UNIVERSITY OF
WESTMINSTER
SCHOOL OF
COMPUTER SCIENCE & ENGINEERING

Portfolio Construction for Cryptocurrency using Machine Learning

By

MAYUR CHAKALASIYA

(W1872798)

Supervised by

Barnabas Gatsheni

Submitted in partial fulfilment of the requirements of
the School of Computer Science & Engineering
of the University of Westminster
for award of the Master of Science

JANUARY 2023

DECLARATION

I, *Mayur Vasharambhai Chakalasiya*, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This has not previously been accepted as part of any other degree submission.

Signed: Mayur V. Chakalasiya

Date: 08-01-2023

FORM OF CONSENT

I, *Mayur Vasharambhai Chakalasiya*, hereby consent that this Project, submitted in partial fulfilment of the requirements for the award of the MSc degree, if successful, may be made available in paper or electronic format for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Mayur V. Chakalasiya

Date: 08-01-2023

ABSTRACT

In quantitative finance, the investment portfolio is the process of asset allocation to minimize the risk and maximize the profit for investors. Price forecasting is a critical prerequisite to constructing a portfolio for cryptocurrencies. This project provides a framework for price prediction and portfolio construction using four different portfolio methods.

The Long-Term Short Memory (LSTM) model is used to predict the price of crypto assets. The construct portfolios are constructed using four portfolios method: *Mean-Variance Portfolio Method, Hierarchical Risk Parity Method, Kelly's Criteria, and Equal-Weight Portfolio*. The performance of the portfolios is analyzed using eight measures.

The LSTM model is built for three Intraday datasets of five cryptocurrencies with optimized hyperparameters after hyperparameter tuning. The LSTM model has predicated the accurate price with minimum RMSE values for a 10-min dataset of all crypto assets. With predicted values, the portfolios are constructed. The most risk-averse portfolio is the Hierarchical portfolio, while the equal-weight portfolio is the riskiest portfolio for given datasets.

The Multivariant LSTM model's performance is increased after hyperparameter tuning and technical indicators (Simple Moving Average 50 days and 100 days) are selected as input features. The Hierarchical portfolio adjusts risk more than other portfolios. The result of the study can contribute to developing the application to construct and optimize the portfolio and guide investors to make risk-free decisions for investment. The application of Multivariant LSMT model can be explored for different business scenarios for time series data from finance sector.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to Professor Barnabas Gatsheni, my project supervisor, for his direction and assistance during the project's completion. I'd also like to take this chance to thank my family, friends, and parents for their unwavering encouragement and support while I finished the course.

Table of Contents

Word Count:

Chapter 1 Introduction

One of the most important recent innovations, cryptocurrency can be compared to the internet boom in the financial industry. It is a well-known form of virtual digital currency that is used as an investment tool and an alternative to institution-regulated traditional money in online transactions. In December 2022, bitcoin had a market valuation of around $322 billion, leading the cryptocurrency market home to other well-known crypto assets including Ethereum, Binance, and Litecoin. Despite the significant risk associated with crypto assets, it has drawn in individual investors and businesses like Tesla and PayPal to invest due to the possibility of huge profits.

In quantitative finance, investment portfolio construction is all about finding the appropriate proposition of the assets to strike the right balance for expected returns against the risk of the portfolio. The most critical process is price forecasting to build an optimal portfolio. The price prediction problem of cryptocurrency is considered the time series prediction problem for stocks. Machine learning models can be employed to solve the prediction problem. Researchers have made several attempts to build a forecasting model using machine learning techniques. D and E, (2019 worked) on bitcoin price prediction using the traditional statistical model Autoregressive Integrated Moving Average (ARIMA). Nevertheless, the ARIMA model is not able to outperform the deep learning model and is unable to capture the non-linear pattern of the price prediction problem (Siami-Namini, Tavakoli and Siami Namin, 2018). The field of computer science and machine learning is constantly evolving and there is scope for improvement in the price prediction of cryptocurrency with high accuracy to construct an effective portfolio. This research study covers the price prediction of the cryptocurrency and builds an optimized portfolio using forecasted prices by using advance machine learning models to achieve high prediction accuracy. The RMSE values of the multivariant LSTM model are lowest for bitcoin (0.0043), Ethereum (0.0053), Binance (0.0055), tether (0.0036), and USD coin (0.0059) for proposed LSTM. The performance of the Multivariant LSTM model has been enhanced with the optimized hyperparameters and technical indicators (Simple Moving Average) as input features to the multivariant LSTM model.

1.1 Background

Following the 2008 global financial crisis, an unknown individual, group, or organization known as 'Satoshi Nakamoto' established an electronic peer-to-peer system based on the cryptocurrency bitcoin (Nakamoto, 2008). Bitcoin is a decentralized digital money that was established in 2008 and first used in 2009. It arose in response to financial firms that frequently privatized profits while passing losses to customers. Cryptocurrencies can solve various difficulties that have been passed down from previous financial systems, including a

lack of confidence, transaction inefficiencies, and volatility. According to CoinMarketCap, 200,066 coins have entered into circulation with a market capitalization of more than $808 billion till December 2022. The technology driving the spread of cryptocurrencies is blockchain which is defined by Treible Maier (2018) as "a digital, decentralized, and distributed ledger in which transactions are documented and appended in chronological order to establish permanent and tamper-proof records."

Cryptocurrency makes use of blockchain technology to facilitate transactions and self-run decentralized platforms. Because blockchain is a distributed ledger system, when someone updates it, other nodes or computers are alerted. Figure 1.1 depicts the sequence of a bitcoin transaction from user A to B.

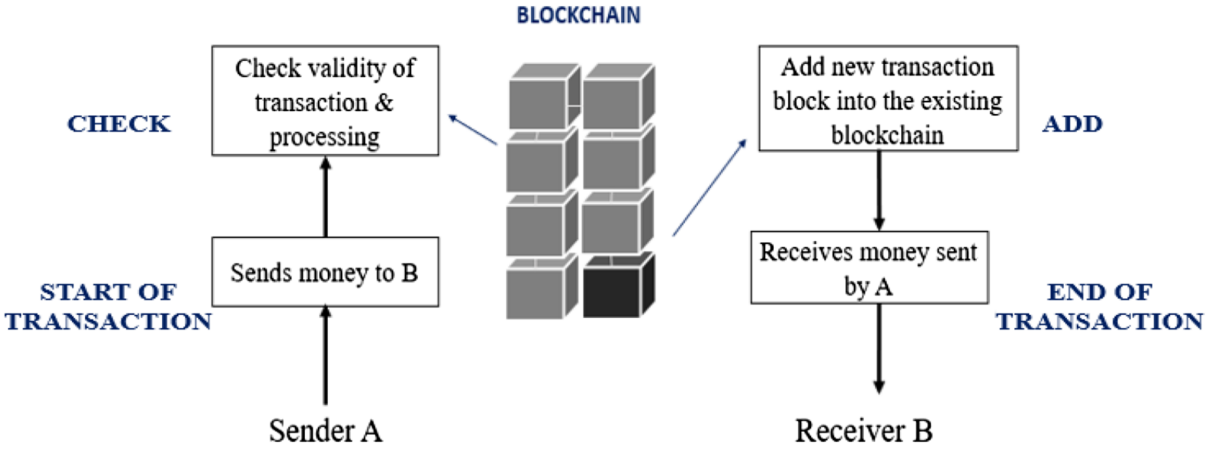


Figure-1.1: Shows a bitcoin transaction from user A to user B Adapted from: (Malik1# and Rana2, n.d.)

Figure-1.1: Shows a bitcoin transaction from user A to user B ***Adapted from: (Malik1# and Rana2, n.d.)***

If Sender A wishes to conduct a transaction with Recipient B, A will begin the transaction and record it on the public ledger, which is held by other cryptocurrency users. The transaction is placed on the blockchain when it has been verified. Miners verify transactions before they are recorded in the public ledger. Miners are compensated with cryptocurrency for verifying and maintaining the blockchain.

Because of the following factors, blockchain technology makes cryptocurrencies more dependable.

1. Transaction encryption techniques prevent identity theft and make entities not hackable.
2. It makes use of the internet to make payments and money transactions quickly.
3. It is a type of virtual bank that can be accessed from anywhere, at any time, and users control the bank.
4. There is no intervention from a third party such as the government or a bank.

Cryptocurrencies have substantially faster settlement times than typical payment options. In the case of bitcoin, the average settlement time is 10 minutes, which is far faster than any non-cash financial transaction, which may take days or weeks. Although cryptocurrencies were established to facilitate the exchange of goods and services, Bitcoin, according to ARK and Coinbase, is the first of its kind in what is quickly becoming a unique asset class. Unlike bonds and stocks, cryptocurrencies may be prone to a "winner takes all" scenario because of the enormous network effects of users and developers. Cryptocurrencies that properly support the flywheel of user and developer interaction have the potential to expand to enormous market capitalizations. (Burniske & White, 2017).

1.2 Problem Statement

Baur, Hong, and Lee (2018) presented **work** based on a bitcoin analysis demonstrating that virtual currency's return attributes differ from traditional instruments, but they provide significant diversity. Cryptocurrencies are volatile assets, and various internal and external variables influence the price of cryptocurrencies. For example, demand-supply, crypto market, macro-economic, and politics (Khedr et al., 2021). Portfolio construction is a crucial process that maximizes the return and minimizes the risk to invest in cryptocurrencies. The efficiency of the portfolio construction/optimization can be increased by changing the input forecast from historical data to a reliable one (DeMiguel, Garlappi and Uppal, 2007). There are several issues with existing statistical models namely completeness of data, identifying the only linear relationship, and univariate data (Ramkumar, G., 2021). This research study addressed the efficient price prediction problem to construct a portfolio of cryptocurrencies. The performance of portfolios is analysed to identify the portfolio construction methods to build risk-free portfolios for the investor.

The key deliverables are captured by the set of objectives in Section 1.3.

1.3 Objectives

1. Data collection of cryptocurrencies from Yahoo Finance, the Bloomberg terminal and exploratory analysis of data.
2. Forecasting the price of time series data.
3. Portfolio construction using several portfolio construction methods
4. Comparison and analysis of the performance of resultant portfolios

1.4 Dissertation Structure

The structure of the report is divided into six sections: Introduction, literature review, Methodology, Experiment, Discussion, and Conclusion, as per the flowchart illustrated in Figure 1.2.

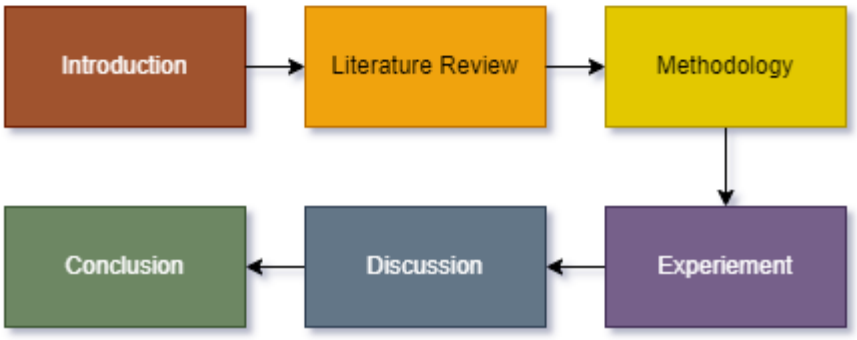


Figure-1.2: Dissertation Structure

The introduction provides context for this project by covering the background, issue description, and objectives, as well as details about cryptocurrencies and portfolio construction. Following that, the Literature review part describes the current literature and research in the field of the selected project themes and identifies gaps. The methodologies and methods are discussed and compared to finalize the approaches in the methodology section. The experiment has been carried out using the procedures described in the previous part, and the results are reviewed in the experiment section. The results of the experiment are discussed and concluded in the last chapter.

Chapter 2 Literature Review

In this section, the existing literature and various machine learning models used for portfolio construction will be discussed.

Most portfolio creation processes, including idea generation, asset allocation, weight optimization, position size, and strategy testing, can be aided by machine learning. It can be categorized as below:

1) **Trading strategies** are a systematic way of buying and selling assets in the market which can be further classified into three different themes (Snow, 2020):

    a) **Price** plays a central role in input data and predicted outcomes of trading strategies. Price techniques include technical, systematic global macro, and statistical arbitrage.

    b) **Events** strategies aid in the prediction of changes such as trends, and soft and hard events.

    c) **The value** comprises risk parity and factor investing, which determine intermediary values that are not directly tied to asset prices.

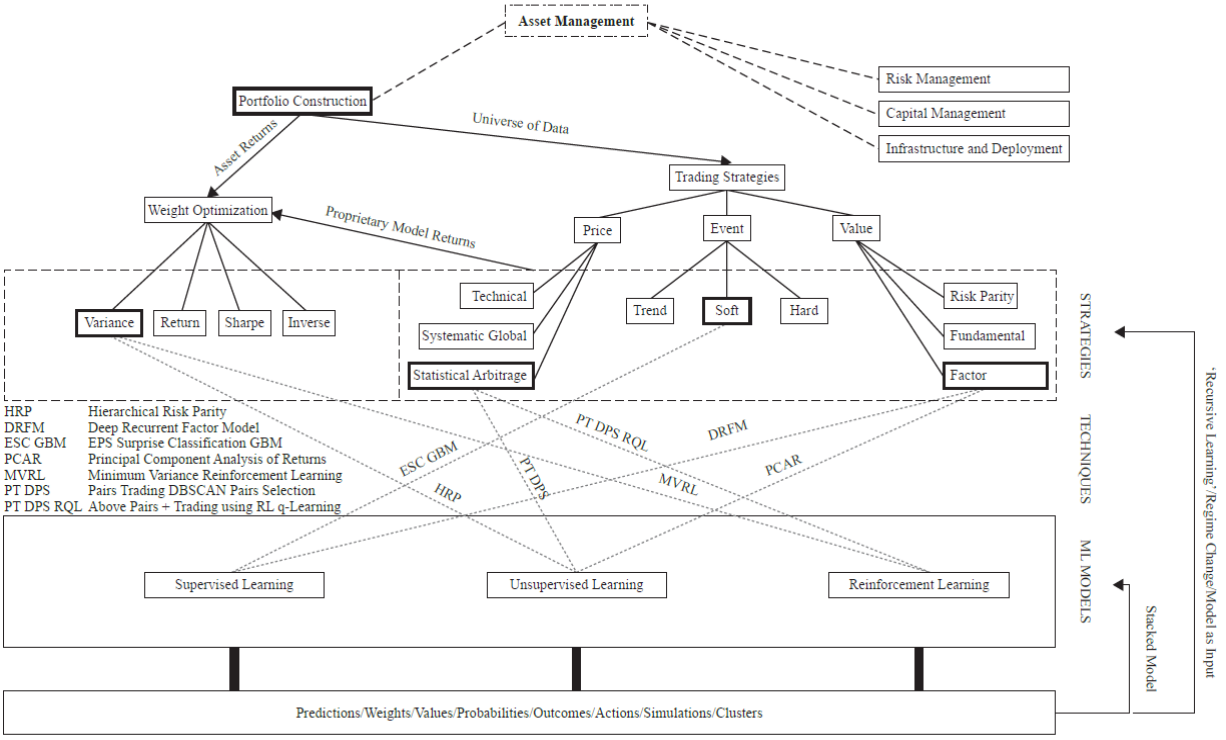


Figure-2.1 Asset Management and Finance Machine learning ***Adapted from:*** *(Snow, 2020)*

2) **Weight optimization** is the process of determining the optimal portfolio from among all portfolios under consideration, based on some objective. When dealing with neural networks, we frequently need to decide which optimization technique will yield faster and better updates to the network's weight and bias parameters. Similarly, current portfolio optimization has shifted away from Markowitz's mean-variance (MV) portfolio

based on historical returns and toward dynamic models based on cutting-edge reinforcement learning techniques (Snow, 2020).

2.1 Understanding of Portfolio Construction Problem

The portfolio construction process aims to maximize the return and minimize the investment risk. If there is an N asset involved in the portfolio, the Portfolio management problem can be defined using below mathematical formula (Yun et al., 2020):

$$W = \underset{W=\{w_1,\ldots,w_N\}}{\operatorname{argmax}} \left\{ \sum_k w_k E(r_k) - \lambda \sigma \left(\sum_k w_k E(r_k) \right) \right\}, \quad \textbf{(2.1)}$$

$$subject\ to\ \sum_k w_k = 1\ and\ 0 \le w_k \le 1,$$

Where,

W_k: portfolio weight for the k^{th} asset
rk: the return of a kth asset
k ∈ {1, …, N}
σ: a method to evaluate the risk
λ: a weight parameter for the risk value
$E(r_k)$: the expected return of the kth asset.

Prediction of the expected return of the k^{th} asset plays a crucial role in order to solve the portfolio management problem.

Traditionally, the $E(r_k)$ is calculated using a statistical procedure that takes a weighted average of the asset's historical returns over a certain time period from t to T.

$$E_{stat}(r_k) = \frac{1}{\sum_i \omega_i} \sum_i \omega_i r_{k,i} \quad \textbf{(2.2)}$$

Where,

ω_i: weight for historical returns
$r_{k,i}$: the historical return of the k^{th} asset at the time i
I ∈ { t, …, T }.

Machine learning techniques provide alternatives for predicting the value of $E(r_k)$. ML Models learn from past return data as well as other essential information of asset in this technique. Following that, the model forecasts future returns based on the sequential data presented.

$$E_{ML}(r_k) = \hat{r}_{k,T+\epsilon} = f_\theta\left(D_{k,T}\right)$$

(2.3)

Where,

$r_{k,T+\varepsilon}$: denotes the expected future returns of the asset at time T + ε

f_θ: the ML model trained with the historical data

$D_{k,T}$: a set of data given at time T

2.2 Application of Machine Learning Algorithms in Portfolio Construction

The phrase "machine learning" refers to an umbrella term for methods and algorithms that enable machines to discover patterns in the absence of explicit programming instructions. ML researchers have done extensive work on portfolio construction using various Supervise, Unsupervised, and Deep learning techniques.

Tomasz Kaczmarek & Katarzyna Perez (2021) demonstrated that a **random forest algorithm** can be used to predict the cross-section of expected excess returns. They have built and compared portfolios using three different methods - mean-variance, Hierarchical Risk parity, and 1/N using returns of stocks from the S&P 500 and STOXX600 for robustness. Excess return estimates obtained from machine learning may help to overcome the well-known issues of quadratic mean-variance optimizers associated with sensitivity to changes in inputs. The conclusion of the study shows that both mean-variance and HRP optimizers portfolio optimization methods outperform the 1/N rule. The experiment was done on monthly price prediction using stock data of the past 20 years and the focus of the study was on the stable prediction rather than amplifying the forecasting performance.

Vivek, Zubayr, and Dr. Goswami worked on portfolio generation using unsupervised Machine learning for the Indian stock market. They have used the **K-mean Clustering algorithm** to ensure Risk Diversification and maintain a healthy rate of return. They choose stocks from these clusters that are closest to the respective centroids and used them to construct an efficient portfolio. The most significant components of this strategy are selecting acceptable investing parameters, determining the optimal value of k, and determining which stocks to select from which clusters. The produced portfolio outperformed the Sensex by 16.21% points and outperformed the BSE100 by 16.89% points. The work's performance clearly demonstrates that machine learning technologies can be utilized to construct varied portfolios that beat market indexes. In this research study, the researchers used the machine learning algorithm to the formed stock clusters to diversify the risk and constructed an equal-weight portfolio.

The historical data was used to construct the portfolio instead of forecasted price data. There has recently been a lot of interest in stock prediction using deep learning algorithms. Deep

learning approaches have recently received a lot of attention, thanks to advances in image processing and natural language processing. *Van-Dai Ta, Chuan-Ming Liu, and Direselign Addis Tadesse (2020)* published a research paper on Portfolio Optimization-Based Stock Prediction using a **Long-Short Memory Network.** Machine learning models such as **Linear Regression** and **Support Vector Regression (SVR)** were also employed to compare the effectiveness of the LSTM prediction model. The portfolio was built by choosing the outperforming stocks from the forecasted results with the highest expected return and the lowest risk. Simulation and optimization models were used to determine the optimal stock allocation for the created portfolio. Equal-weights allocation (EQ), simulation modelling Monte Carlo simulation (MCS), and mean-variance optimization (MVO) were utilized to estimate the best stock allocation weights. When compared to the S&P 500 index, the built portfolios fared well by achieving higher returns in both prediction and real trading.

Ramkumar's (2021) works on price prediction of cryptocurrencies compare deep learning neural networks models: convolutional neural network (CNN) and Long-Short Memory Network (LTSM) with algorithms traditional prediction model AutoRegressive Integrated Moving Average (ARIMA). He concluded that the result of the traditional model is low as compared to the neural networks model. Apart from that, he constructed portfolios using tick-by-tick data from the Binance using different methods like Kelly's criteria, mean-variance portfolio, and risk parity. Also, Portfolios performance analysis using different parameters like Sharpe ratio, annualized return, annualized volatility, Sharpe ratio, Sortino ratio, beta, Treynor ratio, information ratio, and maximum drawdown are calculated for each portfolio, and the best portfolio is chosen. The proposed study was done using the tick data of cryptocurrency, but the frequency of the data and input features of the algorithm to predict the price is not mentioned in the research paper.

The multivariate recurrent neural network (RNN) was used to forecast the price of cryptocurrencies by Hansun, S., Wicaksana, A., and Khaliq (2022). They did a comparative analysis of the prediction performance of the three different RNN algorithms. The feature used for predictions was Date (date), Open (float), High (float), Low (float), Close (float), Adj Close (float), and Volume (int). The study indicates that the GRU performed slightly better in terms of variation LSTM model, but LSTM's execution time was better than GRU. The Daily data used from yahoo finance to forecast the close price of the five crypto assets. According to the paper, there were missing values in the dataset, but the treatment of the miss values was not mentioned.

2.3 Portfolio Construction Methods

There are numerous portfolio construction methods that can help to reduce investment risk while optimizing asset returns. DeMiguel, Garlappi, and Uppal (2007) compared the performance of the mean-variance model and its expansions to that of the naïve 1/N portfolio.

The 1/N portfolio strategy is also referred to as an equal-weighted portfolio. In terms of Sharpe ratio and turnovers, the equal weight portfolio technique beats 14 models assessed across seven sample datasets and suggests optimal diversification. This implies that there are many "miles to go" before the profits promised by optimum portfolio selection can be realized outside of the sample.

Markowitz [1952] represents the beginning of contemporary portfolio theory, in which the issue of portfolio selection is properly articulated and solved for the first time. Before he could establish the "expected returns-variance of returns" rule, Markowitz had to debunk the commonly held belief at the time that an investor picks a portfolio by picking stocks that maximize returns. He believes that the Mean-Variance rule involves not just diversity, but also the correct sort of diversification for the right reasons.

The Kelly criteria are used by gamblers and investors to maximize profits over long periods of investment or betting. The paper by Peterson (2018) explains how the Kelly criterion may be used in the typical portfolio creation procedure, which includes the risk function. The experiment was carried out utilizing ten equities from major stock exchanges and several evolutionary algorithms. He compared the average return of Kelly's criterion portfolio with the mean-variance portfolio using the Monte Carlo simulation. His research indicated that Kelly's technique may be utilized to design a portfolio that achieves the same optimal returns as a mean-variance portfolio.

Risk-based investing methods are also appealing to investors and have grown in popularity in academia and practice. Burggraf (2019) assessed seven cutting-edge portfolio creation approaches for cryptocurrency investors (E Inverse Volatility, Minimal variance, L2-norm restricted minimum variance, L2-norm constrained maximum decorrelation, Maximum diversification, and Risk parity portfolio). The variance-covariance matrix, which is used to find the best portfolio allocation, is a feature shared by all portfolio approaches. As a result, portfolio weights are less sensitive to estimating the mistake in expected input. According to the analysis, risk-based strategies beat 13 cryptocurrencies and equal-weight portfolios in terms of return and volatility. Risk-based techniques aid in the protection of investors during difficult times.

Chapter 3 Methodology and Methods

In this chapter, Methodology, and methods that include Forecasting Algorithms, Portfolio construction methods, and performance measures will be discussed and compared. Among these are candidate methods, which are used to build a portfolio construction framework.

3.1. Outline of The Methodology

It is crucial to follow the precise methodology to build a machine learning model. There are six steps to follow for the development of models: Data Collection, Data pre-processing, Data analysis and Forecasting, Portfolio construction, and portfolio performance analysis. The below figure illustrates the order of six steps methodology:

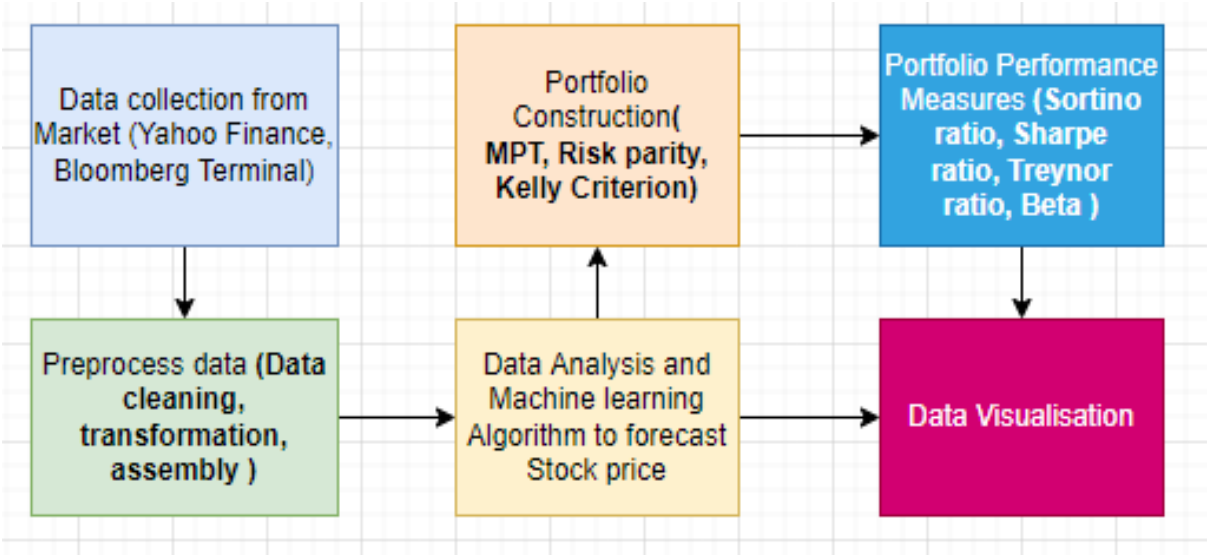


Figure 3.1 Methodology outline for Experiment

1. **Data collections:** There are several cryptocurrency data gathering sources, including CoinMarketCap, Yahoo Finance, Binance, and Bloomberg terminal. Data was collected in two formats: daily data and intraday data from Yahoo Finance (finance.yahoo.com, n.d.) and Bloomberg (Bloomberg Crypto Data, Research & Tools, n.d.). Daily data is available through Yahoo Finance, which provides data from CoinMarketCap. The intraday data is not available free of cost. The Bloomberg terminal is used to collect data for intraday data.

2. **Pre-process Data:** It is a critical phase for machine learning projects since the data collected in step 1 is cleaned and transformed during this phase. It can be used effectively while developing an ML model.

3. **Price Forecasting:** As per the literature review, various machine learning algorithms are used in different studies to predict the price of cryptocurrency/stocks. For example, Random Forest, Linear regression, Support vector Regression, and Neural networks (CNN, RNN, LSTM, etc). As part of this section, one of the algorithms has been employed to predict the price of an asset. An Exploratory Data Analysis (EDA) is performed to fully comprehend the data. The purpose of exploratory data analysis is to uncover potential insights or patterns in the data.

4. **Portfolio construction**: In this stage, the optimal weights for each asset are computed, which maximizes the portfolio's return while decreasing the risk. Various approaches are used to construct the portfolio such as the Mean-variance portfolio method, the Equal-weight portfolio method, Kelly's criterion, and others.

5. **Portfolio performance Measure:** It directs investors to examine portfolio performance to enhance returns while decreasing risk. A few approaches have been employed to measure performance, as explained in section 3.3.

3.2 Machine Learning Algorithms for Crypto Price Prediction

As discussed in Section 2.2, the Forecasting of the asset price is the core problem of portfolio management for cryptocurrency. Different supervised, unsupervised, and deep learning algorithms could be used to forecast pricing, according to many studies and research papers.

The Subfield of Artificial intelligence is called machine learning, which uses algorithms to uncover the pattern and learn from the given datasets to replicate human intelligence. **Support vector Machine** is a supervised and nonlinear machine learning technique. It can be used for time-series forecasting, but it lacks the precision and accuracy of random forest (Khedr et al., 2021). It can be enhanced by removing irrelevant and disorganized data points to enhance its accuracy. **Random forest** is a decision-tree model that adheres to the tree structure. It is effective at removing network instabilities, and if the optimal feature with the maximum information efficiency is chosen, it outperforms SVM in price prediction (Khedr et al., 2021). **K-means algorithm** is an unsupervised machine-learning algorithm that can be used to form clusters of data with similar characteristics. The Elbow method can be used to derive the ideal number of clusters. It can be used to find a group of assets that have some similar characteristics, but it is not suitable for predicting cryptocurrency prices as it is classification algorithm.

Deep learning methods is another common machine learning methodology used to tackle issues in a variety of domains, such as automated driving, aerospace and defense, medical

research, and finance (SearchEnterpriseAI, n.d.). Deep-learning methods use traditional neural networks that's why it is called deep-learning neural network. There are a few deep learning methods that are popular to forecast the price prediction of cryptocurrency i.e. A recurrent neural network (RNN) and Long Short-Term Memory neural network (LSTM).

RNN is a forward and backward dynamic network that has input, output, and context layers. Typically, RNN traditional architectures are as follows:

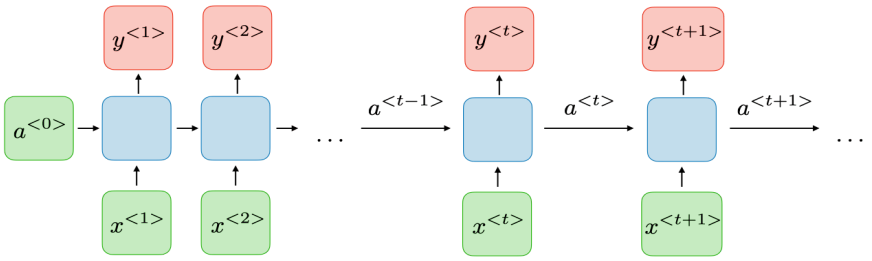


Figure-3.2 Architecture of the RRN network ***Adapted from****: (Stanford.edu, 2021)*

For each timestep t, the activation $a^{<T>}$ and the output y< t > are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{and} \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

(3.1)

Where,

W_{ax}, W_{aa}, W_{ya} and b_a, b_y : coefficients that are shared temporally

g_1* & *g_2: activation functions.

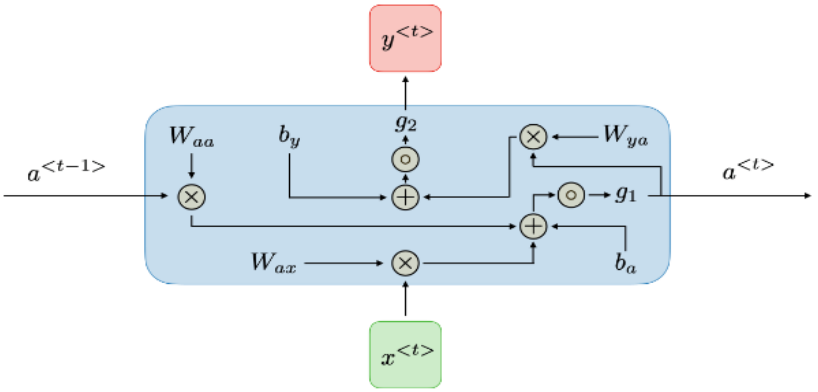


Figure-3.3 RNN cell ***Adapted from:*** *(Stanford.edu, 2021)*

Advantage of RNN as follows (Stanford.edu, 2021):

- The ability to manage input of any duration
- Model size does not increase with input size

- Computation takes into account past data
- Weights are shared throughout time

Disadvantage of RNN as follow (Stanford.edu, 2021):

- Low speed of computation
- Difficulty obtaining info from a long time ago
- Inability to evaluate any future input for the current state

In the context of RNNs, the vanishing, and exploding gradient phenomena are frequently found. They occur because it is difficult to capture long-term relationships due to the multiplicative gradient, which might be exponentially decreasing/increasing with respect to the number of layers. Traditional RNNs suffer from the vanishing gradient problem, which is addressed by gated recurrent units (GRU) and (LSTM), with LSTM being a generalization of GRU.

LSTMs are a kind of RNN that can learn long-term dependencies. Hoch Reiter and Schmidhuber (1997) introduced them, and numerous individuals developed and popularized them in subsequent work. They operate extremely effectively on a wide range of issues and are now frequently employed.

The LSTM model converts a series of previous observations into an output observation. The split-sequence() function is used to divide an input series into output samples, each with input and output timesteps.

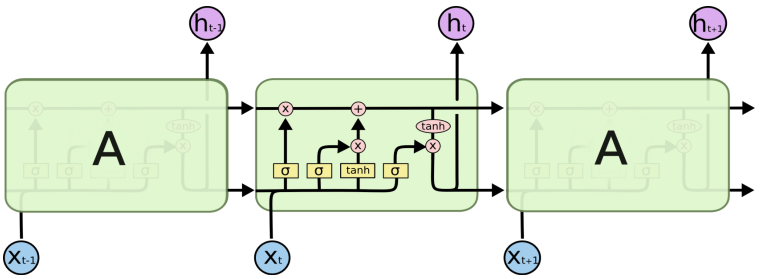


Figure-3.4 LSMT Network ***Adapted from:*** *(Olah, 2015)*

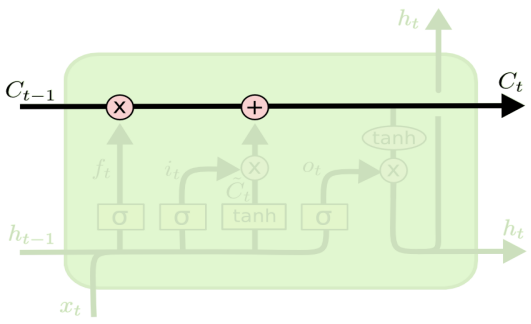


Figure-3.5 LSMT Cell ***Adapted from:*** *(Olah, 2015)*

The horizontal line across the top of the image represents the cell state, which is essential to LSTMs. The cell state is like a conveyor belt. It follows the whole chain, with only a few small linear interactions. It is quite simple for information to just pass over it unmodified. Because LSTMs manage both gradient and signal passing problems with long-term dependence, they are preferred to RNN (Olah, 2015).

The learning of the LSTM model dependence on certain internal parameters (neurons, batch size, dropout, etc) is called the hyperparameter. The process of deciding to optimize the values of those parameters is known as hyperparameter tuning. There are various techniques for hyperparameter tuning. The grid search technique is used in this project. Priyadarshini and Cotton (2021) presented a study on the hyperparameter tuning of the LSTM-CNN model for sentiment analysis. The performance of the model improved by 96% after tuning the parameter.

3.3 Portfolio Construction Methods and Performance Analysis Measures

Portfolio construction methods and portfolio performance analysis measures have been identified and details about those methods are given in this section.

3.3.1 Equal Weight Method

One of the fundamental portfolio construction methods is the equal weight technique, which assigns equal weightage to all assets. It also refers as the 1/N approach or the naïve portfolio creation method.

$$Wi = 1/N, \quad \textbf{(3.2)}$$

For $\forall_i = 1, ..., N$

Where,

W_i: the weight allocated to the i^{th} asset
N: the total assets of the portfolio

3.3.2 Kelly's Criteria

The Kelly criterion's delightfully simple formula estimates the ideal proportion of your bankroll to bet in order to maximize the geometric growth rate of wealth. However, this method not only guarantees maximum profit by successfully maximizing your possibilities, but it also guarantees you safety from a gambler's disaster (Sung, 2021).

$$f = (b*p - q)/b, \quad \textbf{(3.3)}$$

Where,

f = the fraction of the bankroll to bell

b = the odds

p = probability of winning

q = probability of losing i.e., 1-p

Below are the prerequisites of Kelly's criteria:

- Odd and probability should be known
- scale the Kelly output such that your bet equals your possible loss
- If only one of them is in your favour, it must outweigh the other.

3.3.3 Min-Variance Portfolio

Markowitz's mean-variance portfolio optimization methodology can be thought of as the selection of portfolio weights (x) that maximize the Sharpe ratio. The portfolio selection issue as defined by Markowitz [1952] can be written as

minimize $\sigma_r^2 = x^T V_x$ **(3.4)**

subject to $x_{T1} = 1$, $x_{TR} = Rp$.

Where

R_p: n-column vector of mean returns

T: the transpose of a vector

V: the (n * n) covariance matrix with entries σ_{ij}, i, j = 1, 2 . . .. n.

X: n -column vector whose components X_l, . . . X_n denote the weight of the investor's wealth allocated to the i^{th} asset in the portfolio with i = 1, 2 . . .. n.

To minimize the portfolio variance constraints: first, the portfolio weights must total to unity, indicating that all of the wealth is invested, and second, the portfolio must have an expected rate of return equal to Rp.

3.3.4 Hierarchical Risk Parity

Lopez de Prado (2016) described how to create diversified portfolios using the Hierarchical Risk Parity method. The Hierarchical Risk Parity approach constructs a portfolio using data from the assets' covariance matrix. In comparison to conventional risk parity approaches, hierarchical risk parity methods also result in less risky portfolios outside of the sample. Using graph theory and machine learning, this risk parity technique attempts to overcome the flaw in the Critical Line Algorithm (CLA), which Markowitz developed in 1954.

3.3.5 Portfolio Performance Analysis Measures

Below are performance measures that can employ to the performance analysis of the crypto portfolio:

1. **Sharpe Ratio:** William F. Sharpe established the Sharpe ratio, which is the ratio of portfolio total return minus risk-free rate divided by portfolio standard deviation, which is a measure of risk (Stanford.edu, 2019).

   Sharpe Ratio = $(R_p - R_f)/\sigma_p$ **(3.5)**

   where:

   R_p = return of portfolio
   R_f = risk-free rate
   σ_p = standard deviation of the portfolio's excess return

2. **Sortino Ratio:** Sortino ratio is a variant of the Sharpe ratio that utilizes downside deviation rather than standard deviation to assess risk, i.e. only returns that fall below a user-specified goal or needed rate of return is deemed dangerous (Investopedia, 2019).

   Sortino Ratio = $(Rp - r_f)/\sigma_d$ **(3.6)**

   where:

   R_p = Actual or expected portfolio return
   R_f = Risk-free rate
   σ_d = Standard deviation of the downside

3. **Downside Standard deviation:** The portfolio return's downside standard deviation shows if it is or is not below the minimum allowed return. The Sortino ratio is also computed using it. Since upside deviation is uncovered, downside deviation does not provide all the necessary information (Kenton, n.d.).

4. **Drawdown:** A drawdown is a financial word that refers to the drop in the value of a single investment or an investment portfolio from a relative peak to a comparable trough. A drawdown is a significant risk issue that investors must consider.

5. **Volatility:** A stock portfolio's value fluctuates from day to day. This is known as volatility, and it is a type of risk. The formula calculates the standard deviation from the variance of each stock's return in the portfolio by taking the square root of the sum.

   Portfolio variance = $w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1w_2Cov_{1,2}$ **(3.7)**

   where:

   w_1 = the portfolio weight of the first asset
   w_2 = the portfolio weight of the second asset
   σ_1= the standard deviation of the first asset
   σ_2 = the standard deviation of the second asset

$Cov_{1,2}$ = the covariance of the two assets, which can thus be expressed as $p_{(1,2)}\sigma_1\sigma_2$, where $p_{(1,2)}$: the two assets' the correlation coefficient

6. **Annualized Return:** The geometric average return of portfolio for each year over given period of the time is call annualized return. Annualized total return gives details about portfolio's performance does not provide information about price fluctuations or volatility (Investopedia, 2019).

   Annualized Return$=((1+r1) \times(1+r2) \times(1+r3) \times\cdots\times(1+rn))^{1/n}-1$ **(3.8)**

   r = return over the period
   n = number of years investment held

7. **Mean return:** The mean return of the portfolio can be used to determine the expected return or loss for the portfolio. Additionally, it aids in establishing a link between portfolio risk and return (Chen, n.d.).

3.4 Software

Python is an open-source programming language. It is used for web-development, database access, scientific and numerical operations. Python Package Index (PyPi) repository hosts thousands of third-party libraries with large open-source community support. The experiments are carried out effectively using the IDE tools like Jupyter and Google Colab. These tools are provided with features of IDEA tool as well as better presentation also. The project includes the implementation of machine learning model, portfolio construction methods and performance analysis. Hence, various package and open-source libraries are used. Table-3.1 presents the python modules, open-source libraries, and their core functionalities:

Table 3.1 List of Python Packages and Modules

Package Name	Functionality
pandas_datareader	Use for access to remote data for panda's library for example yahoo finance (pandas-datareader.readthedocs.io, n.d.)
Numpy	Open-source library for scientific operations on arrays and matrices (NumPy, n.d.)
Pandas	Open-source library to structure and analysis data (pandas.pydata.org, n.d.)
Datetime	Python Package to perform arithmetic operation on date and time(Python.org, 2020)
Requests	Library to send Http request (Python.org, 2020)
Matplotlib	Library to visualize the data (matplotlib.org, n.d.)

Seaborn	Library for Data visualization that is extension of the matplotlib (seaborn, 2012)
Time	Python Module to perform time related operations (Python.org, 2020)
Yfinance	Library to download the financial data from Yahoo Finance (Aroussi, n.d.)
Tensorflow	Libray to develop machine learning models and application (TensorFlow, 2019)
tensorflow.keras	Implementation of High level keras to use models, layers (Dense, LSTM, Dropout) load_model, callbacks (EarlyStopping, ModelCheckpoint) (TensorFlow, 2019)
sklearn.preprocessing	Library to scale data using MinMaxScaler (Scikit-learn.org, 2019)
Itertools	Python module to perform effective looping (Python.org, 2020)
Random	Module to generate random numbers (Python.org, 2020)
M**ath**	Python module to use math function buit in C programing language (Python.org, 2020)
Os	Python module to access Operation System related related functionality (Python.org, 2020)
Cvxopt	Library to optimize the covex and built on python. (cvxopt.org, n.d.)
Pypfopt	Library to construct portfolio using HRPOpt, expected_returns packages. (pyportfolioopt.readthedocs.io, n.d.)

Chapter 4. Experimentation

The experimentation part covers the six methodology steps, which are discussed in Section 3.1. This section can be divided into four sub-sections:

1. Data collection of cryptocurrencies (Daily and Intraday) price from sources like Bloomberg terminal (Bloomberg Crypto Data, Research & Tools, n.d.) and yahoo finance (finance.yahoo.com, n.d.).
2. Exploratory data analysis on daily data.
3. LSTM model implementation to forecast the close price of the cryptocurrencies using three frequencies (10-min, 30-min, and 60-min) of the intraday data.
4. Analysis of the predicted results and finalization of the result for the next step of experiment.

At the last, construct the portfolios using four methods with help of the close price (train and predicted data) of the cryptocurrencies and the performance analysis of the portfolios in terms of performance measures defined in Section 3.3

4.1 Data Collection and Exploratory Data Analysis

Both Data collection and Data analysis are primary and crucial steps of the experiment to achieve the objectives. In this section, the data sources and acquisition process will be

explained to understand the data collection. After that, the exploratory data analysis will be discussed to understand the characteristics of data using data visualization.

4.1.1 Data Source and Collections

The data for this project has been gathered from reputable and trustworthy sources like Yahoo Finance, and Bloomberg. There are two types of datasets collected as follows:

1) **Daily Price Dataset:** The daily price dataset is time series data of crypto price which have one record per day which is collected by end of the day. There are seven columns available in the dataset which are shown in the table 4.1 (Refer 1-7 fields). The daily price of the cryptocurrencies is collected from the Yahoo Finance portal using data_reader python library. Yahoo Finance uses CoinMarketCap APIs for daily and historical daily data on around 9089 distinct cryptocurrencies (finance.yahoo.com, n.d.). 20 popular crypto coins data have been collected for last five years (2018-2022) to perform the exploratory analysis.
2) **Intraday Price Dataset:** The intraday data is time series data of crypto price, but it has 1 record per frequency for a day. For example, if the frequency of the intraday data is 10-min, It means data is collected for every 10 mins intervals. For this project, intraday data of 10-min, 30-min, and 60-min intervals have been collected for the period of March-September 2022. The datasets are acquired for the top five cryptocurrencies by Market Cap. (Bitcoin, Binance, Ethereum, Tether, and USD) from the Bloomberg terminal and used for price forecasting. Bloomberg has been providing crypto data since 2013, and after 2018, the scope of instruments increased from 10 to 50, as well as providing additional information like headlines, historical intraday data, and other fundamental information (Bloomberg Crypto Data, Research & Tools, n.d.). Bloomberg terminal facility is provided by the University of Westminster at the Marylebone campus. Cryptocurrency data can be downloaded after obtaining credentials for the Bloomberg terminal. Refer to Table 4.1 for columns of the Intraday dataset. (Refer to columns 1-5 and 9-11).

Table 4.1: Data Field Type and Description

SR no	Column name	Type	Description
1	Date	Date/Datetime	Date/Datetime of cryptocurrency price recorded
2	Open	Float64	Opening price of the day in USD
3	High	Float64	Highest price of the day in USD

4	Low	Float64	Lowest price of the day in USD
5	Close	Float64	Closing price of the day in USD
6	Adj Close	Float64	the closing price after adjustments for all applicable splits and dividend distributions
7	Volume	Float64	Volume of the transactions
8	Market Cap	Float64	Market capitalization in USD
9	SMAVG_50d	Float64	Simple Moving Average of the last 50 days
10	SMAVG_100d	Float64	Simple Moving Average of the last 100 days
11	SMAVG_200d	Float64	Simple Moving Average of the last 200 days

4.1.2 Exploratory Data Analysis

An exploratory data analysis is the most efficient technique to comprehend the components of the data and its characteristics. The aim of exploratory data analysis (EDA) is to discover what the data can tell us apart from applying machine learning models to it.

The EDA has been carried out using Python's data visualization packages (matplotlib, plotly, seaborn, pandas, DateTime) and data from 20 cryptocurrencies. First, for all columns, use the pandas' library's isnull() and sum() functions to find null values and convert string values to suitable data types (date value to DateTime and currency to float datatype).

There are no missing values in any row, although data for some coins before 2018 are not available because several cryptocurrencies were introduced after 2018 *(for Example: USD and Tether)***.** As a result of that further analysis has been performed on the top 5 coins by market capitalization.

Market capitalization:

Market capitalization (or market cap) is the total worth of all coins created for a cryptocurrency. It is computed by multiplying the current market price of a single coin by the number of coins in circulation.

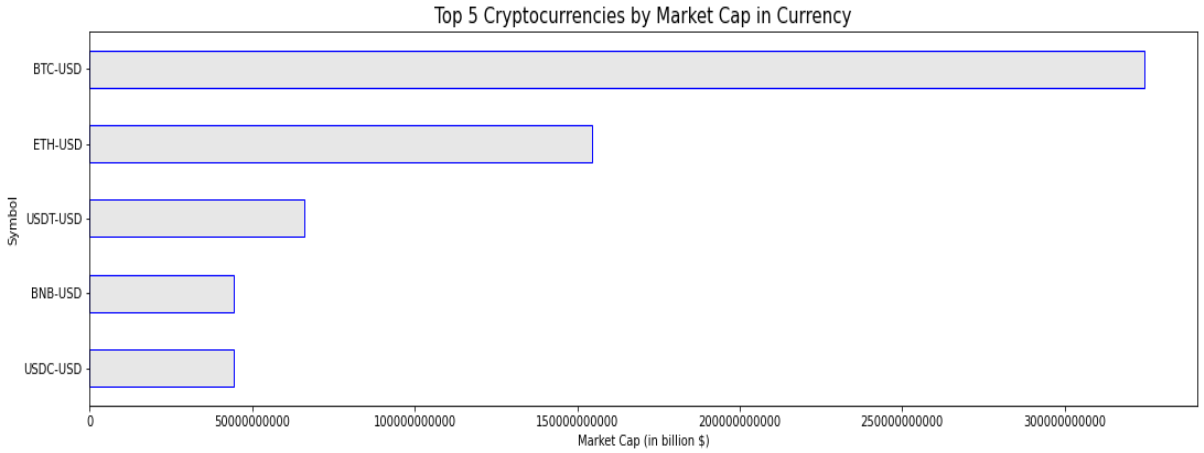


Figure 4.1: Top Five Cryptocurrencies by Market Cap

Figure 4.1 shows horizontal bar chart of the top five crypto coins by market capitalization which are bitcoin, Ethereum, Tether, Binance Coin, and USD Coin. Among these, Bitcoin has the highest, whereas USD coin has the lowest market capitalization.

Close Price:

Further exploring the close price of all the assets, Bitcoin has the highest close price, followed by Ethereum, Binance. The close price of Tether and USD coins is lowest as compared to other coins. Refer to Figure 4.2.1 to Figure 4.2.3

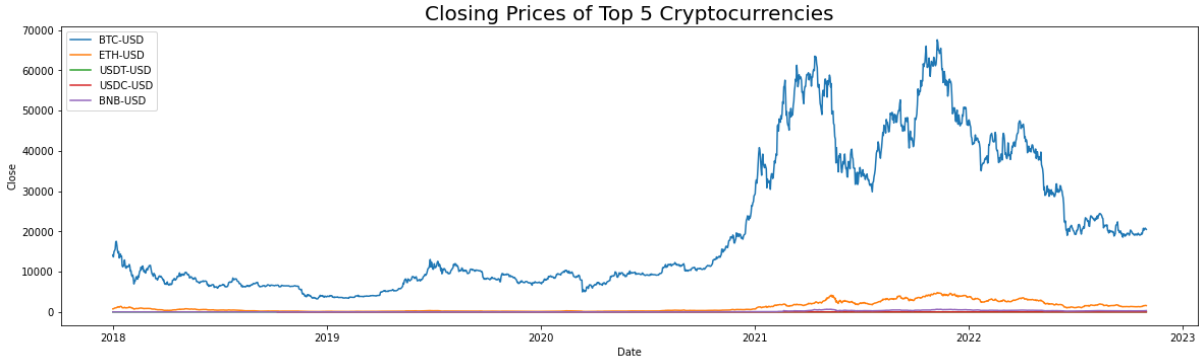


Figure 4.2.1: Top Five Cryptocurrencies by Market Cap

As shown in Figure 4.2.1, the closing price of bitcoin is the highest among all five cryptocurrencies.

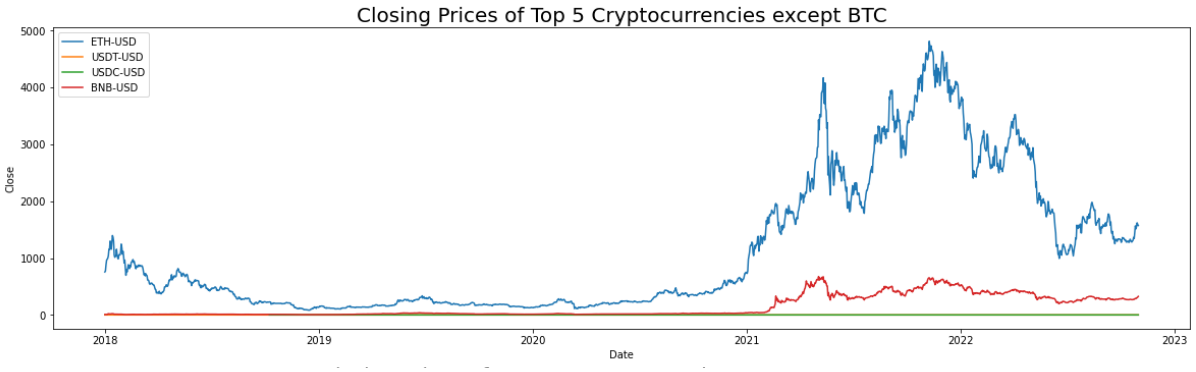


Figure 4.2.2: Closing Price of the Cryptocurrencies by Market Cap (Ethereum, Binance, USD, Tether)

As shown in Figure 4.2.2, the closing price of Ethereum is the Second highest among all five cryptocurrencies for the given period.

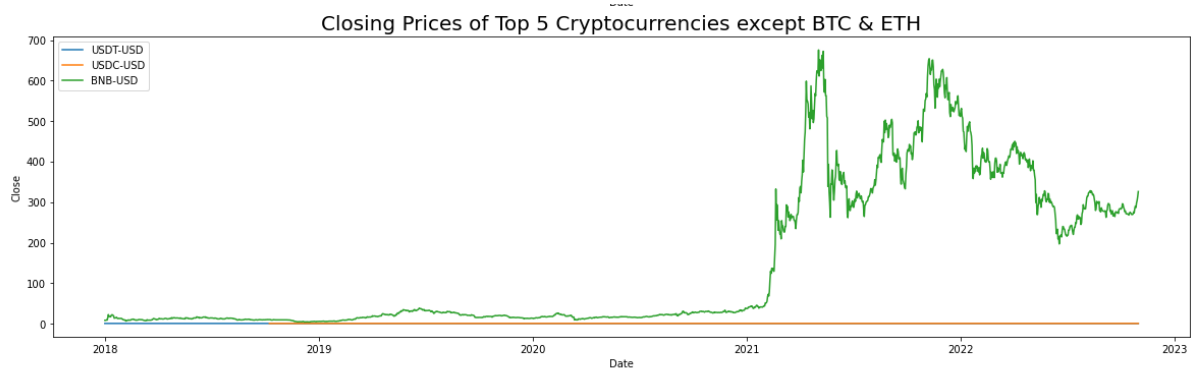


Figure 4.2.3: Closing Price of the Cryptocurrencies by Market Cap (Binance, USD, Tether)

As shown in Figure 4.2.2, the closing price of Binance is the third highest among all five cryptocurrencies.

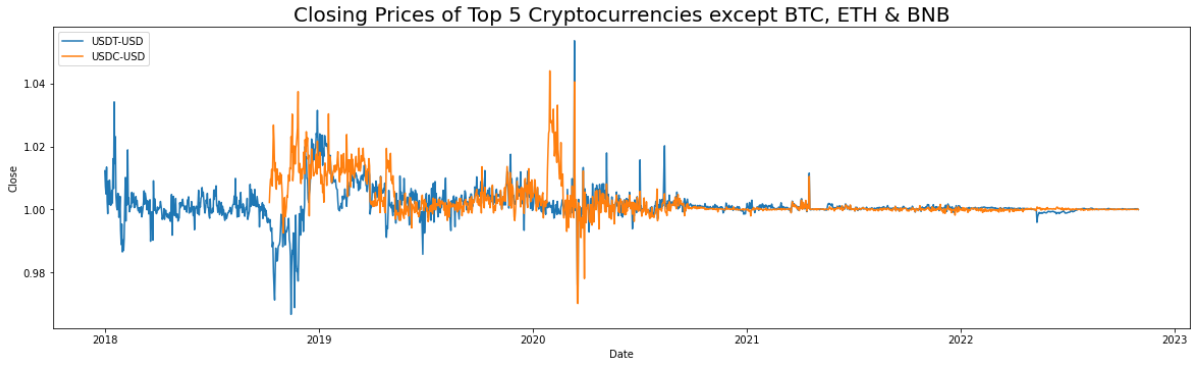


Figure 4.2.4: Closing Price of the Cryptocurrencies by Market Cap (USD, Tether)

As shown in Figure 4.2.2, the closing price of the USD and Tether is the lowest among all five cryptocurrencies and almost the same for both cryptocurrencies.

Simple Moving Average:

Moving Average is a common indicator for asset technical analysis. It aids in identifying bear signals (price drops) and bull signals (price rises). Traders keep an eye on the 50-day and 200-day moving averages, which are used to calculate the death cross (50-day moving average crossing below the 200-day moving average) and golden cross (50-day moving average crossing above the 200d moving average) (Deer, 2022). Both 50 days and 200 days Moving averages are derived using *rolling(window=n).mean()* function of the dataframe of the pandas' package and plot for moving averages versus the close price for all five coins are shown below (Refer to Figures 4.3):

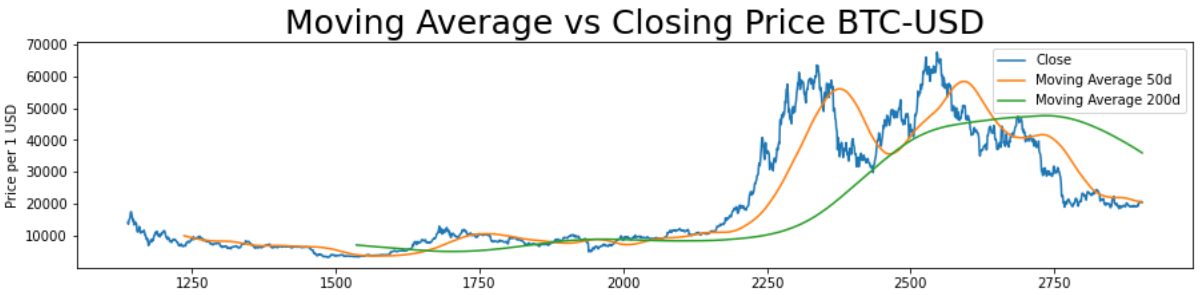


Figure 4.3.1: Moving Average v/s Closing Price of Bitcoin

As shown in Figure 4.3.1, There are approximately four golden (price rise) and death cross (price rise) over the given time, which indicates that bitcoin is a volatile cryptocurrency.

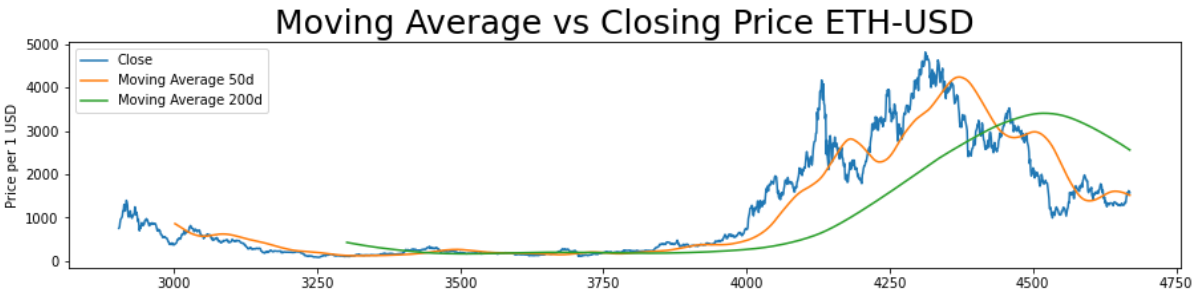


Figure 4.3.2: Moving Average v/s Closing Price of Ethereum

As shown in Figure-4.3.2, approximately two golden (price rise) and three death (death rise) cross over the given time, which indicates that Ethereum is a less volatile cryptocurrency than bitcoin.

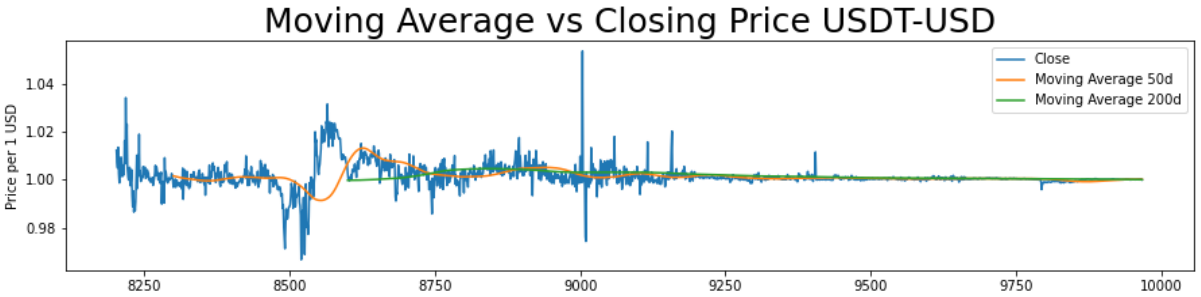


Figure 4.3.3: Moving Average v/s Closing Price of Tether coin

As shown in Figure 4.3.3, The moving average lines almost overlap for 50 and 200 days. It is difficult to conclude the volatility of the Tether coin.

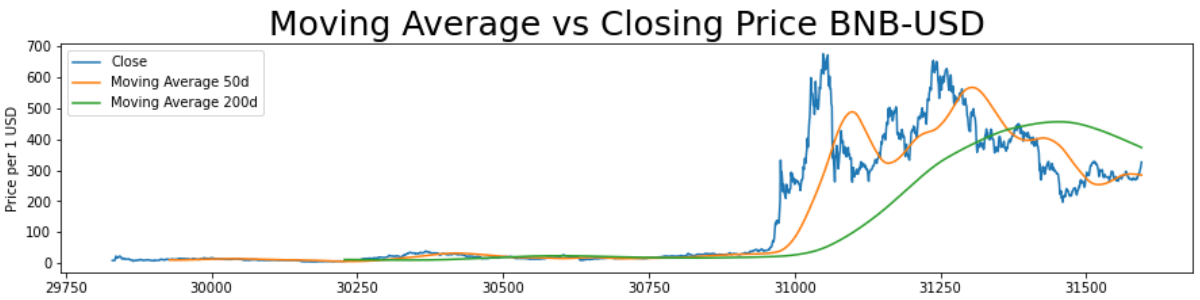


Figure 4.3.4: Moving Average v/s Closing Price of Binance

As shown in Figure 4.3.4, The moving average lines for both averages almost overlap but, in the end, there is one death (price drop) cross which mean Binance is less volatile than bitcoin and Ethereum.

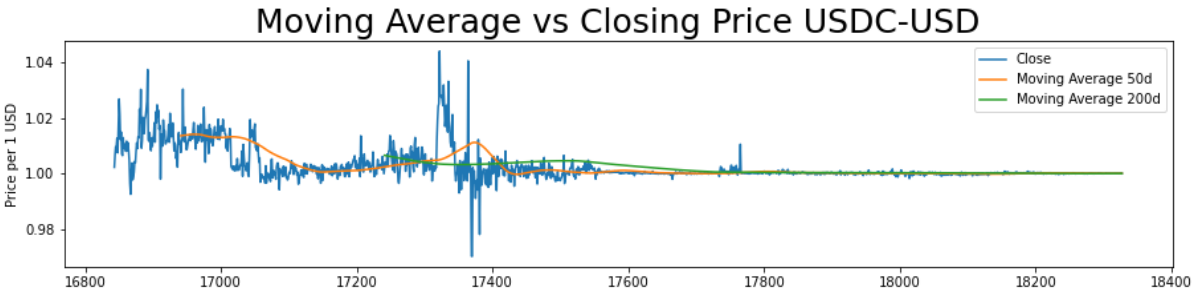


Figure 4.3.5: Moving Average v/s Closing Price of USD

As shown in Figure 4.3.5, The moving average lines almost overlap for 50 and 200 days after 17800 records. It is difficult to conclude the volatility of the Tether coin. Before that, only one death and golden cross for the USD coin.

Daily Return:

A return is a change in an asset's price over specified period of time. Positive returns indicate a profit, whereas negative returns indicate a loss. To compute the return, we will use the pandas *pct_change()* method.

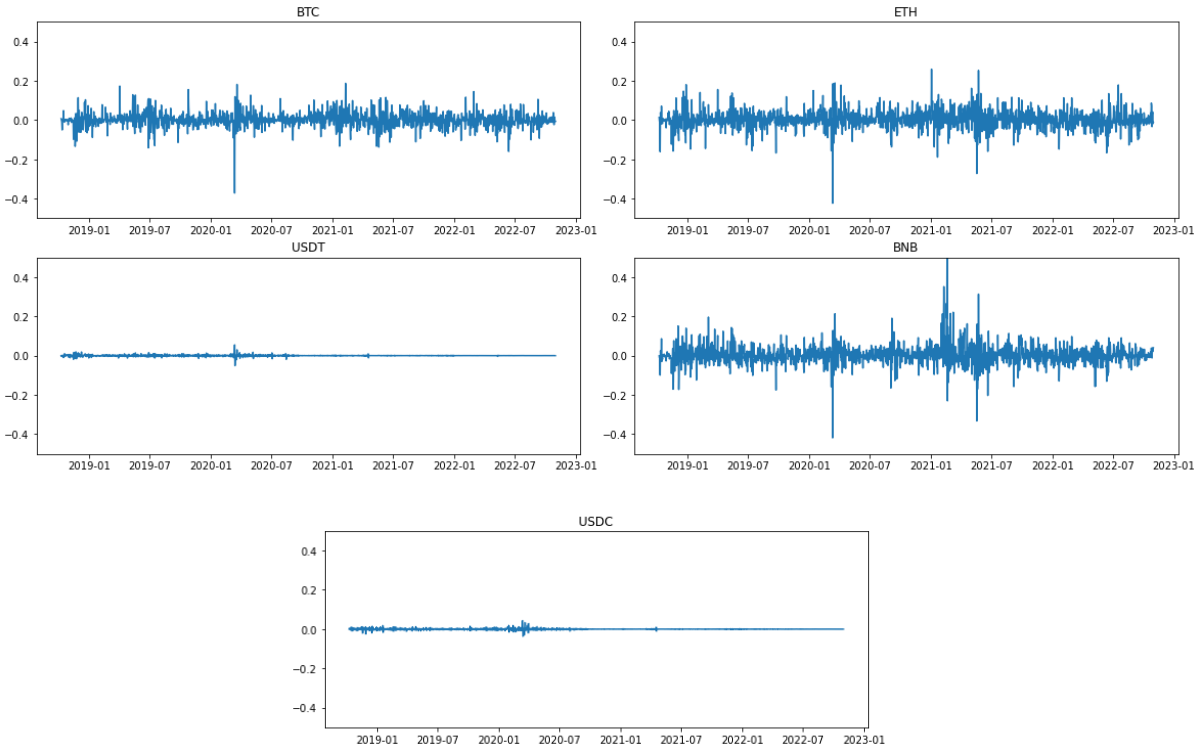


Figure 4.4: Daily Return of the Top Five Cryptocurrencies by Market Cap

Return of all coins are shown in Figure 4.4. BNB is the most volatile cryptocurrency, followed by ETH, whereas BTC is the least volatile. There was a massive fall in all cryptocurrencies around March 2020 during the initial COVID pandemic wave.

Correlation of Daily Return:

The corr() function in the pandas package is used to calculate the correlation of daily returns of the top five bitcoins. It depicts the correlation between daily returns for all coins. The following is a heatmap depicting the correlation of daily returns:

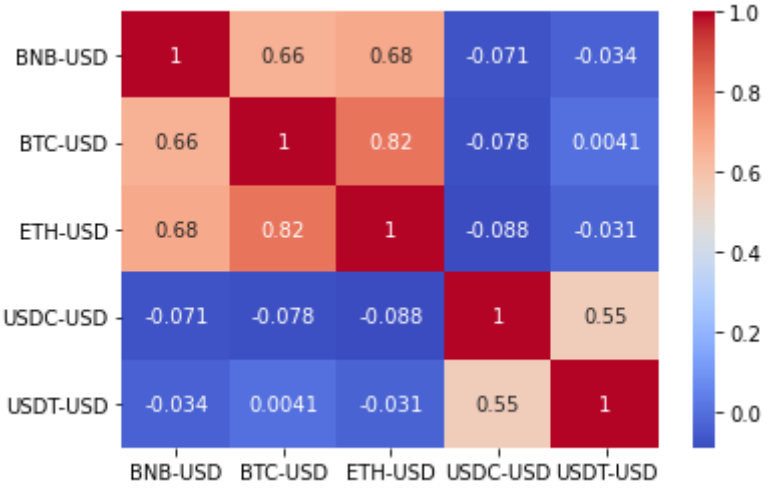


Figure 4.5: Correlation Matrix of the Top Five Cryptocurrencies by Market Cap

According to the heatmap, Bitcoin and Ethereum are significantly correlated (0.82), whereas bitcoin and Tether have the least correlation (0.0041).

Volatility:

Volatility (standard deviation of the returns) is a measure of how an asset's price changes over time. The greater an asset's standard deviation, the more volatile it is.

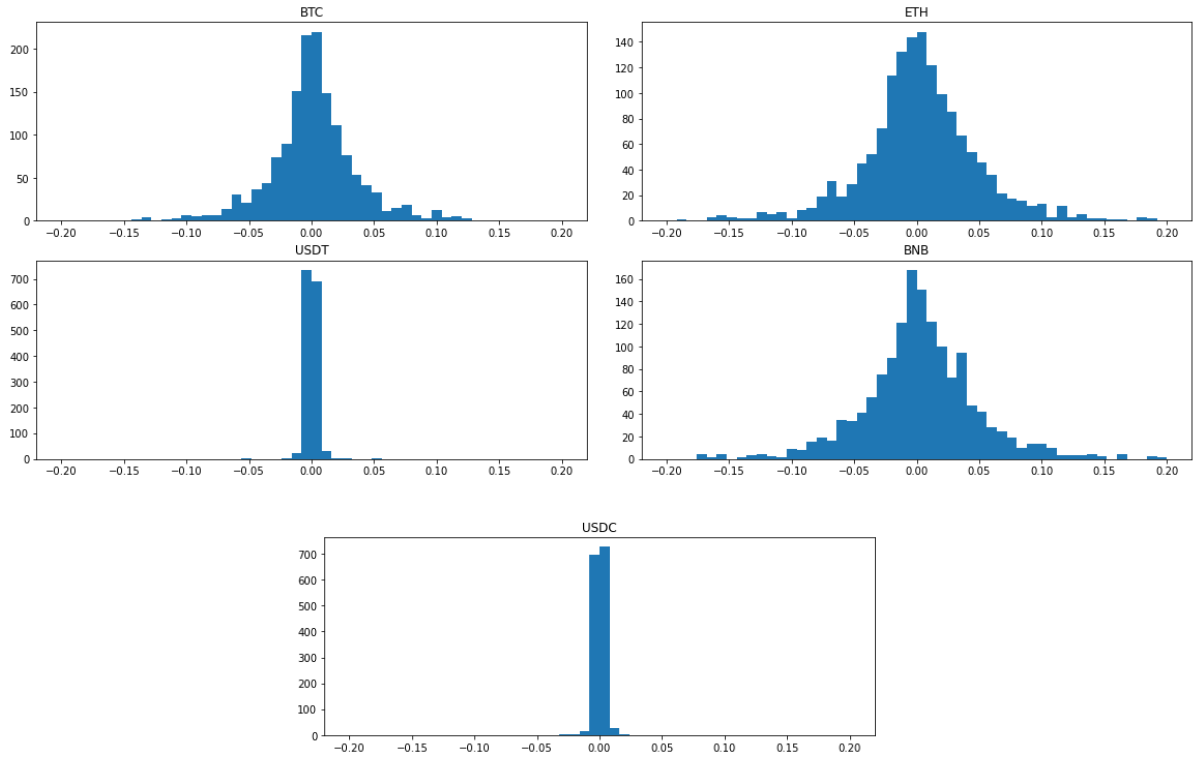


Figure 4.6: Volatility of the Top Five Cryptocurrencies by Market Cap

The histogram in Figure 4.6 represents the volatility of all coins. USD and Tether are more volatile, whereas Binance and Ethereum are the least volatile assets.

Cumulative Return:

The overall change in the price of an asset over time is expressed by the cumulative return. To compute the daily cumulative simple returns, the pandas cumprod() method is utilized. The cumulative returns of all coins are represented using a line chart in Figure 4.7.

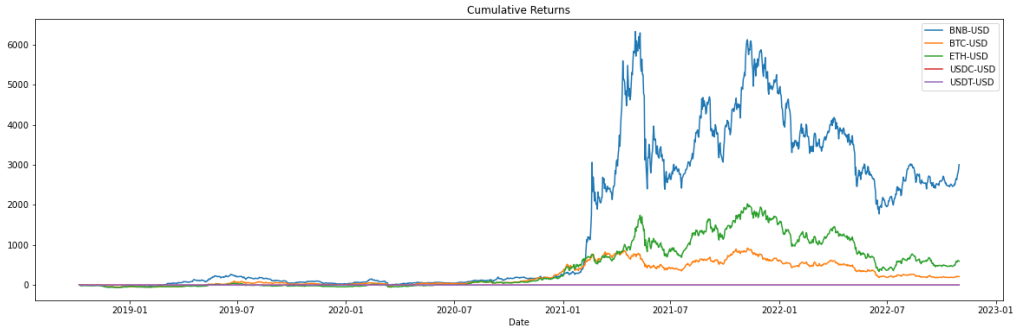


Figure 4.7: Cumulative Returns of the Top Five Cryptocurrencies by Market Cap

From March 2021, Binance beats Ethereum, Bitcoin, and XRP, while Ethereum outperforms Bitcoin and XRP.

4.2 Implementation of LSTM Multivariant Price Prediction Model

The LSTM model has been built for the 15 intraday datasets of five digital currencies (USD, Ethereum, bitcoin, Tether coin, and Binance). The following essential steps were taken to construct the model and forecast the close price of the cryptocurrencies:

1. Data pre-processing to ensure data sanity and perform necessary transformations
2. Scale the data and choose the pertinent features/input variables for modeling.
3. Prepare mini batches that are used to forecast cryptocurrency values using the sliding window method.
4. Design and training of the LSTM model
5. Model performance validation
6. Model performance enhancement through hyperparameter tweaking
7. Predict the close price and reverse the anticipated target values' scale back to their original value.

4.2.1 Data pre-processing

The process of creating a data model must include the pre-processing of the data. The model's performance may suffer if there is an anomaly or data inaccuracy. The first step is to use the *read_csv()* function from the pandas' package to import the intraday data into a DataFrame and perform the following checks:

Duplicate Value Check: this check identifies the duplicate records in the dataset and treats duplicate records. The method *duplicated()* is used. It returns a boolean value as "True" if duplicate records are available in the dataset. There are no duplicate records in all the datasets. The result of the check is as below for all the datasets:

```
Duplicate Values:
Empty DataFrame
Columns: [Date, Open, High, Low, Close, SMAVG_50d, SMAVG_100d, SMAVG_200d]
Index: []
```

Figure 4.8: Dataset Duplicate Value Result

Missing Value Check: if there are null or NA values in the dataset that impact the model performance, they must be treated. To Identify the missing values, the *isnull()* method followed by *sum()* is used that provide a few missing values. There is no missing value for any field. Refer to Figure 4.9.

```
Missing Values:
Date          0
Open          0
High          0
Low           0
Close         0
SMAVG_50d     0
SMAVG_100d    0
SMAVG_200d    0
dtype: int64
```

Figure 4.9: Dataset Missing Values

Data Type of Dataset: The *info()* function, which provides details about the Column name, Non-Null Count, and Data type. It is used to obtain information about a dataset. Since the Date field's data type is an object, the date is converted using the *to_datetime(data['Date'])* function before being sorted using the *sort_values(by="Date", ascending=True)* function. Refer to Figure 4.10 for the *info()* method's output.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13525 entries, 0 to 13524
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        13525 non-null  object
 1   Open        13525 non-null  float64
 2   High        13525 non-null  float64
 3   Low         13525 non-null  float64
 4   Close       13525 non-null  float64
 5   SMAVG_50d   13476 non-null  float64
 6   SMAVG_100d  13426 non-null  float64
 7   SMAVG_200d  13326 non-null  float64
dtypes: float64(7), object(1)
memory usage: 845.4+ KB
```

Figure 4.10: Dataset Detailed Info

Apart from three checks, data from the datasets selected with help of *from (2022-04-01)* and *to (2022-09-30)* date using code snippet:

```
#Dataset Selection by date from dataset
```

data=data[(data.index >= f_date) & (data.index <= t_date)]

4.2.2 Feature Engineering and Scaling

Identifying and developing relevant input features to feed into the Long Short-Term Memory (LSTM) network is referred to as feature engineering for time series analysis or prediction. These features can be determined from the time series' raw data or from additional outside data sources. Multiple input features that each relate to a different variable in the time series data would be present in a multivariate time series problem.

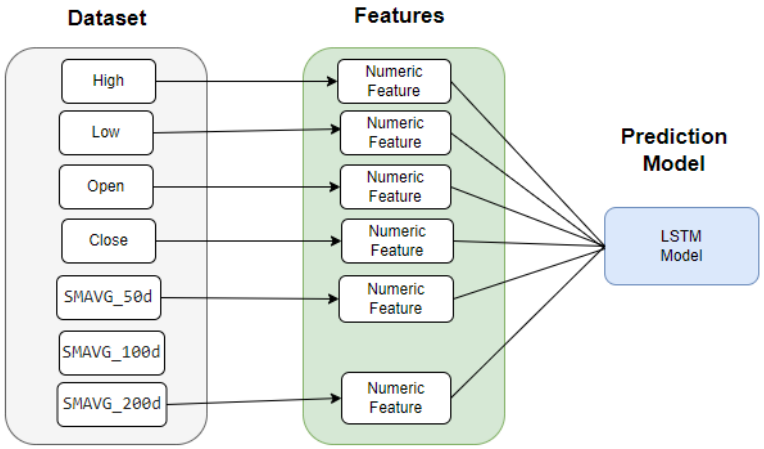


Figure 4.11: Feature Engineering for LSTM Model

There are seven numerical (*High, Low, Open, Close, SMAG_50d, SMAVG_100d, SMAVG_200d*) and one date column in the dataset. A simple moving average (SMAVG) is a statistical tool. It is frequently used to smooth out short-term volatility in data and highlight longer-term patterns. A SMAVG can be used to smooth out price swings and spot underlying trends in the price of coins in the world of cryptocurrencies (Deer, 2022). The SMAG_100d is dropped using *drop(['SMAVG_100d'], axis=1)* functions to create a feature' DataFrame. Selected features are shown in Figure 4.11, which will further pass to the prediction model.

Then feature values are scaled to make them fall within the same range. Additionally, it speeds up network convergence and helps the LSTM network operate better. The feature values are scaled using the Mix-Max scaling, and the value range between 0-1.

```
# Transform the data by scaling each feature to a range between 0 and 1
scaler = MinMaxScaler()
np_data_scaled = scaler.fit_transform(np_data_unscaled)
```

Figure 4.12: Feature Value Scaling Using MixMaxScaler

The *MinMaxScaler* class from the *sklearn.preprocessing* package is used to scale the NumPy array of input features, as illustrated in Figure 4.12.

4.2.3 Prepare the Input Data for LSTM

A particular input (3D array) and output (2D Array) data structure are needed for LSTM. The first dimension of a three-dimensional data structure is the sequence, the second is the time steps, and the third is the features (refer to Figure 4.13). Before transforming data, Dataset is split into training and train sets, in which 80% of data is assigned to the training set and 20% of data allocate to the test set.

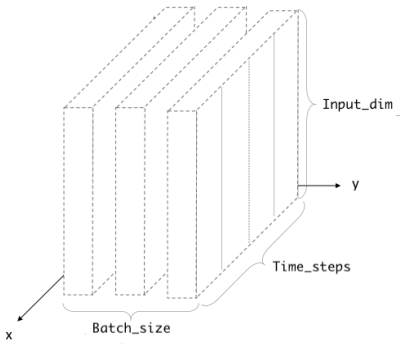
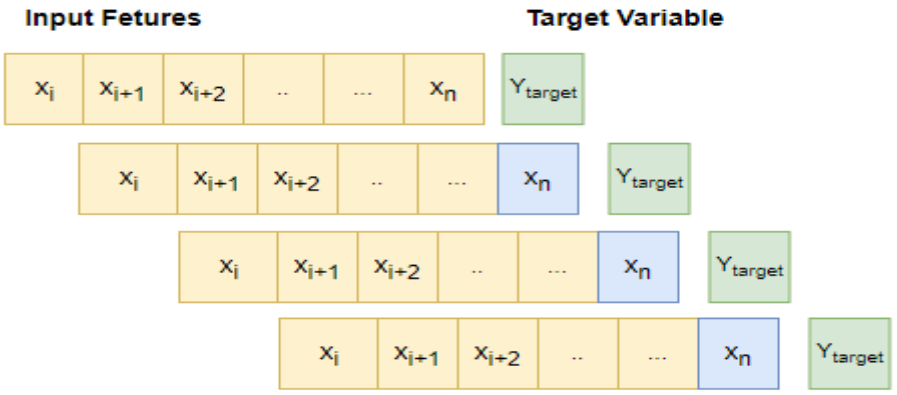


Figure 4.13: 3D Input Data Structure for LSTM

The sliding window technique is used to prepare multivariant data, that can be processed by the neural network. Data is sliced into the window of a specific sequence with the associate target value. This approach adds multiple data point input datasets (x_train or x_test), and the target value follows the window. The target value is stored in a separate target dataset (y_train or y_test). Refer to Figure 4.14 for the sliding window technique and the python script.




```
def partition_dataset(sequence_length, data, index_Close):
    x, y = [], []
    data_len = data.shape[0]
    for i in range(sequence_length, data_len):
        x.append(data[i-sequence_length:i,:]) #contains sequence_length values 0-sequence_length * columsn
        y.append(data[i, index_Close]) #contains the prediction values for validation,  for single-step prediction

    # Convert the x and y to numpy arrays
    x = np.array(x)
    y = np.array(y)
    return x, y
```

Figure 4.14: Sliding Window Technique

The function *transform_multivariate_data()*, which accepts the original dataset, scaled data, and size of the window sequence, is developed to perform the operations mentioned earlier like transformation, splitting, and partition. The function's output is illustrated in Figure 4.15.

```
x_train, y_train, x_test, y_test,train_data_len = transform_multivariate_data(data,np_data_scaled,sequence_length)

x_train.shape: (6823, 50, 6) y_train.shape: (6823,)
x_test.shape: (1718, 50, 6) y_test.shape: (1718,)
0.5630252100840494
0.5630252100840494
```

Figure 4.15: Final Test and Training Dataset for LSTM

The output in Figure 4.15 displays the number of samples in the training (6823) and test (1718) dataset with target output (6823) for training and test (1718). It is an example so it can vary for 15 intraday datasets based on the total number of records in the dataset.

4.2.4 LSTM Model

The functions from the advanced machine learning libraries (Keras and TensorFlow) are used for the LSTM model. The design of the LSTM model requires three steps to complete:

1. Network Layers,
2. Compilation
3. Training

The LSTM layers are created with *the sequential()* function. It allows adding incremental stack layers by using *add()* method. For the experiment, two LSTM layers and one dense layer are used. Dropout layers are added to drop the fraction of input during training after each LSTM layer. It helps to prevent the overfitting problem of the LSTM model. In the First LSMT, The *return_sequence* parameter is configured as True to return the hidden for each timestep. The network layer design of the LSTM model is as follows:

- **LSTM Layer:** units=16, return_sequences=True
- **Dropout Layer:** 0.2 dropout
- **LSTM Layer:** units=16, return_sequences=False
- **Dropout Layer:** 0.2 dropout
- **Dense Layer:** units=1, activation='linear'

The *compile()* function is used to compile the design LSTM model with a loss function (Mean square error), metrics (Root mean square error) and an optimizer. The loss function is a measure of how a good model can predict the target values during training, and the optimizer updates the model parameter based on the input data and gradient of the loss function. Refer to Figure 4.16 for the LSTM model design.

```
regressor = Sequential()
regressor.add(LSTM(units=16, return_sequences=True, input_shape=(x_train.shape[1], x_train.shape[2])))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=16, return_sequences=False))
regressor.add(Dropout(0.2))
regressor.add(Dense(units=1, activation='linear'))
regressor.compile(optimizer='adam', loss='mse', metrics=[tf.keras.metrics.RootMeanSquaredError()])

print('LSTM Regression Summary :')
print(regressor.summary())

es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=5)


# fit model
history = regressor.fit(x_train, y_train, validation_split=0.2, epochs=40, batch_size=64, callbacks=[es])
```

Figure 4.16: LSTM Model Design Details

The model training is done by using the *fit()* function. Input parameters to *fit()* functions are as follows:

Epochs: 40; number of full passes of training dataset through the model

batch_size: 64 ; decide the number of training sample,

callbacks: EarlyStopping; the callback function calls after each epoch to update trigger events. Based on given parameters to EarlyStopping, Event checks if the model performance is not improving, then terminate the training.

input variable: x_train**;** input parameters

target variable: y_train; target parameters

validate_split: 0.3; proposition training data allocate to validation of model training.

4.2.5 The Model Performance Analysis

Performance is assessed using the Root Mean Square Error (RMSE) and Mean Square error (MSE) as Loss Functions, both of which were defined during model design. After being fitted with the training dataset, the model returns the history variable. It contains information about performance metrics and a loss function. The *plot()* function of the *matplotlib.pyplot* library used to depict the error data using two-line graphs (epochs v/s loss) before and after optimization. For details on both line plots, see Figure 4.17.

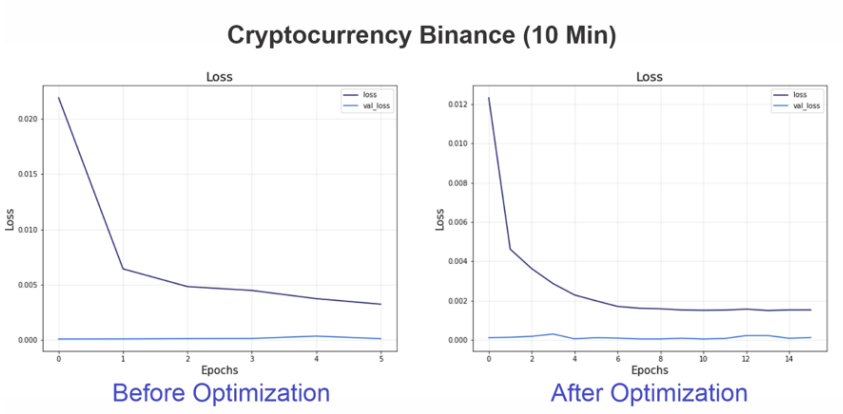


Figure 4.17: Error Line Graphs (Epochs v/s Loss)

The line graph in Figure 4.17 illustrates the decrease in Loss function (MSE) values with each epoch before and after the optimization. For all 15 datasets, this graph is presented and discussed in Section 4.3

4.2.6 Hyperparameter Tuning for Model

The model has two different types of parameters: model parameters and hyperparameters. The hyperparameters are control implementations of the model and needed to be set for better performance of the machine learning model, whereas the model parameters are learned during model training. Tuning the hyperparameters improves the model's precision and generalization.

The GridSearch Method is utilized to adjust the LSTM hyperparameter as part of the experiment. A grid of hyperparameter values is created using the combinations of neurons (16, 32), batch size (32, 64, 128), and dropout (0.1, 0.2). For each combination, The model is trained to determine which hyperparameter combination is optimal. The potential combinations of the hyperparameters are developed using the *product()* function of the itertools package. The same LSTM design described in Section 4.2.4 is employed for the tuning procedure. Two callbacks *Modelcheckpoint* and *EarlyStopping* are passed when creating the model to improve the effectiveness of the grid search parameter tuning (Team, n.d.). To comprehend the layout of the callbacks and model, consult Figure 4.18.

```
regressor.fit(x_train,
              y_train,
              validation_split=0.3,
              epochs=40,
              batch_size=n_batch_size,
              callbacks=[es, mc],
              verbose=0)
```

```
mc = ModelCheckpoint(file_path,
                     monitor='val_loss',
                     mode='min', verbose=1,
                     save_best_only=True)
```

```
es = EarlyStopping(monitor='val_loss',
                   mode='min',
                   verbose=1,
                   patience=5)
```

Figure 4.18: Structure of the fit() Functions and Callbacks used for the LSTM Model

According to the input parameters (*monitor* and *mode)*, EarlyStopping call-backs track the training loss. It determines if the loss is decreasing or not at the conclusion of each period. If the loss is not decreasing, the training is stopped based on the *patience* parameter. The best

model's current state is saved to disc during a ModelCheckpoint call-back so that it can subsequently be used to continue training the dataset.

After model training, the hyperparameters and accuracy scores are recorded for each combination in a list. The hyperparameter with the lowest error score is regarded as the best hyperparameter and model trained with the optimized hyperparameters. The result of before and after hyperparameter tuning is discussed for all datasets in Section 4.3.

4.2.7 Predict Cryptocurrency Price

The test dataset (20% of the total dataset) is passed to predict the close price as input to *predict()* method of the optimized model. The predicted price is scaled and reversed in order, so inverse transformation is performed *inverse_transform()* of *MinMaxScaler* to get original values in the correct order.

```
y_pred_scaled = regressor.predict(x_test)
y_pred = scaler_pred.inverse_transform(y_pred_scaled)
```

Figure 4.19: Functions for Prediction and Unscaled Predicted Values

The line graph of the predicted and real close price (Scaled values) is plotted to evaluate the accuracy of price fluctuations predicted by LSTM model. Another line graph with residual value is prepared to analyze the predicted and real close price deviation.

4.3 Results of the LSTM Multivariate Model

This section discusses below findings that are recorded after running the LSTM model for 15 datasets of five cryptocurrencies with steps mentioned in Section 4.2:

1. Hyperparameters before and after optimization, RMSE score, and overall % improvement of training LSTM model
2. Line graph of Loss function (Mean square error)
3. Line graph of prediction v/s original scaled values for the test dataset
4. Line graph of prediction v/s original values with residual bar plot for Overall dataset

The output of hyperparameter tuning indicates the improvement of the LSTM model's accuracy. The high percentage of improvement is reported for tether coin (30-min dataset). The RMSE value and Loss function graph measure the error in predication after every epoch. The loss function shows the fitting of the model (Brownlee, 2017). The tether coin's 10-min dataset's RMSE value (0.0036) is the lowest among all the datasets. The comparison between fluctuation of the predicted and real price is done by the line graph of the unscaled values. Apart from Tether and USD coins, the fluctuation of price prediction is captured most accurately by the LSTM model for 10-min datasets. The percentage of the min and max residual value is calculated to understand the deviation in the real and predicated close price.

4.3.1 Bitcoin

The steps for Figures (Figure 4.20 to Figure 4.22) and Tables (Table 4.2 & Table 4.3) are common for 15 datasets and are mentioned at the start of Section 4.2.

The results of Bitcoin's before-and-after optimization are displayed in Table 4.2. The 30-minute frequency dataset (44%) has the higher percentage of improvement in performance, whereas the 10-minute (17%) frequency dataset has the lowest percentage of improvement.

Table 4.2: Result of LSTM Hyperparameter Tuning (Bitcoin)

Dataset	Train Length	Test Length	Before Optimization		After Optimization		%Improvement
			[neurons, batch_size, dropout]	RMSE	[neurons, batch_size, dropout]	RMSE	
bitcoin 10min	20918	5241	[16,64,0.2]	**0.0051**	[32, 64, 0.2]	**0.0043**	17.00%
bitcoin 30min	6940	1747	[16,64,0.2]	0.0119	[32, 32, 0.2]	0.0067	44.00%
bitcoin 60min	3446	873	[16,64,0.2]	0.0122	[32, 32, 0.1]	0.0091	25.00%

The RMSE score rises as the number of input samples decreases; for instance, the RMSE score for a 10-min dataset with 20918 training samples is lowest before and after optimization, coming in at 0.0051 and 0.0043, respectively. While the 60-minute dataset with 3446 samples has the higher RMSE score. Compared to other models in the training set, LSTM predicted closing prices for 10 minutes of the frequency with greater accuracy.

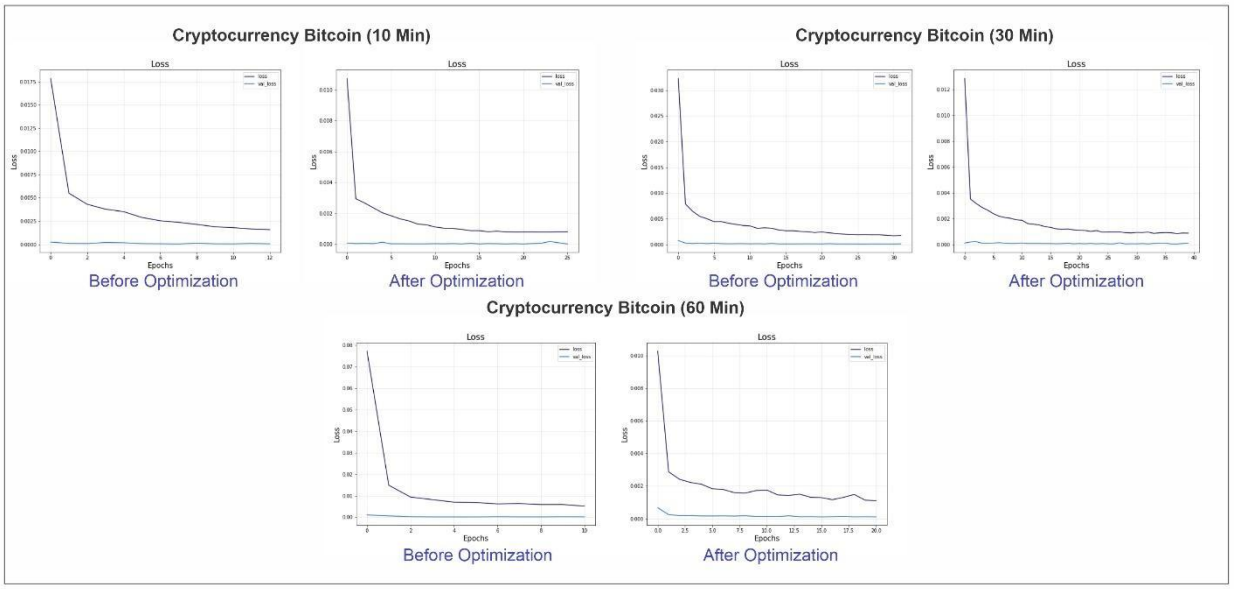


Figure 4.20: Line Graph - Loss Vs Epochs Before/After Hyperparameter Tuning (Bitcoin)

For all bitcoin datasets, the fitting of the model is increased significantly, and epochs are minimized after hyperparameters tuning, as seen in Figure 4.20. In contrast to other bitcoins datasets, LSTM is a good fit for the 10-mins frequency data because the loss function value is less than 0.001. For that dataset, the optimum hyperparameters are neurons 32, batch size

64, and dropout 0.2. It implies that predication for 10-mins dataset is closest to the actual value of close price for training data.

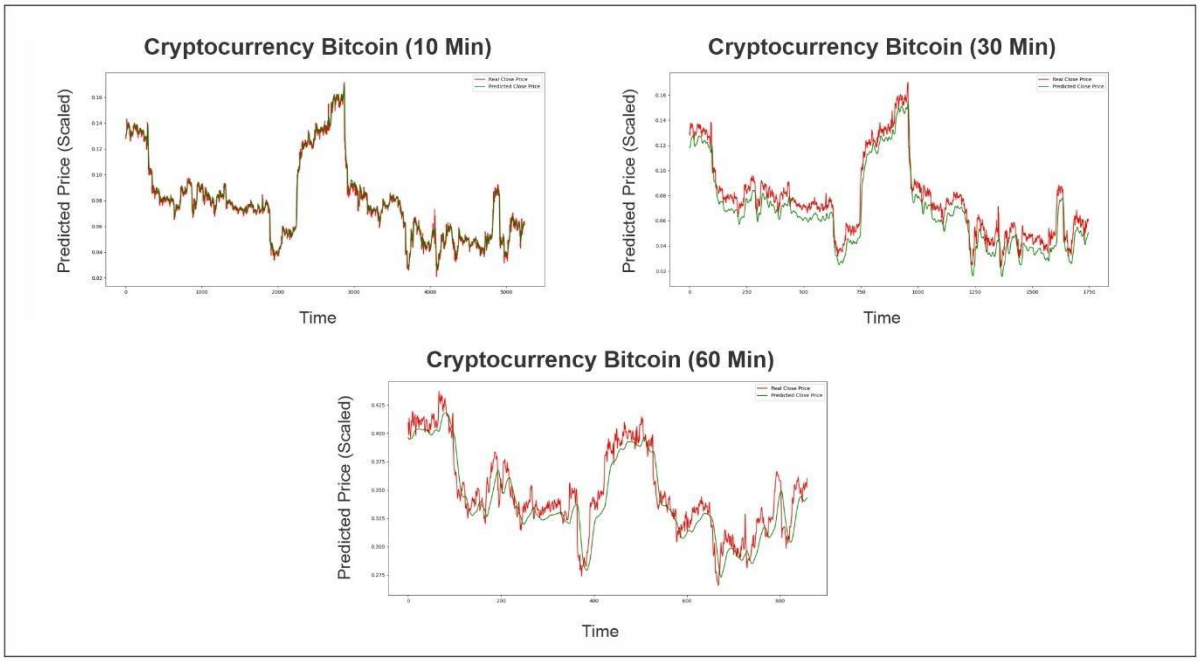


Figure 4.21: Predicated v/s Real Scaled Close Price (Bitcoin)

Above Figure 4.21 illustrates the comparison between the predicted and original values for test dataset using scale close price of the bitcoin. The accurate prediction of price trend is visible in 10-mins bitcoin graph as compared to other line graphs.

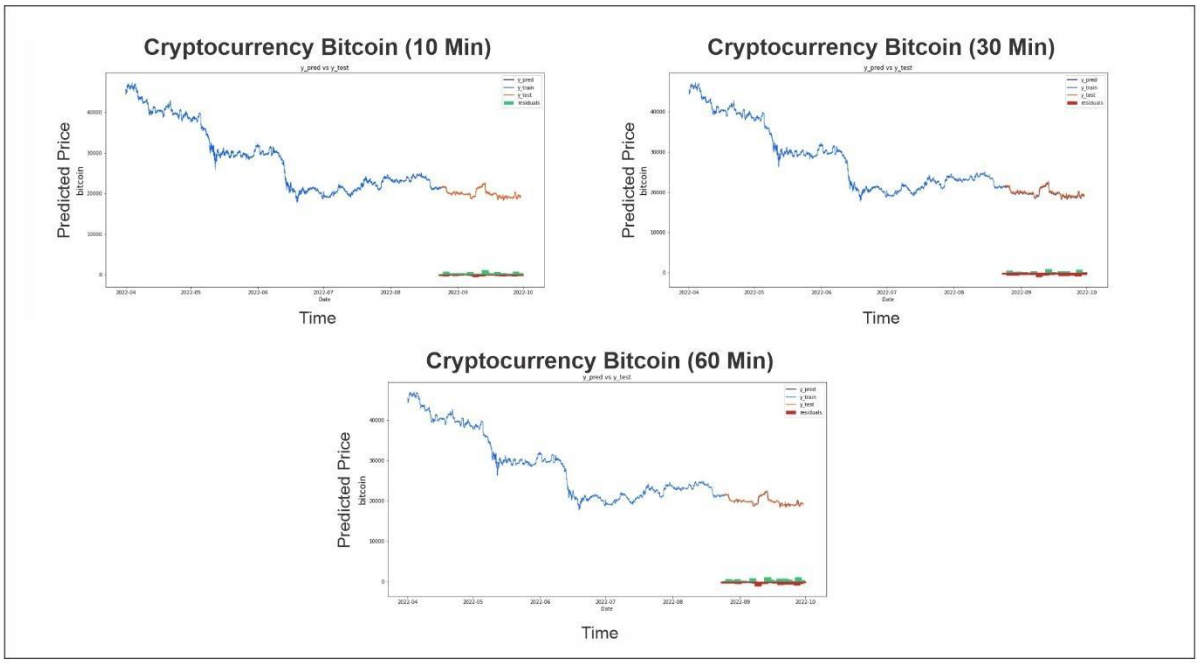


Figure 4.22: Line Graph of the Close Price with Bar Plot of the Residual (Bitcoin)

Figure 4.22 shows a line graph for the test and train datasets with a residual bar graph (the test dataset) at the bottom of the graph. According to the Table 4.3, the percentage minimum and maximum residual values for the 60-min bitcoin dataset are higher and lower than those

for the 10-min dataset. It displays that forecasted values are reasonably near to the real values for the 10-minute dataset, which are also shown in Figure 4.21 above.

Table 4.3: Percentage Min and Max Residual for Predicted Value (Bitcoin)

Dataset	% Min Residual	% Max Residual
bitcoin 10min	-2.97326	5.315676
bitcoin 30min	-5.49713	4.200551
bitcoin 60min	-5.74069	6.063606

In conclusion, the LSTM performance for the training dataset improved and lowered the RMSE score across all datasets after hyperparameter tuning, which also reflects in the prediction of the close price of bitcoin. For the 10-min dataset, the closing price prediction is more precise than other bitcoin datasets.

4.3.2 Binance

The steps for Figures (Figure 4.23 to Figure 4.25) and Tables (Table 4.4 & Table 4.5) are common for 15 datasets and are mentioned at the start of Section 4.2.

The outcomes of the hyperparameter optimization for the Binance coin are shown in Table 4.4. The average improvement performance across all datasets is around 47%.

Table 4.4: Result of Hyperparameter Tuning (Binance)

Dataset	Train Length	Test Length	Before Optimization		After Optimization		%Improvement
			[neurons, batch_size, dropout]	RMSE	[neurons, batch_size, dropout]	RMSE	
binance 10min	20557	5151	[16,64,0.2]	**0.0106**	[16, 32, 0.2]	**0.0055**	47.00%
binance 30min	6823	1718	[16,64,0.2]	0.0162	[16, 32, 0.2]	0.0086	47.00%
binance 60min	3390	859	[16,64,0.2]	0.0263	[32, 128, 0.2]	0.0136	48.00%

The RMSE value is highest for the 60-min dataset (0.0263 before and 0.0136 after optimization), It is the lowest for the 10-min Binance dataset (0.0106 before & 0.0055 after tuning). Compared to other datasets, the 10-min dataset's prediction error is low with new hyperparameters (neurons-16, brach_size-32 and dropout-0.2).

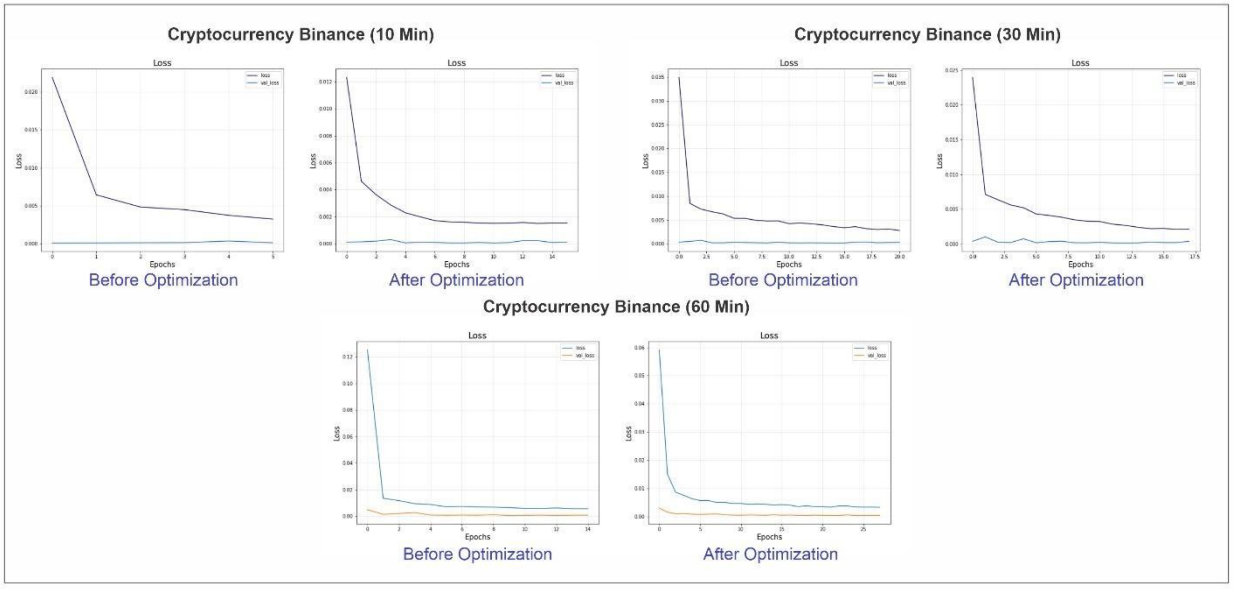


Figure 4.23: Line Graph - Loss Vs Epochs Before/After Hyperparameter Tuning (Binance)

According to Figure 4.23, The loss value decreased for all datasets after optimization with epochs except for the 10-min dataset of Binance. For 10-min dataset, 15 epochs were required to achieve the optimized MSE loss values, but its loss value is the lowest among all the datasets. As a result, the LSMT is a good fit for 10-min dataset, while underfit for the 60-min dataset as it has higher loss value for training.

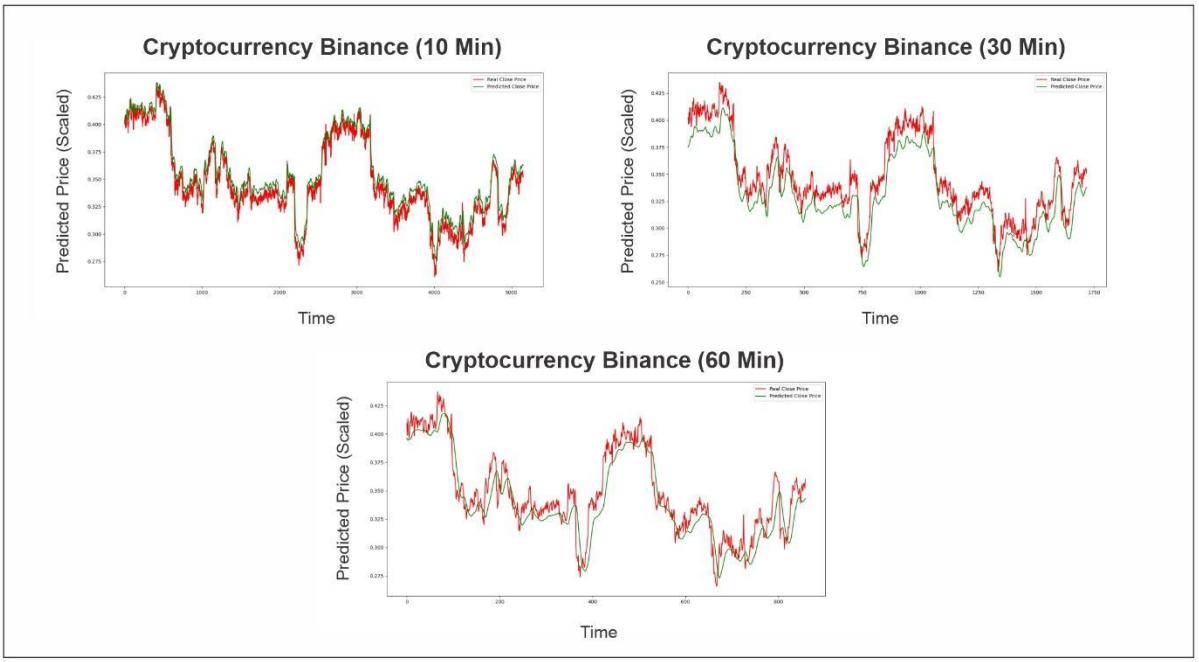


Figure 4.24: Line Graph of Predicated v/s Real Scaled Close Price (Binance)

The trend comparison of the closing price for the test dataset's original and predicted values is shown in Figure 4.24. The 10-min dataset is a close predicted line (green line) to real price (red line) with less deviation than the 30-min and 60-min datasets. The LSTM forecasted the price fluctuations for a 10-min dataset with higher accuracy.

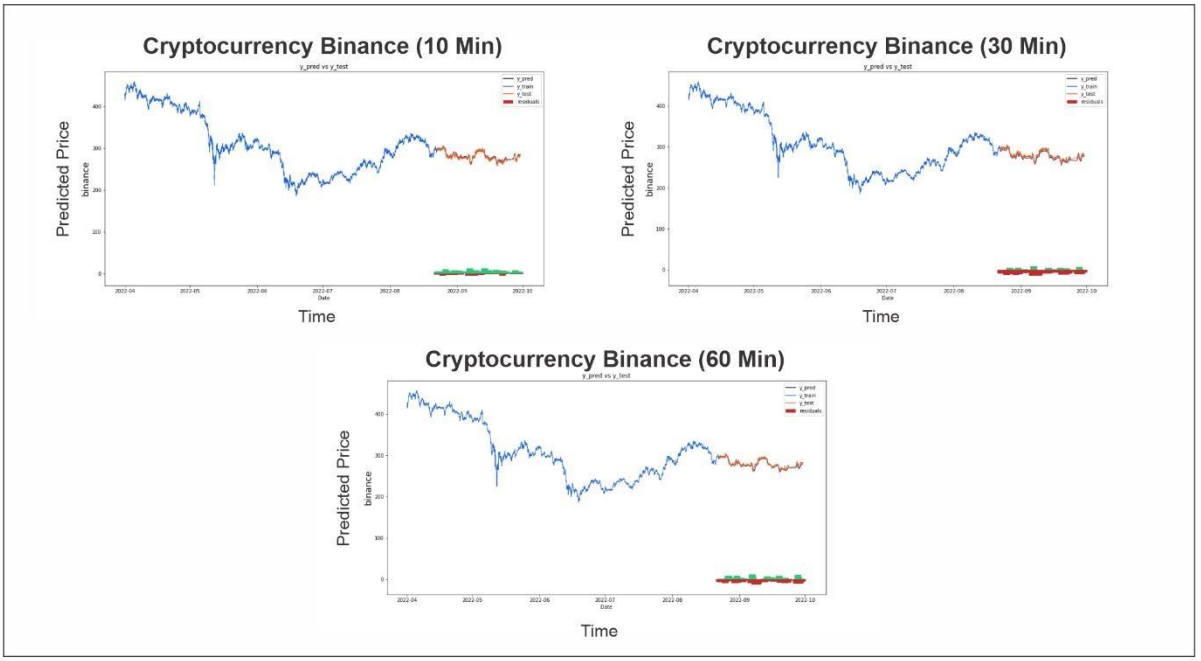


Figure 4.25: Line Graph of the Close Price with Bar Plot of the Residual (Binance)

The Figure 4.25 illustrate line graph of the close price for the overall dataset with bar chart of the residual values. The percentage of residual value range for 10-min data is almost lower as show in Table 4.5. Because of that the prediction line is very close to the real price for 10-mins test data as compared to other datasets.

Table 4.5: Percentage Min and Max Residual for Predicted Values (Binance)

Dataset	% Min Residual	% Max Residual
binance 10min	-1.8015	4.6065
binance 30min	-5.0418	3.4338
binance 60min	-4.7015	4.5350

Overall, The training dataset's performance is improved when the hyperparameter has been tuned. For the 10-min dataset, the closing price is predicted with higher accuracy, that has the lowest RMSE value during the training and test phase. The trend of the line graphs and the residual bar plot further strengthen that claim.

4.3.3 Ethereum

The steps for Figures (Figure 4.26 to Figure 4.28) and Tables (Table 4.6 & Table 4.7) are common for 15 datasets and are mentioned at the start of Section 4.2.

The LSTM performance result is presented in Table 4.6 for all datasets of the Ethereum. The efficiency of the model is improved after the optimization for all the datasets. A high improvement is reported for 30-min (61%) dataset followed by the 10-min (26%) and 60-min (19%) datasets.

Table 4.6: Result of Hyperparameter Tuning (Ethereum)

Dataset	Train Length	Test Length	Before Optimization		After Optimization		%Improvement
			[neurons, batch_size, dropout]	RMSE	[neurons, batch_size, dropout]	RMSE	
ethereum 10min	20918	5241	[16,64,0.2]	**0.0072**	[32, 64, 0.2]	**0.0053**	26.00%
ethereum 30min	6940	1747	[16,64,0.2]	0.0221	[32, 128, 0.1]	0.0087	61.00%
ethereum 60min	3446	873	[16,64,0.2]	0.0124	[16, 32, 0.1]	0.0101	19.00%

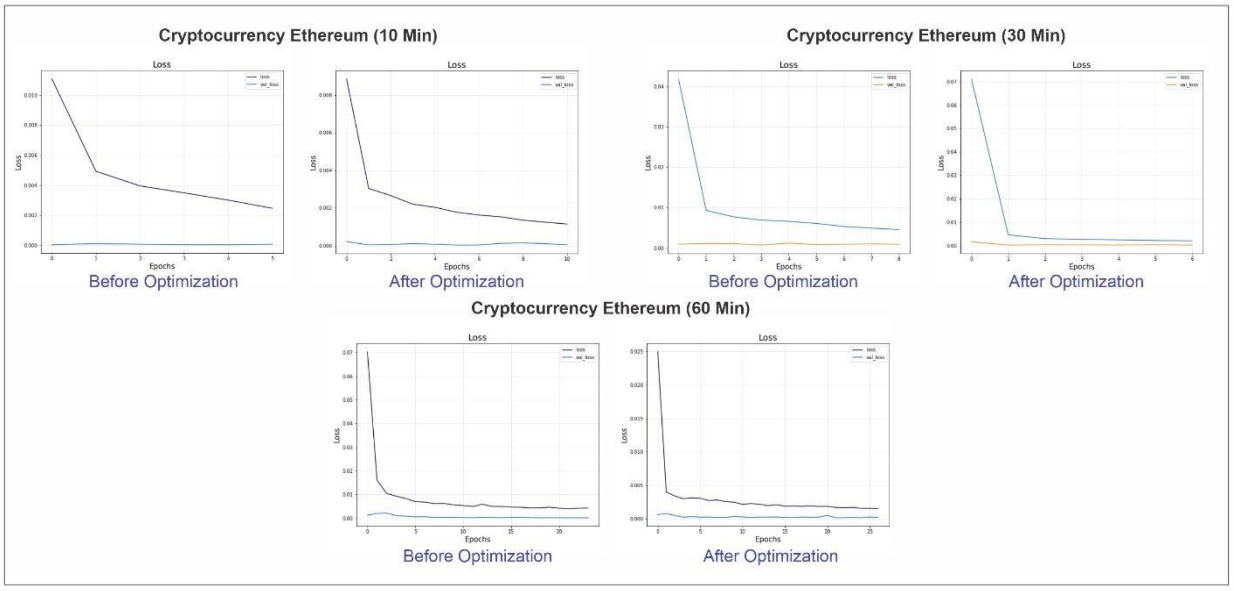


Figure 4.26: Line graph of Loss Vs Epochs Before/After Hyperparameter Tuning (Ethereum)

The RMSE score of the LSTM model for the 10-mins training dataset is the lowest before (0.0072) and after (0.0053) hyperparameter tuning than 30-min (0.0221 & 0.0087) and 60-min (0.0124 & 0.0101) dataset. As shown in the line graph of MSE loss function, the value of the loss is decreased for 10-min (approx. 0.001) and 60-min (approx. 0.0015) after the optimization, which means the LSTM model is good for both datasets. The required epochs increase to achieve efficiency for both datasets. The RMSE score is lower for the 10-minute dataset, so the LSTM model performed better for 10-min than the 60-min dataset during training.

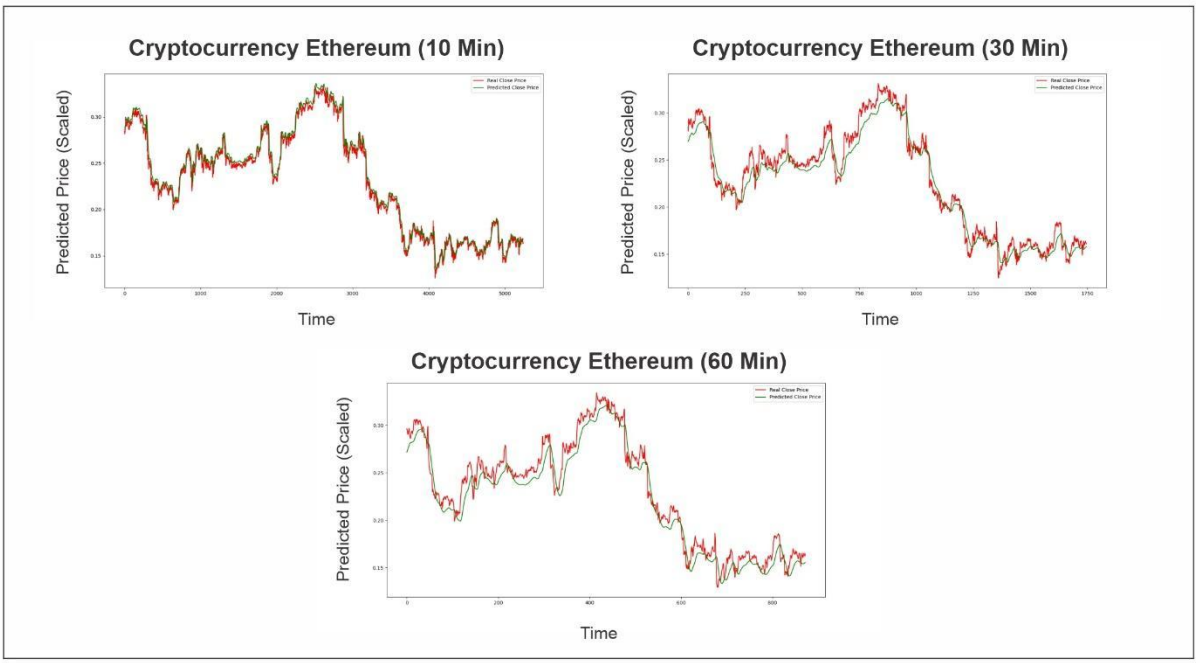


Figure 4.27: Line Graph of Predicated v/s Real Scaled Close Price for Ethereum

For 30-min and 60-min dataset, the clear distance is visible between both lines in Figure 4.27.

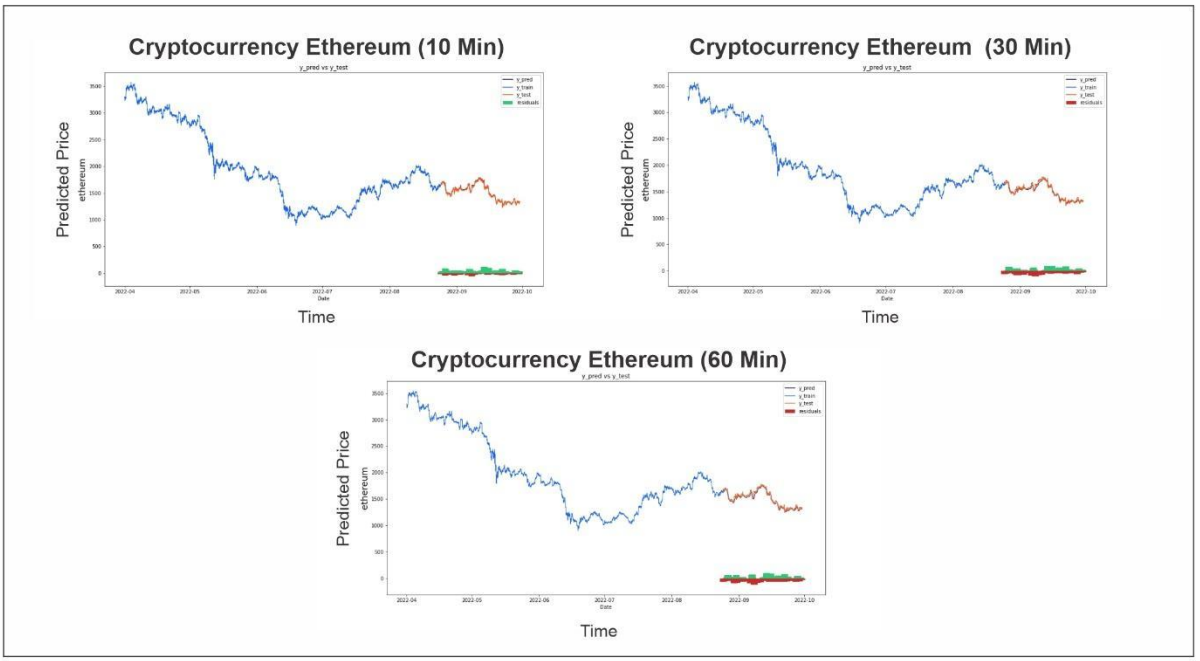


Figure 4.28: Line Graph of the Close Price with Bar Plot of the Residual (Ethereum)

As shown in the Figure 4.28, the line graph of the predicted values for 10-mins are closer to the real value and the fluctuation of the price is accurately predicted by the LSTM model.

Table 4.7: Percentage Min and Max Residual for Predicted Values (Ethereum)

Dataset	% Min Residual	% Max Residual
ethereum 10min	-3.6333	7.2692
ethereum 30min	-6.2667	7.2140
ethereum 60min	-7.6475	6.1576

For 10-mins data, the %- Min-Max residual value range is less than other datasets. Among all, the perform of LSTM model is best for 10-min dataset.

4.3.4 USD coin

The steps for Figures (Figure 4.29 to Figure 4.31) and Tables (Table 4.8 & Table 4.9) are common for 15 datasets and are mentioned at the start of Section 4.2.

Compared to other cryptocurrencies, the USD coin's performance improvement is minimal even after the hyperparameter tuning of the LSTM model. According to below Table 4.8, the maximum improvement is recorded for the 60-min (8%) dataset and the lowest for the 10 min dataset (2%).

Table 4.8: Result of Hyperparameter Tuning (USD)

Dataset	Train Length	Test Length	Before Optimization		After Optimization		%Improvement
			[neurons, batch_size, dropout]	RMSE	[neurons, batch_size, dropout]	RMSE	
USD 10min	20545	5148	[16,64,0.2]	**0.006**	[32, 32, 0.2]	**0.0059**	2.00%
USD 30min	6823	1718	[16,64,0.2]	0.0162	[32, 32, 0.2]	0.0153	5.00%
USD 60min	3390	859	[16,64,0.2]	0.0209	[32, 128, 0.1]	0.0193	8.00%

The RMSE value is significantly higher for 30-min (0.0162 & 0.0153) and 60-mins (0.0209 & 0.0193) datasets as compared to 10-mins (0.006 & 0.0059) datasets irrespective of the hyperparameters optimization of LSTM model. It shows training performance of LSTM for the 10-min data is more efficient and highly accurate than other datasets.

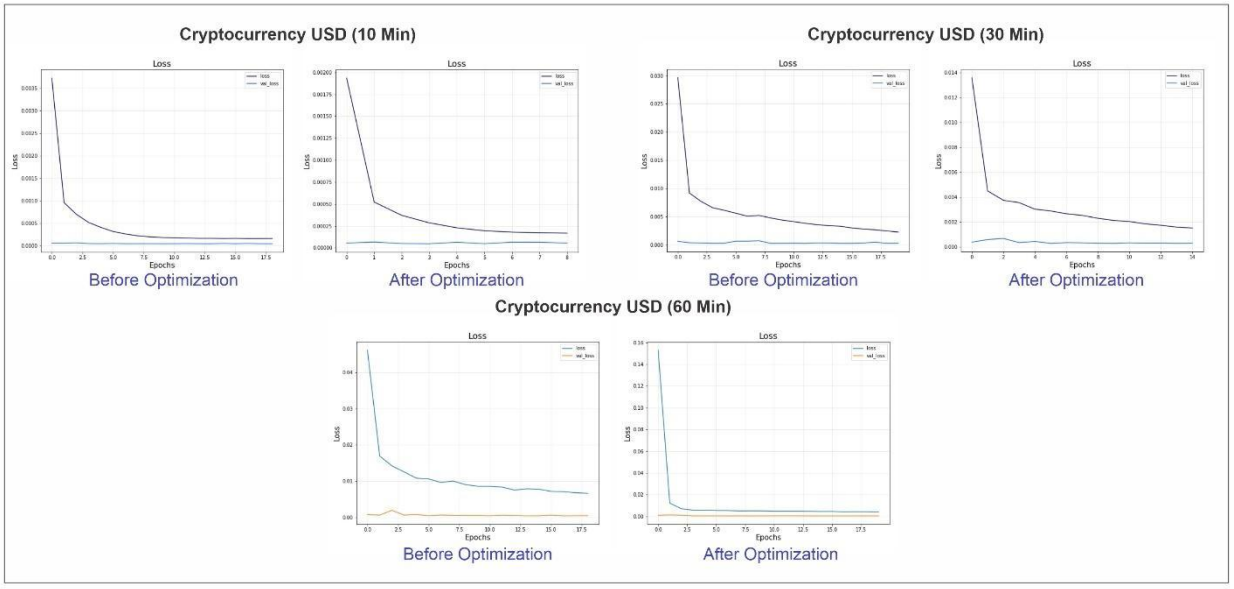


Figure 4.29: Line Graph - Loss Vs Epochs Before/After Hyperparameter Tuning (USD)

Figure 4.29 illustrates the line graph of the Loss values against the epoch of the LSTM model. There are no significant changes in the Loss values after parameter tuning. It is aligned with the percentage of the improvement displayed in table 4.8. The epochs for 10-min data are reduced from 18 to 8 after optimization of the LSTM, and loss value is below 0.00025 for model training. As a result, the LSTM model is a good fit for 10-min data.

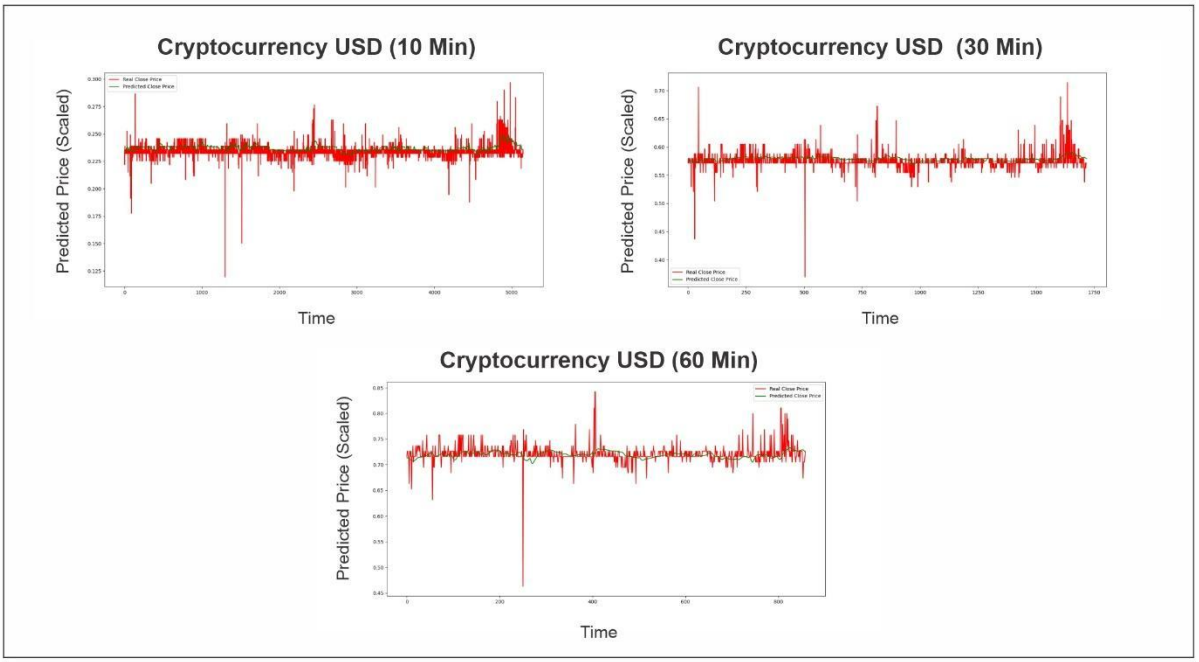


Figure 4.30: Predicated Scaled Close Price v/s Original Scaled Close Price (USD)

According to Figure 4.30, The line graph of scaled closing price values for the LSTM model's prediction output captures the variation between the predicted and actual close price. The line graph of the 30-minute and 60-minute datasets does not adequately predict the close price's patterns. The LSTM model is occasionally predicting and complete trends in the opposite

direction for 60 minutes. The 10-min dataset has the highest level of trend prediction accuracy across all datasets.

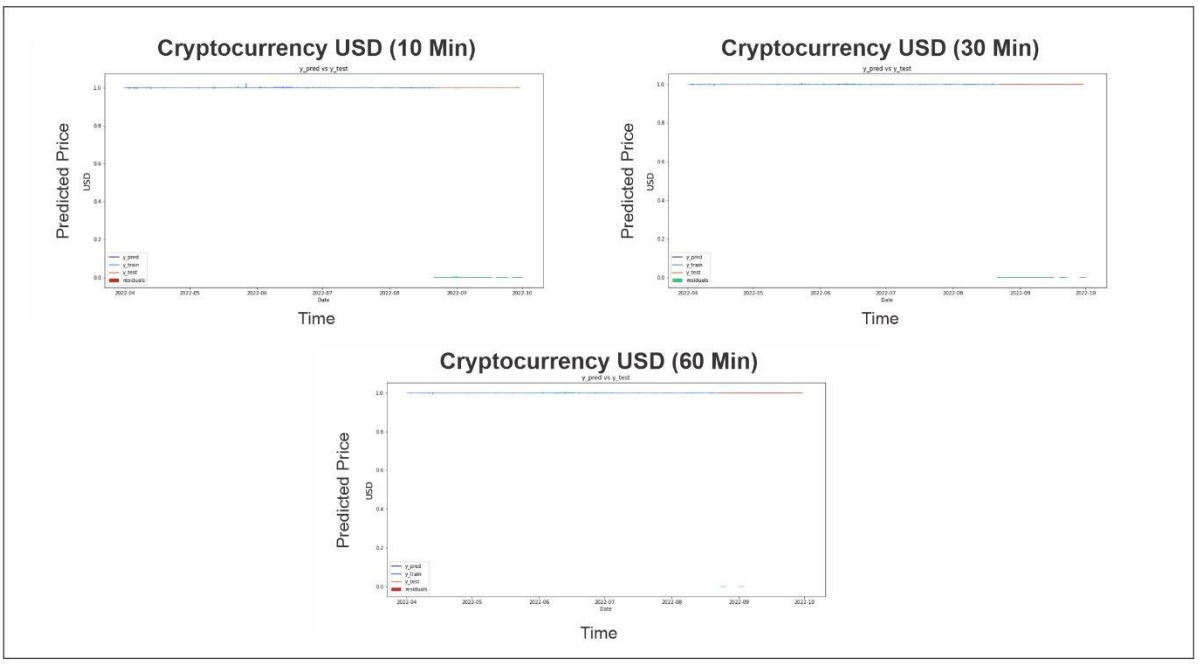


Figure 4.31: Predicated Close Price and Residuals Values (USD)

As shown in Figure 4.31, the training and test dataset's line graph for forecasted and real-time pricing is straight lines with minor changes. The range of prices that USD coins had between March 2022 and September 2022 is the possible cause of it. For the specified interval, the actual closing price of the US coin ranges from 0.9975 to 1.0016. it is preferable to refer to the line graph of scaled values and the percentage of mix-max residual values (Refer to Table 4.9) to understand the performance of the LSTM for USD coin.

Table 4.9: Percentage Min and Max Residual for Predicted Values (USD)

Dataset	% Min Residual	% Max Residual
USD 10min	-0.1524	0.1671
USD 30min	-0.1552	0.2522
USD 60min	-0.1143	0.2429

In comparison to the 30-min and 60-min datasets, the percentage of Min and Max residual values is lower for the USD 10-min dataset. It shows that the predicted values based on 10-min data are more in line with the USD's actual close price. In addition, the training dataset's RMSE score, and loss values show that the LSMT model most accurately predicts the close price for 10 minutes.

4.3.5 Tether

The steps for Figures (Figure 4.32 to Figure 4.34) and Tables (Table 4.10 & Table 4.11) are common for 15 datasets and are mentioned at the start of Section 4.2.

Table 4.10 contains the results of the hyperparameter optimization for the tether coin. The LSTM model's improvement percentage is highest for 30-min data (71%) and lowest for 10-min (61%) and 60-min (10%) data after parameter tuning.

Table 4.10: Result of Hyperparameter Tuning (Tether)

Dataset	Train Length	Test Length	Before Optimization		After Optimization		%Improvement
			[neurons, batch_size, dropout]	RMSE	[neurons, batch_size, dropout]	RMSE	
tether 10min	20568	5154	[16,64,0.2]	**0.0092**	[32, 32, 0.2]	**0.0036**	61.00%
tether 30min	6826	1718	[16,64,0.2]	0.028	[16, 64, 0.2]	0.0082	71.00%
tether 60min	3390	860	[16,64,0.2]	0.0068	[32, 64, 0.1]	0.0061	10.00%

As shown in the Table-4.10, the RMSE score is maximum for 30-min data before and after that hyperparameter tuning that is 0.028 and 0.0082 respectively, Whereas it is lowest for 60-min data (0.0068) before optimization and 10-min data (0.0036) after optimization. It represents the LSTM model prediction accuracy as higher for the 10-min dataset after optimization as compared to other datasets.

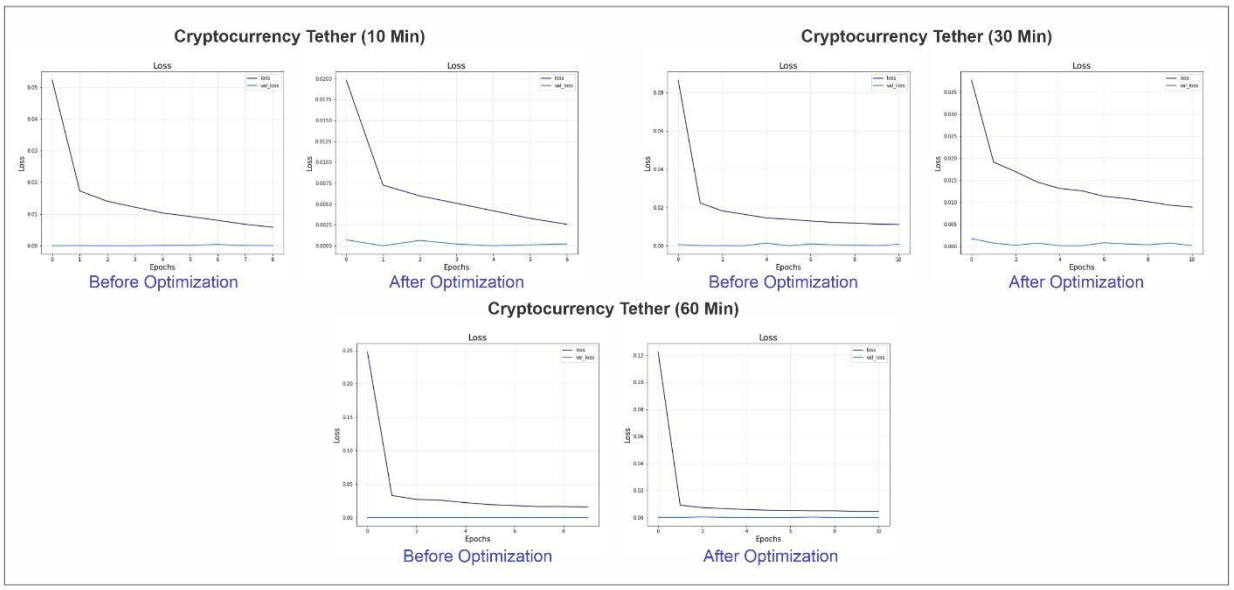


Figure 4.32: Predicated Close Price and Residuals Values (Tether)

Figure 4.32 illustrates the loss value is decreased for the training dataset with epochs for all the datasets of tether coin except the 30-min dataset after optimization of the LSTM model. For 30-min data, there is no change in epochs (10), while the loss values decrease after parameter tuning. The LSTM model is a good fit for 10-min data as the loss value (below 0.0025) is the lowest among all datasets.

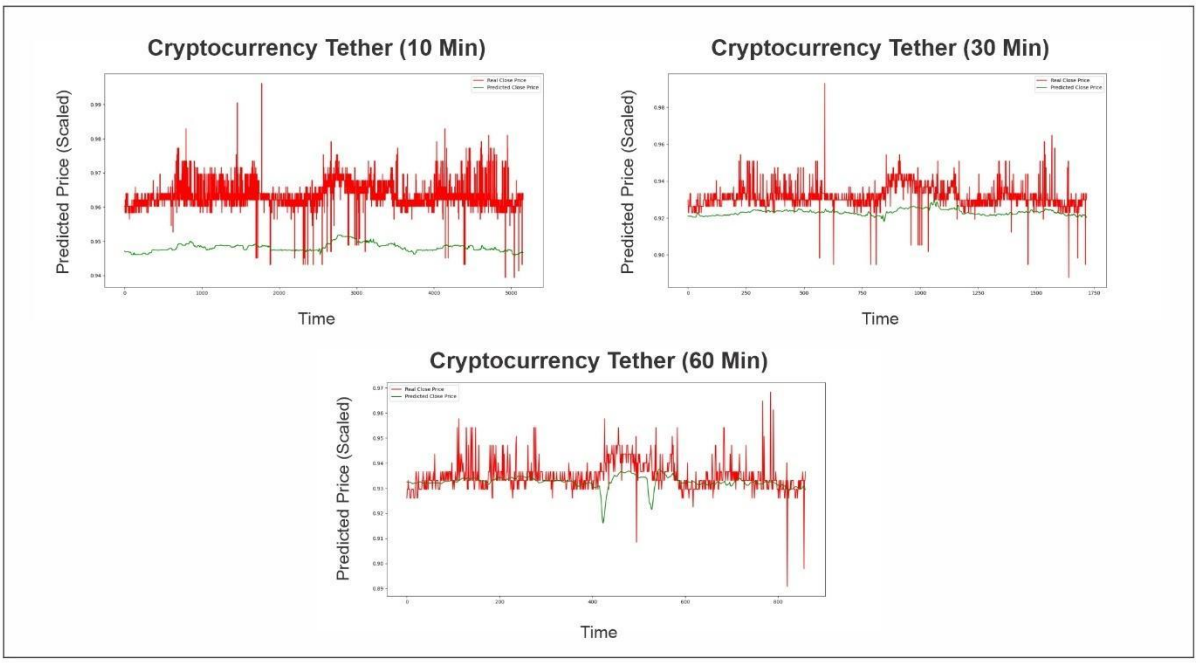


Figure 4.33: Predicated v/s Original Scaled Close Price (Tether)

As shown in Figure 4.33, The predicted close price of the 10-min dataset can capture the trend of the close price better as compared to the 30-min and 60-min datasets. In line graph, the inverse trend of tether price is captured for 30-min and 60-min datasets for predicated values (refer to green line). The prediction model can capture better trends for 10 min dataset.

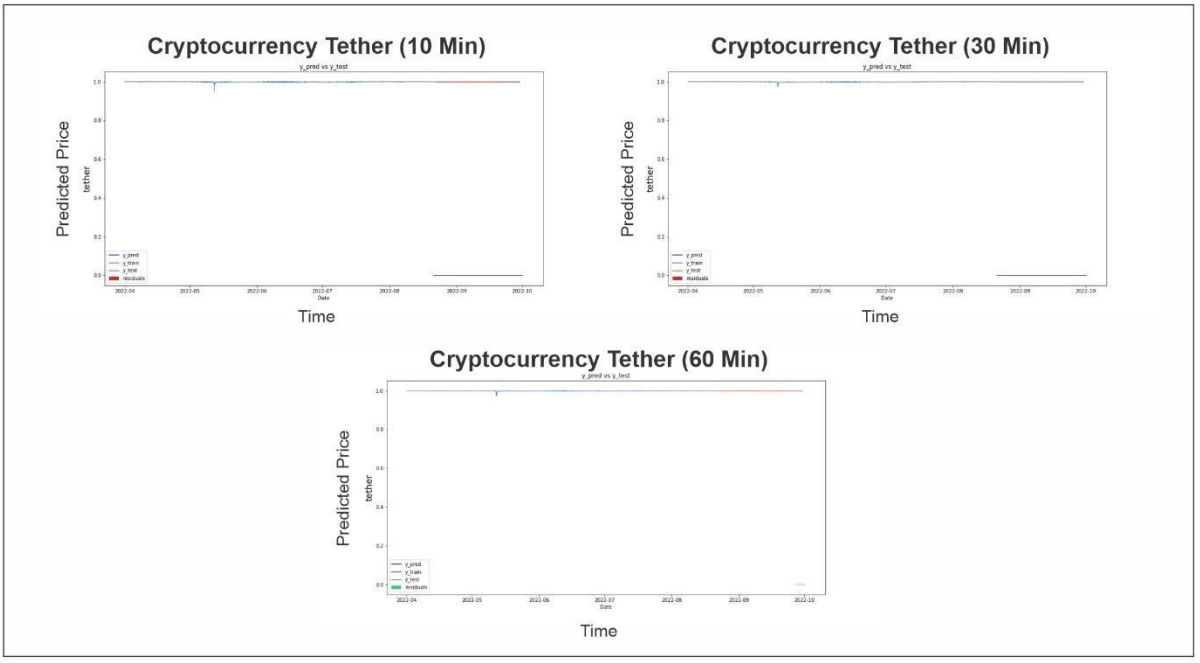


Figure 4.34: Predicated Close Price and Residuals Values (Tether)

Figure 4.34 is difficult to interpret since the graph is a straight line for both real and forecasted values. The close price for March to September 2022 could be the cause. For the collected dataset, the closing price value is observed between 0.9492 and 1.0668.

Table 4.11: Percentage Min and Max Residual Values (Tether)

Dataset	% Min Residual	% Max Residual
tether 10min	-0.2514	0.0425
tether 30min	-0.1973	0.0973
tether 60min	-0.1078	0.1113

The scaled line graph (Figure 4.33) and the percentage change in the residual values (Table 4.11) are preferable references to evaluate the performance of the LSTM model. The change in the percentage residual values is lower for the 10-mins dataset (-0.25% to .042%) than other datasets, which concludes that the predicted values are comparatively accurate, and LSTM performed better prediction for 10-min data.

In conclusion, the performance of the LSTM model is the best for 10-mins dataset for all the cryptocurrencies in terms of the RMSE score, Loss function (MSE), residual value, and trend of the predicted close price for training and test datasets as compared to 30-min and 60-mins dataset. In the next phases of the experiment, the close price of the 10-min dataset have been used for the portfolio construction and performance analysis of the portfolios.

4.4 Portfolio Construction and Performance Analysis

The Experiment of the four portfolio construction approaches that were mentioned in Section 3.3 and their result are discussed in this section. The close price of each cryptocurrency is input, while the weight of each crypto is the output of the portfolio construction methods. Here, the weight is the percentage of the investment for cryptocurrency in the single portfolio. Section 4.2 provides the foundation for the predicted closing prices of all crypto currencies. After that, how to use the calculated weight for all coins to determine the performance metrics is described for all portfolios. To undertake a comparative analysis of portfolio strategies and their performance over different time periods and the close price trend, portfolios are formed using historical daily data of 2021. Because of up and down trend of the close price as show in Figure 4.1.2.1 - 4.1.2.4 of EDA analysis, the historical data of 2021 have been selected. The historical data of 2021 is a subset of the last 5 years (2018-2022) of cryptocurrency data. It was collected for exploratory data analysis at the time of data collection (Section 4.1.1).

4.4.1 Data Preparation for Portfolio Construction

For portfolio creation, Firstly, Data preparation requires step because the output of Section 4.2 is the Intraday close price (10mins frequency of trainset and forecasted values), while daily close price is required as input. Secondly, the close price of all cryptocurrencies is in different CSV files for the experiment dataset and historical dataset of 2021. There are two major objectives of the data preparation 1) Converting intraday prices into daily prices, and 2) combining the close prices of all crypto assets into a single output file.

Converting intraday prices into daily prices

The close prices from the training dataset and forecasted prices have been loaded in the dataframe from the respective CSV files. For each cryptocurrency, certain data pre-processing operations have been performed to remove NA values from predicated dataframe and filter the training dataset. The last price of the day has been fetched using *the sort (), groupby(),* and *agg()* functions of the dataframe, and the date index reset using *index_reset() function* from both datasets. Refer to Figure 4.35 to understand the above process for USD coin.

```
USD_predicted_data = USD_predicted_data.drop(['y_train'], axis=1)
USD_predicted_data['Date'] = pd.to_datetime(USD_predicted_data['Date'])
USD_predicted_data = USD_predicted_data.dropna()
USD_predicted_data = USD_predicted_data
    .sort_values(['Date', 'y_pred'])
    .groupby(USD_predicted_data['Date'].dt.date)['y_pred']
    .agg(['last'])
    .reset_index()
```

```
USD_data['Date'] = pd.to_datetime(USD_data['Date'], errors='coerce')
USD_data=USD_data.sort_values(by="Date", ascending=True)
USD_data = USD_data.dropna()
USD_data = USD_data
     .sort_values(['Date', 'Close'])
     .groupby(USD_data['Date'].dt.date)['Close']
     .agg(['last'])
     .reset_index()
USD_data = USD_data.loc[USD_data['Date'] <= datetime.date(year=2022,month=8,day=21)]
```

Figure 4.35: Derived Daily Price from the Intraday Close Price for USD Coin

Combining the close prices of all crypto assets into a single output file

By executing an inner join on the index column (Date) using the *merge()* function, all coin dataframes are combined into a single dataframe and saved into an output csv file using the *to_csv()* method. Similarly, all coins' history data from 2021 is combined and saved to an output file. Refer to Figure 4.36 for merge operation of final data frame for train and forecasted close price.

```
crypto_predicated_data_1 = pd.merge(USD_data_merged,ethereum_data_merged, how='inner',on ='Date')
crypto_predicated_data_2 = pd.merge(crypto_predicated_data_1,binance_data_merged, how='inner',on ='Date')
crypto_predicated_data_3 = pd.merge(crypto_predicated_data_2, bitcoin_data_merged, how='inner',on ='Date')
crypto_predicated_data_4 = pd.merge(crypto_predicated_data_3,tether_data_merged, how='inner',on ='Date')
```

Figure 4.36: Merge and Save Operation for Close Price of Experiment Price Dataset

The daily close price of the historical dataset and the experiment dataset for all cryptocurrencies are saved in corresponding CSV files as the final output. Both datasets are used while building a portfolio and analyzing performance.

Note: *Going forward, The close price of the training data and forecasted dataset is referred as experiment dataset which is output of the data prepare for portfolio construction and analysis.*

4.4.2 Mean-Variance Portfolio Method

The Mean-variance portfolio method is also known as the Modern Portfolio Theory (MPT). The aim of the MTP is to maximize the Sharpe ratio and minimize the mean variance of the portfolio to achieve expected returns for given level of the risk (Anon, 2017). Following steps are followed to implement the mean variance portfolio methods:

1. Calculate mean daily return and daily variance of return for all cryptocurrencies
2. Find the annual return and variance by annualizing the output of the step 1
3. Generate the random weights for all assets and derive the annual return, sharp ratio, and standard deviation
4. To find optimal portfolio, simulate 500000 portfolios using step 1 to 3 using Monte Carlo Simulation
5. Find the optimal portfolios which have max sharp ratio and min variance
6. Apply step 1 to 5 on Experiment and historical dataset from Section 4.4.1

The Sharpe ratio is defined as the expected return of the portfolio divided by the standard deviation of the portfolio, according to equation-3.5 in Section 3.3. The mean daily return and daily standard deviation of return are determined for all crypto currencies using the *pct_change(), returns.mean(), and cov()* functions. The mean expected annual return and variance are then obtained by simply annualizing the results to find the sharp ratio and mean variance. For the computation of the mean daily return and covariance of daily returns, see Figure 4.37.

```
data = pd.read_csv('./crypto_predicted_close_price.csv')
data.set_index('Date',inplace=True)
data.sort_index(inplace=True)
returns = data.pct_change()
#calculate mean daily return and covariance of daily returns
mean_daily_returns = returns.mean()
cov_matrix = returns.cov()
```

Figure 4.37: Daily Mean Return and Variance Calculation

The annual return and variance need to be computed for the weights of all cryptocurrencies in a mean-variance portfolio. The Monte Carlo simulation is run with 500000 distinct combinations to determine the portfolio's ideal weights for the highest returns and minimal risk. For each set of portfolios, the annual return, standard deviation, and sharp ratio are computed. Refer to Figure 4.38 to construct 5 lacs simulated portfolio using a Monte Carlo simulation.

```
def monte_carlo_simulate_portfolios(num_portfolios, mean_returns, cov, rf, tickers):
  #output = np.zeros((len(mean_returns)+3, num_portfolios))
  output = np.zeros((4+len(tickers)-1,num_portfolios))

  for i in range(num_portfolios):
      weights = np.random.random(len(tickers))
      weights /= np.sum(weights)
      annual_return, std_dev, sharpe_ratio = calc_portfolio_perf(weights, mean_returns, cov, rf)
      output[0,i] = annual_return
      output[1,i] = std_dev
      output[2,i] = sharpe_ratio
      #iterate through the weight vector and add data to results array
      for j in range(len(weights)):
          output[j+3,i] = weights[j]

  results_df = pd.DataFrame(output.T,columns=['ret','stdev','sharpe'] + [ticker for ticker in tickers])

  return results_df
```

Figure 4.38: Monte Carlo Simulation for Mean-Variance Portfolio

By using the *idmax()* function on the sharp ratio and the *idmin()* function on the variance columns of the resultant dataframe of the simulated portfolios, the best mean-variance portfolios are created. For a scatter plot of all simulated portfolios, see Figure 4.39. The green start represents the minimum variance portfolio, while the red star represents the maximum sharp ratio portfolio in the scatter plot.

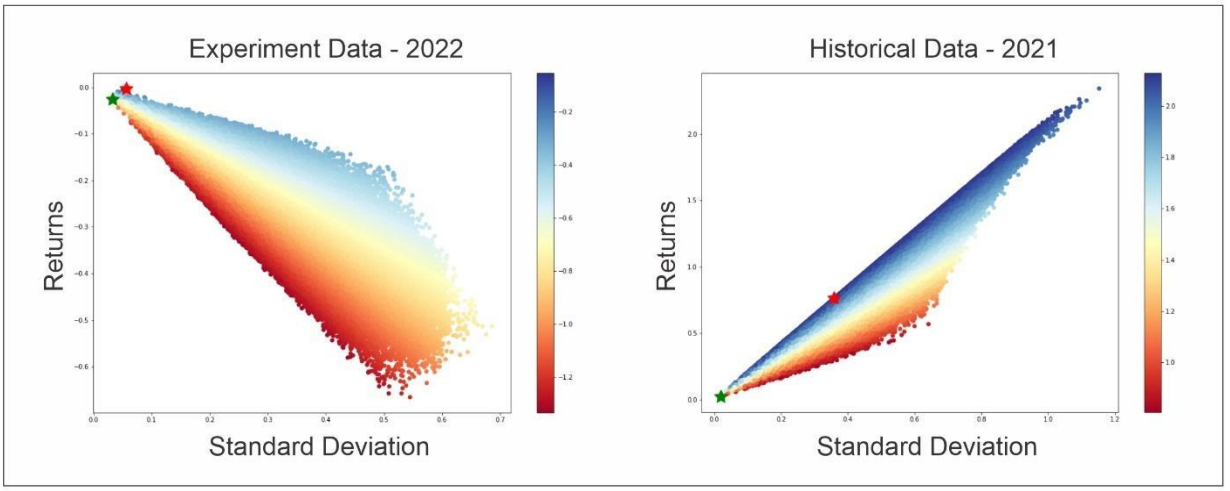


Figure 4.39: Scatter Plot of Simulated Portfolios

Using the above experiment, mean variance portfolios are produced for the historical dataset of 2021 as well as the experiment dataset. Table 4.12 lists the values of weights for the maximum sharp ratio and the minimum variance (volatile) portfolios with annual return, sharp ratio, and standard deviation.

Table 4.12: Derived Weights Using Mean-Variance Portfolio Method

data	portfolio	Weights					ret	stdev	sharpe
		USD	ethereum	binance	bitcoin	tether			
Mar-2022 to Sep-2022	max sharpe ratio	0.637461	0.007488	0.013587	0.006724	0.33474	-0.009786	0.049686	-0.196958
(With Forcasted Data)	min vol	0.187435	0.002884	0.007615	0.03234	0.769727	-0.02615	0.028603	-0.914254
Jan-2021 to Jan-2021	max sharpe ratio	0.633496	0.140884	0.218529	0.000787	0.006304	0.761721	0.357937	2.128087
(Historical Data)	min vol	0.634311	0.009955	0.002225	0.002256	0.351253	0.021699	0.018508	1.172369

According to Table 4.12, the portfolio returns of the experiment dataset are more likely to be zero or negative (-0.009 and -0.2615) than positive (0.76 and 0.021) for the historical data. As a result, the scatter plot of the simulated portfolios for the experiment dataset is pointing in the

opposite direction of that of the historical dataset. It can be seen in Figure 4.39. The downward trend of the major cryptocurrencies from March 2022 to September 2022 is one potential cause of the experiment dataset's portfolios' poor returns.

4.4.2 Hierarchical Risk Parity Method

Hierarchical risk method(HRP) is a recent advance portfolio construction method which use Hierachical clustering models in allocation. As part of the experiement, The implementation of Hierarchical risk parity from *pyportfolioopt* library is used. According to the *pyportfolioopt.readthedocs.io* documentation, the Hierarchical risk parity Method work as mentioned below (pyportfolioopt.readthedocs.io, n.d.):

1) From all the given assets, form distance matrix using correlations among all assets
2) Cluster the assets in tree using Hierarchical clustering with help of distance matrix derived from the step 1
3) Form a means variance portfolio between two assets within clusters
4) Optimally combine the min-portfolios from all nodes by iterating through each level.

To implement the Hierarchical risk parity method, the expected return and clean weight for all the assets are calculated. The expected return is calculated using the *returns_from_prices()* method of *expected_return* package, while clean weights for HRP portfolio is derived with the help of *optimize()* and *clean_weights()* methods of *HRPOpt* package from the *PyPortfolioOpt* library. See in Figure 4.40.

```
hrp = HRPOpt(rets)
hrp.optimize()
weights = hrp.clean_weights()
weights
```

```
OrderedDict([('USD', 0.01124),
             ('ethereum', 0.0),
             ('binance', 0.0),
             ('bitcoin', 0.0),
             ('tether', 0.98874)])
```

Figure 4.40: Output Weight of Hierarchical Risk Method for Experiment Dataset

A historical closing price dataset and an experiment are both used to build clean weights using the Hierarchical Risk Parity approach. Table 4.11 displays the results of both experiments. The *plot_dendrogram()* function of the *PyPortfolioOpt.plotting* package is used to visualize the created clusters of crypto assets. Additionally, the pie chart of clean weights is drawn using the *plot()* function. Figures 4.41 and 4.42 display the dendrogram plot and pie charts for the two datasets, respectively.

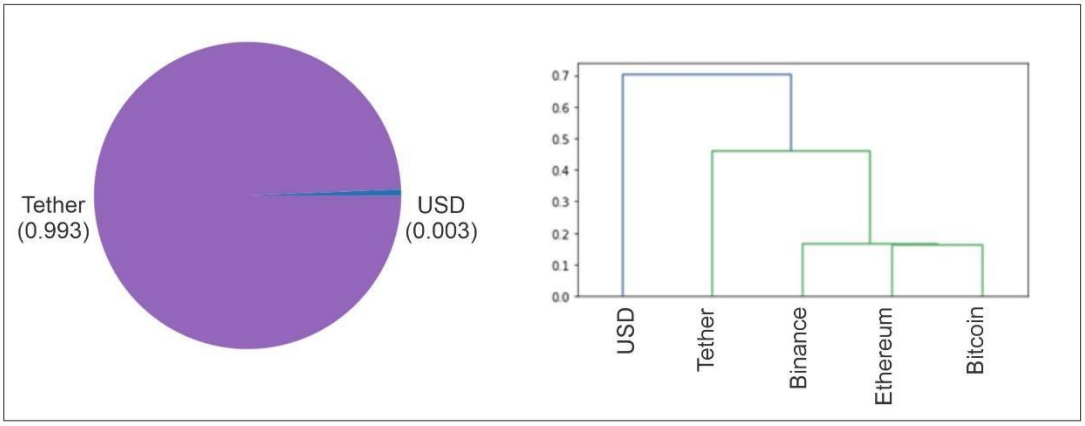


Figure 4.41: Pie Chart and Dendrogram Plot for Experiment Data

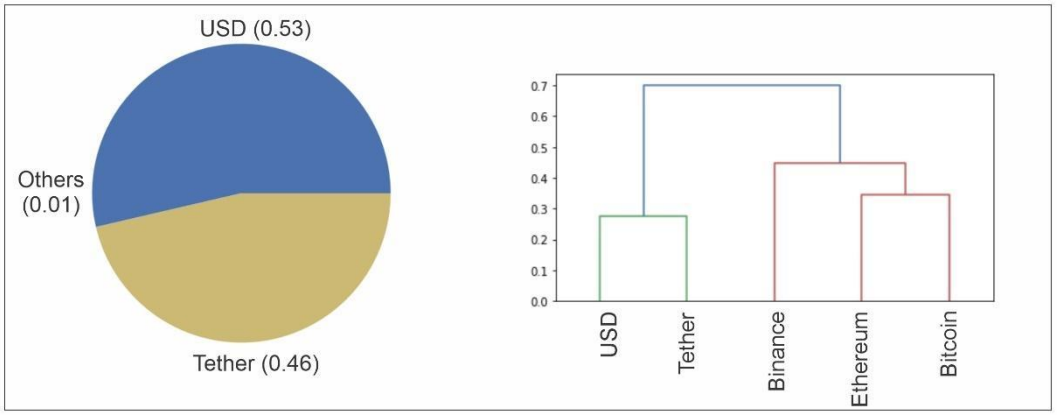


Figure 4.42: Pie Chart and Dendrogram Plot for Historical Data of 2021

The Ethereum, Binance, and bitcoin created clusters, as shown in Figure 4.41 for the experiment dataset, but there is no similarity between USD and tether, so no cluster was created for those assets. According to Figure 4.42, two clusters have been established: one for Ethereum and Bitcoin and another for USD and Tether for the historical dataset. However, there is no cluster for Binance Coin.

Table 4.13: Derived Weights Using Hierarchical Risk Parity

data	portfolio		Weights			
		USD	ethereum	binance	bitcoin	tether
Mar-2022 to Sep-2022 (With Forcasted Data)	Hierarchy risk parity	0.00636	0.00000	0.00000	0.00000	0.99353
Jan-2021 to Dec-2021 (Historical Data)	Hierarchy risk parity	0.53620	0.00012	0.00010	0.00020	0.46337

According to Table 4.13, no allocation is made for Ethereum, Binance, and bitcoin, while the USD and tether assets are given respective weights of 0.00636 and 0.99353 for the experiment dataset of 2022. Tether dominates the asset allocation for the experiment dataset, which is also shown in the pie chart. While for historic closing price data of 2021, tether and

USD are distributed between 0.53 and 0.46, while the remaining assets are allocated between 0.00010 and 0.00020. As seen in Figure 4.42, it also fit with the group established in the historical dataset's dendrogram and pie chart.

4.4.3 Kelly's Criteria

The Kelly's criteria focus on allocation of risk to few assets, and it is considered as too risk. To mitigate that, the risk wagering fraction should be introduced, so Investor can invest the capital to other risk-free assets (Kirschenmann, 2022). As part of the experiment, the wagering fraction is considered.

The close price of the assets is loaded into the dataframe from CSV files. The *pct_change() method is used* to derive the daily and annual returns of the assets. The *cov()* function is applied to the annual return and calculates annual co-variance. The weights of a portfolio are calculated by applying Kelly's criteria with help of the wagering fraction (0.50), and covariance matrix. Refer to Figure 4.43 for Kelly's criteria function.

```
def kelly_optimize_unconstrained(M:pd.DataFrame, C:pd.DataFrame)->pd.DataFrame:
    "calc unconstrained kelly weights"
    results = np.linalg.inv(C) @ M
    kelly = pd.DataFrame(results.values, index=C.columns, columns=['Weights'])
    return kelly
```

Figure 4.43: Kelly's Criteria Function

The output of the Kelly's criteria method for both datasets are recorded in the below Table 4.14.

Table 4.14: Derived Weights Using Kelly's Criteria

data	portfolio		Weights			
		USD	ethereum	binance	bitcoin	tether
Mar-2022 to Sep-2022 (With Forcasted Data)	**Kelly's Criteria**	**0.01000**	**0.00000**	**0.00000**	**0.00000**	**0.49000**
Jan-2021 to Dec-2021 (Historical Data)	Kelly's Criteria	0.37000	0.00000	0.00000	0.00000	0.13

The weight value of Binance, bitcoin, and Ethereum is zero for both datasets, which indicates those assets are risky assets and no weight allocation is done for the portfolio. The portfolio weights are allocated to tether (0.49 & 0.13) and USD coin (0.010 & 0.37) for both datasets as per wager fraction.

4.4.4 Equal Weight Portfolio Method

The Equal weight Portfolio Method is also known as naïve asset allocation method. The allocation of asset is done using equation 5 of section 3.3 for equal weight portfolio. There are five cryptocurrencies for portfolio and allocation is done as follow:

$W_{i(weight)}$ = 1/5 = 0.20 for each crypto asset.

For the experiment and historical datasets, The equal weights are assigned and accordingly the performance analysis is be done.

4.4.5 Portfolio Performance Analysis

The metrics outlined in section 3.3 constitute the basis of the performance analysis of the portfolio. With the help of the portfolio weights and close price of all crypto coins, metrics like volatility, annual return, Sortino ratio, and sharp ratio can be calculated.

The dataframe is loaded with the dataset's close price, and the *df/df.shift(1)* function is used to determine the log return. A list contains weights of portfolios that are derived in Section 4.4. The following metrics are produced for each portfolio as iterate through the list of their holdings: mean, downside standard deviation, annual volatility, annual sharp ratio, drawdown, annual return, Sortino ratio, and volatility skewness. For the computation of each measure, see Figure 4.44.

```
# daily return of the portfolio based on a given set of weights
df['portfolio_ret'] = df.iloc[:,0]*weights[0]+df.iloc[:,1]*weights[1]+df.iloc[:,2]*weights[2]+df.iloc[:,3]*weights[3]+df.iloc[:,4]*weights[4]

# Calculating Mean
E = df['portfolio_ret'].mean()

# drawdown
comp_ret = (df['portfolio_ret']+1).cumprod()
peak = comp_ret.expanding(min_periods=1).max()
dd = (comp_ret/peak)-1
max_drawdown = dd.min()
# Annualizing Mean

E_AN = E * N

n_days = df['portfolio_ret'].shape[0]
# Annulized return
annualized_return = (df['portfolio_ret']+1).prod()**(N/n_days) - 1

# Volatility ratio
annualized_volatility = df['portfolio_ret'].std()*N**0.5

# Sharpe ratio
annualized_sharpe = (annualized_return - rf) / annualized_volatility

# Calculating Downside Standard Deviation
mean = E * N -rf
std_neg = df['portfolio_ret'][df['portfolio_ret']<0].std()*np.sqrt(N)

# Calculating Upside Standard Deviation
std_pos = df['portfolio_ret'][df['portfolio_ret']>=0].std()*np.sqrt(N)

# Calculating Volatility Skewness
VS = std_pos/std_neg

# Sortino
Sortino = mean/std_neg
```

Figure 4.44: Calculation of Portfolio Performance Measure

The performance metrics have been calculated for historical and experiment dataset and the results are displayed in the Figure 4.45 and Figure 4.46 for experiment and Historical data respectively.

	Mean	Downside SD	Upside SD	Volatility Skewness	Sortino	Annualized Return	Annualized Sharpe Ratio	Annualized Volatility	Draw Down
Equal Weight Portfolio	-0.401426	0.290026	0.196507	0.677550	-1.384106	-0.370815	-1.063057	0.348820	-0.464998
Max Sharpe Ration Portfolio	-0.016656	0.045148	0.047242	1.046382	-0.368923	-0.017723	-0.356802	0.049672	-0.047506
Min Variance Portfolio	-0.032884	0.023447	0.020157	0.859690	-1.402503	-0.032750	-1.138755	0.028759	-0.042232
Hierarchical Risk Parity ortfolio	-0.000499	0.006381	0.003468	0.543522	-0.078266	-0.000515	-0.091456	0.005632	-0.005150
Kellys Criteria Portfolio	-0.000042	0.004966	0.004971	1.001052	-0.008385	-0.000054	-0.010735	0.005070	-0.003069

Figure 4.45: Portfolio Measures for Performance Analysis (Experiment Data)

Annual returns for all portfolios are either minus or zero. It indicates portfolios are at loss. The equal weight portfolio is the worst performing portfolio as the annual return (-0.370815), mean (-0.4014), and drawdown (-0.46490) is the lowest as shown in Figure 4.45. For The EWP, Annualized volatility (0.3488), downside deviation (0.2900) is the highest, that indicates the risk is very high. All measures indicate that equal distribution of assets for all cryptocurrencies is not a good strategy for volatile assets. The Annual return of Kelly's criteria and Hierarchical Risk parity portfolios is almost zero, which implies those methods distributed risk better than other methods for experimental data. The annual volatility of HRP and Kelly's portfolio is 0.0056 and 0.0050, that is lowest. It supports the fact about risk distribution for both methods.

	Mean	Downside SD	Upside SD	Volatility Skewness	Sortino	Annualized Return	Annualized Sharpe Ratio	Annualized Volatility	Draw Down	USD	ethereum	binance	bitcoin	tether
Equal Weight Portfolio	0.647192	0.369198	0.302349	0.818934	1.752965	0.696871	1.441650	0.483384	-0.421561	0.200000	0.200000	0.200000	0.200000	0.200000
Max Sharpe Ration Portfolio	0.551887	0.261897	0.242186	0.924738	2.107267	0.633018	1.813279	0.349101	-0.298971	0.633496	0.140884	0.218529	0.000787	0.006304
Min Variance Portfolio	0.015545	0.016790	0.012499	0.744411	0.925853	0.015492	0.834550	0.018563	-0.016163	0.634311	0.009955	0.002225	0.002256	0.351253
Hierarchical Risk Parity ortfolio	-0.000096	0.013941	0.012475	0.894857	-0.006860	-0.000209	-0.013867	0.015077	-0.011735	0.536200	0.000120	0.000100	0.000200	0.463370
Kellys Criteria Portfolio	-0.000100	0.006959	0.006241	0.896730	-0.014360	-0.000128	-0.017072	0.007494	-0.005993	0.370000	0.000000	0.000000	0.000000	0.130000

Figure 4.46: Portfolio Measures for Performance Analysis (Historical Data)

Figure 4.46 illustrates, the annual return (0.6968) is positive, max sharp ratio (0.633018), and min variance (0.015492) for the equal weight portfolio, whereas it is almost zero for HRP and Kelly's portfolio. If we consider the risk distribution of the assets, the equal-weight portfolio has the highest annual volatility (0.4833), Mean (0.6471), and downside standard deviation (0.36198). The numbers are high for the maximum Sharpe ratio portfolio also. The high downside deviation portfolio is not preferable even though good portfolio returns because it indicates the price volatility of the investment (The Balance, n.d.). The risk distribution of Kelly's criteria and HRP portfolio is better than other methods and in line with the finding of the portfolios analysis for the experiment dataset.

Chapter 5 Dissertation Summary, Discussions, and Conclusions

5.1 Summary

The centre idea of the dissertation is to construct the portfolios for crypto currencies. The efficient forecasting of the assets' price is prerequisite to the optimal portfolio construction. The LSTM machine learning model has been employed to forecast the price of the cryptocurrencies. The intraday dataset with three different frequencies (10-min, 30-min, 60-min) were provided as the input to the LSTM model for five cryptocurrencies. The multivariant LSMT model has been employed in which six features (High, Low, Open, Close, Simple average of the last 50, 100 days) used to predict the target close price of the crypto assets. On top of that, Grid Search Hyperparameter tuning was used to optimize the parameters (neurons, batch size and dropout) that improved the performance of the machine learning model. The portfolios were created with the experiment and historical data using the portfolio method. Those portfolios analyzed based on the measures define in the section 3.3.5.

5.2 Discussions

The RMSE score and loss function v/s epochs graph derived before the LSTM model parameter tuning and after as well for 15 datasets of five assets. The RMSE score shows the prediction error, while the loss function implies the model fitting for the given dataset. The lower the RMSE score and loss function values, the higher the performance of the LSTM model. The line graphs for real price and predicated price demonstrate the accuracy of trends and price predication of the LSTM model. The residual values show the deviation of predicted price from the real price. The accurate forecasted price used as input to the portfolio construction, that provide the value of weight for each asset in the portfolio. Using the calculated weigh and the daily return of the cryptocurrency various performance measures were calculated. It provides basis to perform analysis of the constructed portfolio.

The result discussed in section 4.3 concludes that The LSMT model performed highly accurate price prediction for the 10-min dataset as compared to other datasets. The RMSE score was the lowest, and the value of the loss function decreased with epochs after parameter tuning. The fluctuation of the close price was also predicated efficiently for predicated value as shown in Figure 4.21, Figure 4.24, Figure 4.27, Figure 4.30, and Figure 4.33. The 10-min dataset has more data points than the 30-min and 60-min datasets since data points were collected every 10 mins. The optimized hyperparameters and more data points in 10-minute datasets benefited the LSTM model for high performance. The result of section 4.4.5 for the portfolio performance analysis indicates that risk diversification was better for the Hierarchical Risk Parity portfolio followed by Kelly's portfolio and Mean-variance portfolio, whereas the Equal weight portfolio performed the worst in terms of risk adjustment.

The performance of the multivariant LSTM model had better accuracy when predicting the close price for the 10-min Intraday dataset of five crypto assets as the RMSE error was lower for the forecasted result. Ramkumar, G. built three different machine learning models CCN, LSTM, and ARIMA for the price prediction of nine different cryptocurrencies (Binance coin, Bitcoin, Bitcoin cash, Chainlink, EOS, ETH, Litecoin, MCO, XRP). Table 5.1 illustrate the comparison of the RMSE error score of the bitcoin, Ethereum, and Binance coin between the ARIMA, CNN, and LSTM models and the proposed LSTM model.

Table 5.1: Comparison of Proposed LSTM Model

Crypto asset	ARIMA	CNN	LSTM	Proposed LSTM Model
Bitcoin	26.923	47.12	12.764	0.0043
Ethereum	0.96	0.857	0.029	0.0053
Binance coin	0.416	0.175	0.028	0.0055

The comparison of the result indicates RMSE score of the proposed multivariant LSTM is the lowest as compared to the ARIMA, CNN, and LSTM for three cryptocurrencies. In 2022, Hansun, Wicaksana, and Khaliq predicated the crypto price using the multivariant LSTM model. The RMSE values of the multivariant LSTM model are ***2518.021766 for bitcoin, 150.091734 for Ethereum***, ***0.003415 for Tether***, *and* ***27.624510 for Binance.*** *T*he RMSE scores of the proposed multivariant LSTM are ***bitcoin*** for ***0.0043***, ***0.0053 for Ethereum***, ***0.0036 for Tether***, *and* ***0.0055 for Binance***. The error score for the proposed LSMT model is lower than their multivariant model. It indicates the accuracy of the proposed model is superior. Due to hyperparameter adjustment and the technical indicators (Simple Moving Average 50 days and 200 days) of cryptocurrencies supplied as input features to the model, the performance of the multivariant LSTM model increased in comparison to the above research studies.

Tomasz Kaczmarek and Katarzyna Perez's study demonstrated that the hierarchical risk parity (HRP) and mean-variance portfolio construction methods adjust risk more compared to the equal-weight portfolio method for US stocks. The result of the portfolio performance analysis (section 4.4.5) shows that the volatility and downside deviation of the HRP (0.006 & 0.005) and mean-variance (0.028 & 0.023) portfolio is lower compared to the equal-weight portfolio (0.34 & 0.29) for the experiment dataset. The result of the portfolio analysis supports the study of Tomasz Kaczmarek and Katarzyna Perez. The comparative analysis of both results is not possible as the assets used for both experiments are different, but the conclusion is the same.

The finding of the study shed light on the potential application of the LSTM model in different business scenarios such as financial or time series data from the healthcare sectors. The proposed framework of the portfolio construction with forecasted crypto price can contribute to developing a robust application to advise investors. It can construct/optimize portfolio based on their risk profile and advise to take more risk-free decision.

5.3 Conclusion and Further Work

The machine learning algorithm can be used effectively to build a portfolio of cryptocurrencies. The price prediction of the asset is crucial while building an efficient portfolio as it helps to derive the future return. In case of the cryptocurrency, it is necessary to do efficient price forecasting due to more volatility and risk of the assets. Hence, The LSTM machine learning model can be used to solve the problem.

As part of the project, the multivariant LSTM model was employed with intraday data of three different frequencies (10-min, 30-min, and 60-min) as input for price prediction of five crypto assets. The hyperparameter tuning was also used to increase the efficiency of the LSTM model. The lowest RMSE error score was obtained for the 10-min intraday data of the bitcoin (0.0043), Ethereum (0.0053), Binance (0.0055), Tether (0.0036), and USD coin (0.0059), after approximate total 217 successful iterations of LSTM model during the experiment. For the same dataset, the lower residual values were recorded that indicate the LSTM performance was accurate. The results demonstrate that machine learning algorithms can accurately forecast the price of the cryptocurrency. Among all four-portfolio construction methods, the Hierarchical risk parity method efficiently adjusted risk, while the equal weight method performed the worst irrespective of the positive or negative price trend in terms of risk adjustment.

Several limitations were faced during the research project. The Jupyter notebook tool was not working effectively due to the limited computation power of the personal laptop. The Google Colab has been used as an alternative tool for the experiment to generate the results. Bloomberg's intraday dataset does not provide the trade volume. It can be the potential feature of the model for price prediction (Shynkevich et al., 2017). The portfolio construction experiment was carried out with only five cryptocurrencies due to the time limitation of the research.

The conducted research work can serve as the basis for several works. Bloomberg provides crypto news and crypto social media monitoring to analyze the crypto market (Bloomberg Crypto Data, Research & Tools, n.d.). This data can be used to do sentiment analysis with natural language processing and determine the impact of the news and social media on the future trends of cryptocurrencies. A trading strategy is one of the fields of portfolio construction (Refer to Figure 2.1). An effective trading strategy can be developed based on the forecasted crypto price to derive the buy and sell signal. This will help investors to maximize their returns.

6 References

[1] D, O.-J. and E, H.-M. (2019). Forecasting bitcoin pricing with hybrid models: A review of the literature. ijaers.com, [online] 6(9). Available at: https://ijaers.com/detail/forecasting-bitcoin-pricing-with-hybrid-models-a-review-of-the-literature/ [Accessed 20 Dec. 2022] (D and E, 2019)

[2] Siami-Namini, S., Tavakoli, N. and Siami Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). doi:10.1109/icmla.2018.00227. [Accessed 20 Dec. 2022] (Siami-Namini, Tavakoli and Siami Namin, 2018)

[3] Nakamoto, S. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [online] Available at: https://www.ussc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging_Tech_Bitcoin_Crypto.pdf [Accessed 20 Dec. 2022] (Nakamoto, 2008)

[4] Treible Maier, H., 2018. The impact of the blockchain on the supply chain: a theory-based research framework and a call for action. Supply chain management: an international journal. [Accessed 20 Dec. 2022]

[5] Malik1#, S. and Rana2, A. (n.d.). *A Brief Survey of Cryptocurrency Systems*. [online] *IITM Journal of Management*. Available at: http://www.iitmjanakpuri.com/iitmjournal/data/2020_Vol11_No1_it15.pdf [Accessed 20 Dec. 2022] (Malik1# and Rana2, n.d.)

[6] Burniske, C. and White, A., 2017. Bitcoin: Ringing the bell for a new asset class. Ark Invest (January 2017) https://research.ark-invest.com/hubfs/1_Download_Files_ARK-Invest/White_Papers/Bitcoin-Ringing-The-Bell-For-A-New-Asset-Class.pdf [Accessed 1 Jan. 2023] (Burniske, C. and White, A., 2017)

[7] Sovbetov, Y. (2018). Factors Influencing Cryptocurrency Prices: Evidence from Bitcoin, Ethereum, Dash, Litcoin, and Monero. Journal of Economics and Financial Analysis, [online] 2(2), pp.1–27. Available at: https://econpapers.repec.org/article/trp01jefa/jefa0016.htm [Accessed 20 Dec. 2022] (Sovbetov, 2018).

[8] Baur, D.G., Hong, K. and Lee, A.D. (2018). Bitcoin: Medium of exchange or speculative assets? *Journal of International Financial Markets, Institutions and Money*, 54, pp.177–189. doi:10.1016/j.intfin.2017.12.004. [Accessed 20 Dec. 2022] (Baur, Hong and Lee, 2018)

[9] DeMiguel, V., Garlappi, L. and Uppal, R. (2007). Optimal Versus Naive Diversification: How Inefficient is the 1/NPortfolio Strategy? *Review of Financial Studies*, [online] 22(5), pp.1915–1953. doi:10.1093/rfs/hhm075. [Accessed 20 Dec. 2022] (DeMiguel, Garlappi and Uppal, 2007)

[10] Rejeb, A., Rejeb, K. and G. Keogh, J. (2021). Cryptocurrencies in Modern Finance: A Literature Review. ETIKONOMI, 20(1), pp.93–118. doi:10.15408/etk.v20i1.16911. [Accessed 20 Dec. 2022] (Rejeb, Rejeb and G. Keogh, 2021)

[11] Snow, D., 2020. Machine learning in asset management—Part 1: Portfolio construction—Trading strategies. *The Journal of Financial Data Science*, *2*(1), pp.10-23. [Accessed 21 Dec. 2022]. (Snow, 2020)

[12] Snow, D. (2020). Machine Learning in Asset Management—Part 2: Portfolio Construction—Weight Optimization. The Journal of Financial Data Science, 2(2), pp.17–24. doi:10.3905/jfds.2020.1.029. [Accessed 21 Dec. 2022]. (Snow, 2020)

[13] Yun, H., Lee, M., Kang, Y.S. and Seok, J. (2020). Portfolio management via two-stage deep learning with a joint cost. Expert Systems with Applications, 143, p.113041. doi:10.1016/j.eswa.2019.113041. [Accessed 21 Dec. 2022]. (Yun et al., 2020)

[14] Kaczmarek, T. and Perez, K. (2021). Building portfolios based on machine learning predictions. *Economic Research-Ekonomska Istraživanja*, pp.1–19. doi:10.1080/1331677x.2021.1875865.
[Accessed 21 Dec. 2022]. (Kaczmarek and Perez, 2021)

[15] Kedia, V., Khalid, Z., Goswami, S., Sharma, N. and Suryawanshi, K. (2018). *Portfolio Generation for Indian Stock Markets Using Unsupervised Machine Learning*. [online] IEEE Xplore. doi:10.1109/ICCUBEA.2018.8697771. [Accessed 21 Dec. 2022]. (Kedia et al., 2018)

[16] Ta, V.-D., Liu, C.-M. and Tadesse, D.A. (2020). Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading. *Applied Sciences*, 10(2), p.437. doi:10.3390/app10020437. [Accessed 21 Dec. 2022]. (Ta, Liu and Tadesse, 2020)

[17] Ramkumar, G. (2021). Cryptocurrency Portfolio Construction Using Machine Learning Models. *Contemporary Trends and Challenges in Finance*, pp.103–122. doi:10.1007/978-3-030-73667-5_7. [Accessed 21 Dec. 2022]. (Ramkumar, 2021)

[18] Hansun, S., Wicaksana, A. and Khaliq, A.Q.M. (2022). Multivariate cryptocurrency prediction: comparative analysis of three recurrent neural networks approaches. *Journal of Big Data*, 9(1). doi:10.1186/s40537-022-00601-7. [Accessed 21 Dec. 2022]. (Hansun, Wicaksana and Khaliq, 2022)

[19] DeMiguel, V., Garlappi, L. and Uppal, R. (2007). Optimal Versus Naive Diversification: How Inefficient is the 1/NPortfolio Strategy? Review of Financial Studies, [online] 22(5), pp.1915–1953. doi:10.1093/rfs/hhm075. [Accessed 21 Dec. 2022]. (DeMiguel, Garlappi and Uppal, 2007)

[20] Peterson, Z. (2018). The Kelly criterion in portfolio optimization: a decoupled problem. *Journal of Investment Strategies*, 7(2), pp.53–76. doi:10.21314/jois.2017.097. [Accessed 21 Dec. 2022]. (Peterson, 2018)

[21] Burggraf, T. (2019). *Risk-Based Portfolio Optimization in the Cryptocurrency World*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3454764. [Accessed 21 Dec. 2022]. (Burggraf, 2019)

[22] Khedr, A.M., Arif, I., P V, P.R., El-Bannany, M., Alhashmi, S.M. and Sreedharan, M. (2021). Cryptocurrency price prediction using traditional statistical and machine-learning techniques: A survey. *Intelligent Systems in Accounting, Finance and Management*, 28(1), pp.3–34. doi:10.1002/isaf.1488. [Accessed 1 Jan. 2023] (Khedr et al., 2021)

[23] SearchEnterpriseAI. (n.d.). *What is deep learning and how does it work?* [online] Available at: https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network#:~:text=Deep%20learning%20is%20a%20type [Accessed 1 Jan. 2023] (SearchEnterpriseAI, n.d.)

[24] Stanford.edu. (2021). *CS 230 - Recurrent Neural Networks Cheatsheet*. [online] Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture . [Accessed 1 Jan. 2023] (Stanford.edu, 2021)

[25] Olah, C. (2015). *Understanding LSTM Networks -- colah's blog*. [online] Github.io. Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Accessed 1 Jan. 2023] (Olah, 2015)

[26] Hochreiter, E., Rovelli, R. and Winckler, G. (1997). Centrální banky a využití ražebného? studie tří transformujících se ekonomik. *Politická ekonomie*, [online] 45(2), pp.193–206. doi:10.18267/j.polek.274. [Accessed 1 Jan. 2023]

[27] Stanford.edu. (2019). *The Sharpe Ratio*. [online] Available at: https://web.stanford.edu/~wfsharpe/art/sr/sr.htm. [Accessed 1 Jan. 2023] (Stanford.edu, 2019)

[28] Srivastava, P. and Mazhar, S.S., 2018. Comparative analysis of sharpe and sortino ratio with reference to top ten banking and finance sector mutual funds. International journal of management studies, 93(10). [Accessed 1 Jan. 2023]

[29] Fernando, J. (2022). *Sharpe Ratio*. [online] Investopedia. Available at: https://www.investopedia.com/terms/s/sharperatio.asp [Accessed 1 Jan. 2023]

[30] Investopedia. (2019). *Understanding the Sortino Ratio*. [online] Available at: https://www.investopedia.com/terms/s/sortinoratio.asp. [Accessed 1 Jan. 2023] (Investopedia, 2019)

[31] Platanakis, E. and Urquhart, A. (2019). Portfolio management with cryptocurrencies: The role of estimation risk. *Economics Letters*, [online] 177, pp.76–80. doi: 10.1016/j.econlet.2019.01.019. [Accessed 1 Jan. 2023]

[31] Sung, O. (2021). The Kelly Criterion Applied to Long-Term Value Investing. *Junto*. [online] 17 Aug. Available at: https://junto.investments/kelly-criterion/ [Accessed 21 Dec. 2022]. (Sung, 2021)

[32] Lopez de Prado, M. (2016). *Building Diversified Portfolios that Outperform Out-of-Sample*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2708678 [Accessed 1 Jan. 2023] (Lopez de Prado, 2016)

[33] Investopedia. (2019). Annualized Total Return. [online] Available at: https://www.investopedia.com/terms/a/annualized-total-return.asp. [Accessed 1 Jan. 2023] (Investopedia, 2019)

[34] Chen, J. (n.d.). *Mean Return*. [online] Investopedia. Available at: https://www.investopedia.com/terms/m/meanreturn.asp. [Accessed 1 Jan. 2023] (Chen, n.d.)

[36] Kenton, W. (n.d.). *What Is Downside Deviation?* [online] Investopedia. Available at: https://www.investopedia.com/terms/d/downside-deviation.asp. [Accessed 1 Jan. 2023] (Kenton, n.d.)

[37] Priyadarshini, I. and Cotton, C. (2021). A novel LSTM–CNN–grid search-based deep neural network for sentiment analysis. *The Journal of Supercomputing*. doi:10.1007/s11227-021-03838-w. [Accessed 1 Jan. 2023] (Priyadarshini and Cotton, 2021)

[38] cvxopt.org. (n.d.). *Home — CVXOPT*. [online] Available at: https://cvxopt.org/ [Accessed 6 Jan. 2023]. (cvxopt.org, n.d.) pyportfolioopt.readthedocs.io. (n.d.). *PyPortfolioOpt*. [online] Available at: https://pyportfolioopt.readthedocs.io/en/latest/. [Accessed 1 Jan. 2023] (pyportfolioopt.readthedocs.io, n.d.)

[39] Python.org. (2020). *The Python Standard Library — Python 3.8.1 documentation*. [online] Available at: https://docs.python.org/3/library/ [Accessed 1 Jan. 2023] (Python.org, 2020)

[40] Scikit-learn.org. (2019). *5.3. Preprocessing data — scikit-learn 0.21.3 documentation*. [online] Available at: https://scikit-learn.org/stable/modules/preprocessing.html [Accessed 1 Jan. 2023] (Scikit-learn.org, 2019)

[41] TensorFlow (2019). *TensorFlow*. [online] TensorFlow. Available at: https://www.tensorflow.org/ [Accessed 1 Jan. 2023] (TensorFlow, 2019)

[42] Aroussi, R. (n.d.). yfinance: Yahoo! Finance market data downloader. [online] PyPI. Available at: https://pypi.org/project/yfinance/ [Accessed 1 Jan. 2023] (Aroussi, n.d.)

[43] seaborn (2012). seaborn: statistical data visualization — seaborn 0.9.0 documentation. [online] Pydata.org. Available at: https://seaborn.pydata.org/ [Accessed 1 Jan. 2023] (seaborn, 2012)

[44] matplotlib.org. (n.d.). Matplotlib: Python plotting — Matplotlib 3.3.4 documentation. [online] Available at: https://matplotlib.org/stable/index.html [Accessed 1 Jan. 2023] (matplotlib.org, n.d.)

[45] pandas.pydata.org. (n.d.). pandas documentation — pandas 1.0.1 documentation. [online] Available at: https://pandas.pydata.org/docs/ [Accessed 1 Jan. 2023] (pandas.pydata.org, n.d.)

[46] NumPy (n.d.). Overview — NumPy v1.19 Manual. [online] numpy.org. Available at: https://numpy.org/doc/stable/ [Accessed 1 Jan. 2023] (NumPy, n.d.)

[47] pandas-datareader.readthedocs.io. (n.d.). *pandas-datareader — pandas-datareader 0.8.0+4.gec799a0 documentation*. [online] Available at: https://pandas-datareader.readthedocs.io/en/latest/ [Accessed 1 Jan. 2023] (pandas-datareader.readthedocs.io, n.d.)

[48] finance.yahoo.com. (n.d.). *Crypto Real Time Prices & Latest News - Yahoo Finance*. [online] Available at: https://finance.yahoo.com/crypto/ [Accessed 3 Jan. 2023] (finance.yahoo.com, n.d.)

[49] Bloomberg Crypto Data, Research & Tools. (n.d.). *Bloomberg Professional Services*. [online] Available at: https://www.bloomberg.com/professional/solution/bloomberg-crypto-data-research-tools/ [Accessed 3 Jan. 2023]. (Bloomberg Crypto Data, Research & Tools, n.d.)

[50] Deer, M. (2022). *Golden cross vs. death cross explained*. [online] Cointelegraph. Available at: https://cointelegraph.com/explained/golden-cross-vs-death-cross-explained [Accessed 3 Jan. 2023]. (Deer, 2022)

[51] Anon, (2017). Investment Portfolio Optimisation With Python - Python For Finance. [online] Available at: https://www.pythonforfinance.net/2017/01/21/investment-portfolio-optimisation-with-python/ [Accessed 30 Dec. 2022] (Anon, 2017)

[52] pyportfolioopt.readthedocs.io. (n.d.). *Other Optimizers — PyPortfolioOpt 1.5.2 documentation*. [online] Available at: https://pyportfolioopt.readthedocs.io/en/latest/OtherOptimizers.html#id4 [Accessed 30 Dec. 2022]. (pyportfolioopt.readthedocs.io, n.d.)

[53] KellyPortfolio. (n.d.). KellyPortfolio. [online] Available at: https://github.com/thk3421-models/KellyPortfolio [Accessed 30 Dec. 2022]. (Kirschenmann, 2022)

[54] Brownlee, J. (2017). How to Diagnose Overfitting and Underfitting of LSTM Models. [online] MachineLearningMastery.com. Available at: https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/#:~:text=LSTM%20models%20are%20trained%20by [Accessed 31 Dec. 2022]. [Accessed 1 Jan. 2023] (Brownlee, 2017)

[55] Brownlee, J. (2018). *Difference Between a Batch and an Epoch in a Neural Network*. [online] Machine Learning Mastery. Available at:

https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ [Accessed 1 Jan. 2023] (Brownlee, 2018)

[56] The Balance. (n.d.). *Downside Deviation*. [online] Available at: https://www.thebalancemoney.com/what-is-downside-deviation-4590379 [Accessed 1 Jan. 2023].(The Balance, n.d.)

[57] Team, K. (n.d.). *Keras documentation: Callbacks API*. [online] keras.io. Available at: https://keras.io/api/callbacks/. [Accessed 1 Jan. 2023]. (Team, n.d.)

---

[58] Hansun, S., Wicaksana, A. and Khaliq, A.Q.M. (2022). Multivariate cryptocurrency prediction: comparative analysis of three recurrent neural networks approaches. Journal of Big Data, 9(1). doi:10.1186/s40537-022-00601-7. [Accessed 1 Jan. 2023] (Hansun, Wicaksana and Khaliq, 2022)

[59] Bloomberg Crypto Data, Research & Tools. (n.d.). Bloomberg Professional Services. [online] Available at: https://www.bloomberg.com/professional/solution/bloomberg-crypto-data-research-tools/. [Accessed 1 Jan. 2023]. (Bloomberg Crypto Data, Research & Tools, n.d.)

[60] Shynkevich, Y., McGinnity, T.M., Coleman, S.A., Belatreche, A. and Li, Y. (2017). Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, 264, pp.71–88. doi:10.1016/j.neucom.2016.11.095. [Accessed 1 Jan. 2023] (Shynkevich et al., 2017)

Please ignore the in-text citation. It is for my convenience. I will remove it in final version.