any given trading day, such as in a 15 or 30 minute window, usually averaging the price in the given period or representing its last value. It is utilized by active day-traders or "scalpers" interested in the short term price movement of a stock oftentimes without the aim of holding the stock past the market close.

Excluding examples such as these, the list provided below focuses solely on the most commonly seen methods to evaluate a securities value.

Open Price:

The **Open Price** is the price at which the security is first traded, on a particular trading day. Therefore, it is the security's price at the opening of its stock exchange on the trading day. Investors might choose to submit trades to execute "At the Open". Those trades can be submitted before the trading day begins.

High Price:

The **High Price** can be referred to as the highest closing price of a stock over a 52 week period, adjusted for stock splits, but is more commonly used to describe the highest intraday price of a given trading day.

Low Price:

The **Low Price** represents the inverse of the stock's High and describes the lowest price a security was traded for during a given trading day.

Close Price:

The **Close Price** or closing price describes the last price at which a stock is traded during a regular trading day. It is the standard and most commonly used benchmark to quantify a security's performance and to track it over time. Thus, it is also the aim of most stock market prediction tasks.

Adjusted Close Price:

The **Adjusted Close Price** amends a stock's closing price to reflect that security's value after accounting for any corporate actions. While the closing price represents the raw price, which is just the cash value of the last transacted price before the market closes, the adjusted close factors in stock splits, dividends and right offerings. These adjustments allow investors to obtain an accurate record of a stock's performance.

To understand how accounting for these adjustments can add value to investors, it is helpful to observe how a stock's price behaves during corporate actions. When a company issues a stock split, for example, it increases the number of shares in their company. In turn, this decreases the value of each individual share since it then represents a smaller percentage of the company's total share. Still, the value of the shares an investor held before the stock split and after, remains the same. Highlighting the reasons it might benefit a company to execute such corporate actions is beyond the scope of this section.

The Adjusted Close Price is more complex and accurate than the unaltered close price since it displays the true price and accounts for outside factors. For that reason, it is especially useful when examining historical returns or when working with historical prices to attempt to deduce future trends.
Due to these reasons, the choice was made to focus on the Adjusted Close Price throughout the course of this paper as the main benchmark to represent a stock's current and future value, whilst rendering it the target value of our prediction task.

Format:

Generally speaking, stock market data is most commonly found in one of three formats. Commonly referred to as "OHLC", "OHCV" and "OHLCVA" charts. The letters in these acronyms respectively symbolize:

- Open
- High
- Close
- Volume
- Adjusted Close

They are stored in a data frame from left to right, aligning with the order of their initials in the acronyms. Additionally, each format includes the date pertinent to the trading day on which the above listed values were recorded, in a ymd (year-month-day) format. Depending on the data we are working with, the stock ticker of the security, a short letter representation of the underlying company that is listed at the exchange, might also be included. This is the case when multiple securities are jointly stored in a single dataframe for analysis. Similarly, to the above mentioned variaties in stock price representation, storing methods for stock market data also vary depending on the specific prediction or analysis task at hand. As such, intraday data is found in formats useful for their specific purpose.
Moreover, a data scientist working with stock market data will come across additional market and company information such as fundamental data. Fundamental data, such as company balance sheets are naturally stored in their own respective formats, which can vary on the primary source providing that data.
As is the case with any data, it is not only essential to be acutely aware of the format in which the data is stored but also the data's properties, namely what unit of measurement is being utilized to illustrate the values. This is especially true when working with stock market data. The currency in which values are stored can vary dramatically. Wrongfully assuming the currency in which the data was recorded is a common pitfall that must be avoided.

Throughout the analysis in this paper, data will be found in the OHLCVA format and will then be further wrangled. All values found in the datasets used throughout this analysis will be represented in US Dollar.

Forcast Horizon:

Intraday data will not be considered to achieve the stated objective. Different forecast horizons, on the other hand, will be utilized to compare the capabilities of models to accurately predict into the short-, medium- and long-term future. The overall success and promise of predicting said periods will also be analyzed and compared.

These ranges were chosen to represent each forecast period:

Short Term: 1 trading day
Medium Term: 20 trading days (~ one calender month)
Long Term : 260 trading days (~ one calender year)

3.2 Features

Features commonly utilized in the attempt to predict future stock prices and market movements can be classified into one of three core groups: Fundamental Indicators, Technical Indicators and Sentiment Indicators.

Fundamental Indicators:

Fundamental analysis measures a security's intrinsic value by examining related economic and financial factors. Intrinsic value describes the value of an investment based on the issuing company's financial situation, as well as, the current market and economic conditions.

Fundamental analysis can encompass a wide array of factors that can affect the value of a security, ranging from macroeconomic factors such as the state of the economy as a whole, or that of a specific industry, to

microeconomic factors like the effectiveness of management.
The most commonly utilized fundamental factors can be found on a company's balance sheet, including a company's liquidity, their liabilities and assets, their earnings, their profit margins and so forth.
These factors can be utilized by investors to judge whether or not a security is overvalued or undervalued or deduce possible risks involved.
Investors focusing on fundamental analysis usually attempt to find value in long term investment opportunities, putting weight on a company's overall health and long term potential. These investors are known to hold their position for prolonged periods of time.

Fundamental Indicators considered:

- Total Liabilities
- Total Assets
- Liquidity
- Federal Funds Interest Rate
- Gold Price
- US GDP

Technical Indicators:

Technical Indicators are heuristic or pattern-based signals. They are mathematical calculations based on the price, volume and/or interest of a security and aim to forecast market direction. By analyzing historical data these indicators can be used to predict future price movements.
In contrast to fundamental analysis, that attempts to evaluate a security's intrinsic value based on financial or economical data, technical analysis focuses on patterns of price movement and trading signals to evaluate a stock's strength or weakness.
Technical analysis can be used on any security with historical trading data but is most prevalent in commodity and forex, where traders focus on short-term price movements. While they are most commonly used by active traders, they do still provide value for long term investment strategies.

Technical Indicators considered:

- MA (Moving Average) ; Simple and Exponential Moving Average
- MACD (Moving Average Convergence Divergence)
- MTM (Momentum Index)
- RSI (Relative Strength Index)
- ATR (Average True Range)
- VWAP (Volume-Weighted Average Price)
- ADOSC (A/D oscillator)
- CCI (Commodity Channel Index)
- MFI (Money Flow Indicator)
- ULTOSC (Ultimate oscillator)
- Donchian Channels
- DMI (Directional Movement Index)
- VR (Volatility Ratio)
- OBV (On Balance Volume)
- Price Momentum Index(CR) and CR-MA
- Bollinger Bands

Below, the Technical Indicators ultimately chosen for this analysis will be described in more detail.

Sentiment Indicators:

A sentiment indicator is designed to represent how a group feels about the market or the economy.
Sentiment analysis attempts to gauge how market actors are thinking or feeling, which may help to forecast investor's future behavior. These signals may be used in unison with other signals and can confirm trends, as well as inspire a contrarian strategy.

Instead of solely looking at assets or data points, sentiment analysis analyzes the economy from the perspective of participants involved and aims to extract and summarize consumer and investors behaviors and beliefs.
Data extraction for sentiment analysis mostly focuses on extracting information from the internet, through a process called "scalping". News articles, social media and financial reports being the most rich sources to gauge market sentiment and thus the primary targets of sentiment analysis efforts. These resources are then fed into NLP-algorithms (Natural Language Processing) such as Google's BERT (Bidirectional Encoder Representation from Transformers).

This paper will not include sentiment analysis since the process of extracting and analyzing information needed to obtain viable sentiment indicators can be seen as their own discipline and exceeds the scope of this analysis.
Sentiment analysis does provide value, as research proved its ability to improve model's performance. Thus, it will be considered for future research and mentioned later on in this paper.

Summary and Overview of Included Features

Following is a short introduction to each feature ultimately chosen for closer examination. These summaries are intended to supply a framework and basic knowledge of the features presented in this paper and should not be treated as exhaustive definitions.

These features were ultimately chosen for closer examination and are present in this paper:

Basic features:

- Adjusted Close
- Volume

Fundamental Indicators:

- Federal Funds Interest Rate
- US GDP

Technical Indicators:

- RSI
- VWAP
- MACD
- ATR
- CMF
- MFI
- OBV
- Donchian Channels

Volume:

While the inclusion of all five main features in a OHLCV chart is prevalent in many research papers and projects regarding stock market analysis, it was chosen that including any price in addition to the adjusted close price is redundant. The assumption was made that the adjusted close price already encapsulates all the important information the other prices would provide, and that they therefore would only add unnecessary noise.

Volume on the other hand, was included since it might add information not present in the adjusted close price and describes how active a given security is traded. This can be an important factor, as trading volume represents the volatility of a stock and thus also its implied variance.

In simple terms, volume describes the total number of shares traded during the given period.

Fundamental Indicators:

Federal Funds Interest Rate:

The federal funds interest rate is the interest rate at which depository institutions (banks and credit unions) lend reserve balances to other depository institutions. Reserve balances are amounts held at the US Federal Reserve. Institutions with surplus funds in their account lend those to institutions in need of larger balances. The federal funds rate is an important benchmark in financial markets as it influences a wide range of market interest rates.
This influence manifests in the form of a "trickle down effect", as the interest rate for depository institutions increases to keep a balanced liquidity, they charge higher interests for credits issued to individuals.
This in turn effects the overall stock market activity, among various other financial markets. As interest rates rise, both businesses and consumers cut back on spending and instead, choose to save more of their funds, due to the increased returns. In contrast, businesses and consumers will be incentivized to increase spending as saving money yields less returns.
As the interest rate rises, stock prices decrease, as market activity decreases and businesses generally report lower earnings. This can lead to contrarian strategies attempting to buy assets at a lower cost.

The effect of changing interest rates does not effect every industry similarly, as higher interest rates do not tend to negatively affect investments into the financial market sector. Additionally, investors will choose investments that are considered more risk averse in times of high interest rates and uncertainty. The commodity market will feel less of a negative impact as gold, for example, is considered a less volatile, "safe" investment. The same holds true for bonds and ETF's, as investors are more likely to balance their portfolios in favor of "safer" investments, and are more likely to employ hedging strategies.

This all goes to say that there are a multitude of ways interest rates affect the stock market and therefore they are an important indicator to consider.

US GDP:

A country's GDP , or gross domestic product, has effects on the financial market similar to those of the interest rates. As the average funds available to the population increase, they will be more likely to invest.

Furthermore, changes in the GDP can be a sign of impending inflation or deflation, both usually translating into changes in interest rates, hence the GDP acts as a very early indicator of impeding changes causing the effects mentioned above.

The GDP is also often utilized as reliable benchmark to gauge the state of a country's economy and can thus, in theory, be used to judge the current general market strength.

<ins>Technical Indicators:</ins>

RSI:

The Relative Strength Index is a momentum indicator used to measure the speed and magnitude of a security's recent price changes to evaluate overvalued or undervalued conditions in the price of that security. It is a popular momentum oscillator frequently used by technical analysts since its introduction in 1978.

Calculation:

The RSI is calculated using the average price gains and losses over a certain time period. The default time period is typically going to be 14 time periods (in daily chart → 14 days).

$$RSI = 100 - \left(\frac{100}{1 + RS}\right)$$

$$RS = \frac{\text{Average Gain}}{\text{Average Loss}} \text{ (over that time period)}$$

Average Gain:
- Look back during period (usually 14 days)
- How many days did the stock move up during period
- Add up all the values (total points) and divide by days (14)

Average Loss:
- Same as average gain with the difference that we observe how many days a stock moved down during the period

Evaluation:

The RSI will typically spit out a number between 0 and 100.

- Traditionally every time the RSI value is over 70, the stock is said to be **overbought** → sell
- Traditionally every time the RSI value is under 30, the stock is said to be **underbought** → buy

There are more strategies that can be applied with the help of the RSI and more signals that can be deduced from the RSI's behavior, especially when used in tandem with other indicators, we do not describe here.

VWAP:

The Volume-Weighted Average Price Indicator is a benchmark that represents the average price a security has traded throughout the day, based on both volume and price. Its purpose is to provide a smoothed out indication of price, and an idea of whether or not a security is currently undervalued or overvalued. VWAP is important because it provides investors insights into both the trend and the value of a security. It is typically used by traders operating in the short term.

Calculation:

VWAP is calculated by multiplying the *cumulative typical price* by volume, then dividing by *cumulative* price.

The typical price gets calculated by taking the sum of the high price, low price of the time period that is being measured and the closing price, and then dividing it by 3.

The *cumulative volume* is the total volume since the trading session opened.

$$\text{VWAP} = \frac{\text{Cumulative Typical Price x Volume}}{\text{Cumulative Volume-Weighted Price}}$$

$$\text{Typical Price} = \frac{\text{High Price} + \text{Low Price} + \text{Closing Price}}{3}$$

Evaluation:

When we look at the indicator on a chart we notice a colored center line, as well as an upper and lower band. The colored center line is the VWAP line. The upper and lower bands act as our standard deviation lines. Specifically, the upper and lower bands are two standard deviations away from the mean, so 95% of the price action takes place between these bands.

The VWAP virtually acts as the average price of the stock for the day.
-When the price of the stock is <u>below the VWAP</u> line it may be considered <u>undervalued</u>. -When the price of the stock is <u>above the VWAP</u> line it may be considered <u>overvalued</u>.

This does not mean one should enter a short or long position.

These terms of overvalued and undervalued are going to apply to large institutional traders. They will use these as reference points before placing their trades. Many institutional traders will utilize the indicator in their trading to ensure their trading does not move the stock price too extremely in either direction.
Since institutional traders use the VWAP as an indicator or benchmark, it is regarded as a highly influential metric for intraday trading.

More sophisticated methods of utilizing the VWAP to find possible trading signals will not be described in this paper.

MACD:

The Moving Average Convergence/ Divergence indicator is utilized by investors to identify price trends, momentum changes, and market entry points for buying and selling. It is a trend following momentum indicator that shows the relationship between two exponential moving averages (EMA's) of a security's price. The MACD is widely used and one of the most well known technical indicators since its introduction in the 1970's.

Calculation:

The MACD is calculated by subtracting the long-term EMA from the short-term EMA. The study specifically uses the exponential moving average (EMA), rather than the simple moving average to place greater weight on recent price movements.

The MACD is best used with daily periods, where traditionally it is calculated from subtracting the 26 period EMA from the 12 period EMA. Furthermore, the 9 period EMA is commonly the default for the Signal Line.

$$\text{MACD Line} = 12 \text{ Period EMA} - 26 \text{ Period EMA}$$

$$\text{Signal Line} = 9 \text{ Period EMA of MACD}$$

$$\text{MACD Histogram} = \text{MACD Line} - \text{Signal Line}$$

Signal Line:
- Basically acts as an average of the MACD Line
- Slower moving than MACD

MACD Histogram:
- Reflects the distance between the MACD-Line and the Signal-Line
- Changes in the histogram can be observed as the MACD- and Signal-Line separate or diverge and when they move closer together or converge

EMA calculation:

$$\text{EMA} = (\text{Closing} * \text{Multiplier}) + \text{Previous EMA} * (\text{1 - Multiplier})$$

$$\text{Multiplier} = \frac{2}{\text{Number of Periods} + 1}$$

Evaluation:

Divergence can be interpreted as strengthening momentum, while Convergence can be interpreted as weakening momentum. Trading signals are usually found when a crossover of the MACD- and Signal-Line can be observed.

As is the case with the other technical indicators described in this paper, we will not detail more sophisticated methods of utilizing the MACD in practice.

ATR:

The Average True Range is a market volatility indicator. It measures market volatility by decomposing the entire range of an asset price for the target period. This indicator's purpose is to show investors the average range of the price swing over a specified period.

The ATR was originally developed for use in commodity markets but has since been applied to all types of securities. It is typically derived from the 14-day simple moving average of a series of true range indicators; although shorter periods can be used to generate more trading signals, while longer periods have a higher probability to generate fewer trading signals.

Calculation:

Formula for ATR calculation if a previous ATR is present:

$$\text{ATR} = \frac{\text{Previous ATR (Number of Periods - 1)} + \text{TR}}{\text{Number of Periods}}$$

$$\text{TR} = \text{True Range}$$

If there is not a previous ATR:

$$\left(\frac{1}{n}\right)\sum_{i}^{n}\mathrm{TR}_i$$

TR_i = Particular True Range; TR for each specific day

n = number of periods

To calculate True Range (TR):

$$\mathrm{TR} = \mathrm{Max}[(H - L), |H - C_p|, |L - C_p|]$$

H = Today's High L = Today's Low C_p = yesterdays closing price Max = Highest value of the three terms

Evaluation:

A stock experiencing a high level of volatility has a higher ATR, and a lower ATR indicates lower volatility for the period evaluated. It was created to accurately measure a security's volatility and does not indicate price direction. It is commonly used as an exit method, regardless of how the entry decision was made. One of those techniques is the popular "Chandelier Exit", not described here.

Being aware of the underlying volatility of a security can help investors to decide what size trade they should use, adjusted for their willingness to accept risk and the volatility of the markets.

CMF:

The Chaikin Money Flow is a volume-weighted average of accumulation and distribution over a specified period. It shares similarities with the MACD indicator as both are momentum oscillator using exponential moving averages in their calculations.

The Chaikin Money Flow stems from the principle that the nearer the closing price is to the high, the more accumulation has taken place. Conversely, the nearer the closing price is to the low, the more distribution has taken place. If the price action consistently closes above the midpoint on increasing volume, the Chaikin Money Flow will be positive. If the inverse on increasing volume is true, the CMF will be negative.

The typical period for the CMF is 21 days.

Calculation:

$$\mathrm{CMF} = \frac{\text{n-day Average of the Money Flow}}{\text{n-day Average of the Money Flow Volume}}$$

$$\text{Money Flow} = \frac{((\text{Close Value} - \text{Low Value}) - (\text{High Value} - \text{Close Value}))}{\text{High Value} - \text{Low Value}} \text{ ; for each period}$$

$$\text{Money Flow Volume} = \text{Money Flow Multiplier} * \text{Volume for the period}$$

Evaluation:

A CMF above the zero line is a sign of strength in the market, while a value below the zero line is a sign of weakness in the market. The CMF can also be utilized to draw trend lines and to analyze crosses. Crosses can be observed when the CMF intersects the zero line and can be an indication of possible trend reversals.

As is the case with the other technical indicators, the proper detailed utilization of the indicator and its full potential, including more sophisticated analysis techniques, will not be described in this paper.

MFI:

The Money Flow Index is a technical oscillator that uses price and volume data to identify overbought or undersold trading signals in an asset. It can also be used to spot divergences, which can indicate trend changes. Many Traders view this indicator as a volume-weighted RSI, since they are closely related; with the main difference being that the MFI includes both price and volume in its calculation.

The MFI moves between 0 and 100.

Calculation:

$$\text{Money Flow Index} = 100 - \left(\frac{100}{1 + \text{Money Flow Ratio}}\right)$$

$$\text{Money Flow Ratio} = \frac{\text{14 Period Positive Money Flow}}{\text{14 Period Positive Money Flow}}$$

$$\text{Raw Money Flow} = \text{Typical Price} * \text{Volume}$$

→ When the price rose from one period to the next Raw Money Flow is positive and is added to Positive Money Flow, if the Raw Money Flow is negative because the price dropped that period, it is added to the Negative Money Flow

$$\text{Typical Price} = \frac{\text{High} + \text{Low} + \text{Close}}{3}$$

Evaluation:

An MFI reading of above 80 is considered overbought while a reading of under 20 is considered undersold. Values of 90 and 10 can also be used as thresholds.

The primary way of using the Money Flow Index is to spot a divergence. A divergence occurs when the indicator is moving in the opposite direction of price. This is a signal of a potential reversal in price trend. For example, a very high Money Flow Index that begins to fall below the reading of 80 while the underlying security continues to gain in value is a price reversal signal to the downside. On the other hand, a very low Money Flow Index reading that climbs above the threshold of 20 while the underlying security continues to sell off is a trend reversal signal to the upside.

There are many different signals investors can look out for when utilizing the MFI in their analysis that are not described here.

OBV:

On-Balance Volume is a momentum indicator that uses volume flow to predict changes in stock price. While it is not one of the most used indicators it never fell into oblivion since its introduction in 1963.

OBV was designed upon the belief that volume is the key force behind the market and that major market moves can be projected based on volume changes. The assumption is that when volume increases sharply without a significant change in the stock's price, the price will eventually jump upward or fall downward. OBV thus aims to infer crowd sentiment to predict future market trends.

Calculation:

$$\text{OBV} = \text{OBV}_{\text{Prev}} + \begin{cases} \text{volume}\,, & \text{if close} > \text{close}_{\text{prev}} \\ 0, & \text{if close} = \text{close}_{\text{prev}} \\ -\text{volume}, & \text{if close} > \text{close}_{\text{prev}} \end{cases}$$

OBV : Current On-Balance Volume

Volume: Latest trading volume amount

Evaluation:

The theory behind OBV is based on the distinction between institutional investors and the individual retail investors. As mutual funds and pension funds begin to buy into an asset that retail investors are selling, volume may increase even as the price remains relatively level. Eventually, volume begins to drive the price upwards, at which point retail investors begin buying while larger investors begin to sell.
Despite being plotted and measured numerically, the actual quantitative value of the OBV is irrelevant since the real number OBV depends on the start value and is thus arbitrary. Instead, the focus lies in the OBV's movement over time.

To summarize, the OBV attempts to analyze volume numbers to track large institutional investors, or "smart money". Divergences between volume and price are treated as synonyms for the relationship between "smart money" and the disparate masses, in the hopes of showcasing opportunities for buying against incorrect prevailing trends.

There are many ways that tracking institutional money can have benefits, as institutional investors may drive up the price of an asset or are able to recognize trends sooner than retail investors due to their resources.

Donchian Channels:

Donchian Channels are a popular analysis tool, particularly used in commodity trading but applied to many markets. These channels are primarily used to determine the relative volatility of a market and the potential for price breakouts. To form it, three lines are generated from moving average calculations that produce a filled-in channel formed by upper and lower bands around a midrange band. The upper band marks the highest price of a security over a period, while the lower band marks the lowest price of a security over that time. Donchian channels are usually used in unison with other common indicators to paint a more complete picture of the market. While this makes them a versatile tool in technical analysis, their effectiveness hinges on the careful consideration of many factors such as what other indicators to match it with.

Calculation:

$$\text{Upper Channel} = \text{Highest High in last n Periods}$$
$$\text{Middle Channel} = \frac{\text{Upper Channel} + \text{Lower Channel}}{2}$$

$$\text{Lower Channel} = \text{Lowest Low in last n Periods}$$

Evaluation:

Donchian channels depict the relationship between current price and set trading ranges over set periods and are therefore similar to Bollinger Bands, as they both aim to build a visual map of price over time. As there are several applications of the Donchian channels, often in combination with other technical indicators, detailed strategies will not be highlighted here. Examples to what extent Donchian channels can be utilized include: Accessing volatility, finding support and resistance levels, identifying breakouts, finding entry and exit points.

3.3 Algorithms, Models and Theoretical Basis

The popularity of stock market analysis lead to the publication of a multitude of research papers, in addition to wide variety of open resources, describing the potential of various different forecasting techniques and methods. Many of which, apply cutting edge machine learning systems and recently developed data science techniques, developed originally for completely different problem statements, to this field.

While the extensive prior research opens up a lot of opportunities, one might find himself lost in front of the lush forest of possibilities. For that reason, the most prevalent techniques will be highlighted in this section, while the ones this paper focuses on, will be described in more detail.

Generally speaking, the most prominent algorithms in the space of stock market forecasting, can be assigned to one of four core categories as shown in the figure below, with some examples for each respective category.

3.3.1 Basic Machine Learning Algorithms

Linear Regression

Linear Regression is a data analysis technique that predicts the value of unknown data by using another related and known data value. It mathematically models the unknown or dependent variable, and the known or independent variable as a linear equation.
In financial market prediction one or more attributes, such as close price, volume, open price, etc., are employed to forecast the stock price.

At its core, regression modeling aims to simulate the relationship between dependent and independent variables. To that extent, the model produces a best-fit line that describes the relationship between the independent factors, also called explanatory or predictor variables, and the dependent variable, often referred to as response variable or predictor variable.

This straight line is represented by this equation:

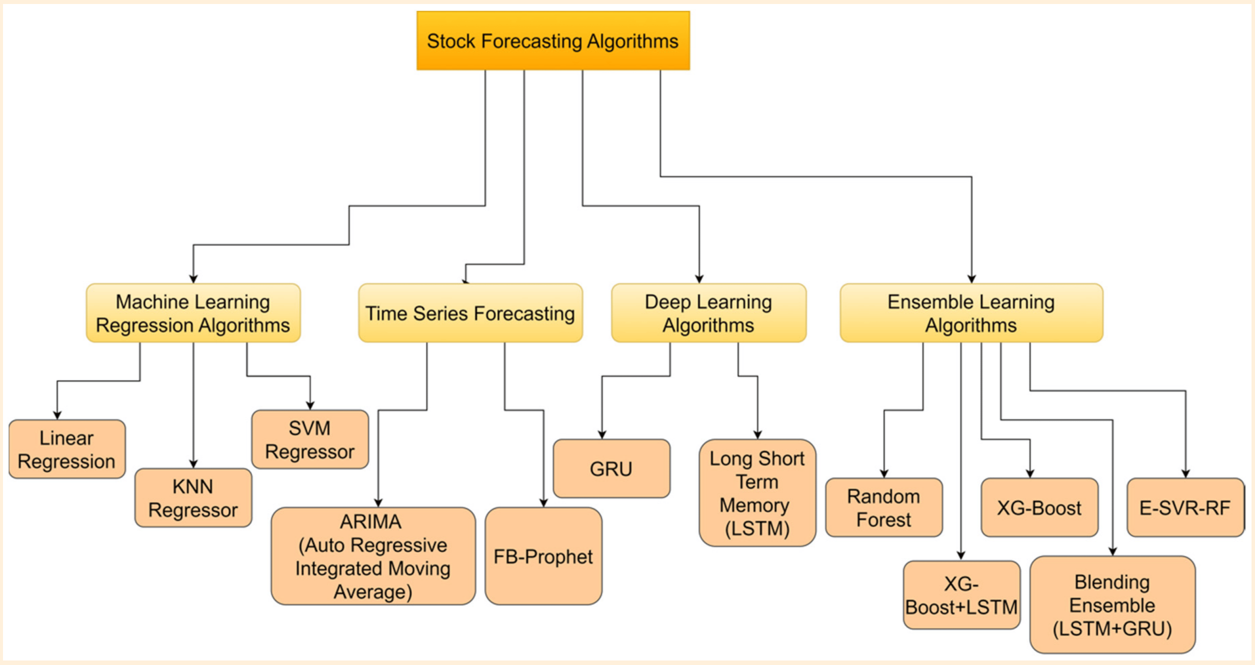

Stock Forecasting Algorithms
Machine Learning Regression Algorithms
Time Series Forecasting
Deep Learning Algorithms
Ensemble Learning Algorithms
Linear Regression
KNN Regressor
SVM Regressor
ARIMA (Auto Regressive Integrated Moving Average)
FB-Prophet
GRU
Long Short Term Memory (LSTM)
Random Forest
XG-Boost+LSTM
XG-Boost
Blending Ensemble (LSTM+GRU)
E-SVR-RF

Figure 1: Overview of Forecasting Algorithms

$$O = S_x + K$$

Where O represents the output, S_x represents the slope and K is a constant.

When charting the dataset's values on a graph, this line is fitted between the points so that the square of the distance or difference between each point and the line is as small as possible. For each given point, the hypothesis line is used to predict the forecast value.
Different types of linear regression include simple linear regression, multiple linear regression and logistic regression.

This technique is relatively simple and easy to interpret and thus often used for preliminary analysis and baseline models. On the other hand, it is less powerful than techniques described later on and offers limited potential on complex analysis tasks.

Logistic Regression
Logistic Regression is a supervised method of machine learning. This technique is used to predict categorical dependent variables using a given set of independent variables. Therefore, the outcome must be categorical or discrete. As mentioned, Logistic Regression is closely related to Linear Regression, the important differentiation being that Linear Regression is used on regression problems, whereas Logistic Regression is used to solve classification problems.

To obtain its outputs, which are given as probabilistic values between 0 and 1, Logistic Regression utilizes the log-odds of an event as a linear combination of one or more independent variables.

When employed in financial market analysis, this method groups several independent factors into one or more mutually exclusive groups and forecasts the likelihood of equities performing well, by utilizing variables for logistic curves. The maximum likelihood is calculated as such:

$$Z_{it} = \beta_1 + \beta_2 EPS_{it} + \beta_2 PB_{it} + \beta_2 ROE_{it} + \beta_2 CR_{it} + \beta_2 DE_{it} + \beta_2 sales_{it} + V_{it}$$

where $z = loglog\left(\frac{Pr}{1-Pr}\right)$ and $Pr =$ probability of outcome being positive.

or:

$y = b_0 + b_1 + b_1x_1...b_nx_n$

where $y = log[\frac{y}{1-y}]$

While being fairly simplistic, one paper in particular showed promising results utilizing this method on intraday data. In addition, variants of this method can be found, such as Binary Logistic Regression, that can improve finance ratios and investor's ability to improve stock price anticipation.

K-Nearest Neighbor (KNN)

While KNN algorithms can be used for classification or regression problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

KNN has been termed as a lazy learner, as it does not need a huge time period for learning. It also offers the advantage of being one of the easiest machine learning algorithms. Namely, it has only a few requirements, the distance metric and the value of k, which defines how many neighbors will be checked to determine the classification of a specific query point. Most commonly, the Euclidean distance is chosen as distance metric.

The calculation for the Euclidean distance is given as:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

where x and y represent the data value and predicted value respectively.

On one hand, the slow learning aspect makes this algorithm quicker than others. On the other hand, this leads to its difficulties in generalizing big data well, as it skips the learning process. It also falls victim to the "curse of dimensionality", as additional features are more likely to increase the amount of classification errors, especially when the sample size is smaller. Due to this, KNN is also more prone to overfitting. While feature selection and dimensionality reduction techniques are leveraged to prevent this from occurring, the value of k can also impact the model's behavior. Lower values of k can overfit the data, whereas higher values of k tend to "smooth out" the prediction values since it is averaging the values over a greater area, or neighborhood. However, if the value of k is too high, then it can underfit the data.

Support Vector Machine (SVM)

Similarly to KNN, SVM is a supervised machine learning algorithm that can be used for both classification and regression, while being better suited for the former. SVM's can be used for a variety of tasks such as image or text classification, spam and face detection, handwriting identification and also for financial market analysis.

A Support Vector Machine categorized inputs using a separator which is discovered when the data is initially mapped to a high-dimensionality feature space. The main objective of the algorithm is to then find the optimal hyperplane in the N-dimensional space that can separate the data points into different classes in the feature space. The hyperplane attempts to keep the margin between the closest points of different classes should be as high as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. Thus, it is difficult to conceptualize when more than 3 features are present.

The performance of SVM can be greatly impacted by its tuning parameters, such as regularization, gamma, and kernel parameters. Advanced variants of SVM are also prevalent, and have shown to improve prediction accuracy further.
We will not describe the various formulas utilized by this algorithm on this occasion since they each require detailed explanations, which is beyond the scope of this paper.

SVM is a popular choice for sentiment analysis and is capable of dealing with both high-dimensional datasets and small scale datasets.

Naive Bayes

Naive Bayes is a supervised machine learning technique build on of the most important principles in statistics, the Bayes Theorem. This algorithm is probabilistic classifier, and thus used for classification tasks such as text classification.

It would be difficult to give an overview of this algorithm without explaining the basics of Bayesian statistics. The Bayes Theorem, also known as Bayes' Rule allow us to 'invert' conditional probabilities. Conditional probabilities represent the probability of an event given some other event has occurred. In other words, it allows us to infer probabilities given the knowledge of the probability of a connected event.

The rule is given by this formula:

$$P(X|Y) = \frac{P(\text{Y and X})}{P(Y)} = \frac{P(Y|X) * P(X)}{P(Y)}$$

Bayes' Theorem is distinguished by its use of sequential events, where additional information later acquired impacts the initial probability.

Naive Bayes classifiers work differently in that they operate under a couple of key assumptions, hence earning the title of "naive". It assumes that predictors in the model are conditionally independent, or unrelated to any other feature present in the model. Additionally it assumes that all features have an equal contribution to the outcome. While this assumption simplifies the classification problem and makes the computations more intuitive, it often does not hold true in real world scenarios, where predictors do usually influence each other and some features offer more predictive value than others. Despite this flawed independence assumption, the algorithm usually performs well, especially on small sample sizes, while also being computationally efficient since it requires only a single probability for each variable.

The formula used for Naive Bayes in stock market forecasting is given as:

$$P(A_i = a_1|B = b_i) = \frac{1}{\sqrt{2\pi\sigma}} * e^{-\frac{(a_i - \mu_{i_j})^2}{2\sigma^4_{ij}}}$$

3.3.2 Time Series Forecasting Algorithms

Predicting stock market data is a time series task and thus many algorithms specifically created to solve those tasks were applied to this field. Before deep learning models became widely adopted, these algorithms were the most utilized, and despite the success of deep learning models they remained influential.
Since time series problems are not unique to stock market analysis, some of these algorithms were intended

for very specific use cases different from the one discussed here. When applying those algorithms, it is important to be aware of the nuances presented by that fact.

ARIMA (Autoregressive-Integrated Moving Average)

To better understand ARIMA, it is helpful to dissect the acronym and to look at each part individually. The "AR" stands for autoregressive which refers to the model that shows a changing variable that regresses on its own or prior lagged values. This means it predicts future values, based on past values.
The "I" stands for integrated, which means it observes the differences between static data values and previous values. This is an integral part of this model, as a successful utilization of this algorithm requires stationarity. To determine if the data is stationary, the Augmented-Dickey-Fuller Test (ADF) is employed; more on this later. If the data is not stationary, it will have to be transformed during pre-processing. The goal is to achieve stationary data, that is not subject to seasonality, so that the statistical properties of the data series, such as mean, variance and autocorrelation, are constant over time.
Lastly, the "MA" refers to the moving average, which is the dependency between an observed value, and a residual error from a moving average model applied to previous observations.

To understand the model theory and building process of an ARIMA model, the concept of stationarity must first be understood. As mentioned, a stationary time series is one whose statistical properties do not depend on the time at which the series is observed. When a time series is stationary it should resemble white noise, meaning it does not matter when it is observed. This can be confusing at times, as time series can be stationary and cyclical, as long as those cycles are not of a fixed length.
In general, time series will have no predictable patterns in the long term. Additionally, time plots will show the series to be roughly horizontal, with constant variance.

Whenever working with ARIMA, it is required that the data is stationary. That is why when working with a non-stationary time series dataset, as is the case for historical stock market data, we first have to make the data stationary. One way of doing this is to compute the differences between consecutive observations. This transformation is known as differencing. Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, thereby removing, or at least reducing, seasonality and trends. In addition to that, a transformation such as logarithms can help to stabilize the the variance of a time series. We can combine both transformations to log difference our data. Applied to stock market price, this means we end up with the returns of any given stock.

Stationarity in data can not only be identified by examining the plot of a time series. ACF and PACF plots are an integral part of ARIMA models and will serve an important role later. They can also be used to identify stationarity as for stationary time series, the ACF will drop to zero relatively quickly while it decreases relatively slowly for non-stationary data, for example.

ACF and PACF plots will be not be discussed in full detail as their analysis involves a lot of detail. We refer to external resources as the analysis of those plots is essential to be able to build ARIMA models.

There are more objective ways to determine whether or not a time series has to be differenced. Unit root tests such as the KPSS test act under the null hypothesis that a time series is stationary. We look for evidence that the null hypothesis is false and thus small p-values suggest that differencing is required.
Occasionally, the data will not appear to be stationary after differencing. In that case it might be necessary to difference the data a second time. This is called second-order differencing. It is almost never necessary to go beyond that.

ARIMA models are a combination of two models, Autoregressive models and Moving Average models. Differencing combined with both of these results in a non-seasonal ARIMA model.
In the autoregression model, the variable of interest is forecasted using a linear combination of past values and the variable. The term *autoregression* indicates that it is a regression of the variable against itself.

An autoregression model of order p can be written as:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \phi_p y_{t-p} + \epsilon_t$$

where ϵ_t is white noise. We refer to this as an AR(p) model, an autoregressive model of order p.

Instead of using past values of a forecast variable in a regression, a moving average model uses past forecast errors. The formula is given as:

$$y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + ... + \theta_q \epsilon_{t-q}$$

where ϵ_t is white noise. We refer to this as an MA(q) model, a moving average model of order q. Since we do not observe the values of ϵ_t it is not a regression model in the usual sense.

When combined the full model can be written as:

$$y_t^{'} = c + \phi_1 y_{t-1}^{'} + ... + \phi_p y_{p-1}^{'} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$$

where $y_t^{'}$ is the differenced time series. This is called an ARIMA(p,d,q) model, where

p = order of the autoregressive part
d = order of differencing
q = order of the moving average part

To determine appropriate values of p and q we make use of the ACF plot and the closely related PACF plots. An ACF plot shows the autocorrelations which measure the relationship between y_t and y_{t-k} for different values of k. PACF plots, on the other hand, show the partial autocorrelations which measure y_t and y_{t-k} after removing the effects of lags up to $k-1$. We cannot use the ACF plot alone since it does not tell us whether or not an observed correlation at a specified point was introduced by new information or correlation present from a previous value. Both plots must therefore be used in unison.

How those plots are utilized to determine p and q will not be explained in this section. Instead, we will touch on the core mechanics and ways to read these plots during the model building process.

Once the model order is identified, the parameters need to be estimated. For that, maximum likelihood estimation (MLE) is used in an ARIMA model. For ARIMA models, the MLE is similar to the least square estimate.

On this occasion it is also important to mention information criteria. They are similar to error metrics in that they are used to determine the quality and validity of each model and estimate prediction accuracy. They are helpful for determining the order of an ARIMA model and are crucial in their evaluation.

Akaike's Information Criteria is given as

$$\text{AIC} = -2log(L) + 2(p+q+k+1)$$

where L is the likelihood of the data, $k = 1$ if $c \neq 0$ and $k = 0$ if $c = 0$.

For ARIMA models, the corrected AIC can be written as

$$\text{AICc} = \text{AIC} + \frac{2(p+q+k+1)(p+q+k+2)}{T-p-q-k-2}$$

and the Bayesian Information Criterion can be written as

$$\text{BIC} = \text{AIC} + [log(T) - 2](p+q+k+1)$$

Good models can be obtained by minimizing the AIC, AICc or BIC. The most commonly chosen one being the AICc. Throughout later analysis the AICc will be utilized.

For more information on the model theory we refer to more comprehensive resources given in section 10.

FB Prophet

Prophet is an open-source time series forecasting tool developed by Facebook's Core Data Science team. It was originally designed for businesses and intended for internal company use, such as forecasting sales, capacity etc. To that extend, Prophet is particularly well suited for forecasting problems that involve daily or seasonal patterns and holiday effects, while also being accessible to a wide array of users with varying levels of expertise.

One of Prophets key feature is its Additive Decomposition Model. Prophet decomposes a time series into three components: trend, holidays, and seasonality. Since the decomposition is additive the time series is modeled as a sum of these parts. Not only is this intuitive, it also often fits well in practical use cases.

Prophet can be considered a nonlinear regression model of the form,

$$y_t = g(t) + s(t) + h(t) + epsilon_t$$

where $g(t)$ describes a piece-wise linear trend also referred to as "growth trend", $s(t)$ describes various growth patterns and $h(t)$ captures the holiday effect.

Prophet offers many advantages as it is easy to use, interpretable and can handle large datasets efficiently. It also offers flexibility as it can be either linear or logistic. For those reasons it has been applied to a multitude of different fields.

3.3.3 Deep Learning Algorithms

Deep Learning is a subset of machine learning based on artificial neural networks (ANN). Since they are based on artificial neural networks they are also known as deep neural networks (DNN). These neural networks are inspired by the structure and function of the human brain's neurons. Therefore a lot of the core terminology of DNN's can be traced back to neurology. For example, similar to how neurons work in the human brain, deep learning architecture contains a computational unit that allows modeling of nonlinear function called perceptron (also referred to as simply "neuron"). Similarly to how neurons work in the human brain, perceptrons receive a list of inputs signals and transform them into output signals.
The perceptron aims to understand data representation by stacking together many layers, where each layer is responsible for understanding some part of the input.
Each layer of perceptrons is responsible for interpreting a specific pattern within the data. A network of these perceptrons mimics how neurons in the brain form a network, so the architecture is called neural networks (or artificial neural networks). Deep neural networks use many layers of interconnected nodes.

These systems gained have wide ranging popularity in many fields of data science and engineering due to their ability to capture complex patterns, handle a large volume of data, for their feature learning and their adaptability. These qualities also make them an ideal choice for financial market forecasting.

While these systems have been popular since their inceptions, in recent years their popularity starkly increased due to the advances in computing power and availability of large datasets.
Deep learning is also used in a wide variety of tasks, such as image recognition, natural language processing, and speech recognition where they have achieved significant success, while playing an important role in artificial intelligence systems.

Long-Short Term Memory (LSTM)

Long Short Term Memory is a special type of Recurrent Neural Network (RNN), which in turn are a special type of ANN, capable of learning long term dependencies. It originated in the attempt to solve the vanishing gradient problem, a common shortcoming of regular recurrent neural networks, which occurs when the gradients propagated backwards through the layers become very small and thus making it difficult for the network to train the weights effectively. This led to small weights being multiplied over and over through several time steps while the gradients diminish asymptotically to zero.

Since LSTM networks are capable of handling and remembering long sequences of data, this powerful algorithm is able to capture historical patterns and therefore predict future values with high accuracy. This characteristic makes these networks an ideal choice for financial market forecasting tasks.

The general structure of an LSTM network is shown in this figure:

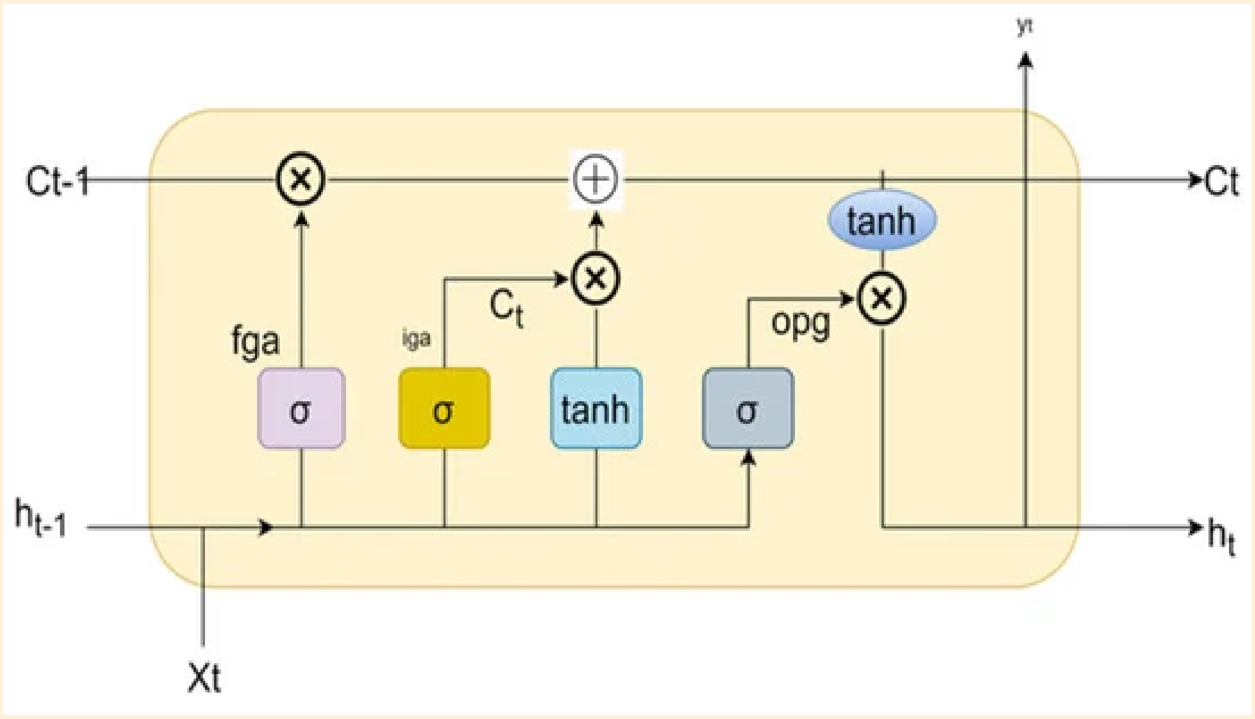


Figure 2: LSTM Cell structure

The key component to understand LSTM networks is the Cell State (C_t). LSTM networks consist of memory blocks, referred to as cells, connected through layers. The Cell State represents the internal short- and long-term memories of a cell. These cells are regulated through mechanisms known as gates employing sigmoid and tanh activation functions.

An LSTM network contains three gates, the input gate, the forget gate and the output gate. These gates can be viewed as filters that decide what information should be let in (remembered) or left out (forgotten).

Forget Gate:

The forget gate, as the name implies, determines which information should be deleted from the current cell state. It applies a sigmoid function to output a value between 0 and 1 for each value from the previous cell state (C_t-1). A returned value of 1 would indicate that all values are being passed through or “remembered”, whereas a returned value of 0 implies all values are being filtered out or “forgotten”.
It is mathematically represented by this equation:

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

Input Gate:

There are two functions in the second gate. The first is the sigmoid function, and the second is the tanh function. The sigmoid function decides what values to let through (0 or 1) and is implemented to reduce the values in the input vector (i_t). The tanh function gives the weightage of the values passed, deciding their level of importance from -1 to 1. Lastly, the cell state is updated by element-by-element matrix multiplication of it and the C_t. The result represents new information that needs to be added to the current cell state.

The sigmoid layer creates an update function as follows:

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

The tanh activation layer is given as follows:

$$\tilde{C_t} = tanh(W_C * [h_{t-1}, x_t] + b_C)$$

The old cell state (C t-1) is updated as follows:

$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C_t}$$

Output Gate:

The third gate is implemented to decide what the final output, flowing into the next cell state, will be. Similar to the input gate, the output gate applies a sigmoid and tanh activation function to filter out unwanted information. Finally, the scaled cell state is multiplied by the filtered output to obtain the hidden cell state ht, which is then passed to the next cell:

$$h_t = o_t * tanh(C_t)$$

For more information on LSTM networks we refer to the additional resources in section 10.

Gated Recurrent Neural Network / Gated Recurrent Unit (GRU)
Comparable to LSTM, GRU is yet another RNN-based model and can be seen as a variation of LSTM since both networks are designed similarly. As seen in LSTM, GRU utilizes gated mechanisms to control the flow of information between current and previous time steps. GRU therefore shares the ability with LSTM to

capture long-term dependencies in sequential data. However, a clear distinction between the two is that GRU networks only utilize two gates, a reset gate and an update gate, whereas LSTM has three gates. This leads Gated Recurrent neural networks to be more computationally efficient and faster to train than LSTM, while also being less powerful. In practice, GRU networks do not perform significantly worse in most tasks, while producing equally excellent results in some cases. Thus, GRU is often preferred, especially for tasks that do not place as much value in slight improvements in prediction accuracy, or for tasks that are less complex.

The structure of a GRU network can be seen here:

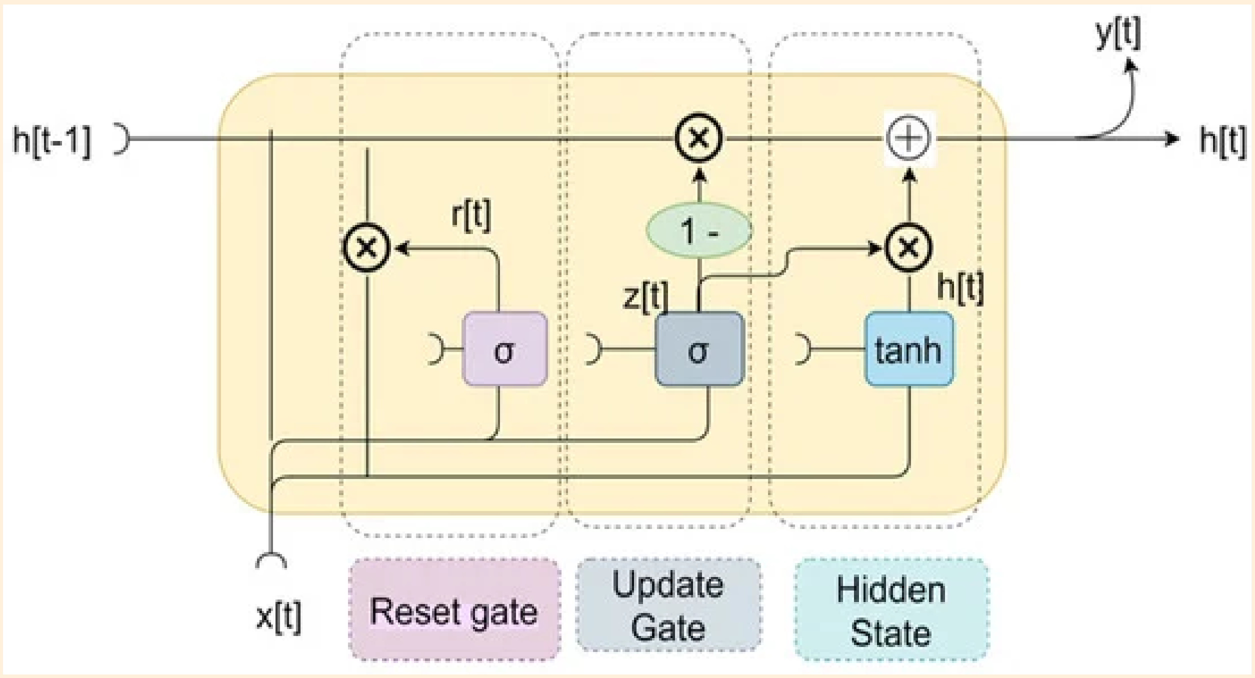


Figure 3: GRU Cell structure

Mathematically the two gates that GRU utilizes can be represented as such:

-Update gate:

$$Z[t] = \sigma(W^z x_t + U^z h_{t-1})$$

-Reset gate:

$$r[t] = \sigma(W^r x_t + U^r h_{t-1})$$

The processes behind these gates is very similar to the ones found in LSTM.

To summarize, GRUs are a simplified version of LSTMs that use a single update gate to control the flow of information into the memory cell. They are easier and faster to run than LSTM, although not as effective at storing data and accessing long-term dependencies. That being said, one cannot be generally preferred over the other, as which type is better depends on the requirements of the specific task at hand.

Convolutional Neural Network (CNN)

Convolutional Neural Networks, also known as ConvNets or CNNs, are a type of neural network that specializes in processing data in a grid like topology, such as an image. It is thus most commonly in computer vision in various image classification tasks.

A typical CNN has three types of layers; a convolutional layer, a pooling layer, and a fully connected layer.

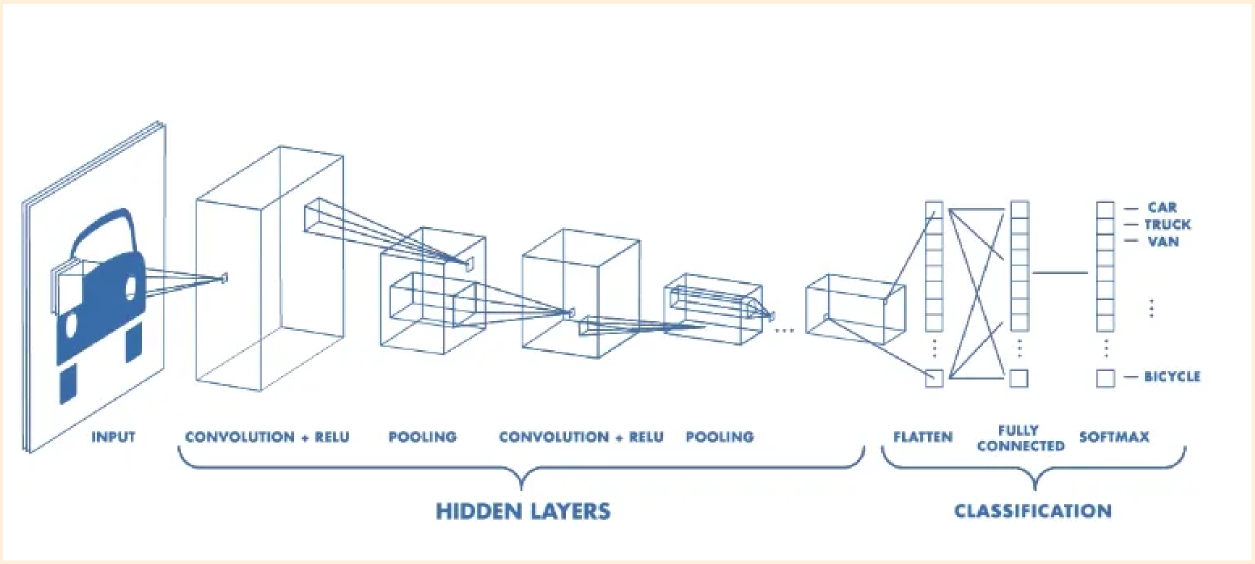


Figure 4: CNN structure

The Convolutional layer applies filters to the input image to extract features or channels, the Pooling layer downsamples the image to reduce computation, and the fully connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

Without detailing the computational process of the model, CNNs may be used to reduce the dimensionality of the feature space and can therefore, also be helpful in fields outside of computer vision.

In stock market analysis, these networks can be utilized to extract the most useful features and can be combined with other models such as recurrent neural networks in various different ways.

Echo State Network (ESN)

Echo State Networks are a type of recurrent neural network specifically engineered to handle sequential data. The key characteristic of these networks is a randomly initialized recurrent layer known as the "reservoir", which allows them to effectively capture temporal input in sequential data.

The structure of an ESN is comprised of three parts; an input layer, a reservoir filled with randomly initialized interconnected neurons, and the output layer. What makes ESNs unique is the reservoir, where weights are fixed and randomly assigned. This creates an 'echo' effect capturing the dynamics of the input signal. Since the weights in the ESNs hidden layer (reservoir) are fixed, they cannot be trained. Strictly the output neurons have weights that are trainable, and can be learned so that the network is able to produce or reproduce particular temporal patterns.

This approach makes echo state networks particularly useful in which capturing temporal dependencies is critical, such as time-series prediction and signal processing. Research suggests that these systems are well suited to deal with the chaotic dynamic of the stock market, as these systems were applied to the field with some success.

3.3.4 Ensemble Learning Methods

Ensemble Learning models work by combining multiple models, often of the same type or different type, to collectively enhance predictive performance. They leverage the collective strength of the ensemble to overcome individual limitations. While each individual part might have low predictive ability, the joint model can make more informed decisions created a better performing model.

Random Forest Algorithm

Random forests, or, random decision forests are supervised learning methods that employ a technique called ensemble learning. They are equally capable of dealing with classification and regression tasks. It derived from the concept of decision trees as it creates several decision trees during its training phase. Each tree is constructed using a random subset of the data set to measure a random subset of features for each position. This randomness introduces variability among trees, which reduces overfitting and improves overall performance. For classification tasks the algorithm aggregates the results of all trees and outputs the class selected by most trees. For regression tasks, the result of all trees is aggregated and averaged. Random Forest models have been a cornerstone in machine learning since their inception in 2001.

A random forest algorithm begins by randomly picking a set number of N data points. Then, a decision tree is build based on the N inputs. Afterwards, the number of trees to be considered is picked. Lastly, the output is predicted based on the steps performed before.

The process of random forest is visually shown here:

A major downside of random forests is, that while they can perform well on large datasets, the increase in tree size slows down the computation substantially. While this characteristic does not make them ideal for the task of stock prediction, which often includes large sets of data, some analyses reported good results on bidirectional models, predicting up or down movements of stocks.

XG-Boost

XG-Boost stands for Extreme Gradient Boosting and is a popular prediction model for stock forecasting. It is a scalable, distributed decision tree, similar to random forest. The difference between the two models is in how the trees are built and combined. In contrast to random forest, XG-Boost uses a number of weak learners and combines them in order to generate a collectively stronger model. This process is referred to as "boosting".

Gradient boosting is an extension of that where the process of additively generating weak models is formalized as a gradient descent algorithm over an objective function. Additionally, targeted outcomes for the next

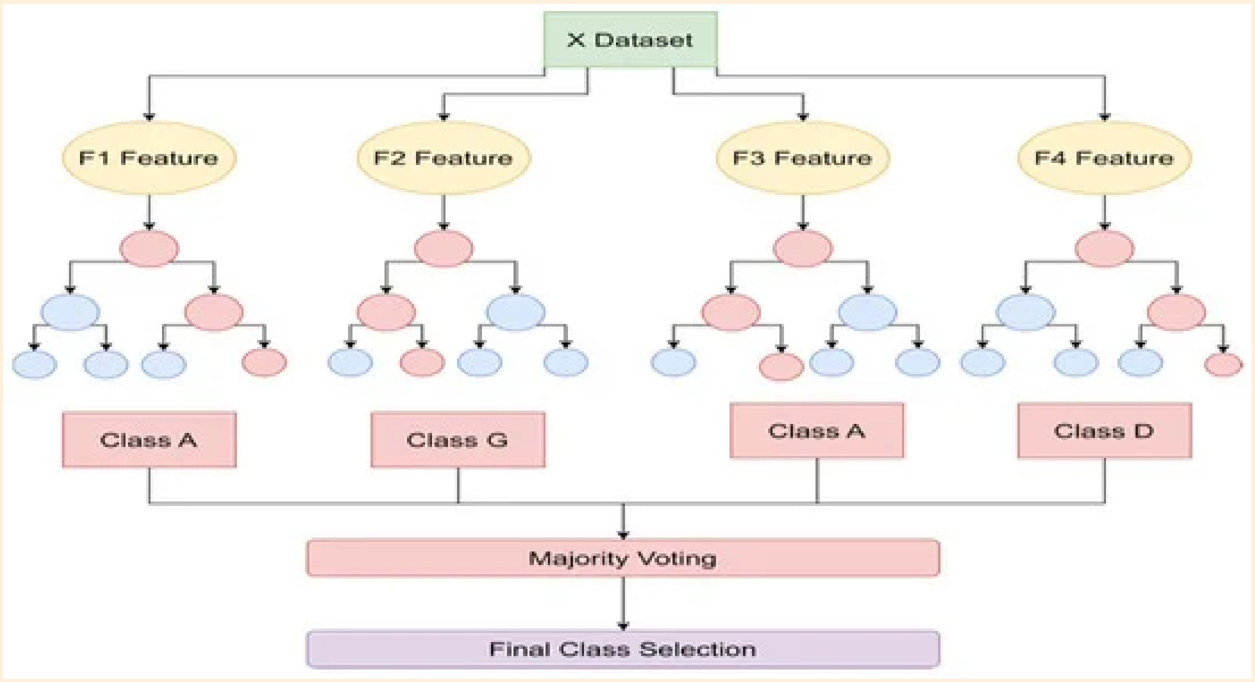


Figure 5: Random Forest Visualization

model are set in an effort to minimize errors, which for each case, are based on the gradient of each error. While decision trees in random forest are built sequentially, here, they are built in parallel. The model follows a level-wise strategy, scanning across gradient values and using these partial sums to evaluate the quality of splits at every possible split in the training set.

The process of XG-Boost is depicted here:

LightGBM

LightGBM is a popular gradient-boosting framework similar to XG-Boost in that it shares many of its advantages. In contrast to XG-Boost and other gradient-boosting frameworks like it, it uses a gradient-based one-sided sampling method to split trees, which reduces memory usage and improves accuracy. It also employs leaf-wise growth instead of level-wise growth, meaning it chooses the leaf it believes will yield the largest decrease in loss. This makes LightGBM faster than traditional depth-wise growth methods. In addition to that, LightGBM replaces the sorted-based decision tree, found in other gradient boosting frameworks, for a highly optimized histogram-based decision tree algorithm, improving memory consumption and efficiency.

Due to these features LightGBM is a valuable machine learning tool and should not be overlooked.

Summary

There are various additional methods present in the space of stock market prediction not described here. Some popular algorithms not highlighted here include Multi-Layer Perceptron(MLP), Subsequent Artificial Neural Network (SANN), ARCH (autoregressive conditional heteroskedasticity), and GARCH (generalized autoregressive conditional heteroskedasticity).

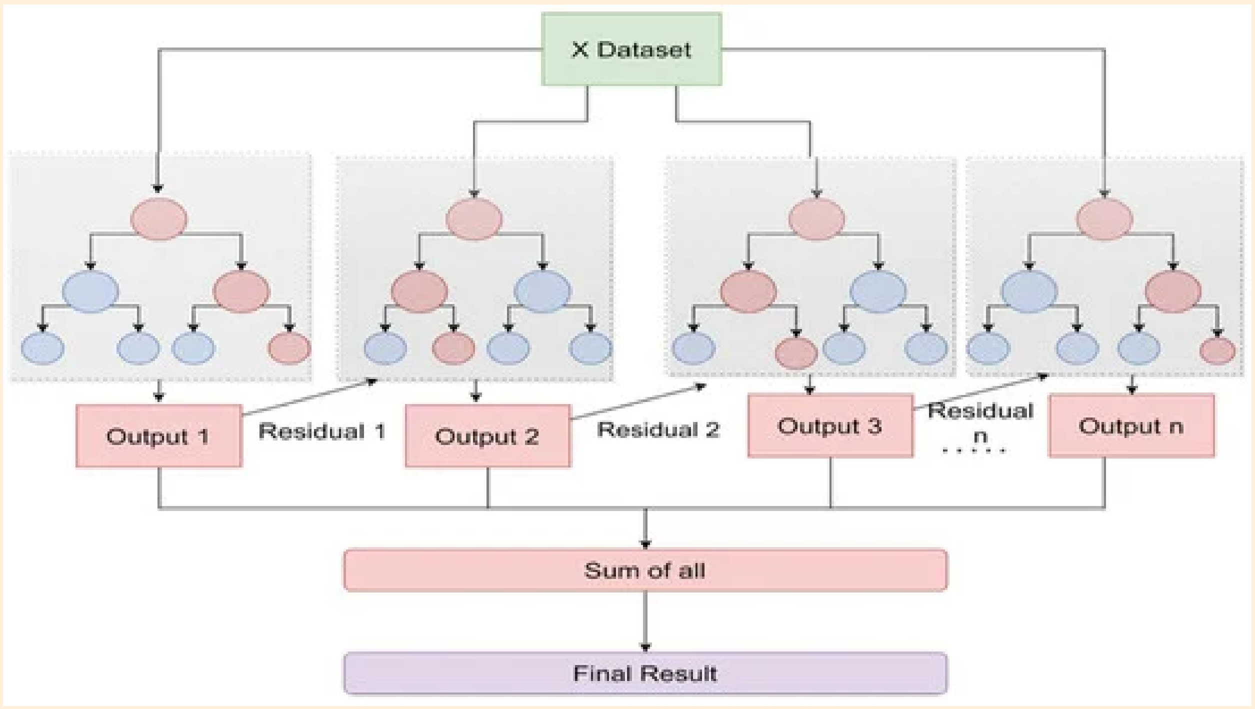


Figure 6: XG-Boost Visualization

Any of the algorithms, mentioned throughout this section, can also be integrated with one another to create ensemble models that potentially combine individual method strengths.

3.4 Error Metrics

To enumerate the performance of any given machine learning algorithm, we have a wide array of error metrics at our disposal. It is indispensable to use such a performance metric when building and testing models, since it makes the quantification of the model's performance reproducible and comparable.

Most commonly, *loss functions* are used to evaluate a model's performance. Loss functions, or "Cost functions", are mathematical functions that calculate the difference, or error, between the model's predictions and the true values. These functions map an event or values of one or more values onto a real number intuitively representing some 'cost' associated with the event. We always strive to minimize a loss function.

The most common loss functions in Machine-Learning are:

- *Mean Absolute Error/L1 Loss (MAE)*
  The *mean absolute error* is measured as the average of the sum of absolute differences between predictions and actual observations. It measures the magnitude of error without considering their direction. *MAE* uses linear programming to compute the gradients and is robust to outliers, since it does not make use of squares.
- *Mean Square Error/Quadratic Loss/ L2 Loss (MSE)*
  The *mean square error* is measured as the average squared difference between predictions and actual

observations. Similarly to the *MAE*, it is only concerned with the average magnitude of error irrespective of their direction. However, due to squaring, predictions that are far away from actual values are penalized heavily in comparison to less deviated predictions. Additionally, *MSE* make its easier to calculate gradients, due to its mathematical properties.

- *Root Mean Square Error (RMSE)*
  The *root mean square error*, or *root mean square deviation*, is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from the measured true values using Euclidean distance. *RMSE* is commonly used in supervised learning applications, as *RMSE* uses and needs true measurements at each predicted data point.

To compute the *RMSE*, the residuals (difference between prediction and true values) are calculated for each data point, the norm of residual is computed for each data point, the mean of residuals is computed, after which the square root of that mean is taken.

That equates to this simple formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - x_0)^2}$$

There are many more loss functions that are of value in different use cases and types of problems, as they each offer their own strength and weaknesses.

Throughout this paper, results will mostly be evaluated based on the RMSE. Although there is value in examining multiple loss functions to evaluate model performance, the deliberate choice was made to focus on a single value for model evaluation to ease comparison between the vast number of models presented in this paper. This is only extrapolated by the fact that not only, will the performance of different methods be analyzed, but additional factors such as multiple different forecasting horizons and features will be considered and compared to one another in separate models. Evaluating multiple loss functions and attempting to visually represent each, could lead to confusion and make clear comparisons more cumbersome.

There will be exceptions to this as the MSE and the MAE will be presented and evaluated on occasion while some methods require the use of additional evaluation metrics, namely the AICc utilized to evaluate ARIMA model performance and build.

4 Preliminary Steps

In this section we will describe the groundwork necessary before beginning the actual building process of models in later sections.
The primary libraries and dependencies utilized throughout the analysis will be described and resources that allow the required data to be freely downloaded will be outlined.
Lastly, the data will be explored to show how stock market data is structured, stored and how stock market data is visually represented.

4.1 Downloading Dependencies and Libraries

In this section all dependencies and libraries needed throughout the analysis will be listed and downloaded. As most of these packages and libraries have specific use-cases, they will be described in more detail once they are utilized.

First, all packages required to conduct the analysis shown throughout this project, are downloaded:

```
#############
################### Packages to Download


if(!require(tidyverse)) install.packages("tidyverse",repos ="http://cran.us.r-project.org")
if(!require(jsonlite)) install.packages("jsonlite",repos ="http://cran.us.r-project.org")

if(!require(caret)) install.packages("caret",repos ="http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate",repos ="http://cran.us.r-project.org")
if(!require(rmarkdown)) install.packages("rmarkdown",repos ="http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr",repos ="http://cran.us.r-project.org")
if(!require(tidyr)) install.packages("tidyr",repos ="http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2",repos ="http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr",repos ="http://cran.us.r-project.org")
if(!require(scales)) install.packages("scales",repos ="http://cran.us.r-project.org")

if(!require(widyr)) install.package("widyr",repos ="http://cran.us.r-project.org")
if(!require(knitr)) install.package("knitr",repos ="http://cran.us.r-project.org")
if(!require(gghighlight)) install.package("gghighlight",repos ="http://cran.us.r-project.org")
if(!require(RColorBrewer)) install.packages("RColorBrewer",repos ="http://cran.us.r-project.org")
if(!require(kableExtra)) install.packages("kableExtra",repos ="http://cran.us.r-project.org")


# Stock API:
if(!require(alphavantager)) install.packages("alphavantager")

if(!require(quantmod)) install.packages("quantmod")
if(!require(tidyquant)) install.packages("tidyquant")


# Prophet:
if(!require(prophet)) install.packages("prophet")

# Forecast and tseries (ARIMA):
if(!require(forecast)) install.packages("forecast")
if(!require(tseries)) install.packages("tseries")

# LSTM:
if(!require(keras)) install.packages("keras")

# To enable parallel computing:
if(!require(doParallel)) install.package("doParallel")
```

The most prominent of the packages listed here is the tidyverse, a collection of core R packages that are designed to work together seamlessly. It is an essential R package, as it simplifies the work flow and includes many useful packages commonly used throughout various analysis tasks. As such, many core features of the tidyverse, packages and corresponding function will be in use throughout this project.

In addition to this, we download the method specific packages needed for the forecasting methods described in later sections and packages that allow us to make use of stock API's, through which the stock data can be downloaded.

The required libraries are downloaded as such:

```r
#############
################### Libraries

library(tidyverse)
library(caret)
library(dplyr)
library(tidyr)
library(scales)
library(widyr)
library(knitr)
library(lubridate)
library(magrittr)
#library(Metrics)
library(jsonlite)
library(reticulate)
library(quantmod)
library(httr)
library(kableExtra)
# library(doParallel) # If wanted
```

While their specific purposes should become clear once they are utilized, following is a short overview of these libraries for reference:

tidyverse – collection of 8 core packages to make data science projects faster and easier caret - package that includes a set of functions and tools that attempt to streamline the process of creating predictive models dplyr – part of the tidyverse; designed for data-frame manipulation tidyr - part of the tidyverse; includes tools to create tidy data, a standard way of storing data scales - Graphical scales map data to aesthetics, and provides methods for automatically determining breaks and labels for axes and legends widyr – for work with wide matrices knitr – general purpose tool for dynamic report creation lubridate – includes functions to work with date-time and time-span data magrittr – part of tidyverse; provides mechanism for chaining commands Metrics- implementation of several evaluation metrics jsonlite – JSON parser and generator reticulate –

provides a comprehensive set of tools for interoperability between Python and R doParallel – allows for parallel computing and thus, faster computation in some cases

4.2 Downloading Data

There are various resources available that offer a multitude of stock market datasets. However, trustworthy institutions that offer clean datasets of high quality either usually require a paid subscription or datasets to be purchased. This is especially the case for recent or up-to-date market data.

Instead of manually downloading stock market data, a more advantageous alternative is the use of a stock market API. They can usually be easily integrated and allow for data to be directly downloaded into the working environment for further analysis. Hence, they are the fastest way to download financial data for analysis and allow for easy reproduce-ability.

This section will describe two options available that can be freely used, and provide market data from credible sources.

The first of which is a stock market API, "Alpha Vantage". This stock API is popular among many investors being a credible resource for accurate and current market data. In addition to traditional stock market data, Alpha Vantage offers data on several financial markets, data on many economic indicators and fundamental data of various publicly traded companies.
While Alpha Vantage is generally free to use, not all of the data is freely accessible. Intraday data, many of the most popular technical indicators and realtime data for instance, is thus locked behind a subscription fee. Additionally, Alpha Vantage bottle caps the amount of downloads per day for non-premium users.
Another caveat of this stock market API is, that to be able to use it, a free API key will have to be downloaded on their website. It is to be noted that the API key shown in the following examples is a place holder and will need to be replaced by a personal key.

A unique personalized key can be obtained via this link:

https://www.alphavantage.co/support/#api-key

First, the Alpha Vantage package will have to be downloaded if it was not already. Afterwards we load the library and submit out personal API key as such:

```r
# Install the API R package and load library:

install.packages("alphavantager")
# Or
# devtools::install_github("business-science/alphavantager")

library(alphavantager)
```

```r
# Set your API key:

av_api_key("YBEX2W45GF6IXNJG")
print(av_api_key())
```

To download the stock market data of a particular stock ticker we can utilize the `av_get()` function.

Throughout this project, Microsoft stock market data will be utilized. It was chosen since it is widely recognized and listed on the NASDAQ stock exchange. It will be the primary example on which the analysis shown throughout this project will be conducted on.

```r
# Download stock tickers:

msft_van <- av_get(symbol     = "MSFT",
                     av_fun     = "TIME_SERIES_DAILY",
                     outputsize = "full")
```

An alternative to Alpha Vantage is quantmod. In contrast to Alpha Vantage, quantmod has more limitations in the types of data that can be downloaded and does not allow for realtime data to be downloaded. It is however, completely free to use. Another benefit of quantmod is that data can be easily downloaded as a tibble with the help of the `tidyquant` package.

Similar to the previous API, we must first download the package and load the required libraries. We utilize the function `getSymbols()` to create a dataframe of the ticker name of the respective stock, which is stored as a class of xsx or zoo. Utilizing the tidyquant function `tq_get()` we can download the stock market data.

```r
getSymbols("MSFT") # creates dataframe with ticker name of class xsx/zoo

## tidyquant

msft <- tq_get("MSFT",get="stock.prices")
```

In addition to the stock price data, quantmod also allows us to download certain financial data pertaining to the underlying company.

```r
tq_get("MSFT", get ="dividends")
```

This is also possible utilizing Alpha Vantage, while a lot more processing is required to store the data in a tibble, so that it can be used for further analysis.
The function `av_get()`, that was previously used to download stock price data, cannot be used in the same way to download financial data through the Alpha Vantage API.

```r
msft_fin <- av_get(symbol     = "MSFT",
                   av_fun     = "BALANCE_SHEET",
                   outputsize = "full")
```

Instead, we utilize the Jsonlite package to obtain Microsofts total assets, total liabilities and their respective fiscal dates.

```r
library(httr)
base_url <- "https://www.alphavantage.co"
myquery <- list("function"="BALANCE_SHEET",
                "symbol"= "MSFT",
                "outputsize" = "full",
                "apikey"="YBEX2W45GF6IXNJG") # Remember to fill in your key

msft_fin <- GET(base_url, path = "query", query = myquery)



# Convert the response content to a list
content_string <- rawToChar(msft_fin$content)
json_list <- fromJSON(content_string)

# Interrogate the list
class(json_list)
names(json_list)
names(json_list[1]) # json is lists of lists in R

# extracting "fiscalDateEnding", "totalAssets" and "total Liabilities"
result <- sapply(json_list[3], `[`)

fiscal_date <- sapply(json_list[3][[1]][[1]], `[`)
total_assets <-  sapply(json_list[3][[1]][[3]], `[`)
total_liabilities <- sapply(json_list[3][[1]][[20]], `[`)


msft_fin_dat <- tibble(fiscal_date=ymd(fiscal_date),
                       total_assets = as.numeric(total_assets),
                       total_liabilities = as.numeric(total_liabilities))
```

A similar process is needed to download technical indicators through the Alpha Vantage API. To download the RSI technical indicator we, again, make use of the Jsonlite package.

```r
# RSI
library(httr)
base_url <- "https://www.alphavantage.co"
myquery <- list("function" = "RSI",
                "symbol"= "MSFT",
                "interval" = "daily",
                "time_period"= "60",
                "outputsize" = "full",
                "series_type"= "close",
                "apikey"="YBEX2W45GF6IXNJG") # Remember to fill in your key

msft_rsi <- GET(base_url, path = "query", query = myquery)


# Convert the response content to a list
content_string <- rawToChar(msft_rsi$content)
json_list <- fromJSON(content_string)

# Interrogate the list
class(json_list)
names(json_list)
names(json_list[1]) # json is lists of lists in R

# extracting RSI
result <- sapply(json_list[2], `[`) %>% head

msft_rsi_av <- sapply(json_list[2][[1]], `[`) %>% as_tibble() %>% unnest(cols=c()) %>%
  pivot_longer(cols = everything(),names_to="date", values_to ="rsi") %>%
  mutate(date = ymd(str_remove(date, "\\.RSI$")),rsi = as.numeric(rsi)) %>%
  rename(ds=date)
```

Quantmod in combination with `tidyquant` offers a far simpler alternative. Rather than downloading each respective technical indicator, they can be calculated individually on previously downloaded stock market data.
To that extent, we utilize the tidyquant function `tq_transmute`, which in turn includes several built in functions, that allow for the calculation of many popular technical indicators. While some require minor adjustments, this greatly simplifies the process.

While this method does not have the bottle cap limitations of the previously shown method, it has the disadvantage of including missing values at the beginning of each dataframe. This is due to the fact, that the technical indicators are calculated on several past values, which are supplied by a dataframe that has a limited range of values.

Ultimately, this method was chosen due to its simplicity and lack of bottle cap limitation. The technical indicators that were chosen for analysis were described in a previous section. They can be downloaded as shown here:

```r
### Basic

# Volume

msft_vol <- tq_get("MSFT",get="stock.prices") %>%
  select(volume,date)




#### Technical Indicators


# RSI (14 day RSI)
msft_rsi <- tq_get("MSFT",get="stock.prices") %>%
  tq_transmute(select=adjusted,n = 14, mutate_fun=RSI) %>%
  replace(is.na(.),0)



# VWAP (10 days)

msft_vwap <- tq_get("MSFT",get="stock.prices") %>%
  mutate(price = (high + low + close) /3 ) %>%
  tq_transmute_xy(x = price, y = volume, mutate_fun=VWAP) %>%
  replace(is.na(.),0)



# MACD
# On adjusted close,
# Long Term EMA = 26
# Short Term EMA = 12
# Signal line = 9 periods

msft_macd <- tq_get("MSFT",get="stock.prices") %>%
  tq_transmute(select = adjusted,maType = "EMA", mutate_fun=MACD) %>%
  select(date,macd) %>%
  replace(is.na(.),0)





# ATR

msft_atr <- tq_get("MSFT",get="stock.prices") %>%
  tq_transmute(select = c(high,low,close),n = 14, mutate_fun=ATR) %>%
  select(date,atr) %>%
  replace(is.na(.),0)
```

```
# CMF (Chaikin Money Flow)

tq_cmf <- function(clv, volume, n = 20){
  runSum(clv * volume, n)/runSum(volume, n)
}

msft_cmf <- tq_get("MSFT",get="stock.prices") %>%
  tq_mutate(select = c(high, low, close), mutate_fun = CLV) %>%
  mutate(cmf = tq_cmf(clv, volume, 20)) %>% select(date,cmf) %>%
  replace(is.na(.),0)




# MFI

msft_mfi <- tq_get("MSFT",get="stock.prices") %>%
  mutate(price = (high + low + close) /3 ) %>%
  tq_transmute_xy(x = price, y = volume, mutate_fun=MFI) %>%
  replace(is.na(.),0)



# OBV

msft_obv <- tq_get("MSFT",get="stock.prices") %>%
  tq_transmute_xy(x = adjusted,y = volume, mutate_fun=OBV)



# Donchian Channels


msft_dc <- tq_get("MSFT",get="stock.prices") %>%
  tq_transmute(select= c(high,low), mutate_fun=DonchianChannel) %>%
  rename(dc_high = high,
         dc_mid = mid,
         dc_low = low) %>% replace(is.na(.),0)
```

Lastly, the federal funds interest rate and the US gross domestic product are downloaded. Alpha Vantage is utilized to download historical interest rate data, while projected future values are obtained with `tidyquant`.

```
### Interest Rate

# Future projected via quantmod/tidyquant
fed_proj_rates <- tq_get("FEDTARMD", get = "economic.data",from = "2023-01-01", to  = "2025-12-31")

# Download via Alpha Vantage
```

```
fred_int_rate_av <- av_get(av_fun     = "FEDERAL_FUNDS_RATE",
                           outputsize = "full") %>% rename(date = timestamp)
```

To find a dataset for the "Real US GDP", multiple sources were checked without success. As the projected GDP could only be found as growth percentage, and Real GDP growth numbers could not be obtained, it was ultimately decided to utilize the potential Real GDP as it includes projections:

```
### GDP

# Download via Alpha Vantage
fred_rea_gdp_av <- av_get(av_fun     = "REAL_GDP",
                          outputsize = "full") %>%
  rename(date = timestamp)


# Tidyquant
fred_rea_pot_gdp_tq <- tq_get("GDPPOT", get = "economic.data",
                              from = "2014-01-01", to  = "2025-12-31")
```

The projections for both the federal funds interest rate and the GDP were downloaded from FRED (Federal Reserve Economic Data), an open resource providing various economic datasets made available by the Federal Reserve Bank of St. Louis.
As such, both datasets can be found on FRED's website to be closer examined:

Federal Funds interest rate: https://fred.stlouisfed.org/series/FEDTARMD

US GDP: https://fred.stlouisfed.org/series/GDPPOT

4.3 Data Exploration

Stock market data throughout this analysis will be stored in an OHLCV format. This acronym refers to the type and order of stored data. From left to right datapoints for the open, high, low, close and volume are stored, in addition to the adjusted close.

```
## First look at dataframe
# Our data will be stored in an OHLCV format

head(msft)
```

```
## # A tibble: 6 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2014-01-02  37.3  37.4  37.1  37.2 30632200     31.2
## 2 MSFT   2014-01-03  37.2  37.2  36.6  36.9 31134800     31.0
## 3 MSFT   2014-01-06  36.8  36.9  36.1  36.1 43603700     30.4
## 4 MSFT   2014-01-07  36.3  36.5  36.2  36.4 35802800     30.6
## 5 MSFT   2014-01-08  36    36.1  35.6  35.8 59971700     30.1
## 6 MSFT   2014-01-09  35.9  35.9  35.4  35.5 36516300     29.9
```

Besides columns storing the price values and the volume, a column specifying the respective trading day can be found. As can be seen in the above sample, historical data available ranges back to January 2^{nd} of 2014. While not always present, the first column denotes the symbol for the trading ticker. It serves as a short form for the publicly listed company who's stock the data refers to. It becomes necessary when working with multiple securities, especially when they are stored in the same dataframe.
All price values and the volume are represented in US Dollar.

On the first look it is immediately noticeable how all values are on a similar range, excluding the volume, which is stored in a significantly higher order of magnitude. It is important to be mindful of that when scaling the values or otherwise processing the data.

As our our data is cleaned up and stored in a tibble, we can begin to conduct some preliminary data exploration to find significant points of interest. Our data being stored in a tibble, greatly simplifies this process.

```
## # A tibble: 1 x 2
##   date       highest_open
##   <date>            <dbl>
## 1 2024-02-12         421.
```

```
## # A tibble: 1 x 2
##   date       highest_adjusted
##   <date>                <dbl>
## 1 2024-02-09             420.
```

```
## # A tibble: 1 x 2
##   date       highest_close
##   <date>             <dbl>
## 1 2024-02-09          421.
```

```
## # A tibble: 1 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2024-02-12  421.  421.  415.  415. 21202900     414.
```

```
## # A tibble: 1 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2024-02-09  415.  421.  415.  421. 22032800     420.
## # A tibble: 1 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2024-02-09  415.  421.  415.  421. 22032800     420.

## # A tibble: 1 x 2
##   date       lowest_open
##   <date>           <dbl>
## 1 2014-01-14        34.7

## # A tibble: 1 x 2
##   date       lowest_adjusted
##   <date>               <dbl>
## 1 2014-01-13            29.4

## # A tibble: 1 x 2
##   date       lowest_close
##   <date>            <dbl>
## 1 2014-01-13         35.0

## # A tibble: 1 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2014-01-14  34.7  35.9  34.6  35.8 41623300     30.1

## # A tibble: 1 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2014-01-13  36.0  36.0  34.8  35.0 45901900     29.4

## # A tibble: 1 x 8
##   symbol date        open  high   low close   volume adjusted
##   <chr>  <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1 MSFT   2014-01-13  36.0  36.0  34.8  35.0 45901900     29.4
```

When we look at the highest and lowest price points found in the data, we first notice that both extremes for the adjusted close and the close are to not much surprise found on the same day. The extremes for the open price, on the other hand, are found a few days after for the highest value and on the next day for the lowest value, so generally all extremes are found around the same time frame.
Most notably, we can observe that the lowest price values can be found close to the earliest available dates for which data is present, while the highest values are close to the most recent dates available. At the same time, we see a stark difference between the two extremes. This implies a general uptrend in the stock price throughout time. Given the difference, a fairly steep continuous uptrend can be assumed.

Below we see the average prices for the security:

```
## # A tibble: 1 x 3
##   avg_open avg_adjusted avg_close
##      <dbl>        <dbl>     <dbl>
## 1     153.         148.      153.
```

The averages further imply a general uptrend in the stock price. Furthermore, a slight exponential growth can be assumed.

To gain a better understanding of the variance found in the data, we look at the day-to-day differences in price.

```
## # A tibble: 1 x 4
##   open_lag_avg open_lag_high open_lag_low    sd
##          <dbl>         <dbl>        <dbl> <dbl>
## 1        0.144          22.6        -17.7  3.12
```

```
## # A tibble: 1 x 4
##   adj_lag_avg adj_lag_high adj_lag_low    sd
##         <dbl>        <dbl>       <dbl> <dbl>
## 1       0.146         19.8       -22.6  3.14
```

```
## # A tibble: 1 x 4
##   close_lag_avg close_lag_high close_lag_low    sd
##           <dbl>          <dbl>         <dbl> <dbl>
## 1         0.143           19.9         -23.4  3.21
```

We observe that the day-to-day price is highly variable, seen in the wide range of price differences and a high standard deviation. The positive, small average value is to be expected as the price increases over a long period.
Furthermore, the up-down split of the stock is fairly balanced while up movements are slightly more common, aligning with our previous findings.

```
## # A tibble: 1 x 3
##      up  down    same
##   <dbl> <dbl>   <dbl>
## 1 0.532 0.461 0.00667
```

When we observe the intraday difference we see that on average, the stock closes lower than it opens. Additionally, we observe high variance similar to the day-to-day difference.

```
## # A tibble: 1 x 2
##   intraday_difference    sd
##                 <dbl> <dbl>
## 1             -0.0586  2.57
```

Market timing is a factor of huge importance to investors. For that reason, there is value in knowing on which days the stock is usually priced the highest, and on which days it is at its lowest price, on average. As most stock exchanges, including the NASDAQ in which we find the Microsoft stock listed, are closed on Saturdays and Sundays, only weekdays will be shown here.

```
## # A tibble: 5 x 5
##   weekday   avg_close avg_open avg_adj   avg_vol
##   <chr>         <dbl>    <dbl>   <dbl>     <dbl>
## 1 Monday         152.     152.    146. 28710784.
## 2 Tuesday        153.     154.    148. 28972278.
## 3 Wednesday      153.     153.    148. 29632900.
## 4 Thursday       153.     153.    148. 29724885.
## 5 Friday         154.     154.    148. 33309818.
```

The price of the security seems to be generally balanced throughout the week. On average we can observe a slight increase in price and volume at the end of the week, while the stock typically begins the week at its lowest value.

Financial markets were strongly affected by the Covid pandemic. As we can see here, this was not the case for Microsoft. The Microsoft stock can be referred to as an active stock, meaning it is highly traded, or in other words has fairly high volume. In addition to this, it is a stock with comparatively low volatility. High volume on fairly tight price movements are common for stocks in the tech sector, especially those with high liquidity, as is the case for Microsoft.

```
## # A tibble: 7 x 7
##   date              open  high   low close    volume adjusted
##   <chr>            <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
## 1 2018              101.  102.  99.9  101. 31590189.     95.0
## 2 First Covid year  130.  131. 129.   130. 24580994.    125.
## 3 2020              193.  195. 190.   193. 37659592.    187.
## 4 2021              276.  278. 273.   276. 26013013.    270.
## 5 2022              269.  272. 265.   269. 31219322.    265.
## 6 2023              314.  317. 311.   314. 27675560.    312.
## 7 2024              397.  400. 394.   397. 24542621.    396.
```

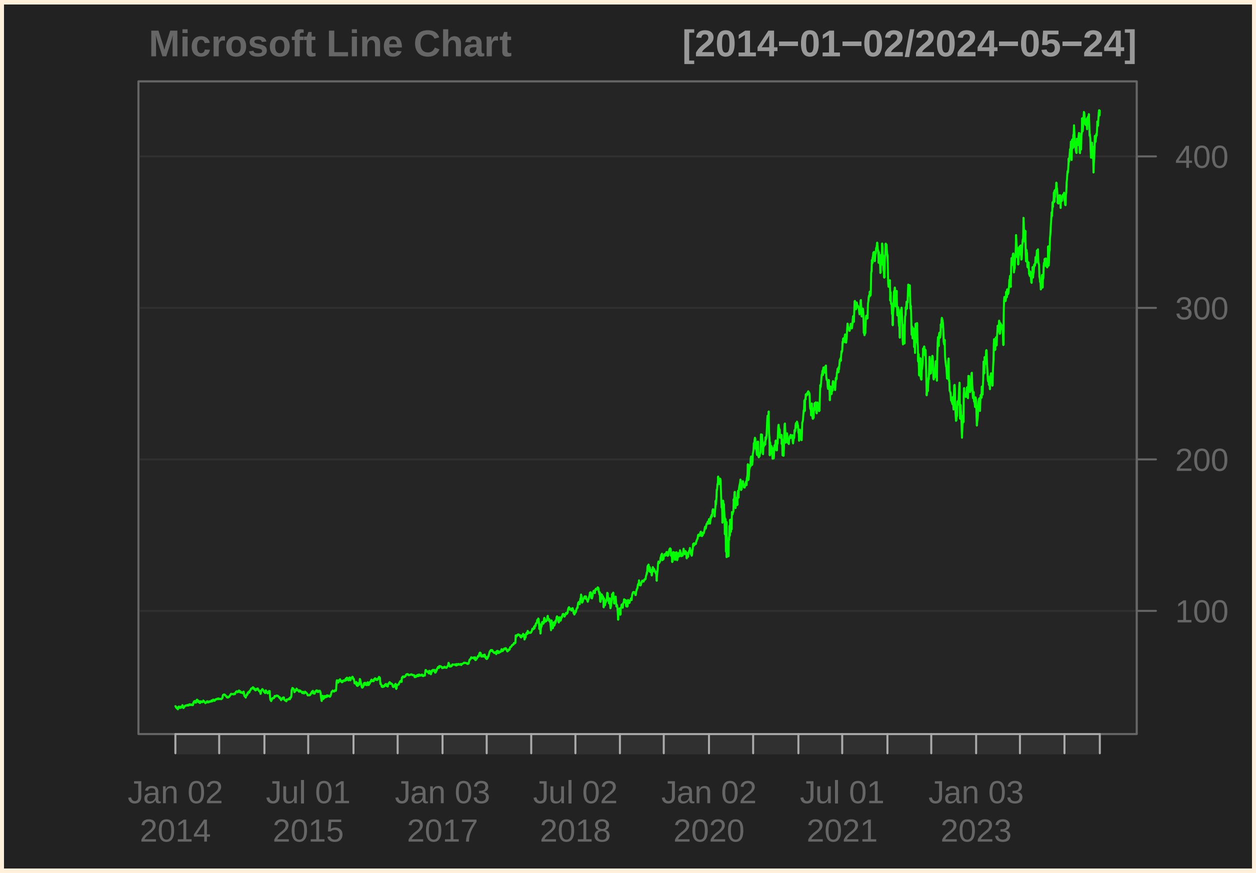
When traders and investors work with stock market data, it is oftentimes visually represented by charts. Subsequently, much of a traders time is spent looking at, and analyzing charts. Unsurprisingly, the general public thus commonly associates the stock market with images of price charts.

There are many ways charts are used for stock market analysis in practice. Active traders overlay their charts with various visual aids and indicators, to find possible buy and sell signals, while oftentimes setting their buy and sell calls directly on the chart.
Humans are visual creatures, and while generally great at pattern recognition, we are much better at memorizing and recognizing patterns when they are visually represented. Much of the structure in data is impossible for most humans to recognize if it is solely represented by numbers.

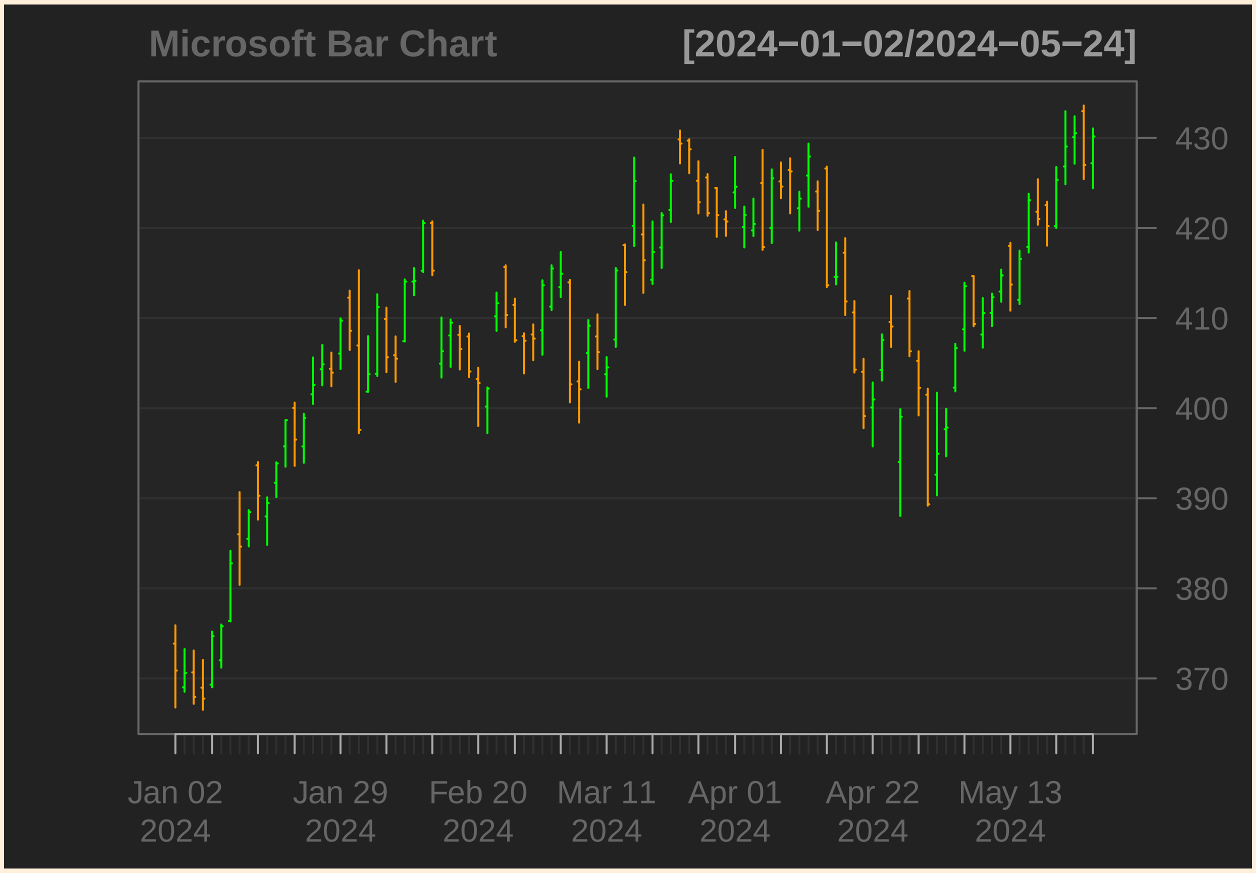
For that reason, various different forms of charts are commonly utilized by traders, investors, and analysts alike.
The most popular forms of charts being the line chart, bar chart and candlestick chart, the latter of which has become highly popular in recent years.

A similarity all these charts share, is the price scale being represented by the y-axis (vertical axis), while the x-axis (horizontal axis) denotes the time scale. Prices are plotted left to right across the x-axis, with the most recent plot being the furthest right.

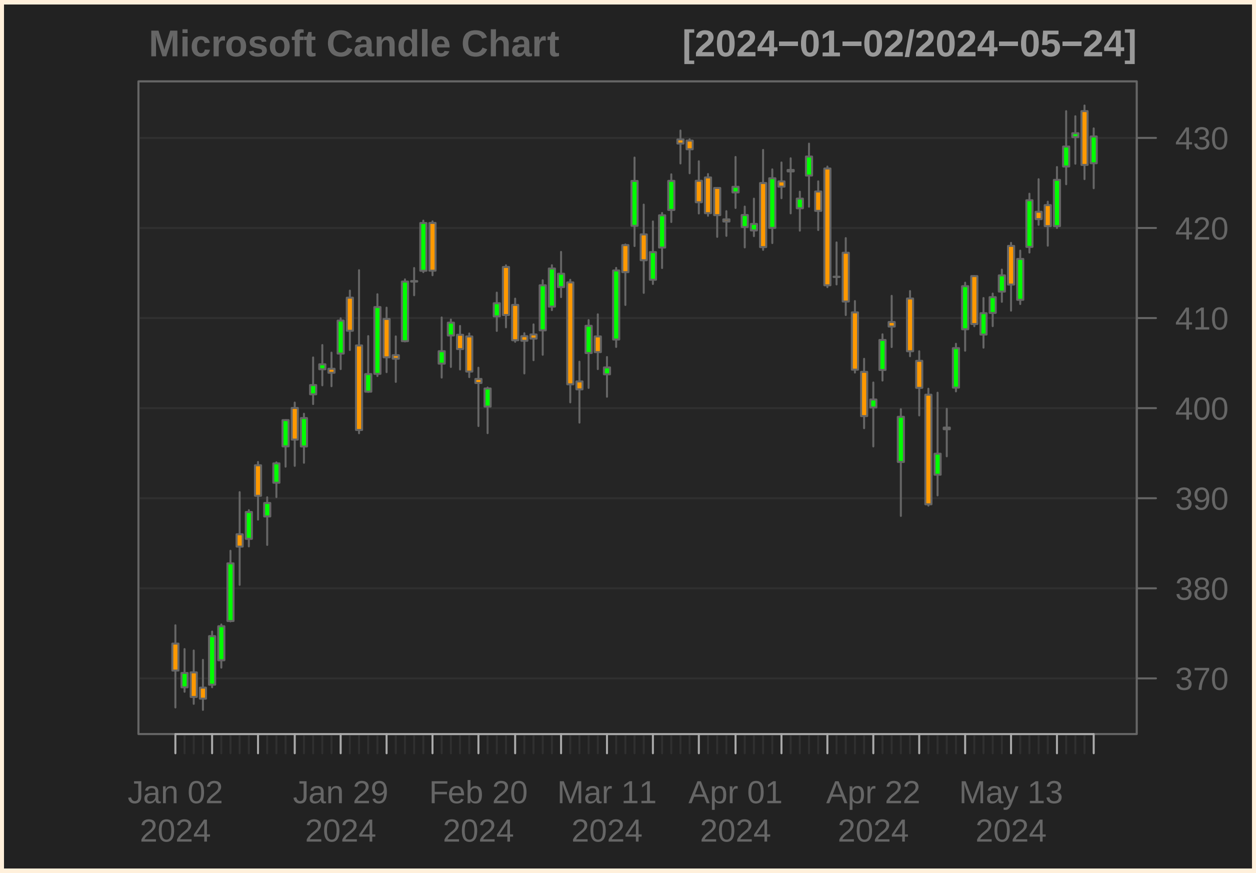
Line charts usually only show the close price of a stock. While not as common, the open, high or close could also be chosen to be represented in isolation instead. Traders utilize this form of chart when one price is considered to be more important than the others. For instance, traders might choose to only pay attention to the close price, so that intraday swing can be ignored.



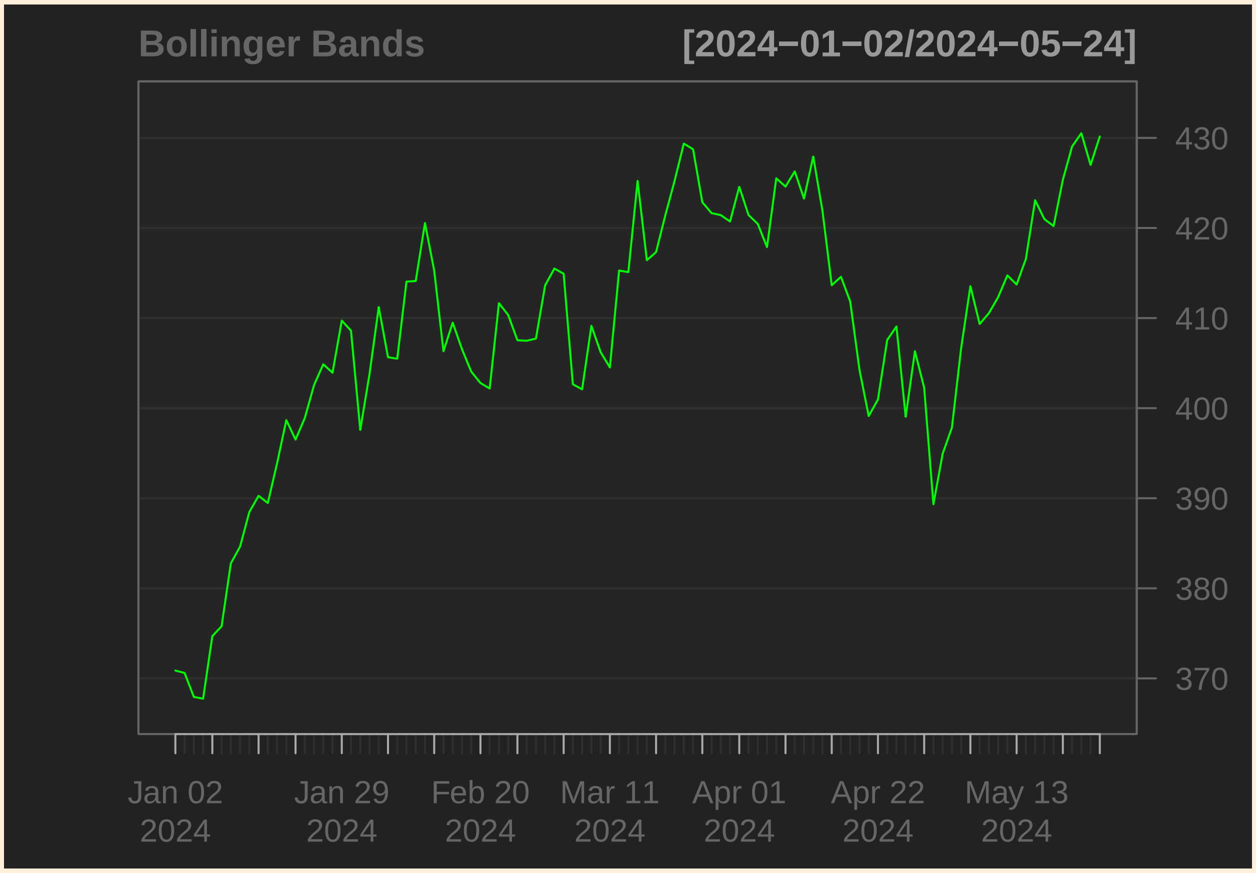
Perhaps the most popular chart are bar charts. Here, the high, low and close price are shown simultaneously for each period.

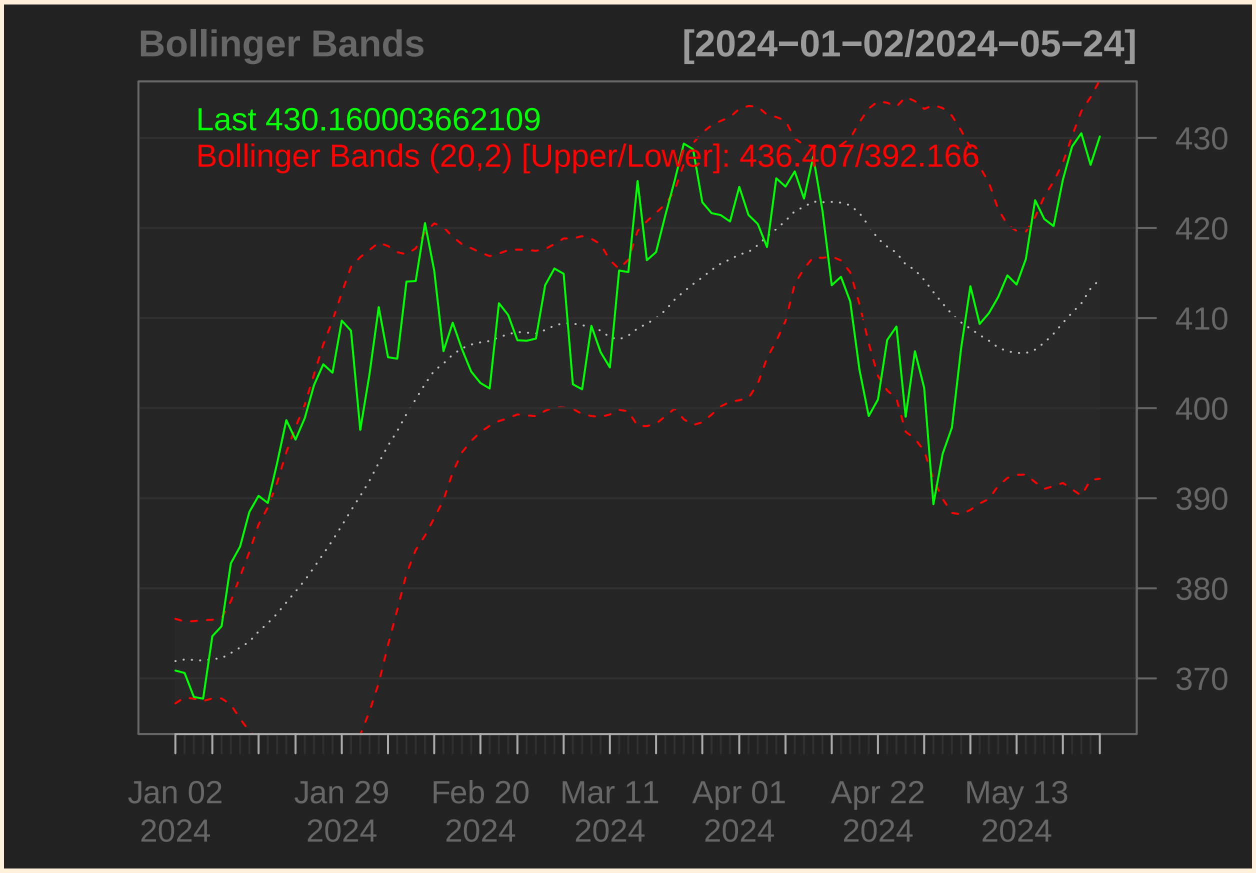


Candlestick charts represent the high, low, open and close of a stock and are visually reminiscent of error-bar plots. The rectangular portion of a candlestick, called the body, represents the opening and closing price of the stock, while the lines above and below, called shadows, stand for the highest and lowest price of the underlying stock in the respective trading period.



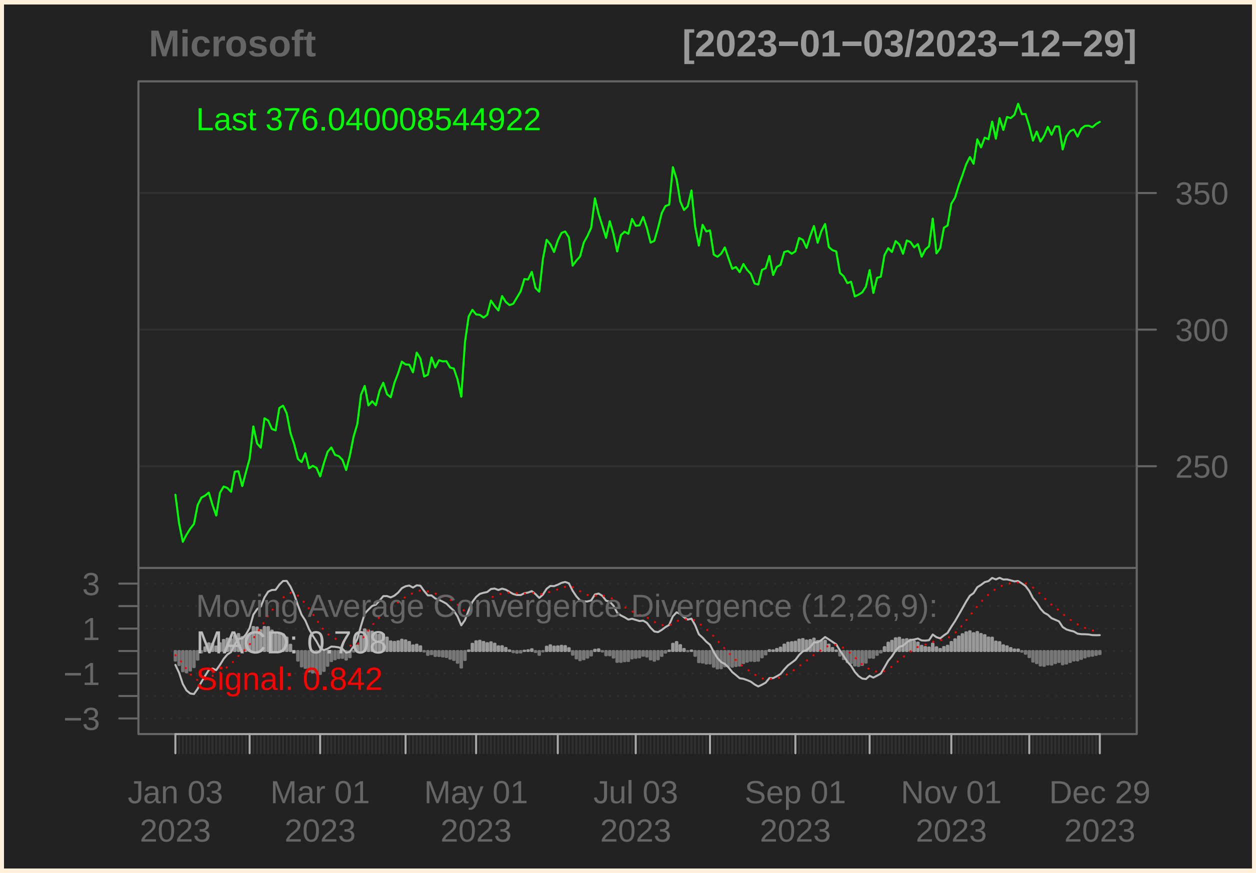
As mentioned, traders usually overlay their charts with various different technical indicators that help them visualize trading signals. A popular visual tool, for example, are Bollinger Bands as they help gauge a securities volatility and can be used to determine if a stock is under- or overvalued.


Bollinger Bands
[2024−01−02/2024−05−24]
430
420
410
400
390
380
370
Jan 02 2024
Jan 29 2024
Feb 20 2024
Mar 11 2024
Apr 01 2024
Apr 22 2024
May 13 2024

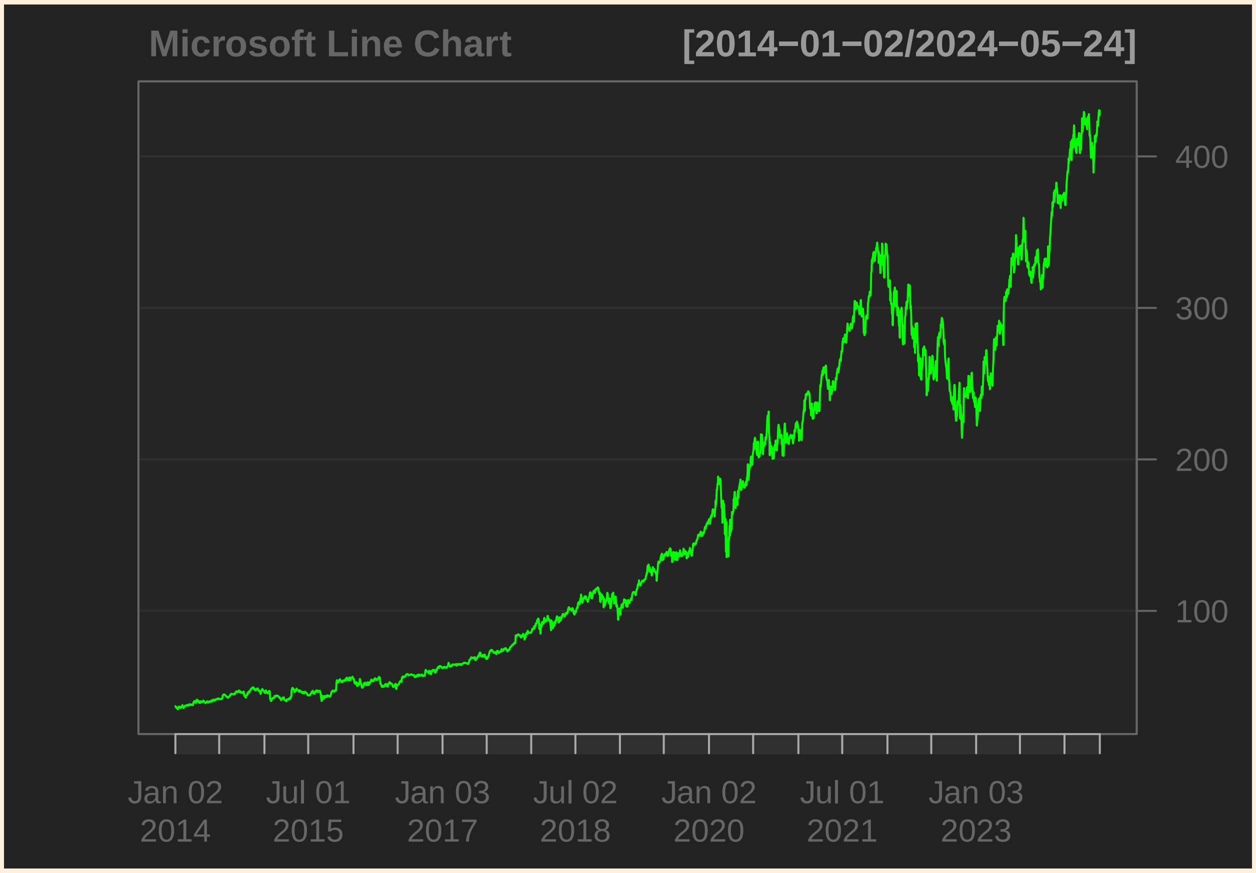


The center line represents the simple moving average of a specified period, usually the 20-day SMA, while the upper and lower bounds are the standard deviation, usually set to 2.

Other technical indicators are often represented underneath the primary chart in a smaller rectangular chart. Oftentimes, traders utilize multiple technical indicators in combination.



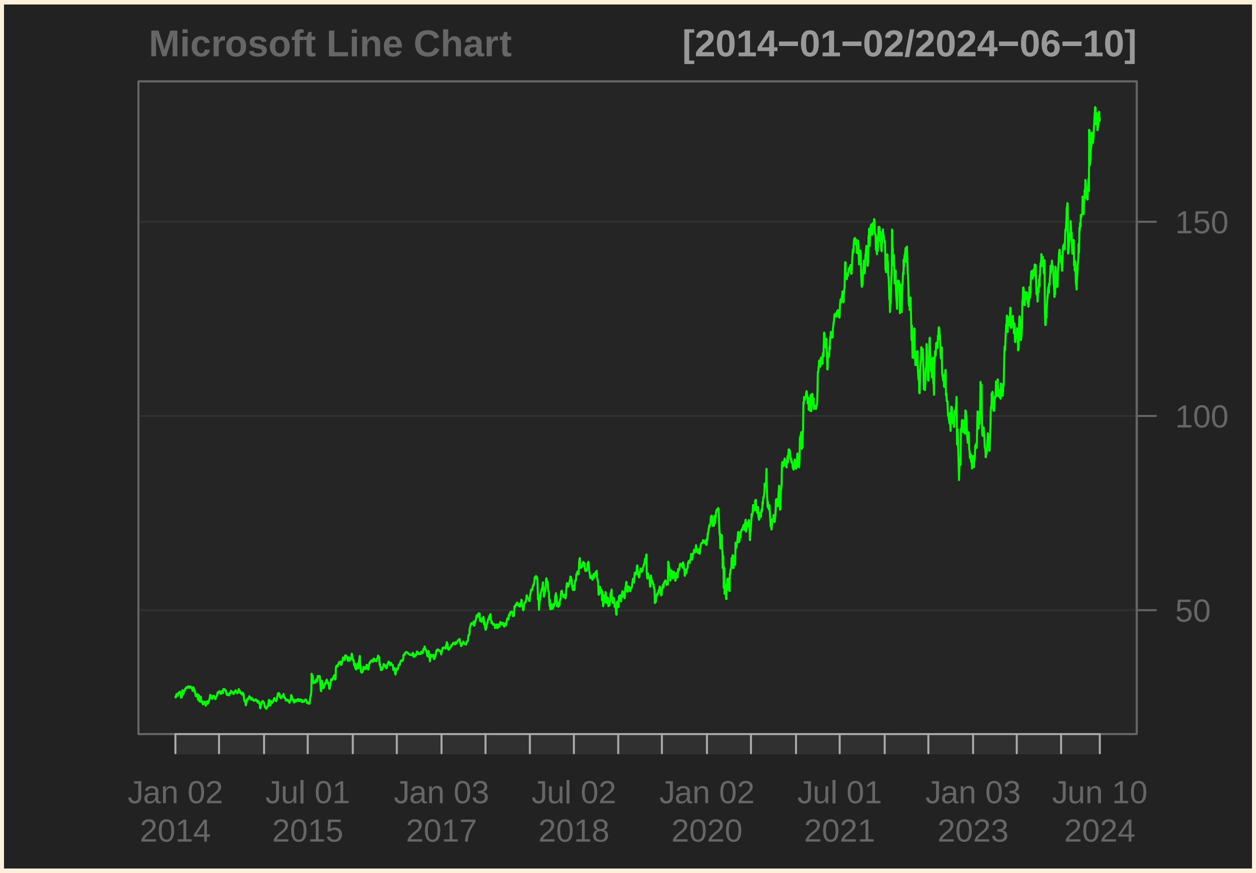
We will refrain from utilizing advanced visual tools to analyze the data as it would go beyond the scope of this project. Instead, we return to the graphical representation of the Microsoft stock as a line chart from 2014 to today, without any overlayed indicators.



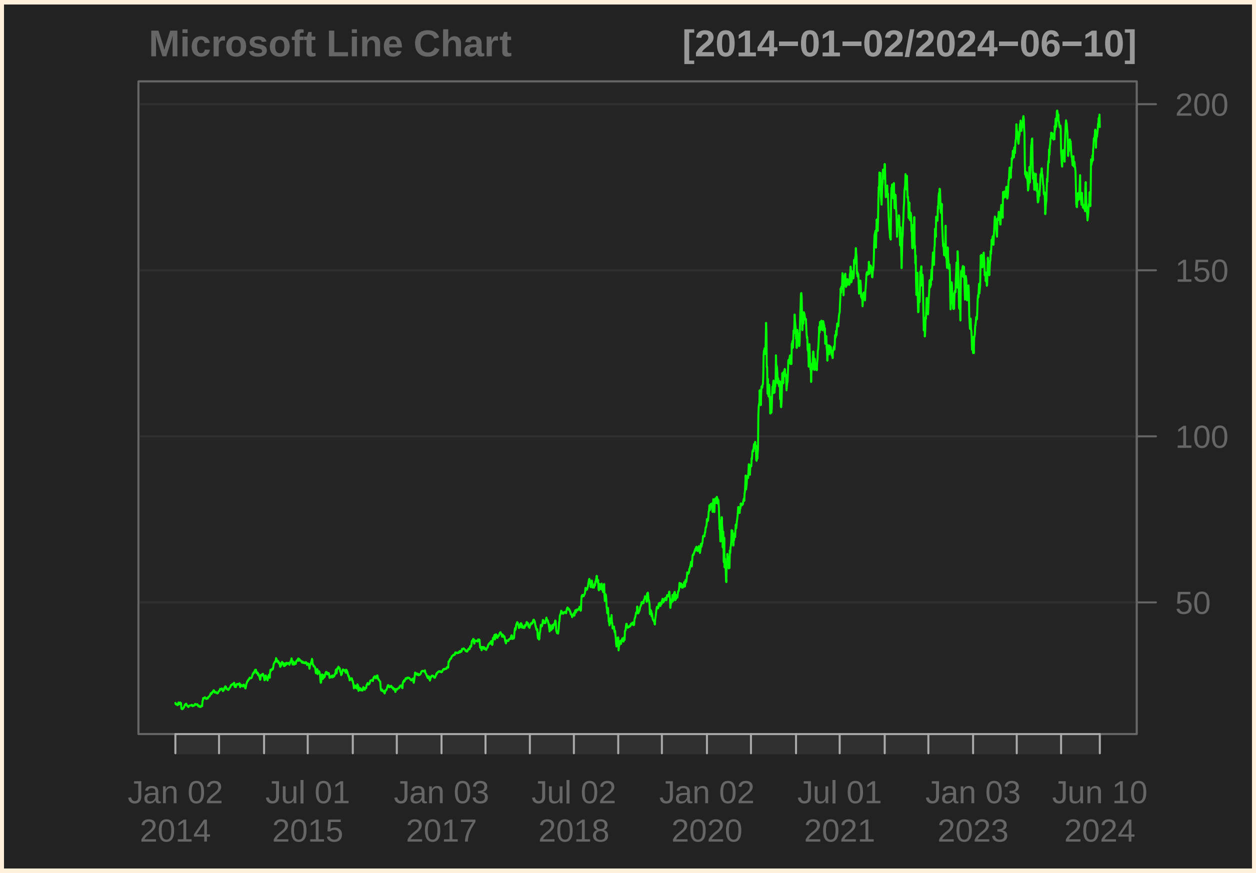
Our earlier assumption holds true, as a general uptrend can be observed, with one major exception.
While the Covid years did not significantly affect the share value, a noticeable price decrease can be seen in 2022. This is not a surprising observation, nor exclusive to the Microsoft stock. In 2022 stock markets globally experienced a decline. For American stock indices, it was the first since the 2008 financial crisis. This decline can be largely attributed to the Russian invasion of Ukraine causing sell-offs across many financial markets.

For comparison, the same observations can be made on Alphabets and Apple stocks, while the latter was affected to a smaller extent.

```
## [1] "GOOG"
```

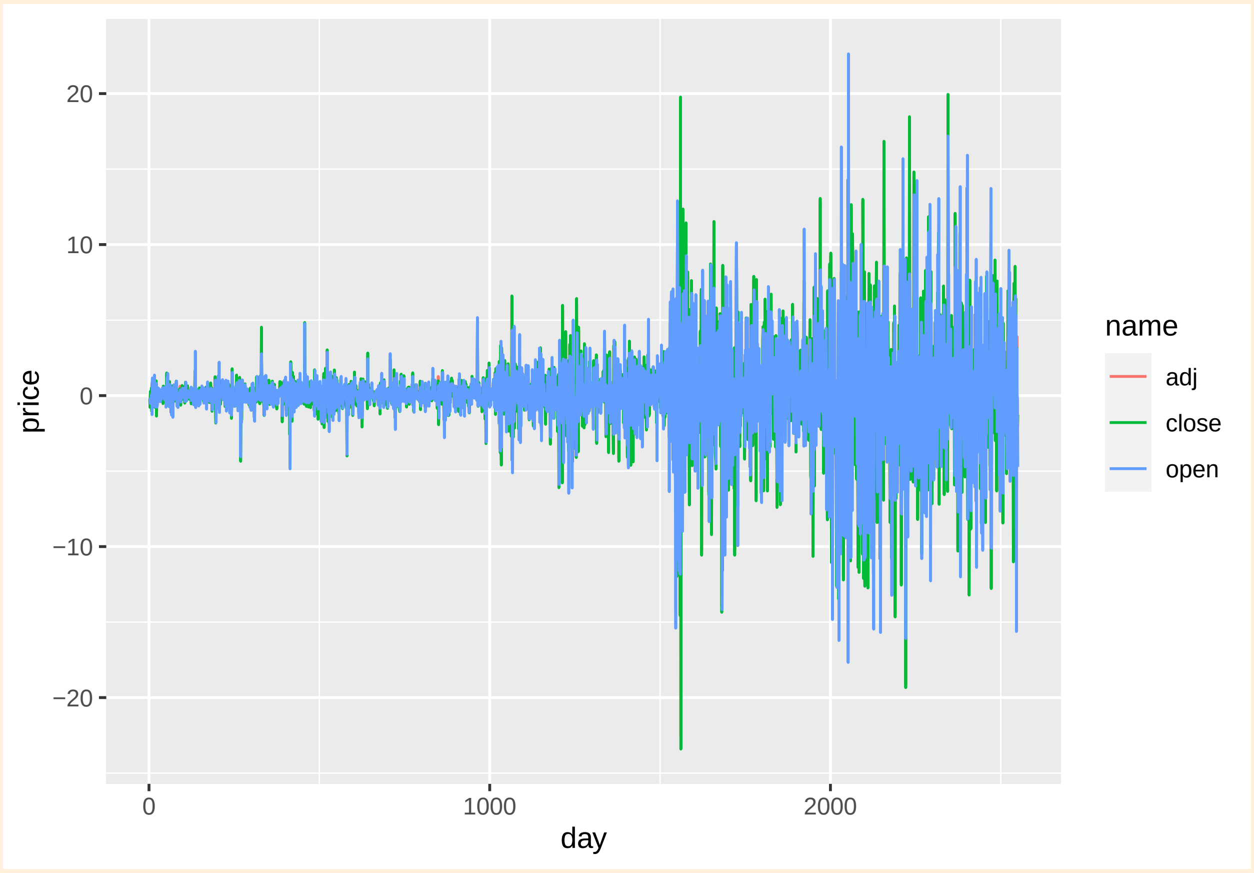

Microsoft Line Chart
[2014−01−02/2024−06−10]
150
100
50
Jan 02 2014
Jul 01 2015
Jan 03 2017
Jul 02 2018
Jan 02 2020
Jul 01 2021
Jan 03 2023
Jun 10 2024

```
## [1] "AAPL"
```

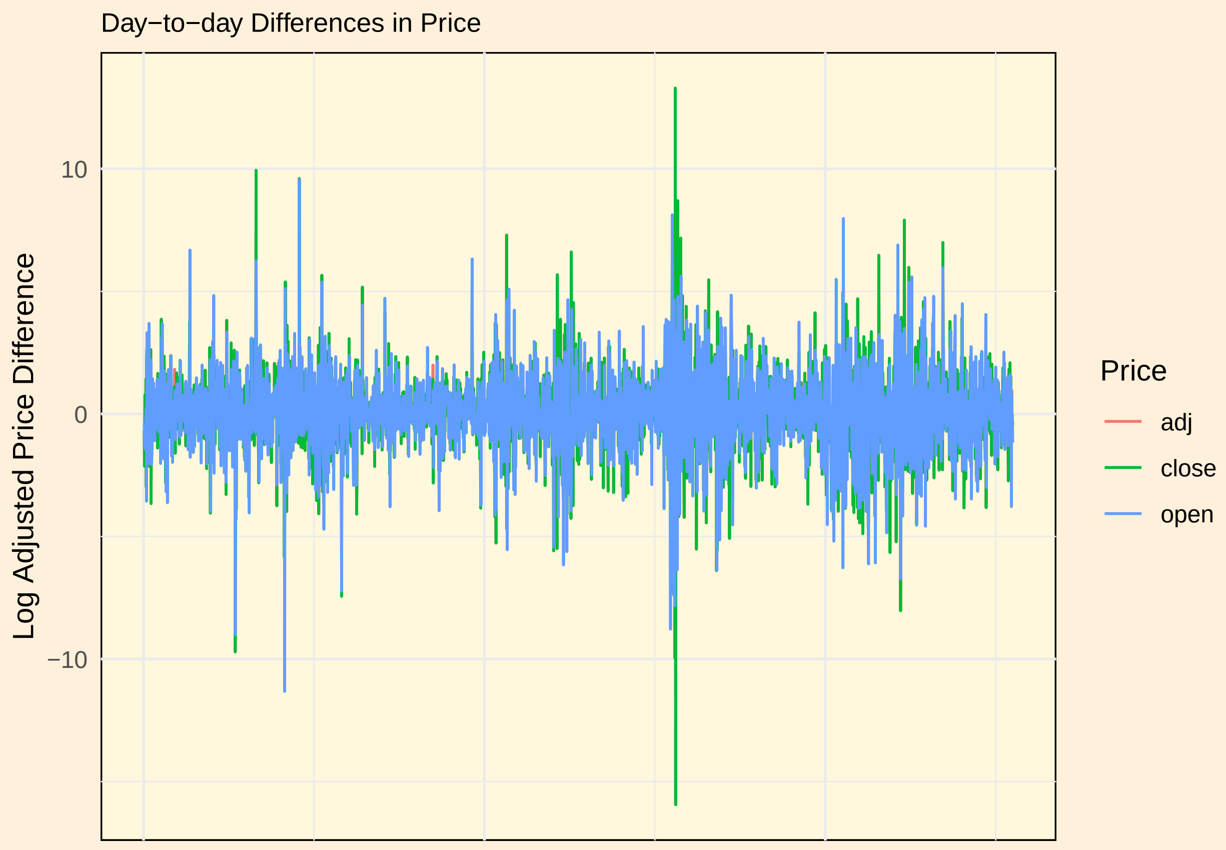


As this event was motivated by outside factors that are challenging to replicate and to quantize in numerical form, it could present a challenge for forecasting models.

While a general uptrend can be observed when we look at the close price of the stock, when taking the daily returns, or in other words, the log adjusted day-to-day percent difference, we obtain a chart reminiscent of white noise.



price
20
10
0
−10
−20
0
1000
2000
day
name
adj
close
open



While this seems to be a random walk process, we will later find out if underlying structure can be found.

Most of the charts shown throughout this section can be easily obtained with the `chartSeries()` function found in the quantmod package.
Many traders and investors can be found utilizing visual analysis and data exploration as their primary means of analysis, or even entirely base their investment decisions on these methods.

Conducting visual analysis and data exploration on stock market data can thus be seen as a discipline in it its own right, but when building stock market forecasting algorithms it provides little value.
The main objective of this section is rather, to build a fundamental understanding of how stock market data can be presented and how the stock market data utilized for later analysis is formatted, in addition to outlining its basic properties.

5 Prophet

The first stock market forecasting method described in this paper will be FB Prophet, an open-source time

series forecasting tool developed by Facebooks Core Data Science Team.
Originally designed for internal company use, it combines ease of usability with powerful underlying algorithms, thus providing a good introduction into the field of forecasting methods. Within this environment, it was intended for tasks such as forecasting sales and optimizing warehouse logistics, which motivated and are reflected by one of its major characteristics, as it focuses on trend forecasting, seasonality, and the effects of holidays.

In the following section, Prophet will be applied to the field of stock market forecasting. The process of which will be described in detail, while the model's unique properties, benefits and limitations will be outlined.
To that extend, Microsoft stock market data will be utilized to forecast its adjusted close, comparing the models performance on forecast horizons of 1, 20 and 260 trading days.

5.1 Prophet Wrangling and Preprocessing

This subsection describes the additional preliminary processing steps needed to build all subsequent Prophet models.

As Prophet requires its input to be in a specific format, we begin by wrangling the previously downloaded data.

```r
msft_x <- msft %>% select(date,adjusted) %>% rename(ds = date, y = adjusted)

msft_x
```