CHARLES UNIVERSITY
FACULTY OF SOCIAL SCIENCES
Institute of Economic Studies

Noise reduction and feature extraction with principal component analysis for cryptocurrency price modeling

Bachelor's thesis

Author: Tomáš Barhoň

Study program: Economics and Finance

Supervisor: prof. PhDr. Ladislav Krištoufek Ph.D.

Year of defense: 2025

Declaration of Authorship

The author hereby declares that he compiled this thesis independently, using only the listed resources and literature, and the thesis has not been used to obtain any other academic title.

The author grants to Charles University permission to reproduce and to distribute copies of this thesis in whole or in part and agrees with the thesis being used for study and scientific purposes.

Prague, April 23, 2025

Tomas Barhon

Abstract

The abstract should concisely summarize the contents of a thesis. Since potential readers should be able to make their decision on the personal relevance based on the abstract, the abstract should clearly tell the reader what information he can expect to find in the thesis. The most essential issue is the problem statement and the actual contribution of described work. The authors should always keep in mind that the abstract is the most frequently read part of a thesis. It should contain at least 70 and at most 120 words (200 when you are writing a thesis). Do not cite anyone in the abstract.

JEL Classification	C01, G00, F23, H25, H71, H87
Keywords	Cryptocurrency, Bitcoin, Ethereum, Litecoin, Machine Learning, PCA, Noise Reduction

Title	Noise reduction and feature extraction with principal component analysis for cryptocurrency price modeling

Author's e-mail	tomas.barhon@hotmail.cz
Supervisor's e-mail	ladislav.kristoufek@fsv.cuni.cz

Abstrakt

Nutnou sout je anotace, kterrnuje vznam pr a vsledky v nsaennotace pr by nem bt del 200 slov a p se v jazyce pr (tj. ky, slovensky nglicky) a v pekladu (tj. u anglicky psan ky lovensky, u ky lovensky psan anglicky). Anotace pr by nem bt del 200 slov a p se v jazyce pr (tj. ky, slovensky nglicky) a v pekladu (tj. u anglicky psan ky lovensky, u ky lovensky psan anglicky). V abstraktu by se nem citovat.

Klasifikace JEL	C01, G00, F23, H25, H71, H87
Klvova	Kryptoměny, Bitcoin, Ethereum, Litecoin, Strojové učení, PCA, Redukce šumu

Nv pr	Redukce šumu a extrakce rysů pomocí analýzy hlavních komponent pro modelování cen kryptoměn

E-mail autora	tomas.barhon@hotmail.cz
E-mail vedouc pr	ladislav.kristoufek@fsv.cuni.cz

Acknowledgments

The author is grateful especially to prof. PhDr. Ladislav Krištoufek Ph.D..

Typeset in FSV LaTeX template with great thanks to prof. Zuzana Havrankova and prof. Tomas Havranek of Institute of Economic Studies, Faculty of Social Sciences, Charles University.

Bibliographic Record

Barhoň, Tomáš: *Noise reduction and feature extraction with principal component analysis for cryptocurrency price modeling.* Bachelor's thesis. Charles University, Faculty of Social Sciences, Institute of Economic Studies, Prague. 2025, pages 75. Advisor: prof. PhDr. Ladislav Krištoufek Ph.D.

Contents

List of Tables

List of Figures

Acronyms

BTC	Bitcoin
ETH	Ethereum
LTC	Litecoin
AI	Artificial Intelligence
ML	Machine Learing
DL	Deep Learing
ANN	Artificial Neural Network
SGD	Stochastic Gradient Descent
LR	Linear Regression
SVM	Support Vector Machines
SVR	Support Vector Regression
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
PCA	Principal Component Analysis
SVD	Support Vector Decomposition
ARIMA	Autoregressive Integrated Moving Average
PoW	Proof of Work

Bachelor's Thesis Proposal

Author	Tomáš Barhoň
Supervisor	prof. PhDr. Ladislav Krištoufek Ph.D.
Proposed topic	Noise reduction and feature extraction with principal component analysis for cryptocurrency price modeling

Motivation Crypto assets have always been exceptionally volatile compared to traditional assets such as stocks or gold. The historical window is relatively short, thus modeling their price or volatility proposes quite a difficult challenge. It is generally believed that noise in any data decreases the precision of predictions. This effect might be reduced, which will improve the performance of traditional models that are used for cryptocurrency price modeling.

The main motivation for researching this topic is that there is still an ongoing discussion about the role of different features in crypto pricing dynamics. (Kukacka; Kristoufek 2023) have shown that a lot of the pricing dynamic emerges from complex interactions between fundamental and speculative components. They also show the different correlations between all of the explanatory variables which have a direct connection to principal component analysis. It is crucial to study the real impact of those variables in different models as many of them might turn out to be obsolete.

There is currently little use of this dimensionality reduction technique in the academic literature about cryptocurrencies. However, for more traditional financial series this technique is already quite established as a preprocessing technique to reduce noise and dimensionality from which financial data inherently suffer (Chowdhury, U. ; Chakravarty, S. and Hossain, M. 2018). Moreover (Bouri, E.; Kristoufek, L.; Ahmad, T. et al. 2022) studied the effect of microstructural noise on idiosyncratic volatility in cryptocurrencies which further supports the need for a technique that will mitigate this effect on the predictions.

The research will address the problem of variable selection for different types of predictive models with respect to the analysis of the principle components aiming to reduce the dimensionality and simultaneously increase precision. The second question is whether it is more appropriate to transform the high dimensionality with PCA into lower dimensionality or simply omit the variables with high multicollinearity from the models. These approaches are fundamentally different and the answer is not clear.

Methodology The data will come from various sources because the aim is to look at all the possible variables even if they might not seem useful at first glance. As already mentioned the dynamic is driven by a lot of completely different effects. Most of it will be collected from: coinmetrics.io, studio.glassnode.com, and for macroeconomic indicators https://fred.stlouisfed.org/. Some of the data might need to be interpolated to daily observations. Lastly, the observations will need to be sliced to different window sizes and shifted by one so that the predictions can be made for the next day with the data available on that day.

Afterward, the multicollinearity in the data will be examined and different approaches to solve it will be used. The two main ones are using only a smaller set of uncorrelated variables (simple dimensionality reduction) and the second being employing PCA transformation to preserve a predefined threshold of variance or directly targeting the number of principal components.

All the different setups will be compared across three models: linear regression, SVM, and LSTM neural network. The hypothesis is that PCA transformation will substantially lower the measured errors for linear regression and SVM although for LSTM it will result in lower performance as it will only decrease the capacity of the model as the model is powerful enough to create such uncorrelated features without the PCA transformation.

Expected Contribution Existing research agrees that financial data and especially cryptocurrency data are significantly affected by noise. The main goal is to extend the research on the topic of variable selection for algorithmic trading models as there are still a lot of unanswered questions. It will most likely become clearer which approach to dimensionality reduction is the most efficient concerning cryptocurrencies.

Also, not only the underlying pricing dynamics will be detected but the results can be used for investors that are trying to lower their risk of loss which is relatively high in crypto markets. The effect of having a more stable and precise model might significantly cut the transaction costs that are associated with more frequent exchanges as the predictions will become less volatile. That is a desirable property needed to maximize profit and increase credibility towards its customers.

Core bibliography

Kukacka, J., & Kristoufek, L. (2023). Fundamental and speculative components of the cryptocurrency pricing dynamics (Vol. 9). Financial Innovation.

Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis (Vol. 109). Expert Systems with Applications.

Chowdhury, U. N., Chakravarty, S. K., & Hossain, M. T. (2018). Short-Term Financial Time Series Forecasting Integrating Principal Component Analysis and Independent Component Analysis with Support Vector Regression (Vol. 6).

Bouri, E., Kristoufek, L., , & Shahzad, S. J. H. (2022). Microstructure noise and idiosyncratic volatility anomalies in cryptocurrencies. Springer Link. https://doi.org/10.1007/s10479-022-04568-9

Rea, A., & Rea, W. (2016). How many components should be retained from a multivariate time series PCA?. arXiv preprint arXiv:1610.03588.

Author

Supervisor

Chapter 1

Introduction

Since the introduction of the first cryptocurrency Bitcoin (BTC) associated with the unknown author Satoshi Nakamoto (2008) cryptocurrencies have become part of our everyday life. Their high volatility, futuristic name and alternative nature are of interest to the media and the general public. According to **coinmarketcap.com** the overall cryptocurrency market capitalization peaked at around 2.8 trillion $USD in the year 2022 which makes them a substantial part of the financial sphere. The initial idea of BTC was to establish an alternative to traditional fiat currencies. The BTC whitepaper pointed out the weakness of the current trust-based model that relies on a third-party instance responsible for verifying transactions. A different approach was suggested to validate transactions known as the proof-of-work which utilizes the computational power of miners in the network. The fact that the power is distributed across the network ensures that it becomes exponentially harder with an increasing number of blocks to generate blocks faster than the rest of the miners (Nakamoto 2008, pg. 6). The mining process is interconnected with the creation of new coins which is a crucial parameter in all monetary systems. This fact gives researchers such as Kukacka & Kristoufek (2023) the possibility to use various attributes of the network to study the pricing dynamics of cryptocurrencies. However, there are a couple of substantial drawbacks that make price modeling relatively challenging. Those are non-stationarity of the target prices, relatively short historical window, the limited power of proxies for speculative components and as pointed out by many researchers such as Bouri *et al.* (2022), Dimpfl & Peter (2021), Wątorek *et al.* (2023) an idiosyncratic noise in volatility. Addressing these issues might potentially lead to better-performing models, especially with longer forecasting periods. Likewise in other fields, the recent rise of machine learning has also affected the cryptocurrency area where various Machine Learing (ML) and Deep Learing (DL) models are often being used to model the price Khedr *et al.* (2021) or volatility Kristjanpoller & Minutolo (2018).

The main objective of this thesis is to try to tackle the problem of idiosyncratic noise in the high dimensional data used for price and returns modeling across three ML models: Ridge Linear Regression (LR), Support Vector Machines (SVM) and LSTM

Recurrent Neural Network (RNN). We will examine the effect of a method known as PCA which was according to Farebrother (2022) developed in 1933 by Harold Hotelling. However, others often refer to the fact that the idea was already introduced before by Karl Pearson in the article *On lines and planes of closest fit to systems of points in space* Pearson (1901). This technique aims to compress data from a higher dimensionality space into a lower space while retaining a maximum amount of variance. It utilizes linear transformation of the covariance matrix to do that. Nevertheless, despite the initial focus on dimensionality reduction different types of PCA are often being used as noise reduction techniques in signal or image processing. Interestingly many studies in recent years have incorporated PCA for time series data as a part of their preprocessing pipeline Chowdhury *et al.* (2018), Kristjanpoller & Minutolo (2018). The idea stems from the fact that removing the most idiosyncratic components might help with capturing clear dynamics that enter the price-making process. We perceive that there is currently a lack of literature that would examine the effects of noise reduction techniques on the performance ML based models for cryptocurrency modelling. We want to mitigate most of the identified challenges using the currently available academic knowledge and focus exclusively on the effect of noise in the data. Admittedly it is always intricate to establish a ceteris paribus relationship in such a scenario where many variables change, the randomness of the training process using Stochastic Gradient Descent (SGD) plays a crucial role and the size of the dataset is relatively limited. We want to contribute with an alternative approach, especially in the preprocessing pipeline that can be used in future studies to decrease the volatility of predictions. We do not aim to provide a universally applicable approach, as different techniques can produce varying outcomes on different datasets. This phenomenon partially corresponds to the *No Free Lunch Theorem* Wolpert *et al.* (1995) which has turned into a buzzword in the ML community over the years.

The remainder of the thesis is organized as follows: The following chapter introduces the fundamentals of cryptocurrencies and their unique characteristics. It also covers the usage of ML methods in this field and especially focuses on the literature about the usage of PCA in various areas. The data chapter explains in detail which data were used and elaborates on the basic resampling methods that we used. In methodology, we focus on each specific ML method and explain the core concepts that are crucial for understanding the training process. Similarly, we propose our complete forecasting framework. Chapter results and discussion evaluates the findings for each currency-model pair across different settings. We also include a limitations section which is especially crucial for our study where we acknowledge those problematic parts of our approach that might be improved in the future. The conclusion focuses on the overall impact and proposes paths that should be explored in the years to come. All the tables, source codes and visualizations can be found in the appendices.

Chapter 2

Literature Review

The following chapter will provide a detailed overview of the existing literature on cryptocurrencies and machine learning methods. We also cover the use of PCA in time-series data and the use of web search data in financial applications.

2.1 Cryptocurrencies

2.1.1 Bitcoin

In the year 2008, an unknown author with the pseudonym Satoshi Nakamoto introduced the idea of a purely peer-to-peer electronic cash system. Interestingly the author mentions small casual transactions as something that the current model relying on third-party financial institutions fails to deliver because of unavoidable transaction costs Nakamoto (2008). In contrast, from today's perspective, Bitcoin is a relatively slow medium for micro-transactions because technically the receiver has to always wait for a certain amount of blocks to be mined such that it becomes statistically unlikely that double-spending has been committed by the payer Conti *et al.* (2018). This phenomenon can be demonstrated on the data from **coinmetrics.io** which show that the mean size of a BTC transaction ranges in thousands of USD$. Another important aspect is that the miners prioritize transactions with higher fees in the block which introduces considerable costs to each payment. Möser & Böhme (2015) have shown the relationship between the transaction fee and the transaction latency meaning the time it takes for the transaction to be almost surely valid. Even though there has been some divergence from the original idea of small transactions all of the security measures regarding the double spending problem in the original whitepaper have turned out to be relatively well-defined in a medium time horizon.

Despite the fact, that Bitcoin is generally regarded as the first cryptocurrency it relies on many older ideas and technologies that are mostly referenced in the original whitepaper. First and foremost stands the conference paper *How to Time-Stamp a Dig-*

ital Document Haber & Stornetta (1991) which focuses on the problem of third parties responsible for verification of digital documents. It makes use of an already established family of functions known as hashes that surpass privacy concerns and generally surpass the need for a third party to be involved in the verification process when combined with the correct consensus algorithm. They define a hash function as follows:

Definition 2.1 (Hash). This is a family of functions $h : \{0,1\} \to \{0,1\}^l$ compressing bit-strings of arbitrary length to bit-strings of a fixed length l, with the following properties:

1. The functions h are easy to compute, and it is easy to pick a member of the family at random.

2. It is computationally infeasible, given one of these functions h, to find a pair of distinct strings x, x' satisfying $h(x) = h(x')$. (Such a pair is called a collision for h)(Haber & Stornetta 1991, see Chapter 4.1)

And suggest hashing documents together with the time of their creation. However, the time-stamping might fail if the users can tweak the time of their machines. Interestingly, the authors have already mentioned that and introduced the idea of chaining the data together with their metadata sequentially in a long chain so that the user can trust that something was not overwritten (Haber & Stornetta 1991, see Chapter 5). This is possible due to the properties of the hash functions. Meaning that we can concatenate arbitrarily long inputs and always produce a fixed-size output. Another significant influence came from b-money which was an idea for an anonymous digital cash system presented in Dai (1998). B-money proposed the concept of Proof of Work (PoW) which is a validation protocol that many cryptocurrencies still use. The idea was to solve computationally challenging puzzles where it can be determined how much effort was used to do so (see Dai 1998, pg. 1). However, certain worries were being raised about how to regulate a system if the computational power of computers is increasing every year (see Dai 1998, pg. 3). This has been addressed by Bitcoin with the regulation of difficulty based on the average time it takes to solve the puzzle rather than the difficulty itself (see Nakamoto 2008, pg. 3).

Since many characteristics of the network are often being used by researchers such as: (Kukacka & Kristoufek (2023), Kristoufek (2023), Kubal & Kristoufek (2022) or Jay *et al.* (2020)) in their research, it is critical to understand the underlying mechanics that form them. Bitcoin takes a completely adverse approach to the general financial system. Whereas traditionally, banks and other institutions try to keep every transaction encrypted Bitcoin makes all the transactions publicly visible and available but hashing the addresses of the sender and receiver. The process of sending Bitcoins to someone else essentially means adding a digital signature to the previous transaction from which you received that money. However, as (see Chapter 2 Nakamoto 2008, pg. 2) suggests this

only mitigates the privacy concerns but the double spending risk needs to be dealt with a smarter design. This is solved by the introduction of the PoW algorithm. The idea is that transactions are collected into blocks by the miners who try to solve a computationally difficult task that can only be solved by a brute-force search. It is simply a race to find a hash with a certain amount of leading zeros which is adjusted based on the mining power of the network. The miners are incentivized by BTC price for winning the race and also by the transaction fees that can be added to each transaction as a reward for being prioritized. The leading concept is that the blocks are connected sequentially in a chain through the hash. If we assume that most of the nodes/miners are honest their profit-maximizing behavior should always be working on the longest chain and thus transactions that have already been spent do not get included in the chain. After the puzzle is solved by a node it can be validated by all other nodes in a linear time and they move on to the next block. Despite that, there remains the risk of an attacker forking a malicious block and sending his money back or elsewhere. Nakamoto (2008) claims that the probability of an attacker catching up (or reaching breakeven from the memoryless property of Poisson distribution) drops exponentially with each block if the mining power (probability of solving the puzzle) of the attacker is lower than the power of honest nodes. This mechanism is the root of the Bitcoin security. However, it also implies that there is an implicit tradeoff between security and the desired liquidity of cash. Another important property is that there will ever exist only a limited amount of 21 million of BTCs which makes it inherently a deflationary currency at least after all of the BTCs are mined. Technically there will be less as some wallets do get lost together with their contents. Limiting supply might be an intentional design choice to contrast the traditional model where banks issue money and cause inflation.

As cryptocurrencies are a relatively new phenomenon they are currently a frontier topic of academic research in many different aspects. They are being studied on multiple levels such as law, technology, cryptography, security, economics or machine learning. Generally, the area of economics and machine learning will be of interest as we want to uncover whether there exist some determinants of the bitcoin price or at least features that can be used to estimate the pricing dynamics. We assume that there are theoretically three simplified possibilities of the pricing model of BTC and other cryptocurrencies. Firstly, it might be a purely efficient market and the price technically follows a random walk process with a potential drift. The second model is that the price is entirely driven by speculative components and lastly, it might be a combination of speculative and fundamental components.

2.1.2 Ethereum

At the time of writing according to the data from **coinmarketcap.com**,
Ethereum (ETH) is the second cryptocurrency based on the market capitalization standing at 318 billion USD. Although, there are a lot of similarities with BTC the idea behind ETH is much more profound and builds an entire technological infrastructure on top of blockchain which provides easily scriptable smart contracts. Tikhomirov (2018) defines smart contracts as follows:

> *Turing-complete programs that are executed in a decentralized network and usually manipulate digital units of value.*

We might want to reformulate this definition for the purposes of this thesis to cover a broader meaning.

Definition 2.2 (Smart contract). A program that is running on the blockchain infrastructure as an endpoint with a specific address that other entities on the platform can interact with in order to execute a transaction for a cost proportional to the number of computational steps. The contract usually has a predefined set of rules which is applied to the input data and automatically executes on hold whenever called. This allows developers to build applications on top of blockchain infrastructure that is run by the distributed network and grants them the unique benefits of the blockchain model.

According to the original whitepaper, the main purpose of ETH was not to create another cryptocurrency but build a simple-to-use scripting language that would allow developers to create custom applications running on the blockchain that share the benefits of the distributed nature of the system but reduce the need for hardware as the transactions are handled by the network and also software as the transactions can be defined in a few lines of code. Notably it describes cryptocurrencies as a state transition system:

Definition 2.3 (Digital currency ledger). From a technical standpoint, the ledger of a cryptocurrency such as BTC can be thought of as a state transition system, where there is a *state* consisting of the ownership status of all existing coins and a *state transition function* that takes a *state* and a *transaction* and outputs a new state *state-new* which is the result.

In this view, we can understand building smart contracts, see Definition 2.2, as creating use case specific *state transition functions* ((Buterin *et al.* 2013, see Chapter Bitcoin As A State Transition System),(Tikhomirov 2018, see Chapter 2)).

We can observe the fundamental difference in philosophy between the BTC and ETH from their respective whitepapers. ETH is much more focused on its role as an application platform whereas BTC was mainly intended to work as a currency. The ETH whitepaper

even presents many ideas for future applications that might benefit from this framework. This fact is crucial in understanding the underlying price-making mechanics as investments into ETH might be affected by its perceived potential as a technological product rather than a typical currency that might be valued mostly as a medium of exchange.

Even though there are a lot of similarities in the blockchain architecture, there have been many functional and implementation differences. The most notable is the ETH scripting language which is a Turing-complete counterpart to the BTC scripting language that did not support infinite looping (Buterin *et al.* 2013, see Chapter Scripting). The second fundamental implementation variation is that each Ethereum block saves the entire *state* of the whole network and thus does not need to store the whole history of the blockchain. On the contrary, BTC is doing exactly that, as the blocks are state-unaware and only validate blocks based on the transactions and the wallet software usually calculates the balances (Tikhomirov 2018, see Chapter 2). As Buterin *et al.* (2013) pointed out there is a workaround that makes this implementation roughly equally efficient which utilizes the propagation properties of the Merkle tree with the fact that only a small part of the state changes with each block allowing for efficient state change using pointers to the specific branches and leaves of this data structure, see Figure 2.1. The data are stored at the bottom of this infrastructure and there is an efficient algorithm that allows the ETH software to reference the affected addresses and paths leading to them.

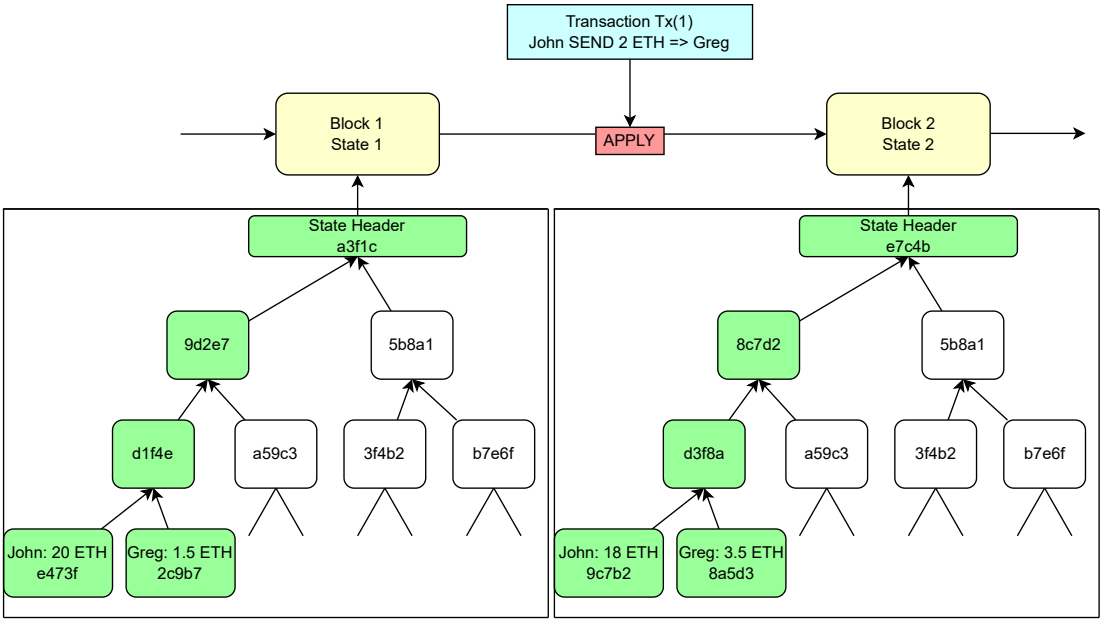


Figure 2.1: Merkel tree with pointers allows efficient state change.

Source: Author

Lastly, there is a difference in the supply of new coins. Contrasting the model of BTC where the supply is limited ETH introduces a model of an infinite linear supply of coins to provide incentives for future users to join the network as they might still obtain new coins and thus limit the wealth concentration common in BTC (Buterin *et al.* (2013), Tikhomirov (2018)). Note that ETH already switched to the proof-of-stake model in September 2022 but our dataset does not include this period and thus this fact is not especially relevant to this thesis.

On the outside ETH acts similarly to BTC. It is a ledger that stores the coin balances of accounts where each has a designated unique address. There are two types of accounts: externally owned accounts which are essentially the typical users and contract accounts which are the abstraction on top of which smart contracts can be built with contract code that executes when the account receives a message from another account (externally owned or a different contract account). Because of the presence of infinite loops in the scripting language ETH employs a strategy that prevents users from essentially exploiting the Denial-of-service attack. Nonetheless stands the Halting problem. That can be simplified to the fact that for a Turing-complete model, there is no way of saying ex-ante whether the program will halt or run indefinitely. As Lucas (2021) suggests the Halting problem is usually attributed to Alan Turing's paper (Turing *et al.* 1936, On computable numbers, with an application to the Entscheidungsproblem), however, the problem was reformulated in various forms by others. This implies that this undecidability also holds for any ETH contract code. This is fixed by introducing a gas currency that acts as a cost of computation and the maximum has to be predefined in each message so that the recipient knows what is at stake Buterin *et al.* (2013). We can think of this as a type of timeout based on a currency. If gas runs out all of the state changes are reverted. This also explains the usage of the term transactions in this setting. Transaction typically refers to a set of instructions in SQL or other databases that are bundled together and executed in an all-or-nothing fashion Kleppmann (2017). The last important fact, as the whitepaper describes, is that the contract code is run by all of the miners verifying the block which essentially means applying all of the transactions and reverting in case of an error.

2.1.3 Litecoin

In comparison to ETH, the goal of Litecoin (LTC) was pronounced from the beginning as building a better version of BTC that shines where BTC has failed. We might say that LTC is a tweaked version of BTC with different parameters or a hard fork of the BTC protocol. To our best understanding, the only document that is wildly considered the original whitepaper is a transcript of a forum post by the founder Charlie Lee where he suggests reading the BTC whitepaper. Essentially, two main differences address the problems BTC

embodies. The first one is faster confirmation time that allows LTC to be used truly in a fashion that was intended for BTC as digital cash with high liquidity sacrificing a bit of security. Achieving that mostly through four times faster block generation. And second one is a different proof-of-work algorithm. As Padmavathi & Suresh (2018) suggests the intention was most likely since BTC mining was dominated by GPU and ASIC miners which led to a concentration of mining power in pools and thus more centralized distribution of BTC. Despite the initial promises Litecoin has most likely not fulfilled its envisaged role and does not currently belong even to the ten most popular cryptocurrencies by market capitalization.

2.2 Machine Learning Methods for Cryptocurrencies

Thanks to transformer-based architectures that are the backbone of most chatbots. ML and artificial intelligence have gotten a lot of public spotlight during the last two years. However, ML methods have been especially prevalent in research in the last decade. Particularly in data-driven fields of academia, there have been various use cases where ML shines and outperforms traditional statistical models. Forecasting has always been an area of interest in many different fields as the idea of predicting the future based on historical data provides intrinsic value in itself. The field of cryptocurrencies is no exception as the significant volatility is a thought-provoking problem to tackle and an opportunity to win against the rest of the market.

In order to use ML or any other data-driven method we implicitly have to assume that there are some underlying dynamics of which we can make sense. This argument is hard to make without the proper data and might have to be studied separately for different time horizons. Thankfully, Kukacka & Kristoufek (2023) argue that some cryptocurrencies especially BTC are driven by the interaction of speculative and fundamental components and thus provide an incentive for us to untangle and study these relationships. Disturbingly, even though there are many other papers focusing directly on the practical implementation of forecasting frameworks for cryptocurrency prices a significant amount of them do not compare their model to a random walk or other simple statistical model. We suggest that their approaches should not be condemned but we strongly emphasize that their results should be interpreted cautiously.

We believe that there is currently an upsetting trend in the studies focusing on modelling the price or returns of cryptocurrencies using ML methods. There seem to be a lot of inconsistencies in terms of splitting the data into training and testing sets, making the results robust using some forms of cross-validation, comparably presenting the results and contrasting the models to a meaningful baseline model. This reality arguably contributes to the fact that the results of most of the studies are relatively underwhelming and have stagnated in the last few years. The other side of the coin, arguably worse, are studies

that present overly optimistic results. A similar critique of the generalizability of results was presented by Akyildirim *et al.* (2020). We propose an idea for future research that would imaginably help the field advance further and stimulate innovation. The suggestion lies in the creation of a standardized dataset with set splits between train, validation and test data that would allow for the comparison of different approaches and would encourage researchers to make their models generalizable. Taking inspiration from the field of image recognition where the Image-net is a state-of-the-art dataset to compare models for image recognition. This would allow for a competitive environment boosting innovation and development. We acknowledge that there is a fundamental difference between a dataset that consists of an unordered series of images and an especially challenging non-stationary time-series. Some compromises would undeniably have to be introduced in terms of set prediction horizons, different data granularity and artificially set splitting points. However, there is always a place for variation and this dataset could have multiple versions. Even though, this solution is sub-optimal we firmly believe the current scattered state of knowledge in this field makes it extremely difficult for further development to blossom.

Despite that, we would like to provide a brief overview of the methods that are often incorporated in price or returns modelling pipelines. As we already mentioned in order for ML to work as a forecasting algorithm there has to exist a possibility of drawing information about the future from the past. That is a non-trivial assumption because it contradicts The Efficient market hypothesis that was formulated for capital markets by Fama (1970). It is thus no oddity that Ren *et al.* (2022) have found out that across 395 scientific articles about the use of ML in cryptocurrencies the most cited article was Urquhart (2016) that studied the efficiency of BTC and concluded that the BTC market is inefficient but may become efficient in the future. And that the keyword inefficiency was the one with the highest burst of emergence among the articles.

There are generally two types of the problem formulation. Either the price is being modelled as a regression problem or the movement of the market is predicted as a binary classification. Modelling market movement is easier and it is also much more comparable across studies. If we can assume that there are approximately the same amounts of ups and downs we can work against the 50% baseline Akyildirim *et al.* (2020). Khedr *et al.* (2021) pointed out that cryptocurrencies lack seasonal trends which makes them challenging to predict for traditional statistical models. They also distinguish four types of factors influencing cryptocurrency price. Namely: demand and supply, crypto market, macro-economic and political. Our work will cover the first three types of factors as the political factors are hard to quantify. Later Ren *et al.* (2022) suggested that future researchers should study measurement tools for political factors. If you want to find out more about research in this field from 2010-2020 please view Khedr *et al.* (2021) which provides a comprehensive review. However, as we suggested earlier the aspect of

comparison between studies is rather limited.

Alessandretti *et al.* (2018) compared multiple models from the perspective of portfolio management comparing them using return on investment and suggested that all of the ML models outperformed the moving average baseline. They also introduce the idea of predicting cryptocurrencies prices in BTC rather than USD to filter out the effect of overall cryptocurrency market growth and ease the effects of spurious correlation. In their study LSTM outperformed the gradient-boosting decision trees with longer time horizons and vice versa. Lastly, they pinpointed the fact that expressing prices in terms of BTC helped the predictions and concluded that forecasting only the trend of individual currencies might be easier than also predicting the overall market trend. Chowdhury *et al.* (2020) used ensemble methods to predict the prices of multiple cryptocurrencies. Utilizing the fact that combining multiple weaker models with uncorrelated errors with voting can increase the overall performance (Bishop & Nasrabadi 2006, see Chapter 14.2). Unfortunately however impressive their results might seem they have been evaluated only on a month of predictions which is in our view insufficient.

2.3 Principal Component Analysis

2.3.1 PCA in Time Series

PCA is a widely adopted signal processing technique that is typically used to lower dimensionality of input features and to mitigate noise in the signal by transforming the data into a different feature space while ensuring that the new principal components remain uncorrelated. Traditional usecases include dimensionality reduction for visualization purpouses (word embedding vectors), noise reduction in ECG, dimensionality reduction for performance reasons of different ML methods and many other signal processing pipelines.

Thanks to its valuable properties it has been recognized also as a tool to simplify and bring insights into multivariate time-series data concretely in the financial sector. However, using PCA in time-series usually comes with many identified challenges. Especially the need for the data to be normalized before being processed by the PCA layer might turn to be almost impossible to fulfill with non-stationary time-series. Despite these shortcoming many researchers have used PCA in their predictive work with a substantial success.

Most notably Nadkarni & Ferreira Neves (2018) designed a technical analysis based trading strategy for stocks and indexes that used PCA to reduce the dimensionality of input features from 6 to 31 that enabled the genetic NeuroEvolution algorithm to design much simpler Artificial Neural Network (ANN), reduce noise in the data and decrease the risk of overfitting the training datasetset. Their approach significantly outperformed the Buy and Hold strategy and they conclude that PCA was vital to achieve such results.

Their work is especially important because of generalizability for future work as their approach used the broad population of ANNs even making each neuron dynamically choose different activation function and thus introduces no constrains on the generality of the results. Lastly they concluded that PCA led to lower risk, less days spent with capital in the market and higher daily profits.

Similarly, Chowdhury *et al.* (2018) used PCA and Independent Component Analysis as a preprocessing layer for stock prediction and found out that the proposed framework outperformed the approach where only the final Support Vector Regression (SVR) was used for prediction. Many other works use PCA only as a dimensionality reduction technique such as Toledo & Souza (2022).

An alternative novel approach used by some researchers (Shah *et al.* (2021), Mohsin *et al.* (2021), Smales (2020)) focuses on creating a cryptomarket wide index using PCA to capture the overall market trends. Smales (2020) found that the first principal component explains a large portion of the variation and is highly correlated with the BTC returns and thus support the notion of BTC returns being an important driver for other cryptocurrencies. Shah *et al.* (2021) critique the use of rule based indexes that do not rely on any fundamental mathematical reasoning and thus create a dynamic index using PCA that captures the overall market trends. However, they discourage the use of the index as an ETF highlighting the fact that the maximal variance optimization may be associated with higher volatility.

2.4 Web Search Data in Financial Applications

The phenomenon of Google Trends has drawn attention of many researchers. First and foremost stands the idea of Google Flu Trends. A project launched by Google which aimed at predicting flu epidemics using the symptom related searches by the users Ginsberg *et al.* (2009). However, as Lazer *et al.* (2014) pointed out there have been many underlying issues both in the data construction and the dynamic nature of the search engine changing with years of development. We should highligh the importance of the correlation-causation fallacy in the use of Google Trends or other search based services. It is even worsened by the Google Trends weekly data granularity which makes it especially challenging to assess any kind of causality and its direction. We have to take into account the possibility that there is a bidirectional influence between Google Trends and BTC prices. Despite these challenges, research suggest that there is an underlying value in using web search data for modelling financial time series.

Hu *et al.* (2018) used ANN to predict the direction of the stock market indexes and compare two setups with and without Google Trends and conclude that inclusion of Google Trends improves the performance of their models. Similarly, Huang *et al.* (2019) studied the effect of investor attention measured by Google Trends as a potential market

signal. Interestingly, they do not find evidence that market movements cause changes in search volumes. However, they conclude that search terms are useful in predicting stock market movements and as a proxy variable for investor attention. Most notably they emphasize that the effect is conditional on the sentiment of the search. Suggesting that we need to distinguish between positive and negative searches. This idea was adressed before in Kristoufek (2013) that found an asymmetric effect of positive and negative searches which were discriminated based on whether the price over or underperformed compared to a moving average of 4 days for Google Trends and 7 days for Wikipedia visits. Furthermore, Arratia & López-Barrantes (2021) studied both linear and nonlinear effect between Google Trends and BTC returns and also conclude that Google Trends can be used to improve accuracy when forecasting BTC prices.

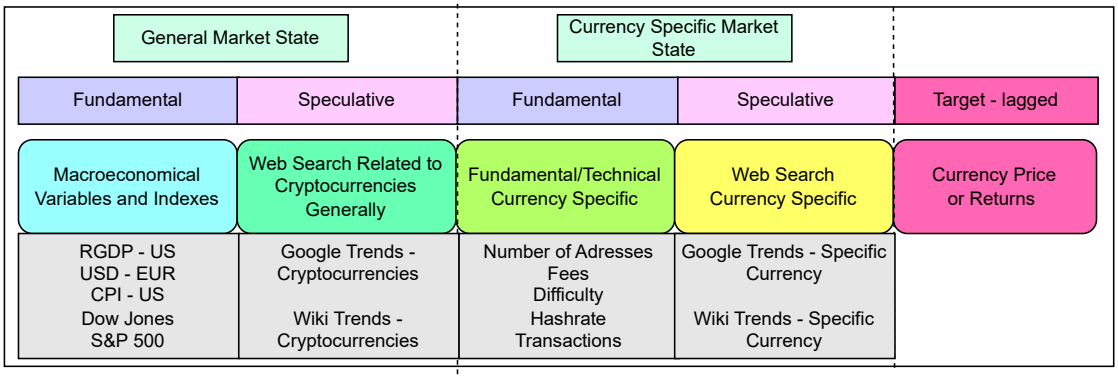
Chapter 3

Data

In this section we will describe the data used in this thesis. Argument for the choice of the data and explain the preprocessing steps that were applied to the data.

Compared to traditional financial data cryptocurrency data tend to be easier to obtain. However, many data providers have already identified the bussiness potential of selling advanced data and have started to monetize them. Since some websites monetize only their APIs we could obtain data from multiple sources and merge them to get to our expected number of features. We have desired to use as many relevant variables as possible and we build on an existing research in the field. Our dataset can be split into 2 main categories and 4 subcategories. We utilize the data that describe the overall market trends in the traditional sense. These data were mostly obtained from **Federal Reserve Bank of St. Louis** and from the **Yahoo Finance** API. To balance fundamental indicators with speculative side we used **Google Trends** and **Wikipedia Page Views** that act as a proxy for market attention about cryptocurrencies in general and should help us to model the market hype periods. These explanatory variables are then used to forecast the price or returns of the specific cryptocurrency of interest lagged back in time.

Figure 3.1: Dataset Variables Overview.



Source: Author

The data were collected between December 2023 to February 2024 varying based on the different sources but they are further shortened to utilize the most overlapping region between data sources for each unique cryptocurrency. This results in a time series from 17.9.2014 to 1.11.2022 for Bitcoin, 4.2.2016 to 27.7.2022 for Ethereum ensuring that the end of the series is before the change to proof-of-stake algorithm and series from 17.9.2014 to 1.11.2022 for Litecoin. The shifted versions of the datasets are of variable length based on the forecasting horizon.

3.1 Cryptocurrency Specific Technical Data

Despite the fact, that we have framed this data as fundamental/technical we should acknowledge that fundamental in our case stands far from its traditional meaning. They are fundamental in a way that they are the typical data researchers and practitioners employ to model cryptocurrencies and that they objectively describe the system. However, as the fundamental factor is very limited in this case it only makes sense for the typical variables such as capitalization, volatility, hashrate and others. But we also used a lot of technical variables that are derived from these fundamental variables or the price itself. These help to identify trends in certain variables from pure mathematical transformation of the original series. All of these data were collected from **coinmetrics.io** and further processed by our pipeline.

We incorporate many variables describing specific technical characteristics of the network. Starting with number of active addresses which acts as a measure of user activity on a particular day that is a cruical parameter potentially capturing the strength of bull or bear market behaviour. However, it does not reflect the direction itself. The difficulty of the network is adjusted dynamically to counteract the changes in the mining power and thus act as a proxy for current mining power of the network or the current efforts of the miners. The size of the block is another characteristic of the network describing the size of the block (in bytes) and has been steadily increasing overtime with significant fluctuation that depend on many other changes in the network. Hashrate is a measure of how fast do the miners solve the hash for one block. In the long term the mean hashrate should be proportional to the difficulty at least that is how the BTC protocol was designed however there are some fluctuation in the short term as the difficulty is adjusted every 2016 blocks to match the average hashrate over that period.

Other variables focus on the economics of the currency and model the market behaviour not the network itself. Beginning with traditional market signals such as capitalization indicating the overall price of all coins and volatility of returns as a standart deviation of log-returns. Furthermore, we employ many other variables about the price of fees provided by the users, revenues of the miners, distribution of wealth in the network and the number of transactions for that interval.

For the full description of the variables for BTC please refer to the Appendix C.

The datasets for Ethereum and Litecoin look similar with the exception of a few variables missing, please see (3.2):

Figure 3.2: Litecoin missing variables

- *Flow, in, to exchanges, USD*
- *Flow, out, from exchanges, USD*
- *Revenue, per hash unit, USD*

3.2 Macroeconomical Data

As the macroeconomical condition is a cruical factor for investor behaviour we decided to include relevant variables that might deliver valuable insights. We utilize five macroeconomical indicators that affect invesment choices. Real Gross Domestic product of the United States represents the overall growth trend of the largest economy in the world with closely related real gross domestic product per capita telling more about individual resouces which gives a more complex picture of the state of the US economy despite the fact that americans are not the main cryptocurrency investors by nation. Furthermore we incorporate Consumer Price Index in the United States that acts as an inflationary measure to capture the spurious correlation between prices of USD and BTC-USD exchange rate. M2 base acts as a measure of dollar liquidity in the circulation. Lastly, the USD-EUR exchange rate can be thought of as a market state information or its returns as an opportunity costs for potential investors. To further adress the problem of spurious correlation and add the information about growth of other markets we include various closing prices and other measures of the stock market and other investment opportunities.

For the full description of the variables please refer to Table 3.1.

3.3 Web Search Data

As mentioned earlier following many other researchers we aimed to obtain a proxy for the market attention. We specifically used **Wikipedia Page Views** to get the page views for Wikipedia pages: Bitcoin, Ethereum, Litecoin and Cryptocurrency and use them respectively for each coin dataset combining the overall Cryptocurrency views with the specific currency. Similarly we hoped to obtain similar data from **Google Trends** but they turned out to be quite cumbersome to use. As West (2020) mentioned there are three main obstacles when using Google Trends. Firstly, the scale is always normalized

Table 3.1: Macro Variables Descriptions

Variable	Description
Close_DJI	Dow Jones Industrial Average
Close_GSPC	S&P 500
Close_GC=F	Gold Futures
Close_VIX	CBOE Volatility Index
Close_IXIC	NASDAQ Composite
Close_SMH	VanEck Semiconductor ETF
Close_VGT	Vanguard Information Technology Index Fund
Close_XSD	SPDR S&P Semiconductor ETF
Close_IYW	iShares U.S. Technology ETF
Close_FTEC	Fidelity MSCI Information Technology Index ETF
Close_IGV	iShares Expanded Tech-Software Sector ETF
Close_QQQ	Invesco QQQ Trust
RGDP_US	Real Gross Domestic Product of the United States
RGDP_PC_US	Real Gross Domestic Product per capita of the United States
CPI_US	Consumer Price Index: All Items: Total for United States
M2_US	M2 Base US
USD_EUR_rate	U.S. Dollars to Euro Spot Exchange Rate

into the range 0-100 based on the selected region and time. Secondly, this implicitly means that the results are rounded to integers and thus loosing a lot of precision. Lastly, there is a limit of 5 queries that you can use at a time. This not only means that you cannot compare more search terms but it also means that when there are over five variations how the term might be searched for one cannot do that effectively. Another problem, that we faced is that the data can be obtained only in weekly granularity for longer periods and thus needs to be interpolated to daily which most likely sacrifices a lot of interesting dynamics on the daily level which is significant regarding the volatility of cryptocurrencies. These reasons, except the weekly sampling frequency, were solved in the **g-tab** Python library, created by West (2020), which uses a two step query sampling process that estimates the searches on a universally common scale with floating precision and allows us to use as many word formulation as needed. We would then sum the popularity for each word formulation that is related to the same thing. We acknowledge that there is a risk that we ommited some word formulations but hopefully we covered all the significant ones.

Following is the list of the variables from this section and their word forms for Google Trends: https://cs.wikipedia.org/wiki/Bitcoin

Table 3.2: Search Variables Descriptions

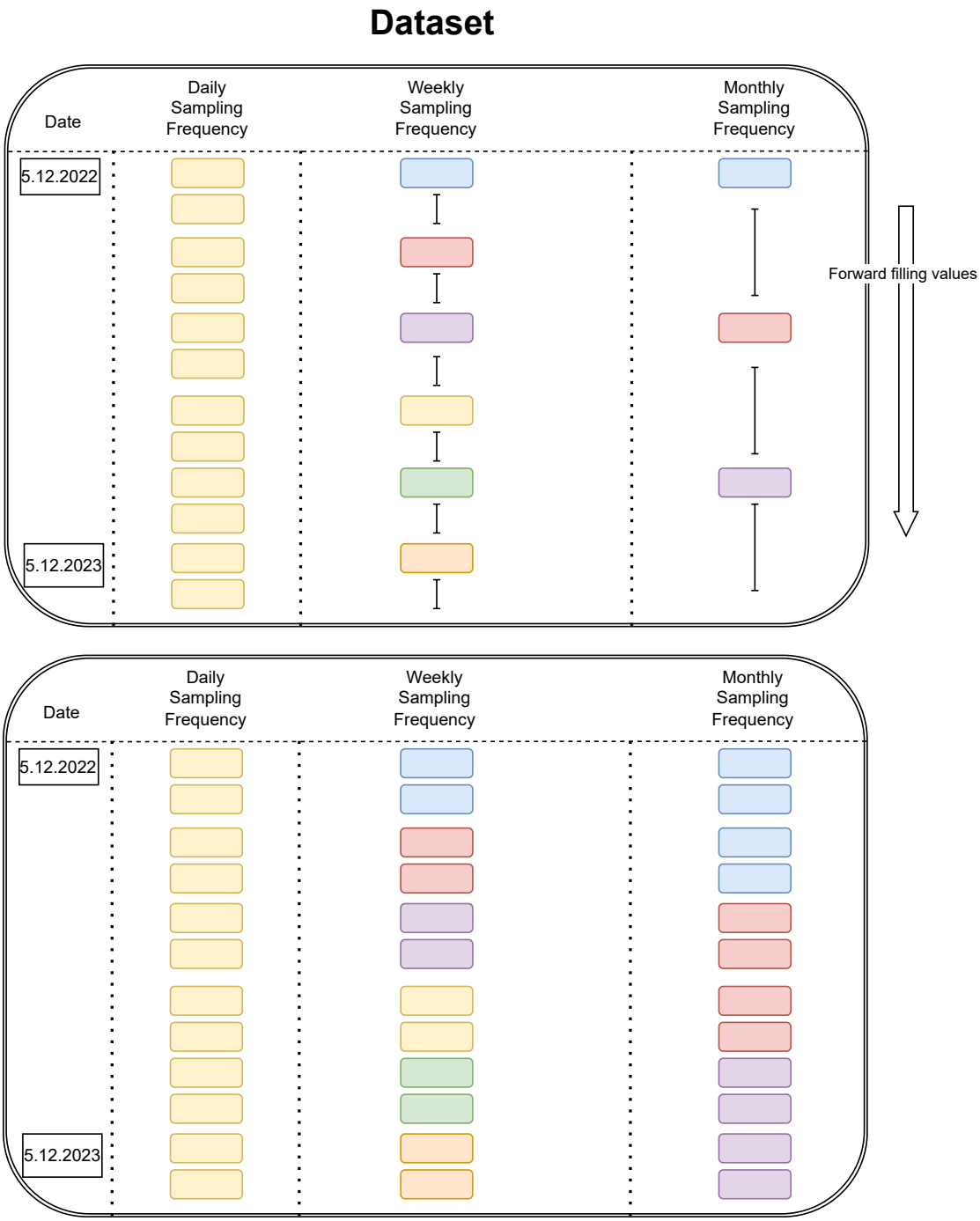
Variable	Description
`Wiki_btc_search`	Wiki pageviews for **Bitcoin**
`Wiki_eth_search`	Wiki pageviews for **Ethereum**
`Wiki_ltc_search`	Wiki pageviews for **Litecoin**
`Wiki_crypto_search`	Wiki pageviews for **Cryptocurrency**
`Google_btc_search`	Google Trends summed across terms: *Bitcoin, bitcoin, BTC*
`Google_eth_search`	Google Trends summed across terms: *Ethereum, ethereum, ether, ETH*
`Google_ltc_search`	Google Trends summed across terms: *Litecoin, litecoin, LTC*
`Google_crypto_search`	Google Trends summed across terms: *Cryptocurrency, cryptocurrency, Cryptocurrencies, cryptocurrencies, crypto, Crypto*

3.4 Preprocessing

Forecasting from historical data comes with many identified challenges. These were especially pronounced in our case as we incorporate data from many different sources with different sampling frequencies. Despite the fact that our focus is on forecasting daily price and returns we use data that come in weekly and monthly frequencies. We suggest two general concerns with such data. As we are using historical explanatory variables to predict a response variable in the future we need to ensure that our explanatory variables are already published at the time of forecasting and not use data from the future. We implement forward filling after the data is resampled to daily frequencies as a remedy. Especially for the macroeconomical data we implicitly match those indicators to future dates that they do not trully corespond to. Concretely, the Consumer Price Index for January is published at the beginning of February but as we forward fill this variable the Consumer Price Index from January will actually be in February. This ensures that when we are forecasting with a 10 day horizon at the beginning of February we only use data that was present at that time. The second concern is the fact that there is a lot of lost signal during those interpolated periods and the daily dynamics are not incorporated in the model. Unfortunately, this is the nature of economic research as especially the macroeconomical indicators come typically in such sampling frequencies. As already suggested these preprocessing steps were applied to the macroeconomical indicators, Google Trends that come in weekly frequency and for ETFs and indexes that are not traded on the weekends, see Figure 3.3.

As some of the variables are missing at the beginning of the time frame there is no clear solution how to impute them without leaking the future distribution. Even though we could say that this rule might be violated on the training set but we opted for a different

Figure 3.3: Resampling data to daily frequency is implemented using forward filling to avoid future distribution leakage.



Source: Author

approach. We start by visually inspecting the structure of the missing values and set the start of the series as such that "almost all" variables are already being collected. The

aim is to keep as much data as possible but avoid having a lot of data points at the beginning with different distribution changed by imputation. After that we fill the data that is missing only at the start with zeros in order to avoid imputing future data. There is also a second reason as generaly these variables are increasing and thus imputing zeros makes even mathematical sense. Especially for data like Googe Trends or Wikipedia Pageviews this is a reasonable imputation. Finally we cut the ETH series at the switch to proof-of-stake as this completely changes the modeling perspective and we merge all the data from different sources on the date column.

The most cruical step is the transformation of this dataset into a supervised learning problem. We will proceed with the example of BTC price, however the same will hold for other coins and returns forecasting. We employ forecasting with 1, 5 and 10 day horizon h. We denote the number of observations $\mathbf{x_i}$ as n and the number of features j as k.

$$h \in \{1, 5, 10\} \tag{3.1}$$

$$j \in \{1, ..., k\} \tag{3.2}$$

$$i \in \{1, ..., n\} \tag{3.3}$$

We start by denoting the original dataset $\mathbf{X} \in \mathbb{R}^{n \times k}$. Now assuming $\mathbf{X}_{:,k} = [\mathbf{x}_{1k}, ..., \mathbf{x}_{nk}]$ is the BTC price feature vector we shift it back in time by h and drop the data without corresponding counterpart. Concretely, we drop all $\mathbf{X}_{n-h:n,1:k-1}$ and $\mathbf{X}_{1:h,k}$. With this technique we always sacrifice $2h$ observations from the original dataset. If we further split the dataset into the explanatory and target features and reset the indexation such that we denote $m = n - 2h$ we end up with the following matrices where the target matrix Y is essentially only a vector where $y^\top = \mathbf{X}_{:,k}$, see Figure 3.4.

$$\mathbf{X} \in \mathbb{R}^{m \times k-1} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1k-1} \\ x_{21} & x_{22} & \dots & x_{2k-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \dots & x_{mk-1} \end{pmatrix} \tag{3.4}$$

$$\mathbf{Y} \in \mathbb{R}^{m \times 1} = \mathbf{y}^\top = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \tag{3.5}$$

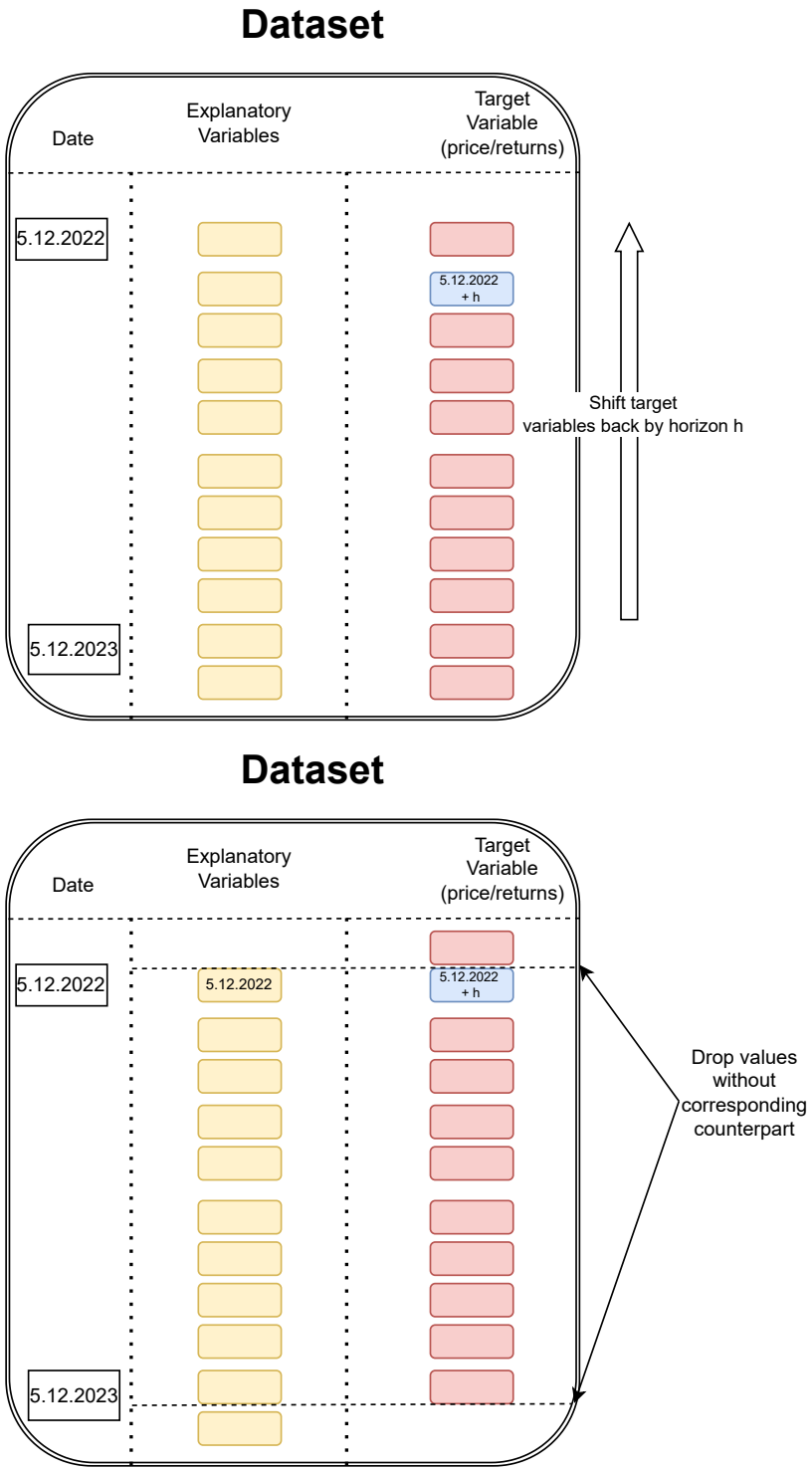
We can now observe that modeling y_i as a function of explanatory variables x_i is a forecasting with horizon h into the future. Where we essentially try to estimate the function $f_{foreacast}$.

$$y_i = f_{forecast}(\mathbf{x}_i) \tag{3.6}$$

The final preprocessing step is specific to the LSTM network which is a type of architecture that requires the input to be of the shape (num timesteps, num features). This means that it can use for example 10 day history for each variable and forecast the target variable. This is a really strong feature and makes it a viable option for our usecase. In order to use it, the input dataset has to be reshaped in such a way that for observation $\mathbf{x_i}$ concatenates values from $\mathbf{x}_i$ up to $\mathbf{x}_{i-lag}$, see Figure 3.5

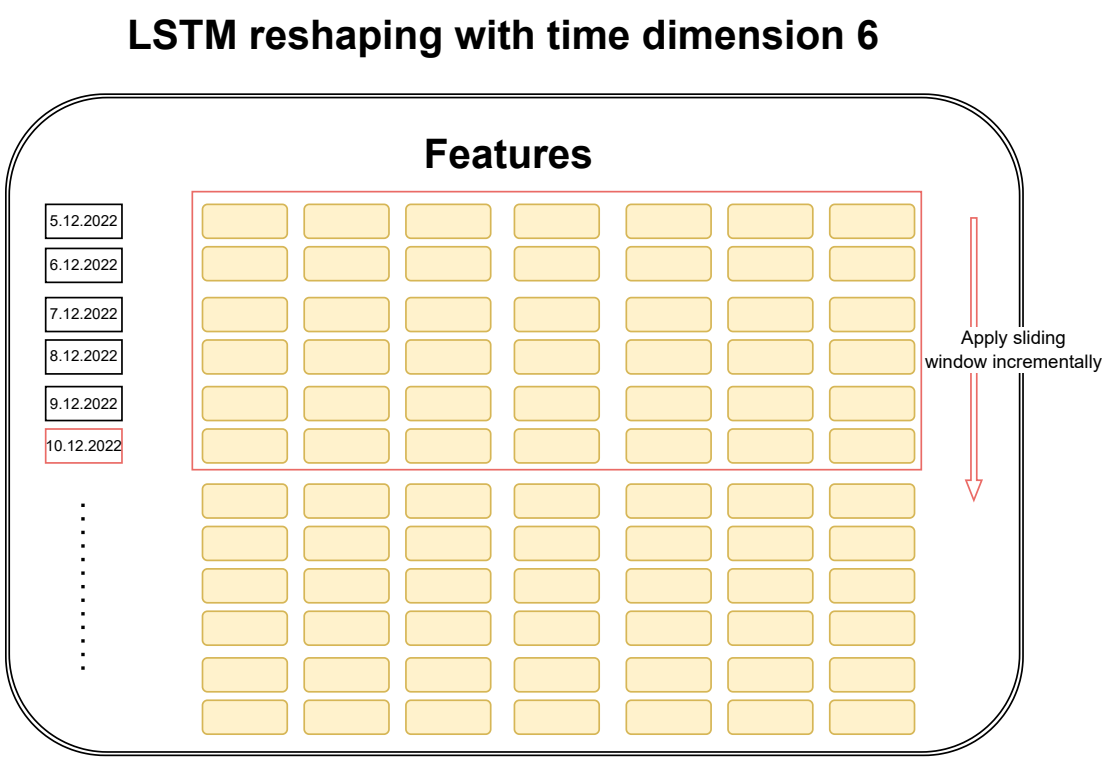
$$\mathbf{X}_{i,:} = [[x_{i1}, ..., x_{ik}], [x_{i-11}, ..., x_{i-1k}], [x_{i-lag1}, ..., x_{i-lagk}], ...] \tag{3.7}$$

Figure 3.4: Transforming data into supervised learning problem by shifting target variable by the forecast horizon back in time.



Source: Author

Figure 3.5: Reshaping for LSTM network requires applying sliding window over the whole dataset to introduce temporal information to the model.



Source: Author

Chapter 4

Methodology

This section starts by providing general overview of the ML field and elaborates on the models used in the experiments. We also comment on the role of stationarity in time series analysis and its implications for the choice of the forecasting framework.

4.1 Machine Learing

It is generally agreed upon that the term ML refers to the field of study that gives computers the ability to learn without being explicitly programmed. This fact was first introduced by Arthur Samuel in 1959 Samuel (1959). However, note that the reference to this paper is used loosely, as the definition of ML was not directly used in the paper and it is rather a retrospective interpretation of the paper.

The term ML was more explicitly introduced by Tom M. Mitchell in 1997 Mitchell (1997):

Definition 4.1 (Machine Learning). A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

Typically, the experience E is represented by a dataset, which is used to train the model. We can generally say that ML is the ability to get better at specified task by learning from provided relevant data without problem domain specific programming of the computer.

Nowadays ML is at the core of many applications that we use daily. The usecases range from spam filters, recommendation systems, medical diagnosis, stock trading, and many more. Currently, machine learning is dominated by deep learning, which is a subfield of ML that focuses on deep neural networks that rely on large datasets in order to be able to generalize well. On the other, hand traditional ML algorithms are still widely used and are often the first choice when the dataset is small or the problem is low dimensional.

Whereas deep learning models can be used in high frequency data, where the datasets are large enough to train the model, daily closing stock price prediction is a task that can be successfully solved with traditional ML algorithms. As the datasets are much smaller.

In general, cryptocurrencies lie somewhere in the middle of the spectrum. Exactly as stocks, either high frequency or low frequency data can be chosen based on the research question. The difference however is that cryptocurrencies are much more volatile than stocks, and they can have much higher dimensionality as we can use many technical analysis indicators to predict the price. That is why we will use a combination of traditional ML algorithms and deep learning models to predict the closing prices or returns of various cryptocurrencies.

ML algorithms can be divided into three main categories: supervised learning, unsupervised learning, and reinforcement learning. As mentioned earlier, we will focus on supervised learning as the process of forecasting can be easily transformed into a supervised learning problem where the input features are historical or current data and the output is the future price or return.

It is important to note on the fundamental difference between machine learning and traditional econometric models. In econometrics the focus is to uncover the underlying relationship between the variables and to understand the size of contribution of each feature. In machine learning, the focus is mainly on maximizing the performance metric for predictions and the magnitude of the effects usually remains unknown. This is definitely a weakness of machine learning which researchers try to adress by developing new field of explainable Artificial Intelligence (AI). Where they focus on developing models that are able to explain their predictions in a human understandable way. They provide a significant promise for the use of ML methods in finance in the future where the interpretability of the model is crucial for customers or regulators in specific subfields.

4.2 Ridge Linear Regression

The simplest model that we can use for forecasting is the Linear Regression model. Generally, there exists a closed form solution for the LR model.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{4.1}$$

But in practice, we use gradient descent to find the optimal parameters. Gradient descent is an optimization algorithm used to minimize the cost function provided to the model. The idea is that we compute partial derivation of the error function with respect to each parameter of the model and update the parameters in the opposite direction of the gradient where the learning rate is the step size of the update. Usually, minibatch

gradient descent is used where the gradient is averaged over a small batch of randomly sampled data points.

The update rule for the parameters θ using gradient descent is given by:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \tag{4.2}$$

where α is the learning rate and $J(\theta)$ is the cost function. We have decided to use linear regression as a model representing the class of simple linear models. However, even though the model is simple, it can still suffer from overfitting the training data. There exists a relatively straight forward solution to this problem called Ridge regression. Ridge regression is a linear regression model with a regularization term added to the cost function which penalizes the model for having large weights.

The cost function for Ridge regression is given by:

$$J(\theta) = \text{MSE}(\theta) + \frac{\lambda}{2} \sum_{j=1}^{n} \theta_j^2 \tag{4.3}$$

where λ is the regularization parameter. Note that the bias term θ_0 is not regularized and that the regularization term is used only during training.

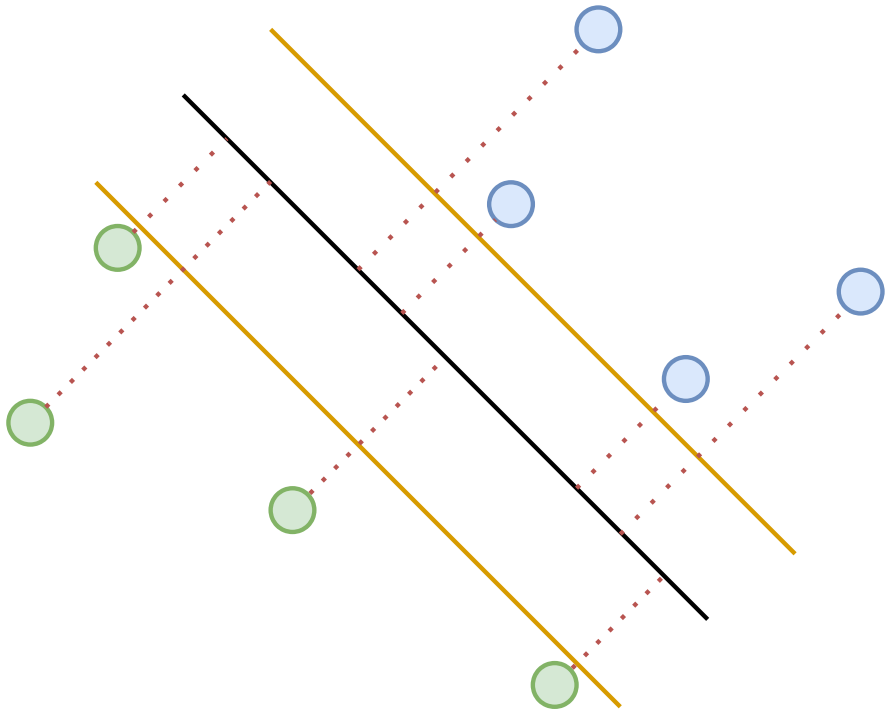
4.3 Support Vector Machines

Support Vector Machines are widely used ML algorithm which was expected to be the future of machine learning in the early 2000s. However, the rise of deep learning has overshadowed the success of SVMs. Nevertheless, SVMs are still widely used as they are really useful for small datasets and can capture complex nonlinear relationships. To ilustrate the idea of SVMs, let us consider a binary classification problem.

The idea is to find the hyperplane that maximizes the margin between the two classes. This implies that the hyperplane actually depends only on the support vectors which are the points closest to the hyperplane. This can be relatively easily reformulated into a regression problem by using the same idea but instead of classifying the points we are predicting the distance from the hyperplane. This is called the support vector regression. Note that the example in Figure 4.1 represents a problem that is linearly separable which is often not the case in practice and there are two remedies how to fix that. The first one is the usage of the kernel trick which is a way to transform the data into a higher dimensional space where the data is linearly separable. The second one is to use the SVM with a soft margin which is what most of the SVM implementations do. That allows us to set a hyperparameter that allows some points to be missclasified and acts as a form of regularization where we do not perfectly, seperate all of the points in the training set.

The optimization details of the SVMs are quite complex and are beyond the scope of

Figure 4.1: Support Vector Machines are minimizing the error by maximizing the margin between the two classes.



Source: Author

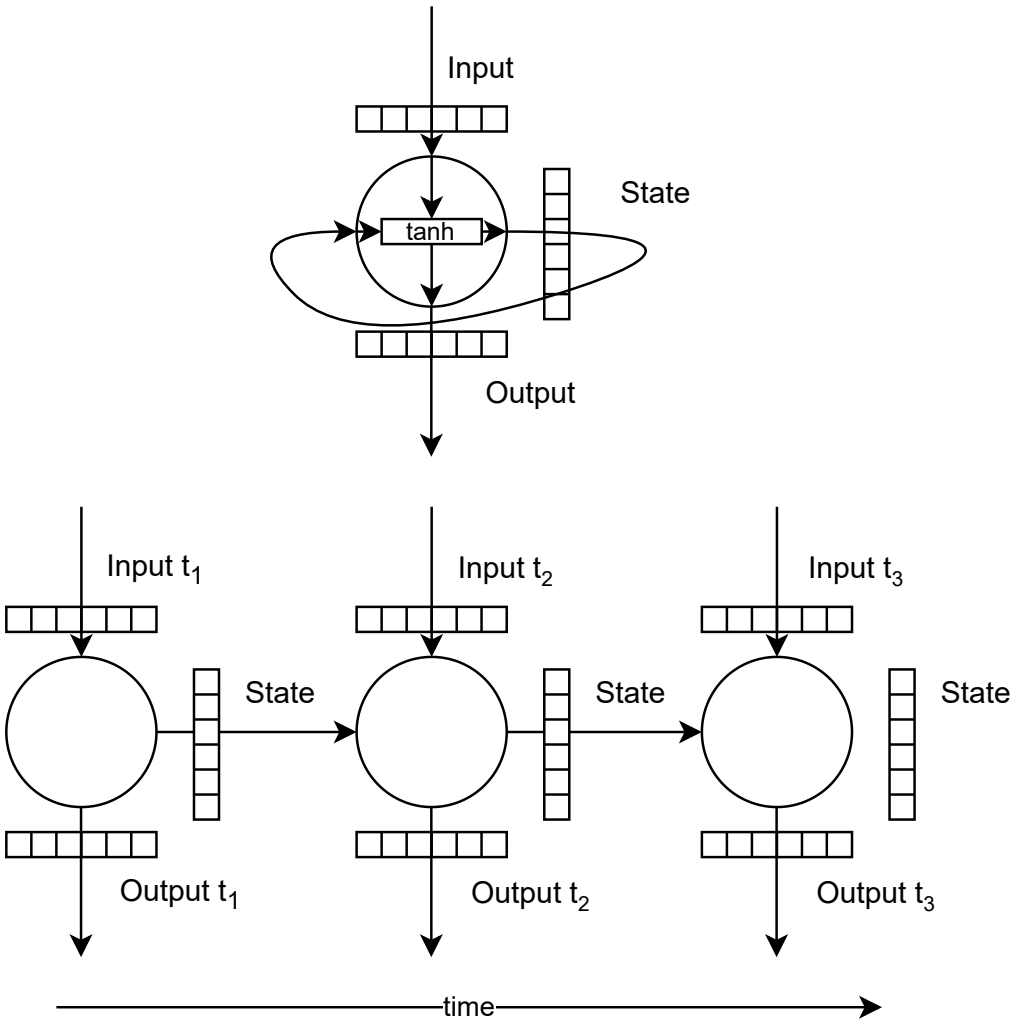
this text. They include the usage of the Lagrange multipliers and the dual optimization problem. The description of the kernel trick is not included as we opted for the linear kernel in our experiments. We thus refer the readers to some traditional ML textbooks such as Bishop & Nasrabadi (2006). The SVMs represent a middle ground between the traditional LR model and the deep learning models.

4.4 Long Short-Term Memory Recurrent Neural Networks

Recurrent Neural Networks could be regarded as the state of the art in time series forecasting. Their architecture allows them to process long sequential data of theoretically arbitrary length. They can capture the temporal dependencies in the data and learn sequential patterns. However, the traditional RNNs suffer from the vanishing gradient

problem which makes them unable to learn long term dependencies. This problem arises due to the fact that the neurons in RNNs are connected to themselves which makes them multiply the same weight matrix multiple times which can lead to the fact that the gradient becomes extremely small if the weights are small. This is the reason why the LSTM cell was introduced by Hochreiter and Schmidhuber in 1997 Hochreiter & Schmidhuber (1997). The idea is to make the errors constant through time by introducing few improvements to the original RNN cell.

Figure 4.2: Recurrent Neural Network Architecture unwinded in time is feeding outputs back into itself.



Source: Author

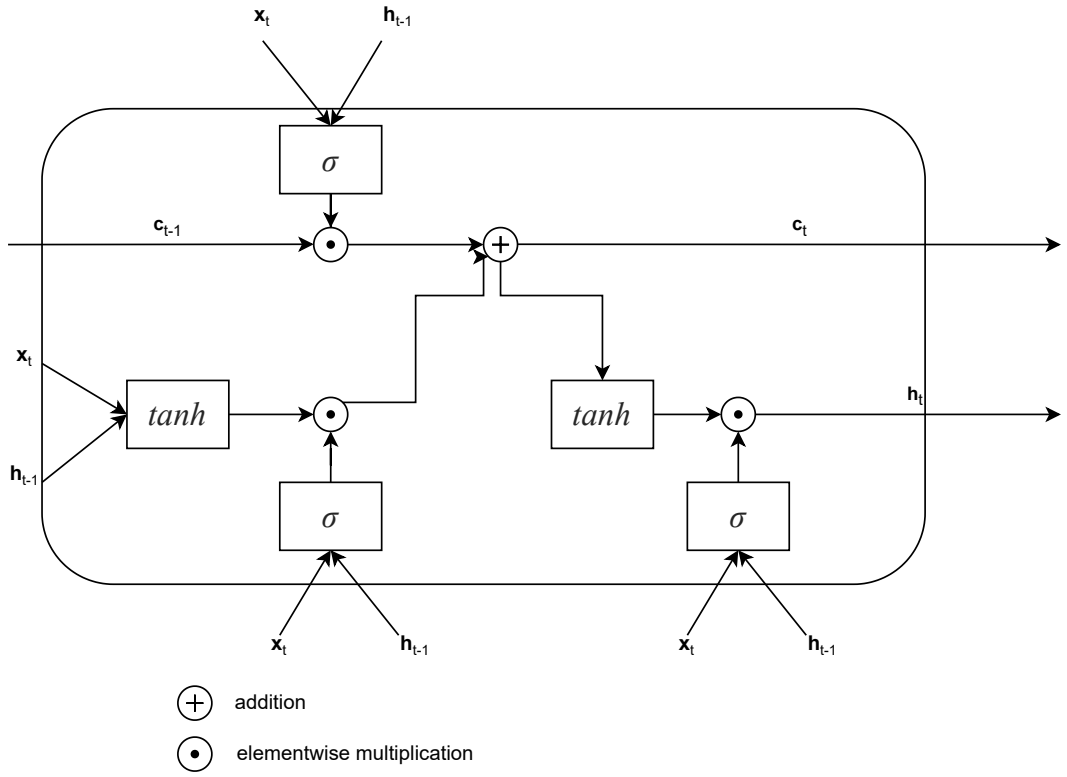
The name Recurrent Neural Network comes from the fact that the cell is able to feed backward into iself its internal state. Conceptually, this is a kind of memory or an

internal representation of the data. Output in the next time step is the combination of the current input and the internal state of the cell usually weighted using the tanh activation function. It is a common practice to to represent RNNs as a chain of cells unwined in time even though it is actually a single cell that is fed back into itself. It is possible to combine multiple such layers of these cells to create a deep RNN which is able to capture more complex patterns in the data.

Despite the promising idea of RNNs, they suffer from many training problems. Multiple remedies have been proposed to fix these problems such as the LSTM cell or the gated recurrent unit cell which are giving the model the ability to learn long term dependencies.

Since the introduction of the LSTM cell, many improvements have been made to the original architecture we will describe the improved architecture from Gers *et al.* (2000) which adds the forget gate to the original architecture.

Figure 4.3: LSTM cell enables consistent long term memory and stabilizes the training process.



Source: Author

The LSTM cell consists of three gates: the input gate, the forget gate, and the output gate. The input gate decides which input information is useful for the cell state. The forget gate decides which information to forget from the long term memory. And finally the output gate decides which information should be passed to the next state. The exact formulas are given in equations 4.4-4.8 where $\mathbf{W}, \mathbf{V}$ represent respective weights of the layers and b is the bias term. We can see that this architecture allow the cell to represent long term memory as the vector $\mathbf{c}$ and short term memory as the vector $\mathbf{h}$ as normal RNNs do.

4.5 Principal Component Analysis

PCA is a widely used dimensionality reduction technique that is able to reduce number of dimensions for the purpouses of visualization, noise reduction, and reduction of the curse of dimensionality. The curse of dimensionality refers to the observation that with increasing number of dimensions the volume of the space increases exponentially and the data points become extremely sparse leading to enormous distances between training

$$i_t = \sigma\left(\mathbf{W}^i x_t + \mathbf{V}^i h_{t-1} + \mathbf{b}^i\right) \tag{4.4}$$

$$f_t = \sigma\left(\mathbf{W}^f x_t + \mathbf{V}^f h_{t-1} + \mathbf{b}^f\right) \tag{4.5}$$

$$o_t = \sigma\left(\mathbf{W}^o x_t + \mathbf{V}^o h_{t-1} + \mathbf{b}^o\right) \tag{4.6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh\left(\mathbf{W}^y x_t + \mathbf{V}^y h_{t-1} + \mathbf{b}^y\right) \tag{4.7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{4.8}$$

points. The fact that PCA can be used as a noise reduction technique is technically a byproduct of the way how this algorithm works. We will start with the general idea to ilustrate how this concept emerges.

Conceptually, PCA can be thought of in two ways: maximum variance formulation and minimum reconstruction error formulation. We will follow with the maximum variance formulation. The idea is to find the hyperplane that maximizes the variance of the data projected onto it. Refer to Figure 4.4 for the illustration of the concept. We can see that the first principal component is the line that maximizes the variance of the projection of the data onto it. The second principal component captures less variance and is orthogonal to the first principal component. PCA can generally create as many principal components as there are dimensions in the data. This means that for n dimensions we can create n principal components which are orthogonal to each other and are odered by the amount of variance they capture. Finally we can reduce the number of dimensions by projecting the data onto the first k principal components where k is the number of dimensions we want to keep or the desired variance we want to preserve.

This is where the idea that PCA can be used as a noise reduction technique comes from. As this algorithm keeps only data which captures most of the variance and theoretically should drop information that is less informative. It is interesting to note that as the principal components are orthogonal to each other they are also uncorrleated which is a useful property for many ML algorithms. We acknowledge that PCA is not in itself a noise reduction technique but it is a rather elegant and simple way that sometimes reduces noise in the data. The implementation details of PCA are beyond the scope of this text but we refer the readers to the original paper Pearson (1901).

Figure 4.4: PCA projection where the red line is representing the variance maximizing hyperplane. All of the principal components are orthogonal to each other.

Source: Author

4.6 Stationarity in Time-Series

We would like to explicitly adress the concept of stationarity as there seems to be a lot of confusion around the topic especially because the boundary between traditional econometric models and machine learning models is not always clear.

Stationarity is a crucial concept in time-series analysis. Despite the fact, that there exist many tests and rigid definitions for it, it is often reduced to the concept of constant mean and variance or generally to the fact that the parameters of the distribution do not change in time. Without this property the errors of the model become function of time which is not desirable for proper inference.

Econometrics usually prefers stationary data for aforementioned reasons because it is cruicial for inference and interpretation of the results. That is why econometricians like to model returns instead of prices, as returns should generally be stationary. Imporantly, there is always the possibility to transform the predictions back to the original prices by adding the forecasted returns to the last observed price and potentially reversing some normalization steps.

This is different for ML as the focus is on prediction and the models are fundamentally different. As the models are trained using gradient descent and not using the analytical solutions, the stationarity is not as crucial. Clearly, the models will perform better on

stationary data because it requiers much less capacity for the model as some of the information removed by differencing. However, models with enough capacity can easily learn non-stationary data and capture information about trends, seasonalities, and other patterns. Thus we decided to test our models on both prices directly and on returns where the results are much less dependent on changes in the distribution of the data.

4.7 Proposed Forecasting Framework

We propose a forecasting framework which is designed to compare the performance between using PCA as a dimensionality reduction technique and using the raw data. The framework is shown in Figure 4.6. The preprocessing layer is responsible for cleaning the data, filling in missing values and tranforming the problem to supervised learning as described in 3. Following stage is responsible for reducing the number of features. The first PCA step transforms the data onto n principal components and the filtering step chooses the most important features such that their cumulative variance adds up to 95%, 98% or 99%. As decreasing the dimensionality would completely change the desired complexity of the algorithms and the differences might be attributed only to the change of dimensionality we upsample the data back to the original dimensionality using the inverse transformation of the PCA algorithm.

Following is the LSTM reshaping layer that is responsible for transforming the data into a 3D tensor for the LSTM RNN as described in 3. The most crucial layer is the forecasting layer that essentially consists of 3 parts. Firstly, it normalizes the variables using robust scaler to ensure that the input features come from the same value range. Secondly, it uses grid search to search for the best hyperparameters for the model. Lastly, it trains the model and evaluates the performance using the best found model. This is the reason why we abstract the metrics layer seperately to make it apparent that the metrics are calculated on the best model found by the grid search. The metrics layer is also the layer where we can statistically compare the significance of the difference between the models. It is important to note that this framework is executed across all specified forecasting models, cryptocurrencies of interest, and forecasting horizons as we expect the effects to be quite different for different models and forecasting horizons.

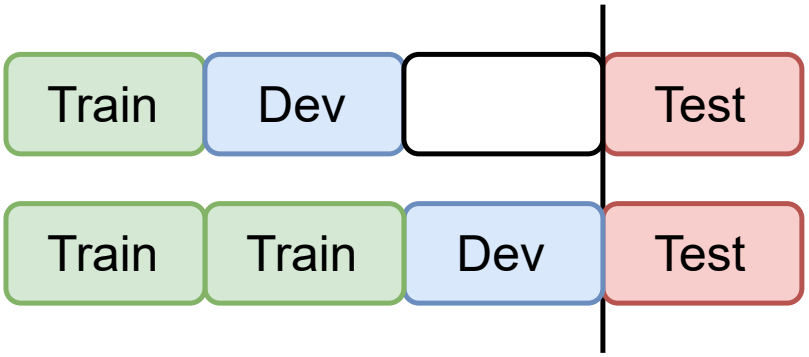
As we have already noted the fact that that the target price is clearly non-stationary, has important implication for the choice of splitting strategy for the grid search hyperparameter optimization. There are two main strategies for splitting the data into training and testing sets. The first one is traditional train, development, and test split where the model is trained on the training set, hyperparameters are optimized on the development set, and the model is evaluated on the test set. This approach is generally sufficient when the dataset is large enough and we believe that the distributions of the three datasets are similar. A more robust approach is to use grid search with cross-validation. This ap-

proach makes sure that we have not overfitted the model to the development set. Firstly, we split the data into training and testing set. And then we iteratively split the training set into training and validation set. Train the model on the training part of the train set and evaluate on the validation set. Finally we average over the results across all validation sets and take the best hyperparameters from the provided grid. This approach avoids overfitting and is typically used when we have enough compute to train the models.

Note that the goal of grid search is to create a robust representation of a reasonable test set distribution as it samples randomly the points into the training and validation set. This approach works if the data is stationary and the distribution of the data does not change drastically in time. But time series rarely have this property and cryptocurrencies suffer especially profound changes in the distribution as they have grown in popularity. The difference between the training and test set is extreme but that is something that we can do very little about as we want the test set to be as close to the future as possible.

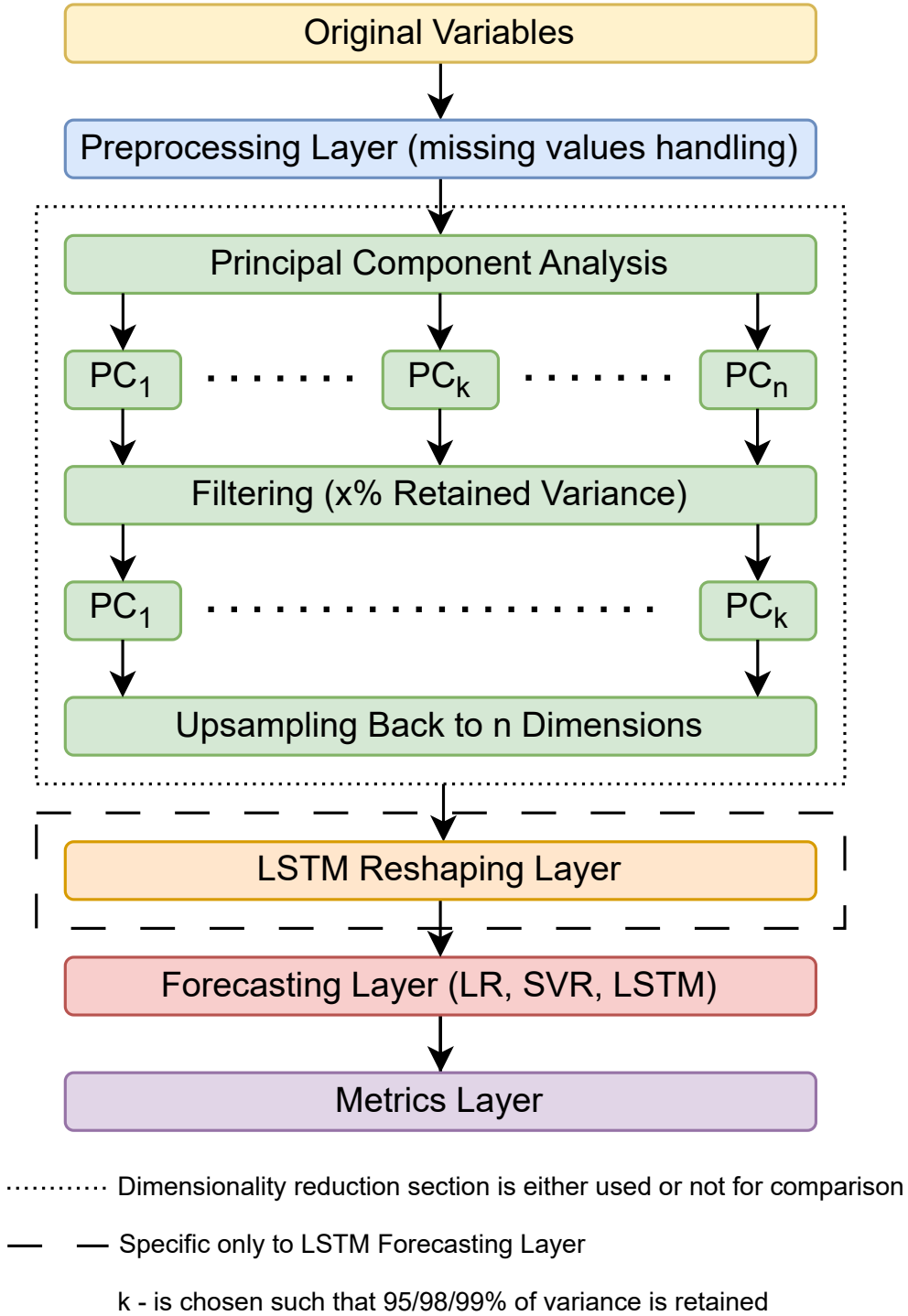
Additionally, there is an extremely common mistake that researchers make when they use grid search for time series data. This problem is called data leakage and it occurs when future distribution of the data is used to optimize the hyperparameters. As ML models typically require data to be scaled and normalized, the parameters for the normalization need to be learned only on the historical data not on the future data otherwise the results will be biased and suggest higher quality of the model than it actually is. Theoretically, this could be relaxed on the training set as the normalization might be fitted on the whole training set but it is crucial that the test set distribution is never leaked to the fitting procedure. However, we believe that the correct methodology is to use a rolling window approach for the grid search where the normalization parameters are fitted on the training part of the train set correctly distributed in time. This is why we use time series split for the grid search without randomly shuffling the data. This approach splits the training set into k folds and iteratively trains the model on *k-1* folds and evaluates on the *k-th* fold. This approach has the advantage that the model is actually trained with different sized training sets which is useful for generalization properties but has the disadvantage that the difficulty of the splits is extremely volatile. This is definitely a limitation as the performances between splits can vary significantly and averaging over the results might be prone to noise. However, we believe that this is the methodologically correct approach. We opted to use only two splits as the the data size is relatively small and the results were extremely noisy when using more splits.

Figure 4.5: Time Series Split with $k=2$ incrementally increases the training set size and leads to different cross validation training split sizes.



Source: Author

Figure 4.6: Proposed Forecasting Framework conists of four independent sequential layers.



Source: Author

Chapter 5

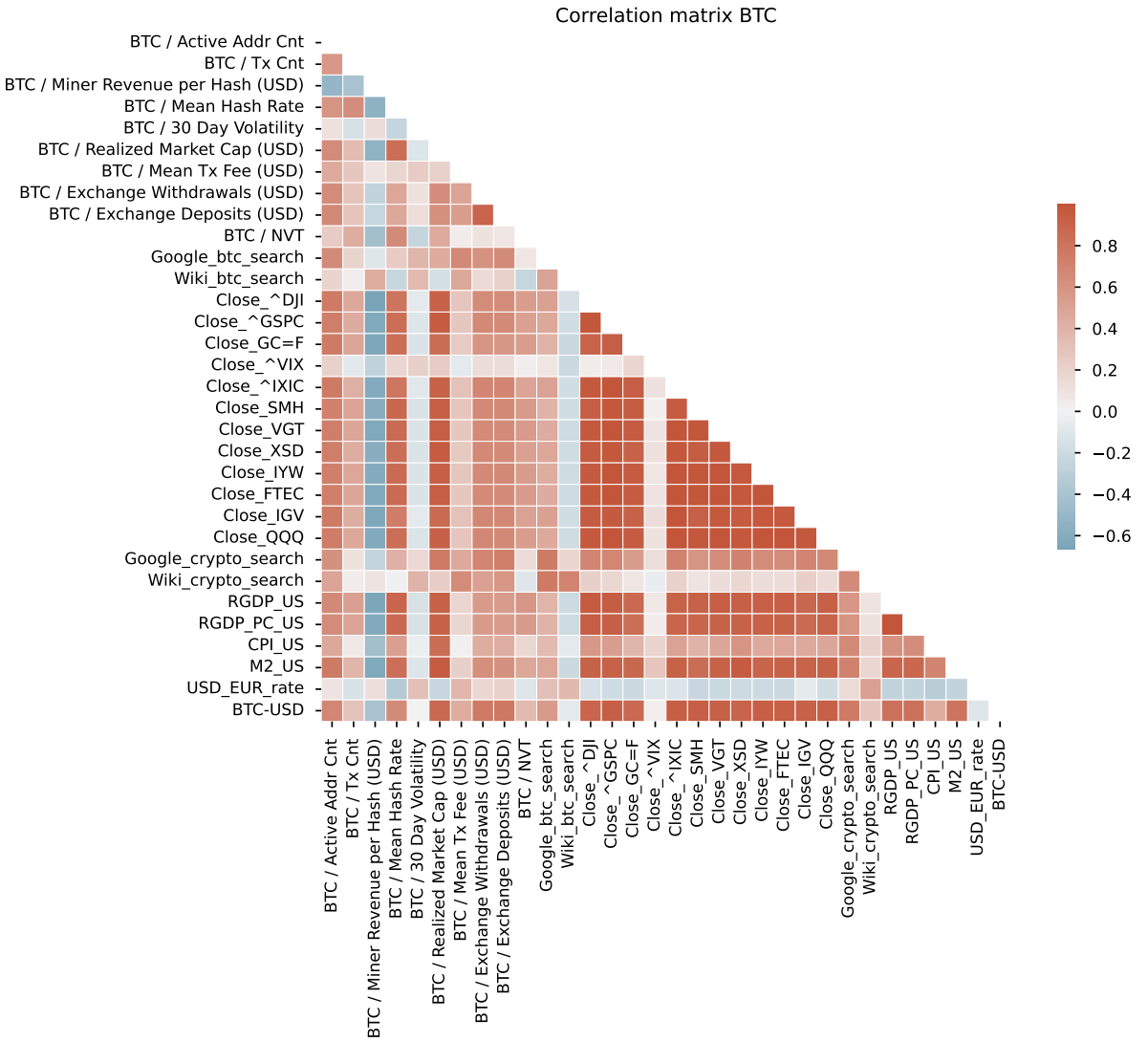
Results and Discussion

In this section we will discuss the results obtained from the performed experiments. We particularly discuss the overfitting phenomenon, which is a common problem for small sample sizes and for low signal to noise problems.

5.1 Results Intepretation

Initially, we intend to focus mainly on the price forecasting problem. After running many experiments with unsatisfactory results, we decided to add a set of technical indicators calculated from the price variable. Concretely, we added the following indicators: simple moving average (SMA), exponential moving average (EMA), relative strength index (RSI) and bollinger bands (BB). The reason for this decision being the presence of strong autocorrelation in the target variable which might be a signal of limited impact of the fundamental pricing side. In order to assess potential variable importance we have evaluated a correlation matrix for the initial dataset, please see Figure 5.1. High levels of multicollinearity highlight the importance of regularization techniques to avoid overfitting.

Figure 5.1: Correlation matrix of the BTC dataset shows high level of multi-collinearity.



Source: Author

$$\mathrm{SMA}_t = \frac{1}{w}\sum_{i=0}^{w-1} P_{t-i} \tag{5.1}$$

$$\mathrm{EMA}_t = \alpha P_t + (1-\alpha)\mathrm{EMA}_{t-1}, \quad \text{where} \quad \alpha = \frac{2}{w+1} \tag{5.2}$$

$$\mathrm{AvgGain}_t = \frac{1}{w}\sum_{i=1}^{w} \max(\Delta P_{t-i}, 0), \quad \mathrm{AvgLoss}_t = \frac{1}{w}\sum_{i=1}^{w} \max(-\Delta P_{t-i}, 0) \tag{5.3}$$

$$\mathrm{RS}_t = \frac{\mathrm{AvgGain}_t}{\mathrm{AvgLoss}_t}, \quad \mathrm{RSI}_t = 100 - \left(\frac{100}{1+\mathrm{RS}_t}\right) \tag{5.4}$$

$$\text{Upper}_t = \text{SMA}_t + k \cdot \sigma_t, \quad \text{Lower}_t = \text{SMA}_t - k \cdot \sigma_t \tag{5.5}$$

After the addition of the technical indicators, the results did not improve significantly and the model was always converging to the prediction of the current price which leads to a typical pattern in the graph when putting the predicted and actual prices into comparison where the prediction is essentially the target price shifted by the prediction horizon. We test this hypothesis by comparing the model with the naive forecast which predicts the last observed price and charaterizes the random walk model. The naive forecast outperforms the model in most of the cases on the test dataset.

In the following steps we decided to focus on the prediction of log returns. The log returns are calculated as follows:

$$r_t = \log(P_{t+h}) - \log(P_t) = \log\left(\frac{P_{t+h}}{P_t}\right) \tag{5.6}$$

where P_t is the price at time t and h is the time horizon. The idea behind this approach is that the log returns are stationary in nature and the model can thus better capture the pricing dynamics without learning the long term trend. Due to log returns stationarity, R^2 is a reasonable measure of the model performance that tells whether the model is better than the naive mean forecast. Positive R^2 indicates that the model is better than the mean forecast. Technically, R^2 is a relatively strict measure when the returns shift systematically in one direction on the test dataset. The reason is that the sum of squares total is calculated on the test data and thus knowing the test data mean. However, the models trained on the training dataset do not capture this shift as they do not know the test data mean. Despite that we believe that R^2 is a reasonable measure as the models should be able to outperform this baseline. Unfortunately, this problem reformulation did not help much and the model was still unable to generalize well to the test dataset despite decent performance on the training dataset. It is worth noting that most of our models have lower root mean square error on the test dataset than on the training dataset so they are not truly overfitting the training data.

We suggest multiple reasons for this behavior. Either the signal to noise ratio is too low and the model is not able to learn the pricing dynamics. The second reason is that the model is overfitting the training data despite extremely strong regularization of many forms and despite cross validation. We suspect that the combination of the small sample size and the nature of the cross validation are the main reasons for this behaviour.

For time series data, the temporal ordering of the data is important. That is the reason why one should not use the standard cross validation in order to avoid data leakage especially for the scalers that scale the data into a range which is required by some models. In practice, one can use normal cross validation on the training dataset but it will most likely lead to overfitting. The second option is to use the time series cross

validation which incrementally increases the training dataset and evaluates the model on the next batch of the data. This approach is more robust but has its own limitations. The first one is that when the sample size is small, the performance of the model is very noisy across the folds which makes it difficult to evaluate the best hyperparameters. Furthermore, varying sample size actually leads to different number of gradient descent iterations for some models which affects the choice of the best hyperparameters. The second weakness is actually the reasons that makes this approach more robust and that is the act of masking the future distribution of the data. As most machine learning models and certainly those used in our experiments including the PCA step expect the data to be standardized or scaled to some predefined range. When the distribution of the data changes significantly, the scalers trained on the training dataset will cause the test data to be scaled outside of those expected ranges and thus breaking the performance of the model. Unfortunately, this is the nature of our data and shows some limitations of the machine learning approach for problems that are changing over time.

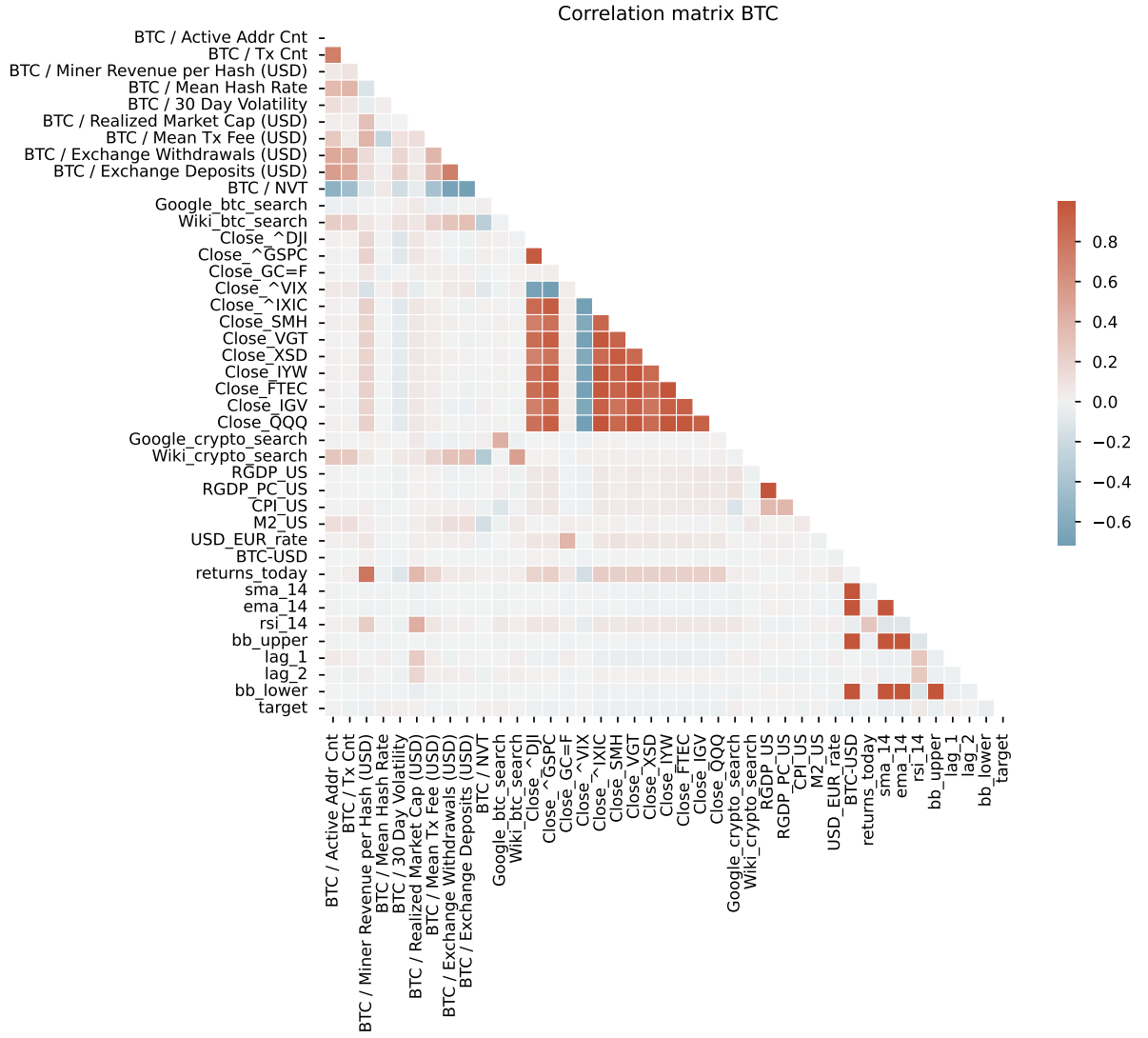
Due to afforementioned problems, we opted for a different approach which includes differencing and applying logarithm to all of the independent variables to push them closer to stationarity. This obviously leads to loss of information but may help with the cross validation problem. Especially, variables that come in weekly and monthly frequencies and are thus interpolated to daily frequency will be zero most of the time and thus may introduce a lot of variance on the edges where the values are changing. For the correlation matrix of this transformed dataset see Figure 5.2. We can observe that the correlation levels between the independent variables and the target variable are really low with highest levels being with the technical indicators.

After changing the model specification, we repeated the experiments using Bayesian cross validation which is an advanced technique that usually converges faster to better hyperparameters than the standard cross validation. We studied the effect of hyperparamters on the model performance and found that the results are mostly insensitive to the choice of the hyperparameters. Similarly, cross validation tends to select highly regularized models that are not able to learn the training data well and only predict the mean value. This is a typical sign of overfitting to the training noise and not to the signal.

As cryptomarket is characterized by significant structural breaks that shift the distribution of the data, we wanted to test whether the model is able to generalize well until the highly volatile period of 2021. We decided to cut the dataset at the beginning of 2021 and rerun the experiments. Howevery, the results were not significantly different and the generalization performance was still very poor.

We suspect that the signal to noise ratio is too low or the the pricing dynamics are changing too often which leads to the inability of the model to generalize to out-of-sample time periods. We inspect the latter hypothesis on the example of the learned coefficients of the Ridge regression model with incremental training that represents the

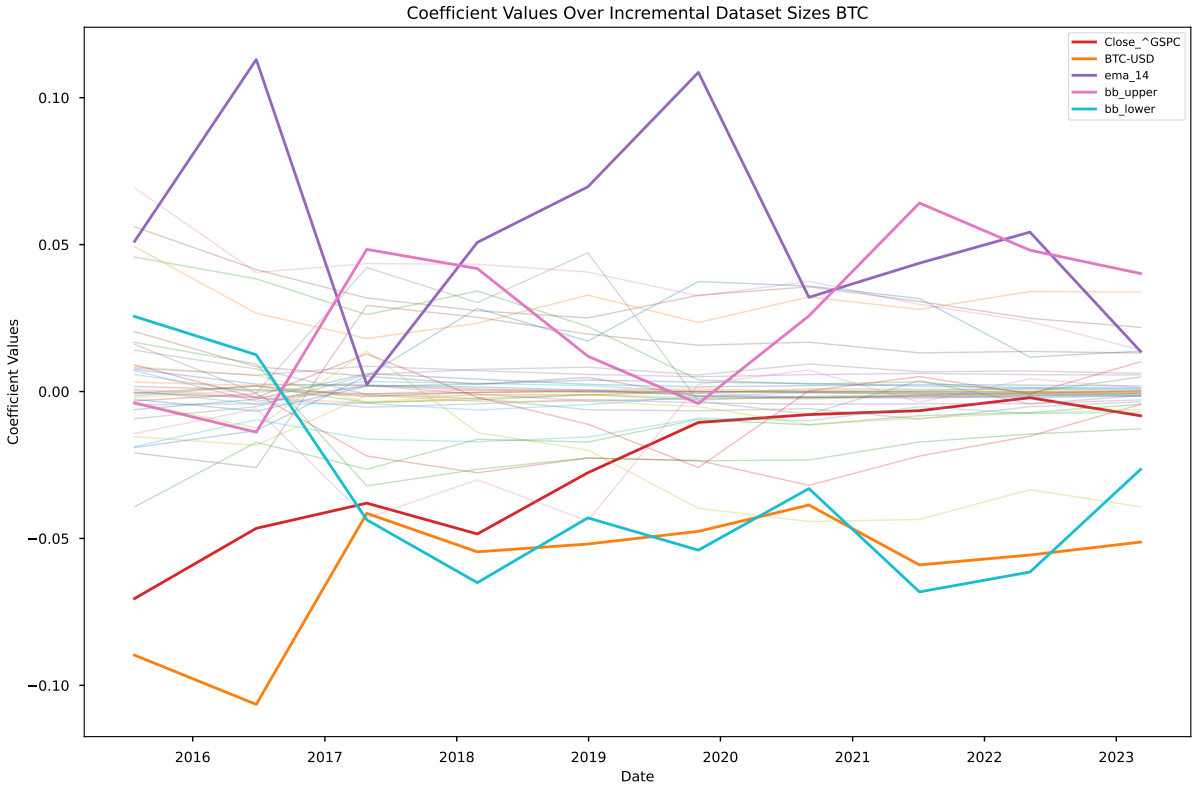
Figure 5.2: Correlation matrix of the BTC dataset after log differencing all of the variables.



Source: Author

time series split and the sliding window approach that represents the true changes in the dynamics in time. The learned coefficients are shown in Figures 5.3 and 5.4, where the coefficients with highest variance are highlighted. We can observe that the coefficients are changing over time relatively dramatically which explains the poor performance of the model. Obviously, some variance is expected due to the low size of these folds that causes noisiness in the learned coefficients especially in the sliding window approach. Similar results were obtained for the cryptocurrencies. To see the final results tables see Appendix A.

Figure 5.3: Learned coefficients of the Ridge regression model with incremental training on the BTC dataset. Five coefficients with highest variance are highlighted.



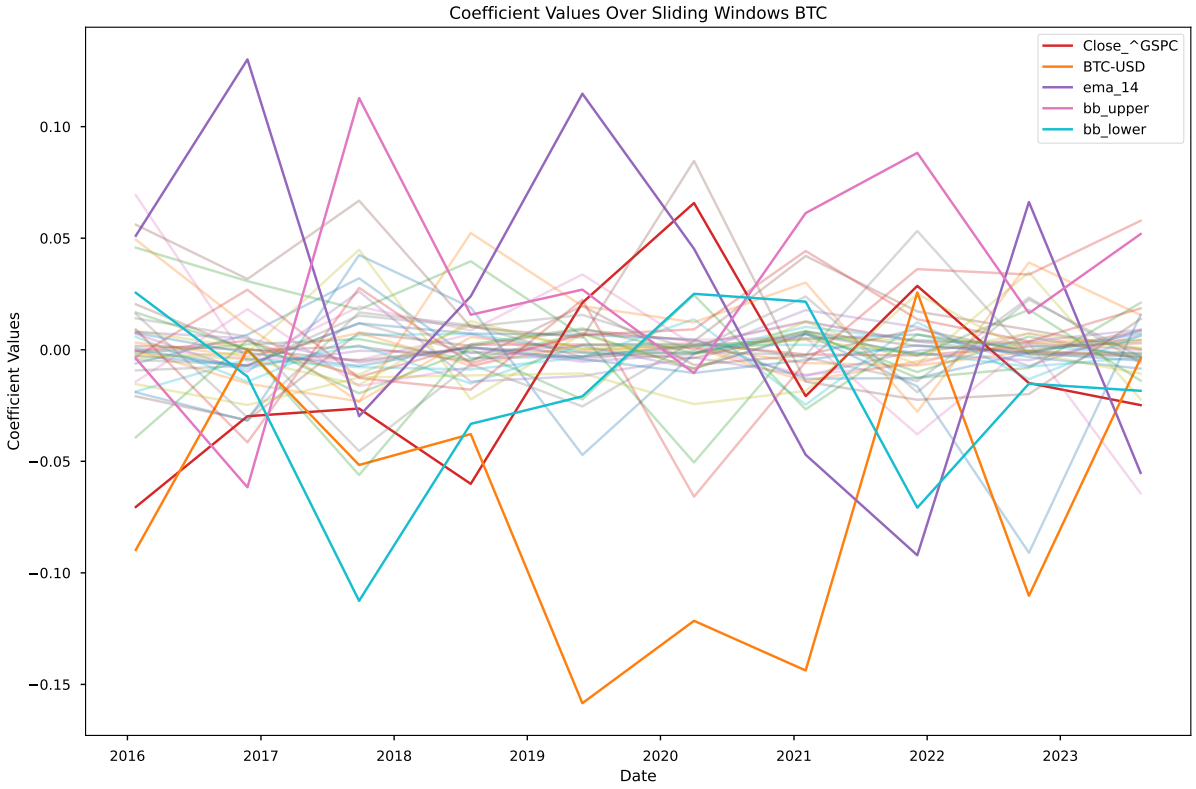
Source: Author

5.2 Limitations

Unfortunately, the poor model performance prevents us from making any conclusions about the initial hypothesis of the thesis. The results are not significantly different even with the PCA layer added. However, there seems to be some limited evidence for less overfitting behaviour in the case of the PCA model but the results are still worse than the naive forecast leading to negative R^2 on the test dataset. We conclude that the model is not able to learn the pricing dynamics due to the low signal to noise ratio and the changing pricing dynamics.

Our study is limited by the choice and quality of the data. Despite having a lot of variables representing the network activity and other fundamental variables, the quality of the trend data is low. Only the wikipedia page views come in daily frequency but we did not expect this variable to be highly informative. Whereas Google trends data was available in weekly frequency which leads to low information for daily forecasting. We believe that better data quality would improve the model performance significantly. Secondly, there might exist a better combination of technical indicators that would help the model to learn the pricing dynamics. Finally, more advanced techniques to increase the dataset

Figure 5.4: Learned coefficients of the Ridge regression model with sliding window training on the BTC dataset. Five coefficients with highest variance are highlighted.



Source: Author

size such as generative adversarial networks might help to improve the generalizability properties of the model.

Chapter 6

Conclusion

In this thesis, we have studied the existing literature on the use of machine learning in the field of cryptocurrencies and other topics in cryptocurrency pricing and prediction. Our aim was to provide a comprehensive overview of the current state of the art in this field and further develop on the gaps identified in the literature. We particularly focused on the use of PCA as a noise reduction technique.

As cryptocurrencies are a new and rapidly evolving field, there is a lack of high quality literature wich leaves a lot of room for improvement. The ability to predict cryptocurrency prices is of great interest for investors and traders, as it can lead to significant profits. Compared to traditional econometric analysis, our approach emphasizes the forecasting ability for out-of-sample data. Which is often overlooked in the literature.

We have iterated through many different model specifications described in the results section 5.1 and experimented with different cross-validation techniques. We conclude that due to poor model performance, we are unable to test whether the use of PCA as a noise reduction technique is beneficial for the prediction of cryptocurrency prices. Our results show the shortcomings of machine learning generalization in case of varying data distributions and signal sparse environments. We show why particular types of non-stationarity can be problematic for time series cross-validation schemes.

Despite the lack of significant results, we believe that we have made a strong argument for the reasons of out-of-sample performance deterioration. We conclude that poor data quality and the changing nature of the pricing dynamics are the main reasons for the poor model performance. We thus suggest that future research should focus on improving the data quality especially on the speculative side of the market. We also suggest to focus on data augmentation techniques to improve model generalization.

Bibliography

Akyildirim, E., A. Goncu, & A. Sensoy (2020): "Prediction of cryptocurrency returns using machine learning." *Annals of Operations Research* **297(1–2)**: pp. 3–36.

Alessandretti, L., A. ElBahrawy, L. M. Aiello, & A. Baronchelli (2018): "Anticipating Cryptocurrency Prices Using Machine Learning." *Complexity* **2018(1)**.

Arratia, A. & A. X. López-Barrantes (2021): "Do google trends forecast bitcoins? stylized facts and statistical evidence." *Journal of Banking and Financial Technology* .

Bishop, C. M. & N. M. Nasrabadi (2006): *Pattern recognition and machine learning*, volume 4. Springer.

Bouri, E., L. Kristoufek, T. Ahmad, & S. J. H. Shahzad (2022): "Microstructure noise and idiosyncratic volatility anomalies in cryptocurrencies." *Annals of Operations Research* .

Buterin, V. *et al.* (2013): "Ethereum white paper." *GitHub repository* **1**: pp. 22–23.

Chowdhury, R., M. A. Rahman, M. S. Rahman, & M. Mahdy (2020): "An approach to predict and forecast the price of constituents and index of cryptocurrency using machine learning." *Physica A: Statistical Mechanics and its Applications* **551**: p. 124569.

Chowdhury, U. N., S. K. Chakravarty, & M. T. Hossain (2018): "Short-term financial time series forecasting integrating principal component analysis and independent component analysis with support vector regression." *Journal of Computer and Communications* **06(03)**: pp. 51–67.

Conti, M., E. Sandeep Kumar, C. Lal, & S. Ruj (2018): "A Survey on Security and Privacy Issues of Bitcoin." *IEEE Communications Surveys amp; Tutorials* **20(4)**: pp. 3416–3452.

Dai, W. (1998): "B-money."

Dimpfl, T. & F. J. Peter (2021): "Nothing but noise? price discovery across cryptocurrency exchanges." *Journal of Financial Markets* **54**: p. 100584.

Fama, E. F. (1970): *Efficient Capital Markets A Review of Theory and Empirical Work*, pp. 76–121. Chicago: University of Chicago Press.

Farebrother, R. W. (2022): "Notes on the prehistory of principal components analysis." *Journal of Multivariate Analysis* **188**: p. 104814.

Gers, F. A., J. Schmidhuber, & F. Cummins (2000): "Learning to forget: Continual prediction with lstm." *Neural Computation* **12(10)**: pp. 2451–2471.

Ginsberg, J., M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, & L. Brilliant (2009): "Detecting influenza epidemics using search engine query data." *Nature* **457(7232)**: pp. 1012–1014.

Haber, S. & W. S. Stornetta (1991): *How to Time-Stamp a Digital Document*, pp. 437–455. Springer Berlin Heidelberg.

Hochreiter, S. & J. Schmidhuber (1997): "Long short-term memory." *Neural Computation* **9(8)**: pp. 1735–1780.

Hu, H., L. Tang, S. Zhang, & H. Wang (2018): "Predicting the direction of stock markets using optimized neural networks with google trends." *Neurocomputing* **285**: pp. 188–195.

Huang, M. Y., R. R. Rojas, & P. D. Convery (2019): "Forecasting stock market movements using google trend searches." *Empirical Economics* **59(6)**: pp. 2821–2839.

Jay, P., V. Kalariya, P. Parmar, S. Tanwar, N. Kumar, & M. Alazab (2020): "Stochastic neural networks for cryptocurrency price prediction." *IEEE Access* **8**: pp. 82804–82818.

Khedr, A. M., I. Arif, P. R. P V, M. El-Bannany, S. M. Alhashmi, & M. Sreedharan (2021): "Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey." *Intelligent Systems in Accounting, Finance and Management* **28(1)**: pp. 3–34.

Kleppmann, M. (2017): *Designing data-intensive applications*. Beijing: O'Reilley, first edition edition. Hier auch später erschienene, unveränderte Nachdrucke.

Kristjanpoller, W. & M. C. Minutolo (2018): "A hybrid volatility forecasting framework integrating garch, artificial neural network, technical analysis and principal components analysis." *Expert Systems with Applications* **109**: pp. 1–11.

Kristoufek, L. (2013): "Bitcoin meets google trends and wikipedia: Quantifying the relationship between phenomena of the internet era." *Scientific Reports* **3(1)**.

Kristoufek, L. (2023): "Will Bitcoin ever become less volatile?" *Finance Research Letters* **51**: p. 103353.

Kubal, J. & L. Kristoufek (2022): "Exploring the relationship betweenb Bitcoin price and network's hashrate within endogenous system." *International Review of Financial Analysis* **84**: p. 102375.

Kukacka, J. & L. Kristoufek (2023): "Fundamental and speculative components of the cryptocurrency pricing dynamics." *Financial Innovation* **9(1)**.

Lazer, D., R. Kennedy, G. King, & A. Vespignani (2014): "The parable of google flu: Traps in big data analysis." *Science* **343(6176)**: pp. 1203–1205.

Lucas, S. (2021): "The origins of the halting problem." *Journal of Logical and Algebraic Methods in Programming* **121**: p. 100687.

Mitchell, T. M. (1997): *Machine learning*. McGraw-Hill international editions. New York [u.a.]: McGraw-Hill, [nachdr.] edition.

Mohsin, M., S. Naseem, L. Ivașcu, L.-I. Cioca, M. Sarfraz, & N. C. Stănică (2021): "Gauging the effect of investor sentiment on cryptocurrency market: an analysis of bitcoin currency." *Romanian Journal of Economic Forecasting* **24(4)**: p. 87.

Möser, M. & R. Böhme (2015): *Trends, Tips, Tolls: A Longitudinal Study of Bitcoin Transaction Fees*, pp. 19–33. Springer Berlin Heidelberg.

Nadkarni, J. & R. Ferreira Neves (2018): "Combining neuroevolution and principal component analysis to trade in the financial markets." *Expert Systems with Applications* **103**: pp. 184–195.

Nakamoto, S. (2008): "Bitcoin: A peer-to-peer electronic cash system." *Decentralized business review* .

Padmavathi, M. & R. M. Suresh (2018): "Secure P2P Intelligent Network Transaction using Litecoin." *Mobile Networks and Applications* **24(2)**: pp. 318–326.

Pearson, K. (1901): "Liii. on lines and planes of closest fit to systems of points in space." *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2(11)**: pp. 559–572.

Ren, Y.-S., C.-Q. Ma, X.-L. Kong, K. Baltas, & Q. Zureigat (2022): "Past, present, and future of the application of machine learning in cryptocurrency research." *Research in International Business and Finance* **63**: p. 101799.

Samuel, A. L. (1959): "Some studies in machine learning using the game of checkers." *IBM Journal of Research and Development* **3(3)**: pp. 210–229.

Shah, A., Y. Chauhan, & B. Chaudhury (2021): "Principal component analysis based construction and evaluation of cryptocurrency index." *Expert Systems with Applications* **163**: p. 113796.

Smales, L. A. (2020): "One cryptocurrency to explain them all? understanding the importance of bitcoin in cryptocurrency returns." *Economic Papers: A journal of applied economics and policy* **39(2)**: pp. 118–132.

Tikhomirov, S. (2018): *Ethereum: State of Knowledge and Research Perspectives*, pp. 206–221. Springer International Publishing.

Toledo, J. d. M. & D. Y. Souza (2022): "Signal prediction in cryptocurrency tradeoperations: A machine learning-based approach." *SSRN Electronic Journal* .

Turing, A. M. *et al.* (1936): "On computable numbers, with an application to the Entscheidungsproblem." *J. of Math* **58(345-363)**: p. 5.

Urquhart, A. (2016): "The inefficiency of bitcoin." *Economics Letters* **148**: pp. 80–82.

West, R. (2020): "Calibration of google trends time series." In "Proceedings of the 29th ACM International Conference on Information amp; Knowledge Management," volume 457 of *CIKM '20*, pp. 2257–2260. ACM.

Wolpert, D. H., W. G. Macready *et al.* (1995): "No free lunch theorems for search." *Technical report*, Citeseer.

Wątorek, M., M. Skupień, J. Kwapień, & S. Drożdż (2023): "Decomposing cryptocurrency high-frequency price dynamics into recurring and noisy components." *Chaos: An Interdisciplinary Journal of Nonlinear Science* **33(8)**.

Appendix A

Detailed Results Tables

Table A.1: Results of the Ridge Regression Model Train Dataset, R^2 metric, BTC

	BTC-LR - 1 day	BTC-LR - 5 days	BTC-LR - 10 days
Full dimensionality	0.01330	0.03727	0.05569
95% retained variance	0.01097	0.03391	0.03929
98% retained variance	0.01143	0.03533	0.05464
99% retained variance	0.01189	0.03600	0.05466

Table A.2: Results of the SVR Model Train Dataset, R^2 metric, BTC

	BTC-SVR - 1 day	BTC-SVR - 5 days	BTC-SVR - 10 days
Full dimensionality	-0.00226	-0.01158	-0.01658
95% retained variance	-0.00226	-0.01136	-0.02180
98% retained variance	-0.00226	-0.01154	-0.01780
99% retained variance	-0.00226	-0.01153	-0.01837

Table A.3: Results of the LSTM Model Train Dataset, R^2 metric, BTC

	BTC-LSTM - 1 day	BTC-LSTM - 5 days	BTC-LSTM - 10 days
Full dimensionality	-0.05479	2e-05	0.09113
95% retained variance	-0.02	-0.00043	-0.46234
98% retained variance	-0.38582	-0.00224	0.07469
99% retained variance	0.00122	-4e-05	0.01605

Table A.4: Results of the Ridge Model Test Dataset, R^2 metric, BTC

	BTC-LR - 1 day	BTC-LR - 5 days	BTC-LR - 10 days
Full dimensionality	-0.00484	-0.01428	-0.04277
95% retained variance	0.00050	-0.01568	-0.01613
98% retained variance	-0.00022	-0.01340	-0.04184
99% retained variance	-0.00170	-0.01352	-0.04186

Table A.5: Results of the SVR Model Test Dataset, R^2 metric, BTC

	BTC-SVR - 1 day	BTC-SVR - 5 days	BTC-SVR - 10 days
Full dimensionality	-0.00037	-0.00221	-0.00667
95% retained variance	-0.00037	-0.00202	-0.00368
98% retained variance	-0.00037	-0.00243	-0.00535
99% retained variance	-0.00037	-0.00199	-0.00517

Table A.6: Results of the LSTM Model Test Dataset, R^2 metric, BTC

	BTC-LSTM - 1 day	BTC-LSTM - 5 days	BTC-LSTM - 10 days
Full dimensionality	-0.1228	-0.01189	-0.03256
95% retained variance	-0.00583	-0.00716	-2.81868
98% retained variance	-1.58044	0.00012	-0.27972
99% retained variance	-0.00477	-0.01145	-0.02553

Table A.7: Results of the Ridge Regression Model Train Dataset, R^2 metric, ETH

	ETH-LR - 1 day	ETH-LR - 5 days	ETH-LR - 10 days
Full dimensionality	0.01709	0.03234	0.04851
95% retained variance	0.01332	0.02988	0.04177
98% retained variance	0.01373	0.0302	0.04185
99% retained variance	0.01687	0.03137	0.04682

Table A.8: Results of the SVR Model Train Dataset, R^2 metric, ETH

	ETH-SVR - 1 day	ETH-SVR - 5 days	ETH-SVR - 10 days
Full dimensionality	-0.00398	-0.01759	-0.02885
95% retained variance	-0.00398	-0.01778	-0.02787
98% retained variance	-0.00398	-0.0177	-0.02888
99% retained variance	-0.00398	-0.01748	-0.02891

Table A.9: Results of the LSTM Model Train Dataset, R^2 metric, ETH

	ETH-LSTM - 1 day	ETH-LSTM - 5 days	ETH-LSTM - 10 days
Full dimensionality	0.0001	0.00024	0.00266
95% retained variance	0.0001	0.01698	-0.00544
98% retained variance	-0.00672	0.00013	-0.03365
99% retained variance	-0.28841	-0.02426	0.03163

Table A.10: Results of the Ridge Model Test Dataset, R^2 metric, ETH

	ETH-LR - 1 day	ETH-LR - 5 days	ETH-LR - 10 days
Full dimensionality	-0.0029	-0.02056	-0.02534
95% retained variance	-0.00278	-0.01793	-0.0089
98% retained variance	-0.00212	-0.01892	-0.00918
99% retained variance	-0.00267	-0.02003	-0.02768

Table A.11: Results of the SVR Model Test Dataset, R^2 metric, ETH

	ETH-SVR - 1 day	ETH-SVR - 5 days	ETH-SVR - 10 days
Full dimensionality	-0.0022	-0.00038	0.00065
95% retained variance	-0.00219	-0.0006	0.00065
98% retained variance	-0.0022	-0.00047	0.00058
99% retained variance	-0.0022	-0.00011	0.00059

Table A.12: Results of the LSTM Model Test Dataset, R^2 metric, ETH

	ETH-LSTM - 1 day	ETH-LSTM - 5 days	ETH-LSTM - 10 days
Full dimensionality	-0.00363	-0.0535	-0.03034
95% retained variance	0.00068	-0.06966	0.01329
98% retained variance	-0.07611	-0.05646	-0.29143
99% retained variance	-0.36521	0.00013	-0.02966

Table A.13: Results of the Ridge Regression Model Train Dataset, R^2 metric, LTC

	LTC-LR - 1 day	LTC-LR - 5 days	LTC-LR - 10 days
Full dimensionality	0.00985	0.03449	0.05824
95% retained variance	0.00921	0.03073	0.03773
98% retained variance	0.0093	0.03362	0.0463
99% retained variance	0.00956	0.03425	0.04657

Table A.14: Results of the SVR Model Train Dataset, R^2 metric, LTC

	LTC-SVR - 1 day	LTC-SVR - 5 days	LTC-SVR - 10 days
Full dimensionality	-0.00056	0.00795	-0.00501
95% retained variance	-0.00056	-0.00291	-0.00502
98% retained variance	-0.00056	-0.00283	0.00095
99% retained variance	-0.00056	-0.00291	-0.00501

Table A.15: Results of the LSTM Model Train Dataset, R^2 metric, LTC

	LTC-LSTM - 1 day	LTC-LSTM - 5 days	LTC-LSTM - 10 days
Full dimensionality	-1e-05	0.0262	0.04409
95% retained variance	-0.02324	0.02099	0.02187
98% retained variance	0.00271	0.01406	-0.0459
99% retained variance	-0.00159	-0.04197	0.03507

Table A.16: Results of the Ridge Model Test Dataset, R^2 metric, LTC

	LTC-LR - 1 day	LTC-LR - 5 days	LTC-LR - 10 days
Full dimensionality	0.00293	-0.00597	-0.06626
95% retained variance	0.00398	-0.00194	-0.0241
98% retained variance	0.00347	-0.00486	-0.01171
99% retained variance	0.00297	-0.00495	-0.00956

Table A.17: Results of the SVR Model Test Dataset, R^2 metric, LTC

	LTC-SVR - 1 day	LTC-SVR - 5 days	LTC-SVR - 10 days
Full dimensionality	-0.00035	0.00435	-0.00409
95% retained variance	-0.00035	-0.00195	-0.0041
98% retained variance	-0.00035	-0.00198	-0.00213
99% retained variance	-0.00035	-0.00194	-0.00409

Table A.18: Results of the LSTM Model Test Dataset, R^2 metric, LTC

	LTC-LSTM - 1 day	LTC-LSTM - 5 days	LTC-LSTM - 10 days
Full dimensionality	-0.00246	-0.02475	-0.02862
95% retained variance	-0.01891	-0.0238	-0.02381
98% retained variance	-0.00205	-0.02936	-0.04612
99% retained variance	0.00146	-0.1762	-0.03606

Appendix B

Additional Figures

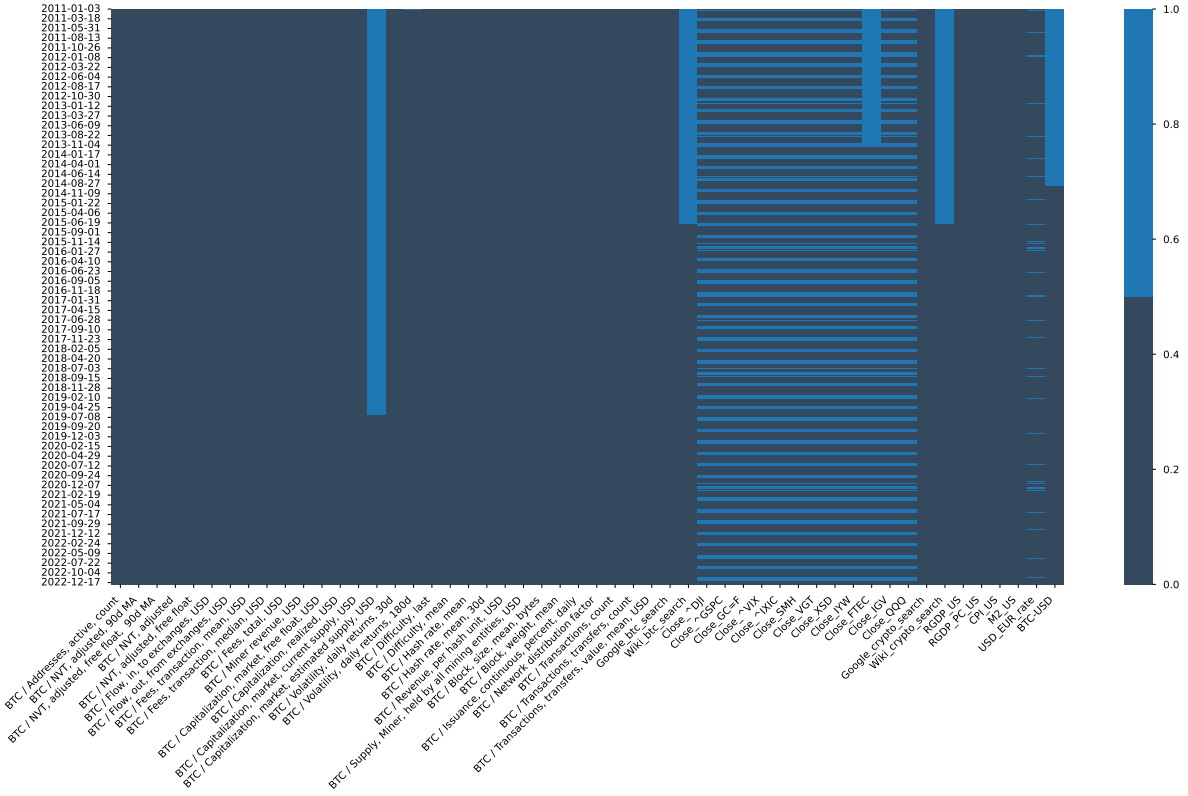


Figure B.1: Missing Values BTC before subsetting

Source: Author

Figure B.2: Missing Values ETH before subsetting

Source: Author

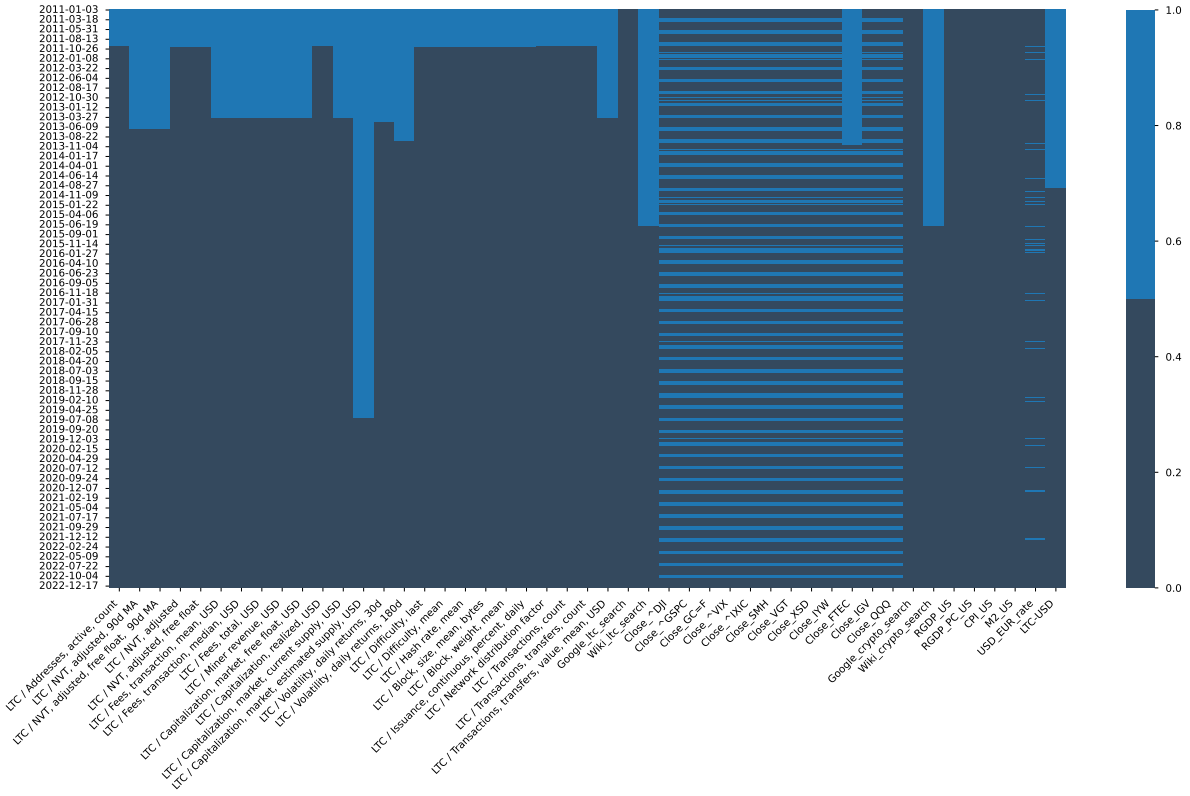
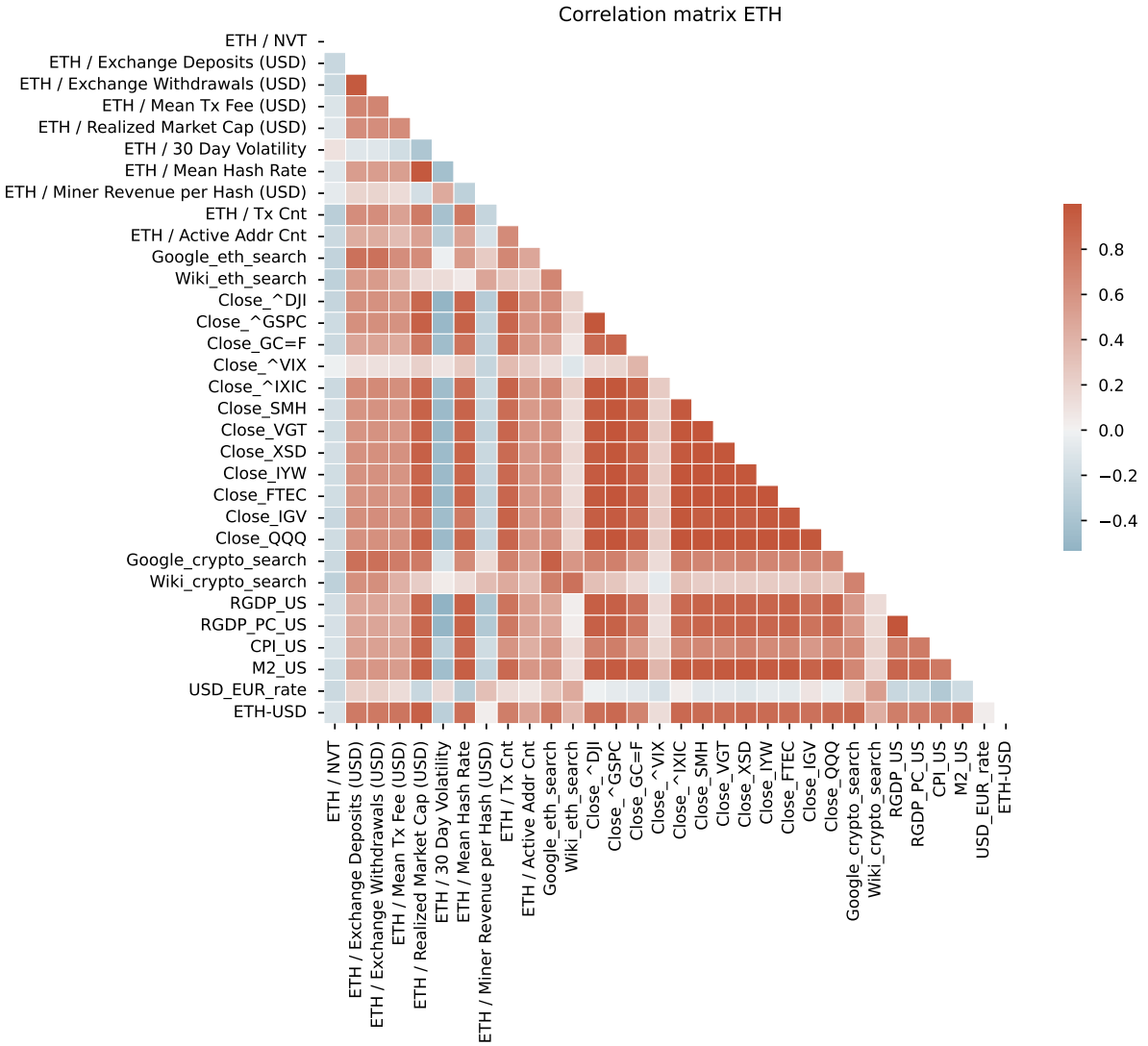


Figure B.3: Missing Values LTC before subsetting

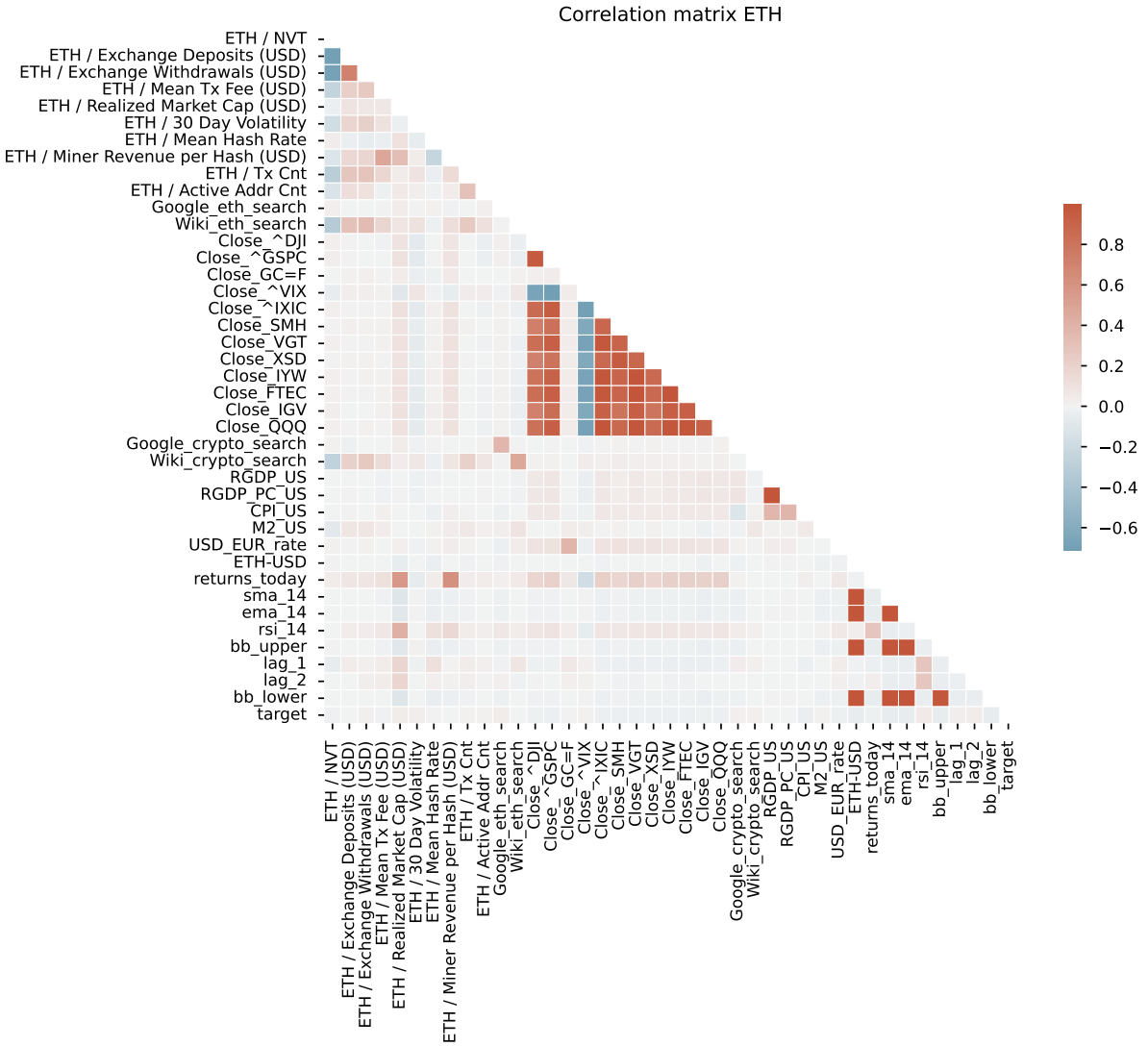
Source: Author

Figure B.4: Correlation matrix of the ETH dataset shows high level of multicollinearity.



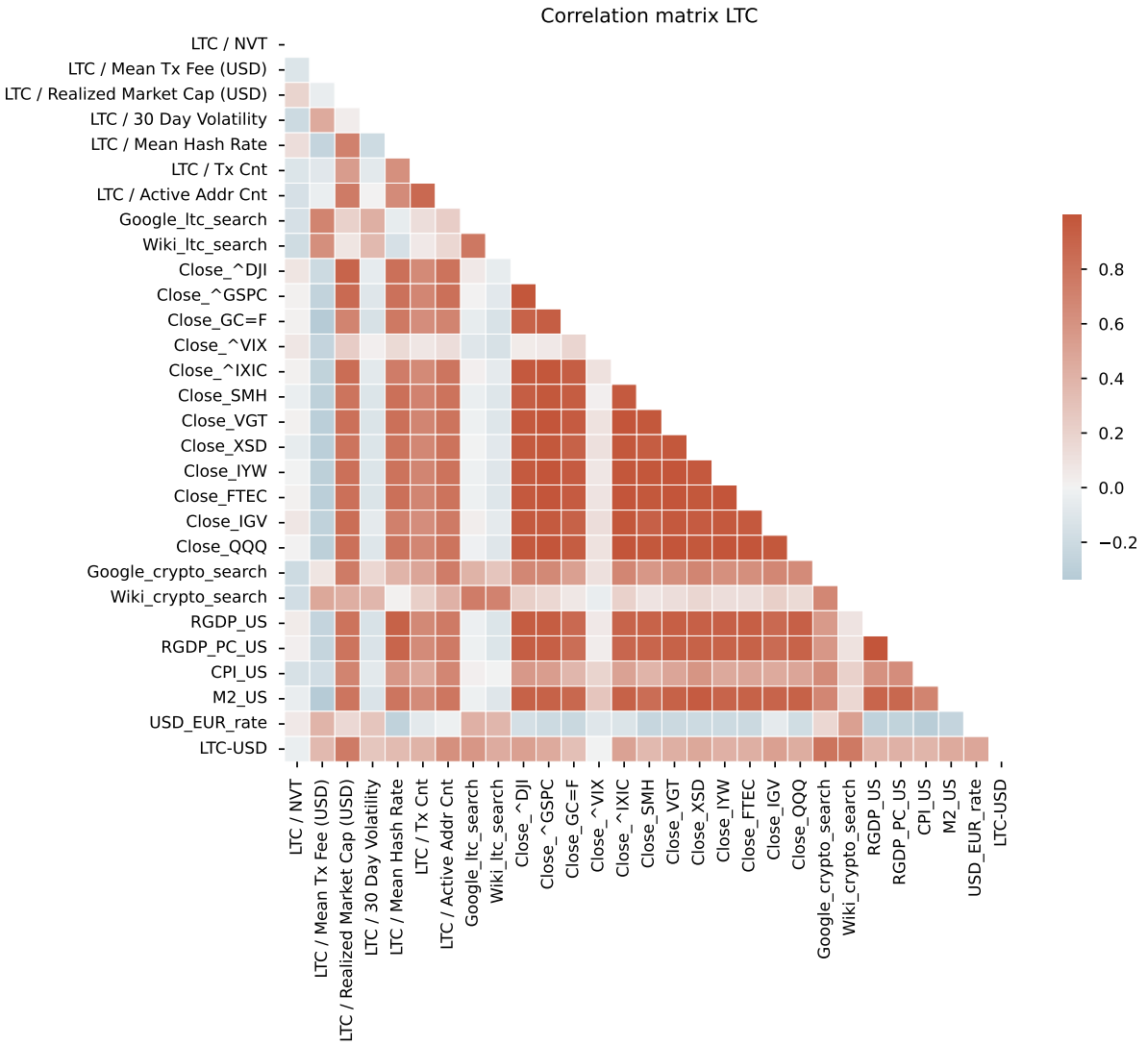
Source: Author

Figure B.5: Correlation matrix of the ETH dataset after log differencing all of the variables.



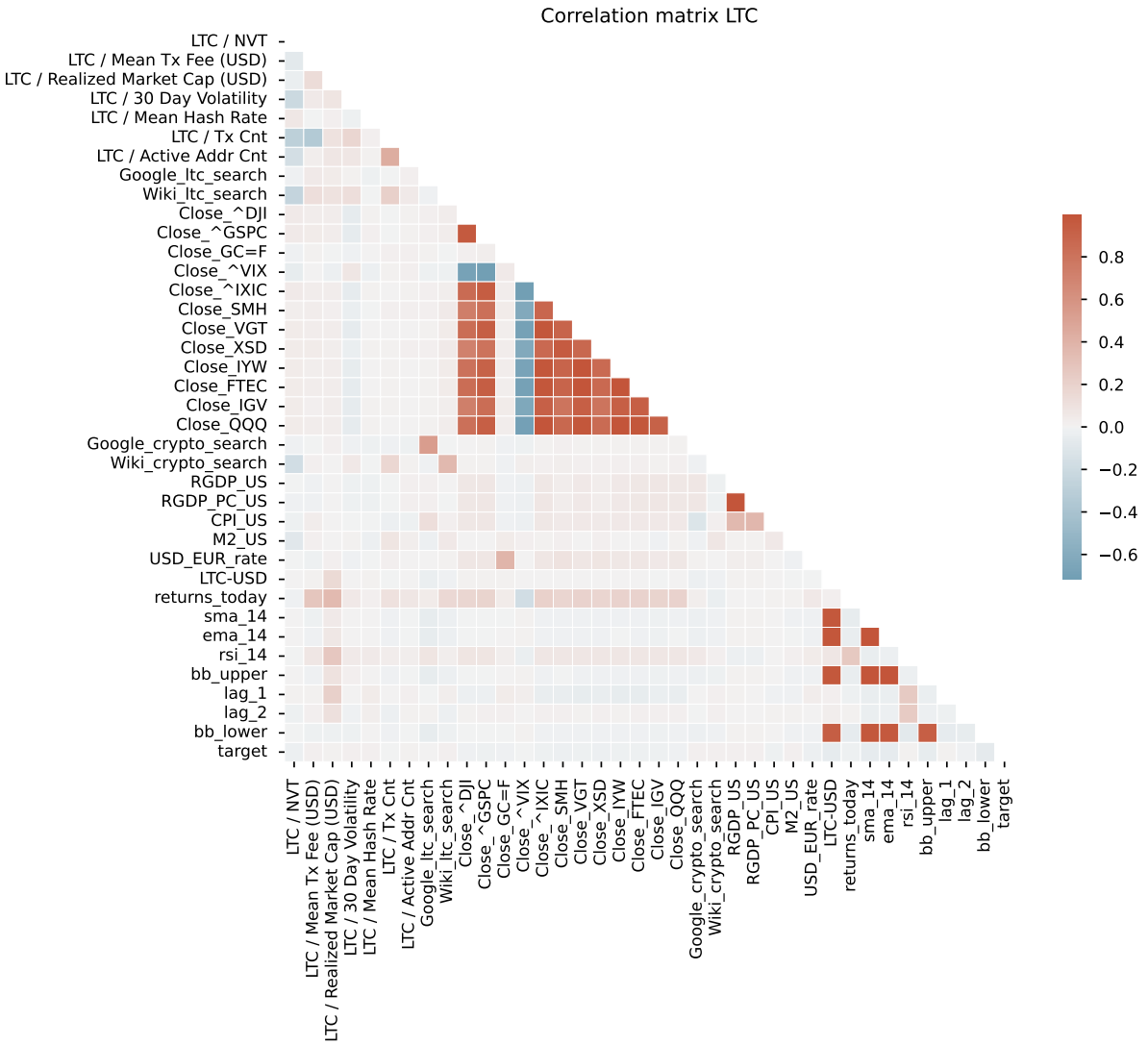
Source: Author

Figure B.6: Correlation matrix of the LTC dataset shows high level of multicollinearity.



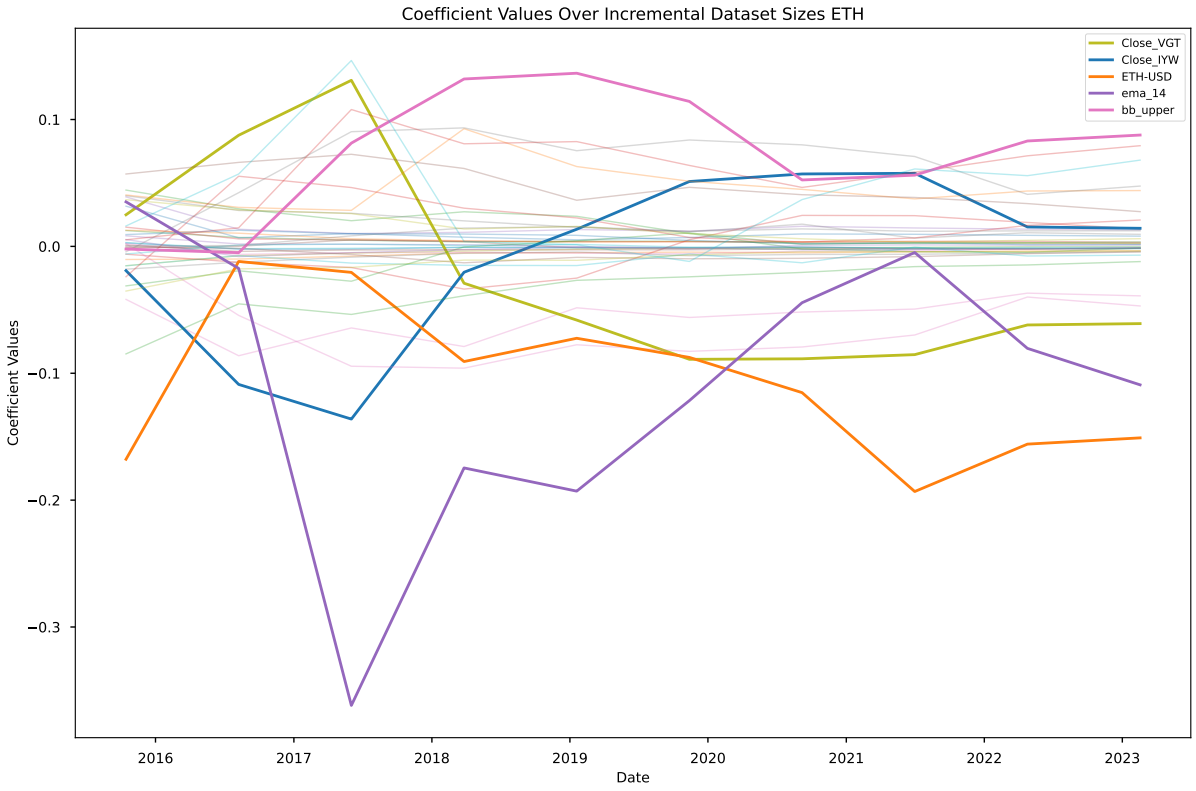
Source: Author

Figure B.7: Correlation matrix of the LTC dataset after log differencing all of the variables.



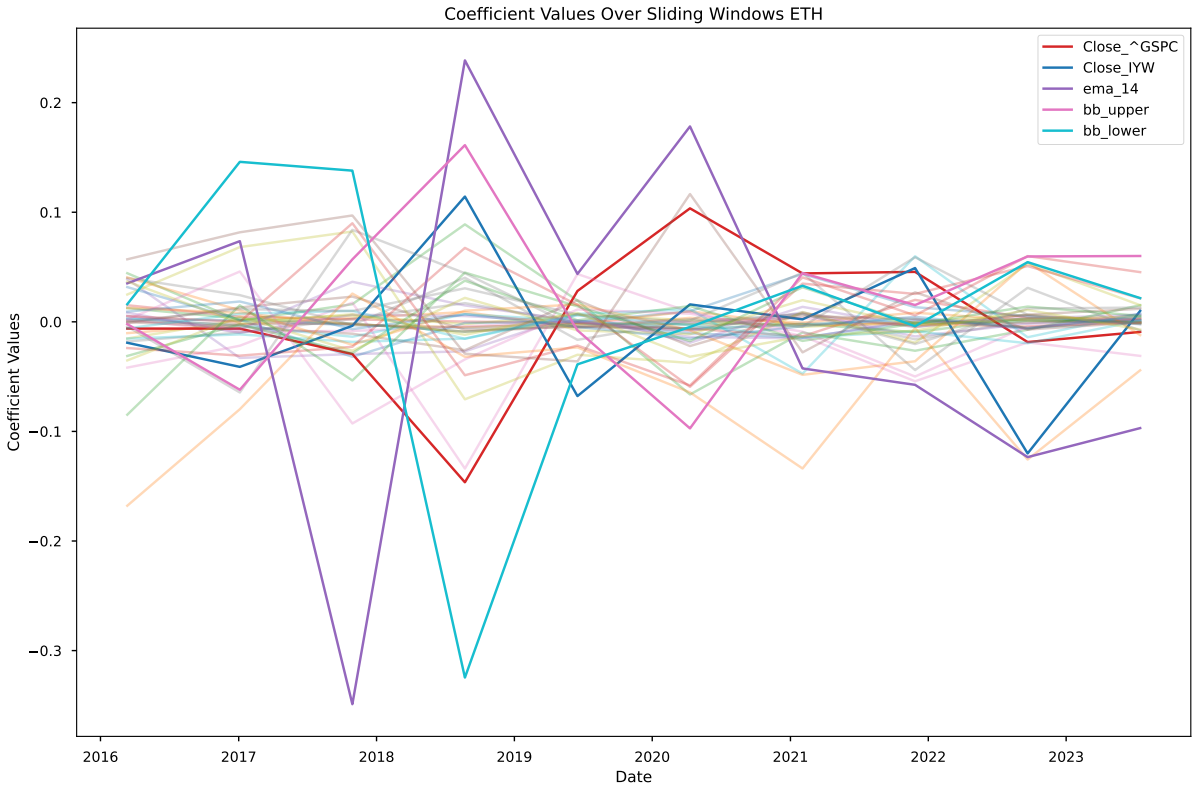
Source: Author

Figure B.8: Learned coefficients of the Ridge regression model with incremental training on the ETH dataset. Five coefficients with highest variance are highlighted.



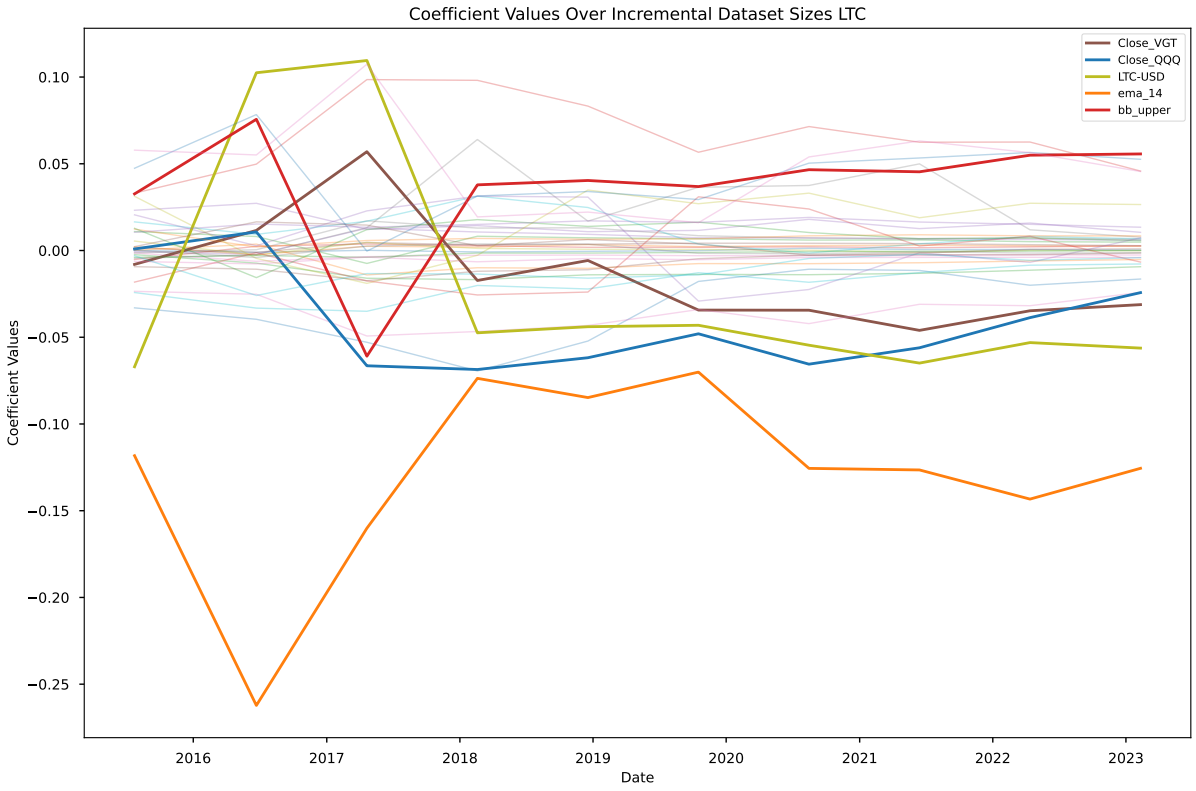
Source: Author

Figure B.9: Learned coefficients of the Ridge regression model with sliding window training on the ETH dataset. Five coefficients with highest variance are highlighted.



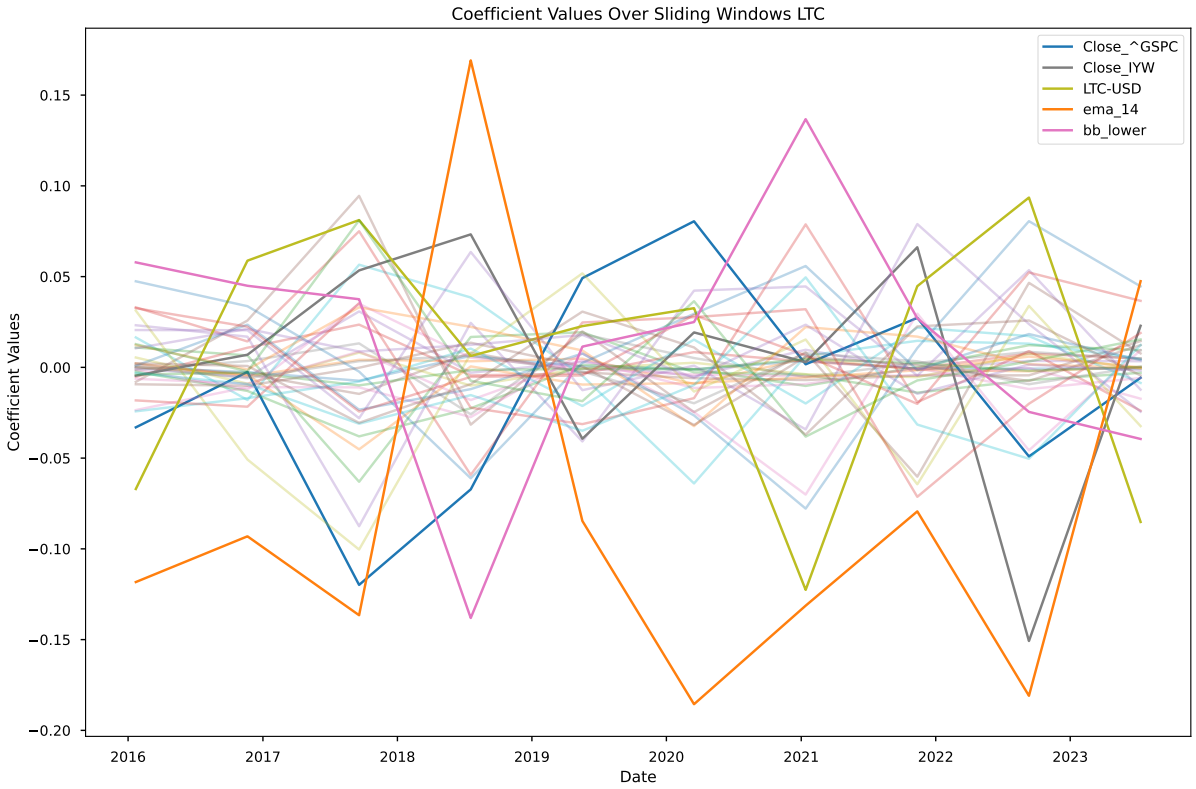
Source: Author

Figure B.10: Learned coefficients of the Ridge regression model with incremental training on the LTC dataset. Five coefficients with highest variance are highlighted.



Source: Author

Figure B.11: Learned coefficients of the Ridge regression model with sliding window training on the LTC dataset. Five coefficients with highest variance are highlighted.



Source: Author

Appendix C

Variables Description

Following is the list of fundamental variables on the example of BTC. The definitions were taken directly from **coinmetrics.io** to avoid any misconceptions:

Table C.1: Fundamental Variables Descriptions

Variable	**Description**
`BTC / Addresses, active, count`	The sum count of unique addresses that were active in the network .
`BTC / NVT, adjusted`	The ratio of the network value (or market capitalization, current supply) divided by the adjusted transfer value.
`BTC / Flow, in, to exchanges, USD`	The sum USD value sent to exchanges that interval, excluding exchange to exchange activity.
`BTC / Flow, out, from exchanges, USD`	The sum USD value withdrawn from exchanges that interval, excluding exchange to exchange activity.
`BTC / Fees, transaction, mean, USD`	The USD value of the mean fee per transaction that interval.
`BTC / Miner revenue, USD`	The USD value of the mean miner reward per estimated hash unit performed during the period, also known as hashprice.
`BTC / Capitalization, realized, USD`	The sum USD value based on the USD closing price on the day that a native unit last moved for all native units.
`BTC / Volatility, daily returns, 30d`	The 30D volatility, measured as the standard deviation of the natural log of daily returns over the past 30 days.
`BTC / Hash rate, mean`	The mean rate at which miners are solving hashes that interval.
`BTC / Transactions, count`	The sum count of transactions that interval.

Appendix D

Aditional Contents

All of the source codes and data to reproduce the results are available at: Noise-reduction-and-feature-extraction. Including all the instructions on how to install the necessary dependencies.