Comparative Study on different Machine Learning techniques for stock price forecasting

Introduction:

ML has become a very important technology in time series forecasting over various fields where researchers apply it to real world challenges. These fields include Healthcare[1], Energy[2], Agriculture[3], Climate & Weather[4] and financial data. Especially in financial data people have a huge interest in making accurate forecasts about developments of for example companies and stock prices given the huge potential for profit in trading.

In this paper we want to evaluate different Machine Learning models against each other based on their performance of forecasting different stock prices and indices. We therefore observe the current literature in this field:

Literature review:

The paper [5] from Kumbure et al. (2022) showed in a comprehensive review the technological development in the field of stock forecasting. The review includes 138 articles that have been published between 2000 and 2019. It looks on machine learning techniques that have been applied, features that were considered for training and metrices with which results have been measured. Like that they propose a differentiation of features into 4 categories which are “Technical Indicators”, “Macro-Economy”, “Fundamental Indicators” and “others”. They explain the importance and use frequency of these features. In section 5.4.8 that paper[5] illustrates what families of machine learning models have been applied to the task of forecasting stock prices. It shows that at least since the year 2000 Artificial Neural Networks (ANNs) in various forms were experimented with excluding Deep Learning models that are listed separately. That differentiation shows that deep learning models have especially been investigated heavily in the more recent years with the first reviewed paper applying them in 2017.

One representative of Deep Learning is mentioned very regularly in research – Long Short-Term Memory (LSTM). The paper [6] by Matharasi et al. (2025) evaluates two models against each other which are LSTM and XGBoost. Both delivered promising accuracy in forecasting AMD inc. stock data. While the LSTM model delivered slightly better accuracy in forecasts, the XGBoost model could excel in computational efficiency. Both models have been shown to have outstanding predictive performance when forecasting certain stock or index data. Teixeira, D. M., & Barbosa, R. S. (2025) have conducted another experiment [13] that shows their effectiveness in capturing the complexity of stock prices. They evaluate the models LSTM and XGBoost as well as other architectures such as GRU, CNN, RNN, but also different hybrid combinations of these like LSTM + CNN, RNN + GRU, GRU + CNN, LSTM + GRU, and RNN + LSTM. The study trains all of these models on historical stock data from Apple inc. and comes to the result that the standalone models of GRU, LSTM and XGBoost perform the best for the Apple inc. stock data. This is a little counterintuitive since one might think that hybrid models usually outperform their standalone component models since they might merge strengths of both models but as evidenced that is not generally the case. Under section 5 they discuss that the performance depends furthermore on the timeframe of the training data. They state that the gap of the performance between the models is smaller for a shorter time frame of training data. They state that the reason for that is the lower complexity of that time frame with fewer large amplitude oscillations. When choosing a longer time frame for training with a more complex data course, the three earlier mentioned models GRU, LSTM and XGBoost outperform the other models. Another paper [7] by Li, Siyuan. (2024) investigates the performance of a Kalman Filter and different types of LSTM models such as a single layer LSTM, a stacked LSTM, a bidirectional LSTM and a hybrid convolutional neural network (CNN)-LSTM. From those the bidirectional LSTM performed best for low-volatility stocks and the CNN-LSTM model performed best for high volatility stocks. Here the hybrid CNN-LSTM model in fact outperformed the standalone LSTM model. XGBoost models as mentioned in the two papers above seem to have a reasonable performance in forecasting stock prices under certain circumstances. This is evidenced by another experiment [10] by Aiyegbeni Gifty, & Dr. Yang Li. (2024) that was conducted to compare the performance of LSTM, ARIMA and XGBoost models. They also come to the conclusion that the XGBoost model outstands in performance under certain circumstances. They trained and evaluated the models on Google inc. stock data. This again emphasizes the impact that the data has on the performance of each model and it shows as the other studies mention as well, that certain models have advantages in dealing with certain characteristics in data like volatility making the choice of a suitable model very much not trivial. They also emphasize the importance of hyperparameter tuning. Like that they could reduce the Mean Absolute Error (MAE) from 17.63 without hyperparameter tuning down to 15.98 and the Root Mean Squared Error (RMSE) from 30.24 down to 27.34.

While the mentioned approaches of deep learning and ensemble methods have been looked into particularly heavy in the last years as shown in the previously mentioned papers and under section 5.4.8 of paper [5], the new transformer based architecture has been proven effective in forecasting stock prices. The paper [14] by Yao, Y. (2025) shows its superiority over the previous mentioned models such as XGBoost, LSTM and CNN under certain circumstances.

Conclusion of Literature review:

In general the advantages and superiorities of different ML algorithms depend strongly on circumstances such as dynamics in the data, feature selection, architecture tuning and hyperparameter tuning. This said, in this study we want to further diversify the results in this area in a way were we can evaluate predictive performance in relation to the characteristics of the data. Out of that we define our research question in the following paragraph.

Research question:

As mentioned in the previous section, to make useful statements about predictive performance of ML algorithms, we need to look on the relation of the performance of the predictions to the data which is used. We observed the dependency of predictive performance on data characteristics during our literature review where we recognized that different ML models can be claimed to be best performing depending on the parameters, input data and setting of the study. For our experiment we want to look on the performance of ML algorithms in forecasting stock prices based on the volatility of the stock data. We want to understand how volatility in the course impacts the forecasting performance of different ML algorithms and we also want to understand which model is most suitable for a certain degree of volatility in the data. To be more precise we are looking for results about the difference in performance of forecasts when using one ML algorithm and stocks of different volatility as input and we want to draw conclusions about the difference in predictive performance when using stock data of a certain volatility as input to train different ML models to see which is best suitable for a certain degree of volatility.

Machine Learning: (This section will tell about the bullet points as general topics but it will focus only on whats necessary to understand later parts of the thesis, how important the parts are for the different models will be concretized in the experimental setup under methodology)

Artificial Intelligence refers to systems that are designed to simulate aspects of human intelligence such as decision, reasoning, perception and adaptation. The important part here is that these systems are capable of learning based on outcomes. With learning we mean that the system can update itself to make better predictions in response to the difference of the predicted outcome and the actual outcome. They don’t have to rely on silicon-based hardware but often do when referred to AI. Machine Learning is a subset of Artificial Intelligence. As IBM puts it [48], “Machine learning is a branch of [artificial intelligence](https://www.ibm.com/think/topics/artificial-intelligence) focused on enabling computers and machines to imitate the way that humans learn, to perform tasks autonomously, and to improve their performance and accuracy through experience and exposure to more data.“. To understand that further we look at an example that clarifies where the difference between any algorithm and ML algorithms comes from. Let’s say we have two algorithms that are designed to tell you how many years you can expect to live. The first one retrieves the average life expectancy for your country from the internet and then returns the difference between the life expectancy and your age as the number of years you have left to live. This algorithm would adapt in a sense where it updates its input which is the life expectancy but it still would not be considered a machine learning algorithm. In other words although the input changes, the algorithm itself never updates so it’s considered non-ML. If the algorithm instead would get the number of years the individual actually had left to live once it died and then update its parameters based on how much the prediction was off, then the algorithm would be considered Machine Learning. So the important difference is that our algorithm directly learns and gets better in its predictions using the value and direction of the error between the predicted and the actual outcome.

Machine Learning is widely accepted to be divided into 3 main subcategories. These subcategories are defined as Supervised Learning, Unsupervised Learning and Reinforcement Learning. There also exist hybrid approaches of these but we don’t need to understand those right now. We just look at these 3 main subcategories. These categories are also referred to as paradigms or learning paradigms.

Supervised Learning:

Supervised Learning is a machine learning technique where labeled data is used for the training. The model makes a prediction for the target variable based on the input and compares the predicted outcome with the actual outcome. This actual outcome is referred to as “label”. The direction and the amount of the error is then processed by the algorithm to learn. We can imagine supervised learning very well as a linear function with multiple input variables and coefficients for every input variable. We are aiming for a certain result value and we want to find the correct coefficients

In that way the algorithm adapts to the actual patterns and relations in the data allowing for better predictions. [48]

Unsupervised Learning:

Unsupervised Learning is a machine learning technique where data is used for the training that is not labeled. So we have no target variable that is to be predicted but instead we just have input data. That means that the algorithm is not designed to predict a certain value based on an input such as in the example with the life expectancy but instead it’s supposed to just work with the input. The algorithm aims for detecting patterns among the data points. We look at an example to get a sense for its usefulness.

Let’s say we have an online shop and we want to give recommendations for items that the customer would be likely to buy so that we sell more products. How can we know which products the customer might buy? Unsupervised learning can be used to improve these recommendations. There are relations or patterns among our items in terms of how they are bought together or separately. Of course this is based on the taste and preferences of the customers. So what we are learning is not an objective relation of items but rather how the customers choose them. Some of these patterns might be obvious such as “these chairs match with this table”, but there are many of patterns some of which are very complex and hard to grasp for humans. As we are constantly adding and removing products from our store it would be infeasible to manually explore and then pick these patterns for a recommendation. So what we do instead is clustering customers into groups. In our example scenario we could say that customers within one cluster share the same taste or preference. The algorithm locates all customers in a multidimensional space where every information a customer can be described with makes up one dimension. the more similar customers are in terms of the information that we use to describe them the closer they are in this space. Areas where many customers are located close together are called clusters. We look at one example for a cluster. Let’s say for whatever reason many customers who buy item a and b also buy item c two weeks later. For humans it can be hard to find patterns as this or much more complex ones. The algorithm would see that there is a higher density of customers in the area of our multidimensional model space for which “bought item a”, “bought item b” and “bought item c” is true. The algorithm looks at how these information fit together and sees that item c is always bought after item a and b was bought and only if both were bought. Like that the simple conclusion is to recommend item c to customers about 2 weeks after they bought item a and b.

The ways unsupervised learning algorithms work differs from architecture to architecture. One common way is to locate the data points in a multidimensional space for which every information about the data is one dimension. Clusters are then defined as areas were the density of data points is higher than elsewhere.

Reinforcement Learning:

Reinforcement learning is a little bit similar to Supervised Learning but the algorithm doesn’t use labeled input data. Instead the algorithm works with trial and error. Successful results will be rewarded and in that way reinforced so the algorithm strengthens that behavior while at the same time bad moves are punished with a penalty and the algorithm learns to avoid that certain behavior.

These 3 Subsets describe *how* we train the system. They specify the data we see, what feedback we get and what objectives to optimize. There is a clear distinction to the ML algorithms. We will discover the 3 algorithms that we are using in this study under methodology. So while the 3 subsets we just mentioned, specify how the algorithm learns in a conceptualized way, the algorithms themselves defines the rules and operations that happen during that learning.

The technique of machine learning that we are using is Supervised Learning. This makes sense because we have an input of stock data and market circumstances and we want the algorithm to learn to predict the next stock price based on that.

The performance of a ML algorithm is measured with by the error. This is usually some way to measure the difference of the predicted value for the target and the actual value of the target. If they differ strongly we would say that the error is greater and the performance is worse. The same vice versa – if the difference between predicted and actual value is small, we say that we have a small error and a better performance. Concrete ways to measure the performance will be explored under Methodology.

(Methodology):

In our experiment we will evaluate different ML algorithms based on their predictive performance in regards of the data used. We choose an LSTM, an XGBoost and a CNN architecture for this evaluation since they have been shown to be relevant algorithms in the field of stock price forecasting as shown in the literature review. We use non-hybrid, vanilla versions of these models to have more meaningful results. We tune their hyperparameters separately.

Model selection:

As we understood before Machine Learning can be divided into 3 subsets. Supervised Learning, Unsupervised Learning and Reinforcement Learning. For our task we are looking at Supervised Learning. LSTM, XGBoost and CNN for time series data are some of the best performing Machine Learning algorithms in stock price forecasting as we conclude from our literature review. We will evaluate all three models on our stock data.

We learned earlier that there are 3 main learning paradigms within Machine Learning. For our purposes we are looking at Supervised Learning at the moment. All paradigms can be further divided into different Machine Learning families. These families are groups of ML algorithms. All algorithms within one group or family rely on the same basic idea or architecture of learning. There are many families we could look into but this would be out of the scope of this thesis. The 2 families under Supervised Learning that we are interested in right now are Neural Networks and Ensemble Learning.

Neural Networks:

Neural networks are systems that are inspired by the structure and functionality of the human brain. They consist of neurons which are connected. The artificial neural network adjusts these connections between the neurons when it learns to improve accuracy and performance. A neural network typically looks like this.

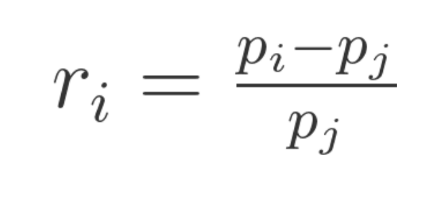
* Machine Learning
  + Supervised Learning
    - Deep Learning
      * Feed Forward (CNNs)
      * Recurrent (RNNs)
        + LSTM
    - Ensemble Learning
      * Gradient Boosting
        + XGBoost
  + Unsupervised Learning
  + Reinforcement Learning

Volatility:

Volatility is known as the degree to which data disperses over time or in other words “Volatility is a statistical measure of the dispersion of returns for a given security or market index. It is often measured from either the standard deviation or variance between those returns.”[15]. We understand that the volatility has to be measured to deliver meaningful results. The most common way to measure the volatility of stock courses in finance is by calculating the standard deviation or more precise the sample standard deviation of logarithmic returns [16]. If we do that we get a measure that tells us how strongly the data points differ on average from the mean of the values. That’s quite good to compare the volatility of different stocks or ETFs.

To do so, we first need to calculate the logarithmic returns. We choose to calculate logarithmic returns instead of simple returns. We can understand why we choose the logarithmic return over the simple return by looking at a simple example.

We calculate the simple return with



… where:

Ri = stands for the current data point’s simple return

Pi = stands for the current data point’s value

Pj = stands for the previous data point’s value

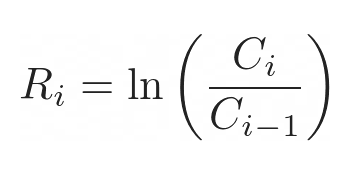
Let’s say we have a stock price development for a company X of the following:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Month 1 | Month 2 | Month 3 |
| Stock price | 100$ | 150$ | 100$ |
| Simple returns | - | +50.00% | -33.33% |

Since the price ends up at 100$ again we know that the total return is 0%. But when looking at the returns we have Month 1-2: +50% and Month 2-3: -33.33%. So we could assume that the average return is (-33.33%+50%)/2 = 8.335% which is clearly wrong. The mistake we made in our calculation is not considering the compound of interest. We need to understand that the 50$ change from Month 2 to Month 3 is already based on a starting value of 150$ which is different than the starting value for the change in value from Month 1 to Month 2, which was again 50$ but this time based on 100$. To make our numbers make sense, we would need to calculate the total return by considering compound of interest by multiplying the returns instead of adding them up. Then we would get the correct 0% total return. The part where the log returns is a more intuitive way of doing it, is when it comes to looking at returns over single time steps as a human. Let’s extend our course of the stock price by another 4 Months:

Our returns now would be

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Month 1 | Month 2 | Month 3 | Month 4 | Month 5 | Month 6 | Month 7 |
| Price | 100$ | 150$ | 100$ | 150$ | 100$ | 150$ | 100$ |
| Simple Return | - | +50.00% | -33.33% | +50.00% | -33.33% | +50.00% | -33.33% |

If we look at returns like these, we could easily think that we end up with a positive return overall but when looking at the actual stock price we see, that we are just oscillating between two values. When we use the logarithmic return all that gets much more intuitive and easier to use. We calculate the logarithmic return with: [17]

… where:

Ri = stands for the logarithmic return at the current data point

ln = stands for the natural logarithm

Ci = stands for the current data point’s value

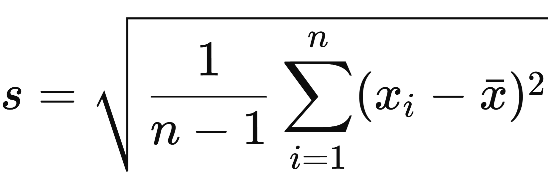
Ci-1 = stands for the previous data point’s value

Let us now also calculate the logarithmic return of our stock course.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Month 1 | Month 2 | Month 3 | Month 4 | Month 5 | Month 6 | Month 7 |
| Price | 100$ | 150$ | 100$ | 150$ | 100$ | 150$ | 100$ |
| Simple Return | - | +50.00% | -33.33% | +50.00% | -33.33% | +50.00% | -33.33% |
| ln Return | - | 0.405 | -0.405 | 0.405 | -0.405 | 0.405 | -0.405 |

If we now add up the logarithmic returns we are left with 3\* (0.405) – 3\* (0.405) which is 0. So using the logarithmic return instead of actual stock prices itself or simple returns of the stock prices, leaves us with numbers that are detached from the compound of interest effect and give us a much more intuitive understanding of the volatility. We now have our logarithmic returns, which leads us to the next step which is calculating the sample standard deviation based on them.

To calculate the sample standard deviation we use this formula.



… where:

n = stands for the number of data points that we calculate the sample standard deviation for

xi = stands for the currently observed data point (for us it’s the current logarithmic return)

= stands for the mean of all data points (for us it’s the mean of all logarithmic returns)

In our codebase we write our own pipeline to calculate the volatility in the way we just discussed. For that we use a pre-implemented function for the sample standard deviation which is located in *pandas.DataFrame.std()*.

The standard deviation gives information about how strongly the data points are deviated from the statistical mean. If the standard deviation is lower it means that the data points are on average closer to the mean and if the standard deviation is higher it means that the data points are spread wider.

We now have defined our metric for volatility and we can start choosing the stock data we want to train the algorithms on.

So far so good. We now have a measure that tells us how volatile the stocks have been in the past. The volatility score is averaged over our time frame which is 5 years. We could go ahead and select the stocks based on that score. If we do so, we might see, that stocks with the same volatility score seem to have much different characteristics in the data.

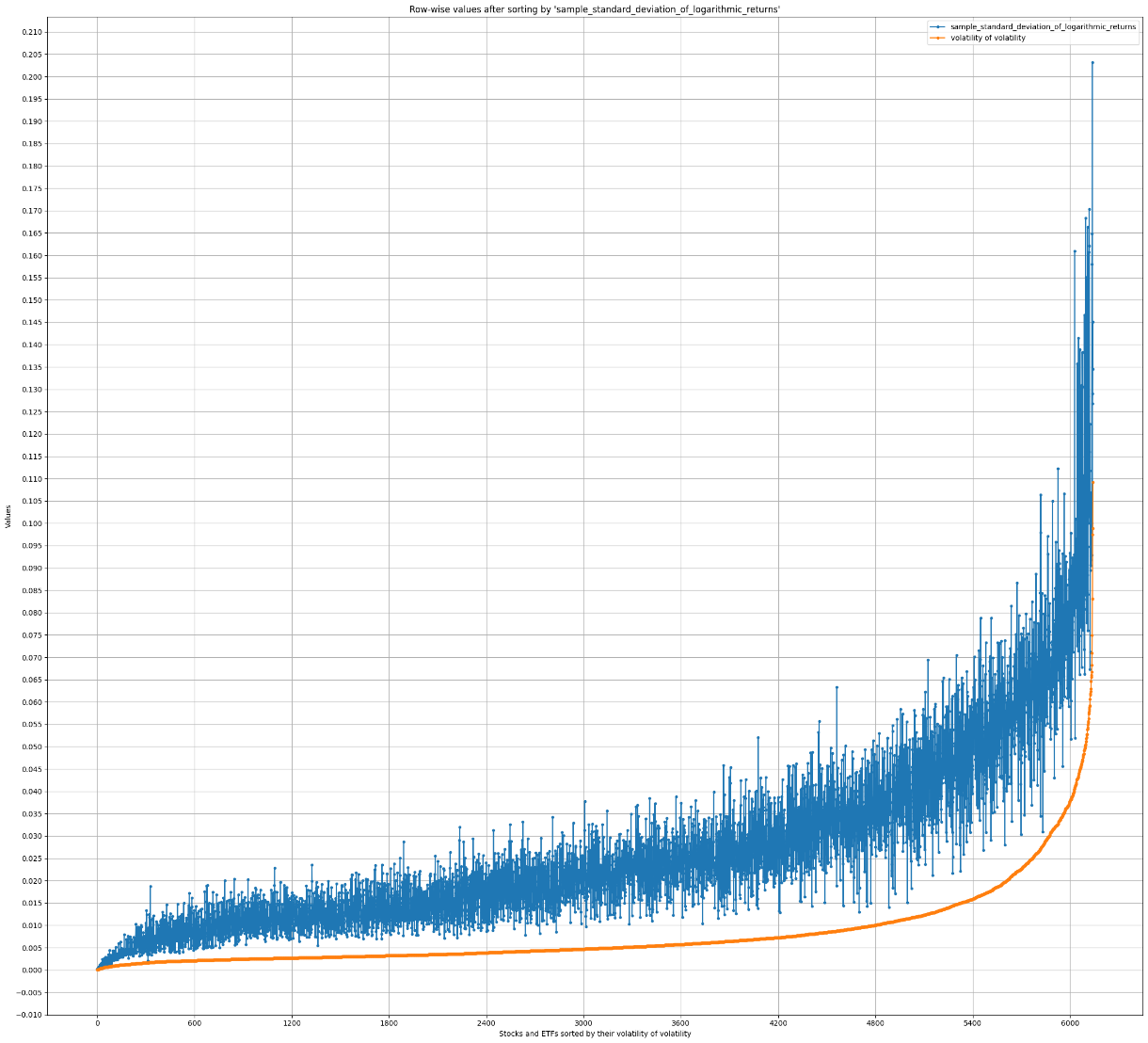
To understand that using an example we take two stocks. One stock is oscillating moderately strong throughout the whole course let’s say its volatility score is 5 at every point in time. The other stock oscillates compared to the first one very little during the first half but a lot during the second half. Let’s say during the first half it has a volatility score of 1 and during the second half it has a volatility score of 9. The average volatility of these two stocks is 5 even though they have very different characteristics and the first stock has a more constant oscillation which doesn’t change as much with time. To account for that problem and to make sure that the stocks we chose are as similar as possible in terms of their volatility we look at another metric called the volatility of the volatility. As one might think this metric simply measures how much the volatility itself changes over time. To measure that we calculate the pure volatility just as described above but not for the whole set of data but instead multiple times for rolling windows. So we basically compute the volatility for every 75 consecutive data points in the data set. After having the volatility for each of these windows we again calculate, just as explained above, the volatility of these volatility measures. Like that we get the volatility of the volatility for a stock or ETF.

Stock selection:

We have defined how we measure the volatility of a stock course. That enables us now to choose stock data based on which we want to train our models. In the codebase the Jupyter Notebook file ‘Volatility\_Pipeline.ipynb’ completely contains this process. At first we are using the alpaca-trade-api to retrieve our historical stock data. This API requires you to verify yourself with an API\_KEY and an API\_SECRET. After we created ourself an account and retrieved the credentials, we now create a list which contains common stocks and ETFs listed on AMEX, ARCA, BATS, NYSE, NASDAQ or NYSEARCA [18]. After that we define the time frame for which we want to retrieve the historical data, we choose it to be the past 5 years until 2025-05-07 (yyyy-mm-dd). We then start to make calls to retrieve the data. We batch the call so that we get 200 stock courses with one call. This is due to the rate limit of the alpaca api [19]. Like that we retrieve the historical prices of about 11500 stocks and ETFs.

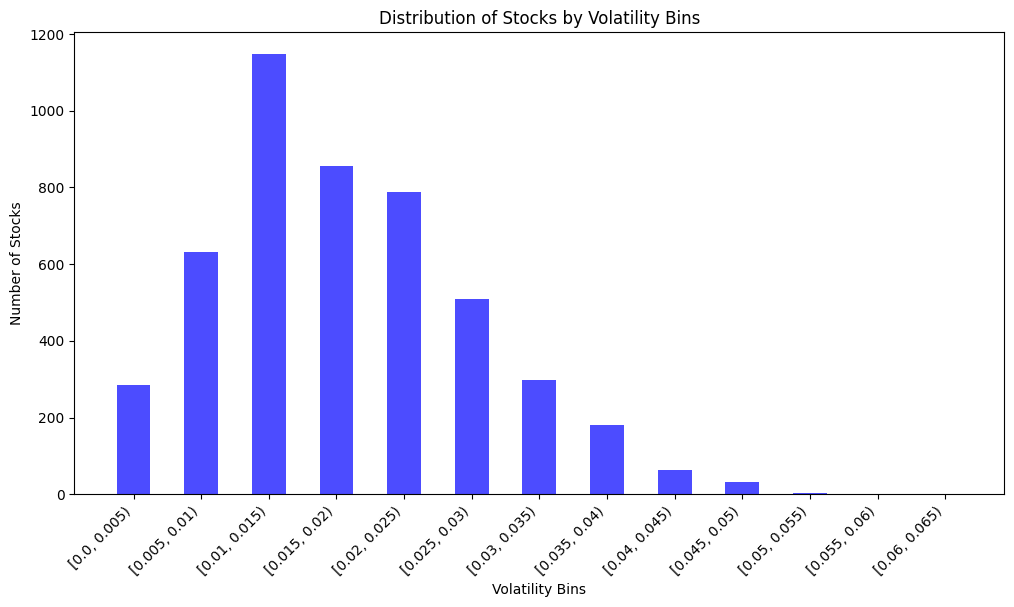
We further process this data to be only left with a table with the stock symbols as columns, the timestamp as index and the close price as a column as well. During preprocessing we would need to account for missing values. It would be much easier to already now just chose stocks and ETFs without missing values. So we look how many stocks and ETFs have missing values and its about 5000. That means we have about 6500 stocks and ETFs left with no missing values. Since we only need handful of historical courses, we simply decide to drop all that have missing values and just proceed with the ones without any missing value. Besides that we also drop all historical courses that have the same closing price over at least 30 days. When a stock or ETF has the same closing price over a number of days it can have different reasons but usually means that the asset hasn’t been traded in that time. For whatever reason the price stays the same for at least 30 days, we don’t want such asset courses in our data because it is obviously not natural behavior of the asset price and it gives the algorithm a hard time to learn.

We now want to extract information regarding volatility. To do so we apply the methods of calculating the volatility and calculating the volatility of the volatility for our stocks as described under methodology.Volatility. We apply the functions and store the returned values in a table. As we understood before we the volatility of the volatility to be as low as possible for all stock and ETF courses while at the same time having a wide range of different volatilities of which we use stocks and ETFs to train our model. To filter out stocks and ETFs that have a higher volatility of volatility we need to visualize our data set. For that we draw a graph. The graph shows the measure of volatility (blue) and volatility of volatility (orange) on the y-axis for each stock or ETF and the number of the stock on the x-axis. We sort the stocks by the volatility of their volatility in increasing order.



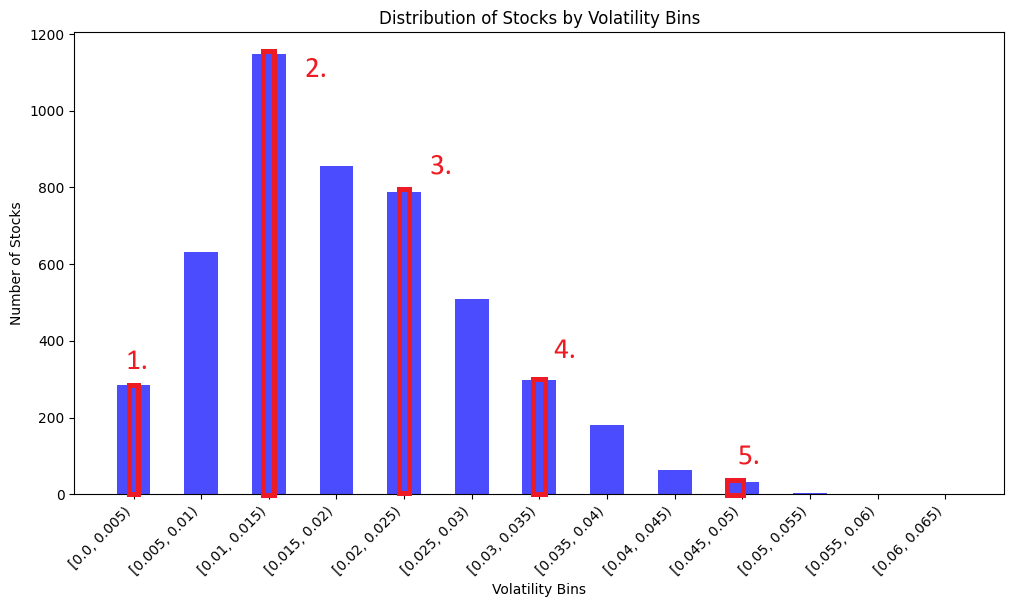
We see that the volatility of the volatility seems to correlate on average with the volatility. The greater the volatility of the volatility the greater the pure volatility. Now, for us two things are most important. The first one is that we want to have stocks from volatilities within a range that is as wide as possible to have a more meaningful result of our experiment. The second important thing is that we want the volatility of the volatility to be as low as possible. We explained before why we want that. Since the volatility seems to correlate with the volatility of the volatility we need to make a compromise when deciding for a value of the volatility of the volatility which serves as a threshold. We will just drop every stock or ETF that has a volatility of volatility greater than that threshold. We take into consideration that we only need a relatively small number of stocks and ETFs. Besides that we take into consideration that the volatility of volatility seems to increase dramatically on the right side of the graph so we definitely want to cut off at some point before that dramatic increase. After careful consideration we decide to cut off all stocks and ETFs that have a volatility of their volatility of more than 0.01. This is pretty much at stock 4800 in the graph. Like that we ensure to have constant oscillation based on the given volatility while still being able to capture a wide range of stocks and ETFs with different volatilities.

Across the range of volatility we want to evenly pick 5 values for each of which we again pick 15 stocks or ETFs that are as close as possible to the values in terms of volatility of the historical data. To get a sense for that we first create bins of volatility and sort our stocks and ETFs into these bins. We plot that distribution.



We now want to manually choose 5 bins and 15 stocks or ETFs for each bin for training. To make the results as meaningful as possible we define the ranges of our bins in a way so that we capture the widest possible range of volatility. In other words we want bin1 and bin5 to be as far apart as possible on the graph and the remaining 3 bins to be in between them while having equal distances to their neighbors. At the same time we want the stocks and ETFs within one bin to be as similar as possible in terms of their volatility. Based on the distribution which is visible in the plot, we choose our volatility steps to be:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Volatility score interval | 0.00235  – 0.0027 | 0.01243  - 0.01248 | 0.022425  - 0.0225 | 0.032416  - 0.0326 | 0.045  - 0.046 |



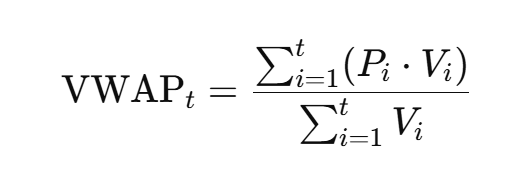
This will allow us to train and evaluate the models based on multiple stock data for each value of volatility which we expect to give us a statistically better and closer to the average measurement compared to just using one stock for each step which is susceptible to outliers. We especially expect the models to have a similar predictive accuracy for stocks within the same volatility interval and to show clear jumps in performance when trained based on stocks of different intervals.

Data Preprocessing:

During the previous section (‘stock selection’) we selected the stocks that can be grouped into 5 intervals or steps of different volatility and which we want our models to be evaluated on. Since we already dropped stocks with missing values, we don’t need to perform any further imputation.

Our historical data so far has the features: open, high, close, low, vwap, volume, trade\_count, symbol – where symbol is the ticker symbol of the stock and will not be used for training.

* trade\_count tells how many transactions have been made in a period.
* Volume is the number of shares that have been traded within the period.
* Volume Weighted Average Price (VWAP) is the typical price of a stock or ETF weighted by its volume [36]



We now preprocess the historical data from each stock by adding other features. Some of them are retrieved from APIs and some are calculated using the existing features.

We choose the features based on which the review study by Kumbure et al. [5] has shown to be mainly used in stock price forecasting. We also add some indicators as features from [20] and [21].

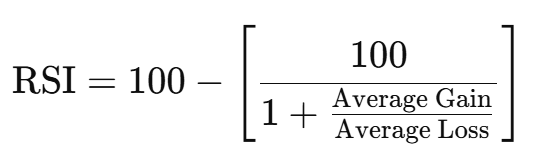
Feature Engineering:

Technical Indicators:

Relative Strength index (RSI):

The RSI is a way to measure speed and directions of price movements. The RSI always lies in the range of zero to 100 and it is commonly considered overbought when above 70 and oversold when below 30. The RSI formula is:

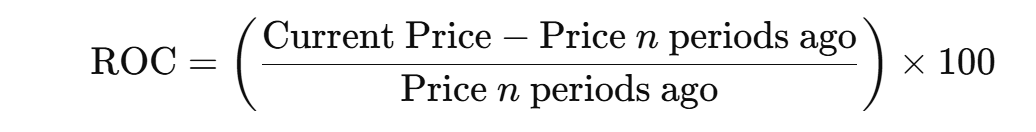
RSI = 100 – [100 / ( 1 + (Average of Upward Price Change / Average of Downward Price Change ) ) ]



Where Average Gain and Average Loss are Exponentially weighted moving averages. [22]

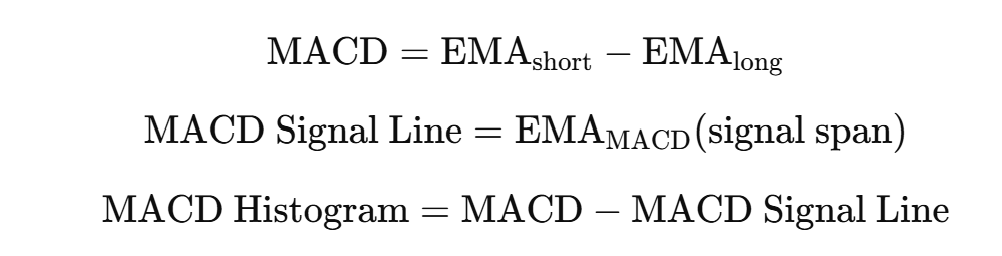
Rate of Change (ROC):

The ROC is an indicator that shows the return in percent not compared to the previous period but to the price n periods ago. A common number for n is 14. [23]



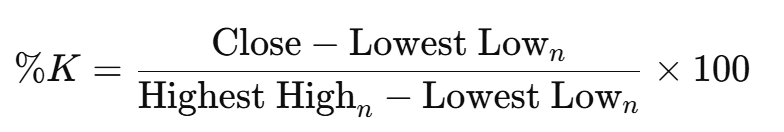
Moving Average Convergence Divergence (MACD):

The MACD is used to measure both the momentum and direction of a price movement. It consists of a the MACD line and the signal line. The MACD line measures the difference between a shorter exponential moving average and a longer exponential moving average. The signal line is again an exponential moving average of the MACD line. Due to the fact that both lines crossing has implications for the price movement and we want the model to better identify that relation, we also add a histogram column that simply is calculated as the difference between MACD line and signal line. [24]



Stochastic Oscillator:

The stochastic Oscillator is another momentum indicator that measures where the current closing price lies in the range of the highest high of a certain past period and the lowest low of that same period in percent. This value is called %K. A second value which is called %D contains the moving average of %K for a short period. Since again the relation of those 2 lines is of importance to us and we need to make the machine learning model consider this relation even though it cannot see the the lines on a chart we add a stochastic difference value that is calculated as the difference of these 2 values. [25]

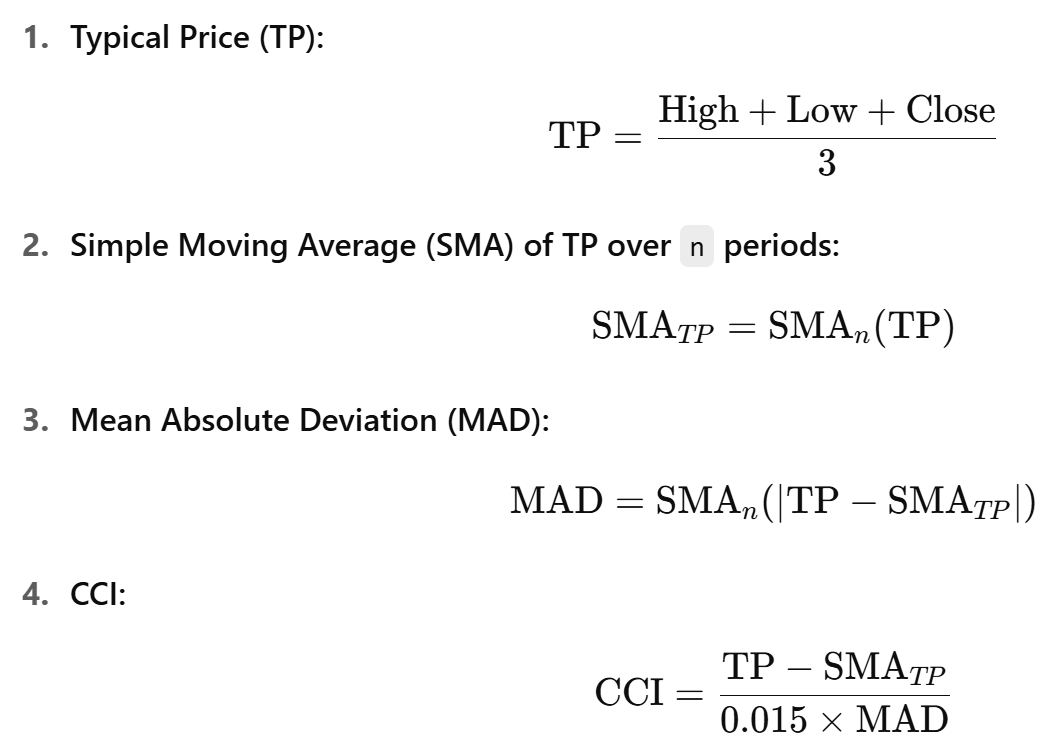






Commodity Channel Index (CCI):

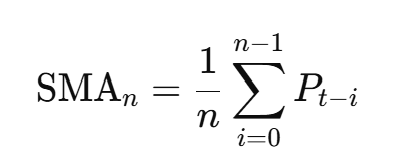
The CCI measures how far a price is from its statistical average indicating overbought or oversold market conditions. [26]



Trend Indicators:

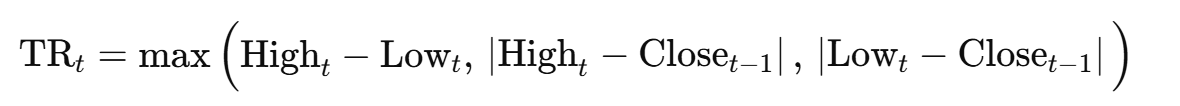
Moving Averages:

We add multiple simple moving averages with different window sizes. Moving averages smooth out price data and indicate an upwards trend when increasing and a downwards trend when decreasing. Besides that they can indicate overbought or oversold conditions. [27]



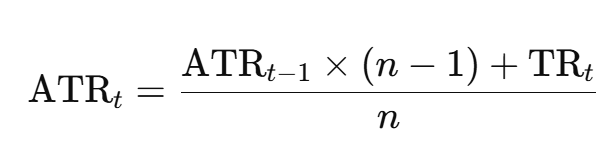
Average True Range (ATR):

The ATR is a volatility indicator that tells how much the price moves on average per period. We therefore first look at the True Range (TR) for every period. This is the maximum price difference within that period. We add that difference to our feature set.



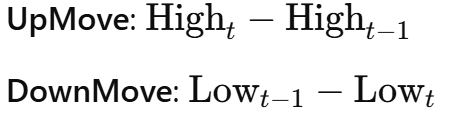
We use the previous day’s close price because the current day’s open price can differ from the previous day’s close price due to overnight news that lead to changes in the sentiment. [30]

After that we use wilder’s smoothing to smooth out the True Range over a certain windows size, usually 14, so that we understand how volatile certain periods are. [31]

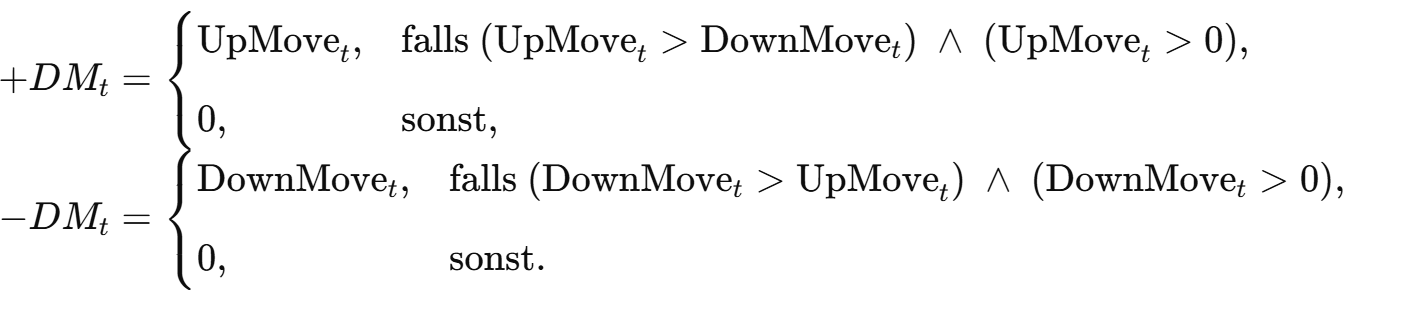


Average Directional Index (ADI):

The ADI uses the ATR as a component. The ADI and its components measure how much of the Average movement within a certain window which we derive from the ATR is upwards movement and how much is downwards movement. For that we calculate our up and downwards movement like this. [32]



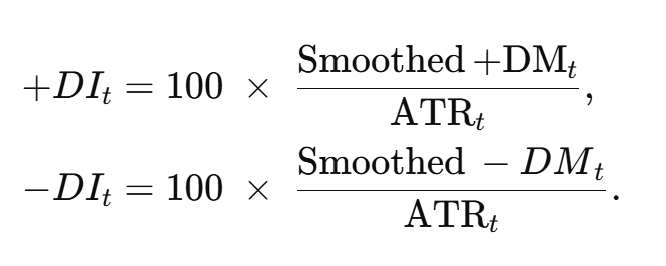
UpMove contains the difference of the high values so that increases have a positive value. DownMove contains the difference of low values in a way that downward movements have positive values and upwards movements have negative values.



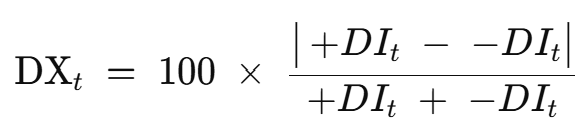
+DM now stores all absolute increases in the high value for periods in which the low value decreased less than the high value increased.

-DM works vice versa.

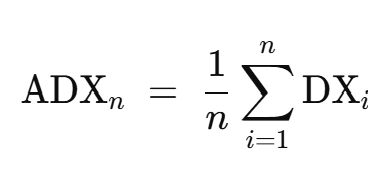
We now average these 2 values over a window that must have the same size as the window size n from our ATR. Like that we get the average up and down movement in absolute values for the given window. We calculate the percent to which our over all movement which we derive from the ATR is upwards movement or downwards movement.



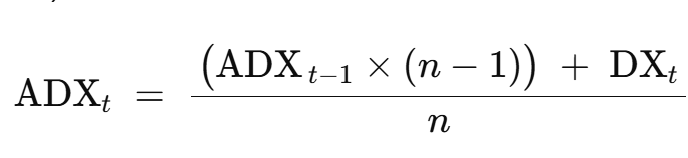
The next step lets us calculate the Directional Movement Index (DX). DX tells us in percent how one-sided a movement is. If up and down movement have the same portion of the overall movement then DX is 0%.



The last step is to calculate the actual ADX. We again use wilder’s smoothing. That is a recursive function so we need to calculate the first ADX value manually by simply averaging the previous n DX values.



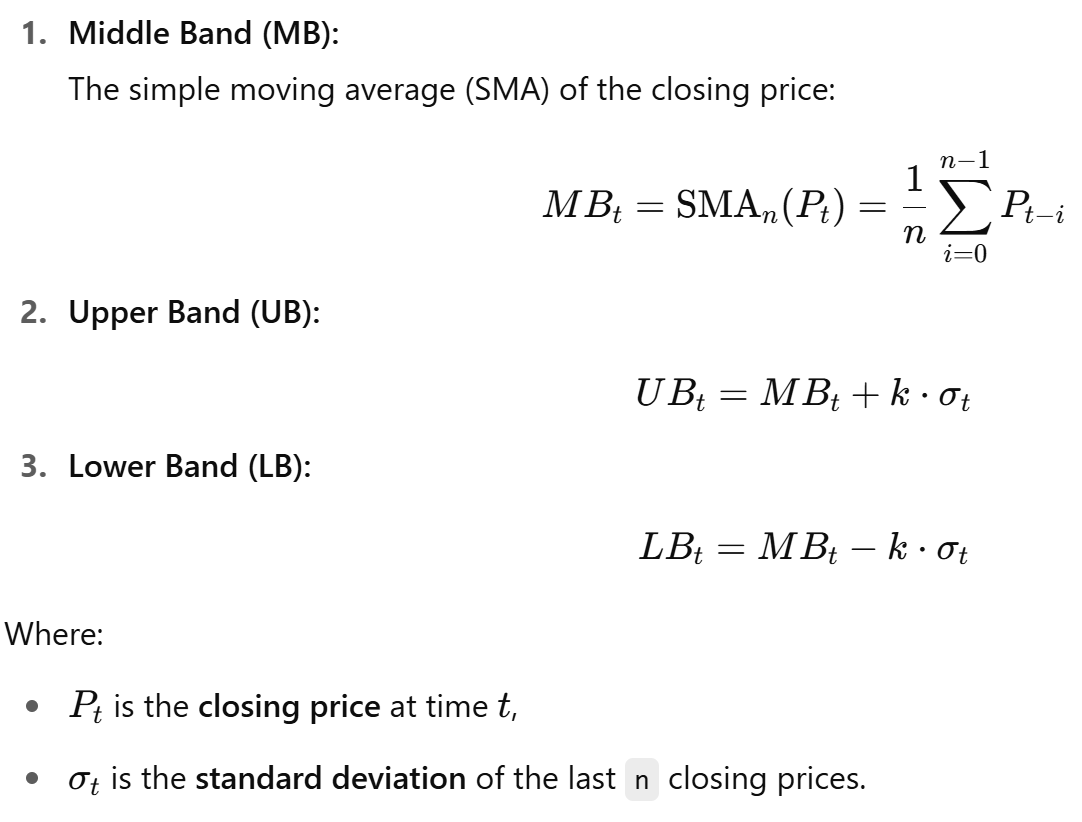
Any further ADX value is calculated recursively.



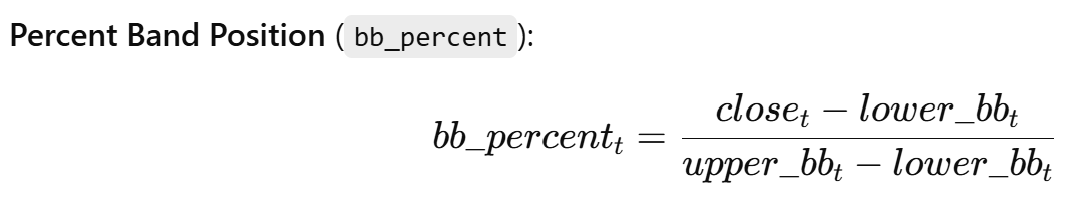
Bollinger Bands:

The Bollinger Bands are indicators to gauge the volatility of a stock to understand whether they are over or undervalued. [33]

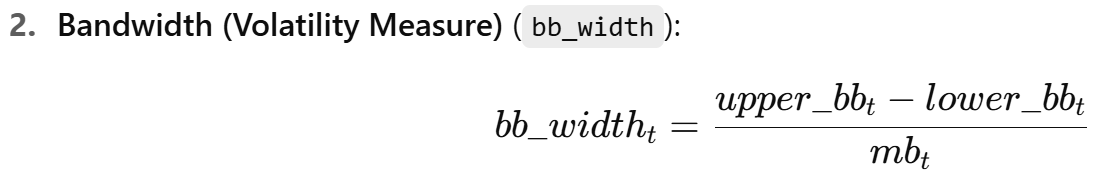
They consist of 3 lines:



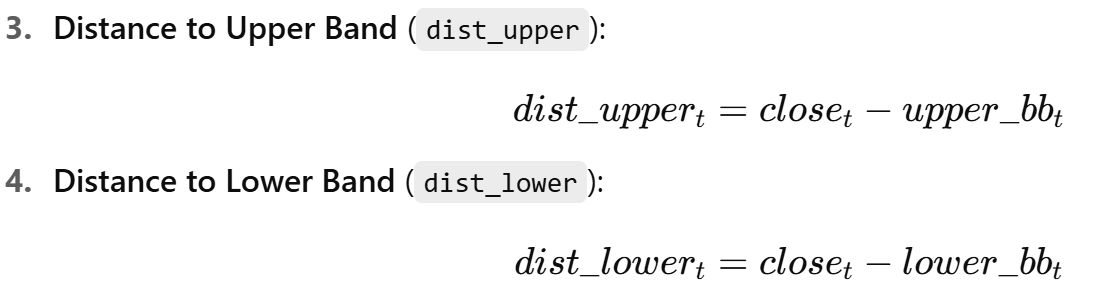
Since we use this indicator for machine learning purposes, we will add the following features to emphasize relations between the bands or our close price and the bands.



* Measures from 0 to 1 where the close price is within lower and upper band



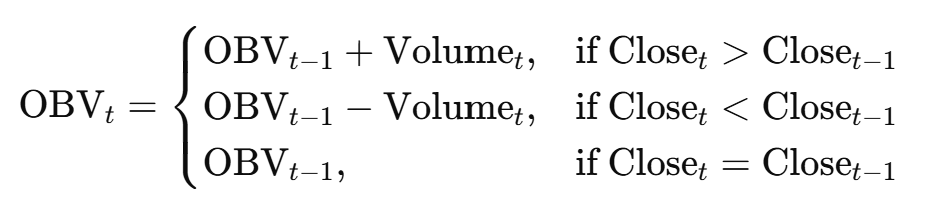
* The band width tells how volatile the course is.



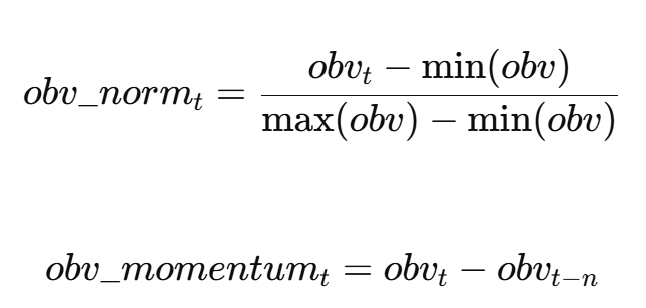
Volume based indicators:

On-Balance Volume (OBV):

The On-Balance Volume Indicator is a momentum indicator that uses volume flow to predict price movement. It assumes that volume precedes the price – so a rising OBV signals buying pressure and a falling OBV signals selling pressure. [34]

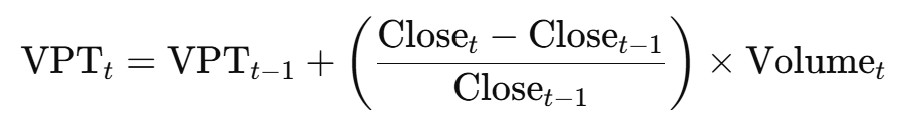


Besides the absolute OBV we add a normalized OBV for better machine learning suitability and an OBV momentum that compares the OBV to the OBV n periods before.

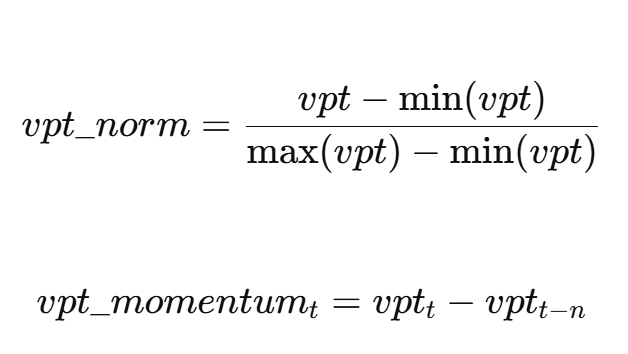


Volume Price Trend (VPT):

The VPT is a momentum indicator that combines the price change with volume. It’s purpose is to track how strongly volume aligns with or contradicts price changes. [35]



For Machine Learning purposes we also add the VPT in a normalized way and the VPT momentum.



Macro-economic features:

We add a couple of Macro-economic features to capture global or US national circumstances and market context [38]. We use the “FRED API” version 0.5.2 to retrieve them [37].

S&P 500:

One important macro-economic indicator is the S&P500 index. Since it includes the biggest 500 US companies it is an important measure of the us economy.

GDP(USA):

The Gross domestic Product is the value of all good and services produced by a country in a given period. It is a great indicator since it reflects the economic performance of a country [38].

Unemployment rate:

The unemployment rate can capture strengths and weaknesses in the economy. It is a great indicator for economic activity [38].

Consumer price index (CPI):

The consumer price index measures the monthly change in prices paid by urban US customers for different basic products and services [39].

Personal consumption expenditures (PCE):

The personal consumption expenditures is an indicator that measures how much US households spend on goods and services [41].

Industrial production index (IPI):

The industrial production index measures the output in manufacturing, gas, oil, mining and electric industries [42].

federal funds rate:

The federal fund rate is the target interest rate range at which commercial banks are supposed to borrow money [43].

10Y treasury rate:

The 10 year treasury is the interest rate which the US government pays to borrow money over a decade. It is an important benchmark for other interest rates and it tells about expectations of the US government about inflation and economic growth [44].

Retail sales:

Retail sales is a main indicator for economic health because it provides information about the buying power of consumers and bigger retail sales usually lead to more profit for companies which has a positive effect on the stock price [40].

Housing starts:

Housing starts is the number of newly started construction projects in a given period that provides information about the housing market and general economic activity [45].

Exchange rates:

Exchange rates are a good way to measure the economic health in comparison to economies from other countries [46] [47]. We include the exchange rates USD-EUR, USD-GBP and USD-JPY.

Calendar Features:

We also add the following calendar features to capture patterns for example the Monday effect [28] and the January effect [29].

* Day of week (1-7)
* Day of year (1-365)
* Month (1-12)
* Year (yyyy)
* Quarter (1-4)

In total we now have included x features in our data set that were products or byproducts of our calculations.

We also add a ‘target’ column with the next days close price to use for training. This target column will be the y\_train, y\_val and y\_test values. The other columns will be our X\_train, X\_val and X\_test values. How we split the data into train, validation and test set will be explored later during preprocessing.

We now have to create 2 further pipelines that serve the distinct needs of the models. LSTM and CNN since they have a similar architecture hereby share same further preprocessing steps so they will share one pipeline. The XGBoost architecture works differently so we need a distinct preprocessing for it.

The scaler will be fitted on training set to not let the algorithm “see” ranges of future data. After having scaled our data, we will create windows of size 20 for training. The last step is to split the data.

The XGBoost model has a different preprocessing. Here we don’t need to scale values since XGBoost is a tree based model where only the relations of the data matter but not their absolute value. In normalization the relation of the data points doesn’t change but only their absolute value so we don’t need normalization for XGBoost. Instead we need to add lag values. We decide to add 5 lag values for the core features such as close, open, high, … and we add 3 lag values for other indicators such as RSI, ROC, volume, adx, obv, … and so on. We don’t add lag values for all features since this would be quite redundant for example for the moving averages.

LSTM,CNN pipeline:

1. Normalization
   1. Scaling based on training set
2. Windowing
   1. Window size of 20
3. Splitting
   1. X\_train y\_train
   2. X\_val y\_val
   3. X\_test y\_test

XGBoost pipeline:

1. Lag values
   1. For some we do 3 lags
   2. For some we do 5 lags
2. Splitting
   1. Train 0.8 ; Validation 0.1 ; Test 0.1

// Set sizes and number of lag values etc. could be great hyperparameters, but we have a lot of other hyperparameters and in order to stay in the scope of this thesis we wont tune the parameters of the preprocessing steps. //

Hyperparameter tuning:

When we looked into the different Machine Learning models that we are using, we understood the hyperparameters for each model. We want to tune these hyperparameters for each model so that it has an optimal performance.

To do so we use the optuna library. With optuna we have a straight forward workflow. In optuna we create an object called study. This study takes 2 parameters which are one function that returns a value and a direction in which the value which is returned from the function should be optimized. These directions can be “up” or “down” for numeric values. Optuna runs the function a certain amount of times and each time uses different hyperparameters inside the function. At places inside the function values can be suggested. These suggested values are recorded. In the end one distribution of hyperparameters is linked to a certain performance. After all trials are done the hyperparameter combination that gave the best result is stored and we can retrieve it as well as the score for the performance of that trial with the best combination of hyperparameters.

To use this functionality we implement our function to train one of our 3 models with the certain stock or ETF data and return the “rmse”. Like that we can say that the hyperparameters of the model are the values that should be optimized. And the value based on which the optimization happens is the rmse value.

Metrices:

To evaluate the performance of our trials during the hyperparameter tuning we use 3 different metrices.

RMSE:

The Root Mean squared error (rmse) is a performance metric. It takes the predicted value and the true val and calculates the difference from it. Then it squares this difference for all pairs of predicted and true value and takes the mean of these squares of differences. That mean is the rmse value. The best possible value is 0.0.

Example:

>>> y\_true = [3, -0.5, 2, 7]  
>>> y\_pred = [2.5, 0.0, 2, 8]  
>>> mean\_squared\_error(y\_true, y\_pred) = 0.375

MAE:

The Mean Absolute Error (mae) calculates the absolute difference of predicted and true value and takes the mean of these differences. The best possible value is 0.0.

Example:

>>> y\_true = [3, -0.5, 2, 7]  
>>> y\_pred = [2.5, 0.0, 2, 8]  
>>> mean\_absolute\_error(y\_true, y\_pred) = 0.5

* R2

Results & Discussion:

1. Ghazaleh Babanejaddehaki, Aijun An, and Manos Papagelis. 2025. Disease Outbreak Detection and Forecasting: A Review of Methods and Data Sources. ACM Trans. Comput. Healthcare 6, 2, Article 13 (April 2025), 40 pages. https://doi.org/10.1145/3708549
2. Qureshi, M., Arbab, M.A. & Rehman, S. Deep learning-based forecasting of electricity consumption. *Sci Rep* **14**, 6489 (2024). https://doi.org/10.1038/s41598-024-56602-4
3. Venkatesan, S., Lim, J., Ko, H., & Cho, Y. (2022). A Machine Learning Based Model for Energy Usage Peak Prediction in Smart Farms. *Electronics*, *11*(2), 218. https://doi.org/10.3390/electronics11020218
4. https://www.ft.com/content/78d1314b-2879-40cc-bb87-ffad72c8a0f4
5. Mahinda Mailagaha Kumbure, Christoph Lohrmann, Pasi Luukka, Jari Porras, Machine learning techniques and data for stock market forecasting: A literature review, Expert Systems with Applications, Volume 197, 2022, 116659, ISSN 0957-4174, <https://doi.org/10.1016/j.eswa.2022.116659>
6. T. Matharasi, S. Lakkaraju and I. Mukherjee, "A Comparative Analysis of Machine Learning Approaches for Stock Price Prediction," *2025 International Conference on Pervasive Computational Technologies (ICPCT)*, Greater Noida, India, 2025, pp. 975-979, doi: 10.1109/ICPCT64145.2025.10941156
7. Li, Siyuan. (2024). Comparison of Machine Learning Models for Stock Prediction. Advances in Economics, Management and Political Sciences. 85. 76-84. 10.54254/2754-1169/85/20240846.
8. Drucker, H., Burges, C. J., Kaufman, L., Smola, A., & Vapnik, V. (1996). Support vector regression machines. *Advances in neural information processing systems*, *9*.
9. Cortes, C., Vapnik, V. Support-vector networks. *Mach Learn* **20**, 273–297 (1995). https://doi.org/10.1007/BF00994018
10. Aiyegbeni Gifty, & Dr. Yang Li. (2024). A Comparative Analysis of LSTM, ARIMA, XGBoost Algorithms in Predicting Stock Price Direction. *Engineering And Technology Journal*, *9*(8), 4978–4986. https://doi.org/10.47191/etj/v9i08.50
11. Gifty, A., & Li, Y. (2024). A Comparative Analysis of LSTM, ARIMA, XGBoost Algorithms in Predicting Stock Price Direction. *Engineering and Technology Journal*, *9*(8), 4978-4986
12. Gururaj, V., Shriya, V. R., & Ashwini, K. (2019). Stock market prediction using linear regression and support vector machines. *Int J Appl Eng Res*, *14*(8), 1931-1934.
13. Teixeira, D. M., & Barbosa, R. S. (2025). Stock Price Prediction in the Financial Market Using Machine Learning Models. *Computation*, *13*(1), 3. <https://doi.org/10.3390/computation13010003>
14. Yao, Y. (2025). Stock Price Prediction Using an Improved Transformer Model: Capturing Temporal Dependencies and Multi-Dimensional Features. *Journal of Computer Science and Software Applications*, *5*(2). <https://doi.org/10.5281/zenodo.14832392>
15. <https://www.investopedia.com/terms/v/volatility.asp>
16. <https://www.macroption.com/historical-volatility-calculation/>
17. <https://medium.com/%40manojkotary/simple-returns-vs-log-returns-a-comprehensive-comparative-analysis-for-financial-analysis-702403693bad>
18. <https://alpaca.markets/deprecated/docs/api-documentation/api-v2/assets/?utm_source=chatgpt.com#asset-entity>
19. <https://alpaca.markets/support/usage-limit-api-calls>
20. <https://en.wikipedia.org/wiki/Template%3ATechnical_analysis>
21. <https://www.investopedia.com/terms/t/technicalindicator.asp>
22. <https://www.investopedia.com/terms/r/rsi.asp>
23. <https://www.investopedia.com/terms/p/pricerateofchange.asp>
24. <https://www.investopedia.com/terms/m/macd.asp>
25. <https://www.investopedia.com/terms/s/stochasticoscillator.asp>
26. <https://www.boerse.de/technische-indikatoren/Commodity-Channel-Index-(CCI)-8>
27. <https://www.investopedia.com/terms/m/movingaverage.asp>
28. <https://www.investopedia.com/terms/m/mondayeffect.asp>
29. <https://www.americancentury.com/insights/the-january-effect-and-stock-market-seasonality/>
30. <https://support.zerodha.com/category/trading-and-markets/trading-faqs/general/articles/opening-price-of-a-stock-different-from-its-previous-day-s-closing-price>
31. <https://www.investopedia.com/terms/a/atr.asp>
32. <https://www.investopedia.com/articles/trading/07/adx-trend-indicator.asp>
33. <https://www.investopedia.com/terms/b/bollingerbands.asp>
34. <https://www.investopedia.com/terms/o/onbalancevolume.asp>
35. <https://www.boerse.de/technische-indikatoren/Volume-Price-Trend-(VPT)-56>
36. <https://www.investopedia.com/terms/v/vwap.asp>
37. <https://fred.stlouisfed.org/docs/api/fred/>
38. <https://www.investopedia.com/articles/investing/031413/economic-indicatiors-affect-us-stock-market.asp>
39. <https://www.investopedia.com/terms/c/consumerpriceindex.asp>
40. <https://www.investopedia.com/terms/r/retail-sales.asp>
41. <https://www.investopedia.com/terms/p/pce.asp>
42. <https://www.investopedia.com/terms/i/ipi.asp>
43. <https://www.investopedia.com/terms/f/federalfundsrate.asp>
44. <https://www.investopedia.com/articles/investing/100814/why-10-year-us-treasury-rates-matter.asp>
45. <https://www.investopedia.com/terms/h/housingstarts.asp>
46. <https://www.investopedia.com/articles/economics/08/leading-economic-indicators.asp>
47. <https://www.investopedia.com/terms/e/exchangerate.asp>
48. <https://www.ibm.com/think/topics/machine-learning>

Words to explain:

Average  
Exponentially weighted moving averages (EWMA or EMA) with formula  
Simple moving average (SMA) with formula  
period. I would say when you read the whole thing you just write down every word you need to explain.

* A system updates itself (same as third)
* Error (in prediction)
* To learn (when a system updates itself and learns from error)
* Data point