Non-Gaussian Time Series Modelling

Ramez Shawqi Qasim

26/09/2022

School of Mathematics,
Cardiff University

A dissertation submitted in partial fulfilment of the
requirements for **MSc (in Data Science and Analytics)**
supervised by Dr. Oktay Karakus

Executive Summary

The introduction of Futures into cryptocurrencies in recent years means that investors can leverage their money and act based on the behaviour of a coin at a certain point in the future, this increases the regulatory burden and uncovers the inner dynamics of the industry, presenting both challenges and opportunities that are worth exploring.

Two major gaps have been identified after reviewing the literature of cryptocurrencies forecasting. Firstly, the lack of studies addressing the efficiency of using deep learning models in its forecasting, and secondly, understanding the differences between the current price and the futures price from a modelling perspective.

This dissertation explores the use of several timeseries models, including ARIMA, XGBoost, Facebook Prophet, and LSTM. FBProphet is relatively new and is not commonly used in this sector. Long-Short-Term-Memory (LSTM) neural network architecture was chosen to represent deep learning models. ARIMA was used as an example of more traditional models, and XGBoost, as an example of ensemble models.

Corresponding to the gaps mentioned earlier, four models were developed, and the following two tables summarise the results:

RMSE Metric	ARIMA	FBProphet	XGBoost	LSTM
BTCUSDT Futures	1240	2147.36	2354.27	1129
BTCUSDT	1352	2408.6	1921.218	811
DOGEUSDT Futures	.00477	.0079	.0412	.0048
DOGEUSDT	.00507	.0108	.0117	.004

MAE Metric	ARIMA	FBProphet	XGBoost	LSTM
BTCUSDT Futures	863.14	2492.52	1915.36	961.6
BTCUSDT	904.3	2822.56	1510.51	608
DOGEUSDT Futures	.00355	.0093	.0396	.0038
DOGEUSDT	.00388	.0118	.00937	.0035

The accuracy of forecasts is measured by two metrics: RMSE and MAE. They are both useful and can reflect unique insights regarding the model's performance; MAE reflects the absolute difference between actual and forecast, whereas RMSE penalises larger gaps while minimising the impact of smaller forecasting gaps.

For non-gaussian data, especially data with high volatility, looking at the MAE gives a more balanced assessment of the performance of the model, since it will not penalise residuals of high jumps like the RMSE. However, if the focus is on the model's ability to capture sudden jumps and volatility, then RMSE provides a more reliable view.

The RMSE and MAE tables can be interpreted from three angles:

1- RMSE vs MAE, which means focussing on capturing jumps and volatility (RMSE) vs focussing on the overall performance of the model (MAE)
2- Established coins (Bitcoin) vs Niche coins (Dogecoin)
3- Regular price vs Futures price

Based on these angles, we can observe the following:

- Overall, LSTM and ARIMA are performing the best on both metrics, and XGBoost and FBProphet performed worse consistently.
- Based on the RMSE, LSTM performed similar or better than ARIMA for all four timeseries data, which suggests that LSTM is the best at capturing jumps and volatilities quickly and minimise the gap between the forecast and the residual.
- Looking again at RMSE, LSTM forecast for the regular closing price is notably more accurate than the futures closing price, while the opposite is true for ARIMA.
- Based on the MAE, the same pattern on the regular vs futures prices persists: LSTM is performing better on the regular price than on the futures, while the opposite is true for ARIMA.
- As we have seen while visualising and analysing the data, the charts for a coin and its futures are very similar with very similar trends, therefore this consistency of LSTM to perform better on the regular price needs an extended study to discover the subtle differences between a coin and its futures that lead to this pattern in modelling performance.
- It is worth noting that deep learning models are almost non-existent currently in cryptocurrency futures literature, the comparative performance presented in this work highly recommends utilising and exploring different forms of Artificial Neural Networks, especially with the recent rise in transformers, potentially replacing LSTMs.
- Although LSTM mostly scored the best accuracy, that does not mean that other more classical models should be neglected. Prophet model, despite being the least accurate,

has provided important insights and analysis such as change points in trends, it also allows developers to specify certain heuristics for the model, these advantages are not easily replicated in classical ARIMA and would require more time and expertise. So, the models should be utilised depending on who and why are they needed, if only for accuracy, then neural networks are the most efficient, if analysis is equally or more important than other models are worth exploring.

- All the models have performed better on well-established coins (Bitcoin) than other Niche coins (DOGECOIN) based on the actual against forecast plots. For this conclusion to be confirmed, the study should be extended to focus on classifying the coins based on a specific set of measures (such as the project it is trying to implement, the team developing the coin, the whitepaper of the coin, the speed of transactions, etc..), then forecasting each coin and comparing the accuracy to the rank of the model.

The other gap this work is addressing is relating the regular price of the coin to its futures price, the following was concluded based on results:

- The high-level analysis of the data itself did not show any noticeable discrepancies between the regular and the futures price, or at least no noticeable trends, this could explain the scarcity of research on futures.
- Nevertheless, the futures prices are subtly different from the regular as confirmed by LSTM's performance, and if we compare cryptocurrencies to other commodities such as stocks, a lot of research has been done to relate the actual prices and the futures prices, which this both impacts and is impacted by economic stability.

An extended work to address this gap would be to dive deeper into the financial side of cryptocurrency futures and analyse different maturities for the same coin, while simultaneously comparing them to the regular price, this ensures that results could capture the subtle insights from futures.

Acknowledgements

I would like to thank my project supervisor Dr. Karakus, who has diligently guided me throughout this dissertation, his insights and guidance have been instrumental in bringing this project to fruition.

I would also like to express my gratitude to Gavin Powell and Emanuele Ragnoli from QPQ Enterprise, their expertise in the field of digital currencies has been invaluable to this project.

And finally, I extend my gratitude to my friends and family.

Contents

Table of Figures

1 Introduction

Cryptocurrency investing is a relatively new field in today's economy with ground breaking innovations such as decentralised authority and anonymity investors. It has been steadily gaining ground and increasing in popularity, especially among amateur and young investors, this growth has pushed governments and policymakers to introduce and try to enforce regulations on the industry, while others refuse to recognise it as a legal tender, like the United States, and some like El Salvador have adopted Bitcoin as a currency.

In November 2008, an anonymous individual or group using the alias Satoshi Nakamoto released the Bitcoin systematic structural specification (Nakamoto 2008). Since then, Bitcoin has remained the most popular and most valuable cryptocurrency in the world, despite the launch of hundreds of other cryptocurrencies and several price variations. At the time of writing, Bitcoin's market capitalization exceeds 400 billion dollars. Moreover, the total market capitalisation of all active cryptocurrencies, including Bitcoin, exceeds 1.1 trillion dollars (CoinMarketCap. 2022).

Bitcoin and other cryptocurrencies have progressively grabbed the focus of investors worldwide. Currently, there are over 2000 types of currencies on the cryptocurrency market, for which the number of transactions and market capitalization vary. Considering the higher investment risk associated with cryptocurrency compared to other items, and that circulation is enormous yet highly volatile, predicting the price volatility trend of cryptocurrencies is significant.

In recent years, machine learning and its linked domains have made significant strides (Jordan and Mitchell 2015). Some of these technical advances have resulted in the development or enhancement of items that are used by billions of individuals globally (Meta 2022). Since the introduction of machine learning research and associated technologies, several researchers have focused their efforts on applying these new approaches to the financial markets. Predictions of the stock market (Kimoto et al. 1990) and the detection of manipulation (Zhai et al. 2018) are two examples of extensive studies on this topic. Many users believe cryptocurrencies to be financial assets, and a substantial amount of research conducted on financial markets may also be applied to this subject.

Since the discovery of Blockchain technology around a decade ago, the majority of published research in this field has focused on non-technical elements of Blockchain technology,

including legal difficulties and its involvement in criminal activity (Holub and Johnson 2018). Due to the novelty of Blockchain technology and the fast development of machine learning methods, research on their combination is still immature and expansive in comparison to many other study fields.

In this dissertation, we review the existing works and research in cryptocurrency machine learning modelling, identify the gaps and edges of existing work, and methodically try to answer relevant questions.

1.1 Motivation

Cryptocurrencies have introduced many innovations in the financial industry and are consistently using refined technology and algorithms to push the narrative of a decentralised economy, therefore, they have garnered the attention of amateur and professional investors, governments and legislative bodies. This highlights the importance of understanding and modelling cryptocurrencies, not just for investing, but also for legislative and regulatory purposes.

The introduction of Futures into cryptocurrencies in recent years, means that investors can leverage their money and try to predict the behaviour of a coin at a certain point in the future, this increases the regulatory burden and uncovers the inner dynamics of the industry, presenting both challenges and opportunities that are worth exploring.

1.2 Problem Definition

After going through the literature on cryptocurrency prediction there were two major gaps recognised: scarce research done on cryptocurrency **futures** prediction, particularly with deep learning models, and a lack of relating the Futures as a commodity to the regular coin's price.

1.3 Applied Methodology

Develop a range of regression models covering both classical and deep learning algorithms and compare their performance using the actual price and the futures price of a coin, therefore answering if the use if deep learning models provide a more holistic and accurate results and therefore should be prioritised.

Since we are also attempting to understand the behaviour of this volatile market and capture the non-gaussian components, the models will be fit to a more stable coin (like Bitcoin), and a

highly volatile coin with no actual goal or value supporting it (like Dogecoin). The results will be compared for both coins and similarities and differences will be noted.

2 Background and literature review

The following titles establish the background and the main concepts and terms used in this work, as well as mention and detail previous works and conclusions existing in the literature.

2.1 Digital Currency

Digital currency is a type of electronic money that is exclusively accessible in digital form. It mimics genuine cash, yet transactions and ownership transfers may be completed quickly. Virtual currencies, cryptocurrencies, and central bank issued money with digital base are all examples of that.

Similar to fiat money, these currencies may be used to obtain tangible items and services. However, it may also be confined to limited groups, like social networks or online gaming use (Coindesk 2022). Digital currency may be either centralised, with a single point of control, or decentralised, with several parties exercising control over the money transactions.

2.2 Cryptocurrency

Cryptocurrency is a sort of digital currency meant to be safe and, in many cases, untraceable. It is an internet-based money that employs encryption, the process of transforming readable data into a nearly impenetrable code.

Cryptography arose from the necessity for secure communication. communication during World War II. It has adapted to the digital age by Utilizing mathematical theory and computer science as a means of securing online communications, information, and money.

Using decentralised technology, cryptocurrencies enable users to conduct safe payments and store funds without using their identity or going via a bank. They operate on a distributed technology known as blockchain, which is a log of all money transfers maintained by the owners themselves.

Applications of cryptocurrencies and blockchain technology are considered in their infancy from a financial perspective, and growth should be anticipated including transactions involving ETFs, fiat, and other financial holdings which might be exchanged utilising the technology.

2.3 Blockchain

A blockchain is a sort of Digital Ledger Technology (DLT) consisting of a growing list of documents, known as blocks, that are cryptographically linked together. (Morris 2016) (Popper 2016). Each block includes the preceding block's cryptographic hash, a timestamp, and

transaction data (generally represented as a Merkle tree, where data nodes are represented by leafs). The timestamp demonstrates that the transaction data existed at the moment of the block's creation. Due to the fact that each block carries information about the block before it, they essentially form a chain (compare linked list data structure), with each block connecting to the ones that came before it. As a result, blockchain transactions are irreversible in the sense that once they are recorded, the data in a specific block cannot be modified retrospectively without affecting all following blocks.

Blockchains are often administered by a peer-to-peer (P2P) computer network for use as a public distributed ledger, with nodes adhering to a consensus method protocol to add and verify new transaction blocks. Despite the fact that blockchain records are not immutable, but blockchain forks are achievable, blockchains may be regarded as safe by design and are an example of a distributed computing system with high Byzantine fault tolerance, which means one component of a system can fail without stopping the whole system. (Marco Iansiti and Lakhani 2017)

2.4 Cryptocurrency exchange

Cryptocurrency exchanges are online platforms that facilitate the trading of one cryptocurrency for another. There are classical cryptocurrency exchanges, brokers, and exchanging platforms.

In classical cryptocurrency exchanges, buyers and sellers trade based on the current bitcoin market price, similar to traditional stock exchanges. In general, these trading platforms impose transaction fees. Some of these exchanges solely handle cryptocurrencies, while others allow customers to trade fiat currencies like the GBP for cryptocurrencies like Ethereum. (CFI 2022).

Brokers are exchanges based in a website, that permit clients to purchase and sell cryptocurrencies at a price determined by the exchange itself. This is the most straightforward approach for new users, although some may pay somewhat costs more than on the exchanges. (CFI 2022).

2.5 Cryptocurrency Futures

The concept of futures contracts originated in Japan in the seventeenth century, where farmers and buyers wanted to secure the price of rice at some point in the future, regardless of anything that could impact rice production or the demand on rice, this enabled them to plan and invest their money more securely. The concept of futures contracts is being consistently used in today’s economy in various stocks and commodities. (Osaka-Exchange 2014)

For cryptocurrencies, Futures contracts are speculations between two investors on the future price of a cryptocurrency. They enable investors to obtain exposure to certain cryptocurrencies without having to purchase them. Crypto futures mimic traditional futures contracts for commodities or equities in that they permit speculations on the price trajectory of an underlying asset (BAJPAI and RASURE 2022). Investors must adhere to the contracts' stipulated unit count, price, margin requirements, and settlement processes decided by the cryptocurrency exchange offering the contract such as Chicago Mercantile Exchange (CME) (CME 2022) .

Depending on Bitcoin's volatility, the investor may either keep or sell the futures contracts. The investor has the choice to either roll over their contracts into new ones or let them expire and receive the cash settlement due.

2.6 Time Series

Time series can be simply referred to as time-ordered sequence of observations, there are many examples of time-series data including rainfall measurements, stock prices, annual retail sales, monthly subscribers, and so on.

A time series' defining trait is that its observations are correlated. The vast majority of typical statistical techniques based on random samples are not suitable for data over time, necessitating the use of alternative approaches. The statistical techniques used for examining time series are known as time series analysis techniques. Some of these are descriptive, with an emphasis on the non-stochastic description of time series. The stochastic method treats a time series as a representation of a stochastic time series process or model. The primary goal of studying time series using the stochastic approach is to develop a plausible underlying process and utilise it for forecasting, inference, and control, and this will be the primary focus of this dissertation.

2.6.1 Components of time series analysis

Time series has three main components; Trend, which is the general direction of data within a defined interval in a continuous timeline, it could be either increasing, decreasing, or null. Seasonality is when a regular or constant interval varies within a dataset across a continuous timeframe, it would either be bell curve or saw tooth. Residuals represent the remaining component after removing trend and seasonality.(Wei 2013)

There are also cyclical and irregularity components, where irregularity represents situations /scenarios and peaks that occur unexpectedly and within a brief time frame, and cyclical is characterised by the absence of a defined interval, unpredictability in movement and pattern.

2.6.2 Stationarity in time series

As a simple definition, stationarity indicates that the statistical features of a process that generates a time series do not vary over time. It does not indicate that the series does not change over time; rather that the manner in which it changes does not change.

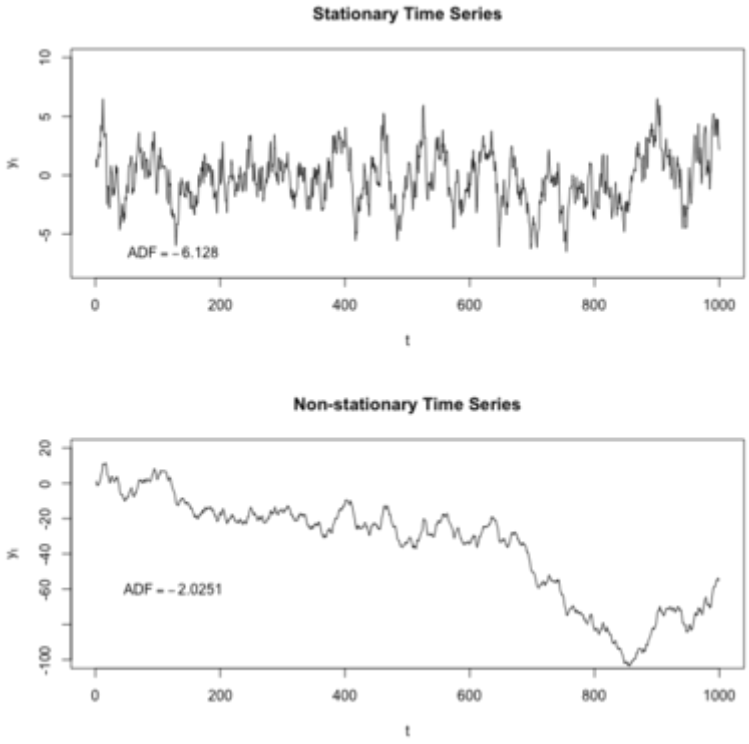

Figure 1. Time series generated by a stationary (top) and a non-stationary (bottom) processes

A time series represents a stochastic process (a time-indexed random variable). The combined probability distribution of these variables defines a time series process denoted by Z. The process is said to be strictly stationary if its joint distribution is invariant with respect to a changing time origin. Therefore, the following describes a strictly stationary process:

$$F_{Z_{t1..ztn}(X_1 \ldots X_n)} = F_{Z_{t1+k \ldots tn+k}(X_1 \ldots X_n)}$$

Since it's very difficult to check that a time series process follows this, the concepts of weakly stationary and strongly stationary are used, where in weakly stationary it's sufficient to know the first few moments of the time series process. (A. Witt and Pikovsky 1998)

2.6.3 Stationarity testing

The most elementary approaches for detecting stationarity include visualising the data and visually examining it for trend and seasonal components.

Examining the plot of a time series to establish if it was formed by a stationary process, is a problematic endeavour since it is subjective and not always reliable. Nonetheless, there are certain fundamental characteristics of non-stationary data that can be observed in a plot.

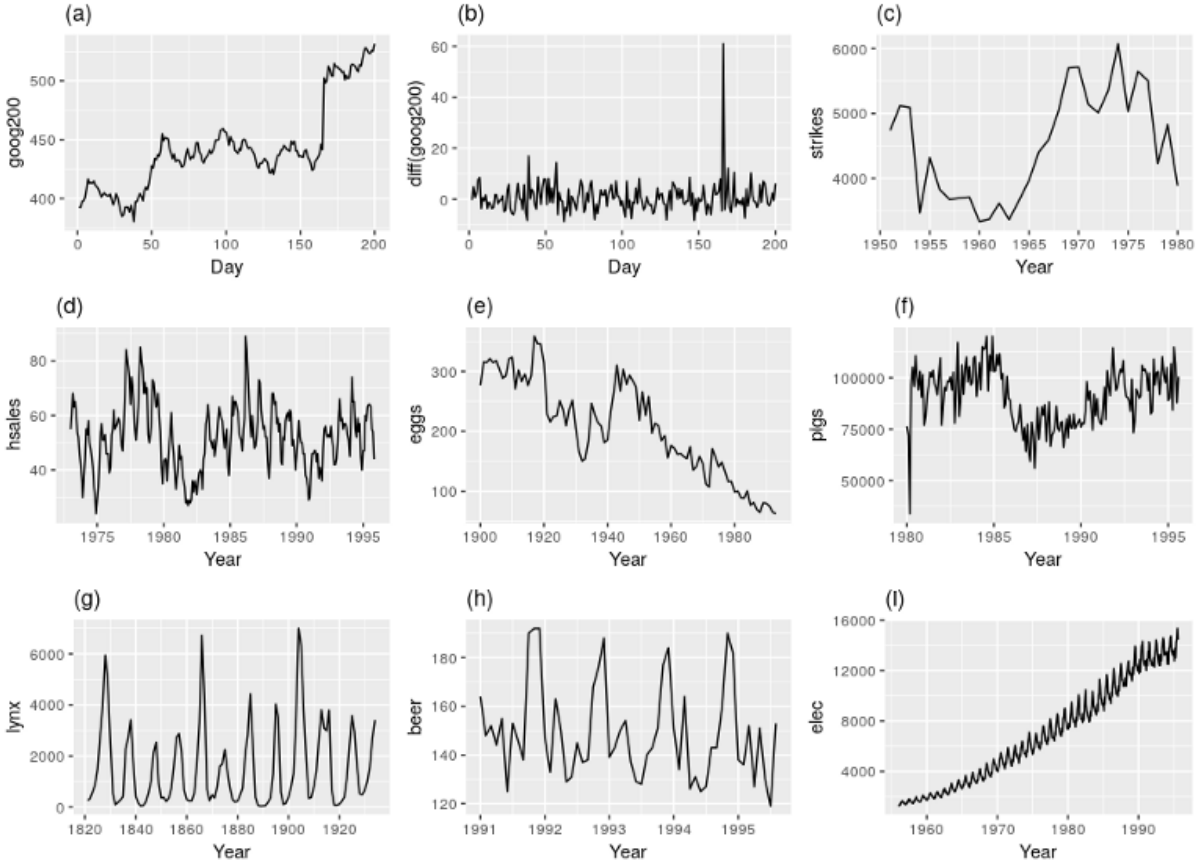

Figure 2. Nine examples of time series data (Hyndman and Athanasopoulos 2018)

Figure 2 shows nine examples of time series data; (a) Google stock price for 200 consecutive days; (b) Daily change in the Google stock price for 200 consecutive days; (c) Annual number of strikes in the US; (d) Monthly sales of new one-family houses sold in the US. Seasonality can be observed in series (d), (h), and (i). The trend can be observed in series (a), (c), (e), (f), and (i), and series (b) and (g) are stationary.

There are many statistical tests for stationarity of time series data, most notably are the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test.

Assumptions that are made regarding the data can only be used to determine the extent to which a null hypothesis may be rejected or not rejected. However, the conclusion must be analysed

before determining the significance. Nonetheless, these tests serve as a rapid confirmation if the time series is stationary or non-stationary.

The Augmented Dickey-Fuller test is a unit root test, a sort of statistical test, which in probability theory is a characteristic of some stochastic processes that may hinder statistical inference utilising time series models. The unit root is non-stationary, though it does not necessarily include a trend component.

ADF testing is undertaken based on the following hypotheses: Neither the series is stationary nor the series has a unit root; Alternate Hypothesis (AH): Either the series is stationary or there is no unit root. This test may offer evidence that the series is non-stationary if the null hypothesis is not rejected.

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) is an abbreviation for the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, which is a form of Unit root test that examines the stationarity of a given series around a deterministic trend. In other words, the test is conceptually comparable to the ADF test. Nevertheless, it is a popular misperception that it may be used interchangeably with the ADF test. This might lead to misinterpretations regarding the stationarity, which can easily go unnoticed and cause more issues in the future.

2.6.4 Autocorrelation Function (ACF)

Autocorrelation can be defined as the similarity between observations as a function of the time lag between them. The covariance function at lag k is sometimes referred to as the autocovariance function at lag k since it reflects the covariance between and from the same process, with k time delays between them. The ACF between Z_t and Z_{t+ki} is hence the standardised autocovariance function,

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{Cov(Z_t, Z_{t+k})}{\sqrt{Var(Z_{t)} * Var(Z_{t+k})}}$$

Noting that for a stationary process, $\rho_0 = 1$, $|\rho_k| \leq 1$, and $\rho - k = \rho_k$. The plot of the ACF for non-negative lags is called correlogram.

2.6.5 Partial Autocorrelation Function

This is the examination of the correlation between Z_t and Z_{t+k} after removing their mutual linear dependency on the intervening variables Z_{t+1} ,…, and Z_{t+k-1},

$$\varphi_{kk} = Corr(Z_t, Z_{t+k}, Z_{t+k-1})$$

2.6.6 Seasonality

Seasonality describes periodic changes. For instance, tourism could be much more vibrant in summer months than in winter months, or traffic declines during the holiday season before rising again.

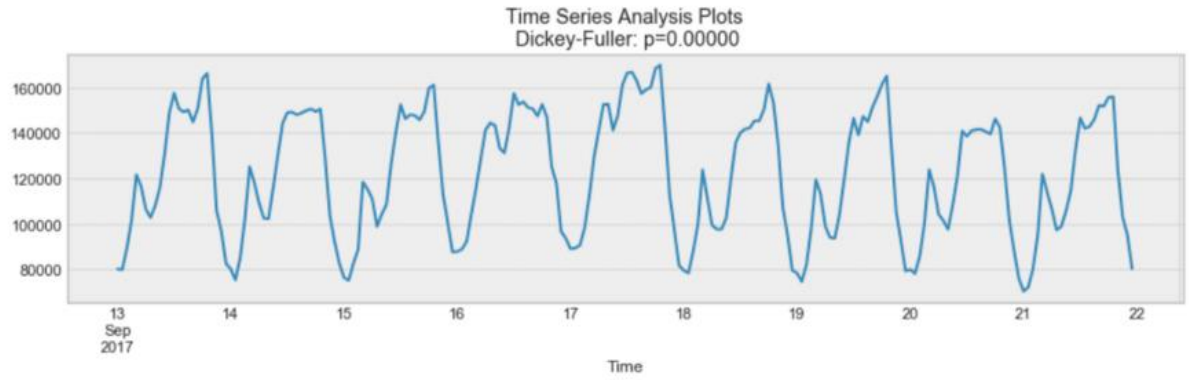

Figure 3. Example of seasonality in a time series data

In Figure 3, there is a distinct daily seasonality. Each day reaches its highest point in the evening, and its lowest points at the morning and conclusion of each day.

Seasonality may also be determined from an autocorrelation plot if the form of the plot is sinusoidal.

2.6.7 Gaussian Processes and white noise

White noise is a succession of uncorrelated random variables from a fixed distribution with a constant mean, often considered to be zero, and constant variance. This, the white noise has the ACF

$$\rho_k = \begin{cases} 1, k = 0 \\ 0, k \neq 0 \end{cases}$$

And the PACF

$$\varphi_{kk} = \begin{cases} 1, k = 0 \\ 0, k \neq 0 \end{cases}$$

It is considered that a time series process is Gaussian or normal if its joint distribution is normal. In time series analysis, the majority of findings obtained are Gaussian processes, due to the fact that a normal distribution is described exclusively by its first two moments(Wei 2013), strictly stationary and weakly stationary are equal for a Gaussian process. Consequently, the mean, the

variance, the ACF, and the PACF become important measurements for identifying time series models.

2.6.8 Non-Gaussian time series

The nature of time series data might be Gaussian or non-Gaussian, numerous models have been suggested for modelling non-Gaussian time series data. According to (Bishop C. M. 2006), the Gaussian distribution is a prominent probability distribution in statistics due to its well-defined probability density function and the ease with which analysis may be performed. However, not all modelled data have Gaussian distributions in nature (Banfieldand and Raftery 2004).

While the Gaussian process has significant modelling benefits, the Gaussian distribution assumption is unreliable for really non-Gaussian time series data (Xu 2014). This challenge may be overcome by performing an elementary transformation on the non-Gaussian data and assuming that the altered data are observations from a Gaussian process (Sun et al. 2008). Nonetheless, this method has drawbacks since a simple change would merely reduce the skewness and/or kurtosis without changing the marginal distribution to a Gaussian shape (Xu 2014). Defining the joint distribution of the non-Gaussian process explicitly is an alternative to the transformation approach; however, unlike Gaussian process models, the specification of a non-Gaussian process's joint distribution extends considerably beyond specifying a mean vector and covariance matrix. Consequently, the objective of this work is to test the performance of various ML models in efficiently capturing the non-Gaussian distribution.

For handling the issue of non-Gaussian time series data, many strategies have been presented and accepted in the literature, such as (Blei 2004; A. Leijon and Kleijn 2013; Bishop et al. 2020), the transformation method, the transition method, and Generalized Linear Mixed Models (GLMMs). However, these strategies each have their own disadvantages.

Bishop compared Gaussian and non-Gaussian distributions in time series analysis and found that the Gaussian distribution can be simply applied to data since its likelihood is non-tractable and its analysis can be obtained explicitly (Bishop C. M. 2006).

Nevertheless, (Banfieldand and Raftery 2004) noted that not every data could always be modelled using Gaussian distribution due to the underlying characteristics of such datasets. Some data have semi-limited or bounded support, but the Gaussian distribution has unbounded support. Non-Gaussian statistical models would be preferred for modelling data that is not normally distributed, according to studies (Blei 2004) and (Fitzmaurice et al. 2008).

Techniques for modelling non-Gaussian time series data in the literature fall into four basic types (Oguntunde et al. 2018). Empirical transformations, such as the log transformation, the square root transformation, and a number of others, are the most used approach. The log transformation is a typical application of the Box-Cox transformation function (Stoica and Moses 1997). Rasmussen (Rasmussen 2006) performed a comprehensive study of the Box-Cox transformation and expanded the Box-Cox transformation using the MLE approach to accommodate responses with missing data. Lipsitz, Ibrahim, and Molenberghs (Lipsitz et al. 2000) expanded the Box-Cox transformation to the linear mixed model and discussed the effect of parameter selection on estimated coefficients and associated standard errors. They noted that an adjustment is required to address the bias in the variance of the estimated coefficients that is caused by the estimated transformation parameter. Gurka, Edwards, Muller, Kupper, and Lawrence (Gurka et al. 2006) introduced univariate and multivariate time series models for non-Gaussian marginal distribution processes. These models include bivariate autoregressive models for bivariate exponential marginal marginal processes. For several of the models mentioned, examples of their applicability to actual data sets were provided. When appropriate, the notion of positive dependency is employed to establish the relationship between the processes.

2.7 Feature selection and extraction

Feature selection is the process of eliminating features with a minimal contribution to the models' performance, or in some cases may negatively impact the performance. This is usually done to reduce computational load and time, which makes a considerable difference if the models are hosted on a website and are used live for other applications or other users, it's also useful for developers with limited time and computational power, as it enables them to do experiments and run models without spending as much time.

Testing this concept in this research is primarily inspired by this paper (Sun et al. 2020) in which the authors developed a cryptocurrency price trend forecasting model, they argued that existing research considers that Bitcoin price volatility is correlated to some macroeconomic variables such as stock index and oil price, therefore they chose 40 macroeconomic indicators as input features, and concluded their work by stating that in the future they will select more potential influencing factors and train deep learning models, which are both used in this research.

Feature selection methods can broadly be classified into supervised and unsupervised (Kjell Johnson 2013a), where the former refers to selecting features by statistically analysing the relationship between features and targets, and the latter does not take target features into account when performing the selection. Selection can also be divided into wrapper methods and filter methods (Kjell Johnson 2013c), wrapper methods run models with different subsets of input features, and then determines which input features contribute the most to the performance of the model, filter methods on the other hand use statistical techniques to evaluate the relationships between variables and filter out redundant and non-informative features (Kjell Johnson 2013c).

The aforementioned types perform feature selection independently from training the model, however, some models intrinsically perform feature selection while training the models (Kjell Johnson 2013b), most notably are decision trees (Leo Breiman, Jerome H. Friedman, Richard A. Olshen 1984) which show the importance of each feature after training.

One of the most common wrapper methods is the recursive feature elimination (RFE), where features are selected recursively by training models on smaller number of features each time (Isabelle Guyon, Jason Weston, Stephen Barnhill 2002).

2.8 Machine learning models

There are many ways to classify time series modelling, one of which are parametric and nonparametric models (CHEN et al. 1997). Numerous time series prediction techniques initially were parametric ones whose objective was to fit time series data to a statistical model. With the uncovering of significant limits in basic parametric techniques, researchers became more interested in nonparametric approaches that make no structural assumptions about the underlying process structure (Meir 2000).

In instances when laborious data analysis is inapplicable and the links between multiple entities are obscure, artificial intelligence has become an ideal choice for predicting time series. Machine learning is the subfield of Artificial Intelligence in which algorithms are created to enable algorithms to better themselves.

2.8.1 Auto Regression (AR)

In a time series, the previous values usually have an impact on the following data points, and this is particularly crucial for time series data in the financial sector. Thus, auto-regression

(AR) is the forecast of a later point based on the prior values of the same data point (Steel 2014) and is defined as:

$$y_t = \alpha_0 + \alpha_1 y_{t-1} + \cdots + \alpha_p y_{t-p} + \epsilon$$

Where α_0 is the intercept, a_1 - a_p are the coefficients of model, y_{t-1} - y_{t-p} are the previous values, and ϵ represents noise in the data.

The auto regressive model is often expressed as AR (p), which indicates that current values are dependent on their own (p) prior values and defines the AR model's order.

2.8.2 Moving Average (MA)

(MA) models suggest that the present divergence from the mean depends on past discrepancies. The moving average (MA) model's equation can be expressed as:

$$m_t = \beta_0 + \beta_1 m_{t-1} + \cdots + \beta_p m_{t-p} + \epsilon$$

Where β_0 is the intercept, β_1 to β_p are the coefficients of the model, $m_{t-1\ldots t-p}$ are the previous discrepancies, and ϵ represents the residuals.

2.8.3 Auto Regressive Moving Average (ARMA)

The auto regressive moving average (ARMA) model, commonly known as the Box-Jenkins (Krollner et al. 2010) method, is a combination of moving average and auto regression. ARMA combines the moving average and auto-regressive processes to create a model that employs both concepts:

$$y_t = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p} - \beta_1 m_{t-1} - \beta_2 m_{t-2} - \ldots - \beta_p m_{t-p} + \epsilon$$

Where c is the intercept and is usually the average of the time series, $\alpha_1 - \alpha_p$ are the coefficients of the auto-regression process, $\beta_1 - \beta_p$ are the coefficients of the moving average process, and $y_{t-1} - y_{t-p}$ are the previous values. $m_{t-1} - m_{t-p}$ are the previous deviations, and ϵ is the residual component.

In accordance with the practise suggested by Box and Jenkins, the MA parameters are specified here using negative signs in the formula. Some authors and applications specify them using plus signs instead.

An ARMA(p,q) model employs the prior p values in the auto-regression component and the moving averages produced from the most recent q values. Therefore, the development of an ARMA model involves three phases (Steel 2014):

1. Identification process that determines the order of AR and MA components
2. Coefficients estimation
3. Forecasting

The necessity that the time series be linear and stationary with no trend, as well as the difficulty in determining the precise parameters to employ in the model, are inborn aspects of ARMA models that might be considered disadvantages.

To circumvent these limitations, researchers have experimented with numerous methods for improving the efficacy of ARMA models (Steel 2014).

2.8.4 Auto-Regressive Integrated Moving-Average (ARIMA)

A disadvantage of the ARMA model, as well as other techniques, is the assumption that the time series is stationary, has no trend, and has a constant variance and mean (Shumway and Stoffer 2011). In practice, many time series data sets have a pattern, and this is true for financial data as well. To express trend in a time series, it is typically transformed into a stationary data set, which is then subjected to modelling before being transformed back to its initial condition.

In practise, the trend component is removed, modelling is performed, and then the trend component is reintroduced into the data. Differencing is one method for reducing trend component (Mills 2011). Differentiation is the process of substituting the actual values of the data with the values of their differences. This is shown as:

$$\Delta_t^{(1)} = y_t - y_{t-1}$$

Since differencing is equivalent to evaluating the series' derivative, a time series that has undergone differencing is regarded as "integrated" (Steel 2014). If using this supposed first difference does not eliminate the trend, one might proceed and apply the second difference:

$$\Delta_t^{(2)} = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$$

The ARMA model is transformed into the auto-regressive integrated moving average (ARIMA) model via the integration step. ARIMA is capable of handling any series, seasonal or not. The ARIMA mathematical expression is:

$$y_t = c + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p} + +\beta_1 m_{t-1} + \beta_2 m_{t-2} + \ldots + \beta_p m_{t-p} + d_1 \Delta_{(1)}\mathrm{t}{-}1 + \ldots + d_d + \Delta^{(\mathrm{d})}{}_{(\mathrm{t\text{-}d})} + \epsilon$$

- c is the intercept.
- $\alpha_1 - \alpha_p$ are the coefficients of the auto-regression component.
- $\beta_1 - \beta_p$ are the coefficients of the moving average component.
- $d_1 - d_p$ are the coefficients of the differencing term.
- $y_{t-1} - y_{t-p}$ are the previous values.
- $m_{t-1} - m_{t-p}$ are the previous deviations.
- ϵ is the random noise.

ARIMA models are commonly denoted as ARIMA(p,d,q), where p is the number of terms used in the auto-regression phase, d is the number of differencing terms, and q is the number of terms used in the moving-average phase.

2.8.5 Facebook Prophet

FBProphet is a forecasting tool package developed by Facebbok (Meta) in 2017 to address two main problems in forecasting tools available for businesses(Taylor and Letham 2017):

- Existing automatic forecasting techniques are inflexible and cannot incorporate useful assumptions and heuristics.
- There is a shortage of analysts who can produce high quality forecasts, since it is a specialised skill requiring extensive experience.

Prophet, as stated by its developers, is not suitable for all types of forecasting, it is optimised for forecasts related to Facebook's line of business, the data typically has the following properties:

- Preferably at least a year of hourly or daily data

- Multiple seasonalities
- Historical trend changes, such as a product launch
- Data contains trends that tend to saturate and hit a limit after some time

Internally, Prophet is an additive regression model that has the follwoing main components: piecewise linear or logistic trend, yearly seasonal component based on Fourier series and weekly seasonal component.

The essence of the Prophet technique was built in Stan's probabilistic programming language, which was used to fit the Prophet model. Stan does the MAP parameter optimization incredibly fast (<1 second), providing developers the option to evaluate parameter uncertainty using the Hamiltonian Monte Carlo approach, and enables them to reuse the fitting process across numerous interface languages. Currently, Prophet is implemented in both Python and R. Both implementations have the exact same characteristics (Taylor and Letham 2017).

2.8.6 Ensemble Models – XGBoost

Extreme gradient boosting technique (XGBoost) is a scalable machine learning system for composed of successive trees. It is an improved distributed gradient boosting library that can rapidly assess the value of all input attributes. It has shown to be a consistent and effective problem solver for machine learning (Hu et al. 2020; Zheng et al. 2020). Compared to other gradient boosting methods, XGBoost is able to collect a strong classifier from a series of weak classifiers and exhibits the following benefits: (1) properly manage missing data; (2) avoid overfitting; (3) parallel and distributed computation reduces running time. The aim of XGBoost is to minimise the loss function by adding weak learners, i.e. to define and optimise the objective function using gradient descent optimization and arbitrary differentiable loss functions. XGBoost aims to reduce the regularised goal in the following manner:

$$\text{obj}(\theta) = \sum L(y_i, y_i) + \sum \Omega(f_k), \quad f_k \in \mathbf{F}$$

where L is the training loss function that measures the deviation between the value y predicted by our model and the actual value y. Ω is the regularization function that measures the complexity of the model, which tends to prevent overfitting. f is a function in the functional space $\mathbf{F}$, and $\mathbf{F}$ is the set of all possible regression trees. In order to minimize the objective function, XGBoost uses

parameters to find an optimal tree structure employing a greedy search algorithm (Luo et al. 2021).

2.8.7 Artificial Neural Networks

Artificial Neural Networks (ANN) are computer techniques attempting to imitate the behaviour of neuron-based biological systems. ANNs are computer models based on the central nervous systems of animals. It has both machine learning and pattern recognition capabilities. These are shown as networks of "neurons" capable of computing values from inputs.

A neural network is a directed network. It is composed of nodes, which, in the biological analogy, represent neurons, and arcs connecting them. It is equivalent to dendrites and synapses. Each arc has a weight associated with it at each node. Apply the input values received by the node and update the activation function along the incoming arcs based on their weights. Given an input or collection of inputs, the activation function of a node specifies the node's output.

A neural network is a method for machine learning based on a model of a human neuron. Millions of neurons make up the human brain. It transmits and processes messages using impulses with certain mechanisms. These neurons are joined by synapses, a unique structure. Synaptic connections enable neurons to transmit messages. Large quantities of active connected neurons constitute a neural network.

An Artificial Neural Network is a mechanism for processing information. It functions similarly to how the human brain handles data. A huge number of interconnected cells work synergistically to process data as part of an ANN. Additionally, they create significant outcomes from it. Neural networks may be used for more than just categorization. Also applicable to regression of continuous target characteristics.

The applicability of neural networks in data mining is extensive. Examples include economics, forensics, and pattern recognition. After extensive training, it may also be used to classify voluminous amounts of data.

Layers in more detail

Typically, an Artificial Neural Network is arranged in layers, which are composed of several linked "nodes" with an "activation function." A neural network may have the three layers listed below (Sharma 2017):

- Input layer: The goal of the input layer is to accept as input the values of each observation's explanatory characteristics. In most cases, the number of input nodes in an input layer corresponds to how many input variables are there. Patterns are presented by the input layer to the network, which then interacts with one or more hidden layers. Input layer nodes are passive, meaning they do not transmit data. Modify the data. They accept a single input value, which is then replicated over several outputs. From the input layer, each value is duplicated and delivered to all hidden nodes.
- The Hidden layers perform specified alterations to the network's input values. Connected to each node are incoming arcs which originate from previous hidden nodes or input nodes. It links to output nodes or the following processing nodes through outgoing arcs. In the buried layer, processing is accomplished using a network of connections with varying weights. There may be several concealed layers. The values entering a hidden node are multiplied by weights, a set of predetermined and stored integers. The inputs are then joined together to get a single number.
- Output Layer: The concealed layers are then linked to a layer's output. The output layer accepts connections from both hidden and input levels. It returns an output value that corresponds to the response variable's prediction. Typically, classification problems have a single output node. Active output layer nodes integrate and transform data to calculate final outputs. The capacity of an ANN to deliver effective data handling is contingent upon the correct update of weights. This is distinct from standard models processing.

Artificial Neural Networks are a prominent application for intelligent time series prediction (Krollner et al. 2010). Efforts to employ neural networks for forecasting time series have been made since the 1970s. However, given the limited understanding of neural networks at the time, the findings of the trials were not very encouraging. With the creation of the back-propagation algorithm in the eighties, however, research interest in ANNs grew fast and the challenge of timeseries forecasting was effectively applied. Several of these outcomes demonstrated that neural networks may beat statistical forecasting techniques including regression analysis and Box- Jenkins forecasting (Palit and Popovic 2006). With the growth of cryptocurrencies, several academics employ neural networks to forecast the price of Bitcoin, Ethereum, and the different altcoins that exist, as well as other stock indexes.

Recurrent Neural Networks

In the 1980s, recurrent neural networks (RNNs) were constructed by John Hopfield who created Hopfield networks. In 1933, a neural history compressor system accomplished a "Very Deep Learning" assignment requiring more than one thousand successive layers in an RNN unfurled over time (Schmidhuber 1993). A recurrent neural network (RNN) is a kind of artificial neural network in which unit connections form a guided loop. This permits it to capture temporally patterns in data. RNNs, which differs from feedforward neural networks (FFNN), may handle arbitrary input sequences using their internal memory.

A portion of the neural network seen in Figure 4 examines the input x_t and outputs the value h_t. A loop permits the transmission of data from one network step to the next. An RNN is comparable to several copies of the same network, with each copy sending a message to its successor (Olah 2015).

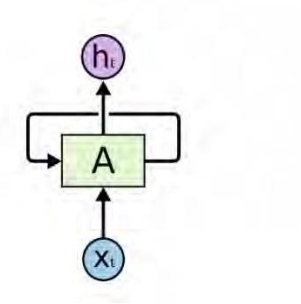

Figure 4. RNN-rolled (Olah 2015)

The cell in Figure 4 can be expanded as shown in Figure 5

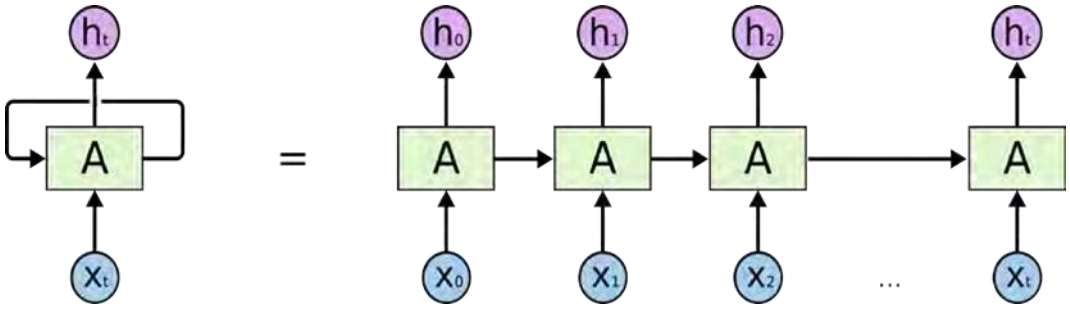

Figure 5. RNN Unrolled (Olah 2015)

This structure similar to a chain demonstrates that RNNs are closely associated data where there is a continuum, future data are a continuation of previous data. They are the optimal ANN design for this type of data. Every RNN consist of a series of neural network modules that

repeat. The most fundamental RNN cell is a neural network with a single layer, whose output serves as both the current output and the current state of the RNN cell as shown in Figure 6.

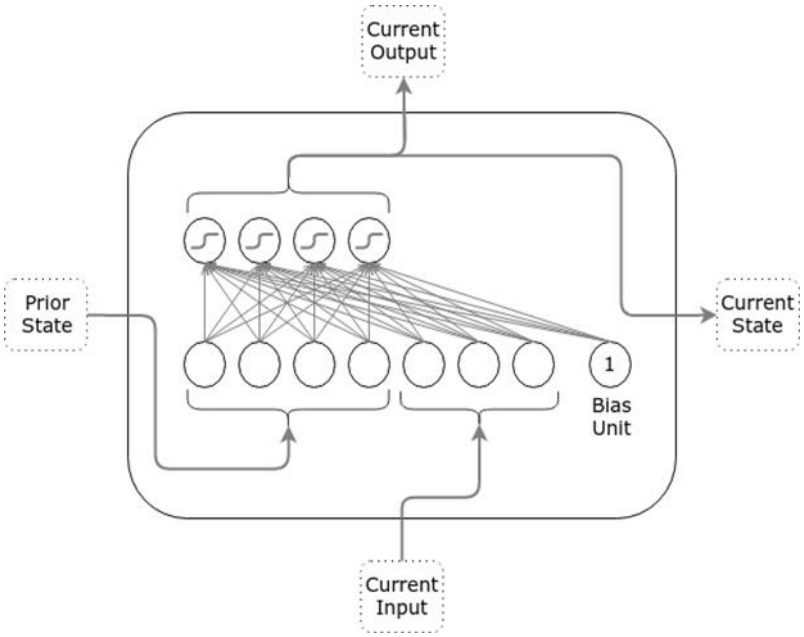

Figure 6. RNN cell internal architecture

Long Short Term Memory (LSTM)

Long Short Term Memory networks, often referred to as "LSTMs," are a specific kind of RNN that can learn long-term dependencies. They were presented by Hochreiter and Schmidhuber in 1997 [40], and subsequent authors developed and popularised them. They are currently frequently utilised and perform very well on a broad variety of challenges. LSTMs are purposefully intended to circumvent the issue of long-term reliance. It was intended to tackle the gradient issue caused by standard RNN. LSTM networks are capable of storing inputs in memory for a larger number of points in a sequence. It is not difficult for them to remember information for extended periods of time [38], since this is basically their normal habit.

In LSTMs, everything is "recorded," and the previous state is carried forward if there is no interference from other state units or external inputs. This implies that any changes to the state are gradual, thus

$$s_{t+1} = s_t + \Delta s_{t+1}$$

Figure 7 shows the internal structure of an LSTM cell.

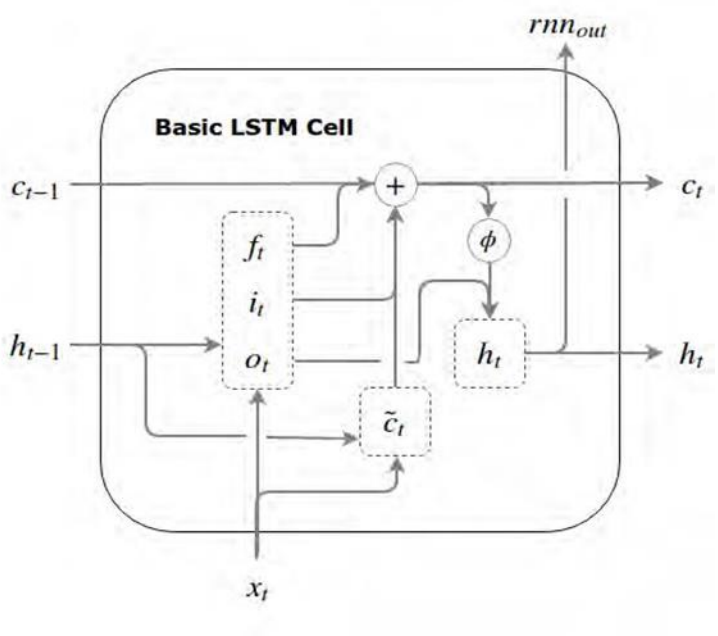

Figure 7. Internal structure of LSTM cells (R2rt 2016)

Mathematical formulation of the cell:(R2R2 2016)

$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$
$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$
$$f_t = \sigma(W_f s_{t-1} + U_f x_t + b_f)$$
$$\bar{c}_t = \varphi(W h_{t-1} + U x_t + b)$$
$$c_t = f_t \, 0 \, c_{t-1} + i_t \, 0 \, \bar{c}_t$$
$$h_t = o_t \, 0 \, \varphi(c_t)$$
$$rnn_{out} = h_t$$

Where the symbols mean the following:

- φ is the activation function.
- i_t, o_t, f_t are the gate functions.
- $s_{t-1} \in \mathbb{R}^n$ is the prior state
- c_t is cell or constant error.
- $\bar{c}_t$ is the candidate cell to write.
- h_t is the hidden state.

This is the fundamental structure of an LSTM recurrent neural network. However, not all LSTMs are identical to those described above. Almost every study using LSTMs seems to utilise a somewhat different variant. Minor differences exist (Olah 2015).

3 Methodology

In order to explore the impact of using deep neural networks in cryptocurrency futures prediction, as well as compare different modelling approaches in capture the non-gaussian traits in data, four different models were chosen to model four different time series data.

The first model is ARIMA, chosen to represent the more classic regression approaches as discussed in the literature review. Then FBProphet model is used, to extend the breadth of exploration of model performance and comparing the performance of the relatively new custom model developed by Meta company. Then, a boosting model with random trees is tested, and finally Long Short Term Memory (LSTM) neural network is also used.

On the data side, four different timeseries data were chosen, Bitcoin and Bitcoin futures, and Dogecoin and Dogecoin futures. Bitcoin is chosen because of its legacy and authority of its prices in the cryptocurrency world, as the market is still heavily influenced and reflected by bitcoin's prices. Dogecoin was chosen to represent the unstable coins which lack real projects backing them up, and are characterised by having rapid and severe fluctuations in its prices as it is easily influenced by speculations and social media posts.

The data for futures prices were collected from Binance exchange, with features like High, which is the highest price within a specific timeframe, Low, Open which is the opening price, and Close, which is the closing price for that currency in that timeframe. In this project all the modelling will be based on the Close price of a currency. Volume is the number of contracts or coins traded during a given period of time.

3.1 Tools and software used

3.1.1 Python Language

Python is a versatile and productive programming language utilised in wide range of applications. Since its inception, the wide base of developers using this open source programming language has created a diverse range of Python-specific tools. In previous years, a variety of data science-specific tools have been developed. Consequently, data analysis using Python has become more user friendly.

Anaconda is a free, open source implementation of the Python and R programming languages designed to facilitate package management and deployment for large-scale data processing, predictive analytics, and scientific computing. NumPy and SciPy, which are included in the

Anaconda installation package, were the most important Python libraries utilised (Inc. 2022). SciPy is a Python-based package used for scientific-related calculations and common functions that is free source. NumPy is another package that adds make it easier to handle massive, multidimensional arrays and matrices, as well as a many high level mathematical functions to apply on these arrays.

All the work was done in Python, using Jupyter notebooks that were hosted on Amazon's Sagemaker Studio Lab (Which uses Anaconda environment)

3.1.2 Keras, Tensorflow, and SKLearn

Keras is a high-level API for neural networks that is written in Python and is compatible with TensorFlow, CNTK, and Theano (Kuppusamy 2018). It was created with the goal of facilitating rapid experimentation. The key to doing effective research is being able to move swiftly from concept to conclusion.

Keras is a deep learning package that:

- Facilitates simple and rapid prototyping, because of its ease of use and flexibility.
- Recurrent Neural Networks, either separately or in combination with CNN is implemented through Keras.
- Can use CPU the traditional processing unit, or GPU which is faster.

TensorFlow is an open source software package for numerical calculations. The edges of the network reflect the multidimensional data arrays (tensors) transferred between the nodes. You may distribute compute to one or more CPUs or GPUs on a desktop, server, or mobile device using a single API thanks to the architecture's flexibility.

TensorFlow was created by researchers and engineers working on the Google Brain Team inside Google's Machine Intelligence research group for the purposes of performing machine learning and deep neural networks research, but the technology is also applicable to a broad range of other disciplines.

Keras using TensorFlow as a backend was utilised to generate LSTM neural networks for predictive modelling.

3.2 Data Analysis and Modelling

3.2.1 Overview

For Bitcoin/USD and Bitcoin/USD Futures, hourly pricing data from 19/September 2019 to 19/September/2022 with 26305 data points. Figure 8 shows a sample of the imported data.

Timestamp	Date	Time	Open	High	Low	Close	Volume
2019-09-19 00:00:00	19/09/2019	00:00:00	10154.70	10185.55	10145.00	10177.46	765.767
2019-09-19 01:00:00	19/09/2019	01:00:00	10178.18	10178.23	10121.40	10124.97	660.266
2019-09-19 02:00:00	19/09/2019	02:00:00	10127.66	10144.83	10081.56	10093.66	1361.359
2019-09-19 03:00:00	19/09/2019	03:00:00	10093.66	10094.44	9530.02	9821.97	1865.750
2019-09-19 04:00:00	19/09/2019	04:00:00	9820.96	9826.81	9780.75	9821.91	977.193

Figure 8. Sample of data imported for BTCUSDT Futures

For DOGE/USD Futures and DOGE/USD, hourly pricing data from 10/July/2020 to 19/September/2022 were imported, because DOGE futures were listed starting from this date and we cannot get previous data.

To get an overview of the differences between the four timeseries data, they were plotted over time as shown in plots Figure 9, Figure 10, Figure 11, and Figure 12.

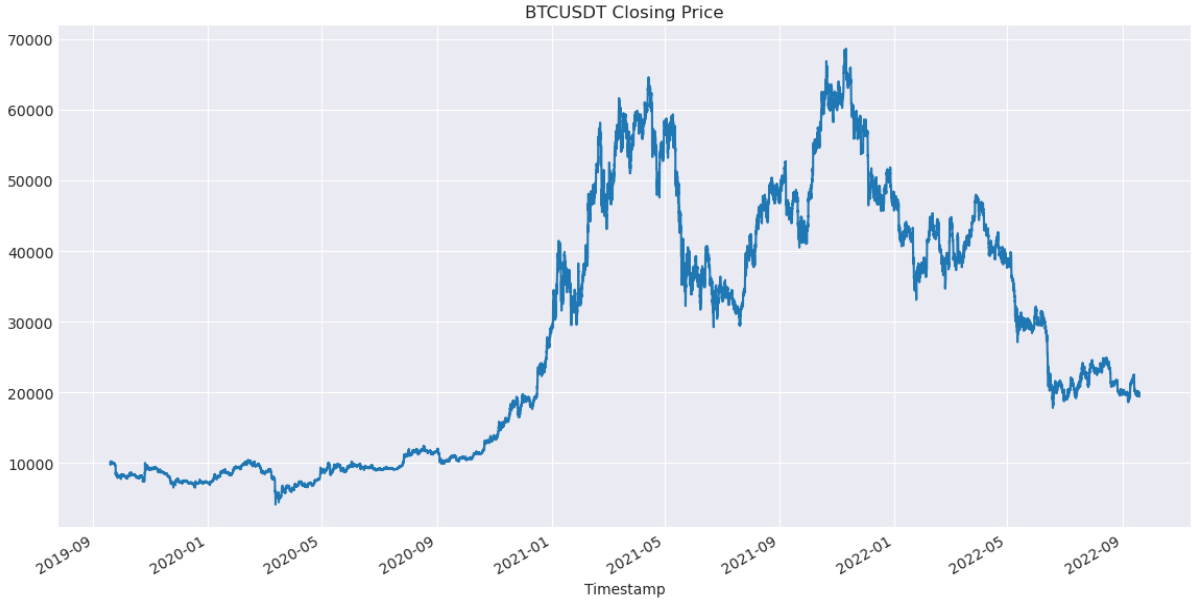

Figure 9. BTCUSDT pair plot over time

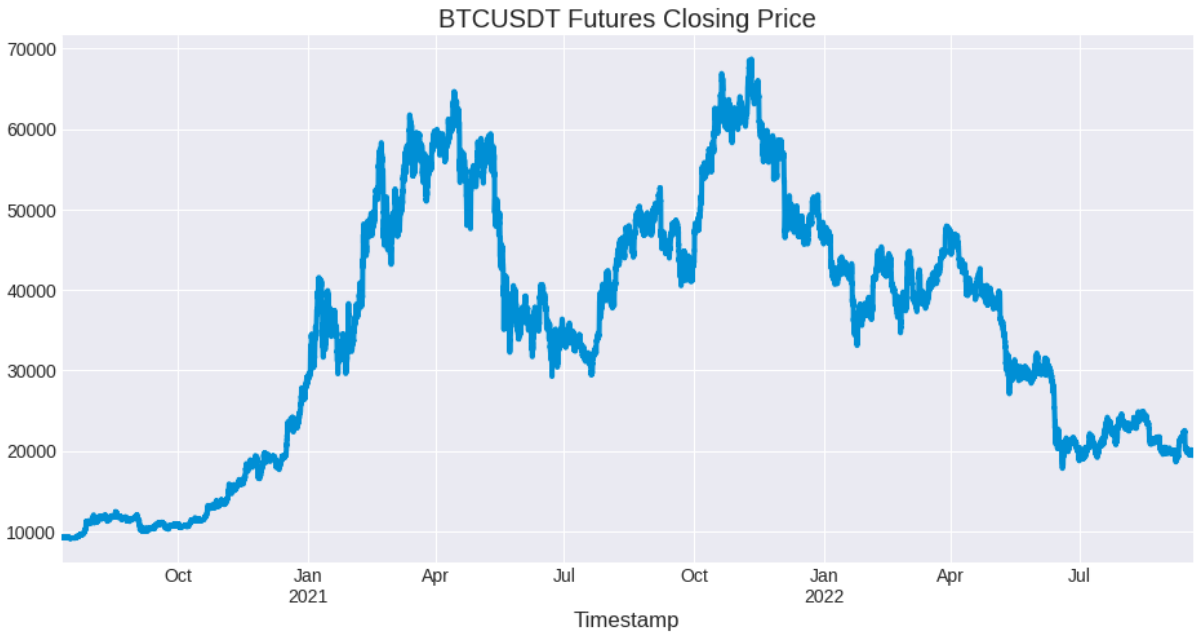

Figure 10. BTCUSDT Futures pair plot over time

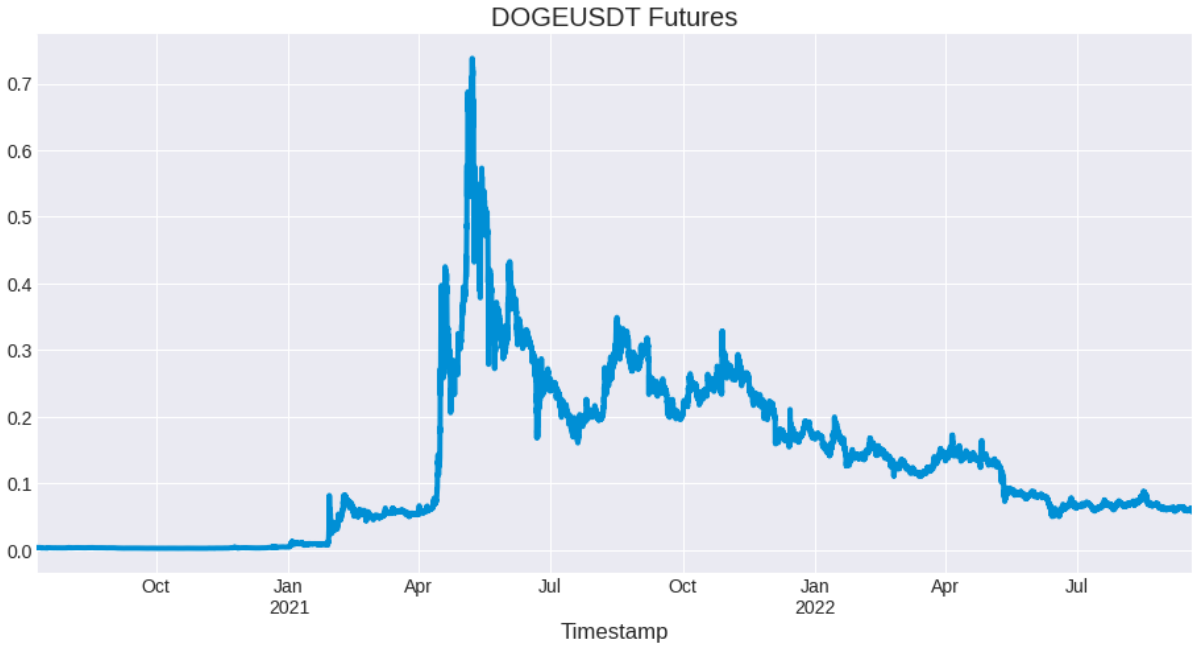

Figure 11. DOGEUSDT Futures pair plot over time

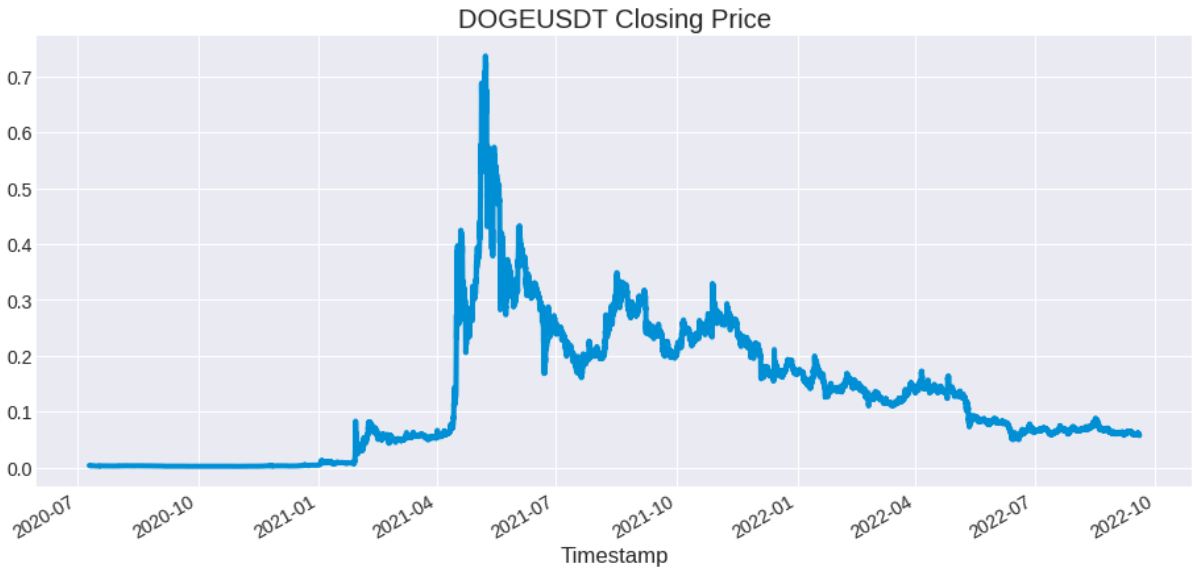

Figure 12. DOGEUSDT pair plot over time

When exploring the data, it is also useful to summarise the data with Density plots to see where the mass of the data is located, as shown in Figure 13 and Figure 14.

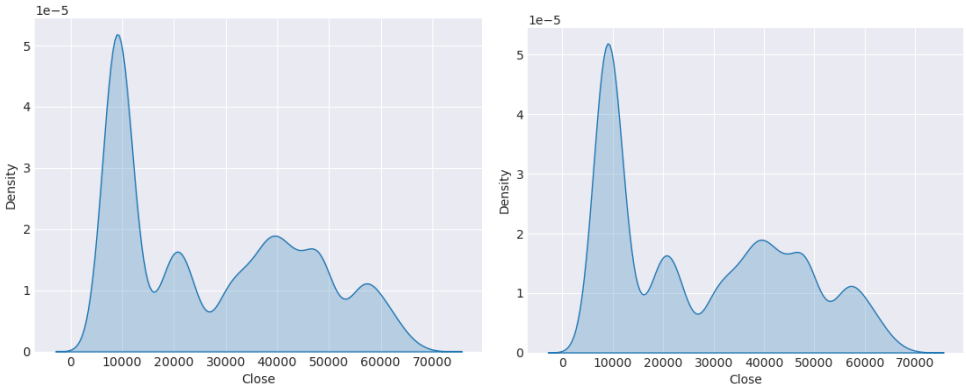

Figure 13. Density plot of BTCUSDT Futures (left) and BTUSDT pair (right)

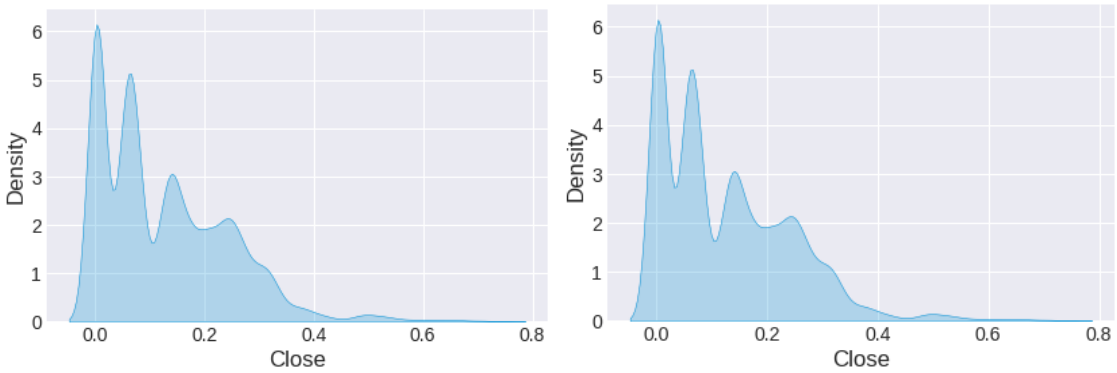

Figure 14. Density Plot of DOGEUSDT Futures (left) and DOGEUSDT pair (right)

Examining stock price data for every hour day can be counterproductive, with many investors more interested in spotting market trends. To make it easier, a process called time resampling to aggregate data into a defined time period was applied, such as by day, month or by quarter. Investors can then see an overview of commodities performance and make decisions according to the trends. The pandas library has a .resample() function which resamples such time series data.

To define the time period of data resampling, lag plots were explored, where the Y-axis lags the X-axis by an hour, a day, a week or a month, as shown in Figure 15 and Figure 16.

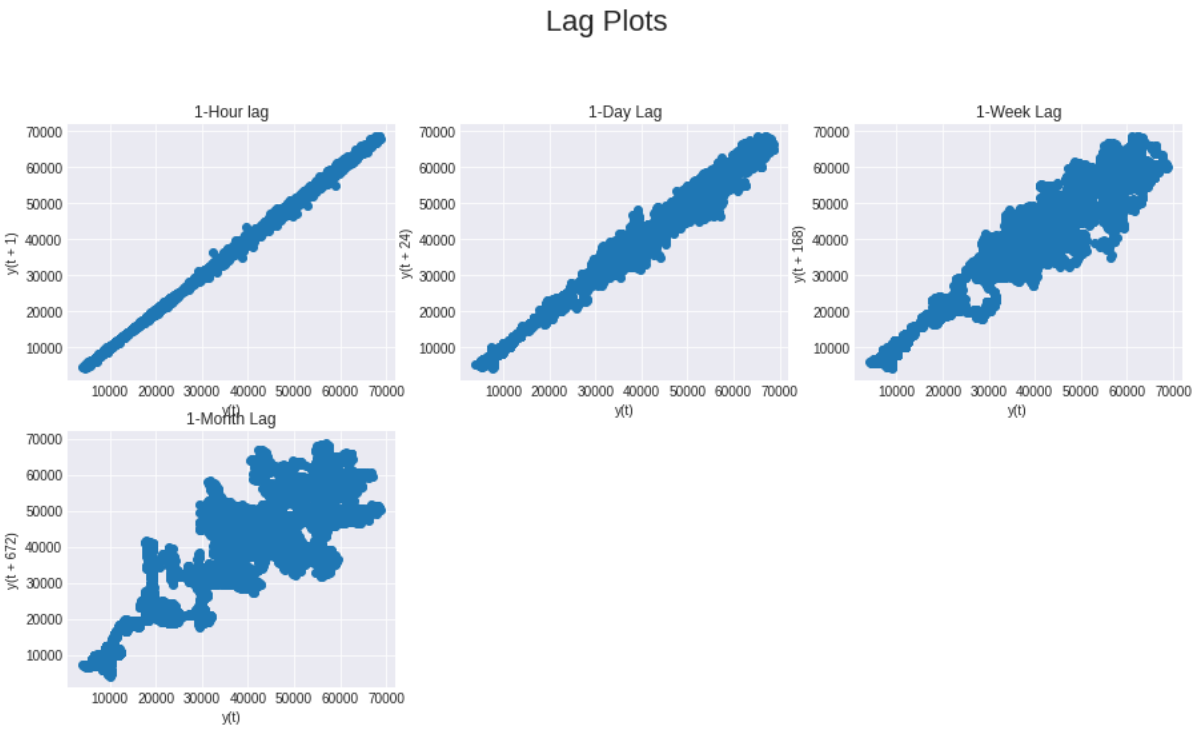

Figure 15. Lag Plots for BTCUSDT Futures

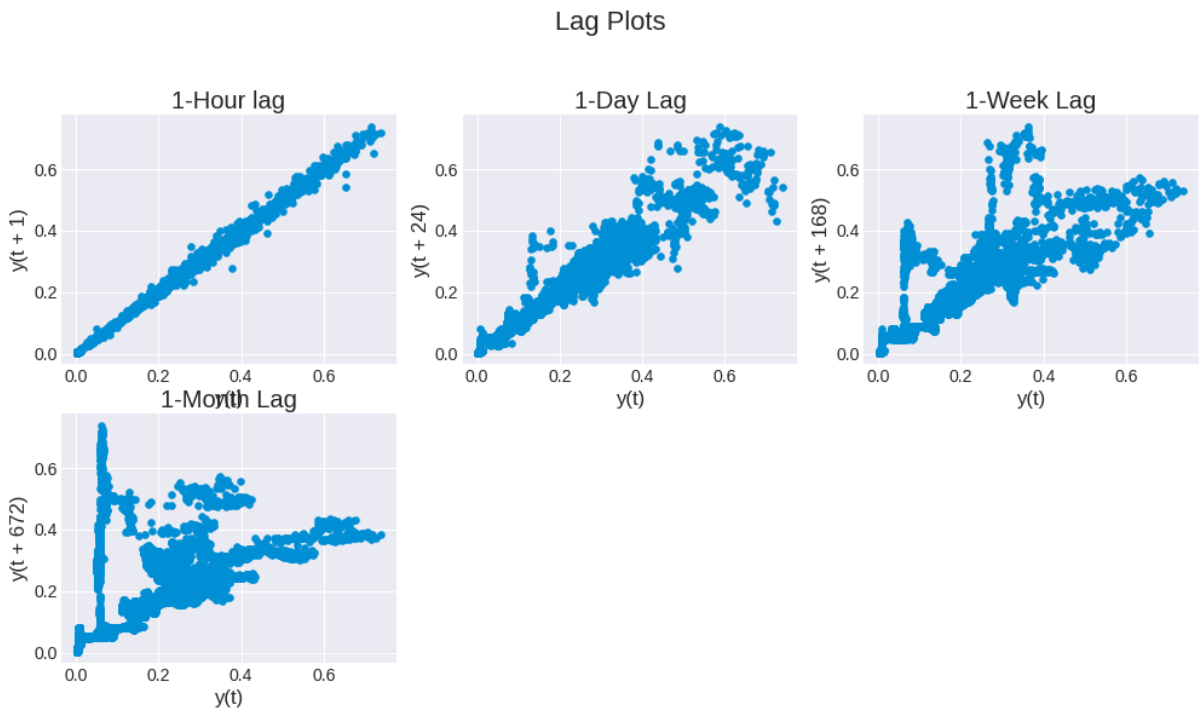

Figure 16. Lag Plots for BTCUSDT

The lag plots of Doge pairs exhibit similar trend. We can see that there is a positive correlation for hourly and daily lag plots, with a weaker correlation for weekly and monthly lags. Therefore, it makes sense to re-sample our data at the Daily level, thereby preserving the autocorrelation as well.

3.2.2 Timeseries decomposition

Timeseries can be decomposed into trend, seasonal and remainder components. The series can be decomposed as an additive or multiplicative combination of the base level, trend, seasonal index and the residual. The seasonal_decompose in statsmodels is used to implements the decomposition.

Figure 17 and Figure 18 show the timeseries additive decomposition for BTC and DOGE futures. Two main takeaways are that there are no seasonality and no constant mean, variance and covariance for both cryptocurrencies, hence the series are Non-Stationary. Statistical tests such as KPSS and ADF will be performed to confirm this conclusion.

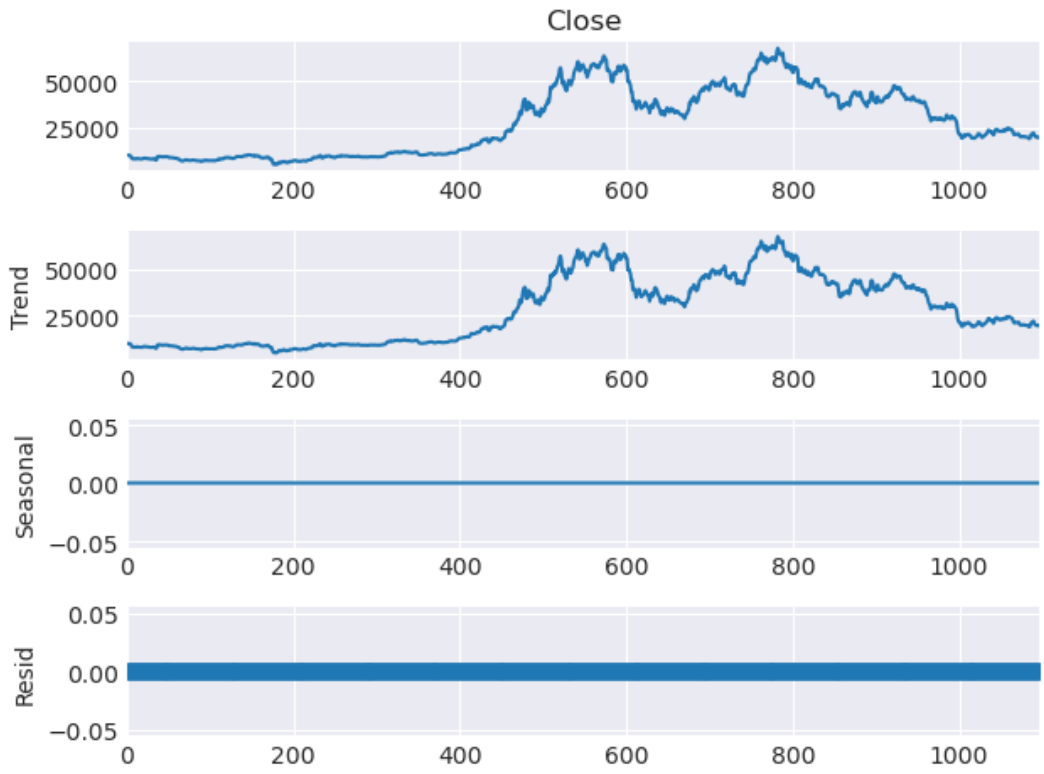

Figure 17. BTCUSDT Futures time series decomposition

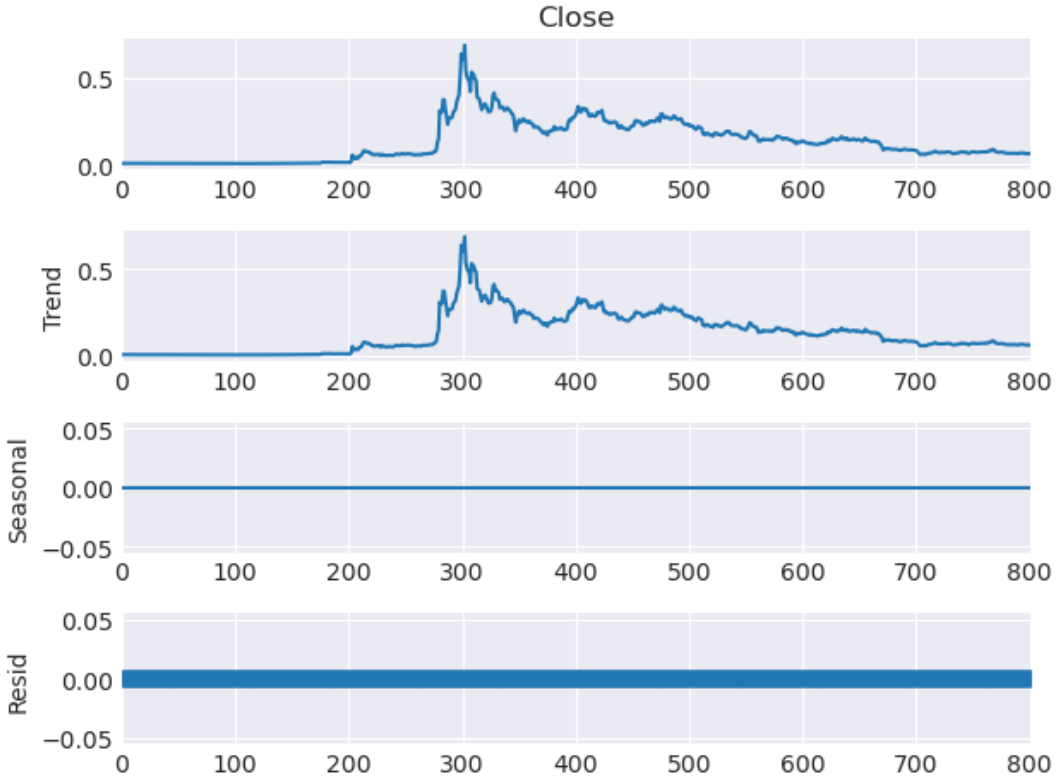

Figure 18. DOGEUSDT Futures time series decomposition

3.2.3 KPSS Test

As discussed previously, the null hypothesis of the KPSS test is that the series is stationary. Figure 19 shows the result of the KPSS on the BTCUSDT Futures pair, and as shown, the null hypothesis is rejected. Similar results were obtained for the other three time series.

```
Test Statistics : 0.7749474235472074
p-value : 0.01
Critical Values : {'10%': 0.119, '5%': 0.146, '2.5%': 0.176, '1%': 0.216}
Series is not Stationary
```

Figure 19. KPSS Test for BTCUSDT Futures

3.2.4 ADF Test

ADF is used to test stationarity as well, a major difference from KPSS is that the null hypothesis of ADF is that the data is non-stationary, with the presence of unit root.

This test indicates that the data is rather stationary, as shown in Figure 20. This inconsistency can mean that the non-stationarity of data can be removed by using differencing, which will be further explored in ARIMA modelling.

```
Results of Dickey-Fuller Test:
Test Statistic                  -1.366337
p-value                          0.598304
#Lags Used                       4.000000
Number of Observations Used    1092.000000
Critical Value (1%)             -3.436353
Critical Value (5%)             -2.864190
Critical Value (10%)            -2.568181
dtype: float64
Series is Stationary
```

Figure 20. ADF stationarity test for BTCUSDT Futures

3.2.5 Feature Extraction and transformation

Time series data can be noisy due to high fluctuations in the market. As a result, it becomes difficult to gauge a trend or pattern in the data. Aiming to use other features such as the opening price, changes in volume, and the highest and lowest prices while reducing the impact of noise each of these features can bring, some transformations were applied to them.

A rolling mean, often known as a moving average, is a data processing technique that helps average away data noise. It functions by partitioning and aggregating the data into partitions based on a function, such as the mean or median.

Figure 21 shows a snippet of the code used to transform the features.

```
df.reset_index(drop=False, inplace=True)

lag_features = ["Open", "High", "Low","Volume"]
window1 = 3
window2 = 7
window3 = 30

df_rolled_3d = df[lag_features].rolling(window=window1, min_periods=0)
df_rolled_7d = df[lag_features].rolling(window=window2, min_periods=0)
df_rolled_30d = df[lag_features].rolling(window=window3, min_periods=0)

df_mean_3d = df_rolled_3d.mean().shift(1).reset_index()
df_mean_7d = df_rolled_7d.mean().shift(1).reset_index()
df_mean_30d = df_rolled_30d.mean().shift(1).reset_index()

df_std_3d = df_rolled_3d.std().shift(1).reset_index()
df_std_7d = df_rolled_7d.std().shift(1).reset_index()
df_std_30d = df_rolled_30d.std().shift(1).reset_index()

for feature in lag_features:
    df[f"{feature}_mean_lag{window1}"] = df_mean_3d[feature]
    df[f"{feature}_mean_lag{window2}"] = df_mean_7d[feature]
    df[f"{feature}_mean_lag{window3}"] = df_mean_30d[feature]

    df[f"{feature}_std_lag{window1}"] = df_std_3d[feature]
    df[f"{feature}_std_lag{window2}"] = df_std_7d[feature]
    df[f"{feature}_std_lag{window3}"] = df_std_30d[feature]

df.fillna(df.mean(), inplace=True)

df.set_index("Timestamp", drop=False, inplace=True)
df.head()
```

Figure 21. feature transformation code snippet

3.2.6 ARIMA

Before building ARIMA models, Autocorrelation and Partial Autocorrelation were plotted to get some useful insights that will guide building the ARIMA model.

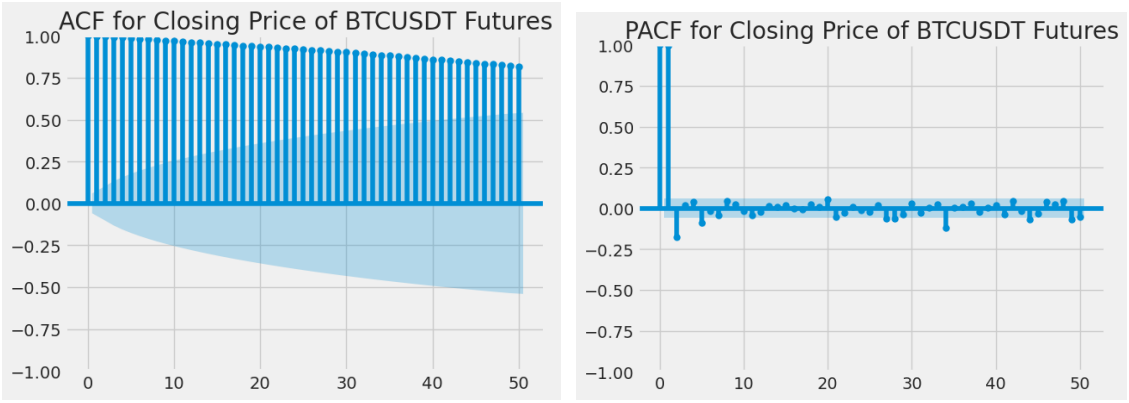

Figure 22. ACF and PACF of the closing price for BTCUSDT Futures

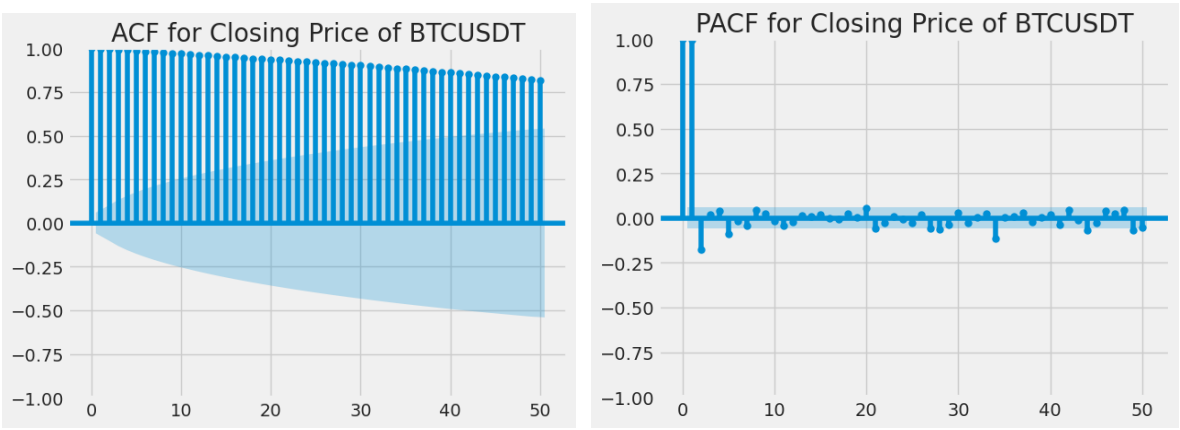

Figure 23. ACF and PACF of the closing price for BTCUSDT

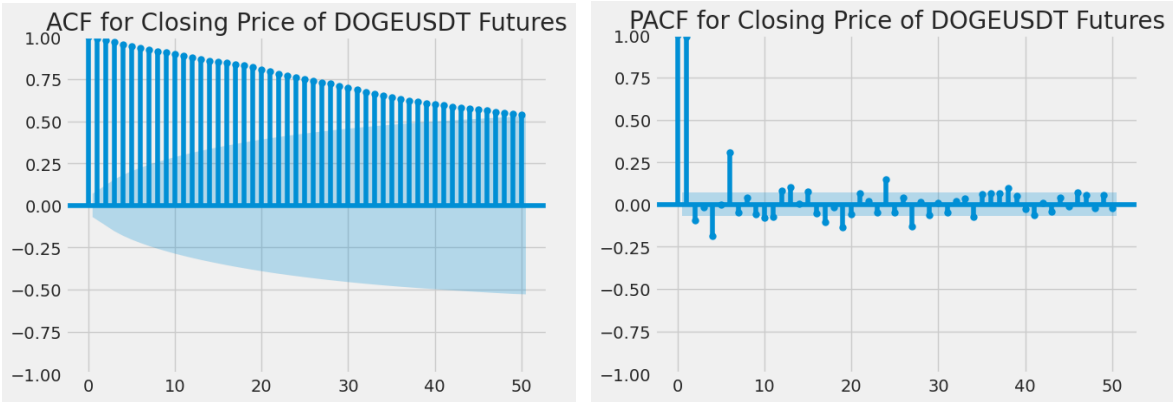

Figure 24. ACF and PACF of the closing price for DOGEUSDT Futures

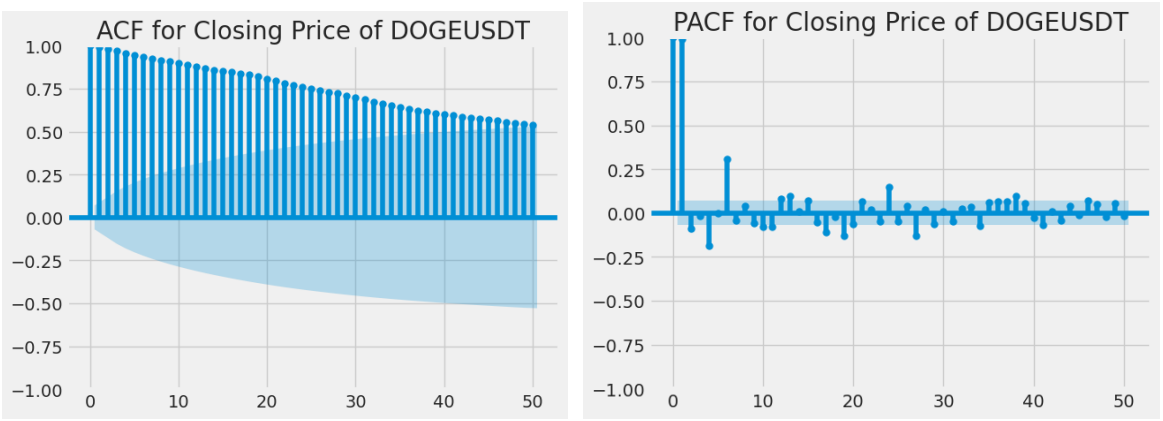

Figure 25. ACF and PACF of the closing price for DOGEUSDT

The charts in Figure 22 to Figure 25 show similar patterns, the ACF only slightly deteriorates over time, and lags up tp almost fifty lags have considerable impact. Moreover, looking at PACF charts, lags beyond five or seven are not statisitcally significant except for few spikes. These charts give an idea of the AR and MA terms used in ARIMA modelling, as discussed in the literature review.

Before building the model, the data was split into training and test sets to assess the performance of the forecasting model and make sure that it is not overfitting or underfitting, and make sure that its performance is generalisable. The split data is chosen to be on the first of May 2022, as shown in Figure 26.

```
split_date = "2022-05-01"
df_train = df[df.Timestamp < split_date]
df_valid = df[df.Timestamp >= split_date]

print('train shape :', df_train.shape)
print('validation shape :', df_valid.shape)
```

```
train shape : (955, 34)
validation shape : (142, 34)
```

Figure 26. splitting data into train and test sets.

For constructing the model, there are different combinations of components of nonseasonal AR (p), difference (d), and moving average window (q), as well as the seasonal components P, D, and Q, that can be iteratively tested to get the best model.

A more efficient and systematic method is to perform a stepwise search to minimise the AIC, using the auto_arima function from pmdarima Python package(Smith 2022).

Figure 27 shows the best combination of components after fitting the model for BTCUSDT Futures closing price, and as expected, there are no seasonal components. A similar procedure was performed for the other three time series data, resulting in orders of ARIMA, some of which have differencing, as suggested by the stationarity tests discussed in the methodology.

```
Best model:  ARIMA(1,0,2)(0,0,0)[0]
```

Figure 27. Best model components of ARIMA for BTCUSDT Futures

3.2.7 Facebook Prophet

For Facebook Prophet model, the same data split was used as in ARIMA, then the model was fit and optimised. The results and performance validation is shown in the Results chapter.

3.2.8 XGBoost

To ensure consistency and optimal results for XGBoost models, a range of values was specified, in a Hyper Parameter Optimization Grid as shown in Figure 28. There are several algorithms to search for the optimal parameters, in this work RandomizedSearchCV was used, due to its swiftness and accuracy.

```
## Hyper Parameter Optimization Grid

params={
 "learning_rate"    : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30],
 "max_depth"        : [1, 3, 4, 5, 6, 7],
 "n_estimators"     : [int(x) for x in np.linspace(start=100, stop=2000, num=10)],
 "min_child_weight" : [int(x) for x in np.arange(3, 15, 1)],
 "gamma"            : [ 0.0, 0.1, 0.2 , 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
 "subsample"        : [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1],
 "colsample_bytree" : [0.5, 0.6, 0.7, 0.8, 0.9, 1],
 "colsample_bylevel": [0.5, 0.6, 0.7, 0.8, 0.9, 1],

}
```

Figure 28. Hyperparameter Optimisation Grid for XGBoost

The optimised parameters for BTCUSDT futures data were:

```
Model Best Score : 0.4398010269991578
Model Best Parameters : {'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'callbacks': None, 'colsample_bylevel': 0.
7, 'colsample_bynode': 1, 'colsample_bytree': 0.7, 'early_stopping_rounds': None, 'enable_categorical': False, 'eval_metric': None, 'gamma':
0.2, 'gpu_id': -1, 'grow_policy': 'depthwise', 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.1, 'max_bin': 256,
'max_cat_to_onehot': 4, 'max_delta_step': 0, 'max_depth': 3, 'max_leaves': 0, 'min_child_weight': 8, 'missing': nan, 'monotone_constraints':
'()', 'n_estimators': 2000, 'n_jobs': 0, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 0, 'reg_lambda': 1, 'sa
mpling_method': 'uniform', 'scale_pos_weight': 1, 'subsample': 0.9, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None}
```

And for BTCUSDT:

```
Model Best Score : 0.4792375399190867
Model Best Parameters : {'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'callbacks': None, 'colsample_bylevel': 0.
8, 'colsample_bynode': 1, 'colsample_bytree': 1, 'early_stopping_rounds': None, 'enable_categorical': False, 'eval_metric': None, 'gamma': 0.
7, 'gpu_id': -1, 'grow_policy': 'depthwise', 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.1, 'max_bin': 256, 'm
ax_cat_to_onehot': 4, 'max_delta_step': 0, 'max_depth': 7, 'max_leaves': 0, 'min_child_weight': 5, 'missing': nan, 'monotone_constraints':
'()', 'n_estimators': 1155, 'n_jobs': 0, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 0, 'reg_lambda': 1, 'sa
mpling_method': 'uniform', 'scale_pos_weight': 1, 'subsample': 0.9, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None}
```

For DOGEUSDT Futures:

```
Model Best Score : -4002.6570675498974
Model Best Parameters : {'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'callbacks': None, 'colsample_bylevel': 0.
5, 'colsample_bynode': 1, 'colsample_bytree': 1, 'early_stopping_rounds': None, 'enable_categorical': False, 'eval_metric': None, 'gamma': 0.
1, 'gpu_id': -1, 'grow_policy': 'depthwise', 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.2, 'max_bin': 256, 'm
ax_cat_to_onehot': 4, 'max_delta_step': 0, 'max_depth': 5, 'max_leaves': 0, 'min_child_weight': 7, 'missing': nan, 'monotone_constraints':
'()', 'n_estimators': 522, 'n_jobs': 0, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 0, 'reg_lambda': 1, 'sam
pling_method': 'uniform', 'scale_pos_weight': 1, 'subsample': 0.3, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None}
```

And finally for DOGEUSDT:

```
Model Best Score : -297.1400906335141
Model Best Parameters : {'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'callbacks': None, 'colsample_bylevel': 0.
5, 'colsample_bynode': 1, 'colsample_bytree': 0.7, 'early_stopping_rounds': None, 'enable_categorical': False, 'eval_metric': None, 'gamma':
0.0, 'gpu_id': -1, 'grow_policy': 'depthwise', 'importance_type': None, 'interaction_constraints': '', 'learning_rate': 0.15, 'max_bin': 256,
'max_cat_to_onehot': 4, 'max_delta_step': 0, 'max_depth': 5, 'max_leaves': 0, 'min_child_weight': 4, 'missing': nan, 'monotone_constraints':
'()', 'n_estimators': 522, 'n_jobs': 0, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 0, 'reg_lambda': 1, 'sam
pling_method': 'uniform', 'scale_pos_weight': 1, 'subsample': 0.1, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None}
```

Afterwards, the optimal hyperparameters were used in fitting XGBoost models and forecasting data.

3.2.9 LSTM

For LSTM, it is necessary to scale the data between zero and one before training the neural network, as is advised by the literature. Afterwards, some data engineering and transformations to the shape of the data were applied to become compatible Keres's implementation of LSTM.

Figure 29 shows the architecture of the LSTM, it consists of an input layer, three LSTM layers where the bulk of learning is done, and each is followed by a dropout of .2 as a regularisation method and to avoid overfitting. Finally, an output layer of 1. The hyperparameter optimiser used is Adam, aiming to minimise the mean squared error of the model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 100, 50)           10400

 dropout (Dropout)           (None, 100, 50)           0

 lstm_1 (LSTM)               (None, 100, 50)           20200

 dropout_1 (Dropout)         (None, 100, 50)           0

 lstm_2 (LSTM)               (None, 100, 50)           20200

 dropout_2 (Dropout)         (None, 100, 50)           0

 lstm_3 (LSTM)               (None, 50)                20200

 dropout_3 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
```

Figure 29. LSTM neural network architecture

4 Evaluation and results

4.1 ARIMA

4.1.1 BTCUSDT Futures

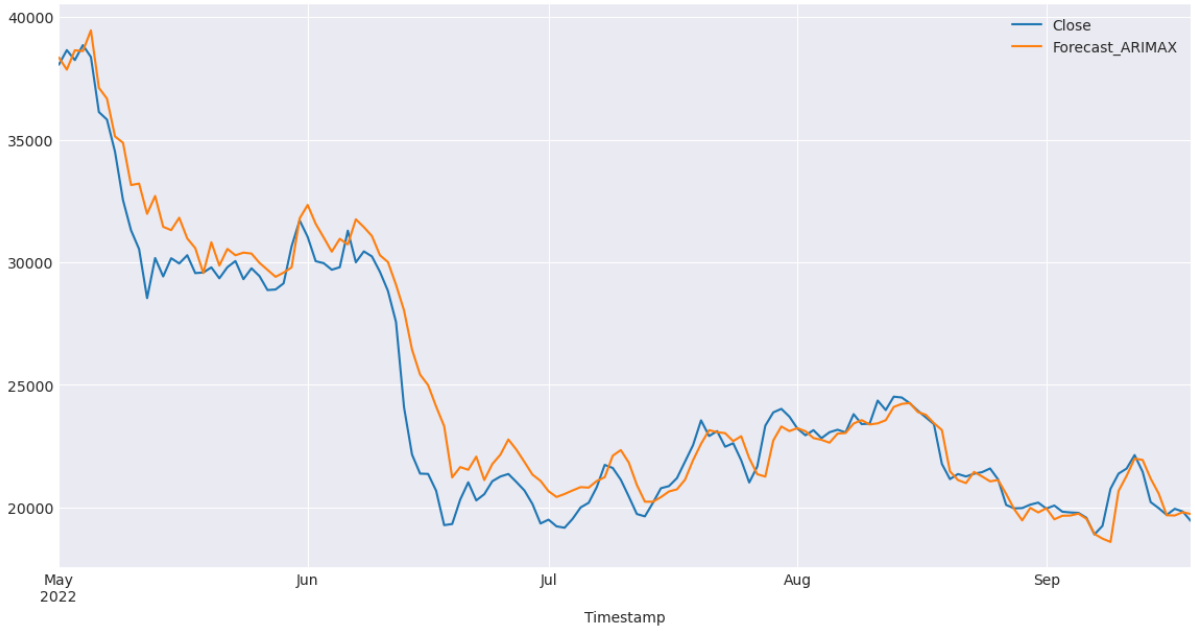

Figure 30. BTCUSDT Futures ARIMAX model forecast

```
RMSE of Auto ARIMAX: 1240.8397540514538

MAE of Auto ARIMAX: 863.1415967208933
```

Figure 31. Forecasting error of ARIMAX for BTCUSDT Futures

Figure 30 and Figure 31 compare the forecast of ARIMA to the actual closing price of BTCUSDT Futures, a noteworthy observation is the deviation between the forecast and the actual when sudden jumps occur, such as the two huge decreases in price after May and between June and July, apart from that, ARIMA seemed to capture the slight variations without as much deviation. We can further analyse the performance of ARIMA after looking at Figure 32.

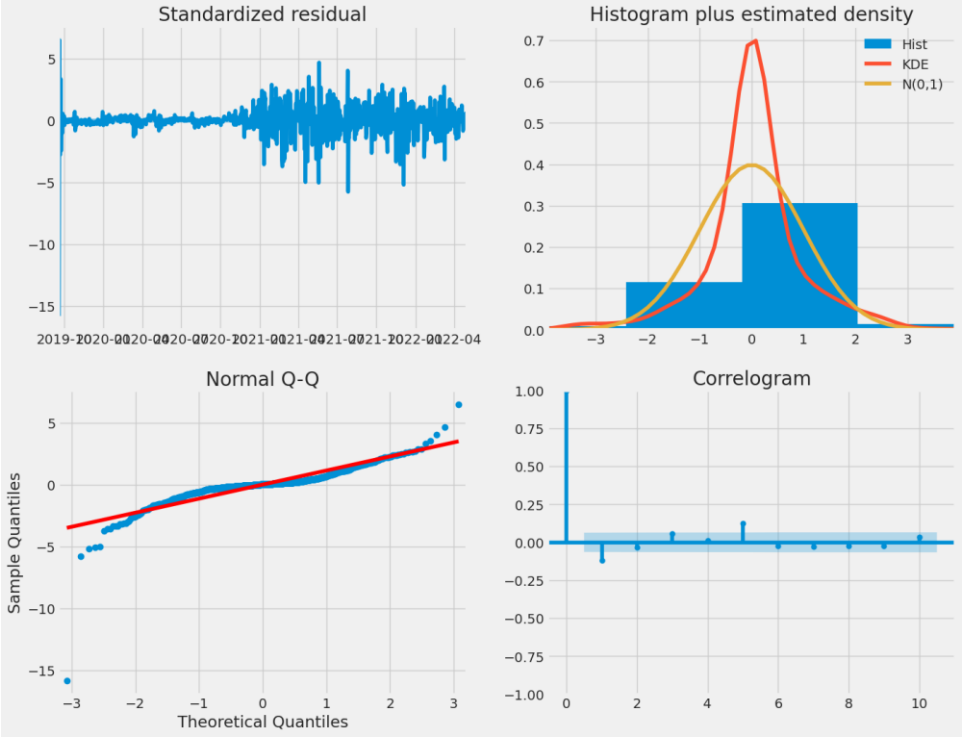

Figure 32. BTCUSDT Futures ARIMA forecast analysis

The primary concern when looking at Figure 32 is to check if the residuals of the model are uncorrelated and normally distributed with zero-mean. Starting from top left, the standardised residuals do not show any obvious seasonality over time, however, the increased volatility signals once again ARIMA's model inefficiency in capturing sudden jumps in price.

The histogram (top right), the KDE line should ideally follow the N(0,1) line (normal distribution with mean 0, standard deviation 1) closely. This is an indication whether the residuals are normally distributed or not.

In the Q-Q-plot the ordered distribution of residuals (blue dots) should follow the linear trend of the samples taken from a standard normal distribution with N(0, 1). Again, this is an indication whether the residuals are normally distributed.

The correlogram shows clearly the time series residuals have low correlation with lagged version of itself, which means that ARIMA successfully captured how previous values impact later ones (while keeping in mind the shortcomings mentioned earlier).

4.1.2 BTCUSDT

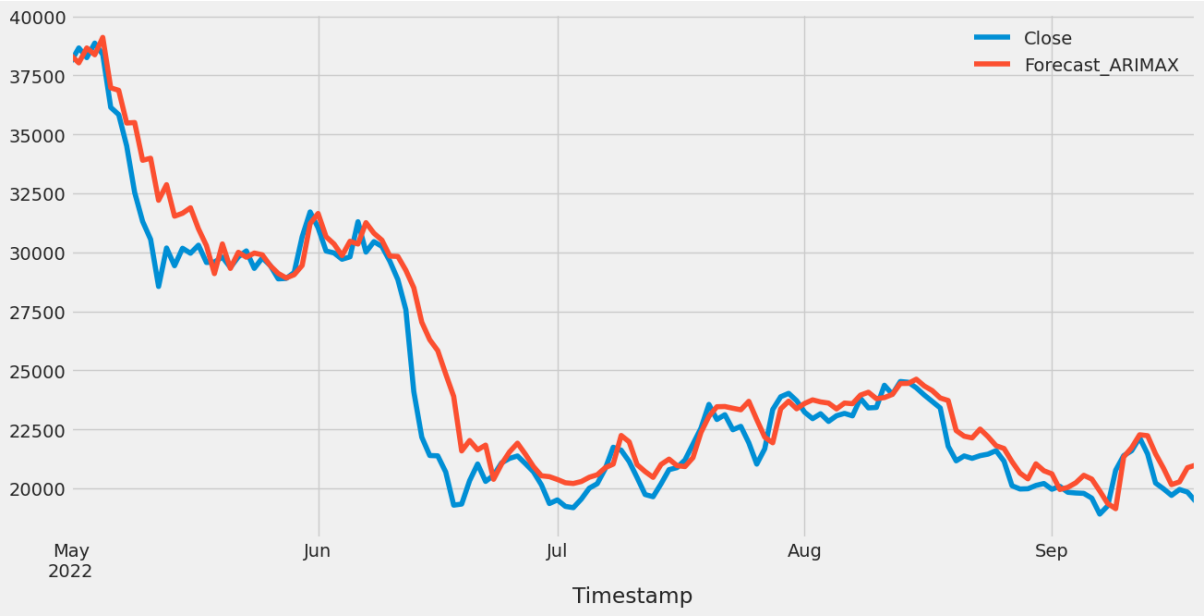

Figure 33. BTCUSDT ARIMA model forecast

Figure 33 shows a similar forecasting performance of ARIMA to the BTCUSDT Futures, a noticeable deviation when the price fell sharply, but a more reliable forecast otherwise.

```
RMSE of Auto ARIMAX: 1352.3872751129745

MAE of Auto ARIMAX: 904.3041012235992
```

Figure 34. Forecasting error of ARIMAX for BTCUSDT

The forecasting error shown in Figure 34 is more than the Futures of the coin, but the trend and general performance are overall similar.

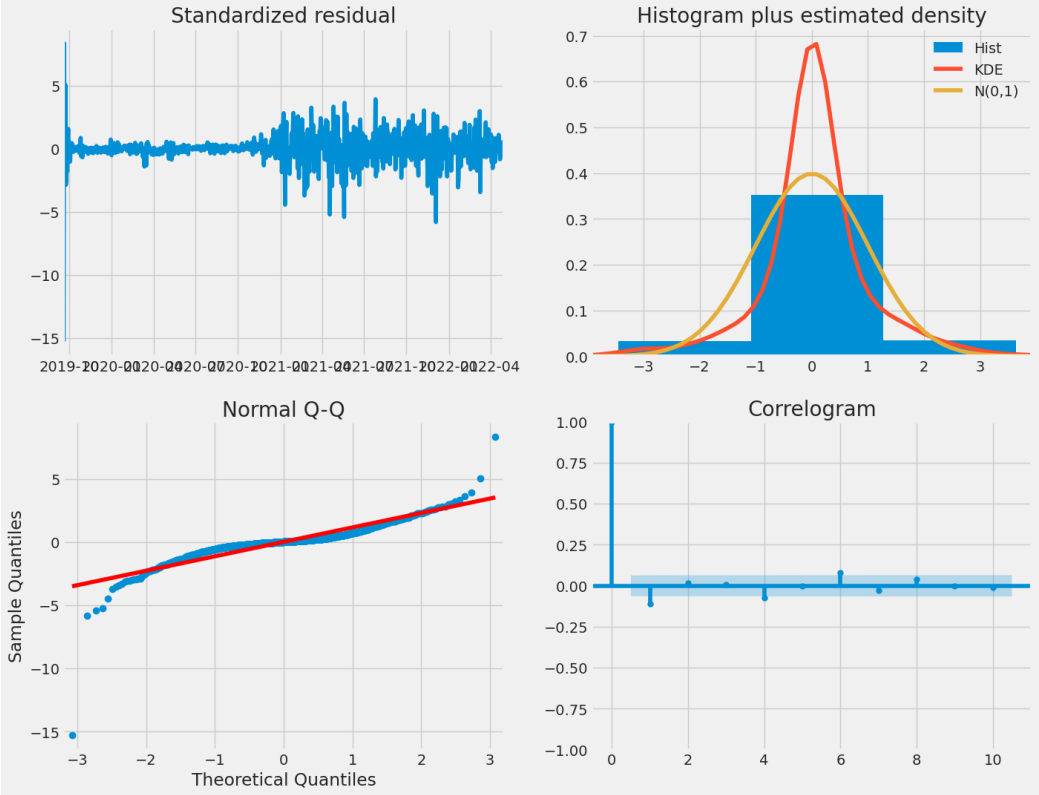

Figure 35. BTCUSDT ARIMA forecast analysis

The model performs similarly in both the coin and its futures as confirmed by Figure 35, with slight variations.

4.1.3 DOGEUSDT Futures

Figure 36. DOGEUSDT Futures ARIMA model forecast

```
RMSE of Auto ARIMAX: 0.004773340115544782

MAE of Auto ARIMAX: 0.0035563223045747606
```

Figure 37. Forecasting error of ARIMAX for DOGEUSDT Futures

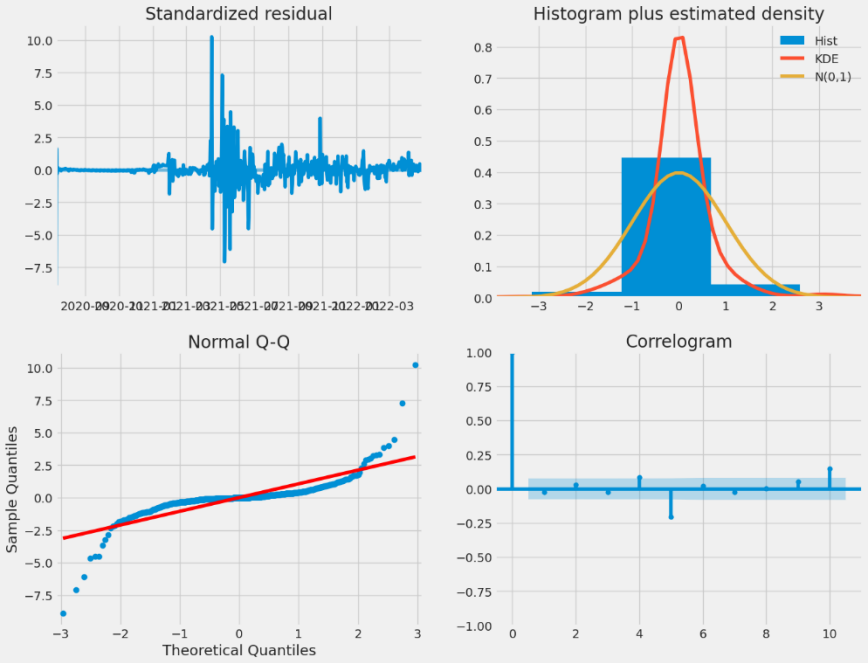

Figure 38. DOGEUSDT Futures ARIMA forecast analysis

4.1.4 DOGEUSDT

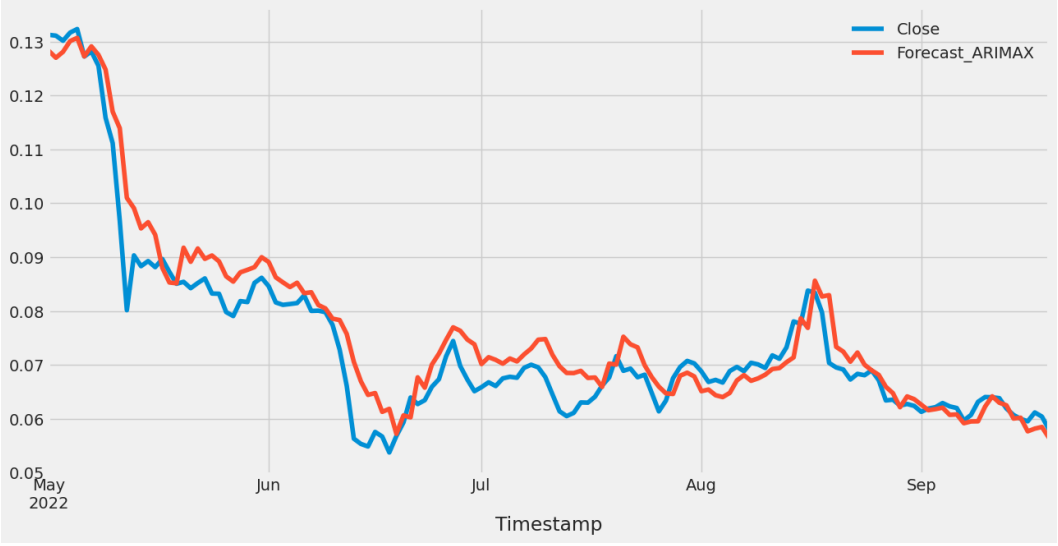

Figure 39. DOGEUSDT ARIMA model forecast

```
RMSE of Auto ARIMAX: 0.0050779340255159245

MAE of Auto ARIMAX: 0.0038852259794653556
```

Figure 40. Forecasting error of ARIMAX for DOGEUSDT

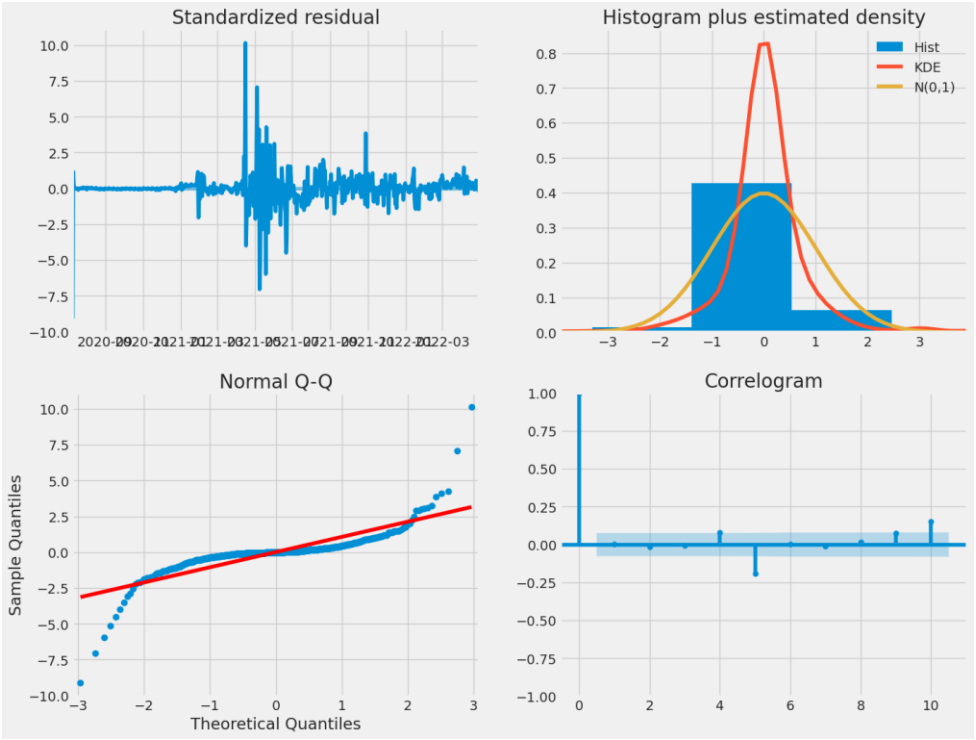

Figure 41. DOGEUSDT ARIMA forecast analysis

From Figure 36 to Figure 41 we can see the same pattern as in BTC, as in the model performs similarly for both the regular coin and its futures, albeit slightly better for the futures. However, from the forecast plots and forecast analysis, we see that ARIMA performs is achieving a better forecast for a stable coin (BTCUSDT) than a more volatile coin (DOGEUSDT). Nevertheless, we cannot generalise this as the futures data is still limited.

4.2 Facebook Prophet

4.2.1 BTCUSDT Futures

Figure 42. BTCUSDT Futures Prophet Forecast

In Figure 42, the black dots represent the actual data points in the dataset, the deep blue line is the predicted forecast/the predicted values, and the light blue line represents confidence levels of forecasts.

The black dots help us spot an important characteristic of the rises and falls of the BTCUSDT Futures, which is the magnitude by which they rise and fall, the charts below will help us answer if that would impact the performance of Prophet.

Figure 43. trends and seasonalities for different time frames (BTCUSDT Futures)

As mentioned earlier, Prophet was developed by Facebook to automate forecasting of their data, therefore there are tools that make it easier to spot seasonalities by the day of the week or

the month of the year as in Figure 43, which is an added bonus that is not easily replicable in ARIMA or other existing packages and can help researchers in understanding if certain yearly events such as holidays impact the price.

Figure 44. Change points in trend as detected by Prophet for BTCUSDT Futures

Figure 44 Prophet also shows the points in time where there is a change in trend, which can help analysers look for the source of these changes and whether its technical forecasting or motivated by external factors such as a government policy.

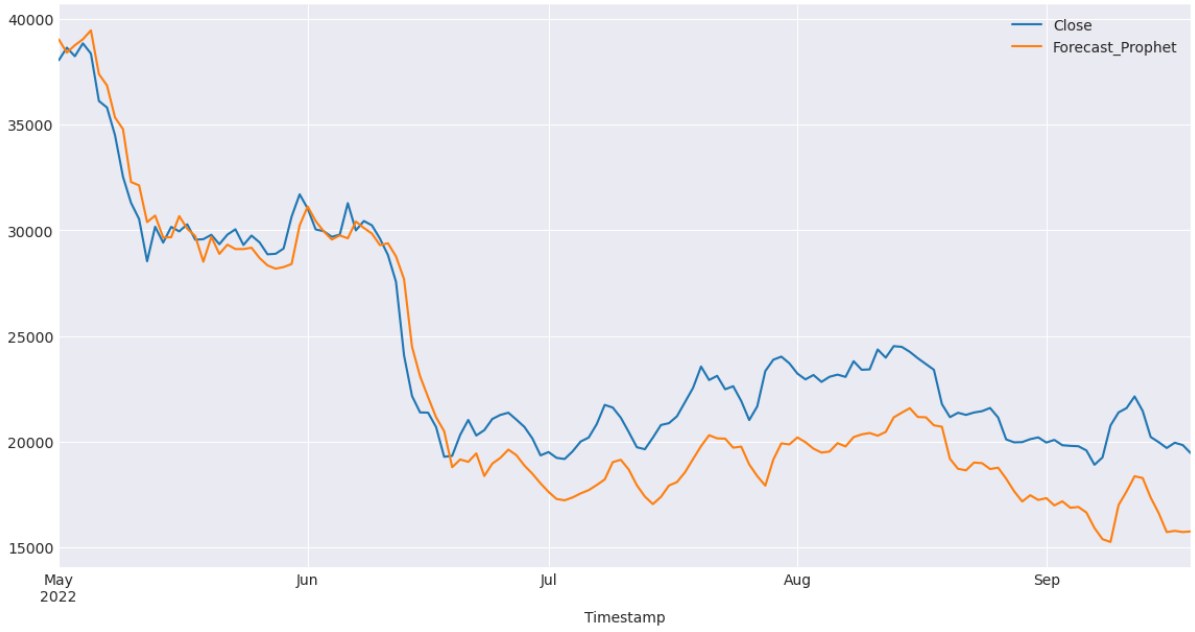

Figure 45. Forecast and actual price comparison for BTCUSDT Futures Prophet

```
Prophet's MAE : 2147.3615136862118
Prophet's RMSE : 2492.5280941053625
```

Figure 46. Forecasting error of FBProphet for BTCUSDT Futures

Figure 45 and Figure 46 show how the performance of prophet deteriorated mid-June onwards, which suggests that it's better suited for short-term forecasting.

4.2.2 BTCUSDT

Figure 47. BTCUSDT Prophet Forecast

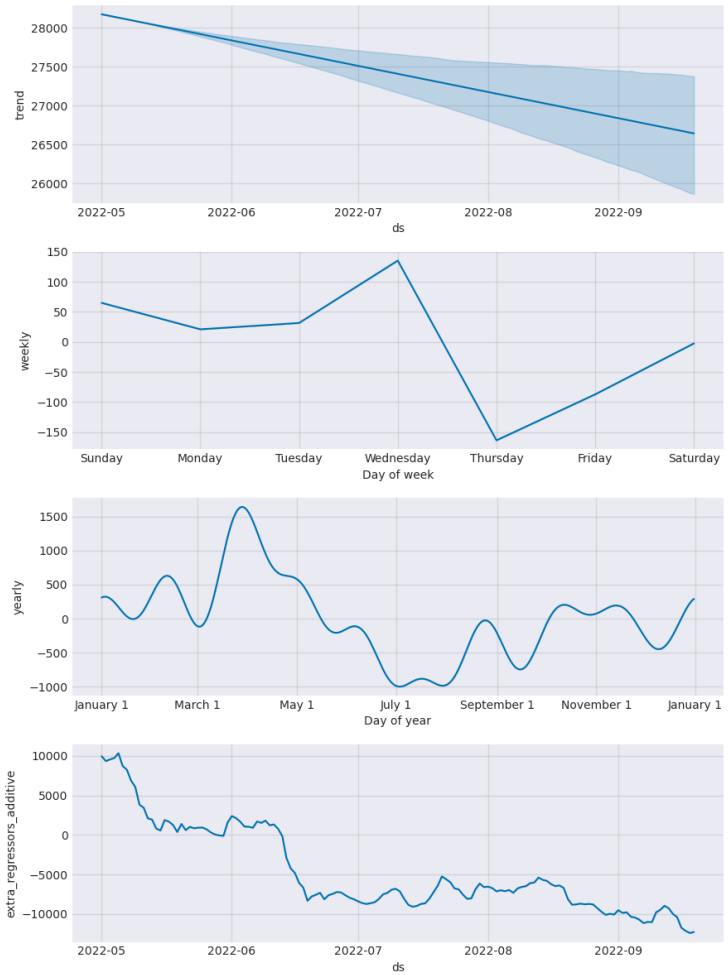

Figure 48. trends and seasonalities for different time frames (BTCUSDT)

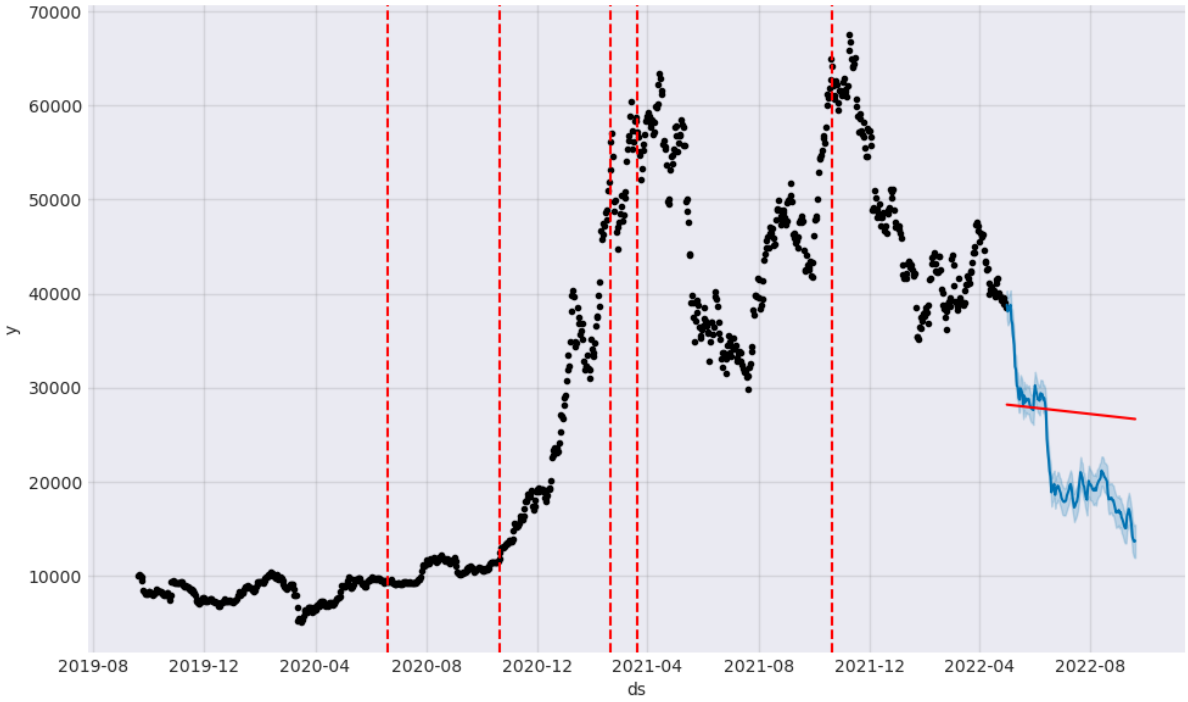

Figure 49. Change points in trend as detected by Prophet for BTCUSDT

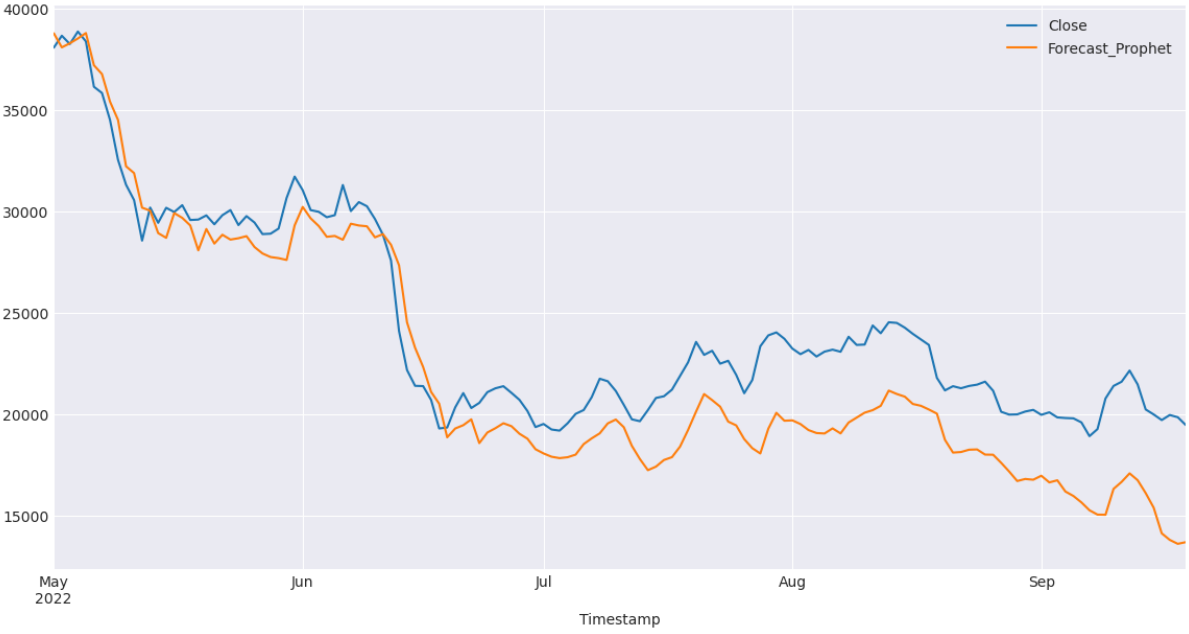

Figure 50. Forecast and actual price comparison for BTCUSDT Prophet

```
Prophet's MAE : 2408.600100526406
Prophet's RMSE : 2822.562480095544
```

Figure 51. Forecasting error of FBProphet for BTCUSDT

4.2.3 DOGEUSDT Futures

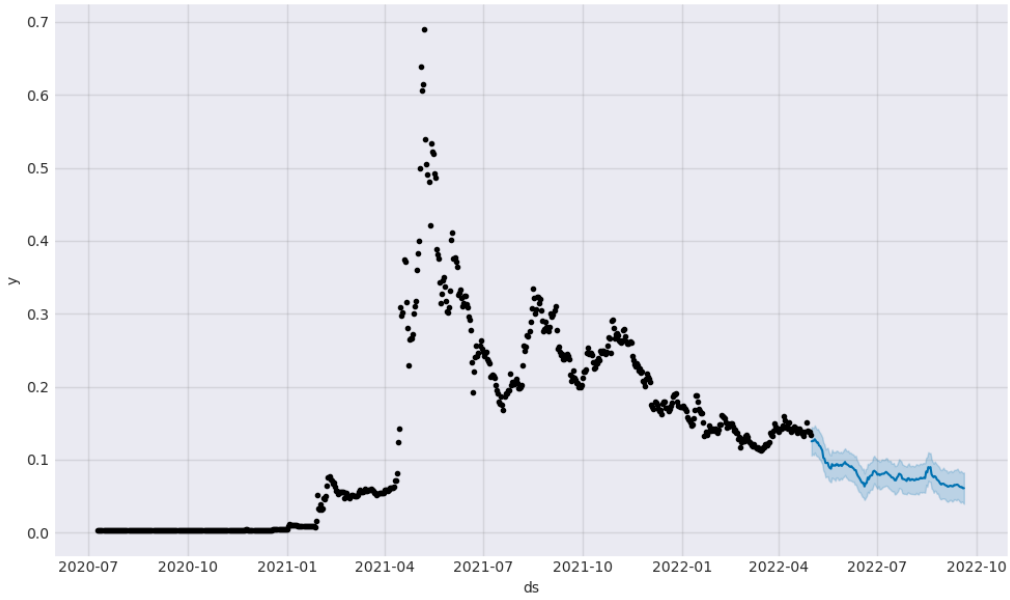

Figure 52. DOGEUSDT Futures Prophet Forecast

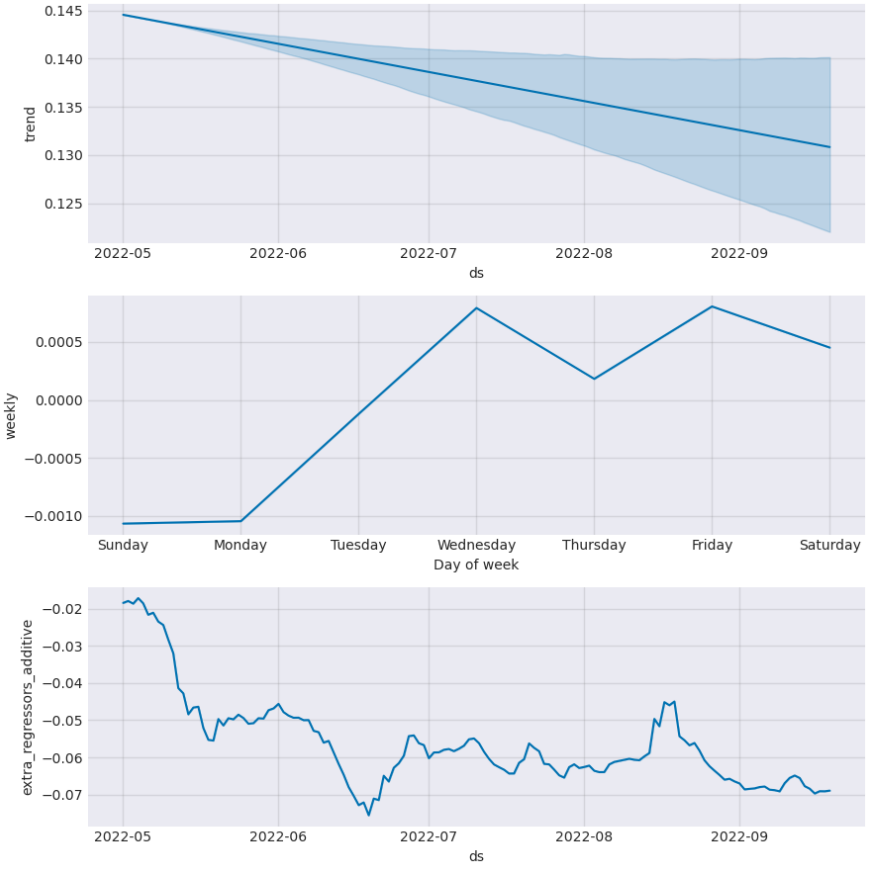

Figure 53. trends and seasonalities for different time frames (DOGEUSDT Futures)

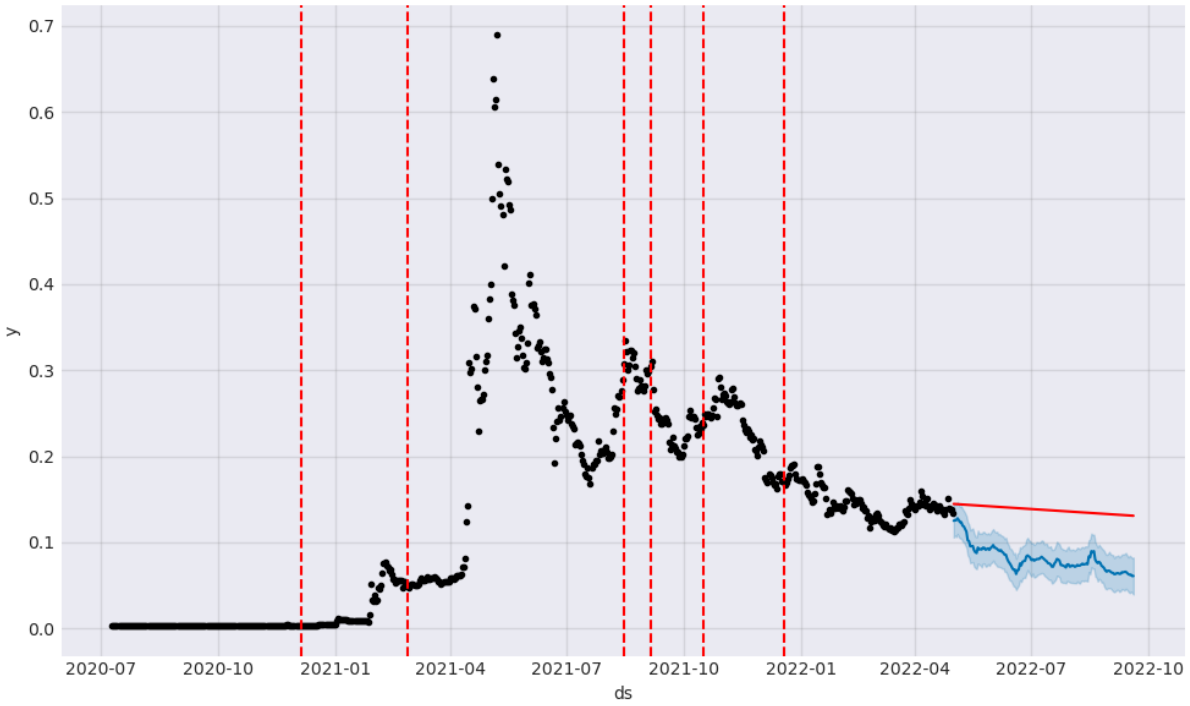

Figure 54. Change points in trend as detected by Prophet for DOGEUSDT Futures

Figure 55. Forecast and actual price comparison for DOGEUSDT Futures Prophet

```
Prophet's MAE : 0.007906665005330542
Prophet's RMSE : 0.009304911999445954
```

Figure 56. Forecasting error of FBProphet for DOGEUSDT Futures

4.2.4 DOGEUSDT

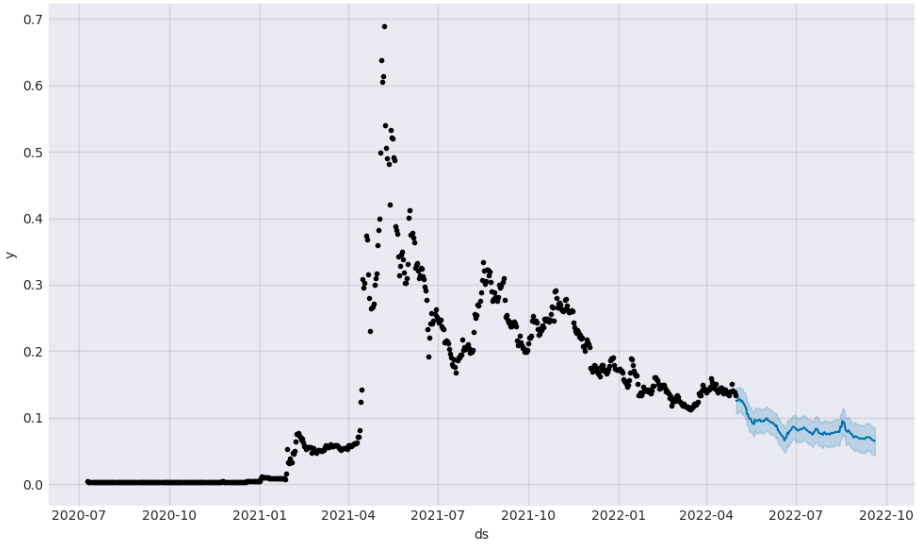

Figure 57. DOGEUSDT Prophet Forecast

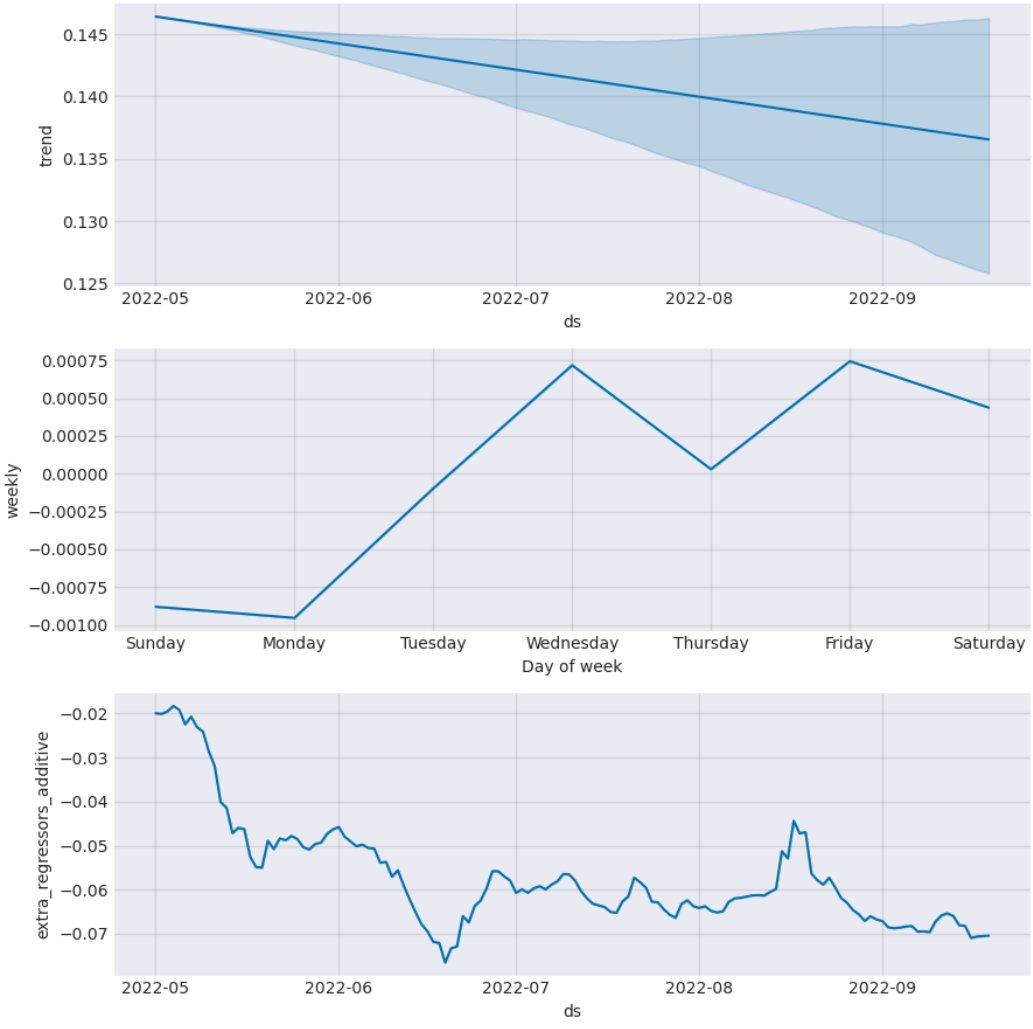

Figure 58. trends and seasonalities for different time frames (DOGEUSDT)

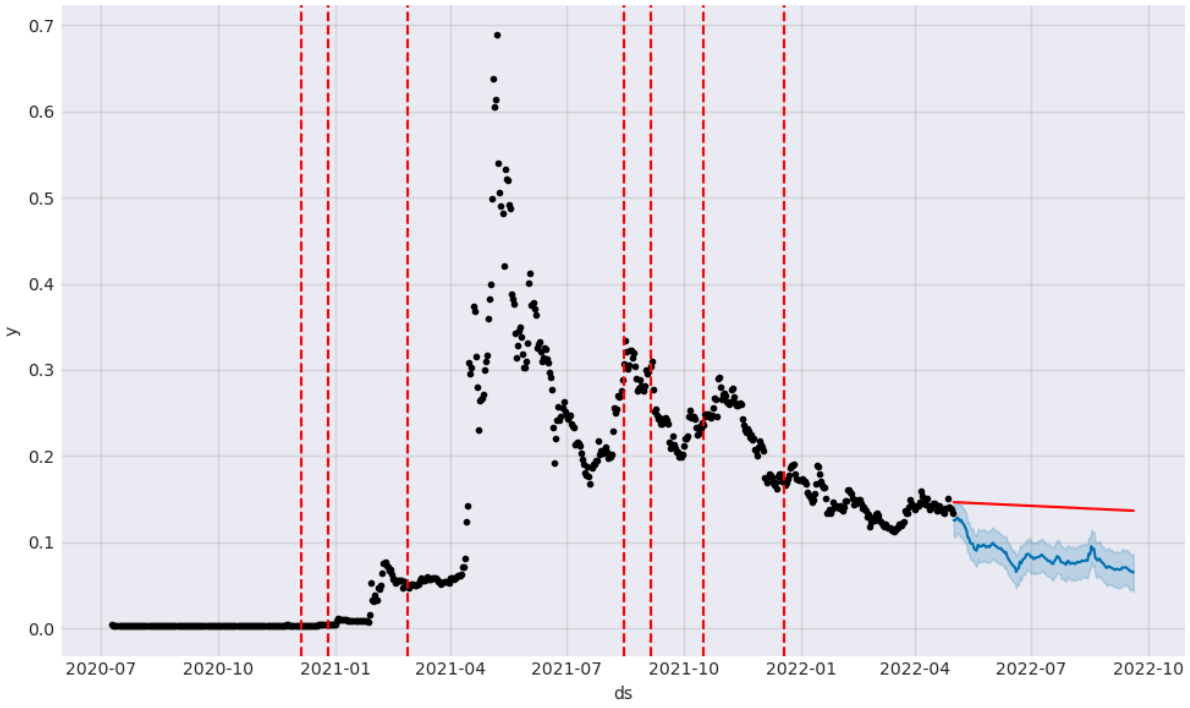

Figure 59. Change points in trend as detected by Prophet for DOGEUSDT

Figure 60. Forecast and actual price comparison for DOGEUSDT Prophet

```
Prophet's MAE : 0.010804889192175048
Prophet's RMSE : 0.011808755621878358
```

Figure 61. Forecasting error of FBProphet for DOGEUSDT

Figure 47 to Figure 61 again confirm that the performance of the model in forecasting the original price and its futures is similar, and the performance for more established coins are better.

4.3 XGBoost

4.3.1 BTCUSDT Futures

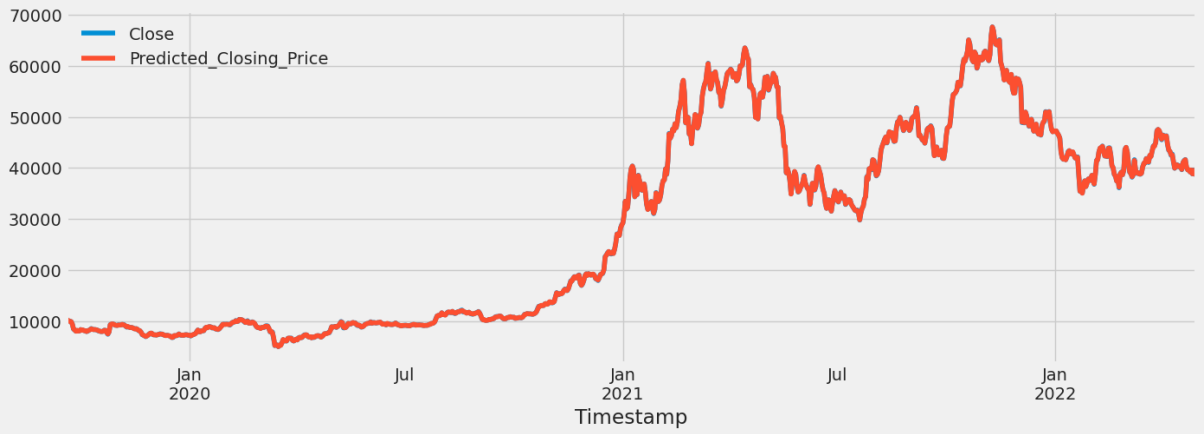

Figure 62. XGBoost fit on training data (BTCUSDT Futures)

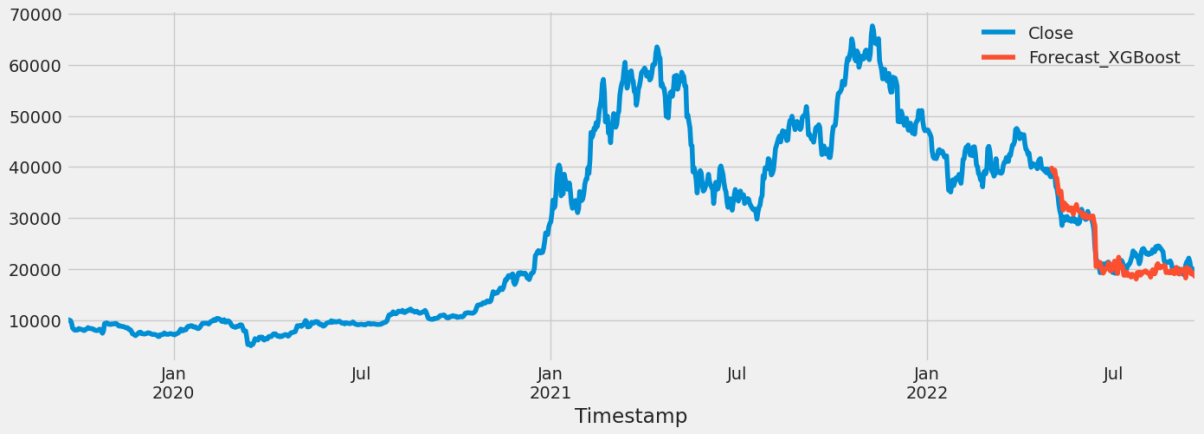

Figure 63. XGBoost fit on test data (BTCUSDT Futures)

Figure 62 shows almost a perfect fit with the training data, with less accuracy on test data in Figure 63, which may suggest that XGBoost is overfitting.

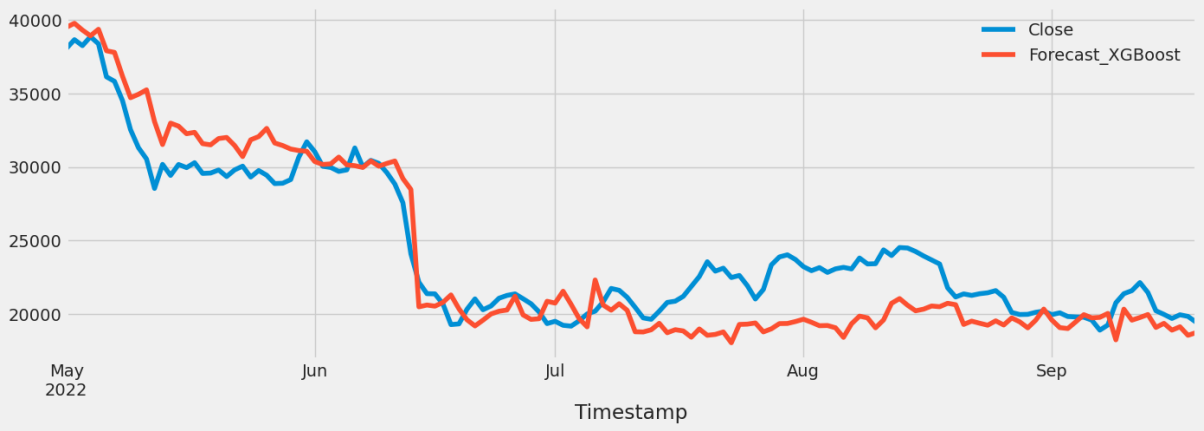

Figure 64. XGBoost performance (BTCUSDT Futures)

Unlike Facebook Prophet, the forecasting of XGBoost shown in Figure 64 is somewhat consistent over time, the deviation does not increase consistently.

```
train MAE : 31.288661172011505
train RMSE : 44.39888017299067
train R2 : 0.9999945752579981
```

Figure 65. Forecasting error of XGBoost for BTCUSDT Futures train set

```
test MAE : 1915.3616068808687
test RMSE : 2354.271251327138
test R2 : 0.7797664178285872
```

Figure 66. Forecasting error of XGBoost for BTCUSDT Futures test set

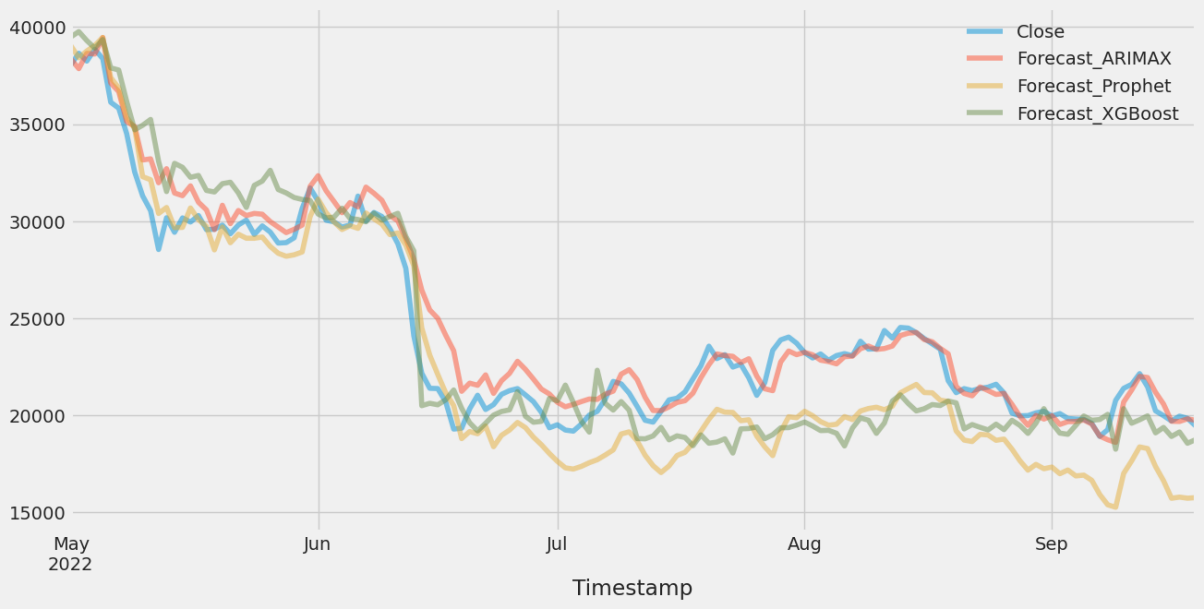

Figure 67. Models' performance summary (BTCUSDT Futures)

```
ARIMAX RMSE : 1240.8397540514538
FB Prophet RMSE : 2492.5280941053625
XGBoost RMSE : 2354.271251327138

ARIMAX MAE : 863.1415967208933
FB Prophet MAE : 2147.3615136862118
XGBoost MAE : 1915.3616068808687
```

Figure 68. Forecasting errors for BTCUSDT Futures overview

4.3.2 BTCUSDT

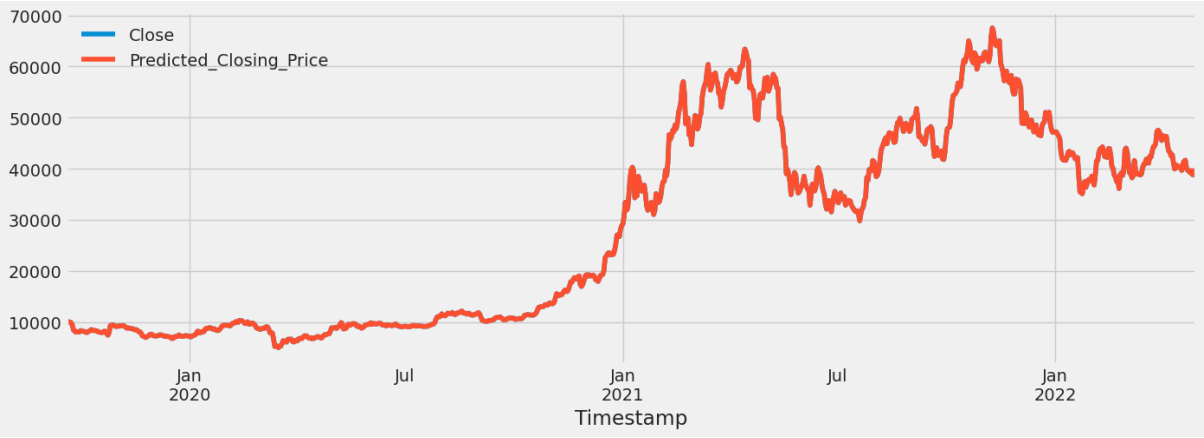

Figure 69. XGBoost fit on training data (BTCUSDT)

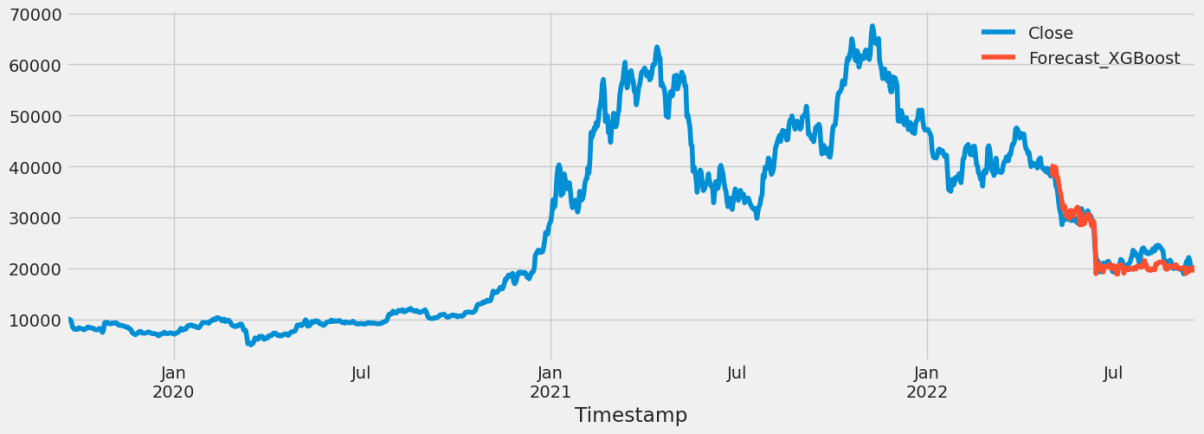

Figure 70. XGBoost fit on test data (BTCUSDT)

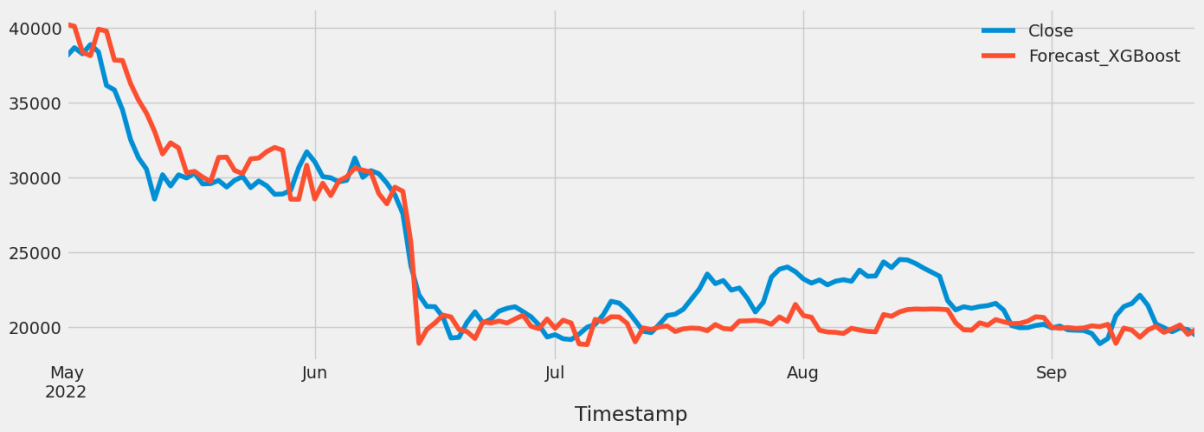

Figure 71. XGBoost performance (BTCUSDT)

```
train MAE : 0.21671519592088262
train RMSE : 0.29367244712568624
train R2 : 0.9999999997623914

test MAE : 1510.512524207747
test RMSE : 1921.2187167935701
test R2 : 0.8533871166116828
```

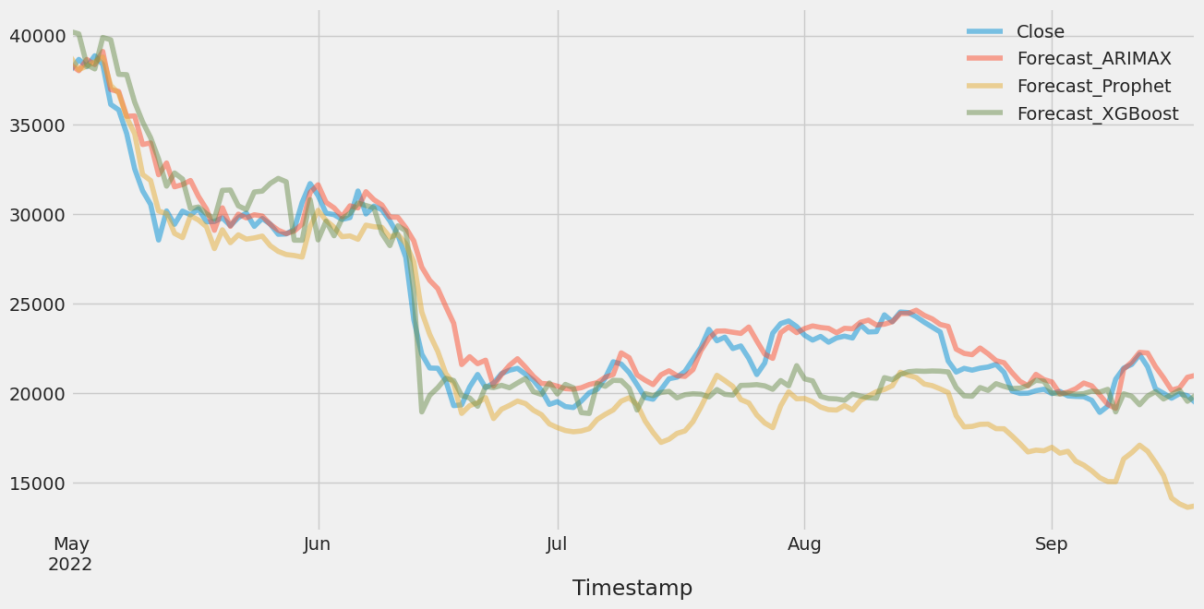

Figure 72. Models' performance summary (BTCUSDT)

```
ARIMAX RMSE : 1352.3872751129745
FB Prophet RMSE : 2822.562480095544
XGBoost RMSE : 1921.2187167935701

ARIMAX MAE : 904.3041012235992
FB Prophet MAE : 2408.600100526406
XGBoost MAE : 1510.512524207747
```

Figure 73. Forecasting errors for BTCUSDT overview

4.3.3 DOGEUSDT Futures

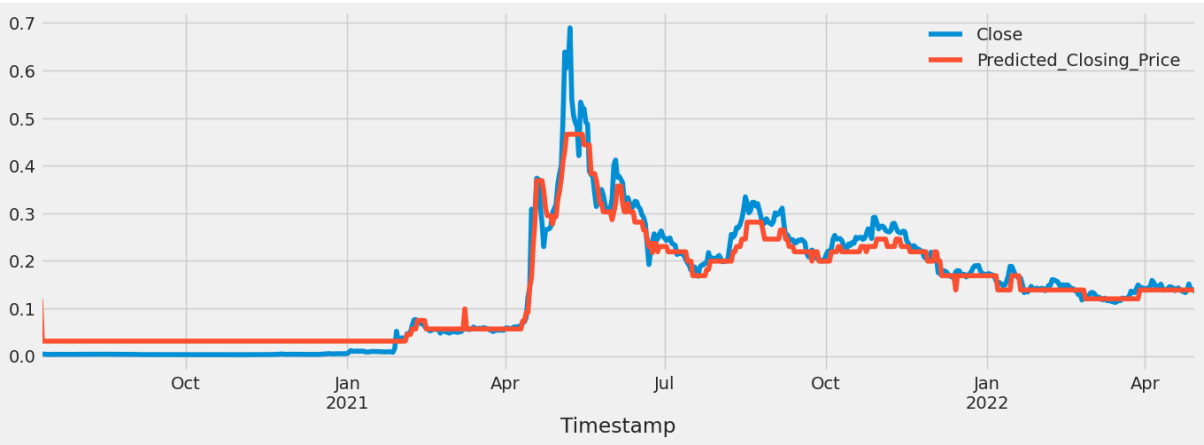

Figure 74. XGBoost fit on training data (DOGEUSDT Futures)

Figure 75. XGBoost fit on test data (DOGEUSDT Futures)

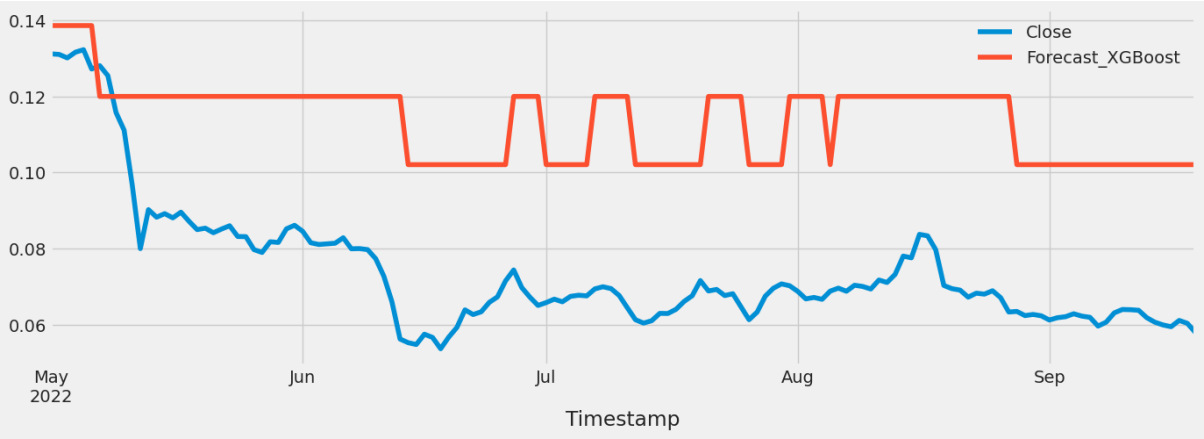

Figure 76. XGBoost performance (DOGEUSDT Futures)

```
train MAE : 0.020700046233969555
train RMSE : 0.029426251885234605
train R2 : 0.946027892674602


test MAE : 0.0396191978336332
test RMSE : 0.041202983771925196
test R2 : -4.8579700761906395
```

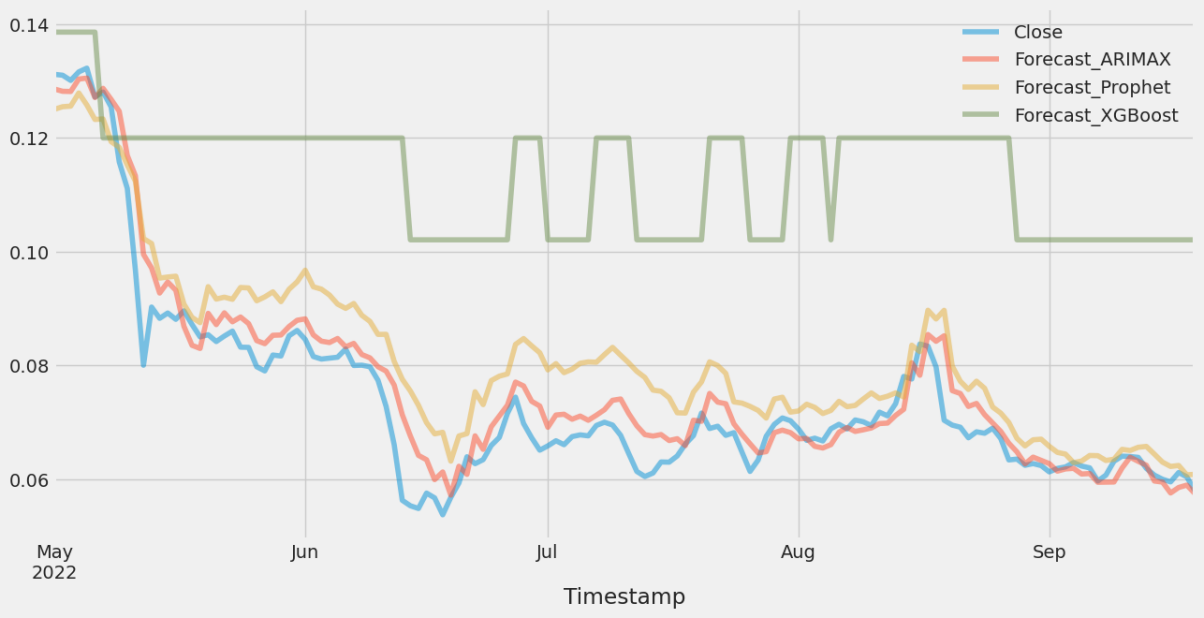

Figure 77. Models' performance summary (DOGEUSDT Futures)

```
ARIMAX RMSE : 0.004773340115544782
FB Prophet RMSE : 0.009304911999445954
XGBoost RMSE : 0.041202983771925196

ARIMAX MAE : 0.0035563223045747606
FB Prophet MAE : 0.007906665005330542
XGBoost MAE : 0.0396191978336332
```

Figure 78. Forecasting errors for DOGEUSDT Futures overview

4.3.4 DOGEUSDT

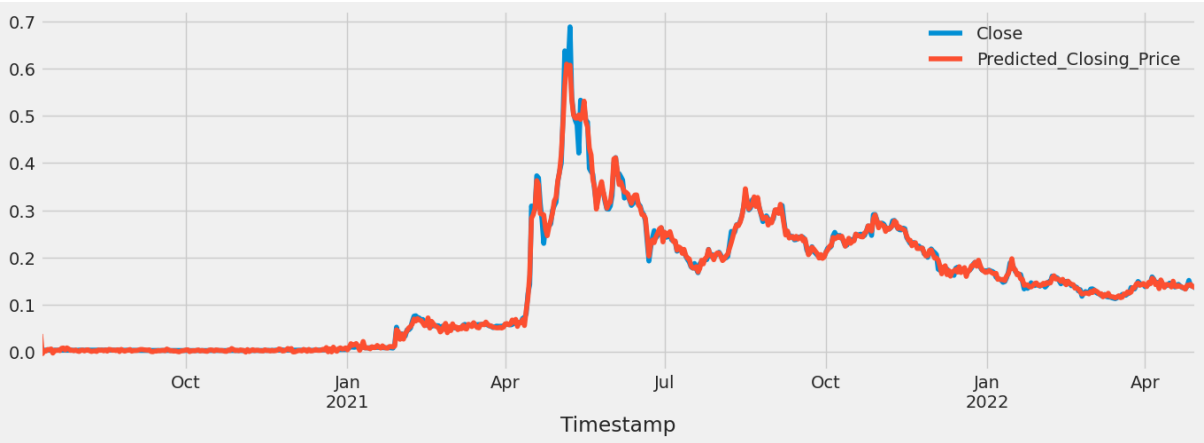

Figure 79. XGBoost fit on training data (DOGEUSDT)

Figure 80. XGBoost fit on test data (DOGEUSDT)

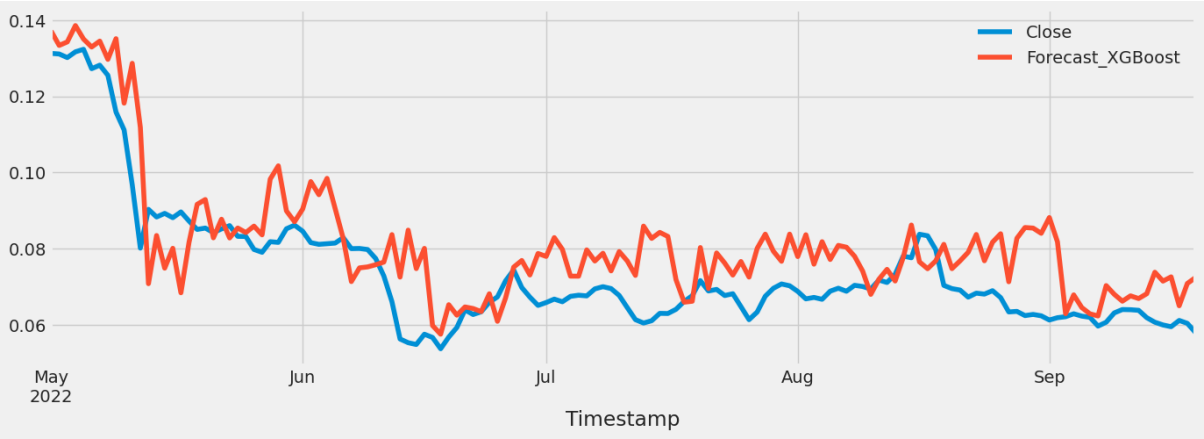

Figure 81. XGBoost performance (DOGEUSDT)

```
train MAE : 0.004312142917428315
train RMSE : 0.00846476350681644
train R2 : 0.9955290818231182


test MAE : 0.009378340304773736
test RMSE : 0.011784306979970545
test R2 : 0.5213779458333041
```

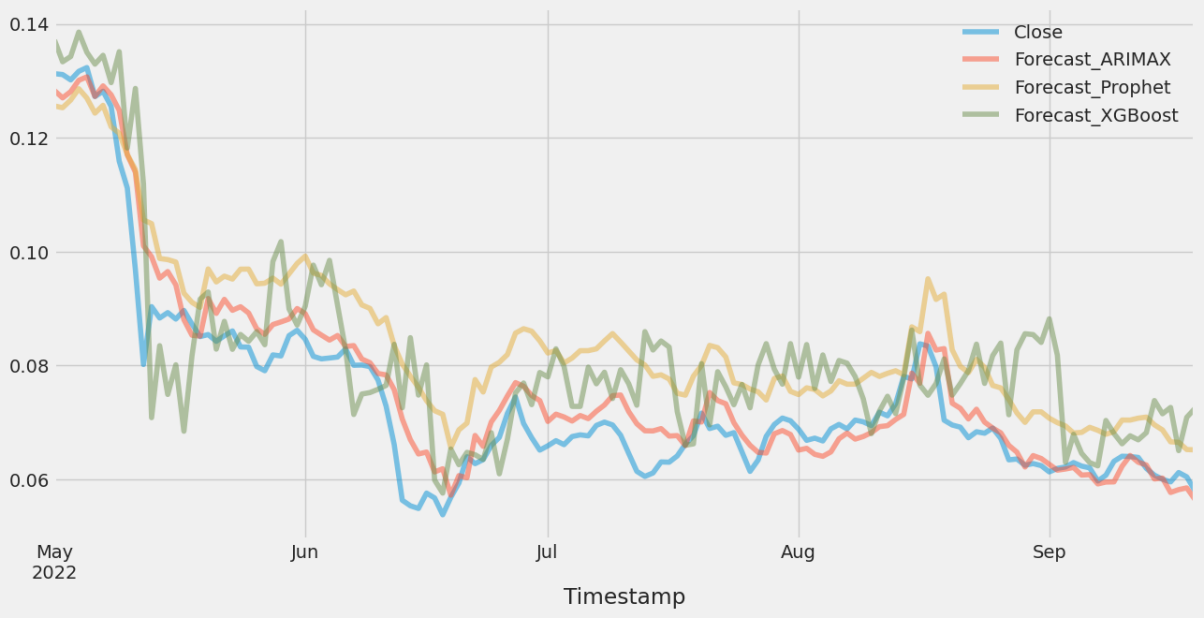

Figure 82. Models' performance summary (DOGEUSDT)

```
ARIMAX RMSE : 0.0050779340255159245
FB Prophet RMSE : 0.011808755621878358
XGBoost RMSE : 0.011784306979970545

ARIMAX MAE : 0.0038852259794653556
FB Prophet MAE : 0.010804889192175048
XGBoost MAE : 0.009378340304773736
```

Figure 83. Forecasting errors for DOGEUSDT overview

4.4 LSTM

4.4.1 BTCUSDT Futures

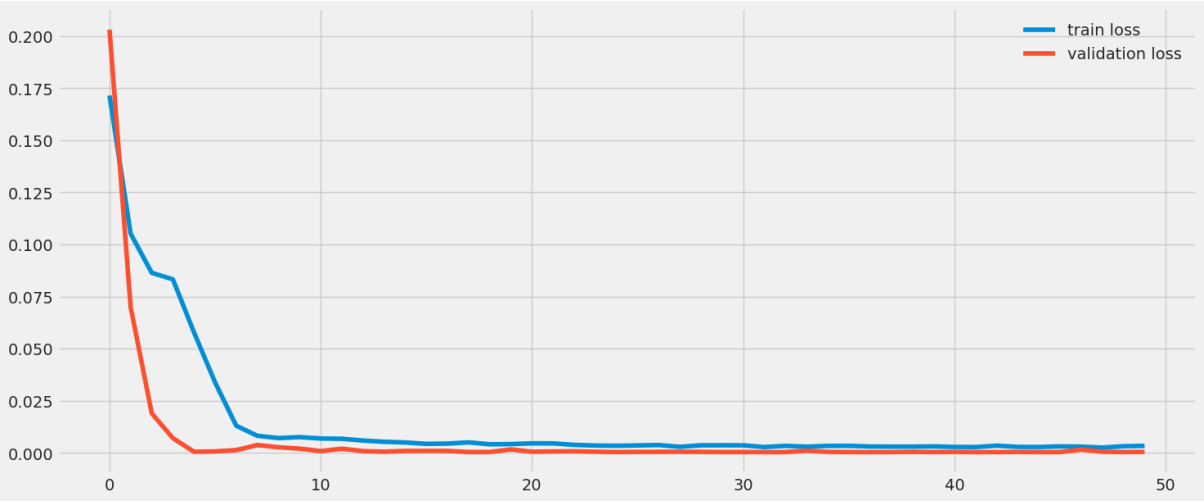

Figure 84. LSTM train and validation loss (BTCUSDT Futures)

A healthy decrease validation loss in Figure 84, although it has been hovering around the same level, which suggests that LSTM could achieve similar performance even if it was terminated earlier, saving training time if resources were constrained.

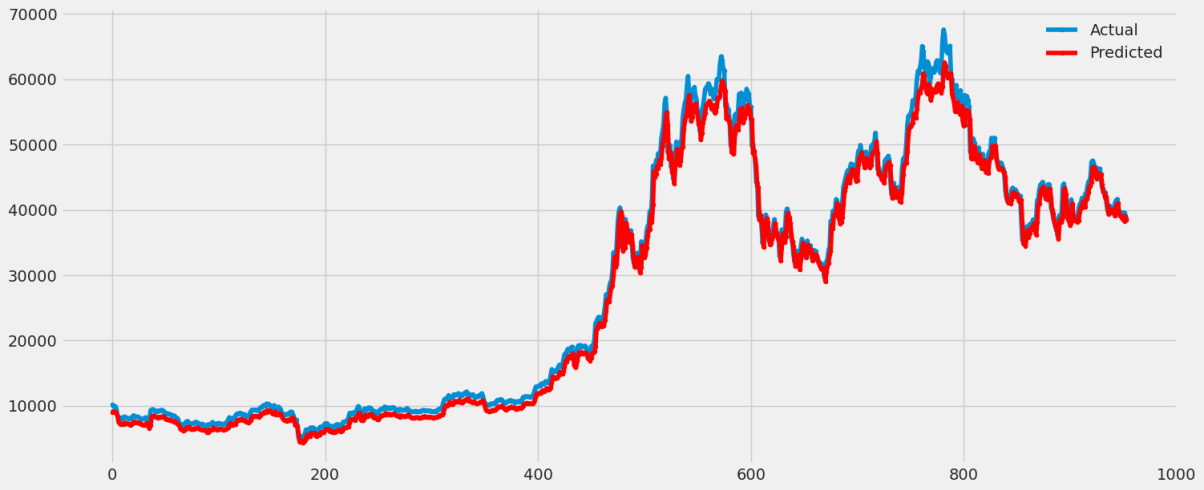

Figure 85. Actual against LSTM predicted on train data (BTCUSDT Futures)

In Figure 85, we do not notice the same overfitting as that observed for XGBoost on training data.

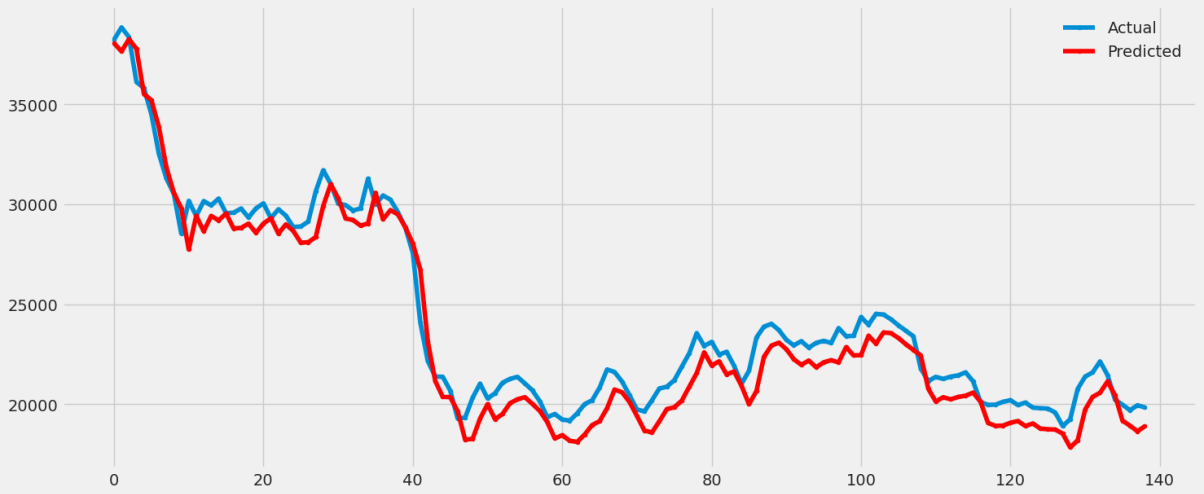

Figure 86. Actual against LSTM predicted on test data (BTCUSDT Futures)

```
Train RMSE: 1655.5487281166493
Train MAE: 1318.944064069597
Test RMSE: 1129.2503736602355
Test MAE: 961.6272435176859
```

Figure 87. train and test accuracy metrics for LSTM (BTCUSDT Futures)

4.4.2 BTCUSDT

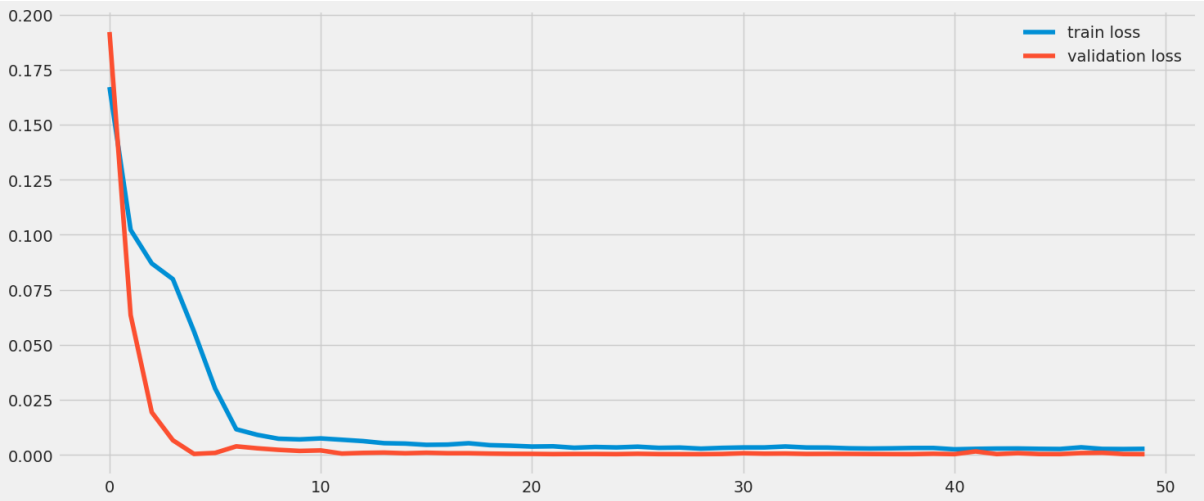

Figure 88. LSTM train and validation loss (BTCUSDT)

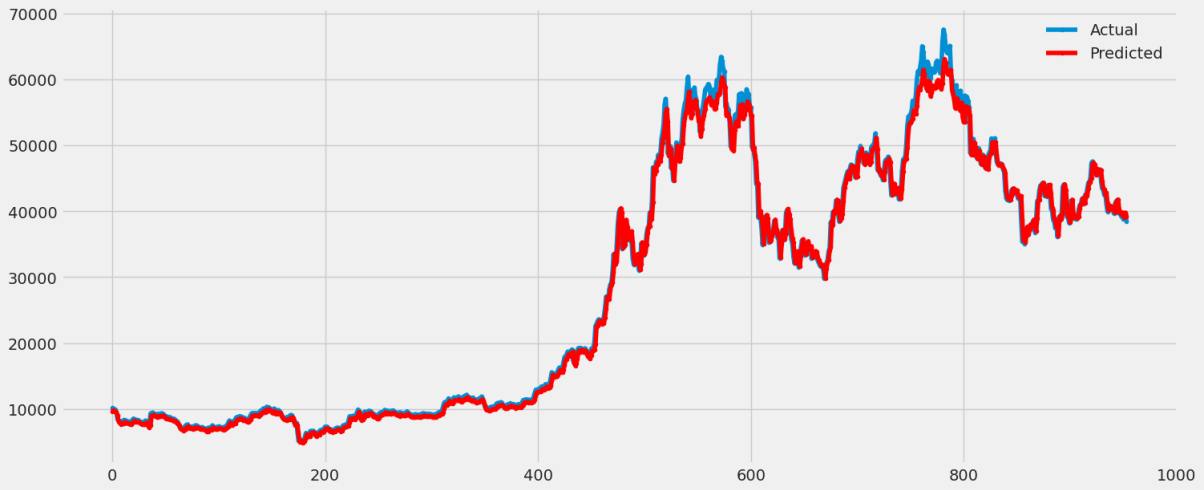

Figure 89. Actual against LSTM predicted on train data (BTCUSDT Futures

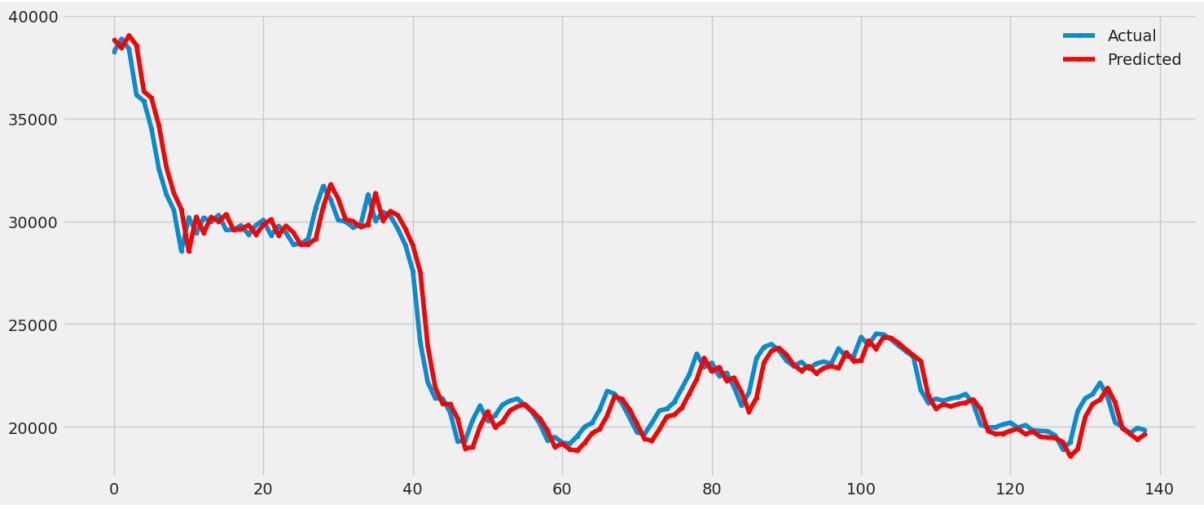

Figure 90. Actual against LSTM predicted on test data (BTCUSDT)

```
Train RMSE: 1311.4777245363293
Train MAE: 886.4743427435421
Test RMSE: 811.0817221396431
Test MAE: 608.8185654601322
```

Figure 91. train and test accuracy metrics for LSTM (BTCUSDT)

4.4.3 DOGEUSDT Futures

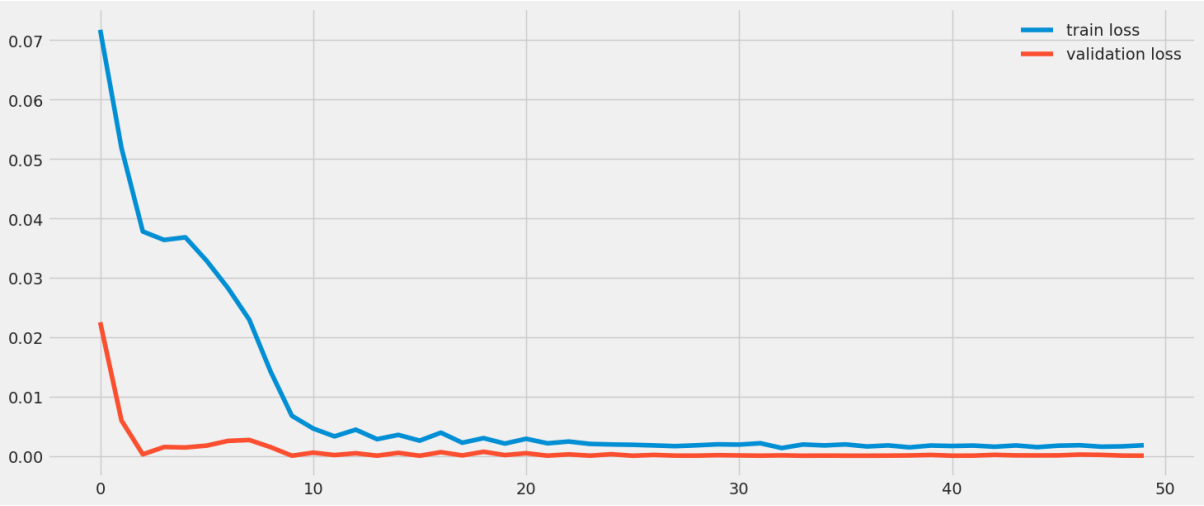

Figure 92. LSTM train and validation loss (DOGEUSDT Futures)

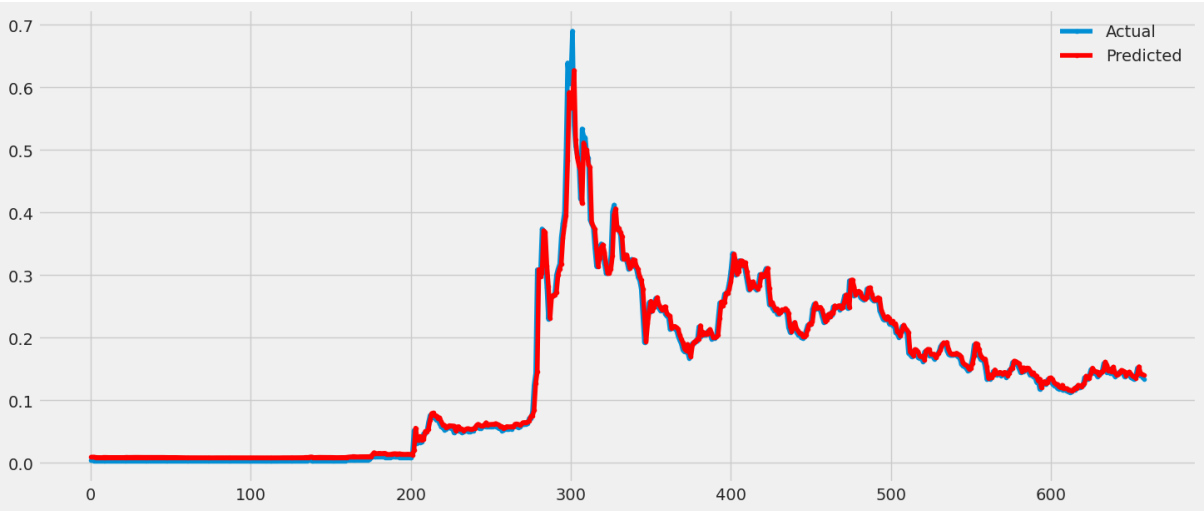

Figure 93. Actual against LSTM predicted on train data (DOGEUSDT Futures)

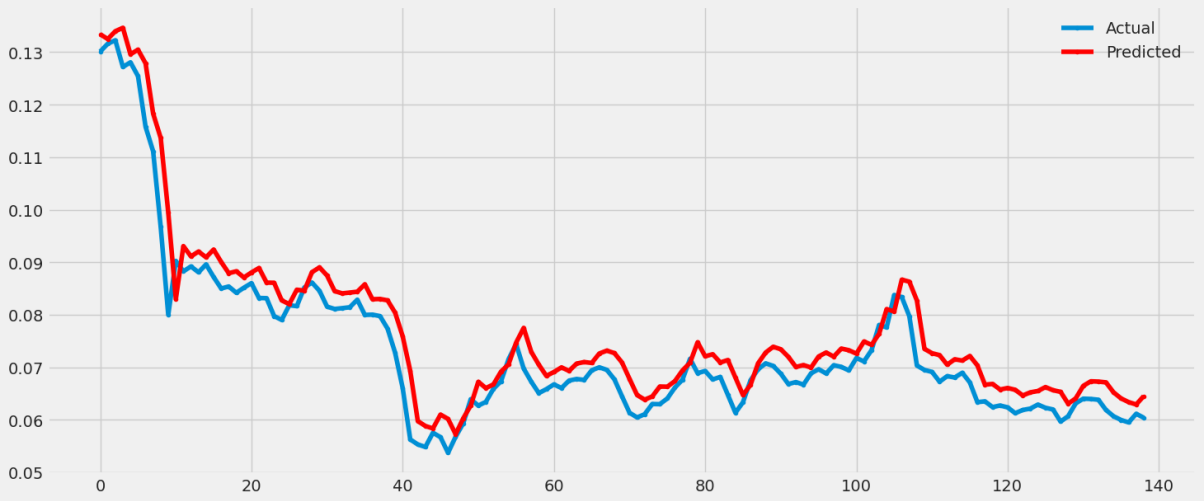

Figure 94. Actual against LSTM predicted on test data (DOGEUSDT Futures)

```
Train RMSE: 0.016407005552932997
Train MAE: 0.00822191847063534
Test RMSE: 0.0048169494824427915
Test MAE: 0.0038797777961312434
```

Figure 95. train and test accuracy metrics for LSTM (DOGEUSDT Futures)

4.4.4 DOGEUSDT

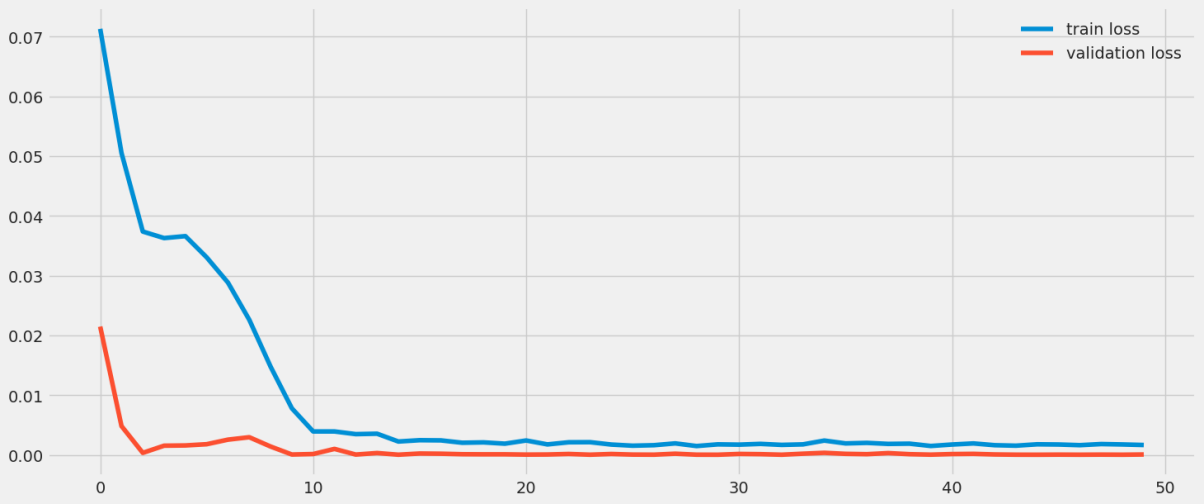

Figure 96. LSTM train and validation loss (DOGEUSDT)

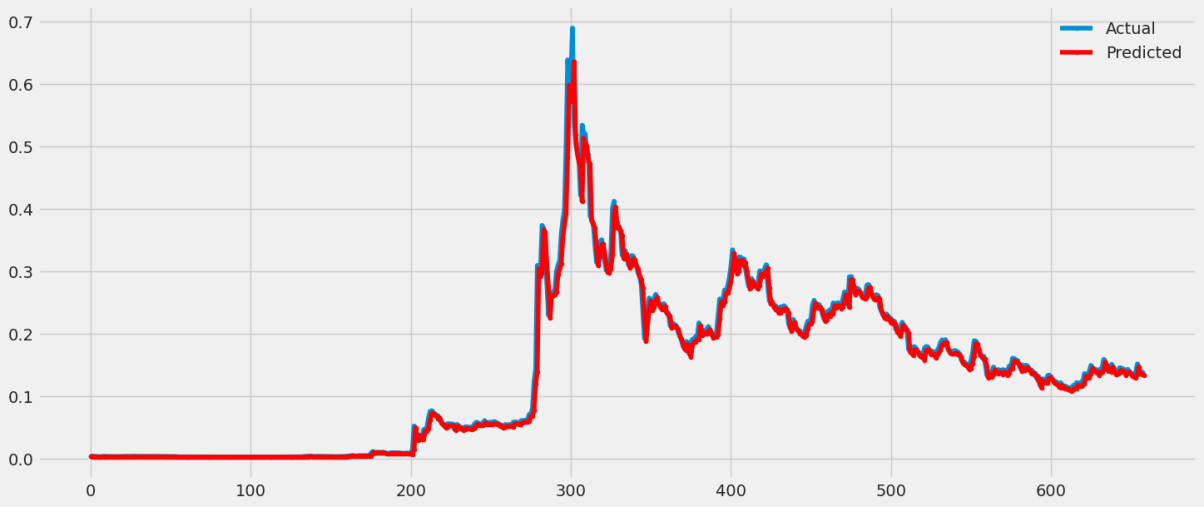

Figure 97. Actual against LSTM predicted on train data (DOGEUSDT)

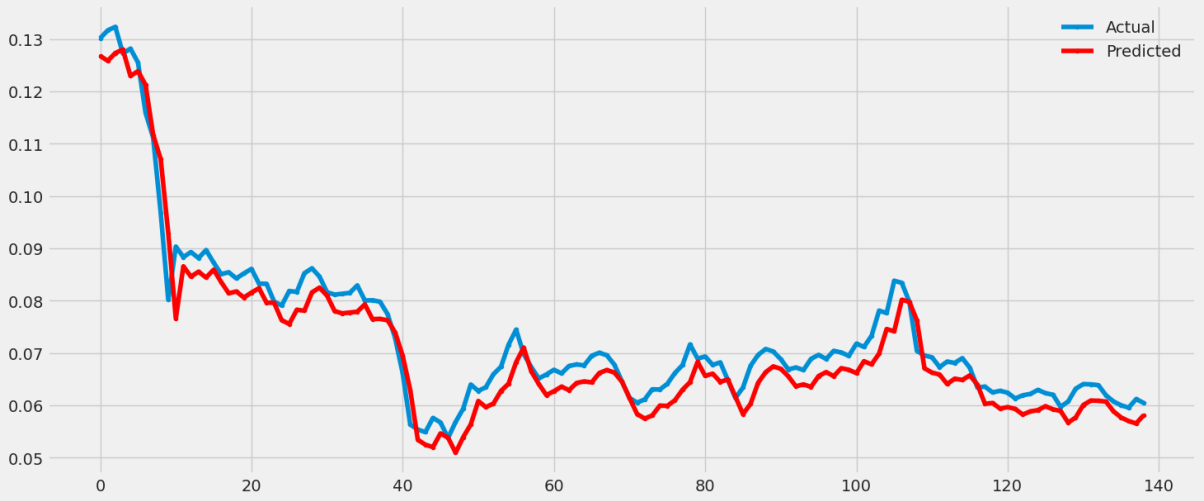

Figure 98. Actual against LSTM predicted on test data (DOGEUSDT)

```
Train RMSE: 0.01660305457278064
Train MAE: 0.007126108629491252
Test RMSE: 0.004237967220071239
Test MAE: 0.003558996021298768
```

Figure 99. train and test accuracy metrics for LSTM (DOGEUSDT)

From Figure 84 to Figure 99 we same the same patterns repeat as with previous model, albeit with slight variations, Bitcoin forecasting is more accurate and fitting than Dogecoin. However, the overall accuracy of LSTM is noticeably better than the other models.

4.5 Results Summary Tables

4.5.1 Root Mean Square Error (RMSE)

Table 1. RMSE metric summary table

	ARIMA	FBProphet	XGBoost	LSTM
BTCUSDT Futures	1240	2147.36	2354.27	1129
BTCUSDT	1352	2408.6	1921.218	811
DOGEUSDT Futures	.00477	.0079	.0412	.0048
DOGEUSDT	.00507	.0108	.0117	.004

4.5.2 Mean Absolute Error

Table 2. MAE metric summary table

	ARIMA	FBProphet	XGBoost	LSTM
BTCUSDT Futures	863.14	2492.52	1915.36	961.6
BTCUSDT	904.3	2822.56	1510.51	608
DOGEUSDT Futures	.00355	.0093	.0396	.0038
DOGEUSDT	.00388	.0118	.00937	.0035

5 Conclusions and recommendations

Corresponding to the problem definition mentioned earlier in the introduction, the first gap that this work is addressing is the scarcity of work being done on cryptocurrency futures despite its importance and high impact, mainly because it is relatively new. In addressing that, four models were developed, and their performance was measured.

The accuracy of forecasts is measured in this work by two metrics: RMSE and MAE. They are both useful and can reflect unique insights regarding the model's performance, for example, RMSE gives more weight to the points that are farther from the actual values, while MAE reflects the absolute difference between the predicted and the forecast.

For non-gaussian data, especially data with high volatility, looking at the MAE gives a more balanced assessment of the performance of the model, since it will not penalise residuals of high jumps like the RMSE. However, if the focus is on the model's ability to capture sudden jumps and volatility, then RMSE provides a more reliable view.

The RMSE and MAE tables can be interpreted from three angles:

4- RMSE vs MAE, which means focussing on capturing jumps and volatility (RMSE) vs focussing on the overall performance of the model (MAE)
5- Established coins (Bitcoin) vs Niche coins (Dogecoin)
6- Regular price vs Futures price

Based on those angles, we can observe the following:

- Overall, LSTM and ARIMA are performing the best on both metrics, and XGBoost and FBProphet performed worse consistently.
- Based on the RMSE, LSTM performed similar or better than ARIMA for all four timeseries data, which suggests that LSTM is the best at capturing jumps and volatilities quickly and minimise the gap between the forecast and the residual.
- Looking again at RMSE, LSTM forecast for the regular closing price is notably more accurate than the futures closing price, while the opposite is true for ARIMA.
- Based on the MAE, the same pattern on the regular vs futures prices persists: LSTM is performing better on the regular price than on the futures, while the opposite is true for ARIMA.

- Interestingly in MAE, ARIMA is performing better than LSTM in Bitcoin Futures and Dogecoin Futures, while LSTM outperformed on the regular price of coins.
- As we have seen while visualising and analysing the data, the charts for a coin and its futures are very similar with very similar trends, therefore this consistency of LSTM to perform better on the regular price needs an extended study to discover the subtle differences between a coin and its futures that lead to this pattern in modelling performance.
- It is worth noting that deep learning models are almost non-existent currently in cryptocurrency futures literature, the comparative performance presented in this work highly recommends utilising and exploring different forms of Artificial Neural Networks.
- Although LSTM mostly scored the best accuracy, that does not mean that other more classical models should be neglected. Prophet model, despite being the least accurate, has provided important insights and analysis such as change points in trends, it also allows developers to specify certain heuristics for the model, these advantages are not easily replicated in classical ARIMA and would require more time and expertise. So, the models should be utilised depending on who and why are they needed, if only for accuracy, then neural networks are the most efficient, if analysis is equally or more important than other models are worth exploring.
- All the models have performed better on well-established coins (Bitcoin) than other Niche coins (DOGECOIN) based on the actual against forecast plots. For this conclusion to be confirmed, the study should be extended to focus on classifying the coins based on a specific set of measures (such as the project it is trying to implement, the team developing the coin, the whitepaper of the coin, the speed of transactions, etc..), then forecasting each coin and comparing the accuracy to the rank of the model.

The other gap this work is addressing is relating the regular price of the coin to its futures price, the following were concluded based on results:

- The high-level analysis of the data itself did not show any noticeable discrepancies between the regular and the futures price, or at least no noticeable trends, this could explain the scarcity of research on futures.
- Nevertheless, the futures prices are subtly different from the regular as confirmed by LSTM's performance, and if we compare cryptocurrencies to other commodities such

as stocks, a lot of research has been done to relate the actual prices and the futures prices, which this both impacts and is impacted by economic stability.

- An extended work to address this gap would be to dive deeper into the financial side of cryptocurrency futures and analyse different maturities for the same coin, while simultaneously comparing them to the regular price, this ensures that results could capture the subtle insights from futures.

References

A. Leijon, Z.M. and Kleijn, W.B. 2013. Vector quantization of LSF parameters with a mixture of Dirichlet distributions. *IEEE Trans. Audio, Speech, Language Process* (9), pp. 1770–1790.

A. Witt, J.K. and Pikovsky, A. 1998. Testing stationarity in time series.

BAJPAI, P. and RASURE, E. 2022. Cryptocurrency Futures. Available at: https://www.investopedia.com/articles/investing/012215/how-invest-bitcoin-exchange-futures.asp.

Banfieldand, J.D. and Raftery, A.E. 2004. Model-based Gaussian and Non-Gaussian clustering. In: *Biometrics*., pp. 803–821.

Bishop C. M. 2006. Pattern Recognition and Machine Learning. *Journal of Information Science and Statistics. Springer.*

Bishop, S.A. et al. 2020. Statistical analysis of childhood and early adolescent externalizing behaviors in a middle low-income country.

Blei, D.M. 2004. *Probabilistic Models of Text and Images*. University of California, Berkeley.

CFI 2022. Cryptocurrency Exchanges. Available at: https://corporatefinanceinstitute.com/resources/knowledge/other/cryptocurrency-exchanges/.

CHEN, G. et al. 1997. Parametric and non-parametric modelling of time series - an empirical study. 8(1), pp. 63–74.

CME 2022. BITCOIN FUTURES - CONTRACT SPECS. Available at: https://www.cmegroup.com/markets/cryptocurrencies/bitcoin/bitcoin.contractSpecs.html.

Coindesk 2022. What is Bitcoin. Available at: https://www.coindesk.com/learn/what-is-bitcoin/.

CoinMarketCap. 2022. Available at: https://coinmarketcap.com/.

Fitzmaurice, G. et al. 2008. Longitudinal Data Analysis.

Gurka, M.J. et al. 2006. Extending the BoxCox transformation to the linear mixed model. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* , pp. 273–288.

Holub, M. and Johnson, J. 2018. Bitcoin research across disciplines. *Information Society* 34(2),

pp. 114–126. Available at: https://doi.org/10.1080/01972243.2017.1414094.

Hu, C.A. et al. 2020. Using a machine learning approach to predict mortality in critically ill influenza patients: A cross-sectional retrospective multicentre study in Taiwan. *BMJ Open* 10(2), pp. 1–10. doi: 10.1136/bmjopen-2019-033898.

Hyndman, R.J. and Athanasopoulos, G. 2018. Stationarity and differencing. In: *Forecasting: Principles and Practice*.

Inc., A. 2022. Data science technology for groundbreaking research.a competitive edge.a better world. Available at: https://www.anaconda.com/.

Jordan, M.I. and Mitchell, T.M. 2015. Machine learning: Trends, perspectives, and prospects. *Science* 349(6245), pp. 255–260. doi: 10.1126/science.aaa8415.

Kimoto, T. et al. 1990. Stock market prediction system with modular neural networks., pp. 1–6. doi: 10.1109/ijcnn.1990.137535.

Krollner, B. et al. 2010. Financial time series forecasting with machine learning techniques: A survey. *European Symposium on Artificial Neural Networks: Computational and Machine Learning*

Kuppusamy, K.S. 2018. Keras: Building Deep Learning Applications with High Levels of Abstraction. Available at: https://www.opensourceforu.com/2018/01/keras-building-deep-learning-applications-with-high-levels-of-abstraction/.

Lipsitz, S.R. et al. 2000. Using a box-cox transformation in the analysis of longitudinal data with incomplete responses. *Journal of the Royal Statistical Society Series C* , pp. 287–296.

Luo, J. et al. 2021. Time series prediction of COVID-19 transmission in America using LSTM and XGBoost algorithms. *Results in Physics* 27, p. 104462. Available at: https://doi.org/10.1016/j.rinp.2021.104462.

Marco Iansiti and Lakhani, K.R. 2017. The Truth About Blockchain. Available at: https://hbr.org/2017/01/the-truth-about-blockchain.

Meir, R. 2000. Nonparametric time series prediction through adaptive model selection.

Meta 2022. Facebook Research. Available at: https://research.facebook.com/research-areas/machine-learning/.

Mills, T.C. 2011. *The foundations of modern time series analysis*.

Morris, D.Z. 2016. Leaderless, Blockchain-Based Venture Capital Fund Raises $100 Million, And Counting. Available at: https://fortune.com/2016/05/15/leaderless-blockchain-vc-fund/.

Oguntunde, P.. et al. 2018. Data analysis on physical and mechanical properties of cassava pellets., pp. 286–302.

Olah, C. 2015. Understanding lstm networks – colah's blog.

Osaka-Exchange 2014. Dojima Rice Exchange - World's First Futures Exchange -. Available at: https://www.jpx.co.jp/dojima/en/index.html.

Palit, A.K. and Popovic, D. 2006. *Computational Intelligence in Time Series Forecasting*.

Popper, N. 2016. A Venture Fund With Plenty of Virtual Capital, but No Capitalist. *Online* . Available at: https://www.nytimes.com/2016/05/22/business/dealbook/crypto-ether-bitcoin-currency.html.

R2R2 2016. Understanding, Deriving and Extending the LSTM.

R2rt 2016. Written memories: Understanding, deriving and extending the lstm. Available at: https://r2rt.com/written-memories-understandingderiving-%0Aand-extending-the-lstm.html.

Rasmussen, C.E. 2006. Gaussian processes for machine learning. *MIT Press*

Schmidhuber, J. 1993. *network architectures, objective functions, and chain rule*. Institut fur Informatik, Technische Universitat Munchen.

Sharma, S. 2017. Artificial Neural Network (ANN) in Machine Learning. Available at: https://www.datasciencecentral.com/artificial-neural-network-ann-in-machine-learning/.

Shumway, R.H. and Stoffer, D.S. 2011. Time series analysis and its applications. *Springer*

Smith, T.G. 2022. pmdarima 2.0.1. Available at: https://pypi.org/project/pmdarima/.

Steel, A.D. 2014. *Predictions in financial time series data*. Institute of Technology Blanchardstown.

Stoica, P. and Moses, R. 1997. *Introduction to spectral analysis*.

Sun, J. et al. 2008. Heavy-tailed longitudinal data modeling using copulas. In: *Insurance: Mathematics and Economics*., pp. 817–830.

Taylor, S.J. and Letham, B. 2017. Business Time Series Forecasting at Scale. *PeerJ Preprints 5:e3190v2* 35(8), pp. 48–90. Available at: https://peerj.com/preprints/3190/%0Ahttp://ezproxy.bangor.ac.uk/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=c8h&AN=108935824&site=ehost-live%0Ahttps://peerj.com/preprints/3190/%0Ahttps://peerj.com/preprints/3190.pdf.

Wei, W.W.S. 2013. Time Series Analysis. In: *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*.

Xu, Z. 2014. *Modeling Non-Gaussian Time-correlated Data Using Nonparametric Bayesian Method*. Ohio State University.

Zhai, J. et al. 2018. Data analytic approach for manipulation detection in stock market. *Review of Quantitative Finance and Accounting* 50(3), pp. 897–932. doi: 10.1007/s11156-017-0650-0.

Zheng, Y. et al. 2020. A Learning-Based Model to Evaluate Hospitalization Priority in COVID-19 Pandemics. *Patterns* 1(6), p. 100092. Available at: https://doi.org/10.1016/j.patter.2020.100092.