Stock Price Prediction Using Decision Tree
Semester V
&
Automation Of Trading Tools Used In The Stock Market
Semester VI

A Report Submitted to
SKVM's
Narsee Monjee Institute of Management Studies (NMIMS)

as a part of Semester V & VI
of
Bachelor of Science (Honours) Mathematics

By

Ansh Kharbanda

Under the Guidance of

External Mentor: Dr. Vinay.S. Kulkarni
Internal Mentor: Dr. Priyabrata Bag
Internal Mentor: Dr. Gaurish Telang

Nilkamal School of Mathematics, Applied Statistics & Analytics
Mumbai
2023

Certificate

The work described in the report entitled **"Stock Price Prediction Using Decision Tree(Semester V) & Automation Of Trading Tools Used In The Stock Market(Semester VI)"** has been carried out by Mr. Ansh Kharbanda under our supervision. We certify that this is his bonafide work. The work described in this report is original and has not been submitted for any degree to this or any other university.

Date: $17^{th} May, 2023$
Place: Vile Parle, Mumbai

External Mentor

eSign
Signed by: Vinay Shridhar Kulkarni
Reason: Report approval
Location: Thane
Date: 03-May-2023 (11:26 PM)

Dr. Vinay S. Kulkarni

Internal Mentor

Dr. Priyabrata Bag

External Examiner

Dean

Dr. Sushil Kulkarni

Statement By The Student

This is to submit that this written submission in my report entitled "Stock Price Prediction Using Decision Tree (Semester V) & Automation Of Tools Used In The Stock Market(Semester VI)" represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I stuck to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/ data/ fact/ source in my submission. I understand that any violation of the above will be cause for disciplinary action by the School and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Mr. Ansh Kharbanda

Forwarded through,

Dr. Vinay.S.Kulkarni
External Mentor
Dr. Priyabrata Bag
Assistant Professor

Nilkamal School of Mathematics, Applied Statistics & Analytics
SVKM's NMIMS,
Vile Parle (W),
Mumbai – 400056

Acknowledgements

I am grateful to Dr. Sushil Kulkarni for providing me with the opportunity to undertake this project. I also extend my appreciation to Dr. Vinay.S.Kulkarni for inspiring the concept behind this project. My sincere thanks go to my mentors, Dr. Gaurish Telang and Dr. Priyabrata Bag, for their invaluable guidance on the technical aspects of the project. Lastly, I would like to extend a special thank you to Mr. Vansh Kharbanda for his guidance & support and Mr. Devansh Sonigra for his suggestion and also Mr. Rakesh Keswani for connecting me with beneficial contacts.

Contents

Abstract

Investing in the stock market is a complex and challenging process that requires careful analysis and prediction of the future performance of companies. With the rise of big data and machine learning, there is an opportunity to automate this process and provide investors with valuable insights into the market.

This project aims to do just that by developing a trading algorithm for predicting stock prices using machine learning methods like Decision Trees. The algorithm is designed to predict the increase or decrease in stock prices with associated confidence levels a month ahead or more for any company listed in the market. The outcome of buying, selling, or holding a stock will be based on the predicted values.
Additionally, by automating a number of tools used by traders and investors that can provide them with valuable insights about when to enter or exit the stock market, as well as at what levels the price will reverse course or continue in that direction, or, in other words, whether the stock price will be bullish or bearish and when.
To begin with the project, we must first familiarizes ourselves with the important concepts of technical analysis, including trends, candlestick patterns, support and resistance, and chart patterns and what do these tell us about the price level or entry/exit in the stock market. These concepts are then researched and learned to observe chart patterns of companies and identify specific features for the increase or decrease in price.

Using Python, Yahoo Finance, TA-Lib library and web scraping methods, we then automate the process of collecting financial data. The code is evaluated over any chosen company to determine whether to buy, sell, or hold the stock. The generated value's precision score is also examined, and the cause of any inaccuracies is determined.
By using the tools automated in this project as features to input in the decision tree upon which the nodes can be split to make better decisions, we can increase the precision score of the decision tree algorithm. In addition, it is also possible to compare the decision tree with other machine learning models such as random forests and XGBoost.

We also examine the potential risks and uncertainties associated with using these models. Finally, we analyse the results of the project and draw conclusions about the effectiveness of the trading algorithm. And discuss the potential applications of the algorithm in real-world scenarios, including its use by individual investors, financial institutions, and other stakeholders in the stock market. We also identify areas for further research and development, including the use of more advanced machine learning models and the incorporation of addi-

tional data sources.

In conclusion, this project offers a practical and automated approach to study the stock market based on technical analysis and using machine learning algorithms. The results obtained can be useful for investors and traders looking to make informed decisions in the stock market. With the continued growth of big data and machine learning, the potential for this type of automated trading algorithm will only continue to increase in the years to come.

Chapter 1

Introduction & Problem Statement

1.1 Topic:

1.1.1 Stock Price Prediction & Stock Market Tools Automation

The method of forecasting future stock prices using statistical and machine learning approaches is known as stock price prediction. The objective is to find patterns and trends in historical data that may be applied to forecast the performance of a certain company or a group of stocks in the future.
Predicting stock prices is difficult since they depend on a variety of variables, including market mood, company news, economic data, and geopolitical events. Additionally, the financial markets are highly volatile and uncertain, which makes it challenging to predict stock values with any degree of accuracy.
For predicting stock prices, a variety of techniques can be utilised, such as time-series analysis, machine learning, deep learning, and artificial intelligence. These methods entail looking at past data to find trends, patterns, and connections between various variables. The model can be used to forecast future stock values once it has been trained on historical data.
Portfolio optimisation, risk control, and investment decision-making are just a few of the many uses for stock price forecasting. It is crucial to keep in mind that predicting stock prices is not a surefire strategy to make money on the stock market, so investors should always proceed with care and do their homework before making any kind of investing decisions.

1.1.2 Problem Statement:

Combining the information each feature like the Open,High,Low,Close,Volume & Return contributes to the prediction of stock price of some chosen organization using machine learning concepts like decision tree.

1.1.3 Approach Used

1. Load 2 years of historical data of chosen stock.
2. Decision Tree Classification On OHLC(Open, High, Low, Close) Of Data.
3. Predict The Next Day's Buy Or Sell Precision Score.
4. Customized Feature Engineering & tool automation.

In Figure 1.1 I show the flow of the model

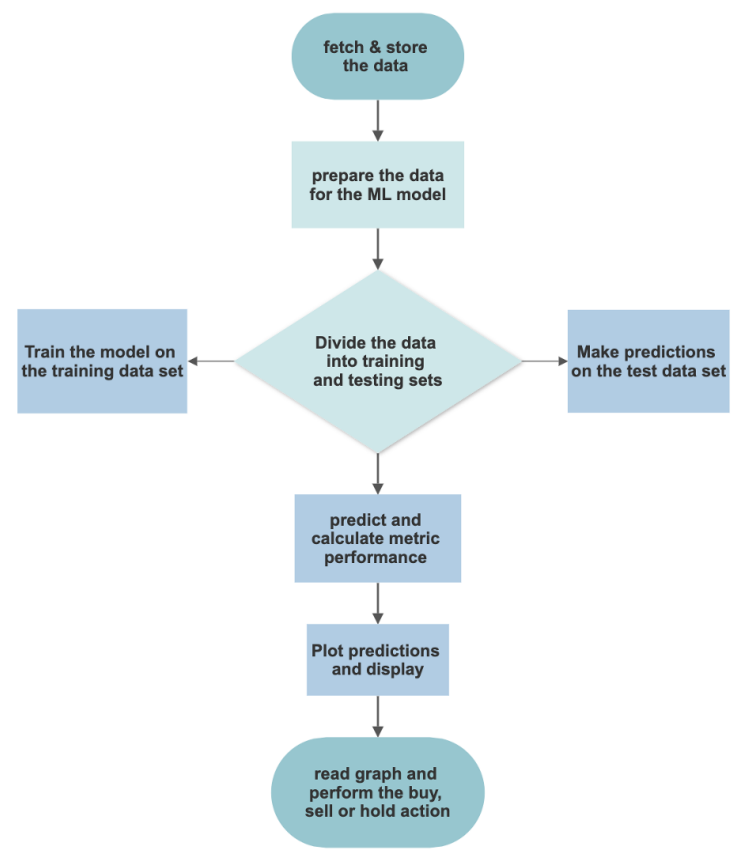

Figure 1.1: Basic Structure
[25]

Chapter 2

Literature Review

Investors are motivated by profit, and stock markets are a hugely profitable and popular means to grow capital. Profit-enhancing factors include artificial intelligence and recommendations for the top businesses in this industry. In accordance with the share price, closing price, opening price, and share value. Several studies have been suggested in these areas to help the investor choose the best and most profitable industry for investment in the year.
In order to make financial and monetary gains, stock market investors attempt to forecast future stock values. Different factors influence how each investor chooses to purchase a stock [7].

Halbert White carried out the first important investigation into neural network models for stock price forecasting in 1988. His model was highly pessimistic during training because it was based on IBM's daily common stock. The accuracy of the neural networks' stock market forecasting predictions was then further investigated[14].

The stock market forecast is made using the random forest method. Given that it is one of the most user-friendly and adaptable machine learning algorithms, it provides good prediction accuracy. Typically, this is applied to categorization tasks. Predicting is a difficult task because of the stock market's extreme volatility. We use a random forest classifier, which has the same hyperparameters as a decision tree, to predict the stock market.

In order to develop a trading strategy or choose the right time to buy or sell a stock, stock price forecasting is crucial. In order to anticipate stock prices, some work using a model called the feature fusion long short-term memory-convolutional neural network (LSTM-CNN) model has also been done that integrates features discovered from several representations of the same data, namely stock time series and stock chart images. A CNN and LSTM are used in the proposed model to extract image and temporal data, respectively. Using data from the SPDR S&P 500 ETF, they compare the performance of the proposed model to that of single models (CNN and LSTM). In terms of predicting stock values, their feature fusion LSTM-CNN model performs better than the standalone models. Furthermore, it was discovered that a candlestick chart is the best stock chart illustration to utilise when predicting stock values. As a result, this study demonstrates that, rather than employing these

features independently, combining temporal and image information from the same data can effectively reduce prediction error[17].

The XGBoost algorithm is used to forecast stock prices. After doing sentiment analysis, we integrate the results with the most recent and in-demand algorithm to analyse stock data and forecast stock price. The most potent machine learning algorithm available right now might be XGBoost. Gradient boosted trees, or XGBoost, is the name of a massive machine learning technique with numerous components. Keep in mind that boosting is an ensemble method. The missing values can be handled automatically by XGBoost. Overfitting is avoided or regularised by boosting. The XGBoost algorithm has a number of characteristics, including multiprocessing and tree pruning [16].

Support vector machines, in addition to or instead of ANNs, have been used in recent research in the field. SVMs may create classification errors inside training data in order to reduce total error throughout test data, in contrast to ANNs, which are models that aim to minimise classification error within the training data [18].

The Experimental Setup algorithm was utilized in one of the studies, and it's important to note that the GRU and LSTM models' prediction capabilities are significantly impacted by different super parameters. The LSTM is a type of RNN that recycles the weight parameters of neurons and effectively utilizes past prediction data. However, the RNN has the potential for gradient explosion and disappearance, relying on historical data for both long and short periods of time. The study aimed to compare prediction results by using different super parameter data and integrating a variety of technical indicators such as financial data and investor sentiment indicators. Additionally, the LASSO and PCA analysis approaches were employed to reduce the dimensions of the various influencing factors of the extracted stock price[9].

Chapter 3

Objective & Methodology

Hypothesis

This project is aimed at developing a model that can predict the stock prices of a selected organization for a specified time frame along with constructing tools that can prove to be useful for investors and traders in the market.
The prediction of stock prices is a challenging task because it depends on various factors, such as economic conditions, company financials, market trends, and investor sentiment. To predict the stock price, the model would need to analyze historical stock price data and other relevant factors that could affect stock prices.
The project may involve using various machine learning algorithms, such as regression analysis, time-series analysis, and deep learning techniques, to build a predictive model. The model would likely need to be trained on historical stock price data, financial statements, news articles, and other relevant data sources.
It's important to note that while stock price predictions and insights can be helpful, they are not guaranteed to be accurate. The stock market can be volatile and influenced by many unpredictable factors, and no model can predict the future or give insights about entry/exit points with complete accuracy due to uncontrollable events. Nonetheless, with careful analysis and the use of sophisticated techniques, it is possible to develop a model that can provide reasonably accurate predictions of stock prices for a specific organization over a given time frame.

Plan Of Work

The final step of this project is to use the tools automated for making predictions about stock prices and using the decision tree algorithm to give some predictions based on the precision score that whether a person should buy/sell the stock.
Both Random Forests and XGBoost are popular machine learning algorithms that are widely used for regression and classification tasks.
Random Forests is an ensemble learning method that combines multiple decision trees to make a prediction. It works by creating a set of decision trees on random subsets of the

training data, and then combining the predictions of each tree to obtain a final prediction. Random Forests are known to be robust and less prone to overfitting, making them a popular choice for regression tasks.
XGBoost, on the other hand, is an optimized implementation of gradient boosting. Gradient boosting is another ensemble learning method that works by iteratively adding weak models to the ensemble and adjusting their weights based on their performance. XGBoost is known for its high accuracy and efficiency and is commonly used in machine learning competitions.
Overall, by using machine learning methods like Random Forests and XGBoost, this project aims to provide accurate predictions of stock prices for a chosen organization over a desired time frame. However, as with any stock price prediction model, the predictions are subject to uncertainty and should be used with caution.

3.1 Methodology

The aim of this project is to provide the investors in the stock market a statistical tool that can give results with adequate results upon which they can make their decisions based on analytical values and without much effort whether to buy or sell at a certain point of time in the market.
I have used quantitative research methods which produces objective data that can be clearly communicated through statistics and numbers to make effortless predictions[24].
In order to accomplish the project's objectives, I have employed novel mathematical ideas like Hausdorff distance and the concept of developing customised attributes for the random forest approach, both of which I could not find anywhere else.
The data which I used for this project was collected from websites such as Yahoo Finance and python libraries such as pandas datareader and nsepy.
And to determine the accuracy of the results we obtained, the next step can be to use machine learning algorithm in the software Anaconda navigator using Python programming language.

Some of the obstacles I faced while constructing customised attributes were dealing with huge amounts of data and the hardest challenge was implementing and automating the ideas that traders and investors use manually to study the chart and make decisions accordingly on python.

Chapter 4

Technical Analysis

4.1 Introduction

Technical analysis is the study of price movements in a market, whereby traders make use of historic chart patterns and indicators to predict future trends in the market. It is a visual representation of the past and present performance of a market and allows the trader to use this information in the form of price action, indicators and patterns to guide and inform future trends before entering a trade.

The below chart is an example of a chart with the use of the MACD and RSI indicator: In Figure 4.1 an example of Candlestick & Indicators in the chart is shown.

Figure 4.1: Candlestick & Indicators
[6]

Benefits of using technical analysis include the following

- Can be applied to any market using any timeframe
- Technical analysis can be used as a standalone method
- Allows traders to identify trends in the market

4.2 Charts in Technical Analysis

The overall trend, whether it be upward or negative, long-term or short-term, as well as range bound situations, can be determined using charts. Line charts, bar charts, and candlestick charts are the three most used types of technical analysis charts.

When using a bar or candlestick chart, the technical analyst will receive data on the price from the opening, the high or low of the period, and the close for each period. Candlestick analysis is particularly helpful since the patterns and relationships found within them can help with forecasting the price's future course.

4.3 Technical Analysis Indicator

Technical traders employ indicators to search the market for possibilities. Despite the fact that there are numerous indicators, traders frequently use volume and price-based indicators. These help identify the locations of the levels of support and resistance, how frequently they are maintained or crossed, and how long a trend has been going.

A trader can view the price or any other indicator utilising multiple time frame analysis, which ranges from one second to a month and provides the trader with a distinct viewpoint of the price action.

The more popular indicators for technical analysis include:

- Moving averages
- Relative strength index (RSI)
- Moving average convergence divergence (MACD)

Moving averages and the MACD are frequently used to spot market trends, while the RSI is frequently utilised to pinpoint potential entry and exit points. Using indicators can help traders analyse the market, verify trade setups, and pinpoint entry locations [6].

4.4 Support & Resistance

4.4.1 Support

In a downtrend, prices fall because there is an excess of supply over demand. The lower prices go, the more attractive prices become to those waiting on the sidelines to buy the shares. At some level, demand that would have been slowly increasing will rise to the level where it matches supply. At this point, prices will stop falling. This is support.

Support can be a price zone or a price level on the chart. In any case, support is a region on a price chart that displays the readiness of buyers to purchase. Demand will typically outpace supply at this point, halting the price decrease, and starting it back up again.

4.4.2 Resistance

The opposite of support is resistance. Because there is a greater demand than there is supply, prices rise. There will come a time as prices rise when the desire to sell will outweigh the desire to acquire. There are several reasons why this occurs. It's possible that traders have decided that the price is too high or that their goal has been reached. It might be that purchasers are reluctant to start new positions at such high valuations. However, a technician can easily identify the point on a price chart where supply starts to outweigh demand. This is opposition. It might be a level or a zone, just like support [20].

4.5 Zones

On a chart, a zone is a location where supply and demand are out of balance and is identified by two horizontal lines surrounding the price level [**?**][23].

- **Distal**: farthest from current price
- **Proximal**: closest to current price.

Steps to identify demand zone:

1. Start with current price on chart.
2. Look left and down until you find the origin of a very strong rally in price (open of last bearish candle).
3. Draw proximal and distal lines from the origin of the rally and extend them forward.
4. look for large green candles (4 candles or less for demand).
5. Make sure the zone is fresh.

In Figure 4.2 the steps to identify a demand zone is depicted.

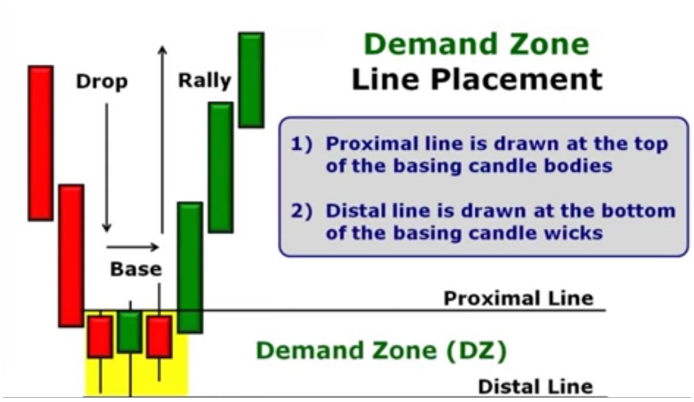

Figure 4.2: Demand Zone
[20]

Steps to identify supply zone:

1. Start with current price on chart.
2. Look left and up until you find the origin of a very strong drop in price (open of last bullish candle).
3. Draw proximal and distal lines from the origin of the drop and extend them forward.
4. Look for large red candles (4 candles or less for supply).
5. Make sure the zone is fresh.

In Figure 4.3 the steps to identify a supply zone is depicted.

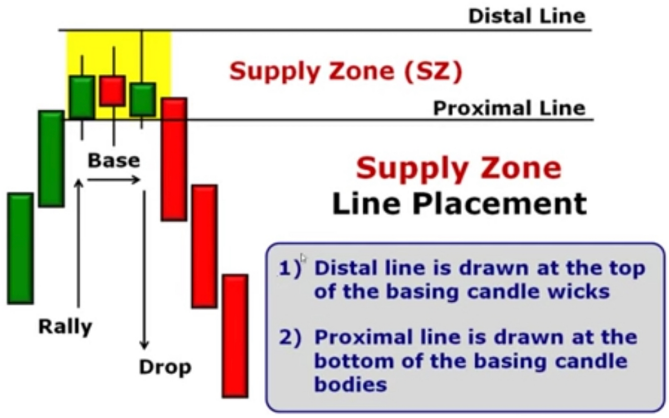

Figure 4.3: Supply Zone
[20]

4.6 Chart Patterns

4.6.1 Head & Shoulders Formation

In technical analysis, a head and shoulders pattern is employed. A unique chart pattern that signals a change in trend from bullish to bearish. The pattern appears as a baseline with three peaks, the centre peak being the highest and the outside two peaks being relatively close in height.
The head and shoulders pattern appears when the price of a stock increases to its top and then drops back to where it started its uptrend. The price then increases above the prior high to create the "head" before falling back to the initial base. Finally, the stock price experiences a second peak at a level close to the formation's first peak before declining once more.
One of the most effective patterns for trend reversal is the head and shoulders formation. It is one of a number of top designs that, to varied degrees, indicate the end of an upward trend.

How Does The Pattern Work ?

1. The price climbs to a peak after prolonged bullish trends, then falls to form a shoulder like pattern.
2. The price then decreases after rising once again to establish a high that is significantly higher than the previous peak(shoulder).
3. A third increase in price only brings it to the first high level before it starts to fall once more.
4. The neckline is indicated by two peak or trough points (inverse).

In Figure 4.6 an example of Head & Shoulders formation is depicted.

Figure 4.4: Head & Shoulders Formation
[13]

The shoulders are formed by the first and third peaks, and the head is formed by the second peak. The neckline is the line that connects the first and second troughs.

4.6.2 Inverse Head & Shoulders Formation

The inverse head and shoulders chart, also known as a head and shoulders bottom, is the inverse of a head and shoulders chart. It is reversed, with the head and shoulders bottoms used to predict downtrend reversals. This pattern is defined when a security's price movement exhibits the following characteristics:

1. The price drops to a low and then rises.
2. The price falls below the previous trough before rising again.
3. The price falls once more, but not as much as the previous trough.
4. After reaching the final trough, the price begins to rise towards the resistance (the neckline) found near the top of the prior troughs.

The second trough is the deepest (the head), while the first and third are shallower (the shoulders). The last rally following the third dip indicates that the bearish trend has reversed, and prices are likely to continue rising.

What Does the Head and Shoulders Pattern Indicate?
The head and shoulders pattern suggests a probable reversal. Traders believe that three sets of peaks and troughs, with a greater peak in the centre, indicate that the price of a stock

will begin to plummet. The neckline denotes the point at which bearish traders begin to sell.
The pattern also suggests that the new downward trend will most likely continue until the right shoulder is broken, at which point prices will go higher than at the right peak[13].

4.7 Candlestick Patterns

Candlestick charts display data from multiple time frames in a single price bar. In this way, they are more useful than traditional OHLC bars or simple lines connecting closing prices. Once completed, candlestick patterns may predict price direction. Originally used by Japanese rice traders in the 18th century, color coding adds depth to this colorful technical tool[21].

How To Read A Candlestick Pattern?
A daily candlestick represents a market's opening, high, low, and closing (OHLC) prices. The rectangular real body, or just body, is colored with a dark color (red or black) for a drop in price and a light color (green or white) for a price increase. The lines above and below the body are referred to as wicks or tails, and they represent the day's maximum high and low. Taken together, the parts of the candlestick can frequently signal changes in a market's direction or highlight significant potential moves that frequently must be confirmed by the next day's candle[8].

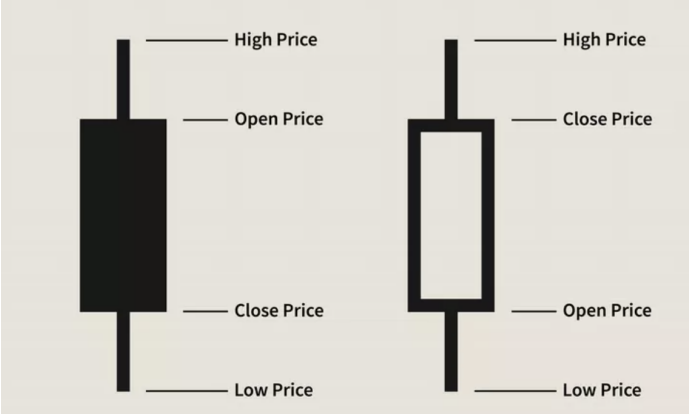

Figure 4.5: OHLC Of A Candle
[21]

4.8 Trend Channel

A trend channel is a collection of parallel trend lines that are determined by the highest and lowest points of the price movement of an asset. When the price is shifting between two parallel trend lines, it forms a trend channel, which is also sometimes referred to as a price channel. On charts, trend channels can be constructed to help identify upward and downward trends.

Figure 4.6: Sample Trend Channel
[1]

The trend channel line is placed above the highs of the price movement in a bullish trend, while the trend line is plotted below the price movement. The trend channel line is below the price action in a negative trend, while the trend line is displayed above. Since they function as a support level in a bullish trend and a resistance level in a bearish trend, trend lines are typically employed as entry signals. When the price tests the trend line without crossing it, an entry signal is generated. When this occurs, a trade is opened with a stop loss exactly below the trend line and is entered in the direction of the trend. A price target for taking profits can be found at the opposing trend line. For instance, traders can consider the indication of the lower trend line in an ascending channel as an entry point for a trade against the trend. With a stop loss slightly below the lower trend line, a long trade is started. The profit goal can be the upper trend line. Since they function as a resistance in up-trends and a support in downtrends, trend channel lines most frequently produce counter-trend entry signals. For instance, aggressive traders may utilise the test of the upper trend line in an ascending channel as an entry point for a trade that is the opposite of the trend (counter-trend). With a stop loss positioned just above the upper trend line, a short trade is started. The profit goal can be the lower trend line[1].

Chapter 5

What Is Machine Learning ?

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms that can learn from and make predictions on data. The main goal of machine learning is to enable computers to learn and improve their performance over time without being explicitly programmed to do so. This is achieved through the use of mathematical models that can recognize patterns in data and make predictions based on those patterns.
There are three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training a model on labeled data, where the correct output is already known. Unsupervised learning involves training a model on unlabeled data, where the correct output is unknown. Reinforcement learning involves training a model to make decisions based on feedback received from its environment.
Machine learning has many applications in various fields, including computer vision, natural language processing, and data analysis. In computer vision, machine learning algorithms can be used to recognize objects in images and videos. In natural language processing, machine learning algorithms can be used to understand and generate human language. In data analysis, machine learning algorithms can be used to discover patterns and insights in large datasets.
Machine learning is a rapidly growing field that has the potential to revolutionize many industries. However, it also presents many challenges, such as the need for large amounts of data and the risk of bias in the models. As such, it is important to approach machine learning with caution and to continually monitor and improve the models to ensure their accuracy and fairness.

Chapter 6

Decision Tree

6.1 Introduction

The decision tree is used to draw conclusions from the provided dataset, as suggested by its name. The idea behind the decision tree is that it aids in choosing suitable features for subdividing the tree into portions. The splitting method utilised is ID3. The decision tree will have less depth if it is well-built or more depth if not. In Figure 6.1 a decision tree is depicted being split on an attribute.

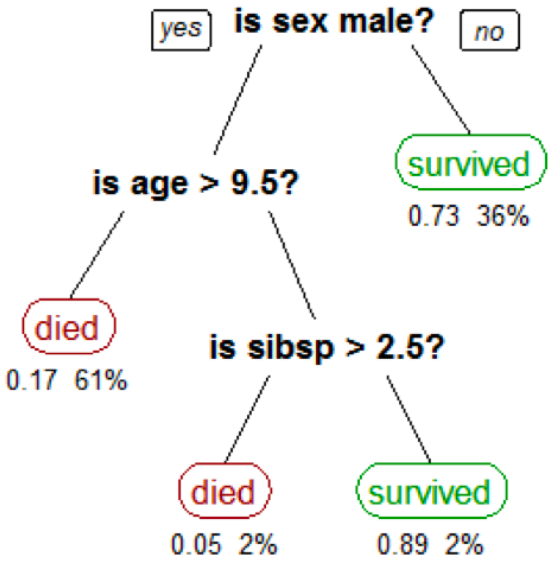

Figure 6.1: A Sample Of Decision Tree
[12]

An upside-down decision tree is depicted in the above image, with the root at the top. A

condition or internal node, upon which the tree divides into branches and edges, is represented by the bold text in black. The choice or leaf, in this case, whether the passenger died or lived, is shown as red and green text, respectively, at the end of the branch that doesn't split any longer.

The feature importance is clear, and relationships are simple to see. This process is more usually referred to as learning decision trees from data, and the tree above is known as a classification tree because the goal is to categorise passengers as having survived or not. The only difference is that regression trees forecast continuous quantities, such as the price of a house. CART, or Classification and Regression Trees, is the common abbreviation for decision tree algorithms. Making choices on which features to employ, what conditions to use for splitting, and when to quit growing a tree.

6.1.1 Gini Impurity

To identify how the attributes of a dataset should split the nodes to construct the tree, decision trees are built using the Gini Impurity measurement. A dataset's Gini Impurity, which is actually a number between $0 - 0.5$, represents how likely it is that new, random data will be incorrectly classified if it is assigned a random class label based on the dataset's class distribution.

Consider the scenario where you want to create a classifier that predicts whether a user will miss a credit card payment. You have some labelled data with characteristics like the person's age, income, credit score, and whether or not they are currently a student or not.

You may determine how poorly each feature divided the data into the appropriate class, defaulted ("yes") or didn't default("no") in order to determine the best feature for the first split of the tree, the root node. The feature with the lowest impurity would be chosen as the optimal feature for splitting the current node after this algorithm measured the impurity of the split. Each other node would go through the same procedure using the remaining features. In Figure 6.2 the gini impurity is depicted when the tree is split on these attributes.

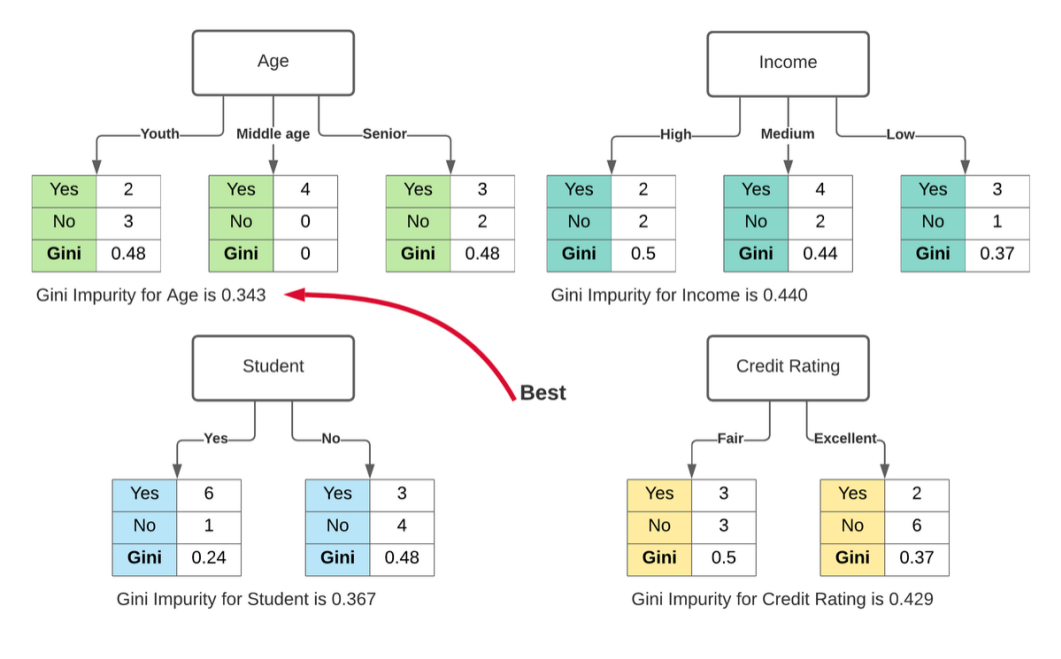

Figure 6.2: Gini Impurity
[15]

In the image above, age has lowest Gini impurity, so age is chosen as the root of the decision tree.

6.1.2 Mathematical Definition of Gini Impurity

Consider a data-set D that contains samples from k classes. The probability of samples belonging to class i at a given node can be denoted as p_i. Then the Gini Impurity of D is defined as:

$$\text{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2 \tag{6.1}$$

The node with uniform class distribution has the highest impurity. The minimum impurity is obtained when all records belong to the same class. Several examples are given in the following table to demonstrate the Gini Impurity computation.

	Count		Probability		Gini Impurity
	n_1	n_2	p_1	p_2	$1 - p_1^2 - p_2^2$
Node A	0	10	0	1	$1 - 0^2 - 1^2 = 0$
Node B	3	7	0.3	0.7	$1 - 0.3^2 - 0.7^2 = 0.42$
Node C	5	5	0.5	0.5	$1 - 0.5^2 - 0.5^2 = 0.5$

An attribute with the smallest Gini Impurity is selected for splitting the node.

If a data set D is split on an attribute A into two subsets D_1 and D_2 with sizes n_1 and n_2, respectively, the Gini Impurity can be defined as:

$$\text{Gini}_A(D) = \frac{n_1}{n}\text{Gini}(D_1) + \frac{n_2}{n}\text{Gini}(D_2) \tag{6.2}$$

When training a decision tree, the attribute that provides the smallest $\text{Gini}_A(D)$ is chosen to split the node[15].

Chapter 7

Random Forest Method

7.1 Introduction

Random forest is a supervised machine learning approach based on ensemble learning. In order to create a more effective prediction model, you can combine several kinds of algorithms or use the same technique more than once in ensemble learning to have more accurate prediction. The name "Random Forest" comes from the fact that the random forest method combines several algorithms of the same type, or different decision trees, into a forest of trees. Both regression and classification tasks can be performed using the random forest approach. In Figure 7.1 the steps to form a random forest is depicted.

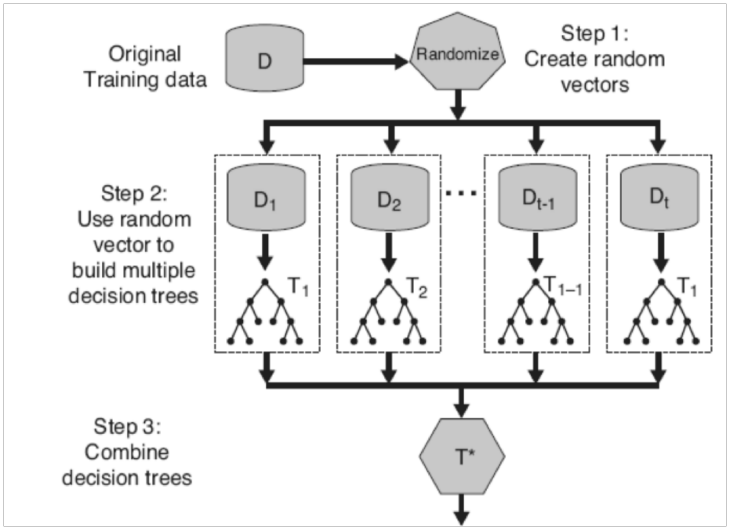

Figure 7.1: Random Forest
[2]

7.2 The Algorithm

The following are the basic steps involved in performing the random forest algorithm:

1. Pick N random datapoints from the data-set.
2. Build a decision tree based on these N records.
3. Decide the number of trees we want in our algorithm and repeat steps 1 and 2.
4. In case of a **regression problem**, for a new datapoint, each tree in the forest predicts a value for Y (output). Then the final value is calculated by finding the average of all the values predicted by all the trees in forest. In case of a **classification problem**, each tree in the forest predicts the class to which the new datapoint belongs. Finally, the new datapoint is assigned to the class that wins the majority vote.

7.2.1 Advantages of the Method

- Since there are several trees and each tree is trained on a portion of data, the random forest approach is not biased. In essence, the random forest algorithm relies on "the crowd's" power, which reduces the system's overall bias.
- The algorithm is remarkably reliable. Since it is exceedingly difficult for fresh data to have an impact on all the trees, even if a new data point is added to the data-set, the overall method is not significantly changed.
- In situations when there are both categorical and numerical features, the random forest approach performs well.
- When data is missing values or not scaled properly, the random forest technique also performs well.

7.3 Disadvantages of the Method

- The complexity of random forests is a significant drawback. Because so many decision trees were combined, they required significantly more computer power.
- They take a lot longer to train than other equivalent algorithms because of their intricacy.

Chapter 8

XGBoost

Conceptually, XGBoost is a gradient boosting algorithm that combines multiple weak decision trees into a strong predictor. In XGBoost, each decision tree is trained on the residuals (the differences between the predicted and actual values) of the previous trees, which allows the algorithm to focus on the data points that are difficult to predict.
The training process involves iteratively adding new trees to the ensemble, with each new tree designed to correct the errors of the previous trees. At each iteration, the algorithm calculates the gradients and Hessians of the loss function with respect to the predicted values, and fits a new decision tree to the negative gradients (which correspond to the residuals) using a technique called gradient boosting.
XGBoost also includes several advanced features to improve performance and prevent overfitting, such as regularization, early stopping, and tree pruning. Additionally, it supports various objective functions and evaluation metrics for classification, regression, and ranking problems, and allows for parallel processing and distributed computing.
Overall, XGBoost is a powerful and flexible algorithm that can be used for a wide range of machine learning tasks, and is capable of achieving state-of-the-art performance on many benchmark datasets.
It's effective for handling of missing values, which enables it to handle real-world data with missing values without requiring a lot of pre-processing, is one of the key characteristics of XGBoost. Additionally, XGBoost includes built-in parallel processing capabilities, enabling the training of models[10].

Chapter 9

Mathematical Background

9.1 Hausdorff Distance

The Hausdorff distance quantifies the distance between two subsets of a metric space. It is formally defined as the largest possible distance between a point in one set and its nearest neighbour in the opposite set. Two sets are "close" if for any one point on either set, the nearest point in the other set is "not too far" [3].
In other words it is the *maximum distance of a set to the nearest point in the other set.*
Hausdorff distance between sets A and B is a maximin function, which is defined as:

$$h(A, B) = \max_{a \in A}\{\min_{b \in B} d(a, b)\} \tag{9.1}$$

where a and b are points belonging to sets A and B respectively, and $d(a, b)$ is any metric between these points; for simplicity, we'll take $d(a, b)$ as the Euclidean distance between a and b [11].
In Figure 9.1 the algorithm to find the hausdorff distance between any two sets is explained.

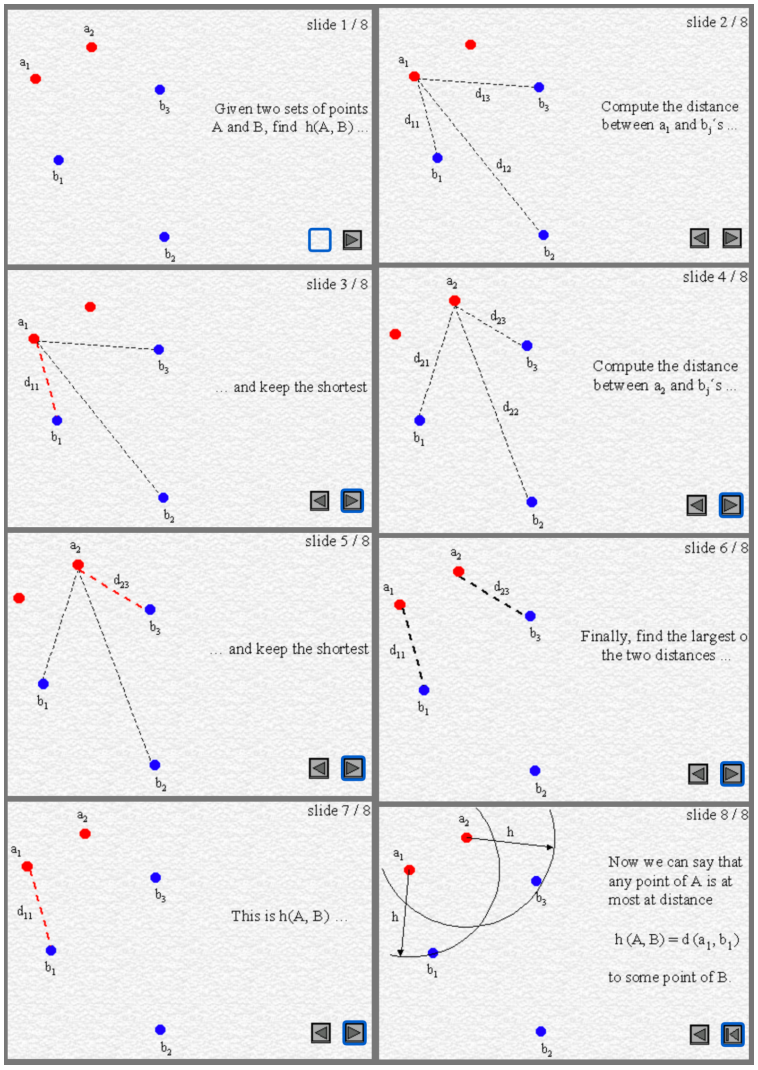

Figure 9.1: Hausdorff distance on point sets.

9.2 Fréchet Distance

Intution:
Imagine walking your dog on a leash down a finite curved path, with the dog travelling a separate finite curved path. Although neither can travel backward, they can both change their speed to maintain slack in the leash. The shortest leash needed for both to travel their own paths from beginning to end is the Fréchet distance between the two curves. Because the Fréchet distance would be the same if the dog were walking its person, it is important to note that the definition is symmetric with regard to the two curves.

What Is Fréchet Distance ?
It is frequently helpful to approximate a curve by a polygonal curve since it might be challenging to conduct mathematical operations on a curve of any shape in particular situations. A continuous piecewise linear curve made up of N connected segments is referred to as a

polygonal curve $P : [0, N]$. As seen in the following image, such a curve can be parametrized with a parameter a such that $P(a)$ corresponds to a specific location on the curve, with $P(0)$ being the polygonal curve's start vertex and $P(N)$ denoting its last vertex.
In Figure 9.2 the parametrization of a polygonal curve is depicted.

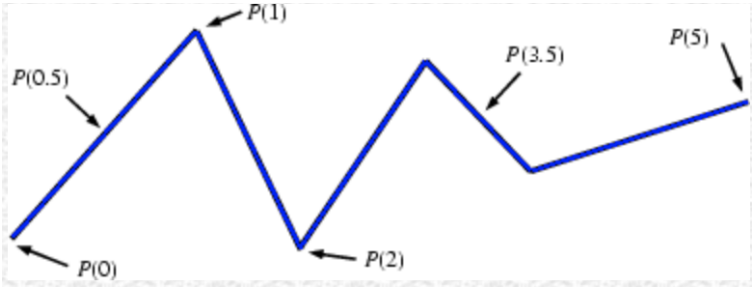

Figure 9.2: Parametrization of a polygonal curve.

Consider the situation where the man from the previous example is required to walk on a polygonal curve P of length N and his dog is required to walk on a polygonal curve Q of length M. A function of t can be used to describe the location of the man and the dog on their respective curves by using two continuous and growing functions, $\alpha(t)$ and $\beta(t)$, where $\alpha(0) = 0$, $\alpha(1) = N$, $\beta(0) = 0$, and $\beta(1) = M$. $P(\alpha(t))$ and $Q(\beta(t))$ are functions of time that describe the position of the man and the dog, respectively. While numerous functions $\alpha(t)$ and $\beta(t)$ can be employed, some of them will result in the dog and the man being separated by a large distance at times.

Mathematically, the Fréchet distance between two curves is defined as:

$$\delta_F(P, Q) = \min_{\alpha[0,1]\to[0,N]} \{ \max_{\beta[0,1]\to[0,M]} d((P(\alpha(t)), Q(\beta(t))))\} \tag{9.2}$$

where $\alpha(t)$ and $\beta(t)$ range over continuous and increasing functions with $\alpha(0) = 0$, $\alpha(1) = N$, $\beta(0) = 0$ and $\beta(1) = M$ only.
This equation explains: for every possible function $\alpha(t)$ and $\beta(t)$, find the largest distance between the man and its dog as they walk along their respective path; finally, keep the smallest distance found among these maximum distances [22].

9.3 Ray Shooting Problem In Polygons

Ray shooting is the problem of determining the first intersection of a ray with a set of obstacles.Due to its significant applications in fields like ray tracing and Monte Carlo techniques to radiosity in computer graphics, and motion tracking in virtual reality applications, this is a basic topic in computational geometry. Preprocessing the obstacles will make query processing as efficient as feasible since several ray shooting queries can be executed to a certain set of obstacles [19].

Chapter 10

Idea Behind Attributes Created

10.1 Attribute 1: General Pattern Recognition

In this feature we have first created a windowed data of the closing price of an index fund in the NSE which can give us the idea about the closing prices of top companies, but starting from that date where we could initially manually identify possible demand or supply zone pattern or a possible head & shoulders pattern or any pattern of the investor's choice and of the window-length based on the requirements of the trader/investor based on whether the trade is long-term or short-term or an intraday trade. Then using L1 distance or hausdorff distance I calculated the approximate distance(similarity) between the original closing price data and the windowed closing price data which will give us an idea if that particular pattern which we traced by observing demand/supply zone or the head & shoulders pattern have been observed before in the entire data or is that pattern fresh or not and if it is not fresh then how many times has it been observed before where our ray-shooting concept come into the picture. Then by concatenating the L1 & Hausdorff distances to the original dataset and converting these distances into a feature and plugging it into the algorithm of a decision tree we can split the tree based on these distances to give a more accurate prediction of the share price of any stock.
And the aim of detecting these patterns will be to conclude that whether the price will fall or rise since patterns like Head and Shoulders will indicate the downfall of the price but the inverse Head and Shoulder will indicate rise in the price level.
So, in our algorithm we are first creating a window function which will produce that window within which we identified our desired pattern or between the price levels in which our pattern occurred. Below is the pseudo-code to give an idea.

```
FUNCTION window(i, j):
    // This function returns a sub-series of the 'close' series.
    // The sub-series starts at index 'i' and ends at index 'i+j-1'.

    // Select the sub-series from 'close' using the indices 'i' and 'i+j-1'.
    sub_series = close.iloc[i:i+j]

```

```
    // Return the sub-series.
    RETURN sub_series
END FUNCTION
```

Then to calculate the L1 distance between the windowed data (where we identify our pattern) and the original data of 2 years observed within that window length starting from the 1^{st} closing date of the original data of our chosen training stock. Below is the pseudo-code to give an idea.

```
FUNCTION dist_L1_1(arr1, arr2):
    // This function calculates the L1 distance between 'arr1' and 'arr2'.
    // 'arr1' is a sub-series of the 'close' series, and 'arr2' is a fixed
    ↪ array.
    // The L1 distance is calculated over a sliding window of length
    ↪ 'window_length'.
    // The distance between 'arr1' and 'arr2' is computed using the absolute
    ↪ difference.

    // Initialize an empty list to store the distances.
    dwocp = []

    // Iterate over a range of indices from 0 to 'len(close) - window_length'.
    for i in range(len(close) - window_length):
        // Select a sub-series of length 'window_length' from 'arr1' starting
        ↪ from index 'i'.
        arr = arr1[i:i+window_length]

        // Compute the L1 distance between 'arr' and 'arr2' using the absolute
        ↪ difference.
        distance = sum(abs(arr2 - arr))

        // Append the distance to the list of distances.
        dwocp.append(distance)

    // Return the list of distances.
    RETURN dwocp
END FUNCTION
```

We are basically rolling our windowed data where the pattern occurred across the entire original data of 2 years to observe the if our manually detected pattern has been observed more than once or not because forming of the pattern we choose will determine if our price will fall or rise and now finally we find the distance using two methods. The second method for distance calculation we are using is the Hausdorff distance. For the Hausdorff distance we converted our dataset of closing price and date into and array of 2-tuple of the form: (Close price,Day number) where, the date is converted into the day number using the famous Zeller's algorithm and our day number is given as:

1. 0: Saturday
2. 1: Sunday
3. 2: Monday
4. 3: Tuesday
5. 4: Wednesday
6. 5: Thursday
7. 6: Friday

where,

1. d: Date
2. m: Month
3. y: Year

and we do, y = y(modulo 100) and c = y/100. Below is the pseudo-code to give an idea.

```
FUNCTION dow1(str1):
    // This function takes a date in the format 'dd/mm/yy' and returns the day
    ↪  of the week.
    // The day of the week is calculated using Zeller's algorithm.

    // Extract the day, month, and year from the input string.
    d = int(str1[0:2])
    m = int(str1[3:5])
    y = int('20' + str1[6:])

    // Adjust the month and year if necessary for Zeller's algorithm.
    if m == 1:
        y = y - 1
        m = 13
    elif m == 2:
        y = y - 1
        m = 14

    // Calculate the day of the week using Zeller's algorithm.
    y = y % 100
    c = y // 100
    dow2 = ((d + 13 * (m + 1) // 5) + y + (y // 4) + (c // 4) + 5 * c) % 7

    // Return the day of the week.
    RETURN dow2
END FUNCTION
```

```
// Example usage:
day_of_week = dow1('01/12/22')
IF day_of_week == 0:
    PRINT "Saturday"
ELSE IF day_of_week == 1:
    PRINT "Sunday"
ELSE IF day_of_week == 2:
    PRINT "Monday"
ELSE IF day_of_week == 3:
    PRINT "Tuesday"
ELSE IF day_of_week == 4:
    PRINT "Wednesday"
ELSE IF day_of_week == 5:
    PRINT "Thursday"
ELSE IF day_of_week == 6:
    PRINT "Friday"
```

We converted our closing price data into an array of 2-tuple as mentioned above since the Hausdorff distance is only calculated between two 2D arrays and the distance between pairs of data are calculated using an euclidean metric. Below is the pseudo-code to give an idea.

```
FUNCTION dist_HD(arr, arrw1):
    // This function calculates the Hausdorff distance between a given array
    ↪  of closing prices
    // and a window of closing prices for a particular stock.

    // Initialize an empty list to hold the Hausdorff distances.
    Hdwocp = []

    // Loop over each possible window of closing prices.
    FOR i = 0 TO len(close) - window_length DO:
        // Extract the window of closing prices for this iteration.
        arr1 = arr[i:i+window_length]

        // Calculate the directed Hausdorff distance between the two arrays.
        dist = directed_hausdorff(arr1, arrw1)[0]

        // Add the distance to the list of Hausdorff distances.
        Hdwocp.append(dist)

    // Return the list of Hausdorff distances.
    RETURN Hdwocp
END FUNCTION
```

Now, after the calculation of Hausdorff distance between any one of the windowed data(pattern) and the original data and if the distance is within some threshold value (say 0.05), then we can say that the chosen pattern was observed before also in the two years of original closing price data.

10.2 Attribute 2: Candlestick Pattern Detection

In this feature using the TA-lib library I have first identified starting & ending date as the input then producing those dates on which any specific chosen candlestick pattern was observed, for example: Doji,Hammer,Morningstar etc. And our algorithm gives us the following outputs:

1. 0: No detection of the chosen candlestick pattern on that date.
2. 100: The desired candlestick pattern(BULLISH) was detected on that date which will indicate about potential rise or fall in price.
3. -100: The desired candlestick pattern(BEARISH) was detected on that date which will indicate about potential rise or fall in price.

This feature can also make our predictions more accurate & robust.

10.3 Attribute 3: Trend Line Channel Detection

In this feature we are first positioning our focus at a certain candle or a certain date and then searching for desired number of back-candles(say 30 days of candles), then we are basically dividing this time-frame of back-candles into some equal parts(say 5 candles) and in each of these time-frames of 5 candles we will find the maxima and the minima which are the highest and lowest points the price will be touching and we are doing this for all the equal windows of that time-frame after that we are looking for all the minima's and maxima's and fitting those points into a polynomial or linear fit that goes across almost all those points and these polynomials will give us 2 slopes describing the approximate trend of the minima and the maxima points and our slopes will be of one degree, since we are not dealing with curves but rather with straight lines.
The final step to implement above algorithm we will adjust and modify our intercept value of our 2 slopes in such a way so that our maxima and minima slope can wrap around the price levels and not just go through those points. To achieve this we modified the intercept which will enable our slopes to go through the highest maxima point and the lowest minima point that could be found in our windowed data[5].

10.4 Attribute 4: Support & Resistance Detection

By marking the support & resistance levels within a specified window, and here each window represents the number of candles before & after the candle taken into consideration or from

where we can observe possible support or resistance level. In this feature we are first defining 3 variables or paramenters namely:

1. L: Candle of interest i.e; the candle which will indicate the support/resistance level.
2. $n1$: Number of candles before candle L
3. $n2$: Number of candles after candle L

To Detect Support Levels:
We will check if the low values of candles before candle L are in a decreasing order and if not then we will break out of this function and return the value as 0. Similarly, if the low values of candles after candle L are in an increasing order and if not then again we break out of the function and return the value 0. And we return the value 1 when we get the setup of a support in our data, i.e; if we did not break out of the above mentioned conditions in our data.

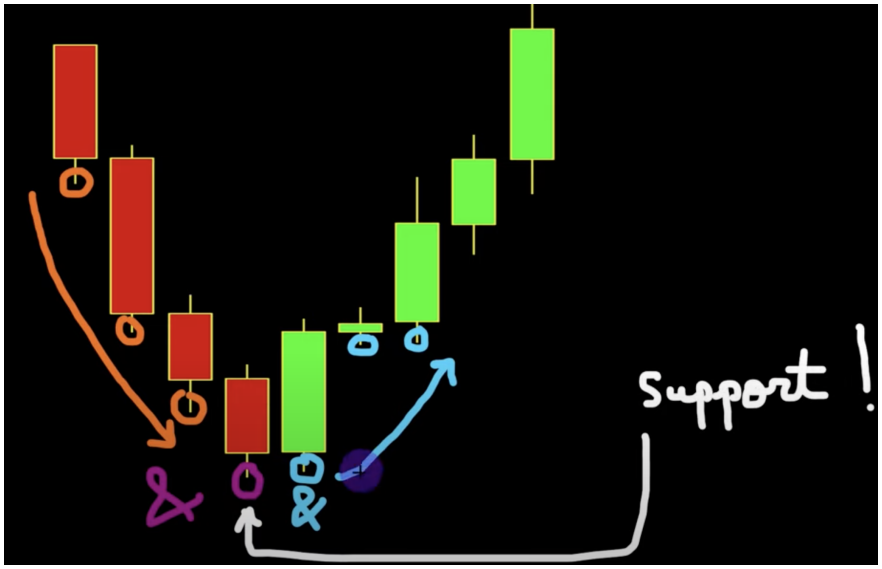

Figure 10.1: Support.
[4]

To Detect Resistance Levels:
We will check if the high values of candles before candle L are in an increasing order and if not then we will break out of this function and return the value as 0. Similarly, if the high values of candles after candle L are in a decreasing order and if not then again we break out of the function and return the value 0. And we return the value 1 when we get the setup of a resistance in our data, i.e; if we did not break out of the above mentioned conditions in our data.

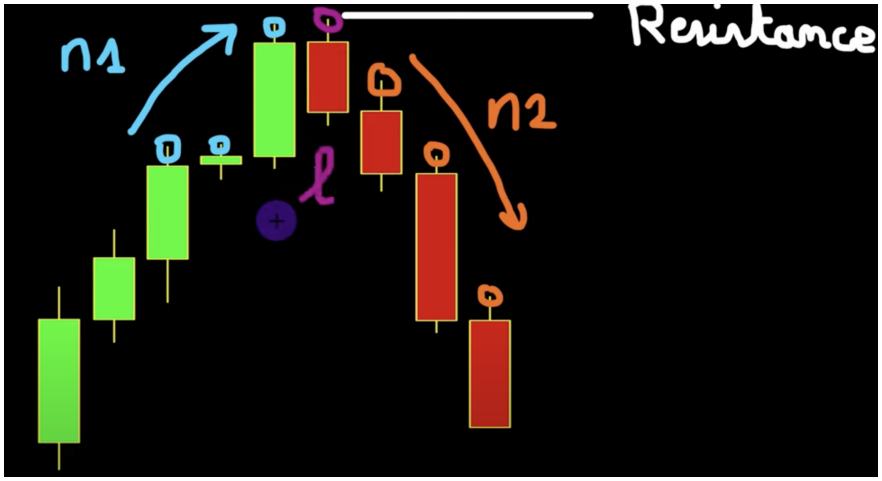

Figure 10.2: Resistance.
[4]

After this we create a list of all those candles which touches the support/resistance levels, i.e; when our function returns the value 1. And for the sake of identification we label a support level as the integer 1 and resistance level as the integer 2. Finally we are producing an array of 3-tuples of the form: [row number(candle of that date), low/high values of that candle, 1 or 2 for the support or resistance identification].
And finally to show the results we are plotting the list of all indices(candles) touching support and list of all indices(candles) touching resistance levels using plotly and matplotlib libraries in python.

Chapter 11

Conclusion

In this project, my aim is to search for features to insert into the Decision Tree Method and then make predictions whether one should buy/sell a certain stock the next day. Additionally, create a Python programme that will automate trading tools like general pattern recognition, support & resistance detection, trend line channel detection, and finally candlestick pattern detection. These tools can tell anyone with a basic understanding of the stock market when to enter or exit the market based on the patterns formed. This program will also predict if the price of the selected stock will increase or decrease. It will determine if we should buy, sell or hold that stock after a month.
lastly to make a decision whether to enter/exit the market for a certain chosen stock of a company we can speculate depending upon our attribute 1 which is the **pattern recognition using the Hausdorff distance** we can conclude that if the distances are approaching towards the integer 0(with some threshold value) then that pattern is very likely to occur again which will determine if now the price will fall or rise depending upon our chosen pattern, e.g:

1. **Head & Shoulders Pattern:** Fall in price level.
2. **Inverse Head & Shoulders Pattern:** Rise in price level.
3. **Saddle Pattern:** Trend Reversal in price level.

Similarly, depending upon our attribute 2 which is candlestick detection if on any of the date we get e.g:

1. 100**:** For the engulfing pattern then that indicates a sharp rise in price level
2. -100**:** For the engulfing pattern then that indicates a sharp fall in price level

Finally, depending upon our attributes 3 and 4 which are **trend line channel and support & resistance detection**, if we can see that if a price level touches a support line then the price is most likely to rally up in price whereas the opposite when it touches the resistance line.

Chapter 12

Summary

I have implemented the code for decision tree classification and the trading tools which I have automated I plan to combine those tools as features and then implement the random forest and XGBoost algorithms for making more accurate predictions than just pattern recognition because, the random forest algorithm will combine all the attributes/features/tools created and then make predictions whether to buy or sell any desired stock. And the main feature engineering like the general pattern recognition within any general window of the original data of the past 2 years for the random forest method has been carried out using ideas like hausdorff and fréchet distance and ray shooting in polygons. I have also learned the fundamentals of ray shooting in polygons and hausdorff and fréchet distance. Other features such as candlestick pattern detection uses the TA-lib library in python whereas support and resistance detection and trend channel detection uses mathematical and various algorithms and concepts like basic understanding of maxima, minima, slope and intercept value modification and updation.

Chapter 13

Future Prospect

I plan to dedicate coming years to improve the functionality of the above mentioned attributes by making them more general and then using these features in implementing a random forest algorithm and the XGBoost combining these features and then constructing a software which can be used by anyone to use the results provided by the random forest classification and XGBoost algorithm based on those features wherein the user will also have the option to select any chart pattern and candlestick pattern upon which they wish to inspect the results on and make decisions accordingly.

Appendix-I

13.1 Code For Decision Tree Classification

Importing Required Libraries & Packages

```
import pandas_datareader.data as web
import datetime
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor
import pandas as pd
import matplotlib.pyplot as plt

from pandas_datareader import data as pdr
import yfinance as yf
yf.pdr_override()
y_symbols = ['some stock ticker']
from datetime import datetime
startdate = datetime(2021,1,1)
enddate = datetime.now()

df = pdr.get_data_yahoo(y_symbols, start=startdate, end=enddate)

df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2021-01-04	17.389999	17.430000	17.059999	17.250000	16.556587	12597600
1	2021-01-05	17.320000	17.670000	17.320000	17.650000	16.940508	8109900
2	2021-01-06	17.400000	17.790001	17.340000	17.730000	17.017292	9136300
3	2021-01-07	17.360001	17.549999	17.260000	17.549999	16.844528	10272000
4	2021-01-08	18.070000	18.610001	18.020000	18.590000	17.842722	17802400
...	...	...	...	...	...	...	...
574	2023-04-17	15.070000	15.290000	15.050000	15.190000	15.190000	17173000
575	2023-04-18	15.170000	15.190000	14.940000	14.960000	14.960000	20768400
576	2023-04-19	14.830000	14.920000	14.760000	14.820000	14.820000	19107300
577	2023-04-20	14.820000	14.990000	14.790000	14.850000	14.850000	16901400
578	2023-04-21	14.910000	14.960000	14.840000	14.870000	14.870000	6908400

579 rows × 7 columns

Figure 13.1: Original Dataset

Performing binary classification based on the returns of the stock by going back 60 days and checking the returns provided by the stock.

```
df['Return'] = df['Adj Close'].pct_change(60).shift(-60)
list_of_features = ['High','Low','Open','Close','Volume','Adj Close']
X = df[list_of_features]
y = np.where(df.Return > 0, 1, 0)
```

Printing df, defined as in the earlier line of code.

```
print(df)
```

Date	Open	High	Low	Close	Adj Close	Volume	Return
2021-01-04	17.389999	17.430000	17.059999	17.250000	16.556587	12597600	0.085217
2021-01-05	17.320000	17.670000	17.320000	17.650000	16.940508	8109900	0.077054
2021-01-06	17.400000	17.790001	17.340000	17.730000	17.017292	9136300	0.092499
2021-01-07	17.360001	17.549999	17.260000	17.549999	16.844526	10272000	0.101994
2021-01-08	18.070000	18.610001	18.020000	18.590000	17.842722	17802400	0.033351
...	...	...	...	...	...	...	...
2023-04-17	15.070000	15.290000	15.050000	15.190000	15.190000	17173000	NaN
2023-04-18	15.170000	15.190000	14.940000	14.960000	14.960000	20768400	NaN
2023-04-19	14.830000	14.920000	14.760000	14.820000	14.820000	19107300	NaN
2023-04-20	14.820000	14.990000	14.790000	14.850000	14.850000	16901400	NaN
2023-04-21	14.910000	14.960000	14.840000	14.870000	14.870000	6908400	NaN

[579 rows x 7 columns]

Figure 13.2: Plotting df

Splitting the data into training and testing data.

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,stratify =
↪  y)
print(X_train.shape)
print(X_test.shape)
print(y_test.shape)
print(y_train.shape)
```

```
562
562
562
```

Figure 13.3: Shapes Of The Testing & Training Data

Creating the decision tree classifier model.

```
treeClassifier=DecisionTreeClassifier(max_depth=3,min_samples_leaf=6)
# random forest or gradient boosted trees(gbm)
treeClassifier.fit(X_train,y_train)
```

Predicting the model.

```
y_pred = treeClassifier.predict(X_test)
```

Calculating Metric Performance.

```
from sklearn.metrics import classification_report
report = classification_report(y_test,y_pred)
print(report)

# precision => 0 indicates SELL signal
# precision => 1 indicates BUY signal
```

	precision	recall	f1-score	support
0	0.56	0.83	0.67	89
1	0.65	0.33	0.44	85
accuracy			0.59	174
macro avg	0.61	0.58	0.56	174
weighted avg	0.61	0.59	0.56	174

Figure 13.4: Metric Performance

Reading the chart and deciding whether to buy or sell the stock.

```
from sklearn import tree
import graphviz
data = tree.export_graphviz(treeClassifier,filled=True,
feature_names=list_of_features,class_names=np.array(['0','1']))
graphviz.Source(data)
```

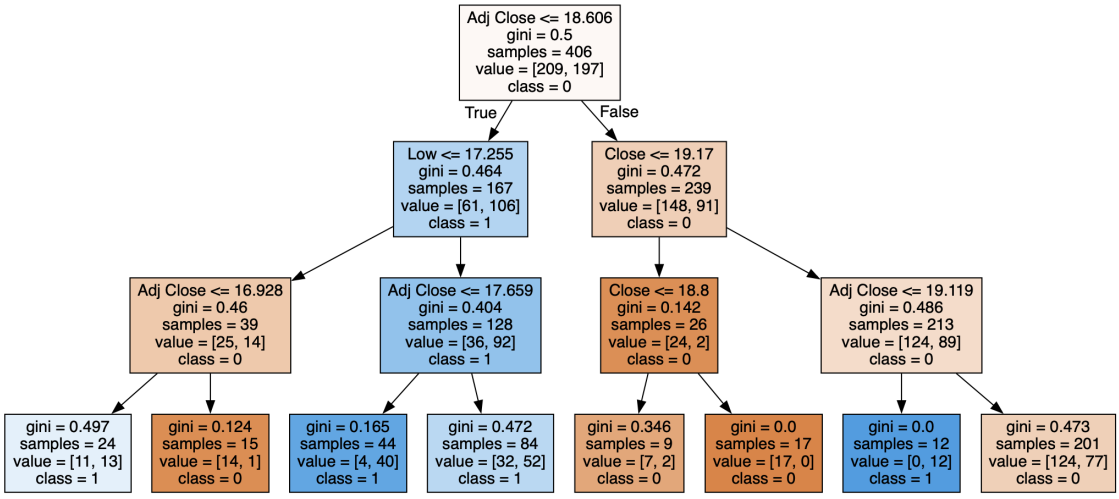

Figure 13.5: Decision Tree Classification

13.2 Code For Attribute 1: Pattern Recognition

Original Dataset

```
df.reset_index(inplace=True)
df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2021-01-04	17.389999	17.430000	17.059999	17.250000	16.556587	12597600
1	2021-01-05	17.320000	17.670000	17.320000	17.650000	16.940508	8109900
2	2021-01-06	17.400000	17.790001	17.340000	17.730000	17.017292	9136300
3	2021-01-07	17.360001	17.549999	17.260000	17.549999	16.844528	10272000
4	2021-01-08	18.070000	18.610001	18.020000	18.590000	17.842722	17802400
...	...	...	...	...	...	...	...
574	2023-04-17	15.070000	15.290000	15.050000	15.190000	15.190000	17173000
575	2023-04-18	15.170000	15.190000	14.940000	14.960000	14.960000	20768400
576	2023-04-19	14.830000	14.920000	14.760000	14.820000	14.820000	19107300
577	2023-04-20	14.820000	14.990000	14.790000	14.850000	14.850000	16901400
578	2023-04-21	14.910000	14.960000	14.840000	14.870000	14.870000	6908400

579 rows × 7 columns

Figure 13.6: Original Dataset

Converted dataset for Zeller's Algorithm conversion

```
close = pd.DataFrame(df.Close)
Dates = df['Date'].dt.strftime('\%d/\%m/\%y')
close['Date'] = Dates
close
```

	Close	Date
0	17.250000	04/01/21
1	17.650000	05/01/21
2	17.730000	06/01/21
3	17.549999	07/01/21
4	18.590000	08/01/21
...	...	...
574	15.190000	17/04/23
575	14.960000	18/04/23
576	14.820000	19/04/23
577	14.850000	20/04/23
578	14.870000	21/04/23

579 rows × 2 columns

Figure 13.7: Converted Dataset

An array of the original dataset

```
# an array of closing values of the original dataset
arr1=np.array(close['Close'])
```

Zeller's algorithm to convert the data into an array of the form (close,day number) to calculate the hausdorff distance between a windowed data where a pattern was observed and the original data.

```
# zeller's algorithm
def dow1(str1):
    d = int(str1[0:2])
    m = int(str1[3:5])
    y = int('20' + str1[6:])
    if m == 1:
        y = y - 1
        m = 13
    elif m ==2:
        y = y - 1
        m = 14
    y = y % 100
    c = y//100
    dow2 = ((d+13*(m+1)//5)+y+(y//4)+(c//4)+5*c)%7
    return dow2


    if   dow1 == 0:
            print("saturday")
    elif dow1 == 1:
            print("sunday")
    elif dow1 == 2:
            print("monday")
    elif dow1 == 3:
            print("tuesday")
    elif dow1 == 4:
            print("wednesday")
    elif dow1 == 5:
            print("thursday")
    elif dow1 == 6:
            print("friday")

dow1('01/12/22')
```

converted array of original dataset

```
arr = []
for i in range(len(close)):
    arr.append([close['Close'][i],dow1(close['Date'][i]) - 1])
```

creating the window function

```
# window function with i as the (index of)starting date and j as the size of
↪  the window(window length(days)).
def window(i,j):
    return close.iloc[i:i+j]  # i will range from 0
#plt.plot(df['Close'])
```

First pattern or the first window where we observed a head and shoulders pattern with a specific window length

```
# 1st head & shoulder
closing1 = window(22,window_length)
closing1.reset_index(drop=True, inplace=True)
print(closing1)
```

	Close	Date
0	17.590000	04/02/21
1	17.629999	05/02/21
2	17.959999	08/02/21
3	17.639999	09/02/21
4	17.500000	10/02/21
5	17.830000	11/02/21
6	18.070000	12/02/21
7	17.750000	16/02/21
8	17.600000	17/02/21
9	17.830000	18/02/21
10	17.940001	19/02/21
11	17.610001	22/02/21
12	17.379999	23/02/21
13	17.680000	24/02/21
14	17.389999	25/02/21
15	17.120001	26/02/21
16	17.660000	01/03/21

Figure 13.8: 1st pattern

converted 1st windowed into (closing price, day number) using zeller's algorithm

```
# for hausdorff distance of 1st head & shoulder
arrw1 = []
```

```
for i in range(len(closing1)):
    arrw1.append([closing1['Close'][i],dow1(closing1['Date'][i]) - 1])
arrw
```

```
[[17.59000015258789, 4],
 [17.6299991607666, 5],
 [17.959999084472656, 1],
 [17.639999389648438, 2],
 [17.5, 3],
 [17.829999923706055, 4],
 [18.06999969482422, 5],
 [17.75, 2],
 [17.600000381469727, 3],
 [17.829999923706055, 4],
 [17.940000534057617, 5],
 [17.610000610351562, 1],
 [17.3799991607666, 2],
 [17.68000030517578, 3],
 [17.389999389648438, 4],
 [17.1200008392334, 5],
 [17.65999984741211, 1]]
```

Figure 13.9: 1st converted pattern

Similarly, the 2nd and 3rd converted windowed datasets(both are inverted heand and shoulders pattern, which indicates a RISE in price) are

```
[[20.540000915527344, 5],
 [20.450000762939453, 1],
 [20.229999542236328, 2],
 [20.3799991607666, 3],
 [20.670000076293945, 4],
 [20.190000534057617, 5],
 [20.530000686645508, 1],
 [20.760000228881836, 2],
 [20.690000534057617, 3],
 [21.190000534057617, 4],
 [21.25, 5],
 [21.290000915527344, 1],
 [21.450000762939453, 2],
 [21.190000534057617, 3],
 [20.959999084472656, 4],
 [21.06999969482422, 5],
 [21.110000610351562, 2],
 [21.18000030517578, 3],
 [20.950000762939453, 4]]
```

Figure 13.10: 2nd converted pattern(inverse h& s)

```
[[23.049999237060547, 2],
 [22.719999313354492, 3],
 [22.239999771118164, 4],
 [22.84000015258789, 5],
 [22.459999084472656, 1],
 [22.15999984741211, 2],
 [22.59000015258789, 3],
 [22.520000457763672, 4],
 [22.459999084472656, 5],
 [22.579999923706055, 1],
 [23.329999923706055, 2],
 [23.899999618530273, 3],
 [23.84000015258789, 4],
 [24.010000228881836, 5],
 [24.690000534057617, 1],
 [24.190000534057617, 2],
 [25.110000610351562, 3],
 [24.489999771118164, 4],
 [25.040000915527344, 5],
 [24.209999084472656, 1],
 [24.790000915527344, 2],
 [24.309999465942383, 3]]
```

Figure 13.11: 3rd converted pattern(inverse h& s)

The calculation of the hausdorff distance between the 1st pattern(usual head and shoulders pattern) and the original dataset

```
def dist_HD(arr,arrw1):
    Hdwocp = []  # Hausdorff distance with original closing price
    for i in range(len(close)-window_length):
        arr1 = arr[i:i+window_length]    # for arr1 (day number,closing on
        ↪  that day)
        Hdwocp.append(directed_hausdorff(arr1, arrw1)[0])

    return Hdwocp
```

3 arrays where each array represents the hausdorff distance between 1st pattern and original dataset and similarly, the hausdorff distance between 2nd pattern and original dataset and also the 3rd pattern

```
# hausdorff distance of 1st h&s with original closing data
h_distance1 = dist_HD(arr,arrw1)
print(h_distance1)
# hausdorff distance of 2nd h&s(inverse) with original closing data
h_distance2 = dist_HD(arr,arrw2)
print(h_distance2)
# hausdorff distance of 3rd h&s(inverse) with original closing data
h_distance3 = dist_HD(arr,arrw3)
print(h_distance3)
```

The length of each distanced arrays considering the 3 identified patterns.

```
print(len(h_distance1))
print(len(h_distance2))
print(len(h_distance3))
```

```
(405, 6)
(174, 6)
(174,)
(405,)
```

Figure 13.12

adding the hausdorff distance calculation of 3 patterns with the original dataset to our entire dataset

```
df['Return'] = df['Adj Close'].pct_change(60).shift(-60)
# so if the distance comes close to 0 say 0.05 then since this was USUAL h&s so we can say
# that the price will FALL
df['Hausdorff_Distance1'] = pd.Series(h_distance1)

# so if the distance comes close to 0 say 0.05 then since these 2 were INVERTED h&s so we can say
# that the price will RISE
df['Hausdorff_Distance2'] = pd.Series(h_distance2)
df['Hausdorff_Distance3'] = pd.Series(h_distance3)
df
```

	Date	Open	High	Low	Close	Adj Close	Volume	Return	Hausdorff_Distance1	Hausdorff_Distance2	Hausdorff_Distance3
0	2021-01-04	17.389999	17.430000	17.059999	17.250000	16.556587	12597600	0.085217	0.85	3.143310	5.010798
1	2021-01-05	17.320000	17.670000	17.320000	17.650000	16.940508	8109900	0.077054	0.85	2.910000	4.795425
2	2021-01-06	17.400000	17.790001	17.340000	17.730000	17.017292	9136300	0.092499	0.85	3.310001	5.452486
3	2021-01-07	17.360001	17.549999	17.260000	17.549999	16.844528	10272000	0.101994	0.85	3.310001	5.452486
4	2021-01-08	18.070000	18.610001	18.020000	18.590000	17.842722	17802400	0.033351	0.85	3.310001	5.452486
...	...	...	...	...	...	...	...	...	...	...	...
574	2023-04-17	15.070000	15.290000	15.050000	15.190000	15.190000	17173000	NaN	NaN	NaN	NaN
575	2023-04-18	15.170000	15.190000	14.940000	14.960000	14.960000	20768400	NaN	NaN	NaN	NaN
576	2023-04-19	14.830000	14.920000	14.760000	14.820000	14.820000	19107300	NaN	NaN	NaN	NaN
577	2023-04-20	14.820000	14.990000	14.790000	14.850000	14.850000	16901400	NaN	NaN	NaN	NaN
578	2023-04-21	14.910000	14.960000	14.840000	14.870000	14.870000	6908400	NaN	NaN	NaN	NaN

579 rows × 11 columns

Figure 13.13: Combined Data

13.3 Attribute 2: Candlestick Detection

Now, here we are using the library TA-Lib in python to identify candlestick patterns in our original dataset which indicates rise or fall in the price.

```
import talib as talib
engulf = talib.CDLENGULFING (df["Open"] , df["High"], df["Low"], df["Close"])
morningstar = talib.CDLMORNINGSTAR(df["Open"] , df["High"], df["Low"],
↪  df["Close"])
hamm = talib.CDLHAMMER(df["Open"] , df["High"], df["Low"], df["Close"])
```

Now, below are some candlestick pattern which were detected on the following dates, where:

1. 100: Indicates that there was a bullish candlestick pattern
2. −100: Indicates that there was a bearish candlestick pattern

```
print(hamm[hamm!=0])
```

```
Date
2021-01-26    100
2021-03-17    100
2021-05-11    100
2021-07-08    100
2021-11-18    100
2021-12-06    100
2021-12-20    100
2022-01-24    100
2022-05-06    100
2022-08-25    100
2022-09-01    100
2022-09-15    100
2022-09-29    100
2023-01-10    100
2023-03-08    100
2023-03-21    100
dtype: int32
```

Figure 13.14

```
print(engulf[engulf!=0])
```

```
Date
2021-01-12   -100
2021-03-08   -100
2021-04-21    100
2021-05-19    100
2021-06-15   -100
2021-07-06    100
2021-08-06    100
2021-08-19    100
2021-10-08   -100
2021-12-10   -100
2021-12-29    100
2022-02-08    100
2022-03-18    100
2022-06-24    100
2022-08-05    100
2022-08-19   -100
2022-10-06   -100
2022-10-13    100
2022-10-18   -100
2022-12-28   -100
2023-01-11    100
2023-01-31    100
2023-02-07    100
2023-04-12   -100
dtype: int32
```

Figure 13.15

```
print(morningstar[morningstar!=0])
```

```
Date
2021-03-01    100
2021-07-20    100
2022-01-14    100
dtype: int32
```

Figure 13.16

13.4 Attribute 3: Trend Line Channel Detection

Final Code For Trend Channel Detection Of Our Original date.

```
import numpy as np
from matplotlib import pyplot
backcandles= 50
brange = 10 #should be less than backcandles
wind = 5

candleid = 123

optbackcandles= backcandles
sldiff = 100
sldist = 1000
for r1 in range(backcandles-brange, backcandles+brange):
    maxim = np.array([])
    minim = np.array([])
    xxmin = np.array([])
    xxmax = np.array([])

    for i in range(candleid-r1, candleid+1, wind):
        minim = np.append(minim, df.Low.iloc[i:i+wind].min())
        xxmin = np.append(xxmin, df.Low.iloc[i:i+wind].idxmin())
    for i in range(candleid-r1, candleid+1, wind):
        maxim = np.append(maxim, df.High.loc[i:i+wind].max())
        xxmax = np.append(xxmax, df.High.iloc[i:i+wind].idxmax())
    slmin, intercmin = np.polyfit(xxmin, minim,1)
    slmax, intercmax = np.polyfit(xxmax, maxim,1)

    dist = (slmax*candleid + intercmax)-(slmin*candleid + intercmin)
    if(dist<sldist): #abs(slmin-slmax)<sldiff and
        #sldiff = abs(slmin-slmax)
        sldist = dist
        optbackcandles=r1
        slminopt = slmin
        slmaxopt = slmax
        intercminopt = intercmin
        intercmaxopt = intercmax
        maximopt = maxim.copy()
        minimopt = minim.copy()
        xxminopt = xxmin.copy()
        xxmaxopt = xxmax.copy()


print(optbackcandles)
dfpl = df[candleid-wind-optbackcandles-backcandles:candleid+optbackcandles]
```

```
fig = go.Figure(data=[go.Candlestick(x=dfpl.index,
                open=dfpl['Open'],
                high=dfpl['High'],
                low=dfpl['Low'],
                close=dfpl['Close'])])

adjintercmax = (df.High.iloc[xxmaxopt] - slmaxopt*xxmaxopt).max()
adjintercmin = (df.Low.iloc[xxminopt] - slminopt*xxminopt).min()
fig.add_trace(go.Scatter(x=xxminopt, y=slminopt*xxminopt + adjintercmin,
↪  mode='lines', name='min slope'))
fig.add_trace(go.Scatter(x=xxmaxopt, y=slmaxopt*xxmaxopt + adjintercmax,
↪  mode='lines', name='max slope'))
fig.show()
```

```
55
```

Figure 13.17: Final Trend Channel Detected

13.5 Attribute 4: Support & Resistance Detection

Final Code For Support & Resistance Detection Of Our Original date.

```
# L: the candle of interest i.e; the candle which will indicate the sup/res
↪  level(ROW OF THAT CANDLE)
# n1: the number of candles before candle L
# n2: the number of candles after candle L

# This function returns 1 if the candle L touches a support or resistance
↪  level and 0 otherwise

def support(df1, l, n1, n2): #n1 n2 before and after candle L
```

```
    for i in range(l-n1+1, l+1): # checking if low values of candles before L are in decreasing order, if not then we break out of the function with 0
        if(df1.Low[i]>df1.Low[i-1]):
            return 0
    for i in range(l+1,l+n2+1):
        if(df1.Low[i]<df1.Low[i-1]): # checking if low values of candles after L are in increasing order, if not return as value 0
            return 0
    return 1  # we return 1 when we get the setup of a support in our data, i.e; if we didn't broke out of the above conditions in our data

#support(df,46,3,2)

def resistance(df1, l, n1, n2): #n1 n2 before and after candle l
    for i in range(l-n1+1, l+1):
        if(df1.High[i]<df1.High[i-1]):
            return 0
    for i in range(l+1,l+n2+1):
        if(df1.High[i]>df1.High[i-1]):
            return 0
    return 1
#resistance(df, 30, 3, 5)

# making a list of all those candles which touch support or the resistance levels(when above function return the value = 1)
# we identify a support level as the integer, 1
# we identify a resistance level as the integer, 2
# an array of 3 tuples(row number, low/high value of that candle,1 or 2 for support/resistance)

sr = []
n1=3
n2=2
for row in range(3, 560): #len(df)-n2 and since n1=3 so range starts from 3 as we are looking back 3 candles before
    if support(df, row, n1, n2):
        sr.append((row,df.Low[row],1))
    if resistance(df, row, n1, n2):
        sr.append((row,df.High[row],2))
print(sr)

# splitting support/resistance levels in different colors to see better
# plotlist1 = list of all indices (candles) touching support
# plotlist2 = list of all indices (candles) touching resistance

plotlist1 = [x[1] for x in sr if x[2]==1]
```

```
plotlist2 = [x[1] for x in sr if x[2]==2]
plotlist1.sort()  # in increasing order
plotlist2.sort()  # in increasing order
# if any consecutive values(LEVELS) are very close to each other then we merge them into ONE single level
for i in range(1,len(plotlist1)):
    if(i>=len(plotlist1)):
        break
    if abs(plotlist1[i]-plotlist1[i-1])<=0.005: # if any of those 2 similar lines(LEVELS) are at a distance < 0.005 then we remove one of those lines
        plotlist1.pop(i)

for i in range(1,len(plotlist2)):
    if(i>=len(plotlist2)):
        break
    if abs(plotlist2[i]-plotlist2[i-1])<=0.005:
        plotlist2.pop(i)
plotlist2

# separating the list sr into 2 lists
# ss: all support levels identified
# rr: all resistance levels identified

ss = []
rr = []
n1=2
n2=2
for row in range(3, 205): #len(df)-n2
    if support(df, row, n1, n2):
        ss.append((row,df.Low[row]))
    if resistance(df, row, n1, n2):
        rr.append((row,df.High[row]))

s = 0
e = 200
dfpl = df[s:e]
import plotly.graph_objects as go
from datetime import datetime
import matplotlib.pyplot as plt

fig = go.Figure(data=[go.Candlestick(x=dfpl.index,
                open=dfpl['Open'],
                high=dfpl['High'],
                low=dfpl['Low'],
                close=dfpl['Close'])])

```

```
c=0
while (1):
    if(c>len(ss)-1 ):
        break
    fig.add_shape(type='line', x0=ss[c][0], y0=ss[c][1],
                  x1=e,
                  y1=ss[c][1],
                  line=dict(color="MediumPurple",width=3)
                  )
    c+=1

c=0
while (1):
    if(c>len(rr)-1 ):
        break
    fig.add_shape(type='line', x0=rr[c][0], y0=rr[c][1],
                  x1=e,
                  y1=rr[c][1],
                  line=dict(color="RoyalBlue",width=1)
                  )
    c+=1
fig.show()
```

Figure 13.18: Final Support & Resistance Detected

Bibliography

[1] Babypips. Trend channel. https://www.babypips.com/forexpedia/trend-channel#:~:text=A%20trend%20channel%20is%20a,help%20see%20uptrends%20and%20downtrends.

[2] Babypips. Trend channel. https://www.babypips.com/forexpedia/trend-channel#:~:text=A%20trend%20channel%20is%20a,help%20see%20uptrends%20and%20downtrends.

[3] D. Birsan, T. & Tiba. One hundred years since the introduction of the set distance by dimitrie pompeiu. In A. & Futura H. & Marti K. & Pandolfi L. Ceragioli, F. & Dontchev, editor, *System Modeling and Optimization*, pages 35–39, Boston, MA, 2006. Springer US.

[4] CodeTrading. Automated support and resistance detection in python. https://www.youtube.com/watch?v=aJ8Og-iLaas&ab_channel=CodeTrading.

[5] CodeTrading. Price trend channels automated in python. https://www.youtube.com/watch?v=phy57DZOIwc&ab_channel=CodeTrading.

[6] Tammy Da Costa. The basics of technical analysis. https://www.dailyfx.com/education/learn-technical-analysis/basics-of-technical-analysis.html, 2019.

[7] Alanazi A Daori H, Alharthi M. Predicting stock prices using the random forest classifier. *Research Square*, (24):3–4, 2022.

[8] ALAN FARLEY. Learn about an ancient method of chart analysis. https://www.investopedia.com/articles/active-trading/092315/5-most-powerful-candlestick-patterns.asp#citation-1.

[9] Rong & Zhou Enmin Gao, Ya & Wang. Stock prediction based on optimized lstm and gru models. *Scientific Programming*, 2021:1–8, 09 2021.

[10] geeksforgeeks. Xgboost. https://www.geeksforgeeks.org/xgboost/.

[11] Mikael Grégoire, Normand & Bouillot. Hausdorff distance between convex polygons. http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html, 1998.

[12] Prashant Gupta. Decision trees in machine learning. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

[13] ADAM HAYES. What is a head and shoulders chart pattern in technical analysis? https://www.investopedia.com/terms/h/head-shoulders.asp.

[14] Nguyet Islam, Mohammad Rafiqul & Nguyen. Comparison of financial models for stock price prediction. *Journal of Risk and Financial Management*, 13(8), 2020.

[15] Fatih Karabiber. What is gini impurity? https://www.learndatasci.com/glossary/gini-impurity/.

[16] Ashutosh & Khan Tuba & Shende Abhishek & Mundhe Vaibhav Khadse, D.B *et. al* Ambule. Stock price prediction using random forest method and twitter sentiment analysis. 2(8):3–4, 2022.

[17] Ha Young Kim, Taewook & Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PLOS ONE*, 14:1–23, 02 2019.

[18] Swati Madge, Saahil & Bhatt. Predicting stock price direction using support vector machines. 2015.

[19] David & Suri Subhash Mitchell, Joseph *et al.* Mount. Query-sensitive ray shooting. *International Journal of Computational Geometry and Applications*, 7, 03 1996.

[20] Casey Murphy. Support and resistance basics. https://www.investopedia.com/trading/support-and-resistance-basics/, 2021.

[21] S. Nison. *Japanese Candlestick Charting Techniques: A Contemporary Guide to the Ancient Investment Techniques of the Far East, Second Edition.* Penguin Publishing Group, 2001.

[22] Stéphane Pelletier. Computing the fréchet distance between two polygonal curves. http://cgm.cs.mcgill.ca/~athens/cs507/Projects/2002/StephanePelletier/, 2002.

[23] Brandon Wendell. Cmt: Finding high quality turning points. https://www.youtube.com/watch?v=sxYcGsAUkIE, Jan 2013.

[24] Traci Williams. Why is quantitative research important? https://www.gcu.edu/blog/doctoral-journey/why-quantitative-research-important.

[25] Maryamu Zakariya. Predict stock prices using random forest regression model in python. https://medium.com/@maryamuzakariya/project-predict-stock-prices-using-random-forest-regression-model-in-python-fbe4edf~:text=The%20r.