# ALGORITHMIC TRADING WITH CASE BASED REASONING AND MULTI-AGENT SYSTEMS

by

**KEE CHONG WEI, ULYSSES**
*(B.S. Management with Finance, University of Warwick)*

**A CAPSTONE REPORT SUBMITTED FOR THE DEGREE OF**

**MASTER OF COMPUTING**

in the

**GRADUATE DIVISION**

of the

**NATIONAL UNIVERSITY OF SINGAPORE**

**2022**

Supervisor:
Professor Keith Barrett Carter
Professor Huang Ke-Wei

# Declaration

I hereby declare that this project report is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in this report.

This report has also not been submitted for any degree in any university previously.

Kee Chong Wei, Ulysses

04 December 2022

## Acknowledgments

This capstone project would not have been possible without the guidance of my two supervisors, Dr. Huang Ke-Wei and Keith Barrett Carter. I would like to thank them for the support provided throughout the duration of the capstone project.

# Contents

# Abstract

Algorithmic Trading with Case Based Reasoning and Multi-Agent Systems

by

Kee Chong Wei, Ulysses

Master of Computing

National University of Singapore

Cryptocurrencies have taken the world by storm in recent years. In this capstone project, we provide an empirical study of Bitcoin price movement prediction using two different machine learning models, Case Based Reasoning, which was designed and constructed by us, and LightGBM. This study was done using two different sets of data. The first dataset consists of only Open, High, Low, Close, and Volume (OHLCV) data, while the second dataset consists of OHLCV data along with further secondary data and price-derived features. We found that the usage of supplementary data resulted in the slight improvement of certain metrics during the backtests. We also constructed a multi-agent system to allow for the automation of data storage, data retrieval, model training, and price movement predictions when the algorithm is deployed to trade on an exchange. In addition, we investigated the impact of technical trading indicators and do find that they provide at least some useful information to help the models work better in some aspects. Our results indicate that the two models exhibit positive characteristics when considering their performance during backtesting and suggests that combining more models together could provide better signal aggregation across different trading regimes.

# List of Figures

# List of Tables

# Chapter 1

# Introduction

For as long as capital markets have existed, people have tried to run infinitely many different kinds of strategies to make predictions on the price movements of tradable assets to generate abnormal returns, also known as *alpha* in the trading industry. However, most active fund managers fail to outperform stock indices benchmarks once costs and fees are incorporated [18, 55] - as the proverbial adage goes, "time in the market beats timing the market". But this does not mean that it is impossible to beat the market. Successful quantitative trading firms, while a small proportion compared to the general trading population, do post outsized returns that beat popular trading benchmarks. Renaissance Technologies' Medallion Fund is perhaps the most notable one as it had a compound return of 63.3% when calculating returns from 1998 to 2018 [3].

Even while quantitative trading firms do not publish their successful strategies for the public, they are known to employ a significant amount of mathematical methods to assemble trading strategies. As such, with the rapid rate of development in machine learning (ML) and artificial intelligence (AI) in recent years, it is unsurprising that many researchers have attempted to use these computational methods to derive trading strategies. Furthermore, the availability of data is ever growing, with electronic sources like Twitter and Reddit forums providing a plethora of avenues for quantitative researchers to analyse investor sentiment on a real-time basis.

As such, it is inevitable for the rise of cryptocurrencies such as Bitcoin (BTC) and Ethereum (ETH) to capture the attention of the general trading population. Being a relatively new asset class compared to stocks that have existed for a much longer time, it can be argued that the cryptocurrencies market may not be as efficient and there may be a much higher probability of achieving success in finding new prediction signals to explain price fluctuations. This is supported by studies from Y. Kurihara and A. Fukushima, and A. Noda, where they concluded that the market efficiency for Bitcoin varies over time and is still evolving [27, 35]. These characteristics make Bitcoin an interesting asset to focus on and hence is the asset we attempt to predict price fluctuations for.

## 1.1 Background and Scope

Many themes for trading strategies categorically exist. At a high level, there are three distinct kinds of trading strategies, those that are *price* dependent, those that are *event* dependent, and those that are *value* dependent [43]. As such, there are many avenues we could explore in the process of creating our trading strategy. However, some trading strategies which are applicable to stocks may not be as applicable to Bitcoin. For example, for stocks, where the underlying asset is a business, we are able to create a strategy that aims to generate returns when corporate actions such as a merger or acquisition occur. In comparison, a corporate action for Bitcoin does not really exist in the same way. By the same token, fundamental analysis which typically aims to derive the value of a business, and its corresponding stock, through the analysis of financial statements and macro-industry reports is not applicable to Bitcoin in the same way. Therefore, we decided to focus on technical trading and trend trading strategies, in which we analyse the historical price data for Bitcoin alongside commonly used technical indicators by traders.

Typically, deployed algorithmic trading systems ingest large amounts of data rapidly and have many software engineers working on different parts of the trading pipeline. From data engineering, to data analysis, to execution of buy and sell orders, it is difficult to encompass the entire trading pipeline in this project. Furthermore,

to compete with quantitative traders on execution speed, the execution engine will most likely have to be written in a low-level language such as C++, which is trickier to work with than a higher-level language like Python, and would require much more time to develop. For these reasons, we focus on the modelling aspect of the trading pipeline, where we can potentially create a sound trading strategy. We also only focus on one asset, Bitcoin, and do not include the potential to hold a portfolio of different financial assets. The reasoning behind this is that once a profit generating strategy can be created for Bitcoin, one can then look at other financial assets to create more profitable trading strategies and then combine these into a portfolio. As such, portfolio construction and portfolio optimisation aspects of a trading strategy are not being investigated in this project.

This project has undergone the supervision of two different professors:

1. Professor Keith Barrett Carter: January 2022 to June 2022.
2. Professor Huang Ke-Wei: August 2022 to November 2022.

In the first phase of the capstone project, under Professor Carter, there was a strong emphasis placed on producing an algorithm which used Case Based Reasoning and can be deployed to trade on a cryptocurrency exchange. As such, in the first phase of the project, most of the time spent on the project was focused on finding suitable APIs to build the algorithm upon and creating an automated system to ingest new price data and actively trade on an exchange without human intervention. For the automated system, we created a PostgreSQL database to store security, price and metadata data.

In the second phase of the capstone project, under Professor Huang, the focus was placed more on the research aspect of building the trading strategy, specifically, the choice of model used and further exploratory data analysis. Additionally, there was less of a focus on being able to deploy the model at the end of the project, and hence, more exploration was done in a data science fashion using Jupyter Notebooks to generate useful visualizations and analysis. To compare the performance of the Case

Based Reasoning algorithm, we utilized LightGBM, which is a gradient boosting framework.

## 1.2 Primer on Bitcoin and Cryptocurrencies

Since the introduction of Bitcoin by Satoshi Nakamoto in 2008 [34], there has been much fervor about using cryptocurrencies as part of the foundations of a new financial ecosystem, one that can be transparent, and accessible by everyone. Cryptocurrencies are a unique form of monetary value as they are not issued by any central authority or government unlike the common fiat money we use daily. This makes cryptocurrencies part of alternative investments as they are strikingly different from traditional currencies. Blockchains utilize peer-to-peer (P2P) networks to communicate with any party interested in joining the network to verify transactions across the protocol. Every approved transaction is written into what is known as a distributed ledger and each successful block that is mined is then broadcasted across the network. Every block contains a cryptographic hash of the previous block and hence the term blockchain is used to describe this technology [54].

## 1.3 Prediction of Price Movements

The efficient market hypothesis (EMH) is one of the most famous thesis in Finance proposed by Eugene Fama [11] that is widely studied, and especially important to take note of when doing financial market predictions [30, 49]. The EMH states that stock markets are incredibly efficient at reflecting information about stocks and the stock market in general, and as such, any attempts to use existing information to derive future price movements is futile. In effect, this means that any analysis, technical or fundamental, used to find undervaluation or overvaluation of stocks to derive alpha is a fool's errand [30]. Furthermore, for any increased returns, there will be a required increase in risk taken.

However, several papers have found statistically significant evidence that this

may not be the case [5]. For example, Antweiler and Frank [4] in 2004 found that Internet stock message boards do contain relevant information in predicting volatility of the underlying assets. D. Garcia and F. Schweitze [13] found in a study on the impact of social signals on Bitcoin that social media sentiment from Twitter does precede rises in trading volumes on exchanges, which suggest that markets may not be fully efficient. Therefore, given that Bitcoin itself is a relatively new asset, it can be assumed that there remains adequate space for predicting price movements.

## 1.4 Asset Pricing Models

While Bitcoin is not a traditional financial asset, it still gives good grounding to cover the most widely studied asset pricing model in finance. In modern finance theory, the Capital Asset Pricing Model (CAPM) is often used to model the expected returns for a certain stock given an amount of risk taken. This can be thought of as a method to quantify the amount of risk an investor is taking.

$$E(R_i) = R_f + \beta_i(R_m - R_f)$$

In the above definition of the CAPM, $R_i$ is the return for a given stock $i$, $R_f$ is the risk-free rate (people often use treasury bills as the risk-free rate) and $R_m$ is the return for the stock market. $\beta_i$ is used to measure the systematic risk of the particular stock and is calculated using the following definition:

$$B_i = \frac{covariance(R_i, R_m)}{variance(R_m)}$$

As evident in the formula used to calculate the CAPM, only systematic risk is being accounted for as the model assumes that any unsystematic risk is diversifiable and will be diversified by these rational investors and hence be ignored in the model [8]. On the other hand, systematic risk is deemed undiversifiable as even if you decided to split your portfolio among all assets available in the market, your portfolio is still subject to the movement of the market as a whole [47]. It is important to mention the assumptions of the CAPM at this point. Aside from the two main assumptions of modern financial theory:

1. Markets are efficient.

2. Markets are predominantly comprised of rational and risk-adverse individuals seeking to maximize their utility.

The CAPM assumes [8, 20]:

1. Investors are risk-adverse, rational, and utility-maximizers.
2. Markets are frictionless such that (1) investors are able to borrow and lend at the risk-free rate with no restrictions, (2) there are no transaction costs and zero taxes.
3. Investors have the same holding horizon.
4. Assets are infinitely divisible.

Evidently, some of these assumptions do not hold up in the real world. For example, there are transaction costs when trading any asset and in most sovereignties taxes do exist. Furthermore, investors do not often times share the same investment horizon, especially when we consider speculators who tend to be short-term focused, as compared to investors, who tend to be more long-term focused. Nonetheless, a theoretical model like the CAPM serves to provide an indication of what to look out for when valuing any asset even if it makes a significant amount of assumptions.

## 1.5 Case Based Reasoning

Case Based Reasoning (CBR) is a traditional technique found in artificial intelligence and is based on the idea of using previous experiences to make a decision on how to approach the current situation [25]. This method is rather intuitive to understand as it mimics typical human reasoning. For example, take the situation of going to the fruits market to purchase some apples for consumption. For the very first time, assuming no prior knowledge of how to spot bad apples, a consumer might end up purchasing rotten apples thinking that all apples are the same. However, on the next time round at the fruits market, the consumer will be able to recognize

what a rotten apple looks like using their previous experience and avoid purchasing the rotten ones. Another common example of Case Based Reasoning in action is in law practice, whereby a lawyer in a trial will use previously set precedents to argue for a certain case outcome.

Consequently, for a Case Based Reasoner to work effectively, there are four main aspects to consider [25]:

1. The variety and number of experiences that it has had.
2. Its ability to assess new experiences using the information it has learnt from past experiences.
3. Its ability to derive different solutions to its new experiences - using an old solution directly from an old case might not be as effective as customizing the solution.
4. Its ability to learn from previous cases.

As with most things in machine learning and artificial intelligence, data quality massively affects the performance of models and programs. This is a big reason why in a data science pipeline, most time is still spent on data cleaning and preparation [56]. As such, for a Case Based Reasoner, it follows a similar reasoning where the data provided as its past experiences has to be informative such that the reasoner is able to understand what a successful outcome is and be able to find common areas when a solution is successful for similar cases, and vice versa for unsuccessful outcomes.

Hence, cases form the core of any Case Based Reasoner. For one to reason about new experiences from its knowledge gained from past experiences, it must first be able to retrieve those relevant past cases. Kolodner [25] terms this the **indexing problem** as it can be understood similar to indexing a set of experiences in memory and retrieving the index of those experiences that are most alike to the current one. Once these relevant past experiences are retrieved, the Case Based Reasoner must then adapt their previous solutions to the current situation. In order to allow for

the Case Based Reasoner to be more effective overtime, there must also be some learning involved. This can come in the form of evaluation of proposed solutions and desired outcomes where if the proposed solution results in a desired outcome, it is evaluated as a satisfactory solution, and an unsatisfactory solution otherwise.

With this theoretical understanding of how a Case Based Reasoner works, we can then explore what it means to attempt to use one for making a decision on whether to buy, sell, or hold a certain financial asset. While we focus on Bitcoin as our asset in this project, one can generalize this idea to other assets as well. However, as each financial asset is different, the representation of a case for a different financial asset may be different.

Before diving further, let us first establish exactly what a case consists of. A case in our context should provide the necessary information for the Case Based Reasoner to understand what kind of financial environment the past experience belongs to. For ease of understanding and modelling, we set our time resolution, or horizon, to be daily. This means each case must provide details about the type of financial and trading environment on a daily basis so that we are able to classify each new data point at every time epoch to a certain kind of environment. Formally, at each new time epoch $t$, we will have a set of independent features $x_1, x_2, \ldots, x_n$ that we use as a measure of the current epoch's environment. Hence, with this setup, the Case Based Reasoner can look back into its past experiences to determine what sort of environment the new situation is like and the type of solutions that are more likely to find success.

Consequently, there are many features we can add to characterize a case and the kind of trading environment it belongs to, but we should not be including any irrelevant data, otherwise we risk running into overfitting problems from the inclusion of noisy data [53, 54]. Therefore, while it is difficult to say for sure which features exactly to include, a useful heuristic would be to first look to include those independent features that exhibit statistical significance when regressed against the Bitcoin response variable. In an empirical study done by H. Jang and J. Lee [21], a number of independent variables were regressed against the Bitcoin *price*, *return*, $log(price)$, and $log(volume)$. Most of the regressors exhibited a statistically

significant linear relationship as the results of the t-test which tested the null hypothesis that no linear relationship exists between the two variables were rejected. We use a number of these variables as independent features in our case - where a "case" represents a row of data. We list them here:

- Standard & Poor's 500 Index (S&P 500)
- Dow Jones Industrial Index (DJI)
- Nasdaq Composite Index (NDQ)
- Shanghai Stock Exchange Index (SSE)
- Financial Times Stock Exchange 100 Index (FTSE 100)
- USD/GBP
- USD/CNY
- USD/JPY
- USD/EUR
- USD/CHF

The inclusion of the above data as independent variables is also supported by other studies that show stock market indices like the S&P 500 to have a relationship with Bitcoin. In 2020, T. Conlon and R. McGee [10] discovered that during the bear market that followed after the Covid-19 pandemic, the S&P 500 and Bitcoin moved with close adherence to each other. Bitcoin itself is a form of currency, albeit a cryptocurrency, and hence the addition of other currencies make financial sense too.

To use Case Based Reasoning, we first need a set of past cases for the algorithm to decide from when faced with each new situation. In an abstract layout, for each new situation, our algorithm should look at past cases to gather data on the previous actions it took and the results of those actions, then feed this data into the algorithm's decision logic to produce the decision to take at the current situation. Consequently, in our production setting, we require two main abilities. The first is

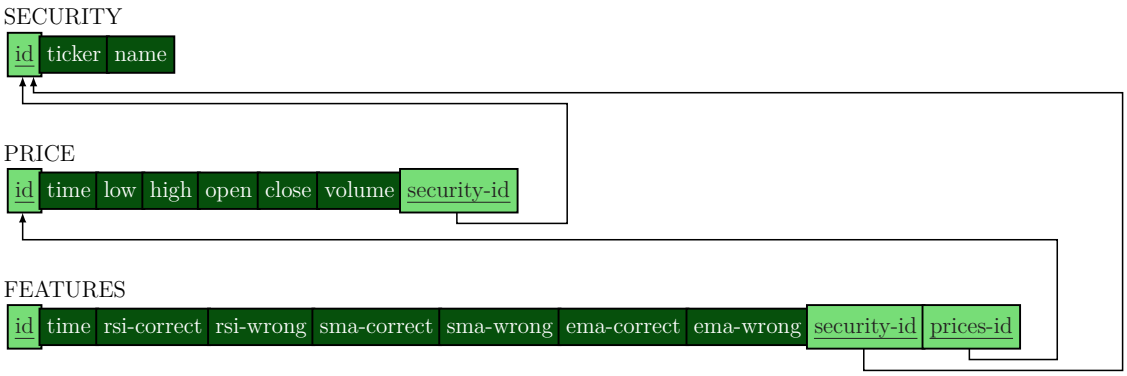



Figure 1.1: Relational Schema

to be able to store cases (old and new) in a database such that we can retrieve old cases and write new cases, and the second is to have some way of finding out which are the relevant cases when encountering a new situation.

### 1.5.1 Case Storage

We first look at the storage aspect of the Case Based Reasoner. To implement this functionality, we wanted to go for a type of database that would be extensible in the future to a greater workload than as required in the project and also an option that is an open sourced project. Additionally, since we are dealing with financial data, we prefer to have ACID properties instead of BASE properties for our database. As such, we settled on a SQL database and chose PostgreSQL as it has been noted to be well-suited for applications that involve high reads and writes [36, 33].

For our Case Based Reasoner, we maintain three distinct tables. The first table `SECURITY` stores information about a specific security, with its `ticker` and `name`. The second table `PRICE` stores prices information for any security. For a security's price, we only consider the `Open`, `High`, `Low`, `Close`, and `Volume` price data that is often available from exchanges. For the third table, it contains the metadata that we want to store for our Case Based Reasoner. The metadata serves to inform our Case Based Reasoner of the effectiveness of previously used cases in terms of predicting a

signal. The `id` of each security in the `SECURITY` serves as the table's primary key. The `PRICE` table contains a foreign key `security-id` that references the `id` field in the `SECURITY` table. The `FEATURES` table contains two foreign keys, a `security-id` field that references the `id` field in `SECURITY`, and a `prices-id` field that references the `id` field in `PRICE`.

### 1.5.2 Case Retrieval

In this section, we explore the retrieval aspect of the Case Based Reasoner. Case retrieval is a key aspect of the program as it is ultimately this step that initiates the reasoning logic of the Case Based Reasoner by surfacing the most appropriate past cases so that it can apply the knowledge from these past cases to generate a solution to the current one.

There are two approaches that we can take to this problem. We can try supervised learning, whereby we label each case with an appropriate tag so that the retrieval algorithm is able to know how to select related cases, or we can utilize unsupervised learning algorithms that do not need an explicit label to learn the patterns in the dataset. To label the dataset with explicit labels as in the case of supervised learning, we must have some specific domain knowledge that allows us to specify the type of regimes available and which regime a particular data point belongs to. However, it is difficult to establish with certainty the types of regimes that may exist. For example, consider that in a bull market, there may be further underlying subregimes e.g. an interest rate hike environment that exist, and labeling each data point in a bull market is too general a blanket statement. As such, we take the approach of unsupervised learning, whereby no pre-existing labels exist and we utilize an unsupervised learning algorithm to figure out the patterns in the data.

For such a task, clustering methods are highly appropriate and applicable. Clustering in machine learning can be defined as the separation of unlabeled data points to form groups that exhibit similarity in terms of characteristics [54]. It is key to mention again that this is an unsupervised machine learning task, which means that there are no indicative labels in the dataset to inform the algorithm explicitly

of a true cluster grouping. Clustering aims to separate the data into groups such that they have high intra-cluster similarity and low inter-cluster similarity [52]. To conduct clustering for the retrieval of similar past cases, we utilize the K-means algorithm. We chose this algorithm as it is widely studied and documented, and commonly used in industry.

The K-means algorithm takes as input a parameter $K$ which specifies the number of clusters and works to separate the data into clusters such that it minimizes the *inertia*, which is a benchmark for how well the K-means clustering has performed. *Inertia* measures the within-cluster-sum-of-squares, which simply means the sum of the total distance between points in a cluster and the cluster centroid. Formally, this objective can be specified as [1]:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2)$$

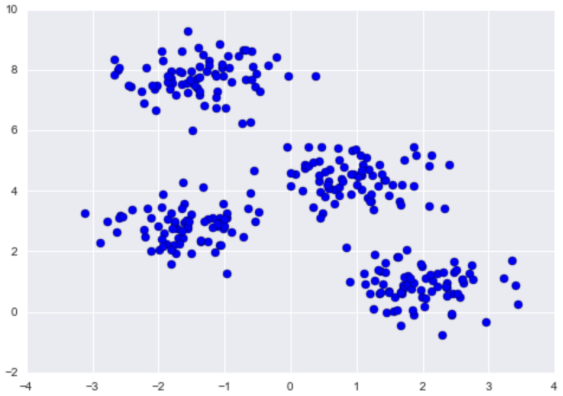
The algorithm works by initializing a number of samples (as given by number of clusters chosen) as centroids and has two main alternating steps [22]:

1. **Assignment**: The algorithm iterates over all samples in the dataset and calculates the distance from the sample to each cluster centroid. Following, it assigns the membership of the sample to its nearest cluster centroid.
2. **Update**: The algorithm uses the newly assigned clusters to recalculate the new cluster centroids.

The algorithm halts once the cluster assignments no longer change or when a specified tolerance level is reached. An example of raw data being clustered into the separate clusters is shown in Figure 1.2.

## 1.6 LightGBM

LightGBM stands for Light Gradient Boosting Machine and is a relatively recent model for implementing Gradient Boosted Trees in machine learning. The




(a) Before clusters are assigned


(b) After clusters are assigned

Figure 1.2: KMeans Clustering Example [19]

paper on LightGBM was published in 2017 [23] and highlights two main innovative ideas, Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), that combined with Gradient Boosted Decision Trees created the LightGBM framework. GOSS is the idea that to preserve the accuracy of computed information gain, when down sampling data, it is much better to randomly drop only those data instances with small gradients and keep all those data instances that exceed a certain threshold. EFB can be described as a technique to reduce the feature space by bundling those mutually exclusive features together into a single feature. Mutually exclusive features can be defined as those features in the feature space that do not take on non-zero values concurrently. An example of this as provided in the LightGBM paper is that of One-Hot Encoding which is commonly used in machine learning to encode dummy variables. It does this using an almost lossless approach with regards to accuracy and this reduction in feature space offers a remarkable

speedup in training of the Gradient Boosted Trees. Hence this Gradient Boosted Decision Tree is prefixed with **Light**, to indicate its fast speed [29].

As explained in the documentation [51] and [29], a very distinctive feature that differentiates LightGBM from XGBoost (another highly popular decision tree based algorithm) is that the trees in LightGBM grow leaf-wise (best-wise) instead of level-wise (depth-wise). It does this by choosing the leaf with the max delta loss to grow and because the leaf-wise splits are selected based on their impact on the global loss, leaf-wise algorithms tend to result in models with lower loss than level-wise algorithms. A last thing to note is that if both trees are fully grown, through leaf-wise or level-wise exploration, they do result in the same tree. However, in most cases, the full tree is not expanded and thus the method of expanding the tree matters a lot for performance [41].

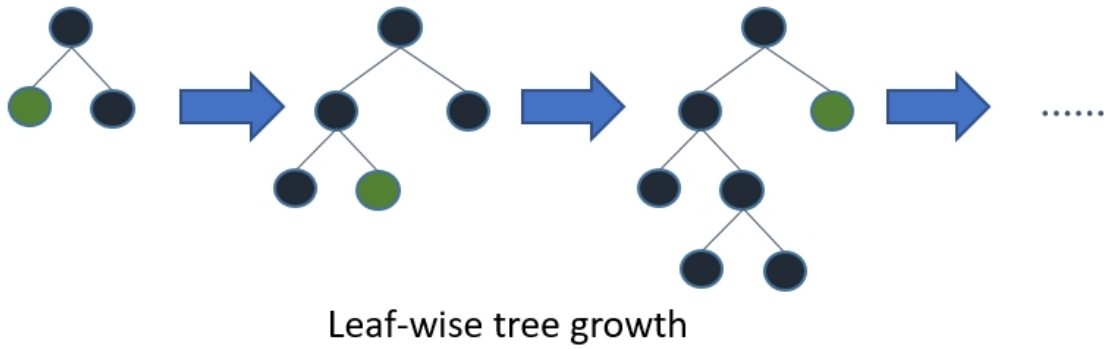



Figure 1.3: Leaf-wise Tree Growth (LightGBM) [51]

Gradient Boosted Trees are a form of supervised learning. As such, we have to define our target variable $y$ alongside our independent variables $x_1, x_2, \ldots, x_n$. This is different from the K-means algorithm explained in the Case Based Reasoning section that uses unsupervised learning. To reduce the price prediction problem, we transform the problem into a multi-class classification problem, where we discretize our response variable $y$ by binning the return values of Bitcoin into three different buckets: buy, hold, and sell. We elaborate further on how values are binned in Chapter 3, where we discuss the methodology of the project. By reducing the problem into a classification problem, we only have to make a decision to buy if the

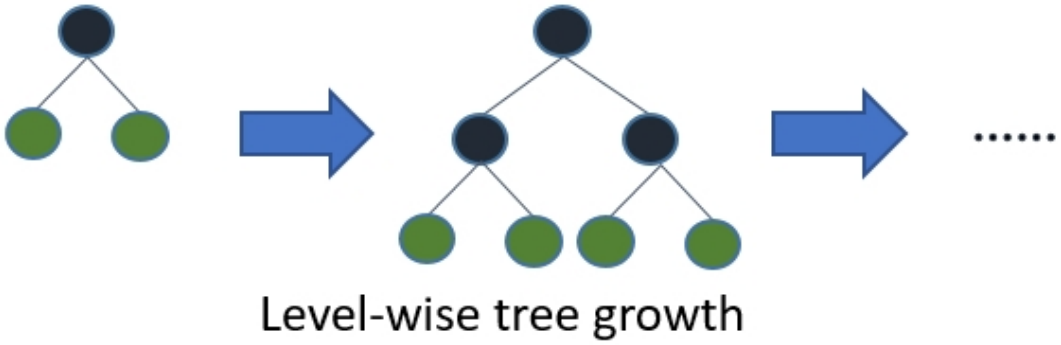



Figure 1.4: Level-wise Tree Growth (XGBoost) [51]

predicted price is to go up, to sell if the predicted price is to go down, and to hold otherwise. This is much easier in comparison to a regression problem where we have to predict a numerical value for Bitcoin in the next time step.

## 1.7 Project Code and Implementation

With regards to the code written to implement the ideas illustrated in this project, they will be made available here. If there are any questions, please feel free to contact me at ulysseskcw96@gmail.com.

# Chapter 2

# Literature Review

Before we begin the literature review section, it is pivotal to mention that many successful research works that find success in developing new and novel trading strategies will not be published to the public, and as such, not be available for open researchers to view [48]. This is largely due to the fact that any profitable strategy may lose its edge once announced to the public as other traders seek to trade on the same information. Many trading strategies also work differently once they start moving larger amounts of liquidity in the market as people will be able to more easily track these large movements and reverse engineer an idea of the strategy. Furthermore, as it is difficult to construct a profitable strategy that works over time, even through new events and regime changes, there is very little incentive for these successful researchers to publish their work publicly. This is especially so in the context of a quantitative trading firm as research and development work would be costly and they would be giving away their profit making strategies if they were to publish them [13].

In a survey done by Khedr et al. [24] on statistical and machine learning techniques used in cryptocurrency price prediction, they enumerate the common models used in statistical and econometric methods, as well as the common models used in machine learning. For econometric models, researchers often used the generalized autoregressive conditional heteroscedasticity (GARCH) model, linear regression, and vector autoregressive models. This is expected as price prediction involves time series analysis and autoregressive models are often used in this context.

On the machine learning side, they find that researchers often use logistic regression, support vector machines (SVM), artificial neural networks (ANN), random forest (RF), deep learning (DL), reinforcement learning (RL), and gradient boosting (GB).

In 2018, Sovbetov [45] investigated factors that influenced the prices of the top five most popular cryptocurrencies at the time, namely Bitcoin, Ethereum, Dash, Litecoin, and Monero using the Autoregressive Distributed Lag (ARDL) technique. Sovbetov found that the market beta, trading volume, and volatility of the crytocurrency market exhibit statistically significant impact on the prices of these five cryptocurrencies. As a cryptocurrency, Bitcoin displays very different social characteristics to traditional stocks. The online nature of cryptocurrencies and readily available blockchain data which is accessible by anyone with access to a computer amplifies the social discussions taking place on Twitter and other online avenues. Garcia D. [13] found in a time-series analysis of exchange volume and Twitter emotional valence that there exists temporal patterns between the aforementioned data and Bitcoin prices.

Several studies have also explored the ability of technical indicators to predict price movements. This aspect of price prediction is documented well in a survey done by Fang et al. [12]. R. Hudson and A. Urquhart [17] studied in detail five popular variants of technical indicators: Moving Average, Filter, Support-Resistance, Oscillator, and Channel Breakout. They found that the annualized average return for each of these sets of rules were statistically significant at the 5% level. Interestingly, they also reported that in the out-of-sample period, they did not find statistically significant evidence of predictability power for Bitcoin unlike other cryptocurrencies. K. Grobys, S. Ahmed, and N. Sapkot [14] in their study of technical trading rules for the cryptocurrency market found success in utilizing moving average trading strategies, and further find that the cryptocurrency markets do not encompass weak form market efficiency.

As such, we also aim to incorporate technical trading indicators as unlike traditional stocks, fundamental analysis does not exist in the same way for cryptocurrencies [17]. There exist no balance sheet, nor income statement in a blockchain for investors and traders to analyse. One can argue that blockchain native data such as

hash rates could constitute as a new form of fundamental analysis for blockchains, however, this would be a relative new research area that is out of scope for the work of this project.

# Chapter 3

# Methodology

## 3.1 Data Sources and APIs

In the first phase of the project, where the focus was on constructing a Case Based Reasoning algorithm that could be deployed, we focused on finding APIs that provided connectivity to exchanges. There are many centralized exchanges that offer cryptocurrency trading services and they usually provide an API for programmatic access as well. Examples include Coinbase, Binance, Kraken, Alpaca Markets, and many more. Those familiar with the decentralized finance (DeFi) space will also note that there exists decentralized applications (DApps) such as Uniswap that offer similar trading functionalities to these centralized exchanges. However, these decentralized exchanges are relatively new and tend to be more complicated to work with as they utilize different protocols to offer trading services. As such, we stick to the centralized exchanges for the scope of this project, however, it should be noted that further exploration should include such DApps as they provide alternative sources of liquidity that could be beneficial for trading strategies.

For the scope of the project, as aforementioned, we sought to work at a higher level layer than the execution layer in the trading application stack, and as a result, we sourced for APIs that abstracted away the requirements to write execution level layer related code. This brings about several benefits. Firstly, this allows us to focus

on building the model and ingesting the OHLCV data provided from the exchange. Secondly, this allows us to be more exchange agnostic and therefore enables the resulting trading algorithm to work on many exchanges, as long as the API used handles the routing to the exchanges.

Another consideration for the API used for obtaining data is the requirement to allow for backtesting on the same platform. This is relatively important as it greatly streamlines the workflow for building and testing the Case Based Reasoner as it means we can both backtest and deploy the algorithm on the same service. This allows for consistent prices to be used from the exchange for both backtesting and deployment. After thorough analysis, we decided on the blankly.finance API which is an open-sourced project that has 1.3k stars on GitHub. The `blankly.finance` API was chosen because it has the previously mentioned qualities we were looking for, and also that it is exchange agnostic and provides the user with the ability to route orders to any supported exchange as long as they have the required API keys. Additionally, the package has relatively good user documentation and the team that is responsible for the project can be contacted in their Discord community, which makes it easier to get answers on questions regarding the API.

Consequently, often used backtesting libraries such as Backtrader, vectorbt, and backtesting.py were not used in the first phase of the project. However, as the second phase of the project focused more on the research aspect, we migrated to backtesting.py for all backtesting purposes. We chose `backtesting.py` as the backtesting framework of choice as it is relatively lightweight, fast and has good user documentation. As well, we sought to add more features to the dataset in the second phase of the project as working with just the requirement of being able to backtest the model provided more flexibility in the development process. For this, we utilized the pandas-datareader library. Lastly, to feed Bitcoin OHLCV data into the backtesting framework, we obtained historical data from here.

## 3.2 Exploratory Data Analysis

In the exploratory data analysis phase (EDA), we sought to gain a better understanding of the data that we are working with. The main data used for this analysis was the `Open`, `High`, `Low`, `Close`, `Volume` (OHLCV) data for Bitcoin obtained from Kraken. In Figure 3.1, we chart the closing price for Bitcoin from the start of the dataset, `06-10-2013`, to the end of the dataset, `30-06-2022`. The data consists of 3190 daily observations that spans roughly 8 years and 9 months which is a significant duration as a Bitcoin itself is relatively new. In general, it is clear that over time, the price of Bitcoin is upward trending and does not seem to be a stationary process. In addition, there seems to be a few different time periods in which the market exhibited different characteristics. For example, there was a bull run in late 2017, from mid of 2020 to early 2021 and from mid 2021 to late 2021. There also seems to be a few bear markets from early 2018 to 2019, from early-mid 2021 to mid 2021 and from late 2021 till now.

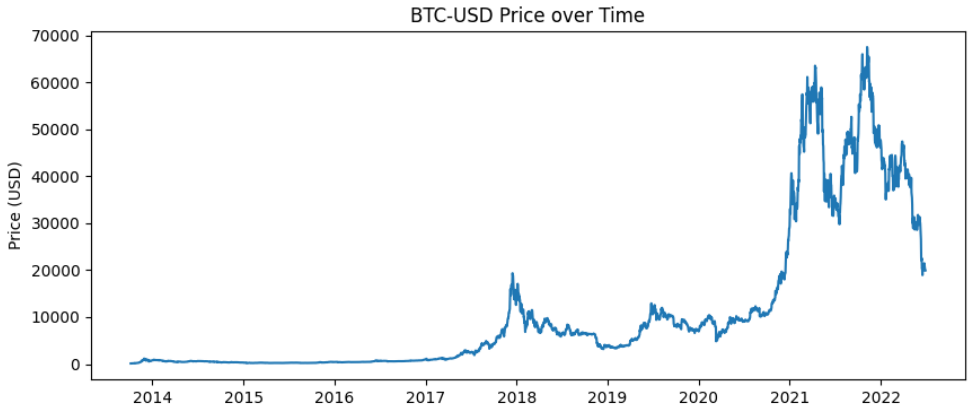



Figure 3.1: BTC-USD Price over Time

To verify that the Bitcoin price series is in fact non-stationary, we investigated further using an Autocorrelation Function (ACF) plot, this is shown in Figure 3.2. From the ACF plot, we see that the ACF decays very slowly and this is a big indication that the process we have is non-stationary as a process that is stationary should have an ACF that decays very quickly. This is further confirmed by conducting an Augmented Dickey-Fuller (ADF) test, which tests for the existence of a unit

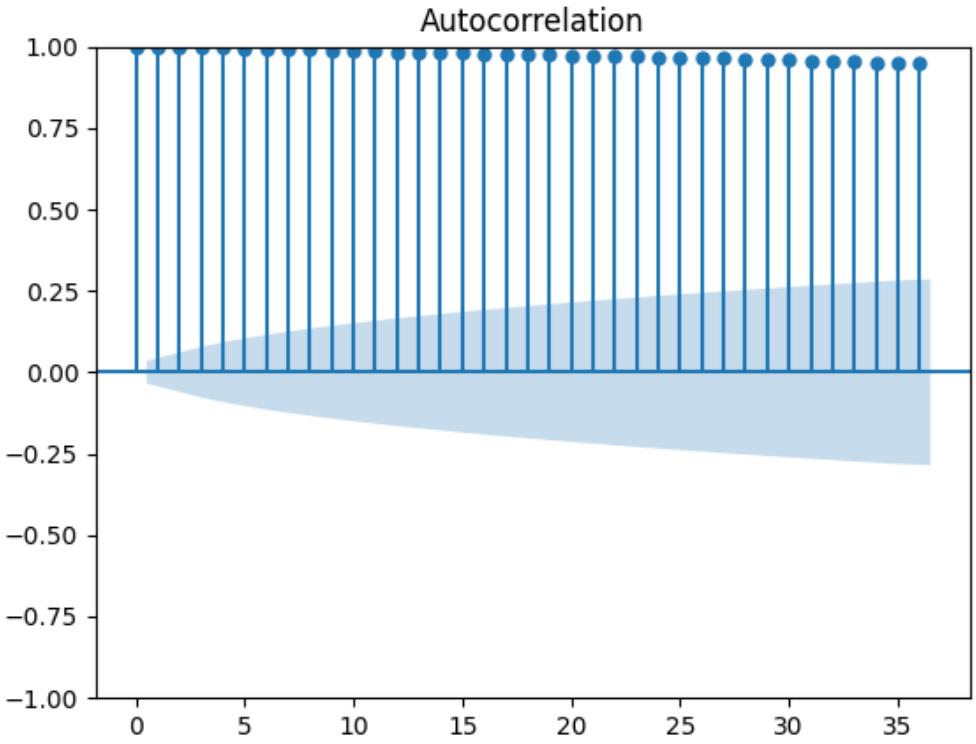



Figure 3.2: BTC-USD ACF plot

root in the time series, where we obtained an ADF test statistic of -1.59, which is greater than the 5% critical value of -2.86. Additionally, the p-value was 0.49 which is greater than 5% and as such, we do not reject the null hypothesis and conclude that there is presence of a unit root in the series. An example of an ACF for a stationary process is shown in Figure 3.3.

Non-stationarity is a common problem for time series analysis and it is important to use a roughly stationary process for analysis as otherwise the statistical inferences drawn from subsequent time series analysis will be misguided [46]. For example, a trend-stationary series will have its expected mean value dependent on the time in which it is being assessed at [26]. In econometrics, practitioners often use a first difference of the series to obtain a stationary series when the series has a unit root. As such, we apply a first difference to the Bitcoin price process. From the ACF plot in Figure 3.4, we see that the ACF decays rapidly and is remarkably different

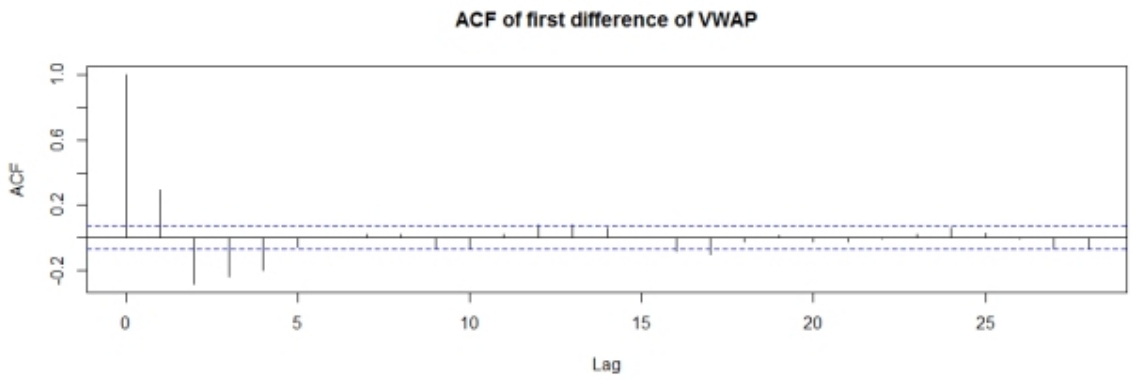



Figure 3.3: Example ACF plot for a stationary process

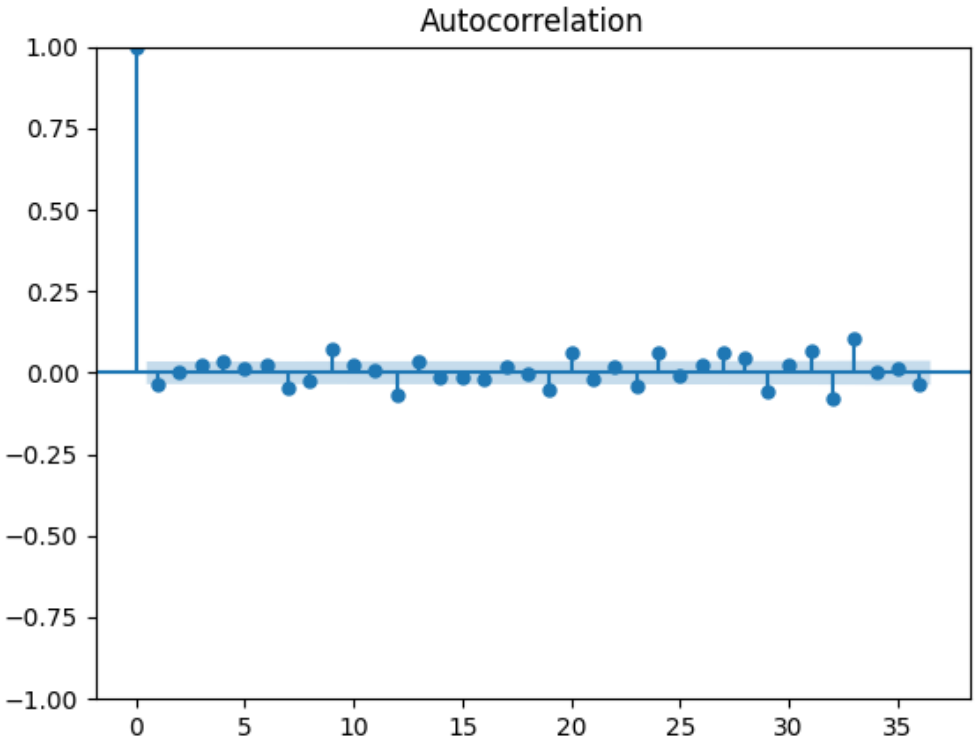



Figure 3.4: BTC-USD First Difference ACF plot

from Figure 3.2, where the ACF decays very slowly. This is a good start to the data transformation. The plot of the first differenced series in Figure 3.5 also shows a roughly stationary mean over time and an ADF test where we obtained a test

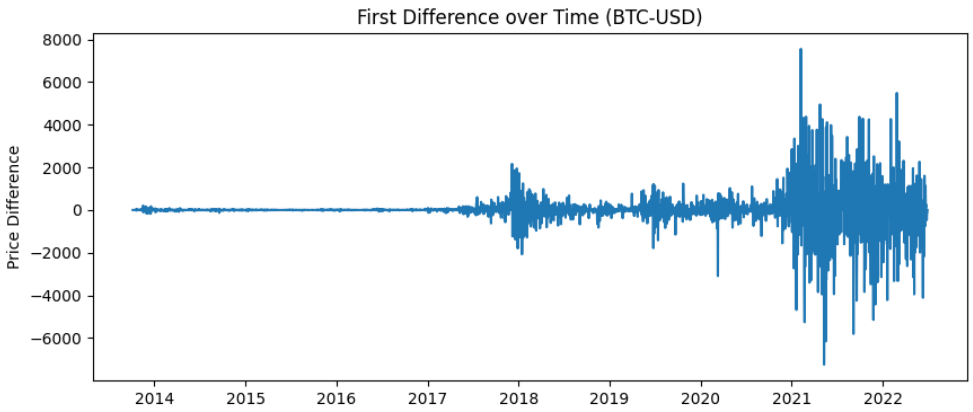



Figure 3.5: BTC-USD First Difference plot

statistic of -9.31 and a 5% critical value of -2.86 now confirms that the transformed series does not contain a unit root. However, it seems that the transformed process' variance is non-constant as there seems to be a significant change in variation post late-2020 [37]. As such, we try another transformation where we convert the series of prices into a series of returns using the following formula:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}}$$

While this transformation decreases the length of the series by one, as the first observation has no previous observation to calculate percentage change, this is not a concern as we still have a significant amount of observations left. The corresponding ACF plot in Figure 3.6 is similar to ACF plot for the first differenced series where the ACF value decays rapidly. This visualization is in agreement with the ADF test statistic calculated where we obtained a value of -14.42 and at a 5% critical value of -2.86, we are able to reject the null hypothesis of the existence of a unit root in the sample data. On further inspection of the transformed series in Figure 3.7, we also see that the variance remains roughly similar throughout the entire sample with only a few outliers.

We stick with this transformation of raw price data to returns data for two reasons. Firstly, with regards to the statistical tests aforementioned regarding stationarity, we are satisfied with the transformation. Secondly, in finance, and

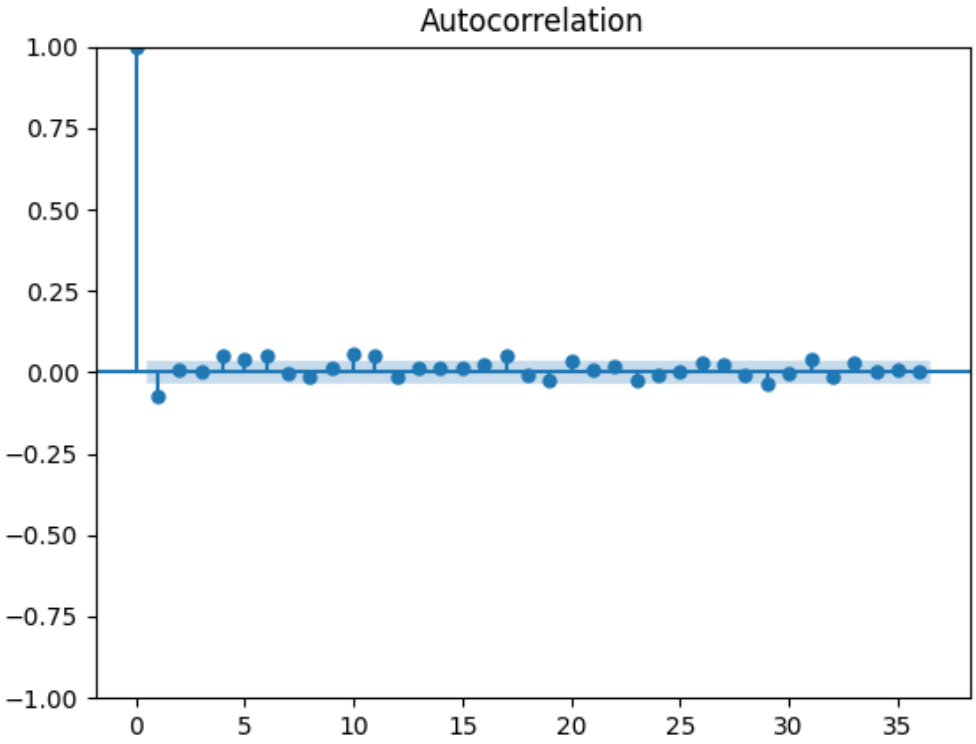



Figure 3.6: BTC-USD Percentage Change ACF plot

trading in particular, industry practitioners often use returns to model their data as the transformation is intuitive to understand and allows one to directly interpret the results of their model. Consequently, for all price data, we transform the raw price data into percentage change data.

## 3.3 Data Preprocessing and Feature Engineering

In this section, we elaborate on the processing of the obtained raw data from multiple sources prior to training and testing the Case Based Reasoner and LightGBM algorithms on the resulting processed data.

As mentioned in subsection 3.1, other than the historical Bitcoin OHLCV data obtained from Kraken, we further obtained data for the S&P 500 index, the DJI

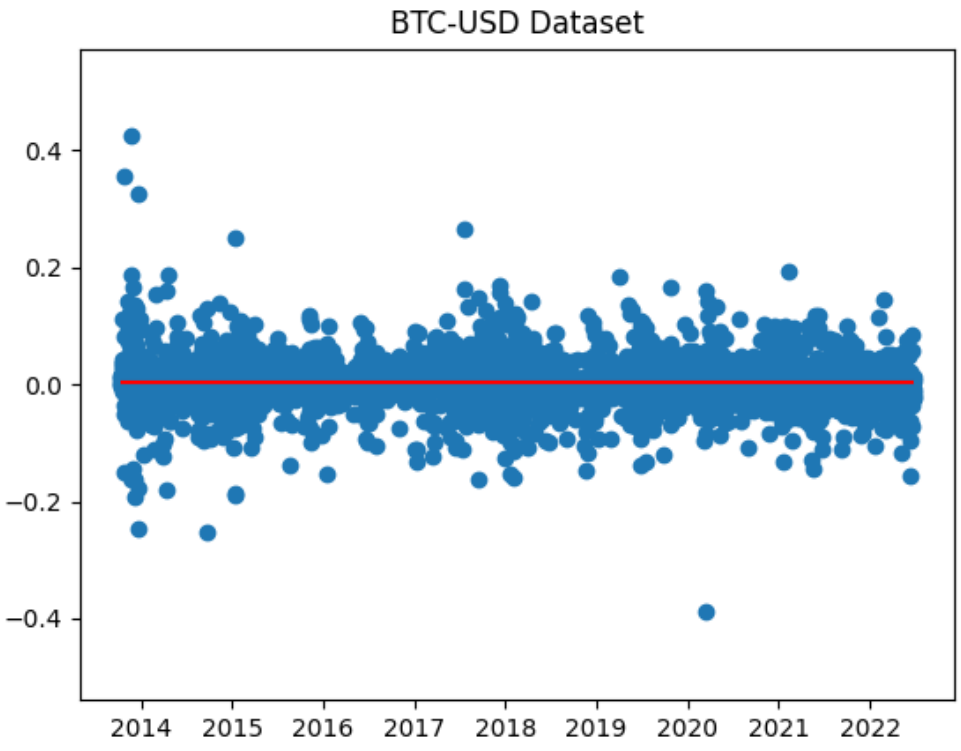



Figure 3.7: BTC-USD Percentage Change plot

index, the NDQ index, the SSE index, the FTSE 100 index, as well as the foreign exchange rates for USD to GBP, USD to CNY, USD to JPY, USD to EUR, and USD to CH. We include this additional data as they have been noted by studies to exhibit some statistically significant relationship with Bitcoin, and as well, to better map out the financial environment for the Case Based Reasoner since industry professionals often do not solely look at a single asset's price to determine the state of the financial environment they are trading or investing in.

We use `DataFrames` from the `pandas` library to manipulate and store the tabular data in memory. When including the additional data, we check for missing dates within the duration from received from the API and forward fill any missing values. For example, if we have data for `11-10-2013`, `13-10-2013`, and `14-10-2013`, we synthetically introduce data for `12-10-2013` using values from `11-10-2013`. As a result, the corresponding percentage change for `12-10-2013` will be 0%.

### 3.3.1 Price Derived Features

We include four price derived features in the form of technical trading indicators, namely Moving Average (MA), Exponential Moving Average (EMA), Relative Strength Index (RSI), and Momentum (MOM).

#### 3.3.1.1 Moving Average (MA)

The moving average is a technical indicator that is commonly used to provide technical traders with an idea of the moving trend of an asset. It is calculated by taking the mean of the prices $P$ across an $n$ number of days [31]:

$$MA_t = \frac{P_t + P_{t-1} + P_{t-2} + \cdots + P_{t-n+1}}{n}$$

For example, for a 5 day moving average, we calculate it as follows:

$$MA_t = \frac{P_t + P_{t-1} + P_{t-2} + P_{t-3} + P_{t-4}}{5}$$

In Figure 3.8, we plot the 30-day and 200-day moving average on top of the BTC-USD price. We see that the longer the duration of the moving average, the smoother the curve and the longer the lag with regards to seeing a change in trend. This is expected as the longer the averaging interval, the slower it is to respond to a change in prices. As such, this indicator can be used in a multitude of ways, to simply smooth out prices in the short term, or provide a more general long-term trending signal.

#### 3.3.1.2 Exponential Moving Average (EMA)

The EMA is similar to the MA indicator previously mentioned, however, its distinctive feature that makes it different to the regular, or simple, MA is that it does not weight all prices equally. For the MA, all prices used are given the same weight when calculating the signal regardless of how recent they are. Conversely, for the EMA, a multiplier is used to give the most recent price a heavier weighting [32]. Where $n$ is the duration of the EMA given by the user, it is calculated as follows:

$$multiplier = \frac{2}{n+1}$$

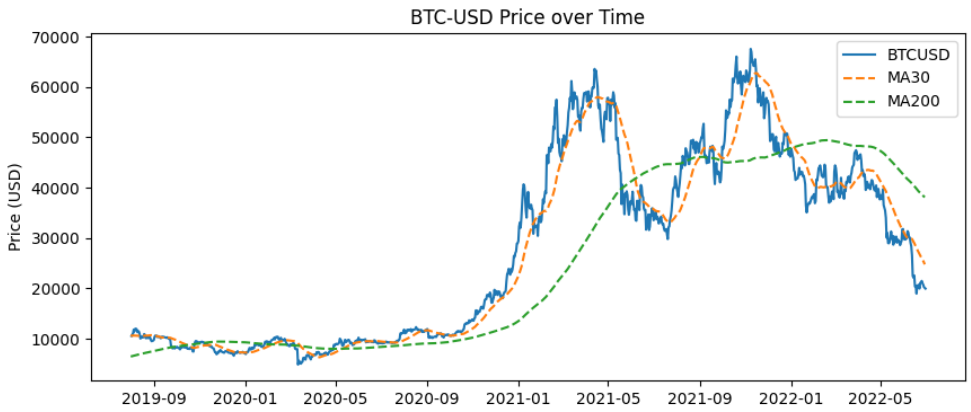



Figure 3.8: BTC-USD Price with Moving Average Indicators

$$EMA_t = (P_t - EMA_{t-1}) \times multiplier + EMA_{t-1}$$

From Figure 3.9, we are able to observe that the two charted EMA lines follow the original BTC-USD price more closely than their MA counterparts. This is due to the EMA giving a heavier weightage to the more recent prices and, as a result, produce a series that has less lag than the MA indicator.

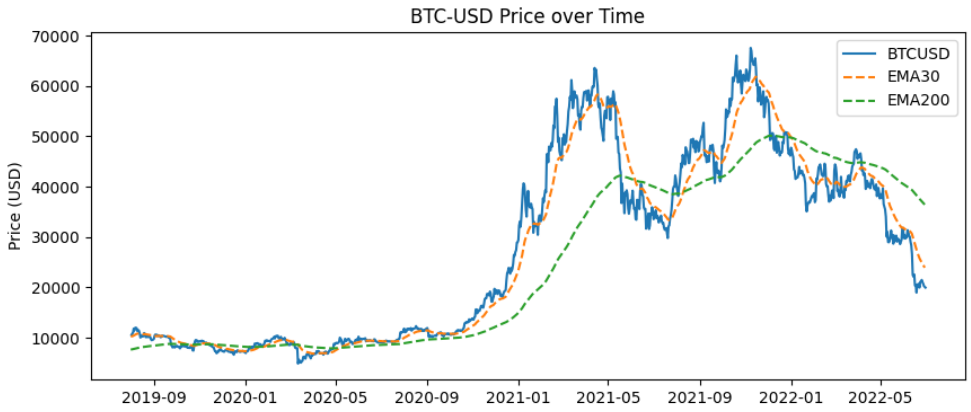



Figure 3.9: BTC-USD Price with Exponential Moving Average Indicators

#### 3.3.1.3 Relative Strength Index (RSI)

The RSI is a technical indicator which provides a measure of the rate of changes in price movements [39]. As well, the RSI is known as a momentum oscillator as its values are bounded between 0 and 100. It was created by J. Welles Wilder and is calculated as follows:

$$RS = \frac{Average\ Gain}{Average\ Loss}$$

$$RSI = 100 - \left[\frac{100}{1 + RS}\right]$$

Average gain and average loss are simply the total gain, or loss, divided by the $n$-period respectively. In the case that average gain in the duration measured is 0, RSI will equal 0. On the other hand, if the average loss in the duration measured is 0, RSI will equal 100 (by definition, to bypass the division by zero error). Wilder in his book suggests using 14 for the number of periods, however, one can increase (decrease) the number of periods to decrease (increase) the sensitivity of the indicator [39]. Typically, when the RSI value is below 30, the asset is considered oversold, and when the RSI value is above 70, the asset is considered overbought. From Figure 3.10, we are able to observe that the 30-day RSI is more sensitive than the 200-day RSI and there have been multiple times where the 30-day RSI exceeded the threshold of 70 and the BTC-USD price series had a reversal signalling that BTC-USD had been overbought.

#### 3.3.1.4 Momentum (MOM)

The last technical indicator we include is the MOM, where it measures the nominal difference in BTC-USD from $n$ periods ago. It is calculated as follows:

$$MOM_t = P_t - P_{t-n}$$

### 3.3.2 Feature Scaling and Normalization

Machine learning algorithms are commonly sensitive to the scale of data. For example, in a gradient descent based algorithm, using different ranges for each feature

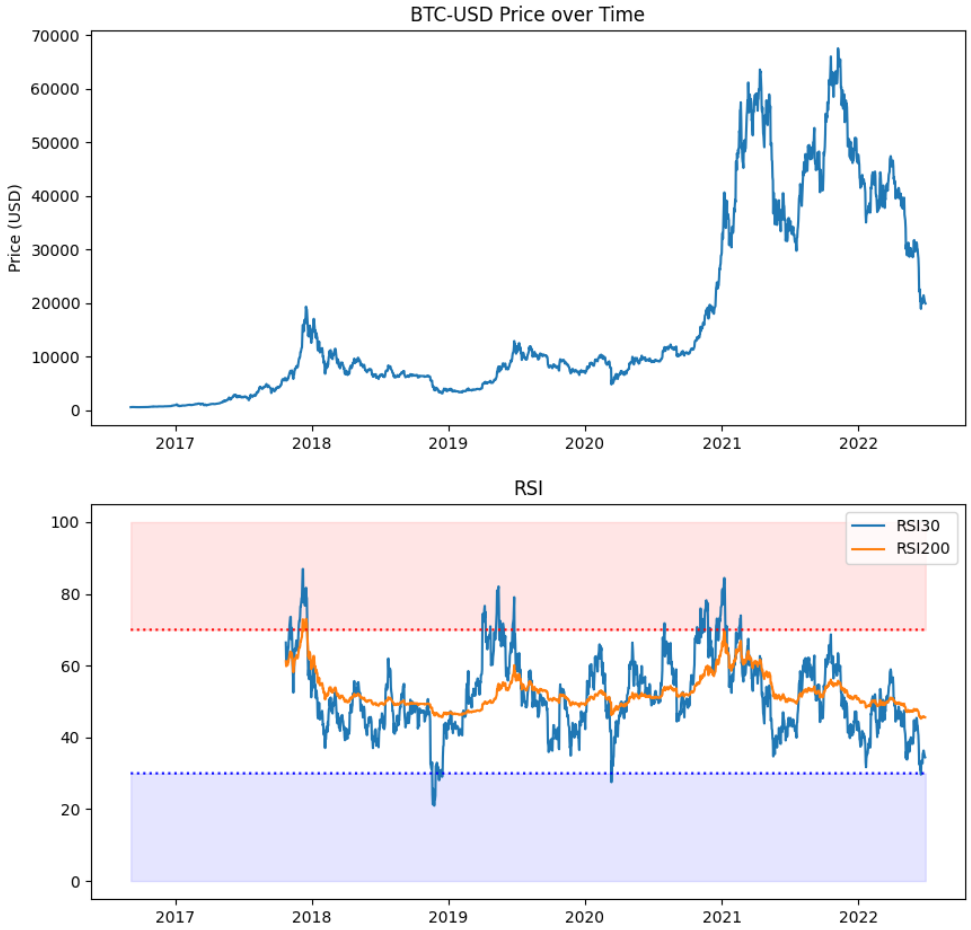



Figure 3.10: BTC-USD Price with Relative Strength Index Indicators

included in the model will result in different step sizes [6]. Clustering algorithms that use a distance measure to cluster different points together such as the K-means algorithm that we use are also affected by different feature ranges. As such, we have to scale our data before feeding it into our algorithms. There are two ways to go about this: standardization and normalization. Standardization is the process of transforming the features such that each feature has a mean of 0 and a standard deviation of 1 and is defined by:

$$z = \frac{x - \mu}{\sigma}$$

Normalization on the other hand is the process of transforming the features such that each feature has a bounded range between 0 and 1, and is defined by [2, 6]:

$$X_{norm} = \frac{X_i - X_{min}}{X_{max} - X_{min}}$$

Standardization is often used when the underlying features exhibit a normal distribution whereas normalization is used when the distribution of the features do not. Additionally, standardization does not bound the resulting values, whereas normalization does. This means that outliers can be affected when using normalization. On initial testing, using normalization in the form of the `MinMaxScaler` from the `scikit-learn` library provided better results than using standardization (`StandardScaler`), hence we stuck with using the `MinMaxScaler`.

## 3.4 Backtests

As described in subsection 1.1, we migrated from `blankly.finance` to `backtesting.py` in the second phase of the project for all backtesting purposes. `backtesting.py` is a lightweight backtesting library that is relatively unopinionated and allows for the user to backtest composable strategies easily. The library is great as it also assists with calculating common metrics used to analyze investment and trading strategies such as:

- Return (%)
- Volatility (%)
- Sharpe Ratio
- Sortino Ratio
- Calmar Ratio
- Max Drawdown (%)
- Average Drawndown (%)
- Win Rate

- Profit Factor

In this section, we give an overview of how the Case Based Reasoner and LightGBM algorithms were implemented to conduct backtesting tests, followed by the corresponding results and evaluation. Of the data sample obtained, we use 70% for training and 30% for testing.

### 3.4.1 Case Based Reasoner

Clustering is at the core of the Case Based Reasoner. We utilized the K-means algorithm to conduct clustering of the sample data. As previously mentioned, the K-means method takes as input the number of clusters $n$ from the user and runs its algorithm to assign each data point to a cluster. As such, we first have to decide on the number of clusters to specify. The Elbow method and Silhouette method are two frequently used methods used to identify the optimal number of clusters to use.

The Elbow method requires an iterative calculation of the total within-cluster-sum-of-squares (WCSS) over a given range of potential number of clusters to use. The optimal number of clusters is then chosen by selecting the number whereby the addition of a new cluster results in a marginally smaller decrease in the WCSS. This is commonly identified where there is a kink in the inertia chart, hence why this method is known as the Elbow method. We use this number of clusters instead of where the inertia is at its minimum because the WCSS will always decrease as more number of clusters are added (eventually each point becomes its own cluster center).

Upon visualizing the inertia plot in Figure 3.11, even though it seems that the optimal number of clusters range between 3 to 6 clusters, there is no obvious kink in the graph for us to determine the number of clusters to use. This is a common problem faced when using the Elbow method as it does not always produce a distinct kink. As such, we complement this analysis with the Silhouette method.

The Silhouette method measures a Silhouette Score that is calculated as follows:

$$\frac{b-a}{\max(a,b)}$$

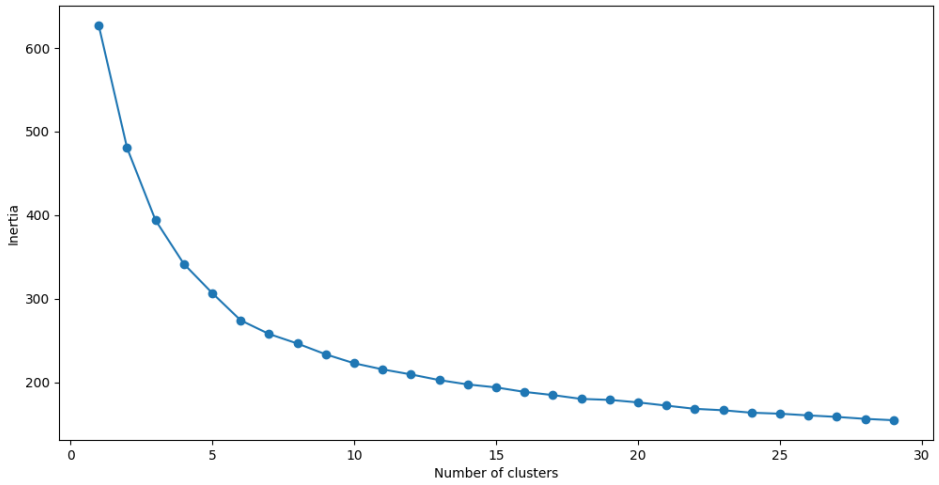



Figure 3.11: Inertia Plot

where $a$ represents the mean intra-cluster distance and $b$ represents the mean nearest-cluster distance for each sample [42]. As a result, the score ranges between -1 and 1, where 1 is the best possible value and values near 0 indicate overlapping clusters. We obtained the following average silhouette scores after running the above calculation for 2 to 6 clusters:

- 2 Clusters: 0.23351
- 3 Clusters: 0.25569
- 4 Clusters: 0.27172
- 5 Clusters: 0.20882
- 6 Clusters: 0.21590

The Silhouette method uses both the average Silhouette score, as well as a visual observation of the size of the clusters to decide on the optimal number of clusters. As we obtained the best average Silhouette scores for 3 and 4 clusters, we only mention an analysis of the plots for these two number of clusters for brevity. With regards to analyzing the Silhouette plots, we prefer to choose the number of clusters

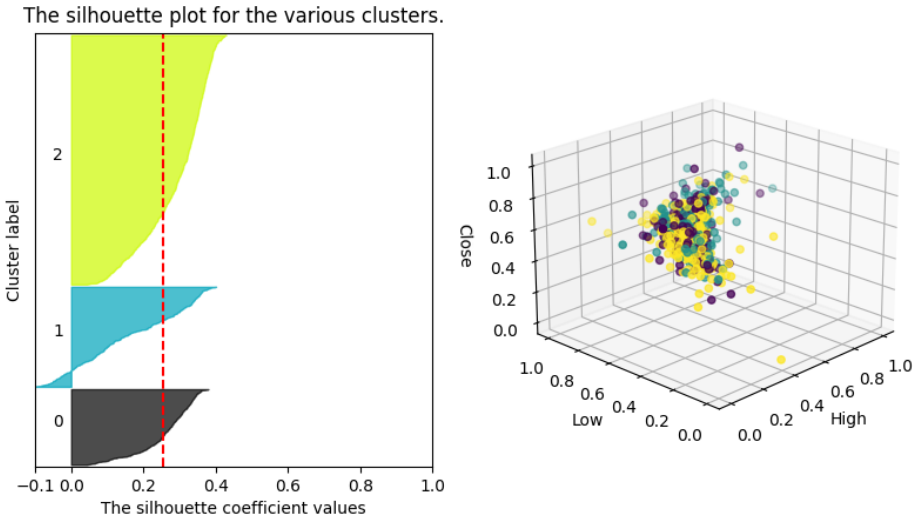



Figure 3.12: Silhouette Plot for 3 Clusters

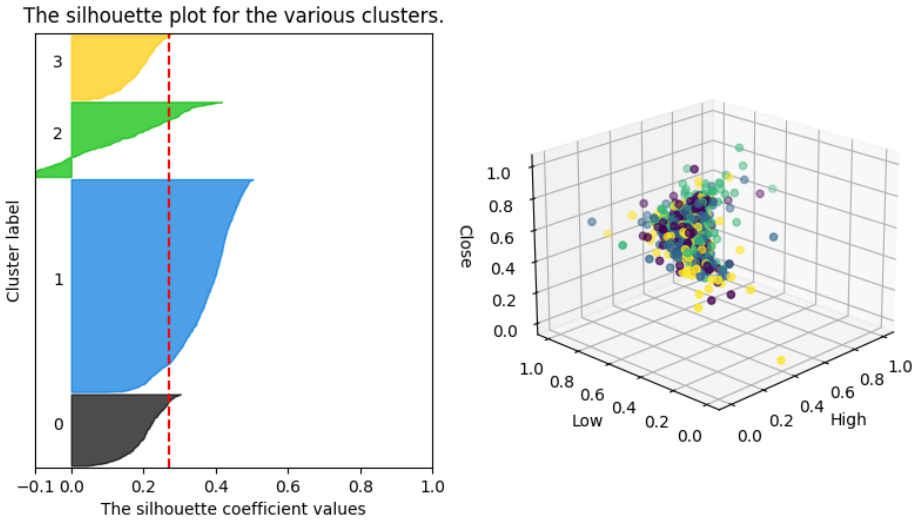



Figure 3.13: Silhouette Plot for 4 Clusters

where each cluster has a more balanced representation and exceed the threshold of the red line, which is the average Silhouette score. As such, while using 4 clusters result in a higher Silhouette score, we chose to go with 3 as the optimum number of clusters. A chart of the clusters superimposed onto the BTC-USD data sample is provided in A.1.

We can now cover the implemented algorithm for the Case Based Reasoner after determining the number of clusters to use for the K-means algorithm. The Case Based Reasoner keeps track of a set of cases, which in this case is the training data, as we do not include the testing data as this would introduce lookahead bias. A table of correctness of predicted results for each feature is also kept in the database which we call the features table. For each new time epoch, we store the current set of predictions provided by the indicators we use (which can be either buy, hold, or sell), and on the next iteration $time_{t+1}$, we calculate the returns and backfill the prediction correctness for $time_t$. We consider a buy signal at $time_t$ to be correct if $y_{t+1}$ is inside the buy bin. On the other hand, we consider a sell signal at $time_t$ to be correct if $y_{t+1}$ is inside the sell bin. For a hold signal at $time_t$, we consider it to be correct if $y_{t+1}$ is inside the hold bin. Note that all return values, which is our dependent variable, is binned according to the following relation:

$$y_t = \begin{cases} 0 \text{ (Sell)}, & \text{if } y_t < \mu_{\text{Close}} - \sigma_{\text{Close}} \times 0.5. \\ 1 \text{ (Hold)}, & \text{if } \mu_{\text{Close}} - \sigma_{\text{Close}} \times 0.5 \leq y_t \leq \mu_{\text{Close}} + \sigma_{\text{Close}} \times 0.5. \\ 2 \text{ (Buy)}, & \text{if } y_t > \mu_{\text{Close}} + \sigma_{\text{Close}} \times 0.5. \end{cases}$$

To obtain the current forecast at time $t$, we use the following formula:

$$forecast = \sum_{i=1}^{N} \frac{signal_{i,0}}{signal_{i,0} + signal_{i,1}} \times indicator_i$$

where N is the number of indicator signals being used, $signal_{i,0}$ indicates the number of times the indicator's signal was correct in the most similar cluster, $signal_{i,1}$ indicates the number of times the indicator's signal was wrong in the most similar cluster, and $indicator_i$ indicates the indicator's signal at the current time $t$. The indicator value is 1 if the signal is to buy, -1 if the signal is to sell, and 0 if the signal is to hold. Note that the signal counts are aggregated across the most similar cluster to the current data sample at time $t$, generated using the K-means algorithm. We provide a sample pseudocode (in the form of comments) for the algorithm written in Python below:

```
class CBRStrategy(Strategy):
    def init(self):
        # 1. initialize all rows of the features table to 0
        # 2. initialize an empty set of predictions (no predictions exist at
        #    initialization)

    def next(self):
        # if data row is in training data set:
            # 1. do updates to keep track of prediction error of indicator
            #    signals
            # 2. generate indicator signal predictions and save this for the
            #    next iteration
        # if data row is in test data set:
            # 1. do updates to keep track of prediction error of indicator
            #    signals
            # 2. generate indicator signal predictions and save this for the
            #    next iteration
            # 3. get related clusters using independent values from current
            #    data row
            # 4. aggregate indicator signal prediction accuracy for each
            #    indicator
            # 5. generate forecast for next time epoch using aggregated
            #    prediction accuracy and current indicator signal prediction

            # 6. if forecast indicates to buy and we do not currently hold a
            #    long position:
                    # issue a buy order of size 1

            # 7. if forecast indicates to sell and we currently hold a
            #    long position:
                    # issue a sell order of size 1
```

Of the price derived features, we use only MA, EMA, and RSI to produce an action signal at each time epoch. For the MA and EMA indicators, the signal to buy is produced when the price at time $t$ is greater than the corresponding indicator value. Conversely, the signal to sell is produced when the price at time $t$ is lesser than the corresponding indicator value. The signal to hold is produced otherwise. For the RSI indicator, we use the aforementioned values of 30 and 70 and to determine

our buy, sell and hold signals. We decide to buy if the value is below 30, to sell if the value is above 70, and to hold otherwise.

### 3.4.2 LightGBM

Before backtesting the LightGBM strategy, we do a few pre-processing steps that are not included for the Case Based Reasoner strategy. When using a model like the LightGBM, we need to do some preliminary exploration with the model first to assess the accuracy of the model and to tune the hyperparameters to try and obtain a model that performs better according to our evaluation metrics. To conduct cross validation for time series, we use `TimeSeriesSplit` instead of K-fold cross validation. We do this to maintain the order of the sample as there is a relational ordering to the samples across time which would be removed if K-fold cross validation is used.

From initial results of fitting the LightGBM model, we obtained very high accuracy scores for the training data set, but much lower accuracy scores for the testing data set:

| **Data Used** | **Accuracy** | |
|---|---|---|
| | **Training Data** | **Testing Data** |
| OHLCV | 0.9341 | 0.4963 |
| OHLCV with Feature Engineering | 0.9995 | 0.5282 |

Table 3.1: Comparison of accuracy for LightGBM with default parameters.

This signalled that we most likely have some overfitting issues present in our model. As a result, when tuning the hyperparameters, we focused on those parameters that are most likely to reduce overfitting. A short list of parameters recommended to focus on is provided in the LightGBM documentation. We incorporated hyperparameter tuning using the `GridSearchCV` from `scikit-learn`, which is an exhaustive search over all parameters provided. This means that the search will involve all possible combinations provided and hence may take a long time to run. An alternative to this is `RandomSearchCV`, which does not exhaustively search the parameter space but randomly searches the parameter space a fixed number of times. Following the guide from the LightGBM documentation, we did hyperparameter tuning across the

parameters and values given in Table 3.2:

| **Parameter** | **Values** |
|---|---|
| num_leaves | [10, 20] |
| num_iterations | [50, 150] |
| max_depth | [4, 5, 6, 7, 8] |
| colsample_bytree | [0.6, 0.8, 1] |
| boosting | ['dart', 'gbdt'] |

Table 3.2: Parameter space tested for GridSearchCV.

After tuning the hyperparameters for the LightGBM model, we obtained the following parameter mapping:

- `boosting_type = 'dart'`
- `num_leaves = 10`
- `max_depth = 4`
- `n_estimators = 50`
- `colsample_bytree = 0.6`

Using these new parameter values, we obtained a training set score of 0.7249 and a testing set score of 0.5474 for the LightGBM model trained on OHLCV with Feature Engineering data. While this is a good improvement from before, it seems that there still is presence of overfitting. Compared to the implementation of the Case Based Reasoning algorithm, the implementation of the LightGBM strategy we use is simpler as we do not need to do bookkeeping for the correctness of the technical indicators used. The dependent variable $y$ is defined using the same bins as explained in the Case Based Reasoner strategy. As before, we provide a sample pseudocode (in the form of comments) for the algorithm written in Python below:

```
class LGBStrategy(Strategy):
    def init(self):
        # 1. fit LightGBM model on training data

    def next(self):
```

```
        # if data row is in training data set:
            # 1. continue to the next iteration, we do not take actions
            #    on data from the training set
        # if data row is in test data set:
            # 1. transform the sample data appropriately
            # 2. generate forecast for next time epoch using predict method

            # 3. if forecast indicates to buy and we do not currently hold a
            #    long position:
                    # issue a buy order of size 1

            # 4. if forecast indicates to sell and we currently hold a
            #    long position:
                    # issue a sell order of size 1
```

### 3.4.3 Results

For all backtests, each strategy was ran with a starting cash value of $100,000 with realistic assumptions of commission fees at 0.2% and margin of 0.5. The typical metrics used in finance and trading to measure the performance of investment and trading strategies are the Sharpe Ratio, Sortino Ratio, Calmar Ratio, Drawdown, Win Rate and Profit Factor. For completeness, we explain these concepts before analyzing the results of the backtests.

#### 3.4.3.1 Sharpe Ratio

The Sharpe Ratio is a highly used, and studied, ratio used to assess the risk-adjusted performance of a strategy [40]. It is defined as:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

where $R_p$ and $\sigma_p$ represent the return and standard deviation of a particular portfolio, or strategy, respectively, and $R_f$ represents the risk-free rate. As such, an intuitive understanding of the Sharpe Ratio is that it measures the additional risk-adjusted returns provided by a given strategy over the risk-free return rate. In general, a higher Sharpe ratio is always better. Usually, a Sharpe ratio between 1 and 2 is

considered good and anything above 2 is considered very good.

#### 3.4.3.2 Sortino Ratio

The Sortino Ratio is a variation of the Sharpe ratio that uses downside standard deviation instead of both upside and downside to calculate the risk-adjusted returns of strategy [44]. It is defined as:

$$Sortino\ Ratio = \frac{R_p - R_f}{\sigma_d}$$

where $R_p$ and $\sigma_d$ represent the return and downside standard deviation of a particular portfolio, or strategy, respectively, and $R_f$ represents the risk-free rate. Similar to the Sharpe ratio, a higher Sortino ratio is typically always preferred. Intuitively, the Sortino ratio gives an indication of the additional returns of a portfolio for each unit of downside risk taken on.

#### 3.4.3.3 Calmar Ratio

The Calmar Ratio is another measure used to measure risk-adjusted performance of a portfolio. It is defined as [7]:

$$Calmar\ Ratio = \frac{R_p - R_f}{Maximum\ Drawdown}$$

where $R_p$ and $Maximum\ Drawdown$ represent the return and maximum drawdown of a particular portfolio, or strategy, respectively, and $R_f$ represents the risk-free rate. Similar to the two previous ratios, a higher Calmar ratio is typically always preferred as it means that the returns made on the portfolio is less susceptible to drawdowns.

#### 3.4.3.4 Drawdown

The Drawdown is a commonly used metric to determine the decrease in value of a portfolio, from one of its peak to one of its trough. Typically, investors are concerned with the maximum drawdown, which indicates the biggest swing in value of a portfolio, or strategy. A lower maximum drawdown is better as it means that the investment value is less volatile.

#### 3.4.3.5 Win Rate

The Win Rate is a simple measure of the proportion of winning trades as compared to the total number of trades made where a winning trade is defined as a trade that is profitable. It is defined as:

$$Win\ Rate = \frac{Winning\ Trades}{Total\ Trades}$$

As expected, a higher win rate is usually preferred as it means that your strategy is making profitable trades on a higher basis.

#### 3.4.3.6 Profit Factor

The Profit Factor is a measure of the profit gained per dollar risked in a certain strategy [38]. It is defined as:

$$Profit\ Factor = \frac{Total\ Value\ of\ Winning\ Trades}{Total\ Value\ of\ Losing\ Trades}$$

Intuitively, for each dollar risked in a strategy, the profit factor tells us how much we would be earning in terms of profit. As such, a higher profit factor is always preferred.

#### 3.4.3.7 Results Analysis

In Table 3.3, we highlight in green the best performing combination of model and data for each relevant metric measured. We obtained a best return of 40.29% using the LightGBM model with OHLCV data. This appears to be a very good return on investment, however, if we compare it to the return of the buy-and-hold strategy across the test period which was 136.72%, 40.29% seems to pale in comparison. This is perhaps due to the large worst trade (%) where a single trade caused a lost of 35.67%. Nonetheless, this particular combination does exhibit some good characteristics. Its volatility (%) is lowest amongst the tested combinations and has a very high win rate of 75%. This suggests that further tuning of this model combination should look at reducing the impact of negative returns. Compared to these results, the Case Based Reasoner ran using the same OHLCV data did

perform decently well when compared using total returns (36.6%), however, it has a much lower win rate. Across exposure time, best trade (%), and worst trade (%), the Case Based Reasoning strategy strikes a seemingly good balance as compared to the LightGBM strategy. This finding suggests that the technical indicators used do provide some value in helping interpret the environment for the Case Based Reasoning algorithm and agrees with findings from [17] whom found that using technical trading rules reduced the volatility risk experienced by traders.

Most notably, those additional features that we added in the feature engineering section did not seem to improve the backtesting performances of the models by a significant margin, contrary to our literature review of papers that highlighted statistically significant links between Bitcoin's price and these features. This suggests that perhaps other methods of data transformation could be trialed to try and obtain better results. We tried using Principal Component Analysis (PCA) to assess if dimensionality reduction would result in better results. From Figure 3.14, we observe that roughly 80% of the variance is explained by 5 components, and roughly 90% of the variance is explained 9 components. We chose to use 15 components to trial the PCA method as by about 15 components, there is marginal loss in variance by not including more components. Our results from these backtests, however, were not worth noting as they did not perform better than the previously listed combinations. Perhaps the additional price-derived features can be included as a signal value, similar to the features table used in the Case Based Reasoner, as this transformation might be more informative than the normalized values used. This will have to be tested in future works.

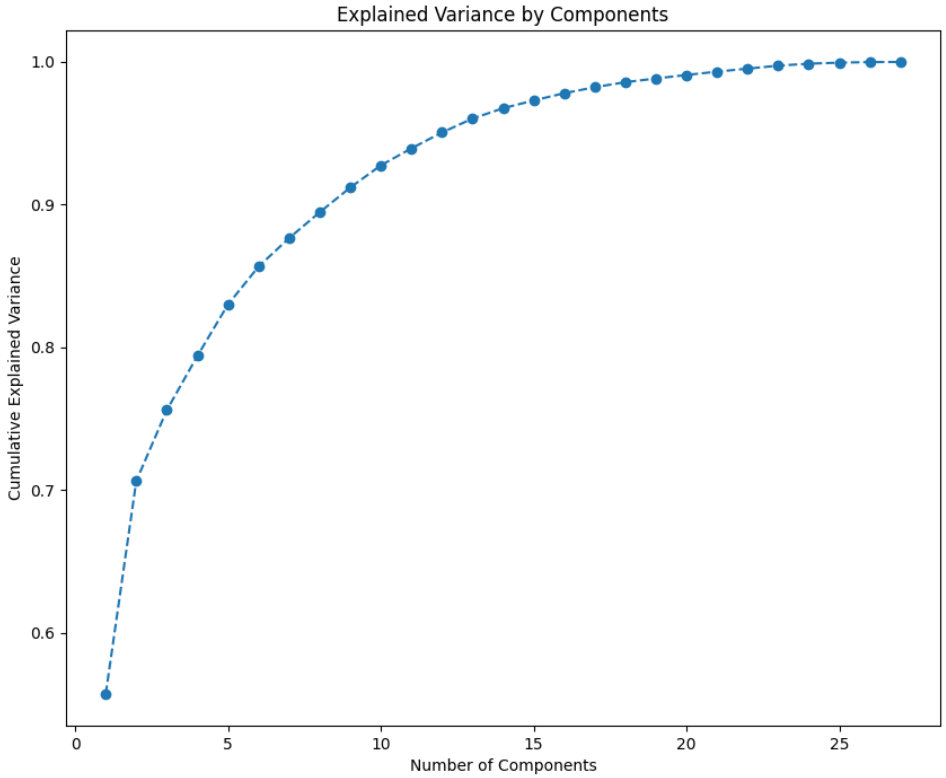



Figure 3.14: PCA Plot

| Model | LightGBM | Case Based Reasoner | LightGBM | LightGBM (tuned) | Case Based Reasoner |
|---|---|---|---|---|---|
| **Data Used** | OHLCV | OHLCV | OHLCV (w F.E.) | OHLCV (w F.E.) | OHLCV (w F.E.) |
| **Exposure Time (%)** | 14.65 | 16.186 | 16.76 | 14.92 | 16.44 |
| **Equity Final ($)** | 140290.00 | 136599.87 | 138839.69 | 103188.19 | 134212.49 |
| **Equity Peak ($)** | 153120.90 | 148296.31 | 146184.87 | 138965.31 | 145908.92 |
| **Return (%)** | 40.29 | 36.60 | 38.84 | 3.19 | 34.21 |
| **Return (Ann.) (%)** | 3.95 | 3.64 | 4.41 | 0.41 | 3.95 |
| **Volatility (Ann.) (%)** | 7.31 | 7.63 | 7.74 | 9.23 | 8.32 |
| **Sharpe Ratio** | 0.54 | 0.48 | 0.57 | 0.04 | 0.47 |
| **Sortino Ratio** | 0.87 | 0.72 | 0.94 | 0.06 | 0.72 |
| **Calmar Ratio** | 0.34 | 0.22 | 0.30 | 0.02 | 0.24 |
| **Max. Drawdown (%)** | -11.64 | -16.35 | -14.91 | -27.01 | -16.63 |
| **Avg. Drawdown (%)** | -1.61 | -1.60 | -1.19 | -1.94 | -2.05 |
| **# Trades** | 44 | 54 | 38 | 19 | 50 |
| **Win Rate (%)** | 75.00 | 42.60 | 73.68 | 63.16 | 42.00 |
| **Best Trade (%)** | 27.95 | 55.90 | 41.27 | 92.91 | 55.90 |
| **Worst Trade (%)** | -35.67 | -16.45 | -25.75 | -33.05 | -16.45 |
| **Profit Factor** | 3.01 | 2.76 | 4.03 | 2.38 | 2.53 |

Table 3.3: Backtest Results.

# Chapter 4

# Conclusion and Future Work

## 4.1 Conclusion

In this capstone project, we provided an empirical study of the usage of technical indicators to inform trading strategies, as well as developed a Cased Based Reasoning algorithm that exhibits certain positive characteristics for trading such as having the lowest worst trade (%) amongst all tested model and data combinations. We also introduced a working and deployable prototype of the Case Based Reasoner with other agents such as the database helper to fully work in a production setting. As well, we incorporated a state of the art GBDT framework in the form of LightGBM to compare and contrast performance with our initial Case Based Reasoning strategy.

From the results of the backtests, it seems reasonable to conclude that the usage of momentum and trend based indicators do provide some useful information to the underlying models for determining a direction to make profitable trades. On the other hand, it seems that the features that we included from other studies which are heavily suggested to have an impact on Bitcoin price do not exhibit as much of an impact on information. However, other data transformations such as using log-transformations should be statistically tested first before coming to such a conclusion. Future studies should also focus on assessing the statistical significance of the aforementioned features across different historical time frames as this could explain why certain studies find the features statistically significant.

## 4.2 Future Research Directions

There are many avenues for future works to trial and experiment with and we discuss these potential approaches in this section.

### 4.2.1 Model and Hyperparameter Tuning

Hyperparameter tuning is one of the most important activities when working on machine learning models as this process tailors the machine learning algorithm to the specific requirements of the task. While we explored tuning the hyperparameters for the LightGBM as in Table 3.2 using GridSearchCV, there are still other optimizers that can be trialed such as Bayesian Optimization which can found in the Hyperopt package. Additionally, more parameter values could be tested to see if there is a better fit in terms of the combination of parameters to construct the model.

Aside from the hyperparameters, the values used in the technical indicators could also be optimized. For example, by default, we only try 30 and 70 for the lower and upper bound of RSI respectively. Further research in this area could try to optimize the different indicator parameter values for different regimes, and as well, the optimal number of periods to use for each technical indicator. For future research, we also propose the idea of walk-forward optimization to re-train the models every $n$ number of periods. This could be initiated when regimes change.

### 4.2.2 Inclusion of other sources of data

Bitcoin and other cryptocurrencies are unlike traditional stocks. As such, researchers might have to be more creative in their approach when creating trading strategies for these alternative assets. For example, while fundamental data in the form of balance sheets or statement of incomes do not exist, other types of crypto-fundamental data do exist in the form of transactions per block, miners revenue (%), hash rates, and median confirmation time amongst other metrics that can be measured [21]. As such, the inclusion of this alternative source of data might

help in modelling the price movement of Bitcoin, as well as provide the Case Based Reasoner with a better sense of the financial and trading environment at each time epoch. Other studies surveyed by Khedr et al. [24] have also suggested to include data from exchanges offering liquidity for Bitcoin in addition to the native blockchain data.

Apart from blockchain native data, future works should also seek to include text-based sentiment data as much of the crypto-native world thrives on communication through publicly available channels such as Twitter and Reddit. For example, Li et al. [28] experimented using sentiment-based data to predict cryptocurrency prices using a similar GBDT model and found promising results. Similar findings on the predictive ability of Twitter sentiment were found in [15]. Furthermore, future work that incorporate sentiment-based analysis could then include event-based trading strategies as tweets by famous people such as Elon Musk, or Jim Cramer, could serve as event triggers.

### 4.2.3 Unbalanced dependent variable data

In this project, using the strategy of binning the response variable according to:

$$y_t = \begin{cases} 0 \text{ (Sell)}, & \text{if } y_t < \mu_{\text{Close}} - \sigma_{\text{Close}} \times 0.5. \\ 1 \text{ (Hold)}, & \text{if } \mu_{\text{Close}} - \sigma_{\text{Close}} \times 0.5 \leq y_t \leq \mu_{\text{Close}} + \sigma_{\text{Close}} \times 0.5. \\ 2 \text{ (Buy)}, & \text{if } y_t > \mu_{\text{Close}} + \sigma_{\text{Close}} \times 0.5. \end{cases}$$

resulted in ground truth counts of 675 sell values, 667 buy values, and 1846 hold values. This resulted in a rough distribution of 20% sell values, 20% buy values, and 60% hold values which is considerably more uneven than a preferred $\frac{1}{3}$ split between all three potential values. Future work could include using sampling strategies to make the sample data more even by creating synthetic data. As well, the binning multiplier of 0.5 by the standard deviation for each direction could be adjusted using a sensitivity analysis.

### 4.2.4 Clustering Techniques

While we only managed to explore K-means clustering fully with the time given for the project, other clustering techniques exist and should be experimented with to see if they produce better backtesting results. Other clustering algorithms such as DBSCAN utilize different clustering ideas. For example, K-means utilizes a centroid-based clustering technique typically using Euclidean distance as a measure of distance between different points, whereas DBSCAN uses density based spatial clustering. As a result, using these two algorithms can result in different clusters depending on the distribution of the sample data.

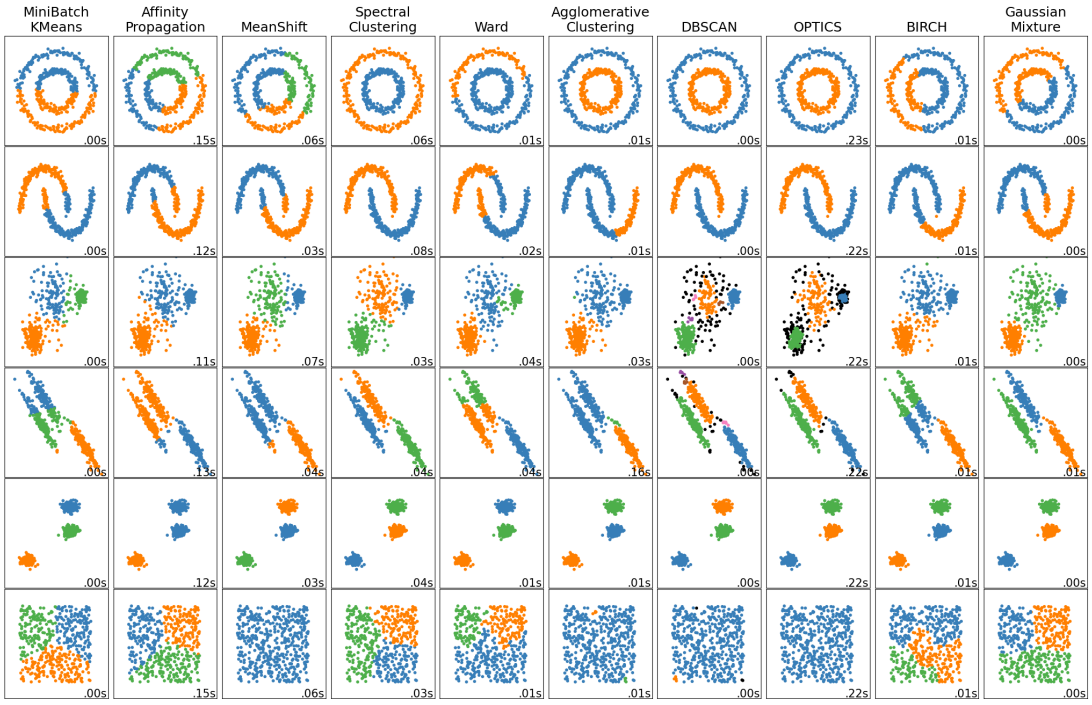



Figure 4.1: Comparison of different Clustering Algorithm [1]

Taking a step back, philosophically, the usage of clustering in the Case Based Reasoner can be thought of as a form of regime detection where we try to say that in a particular regime, certain technical indicators and signals work better than others. As such, it makes logical sense for future work to attempt to find better ways of doing regime detection such as through Hidden Markov Models (HMM) [16].

### 4.2.5 Machine Learning Models

Neural networks and deep learning have been shown to provide moderate success in predicting price movements. For example, Chen et al. [9] in their study on machine learning and statistical methods found that the machine learning models outperformed statistical methods in terms of accuracy when used to predict the Bitcoin price in 5-minute intervals. Additionally, in their study, a Long Short-Term Memory (LSTM) neural network performed the best amongst the machine learning models. LSTM models are also discussed to have better predictive power for financial time-series problems as compared to non-gated Recurrent Neural Networks (RNN) as they have the ability to selectively pay attention to different sets of information across time [9, 50]. Hence, we highly recommend any future work to incorporate an LSTM based signal for predicting price fluctuations.

### 4.2.6 Portfolio Optimization and Execution Logic

It would be disingenuous to represent algorithmic trading only as a problem of creating the best model to forecast future prices. There are many other factors that go into an algorithmic trading problem. Firstly, to narrow the scope for this project, we only allow for the selection of one single asset to trade. More often than not, investors and traders alike will want to include a universe of stocks and alternative assets to base their portfolio on. As such, to be able to effectively choose between a selection of potential financial assets, portfolio optimization techniques will need to be employed to decide the ideal allocation of portfolio value to each different asset.

As well, in this project, we assume no slippages of prices and that our trading strategy does not affect the market. This assumption is usually fine, however, for those working in a quantitative firm with very large amounts of assets under management (AUM), care will have to be taken when moving in and out of the markets so as to affect price as little as possible.

Apart from these aforementioned points on potential future research on portfolio optimization and execution logic, future research could also look to allow for a

variety of order types typically offered by trading exchanges such as limit orders and going short on a stock. For simplicity, this project assumes that we can only submit market orders to go long and buy a single unit of Bitcoin, or to sell an existing unit of Bitcoin that we already own.

### 4.2.7 Combination of Case Based Reasoner and LightGBM

From the results obtained by backtesting the Case Based Reasoner and LightGBM strategies, we hypothesize that a combination of the two strategies could provide better results than when used individually. For example, we could include the output forecast of the LightGBM strategy as a signal akin to the technical indicators used in the Case Based Reasoner and aggregate across all signals as such. This could potentially result in a reduction in the percentage of the worst trade and volatility level as seen in the LightGBM strategy and complement the Case Base Reasoner with the more promising returns and ratios obtained by the LightGBM strategy.

This would result in an overall model that works similar to what is described as a stacked model in [43] where each model's prediction is used as a feature in a secondary level model.

# Bibliography

[1] "2.3. Clustering", en. [Online]. Available: https://scikit-learn/stable/modules/clustering.html (visited on 11/22/2022).

[2] "About Feature Scaling and Normalization", en, Sep. 2020. [Online]. Available: https://johdev.com/machine%20learning/2020/09/19/About-Feature-Scaling-and-Normalization.html (visited on 12/02/2022).

[3] admin, "Medallion Fund: The Ultimate Counterexample?" en, Feb. 2020. [Online]. Available: https://www.cornell-capital.com/blog/2020/02/medallion-fund-the-ultimate-counterexample.html (visited on 11/29/2022).

[4] W. Antweiler and M. Z. Frank, "Is All That Talk Just Noise? The Information Content of Internet Stock Message Boards", en, The Journal of Finance, p. 36,

[5] Z. Ben Ami and R. Feldman, "Event-Based Trading: Building Superior Trading Strategies with State-of-the-Art Information Extraction Tools", en, SSRN Scholarly Paper, Rochester, NY, Jan. 2017. [Online]. Available: https://papers.ssrn.com/abstract=2907600 (visited on 09/11/2022).

[6] A. Bhandari, "Feature Scaling | Standardization Vs Normalization", en, Apr. 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/ (visited on 12/02/2022).

[7] "Calmar Ratio", en. [Online]. Available: https://www.investopedia.com/terms/c/calmarratio.asp (visited on 12/03/2022).

[8] "CAPM: Theory, advantages, and disadvantages | F9 Financial Management | ACCA Qualification | Students | ACCA Global", [Online]. Available: https://www.accaglobal.com/gb/en/student/exam-support-resources/fundamentals-exams-study-resources/f9/technical-articles/CAPM-theory.html (visited on 11/23/2022).

[9] Z. Chen, C. Li, and W. Sun, "Bitcoin price prediction using machine learning: An approach to sample dimension engineering", en, Journal of Computational and Applied Mathematics, vol. 365, p. 112 395, Feb. 2020, ISSN: 03770427. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S037704271930398X (visited on 12/03/2022).

[10] T. Conlon and R. McGee, "Safe haven or risky hazard? Bitcoin during the Covid-19 bear market", en, Finance Research Letters, vol. 35, p. 101 607, Jul. 2020, ISSN: 15446123. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1544612320304244 (visited on 11/29/2022).

[11] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work", The Journal of Finance, vol. 25, no. 2, pp. 383–417, 1970, Publisher: [American Finance Association, Wiley], ISSN: 0022-1082. [Online]. Available: http://www.jstor.org/stable/2325486 (visited on 11/23/2022).

[12] F. Fang, C. Ventre, M. Basios, L. Kanthan, D. Martinez-Rego, F. Wu, and L. Li, "Cryptocurrency trading: A comprehensive survey", Financial Innovation, vol. 8, no. 1, p. 13, Feb. 2022, ISSN: 2199-4730. [Online]. Available: https://doi.org/10.1186/s40854-021-00321-6 (visited on 09/13/2022).

[13] D. Garcia and F. Schweitzer, "Social signals and algorithmic trading of Bitcoin", Royal Society Open Science, vol. 2, no. 9, p. 150 288, Publisher: Royal Society. [Online]. Available: https://royalsocietypublishing.org/doi/full/10.1098/rsos.150288 (visited on 09/12/2022).

[14] K. Grobys, S. Ahmed, and N. Sapkota, "Technical trading rules in the cryptocurrency market", en, Finance Research Letters, vol. 32, p. 101 396, Jan. 2020, ISSN: 15446123. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1544612319308852 (visited on 11/30/2022).

[15] A. Groß-Klußmann, S. König, and M. Ebner, "Buzzwords build momentum: Global financial Twitter sentiment and the aggregate stock market", en, Expert Systems with Applications, vol. 136, pp. 171–186, Dec. 2019, ISSN: 09574174. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417419304270 (visited on 12/03/2022).

[16] "Hidden Markov Models - An Introduction | QuantStart", [Online]. Available: https://www.quantstart.com/articles/hidden-markov-models-an-introduction/ (visited on 12/03/2022).

[17] R. Hudson and A. Urquhart, "Technical trading and cryptocurrencies", en, Annals of Operations Research, vol. 297, no. 1-2, pp. 191–220, Feb. 2021, ISSN: 0254-5330, 1572-9338. [Online]. Available: http://link.springer.com/10.1007/s10479-019-03357-1 (visited on 11/30/2022).

[18] D. Ikenberry, R. Shockley, and K. Womack, "Why Active Fund Managers Often Underperform the S&P 500: The Impact of Size and Skewness", The Journal of Wealth Management, vol. 1, pp. 13–26, Jan. 1998.

[19] "In Depth: K-Means Clustering | Python Data Science Handbook", [Online]. Available: https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html (visited on 11/26/2022).

[20] O. Jan, "CAPM - assumptions, limitations and SML – Portfolio Management", en-US, Nov. 2020. [Online]. Available: https://alphabetaprep.com/cfa-level-1/portfolio-management/capm-assumption-limitions/ (visited on 11/23/2022).

[21] H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices With Bayesian Neural Networks Based on Blockchain Information", IEEE Access, vol. 6, pp. 5427–5437, 2018, Conference Name: IEEE Access, ISSN: 2169-3536.

[22] "$k$-means clustering", en, Page Version ID: 1123233551, Nov. 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=1123233551 (visited on 11/29/2022).

[23] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", in Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html (visited on 11/29/2022).

[24] A. M. Khedr, I. Arif, P. R. P V, M. El-Bannany, S. M. Alhashmi, and M. Sreedharan, "Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey", en, Intelligent Systems in Accounting, Finance and Management, vol. 28, no. 1, pp. 3–34, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/isaf.1488, ISSN: 1099-1174. [Online]. Available: http://onlinelibrary.wiley.com/doi/abs/10.1002/isaf.1488 (visited on 09/13/2022).

[25] J. L. Kolodner, "An introduction to case-based reasoning", en, p. 32,

[26] b. K. Kotzé, "Nonstationarity", [Online]. Available: https://kevinkotze.github.io/ts-6-unit-roots/ (visited on 11/30/2022).

[27] Y. Kurihara and A. Fukushima, "The Market Efficiency of Bitcoin: A Weekly Anomaly Perspective", en, p. 8,

[28] T. R. Li, A. S. Chamrajnagar, X. R. Fong, N. R. Rizik, and F. Fu, "Sentiment-Based Prediction of Alternative Cryptocurrency Price Fluctuations Using Gradient Boosting Tree Model", Frontiers in Physics,

vol. 7, 2019, ISSN: 2296-424X. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fphy.2019.00098 (visited on 09/13/2022).

[29] "LightGBM Classifier in Python", en. [Online]. Available: https://kaggle.com/code/prashant111/lightgbm-classifier-in-python (visited on 11/30/2022).

[30] B. G. Malkiel, "Efficient Market Hypothesis", en, in Finance, ser. The New Palgrave, J. Eatwell, M. Milgate, and P. Newman, Eds., London: Palgrave Macmillan UK, 1989, pp. 127–134, ISBN: 978-1-349-20213-3. [Online]. Available: https://doi.org/10.1007/978-1-349-20213-3_13 (visited on 11/23/2022).

[31] "Moving Average (MA): Purpose, Uses, Formula, and Examples", en. [Online]. Available: https://www.investopedia.com/terms/m/movingaverage.asp (visited on 12/01/2022).

[32] "Moving Averages - Simple and Exponential [ChartSchool]", [Online]. Available: https://school.stockcharts.com/doku.php?id=technical_indicators:moving_averages (visited on 12/01/2022).

[33] "MySQL vs PostgreSQL – Choose the Right Database for Your Project", en. [Online]. Available: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres (visited on 11/28/2022).

[34] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", en, p. 9,

[35] A. Noda, "On the Evolution of Cryptocurrency Market Efficiency", Applied Economics Letters, vol. 28, no. 6, pp. 433–439, Mar. 2021, arXiv:1904.09403 [q-fin], ISSN: 1350-4851, 1466-4291. [Online]. Available: http://arxiv.org/abs/1904.09403 (visited on 11/29/2022).

[36] "PostgreSQL vs. MySQL: What you need to know | Blog | Fivetran", en. [Online]. Available: https://www.fivetran.com/blog/postgresql-vs-mysql (visited on 11/28/2022).

[37] S. Prabhakaran, "Time Series Analysis in Python - A Comprehensive Guide with Examples - ML+", en-US, Feb. 2019. [Online]. Available: https://www.machinelearningplus.com/time-series/time-series-analysis-python/ (visited on 11/30/2022).

[38] "Profit Factor: The Ultimate Guide with Examples - Analyzing Alpha", en-us, Section: Testing and Performance, Jun. 2021. [Online]. Available: https://analyzingalpha.com/profit-factor (visited on 12/03/2022).

[39] "Relative Strength Index (RSI) [ChartSchool]", [Online]. Available: https://school.stockcharts.com/doku.php?id=technical_indicators:relative_strength_index_rsi (visited on 12/02/2022).

[40] "Sharpe Ratio Formula and Definition With Examples", en. [Online]. Available: https://www.investopedia.com/terms/s/sharperatio.asp (visited on 12/03/2022).

[41] H. Shi, "Best-first Decision Tree Learning", en, p. 121,

[42] "Sklearn.metrics.silhouette_score", en. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.metrics.silhouette_score.html (visited on 12/03/2022).

[43] D. Snow, "Machine Learning in Asset Management— *Part 1* : *Portfolio Construction—Trading Strategies*", en, The Journal of Financial Data Science, vol. 2, no. 1, pp. 10–23, Jan. 2020, ISSN: 2640-3943. [Online]. Available: http://jfds.pm-research.com/lookup/doi/10.3905/jfds.2019.1.021 (visited on 11/29/2022).

[44] "Sortino Ratio: Definition, Formula, Calculation, and Example", en. [Online]. Available: https://www.investopedia.com/terms/s/sortinoratio.asp (visited on 12/03/2022).

[45] Y. Sovbetov, "Factors Influencing Cryptocurrency Prices: Evidence from Bitcoin, Ethereum, Dash, Litcoin, and Monero", en, vol. 2, no. 2, p. 28, 2018.

[46] "Stationarity in Time Series Analysis Explained using Python", en, Feb. 2021. [Online]. Available: https://blog.quantinsti.com/stationarity/ (visited on 11/30/2022).

[47] "The Capital Asset Pricing Model: What You Need to Know", [Online]. Available: https://crypto.com/university/portfolio-management-like-a-pro-the-capital-asset-pricing-model (visited on 11/29/2022).

[48] T. Théate and D. Ernst, "An Application of Deep Reinforcement Learning to Algorithmic Trading", Expert Systems with Applications, vol. 173, p. 114 632, Jul. 2021, arXiv:2004.06627 [cs, q-fin], ISSN: 09574174. [Online]. Available: http://arxiv.org/abs/2004.06627 (visited on 11/30/2022).

[49] A. Timmermann and C. W. Granger, "Efficient market hypothesis and forecasting", en, International Journal of Forecasting, vol. 20, no. 1, pp. 15–27, Jan. 2004, ISSN: 01692070. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0169207003000128 (visited on 11/23/2022).

[50] "Understanding LSTM Networks – colah's blog", [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 12/03/2022).

[51] "Welcome to LightGBM's documentation! — LightGBM 3.3.3.99 documentation", [Online]. Available: https://lightgbm.readthedocs.io/en/latest/index.html (visited on 11/30/2022).

[52] "What is Clustering? | Machine Learning", en. [Online]. Available: https://developers.google.com/machine-learning/clustering/overview (visited on 11/26/2022).

[53] "What is Overfitting?" en-us, Mar. 2021. [Online]. Available: https://www.ibm.com/cloud/learn/overfitting (visited on 11/29/2022).

[54] "What is Overfitting? - Overfitting - AWS", en-US. [Online]. Available: https://aws.amazon.com/what-is/overfitting/ (visited on 11/29/2022).

[55] "Why Active Managers Have Trouble Keeping Up with the Pack", en. [Online]. Available: https://www.chicagobooth.edu/review/why-active-managers-have-trouble-keeping-up-with-the-pack (visited on 11/29/2022).

[56] A. Woodie, "Data Prep Still Dominates Data Scientists' Time, Survey Finds", Jul. 2020. [Online]. Available: https://www.datanami.com/2020/07/06/data-prep-still-dominates-data-scientists-time-survey-finds/ (visited on 11/27/2022).

# Appendix A

# Methodology

## A.1 BTC-USD Clusters

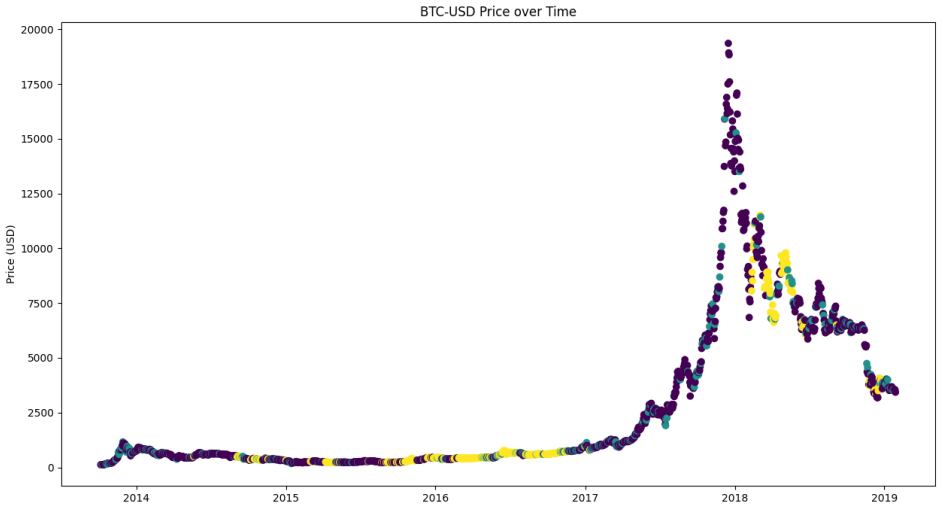



Figure A.1: BTC-USD data sample clustered using 3 clusters

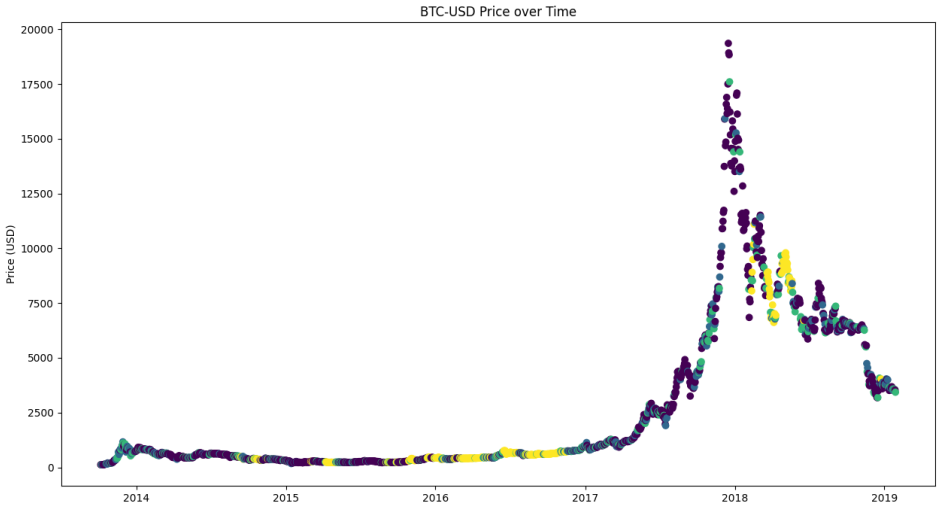



Figure A.2: BTC-USD data sample clustered using 4 clusters

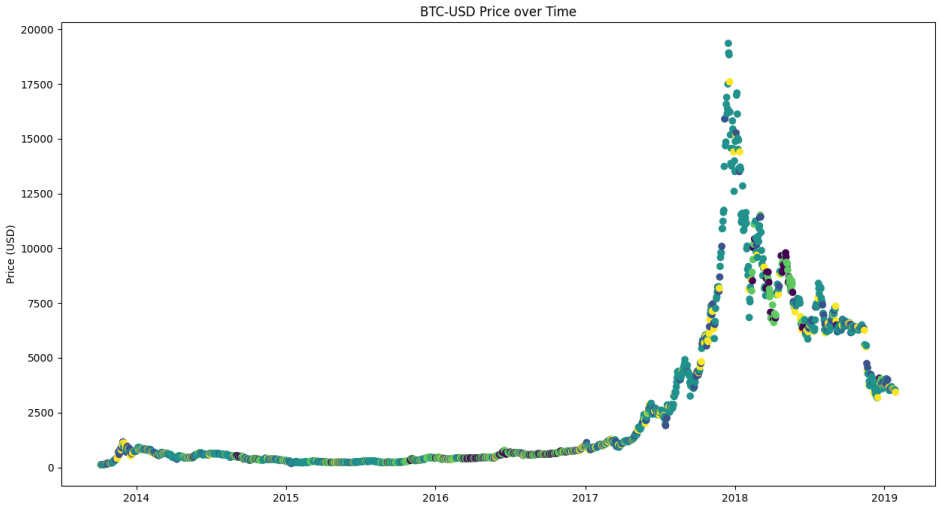



Figure A.3: BTC-USD data sample clustered using 5 clusters