Determining Cryptocurrency Trends using LSTM

Shreya Chilumukuru

Department of Applied Data Science, San Jose State University

Data 270: Data Analytics Processes

Dr. Linsey Pang

May 13, 2024

Abstract

Cryptocurrencies are digital or virtual currencies typically exchanged on decentralised computer networks known as blockchains, ensuring transparency and immutability. They use cryptography (which is a study of encrypting and decrypting the information to ensure only the sender and intended recipient can understand it) for secure financial transactions. The uprise of cryptocurrencies in the last few years and the introduction of blockchain technology have created several possibilities for safe and transparent digital transactions. Reflecting on these changing dynamics of the crypto world, with this project, we aim to build a machine learning model that captures the highs of the cryptocurrency and identifies the right peak suitable to invest in and the type of crypto to invest in.

We look forward to building and comparing the performance of linear regression, random Forest, Gradient Boost, and LSTM. We will present comprehensive details of the algorithms' performance and effectiveness in predicting the highs of cryptocurrency and the appropriate cryptos to invest in.

Furthermore, we aim to generate engaging visualisations that help users immediately capture conclusions and critical takeaways. After completing this project, we aspire to build a model that can be used for crypto asset selection, real-time market updates, risk management, market timing, and investment strategy. The evaluation metrics used to determine the best model are MSE and R-squared.

1. Introduction

1.1. Project Background and Execute Summary

A cryptocurrency is a digital or virtual currency that uses cryptography for security purposes when conducting financial transactions. Unlike conventional government-issued currencies, cryptocurrencies operate on blockchains, which are decentralised networks guaranteeing openness, non-changeability, and reliance throughout an entire system. It is a public ledger without links to any central authority or national government. The rise of cryptocurrencies was first experienced in 2009 with the introduction of Bitcoin by an anonymous person/group called Satoshi Nakamoto.

Cryptocurrencies use cryptographic techniques to secure transactions and control the production of additional coins. A collection of nodes or computers checks transactions on one blockchain instead of banks and financial organisations acting as intermediaries. This decentralisation means that individuals can do peer-to-peer transactions directly using their funds, thus eliminating the conventional barriers in a financial transaction. A critical characteristic of cryptocurrencies is that they are not under any central authority or government. This feature and cryptography ensure security and privacy when making transactions.

People care about predicting how much cryptocurrencies will cost in the future. This is because digital money is quite essential these days. Bitcoin, Ethereum, and other cryptocurrencies are popular now. They are used as alternative digital money and for trading. The crypto market keeps changing, so investors want to know the trends and cost changes. They can use this info to make intelligent choices. If we can guess the prices well, it can help with investing, risk, and creating a varied portfolio. Many people want to trade with cryptocurrencies to make a lot of money. But it's tough to guess the right moves because the market constantly

shifts. We need better tools and ways to help us with this. This plan provides a simple but good way to tackle this issue. It looks at old data to guess what a specific cryptocurrency will be worth. Linear regression is a way to look at a connection between two things. Here, we'll check the link between a cryptocurrency's cost and a few things that affect it. We can guess what a cryptocurrency will be worth using linear regression on old data.

Also, we will check the guesses using methods like Gradient Boost, Random Forest, and Long Short-Term Memory (LSTM). These methods are robust and use different ways to work. We use Gradient Boost to guess well. This method makes lots of guesses to be very fitting. On the other hand, Random Forest uses many decisions to make good guesses. LSTMs are a particular type of recurrent neural network that is very good at recognising patterns and relationships in sequential data. Cryptocurrency prices show patterns over time, whereas past prices can often predict future prices. Long Short-Term Memory (LSTM) networks are well-suited for modelling these sequential relationships. Each of these ways brings a unique set of ideas that help us to make better guesses.

The project aims to predict the high costs of specific cryptocurrencies as accurately as possible, offering market insights to users and acting as a valuable guide on their investment path. It uses machine learning techniques to give high-quality forecasts, including linear regression and sentiment analysis.

1.1.1 Contributions of this project:

Our system produces superior forecasts using sophisticated machine learning algorithms, thus enhancing accuracy compared to traditional forecasting methods and showcasing improved accuracy in predicting cryptocurrency.

The project predicts high prices for selected cryptocurrencies, providing helpful market insights to investors who can use them to make informed decisions about buying or selling these digital assets and provisioning market insights for cryptocurrency investors.

The study helps cryptocurrency investors by providing accurate price predictions and essential market information for better investment decision-making. We aim to demonstrate how machine learning could predict cryptocurrency prices and broaden the concepts of these techniques. This can further elevate the accuracy of forecasts, assisting investors in making more informed choices.

The project promises to impact cryptocurrency increasingly, investing by furnishing precise price predictions and market insights.

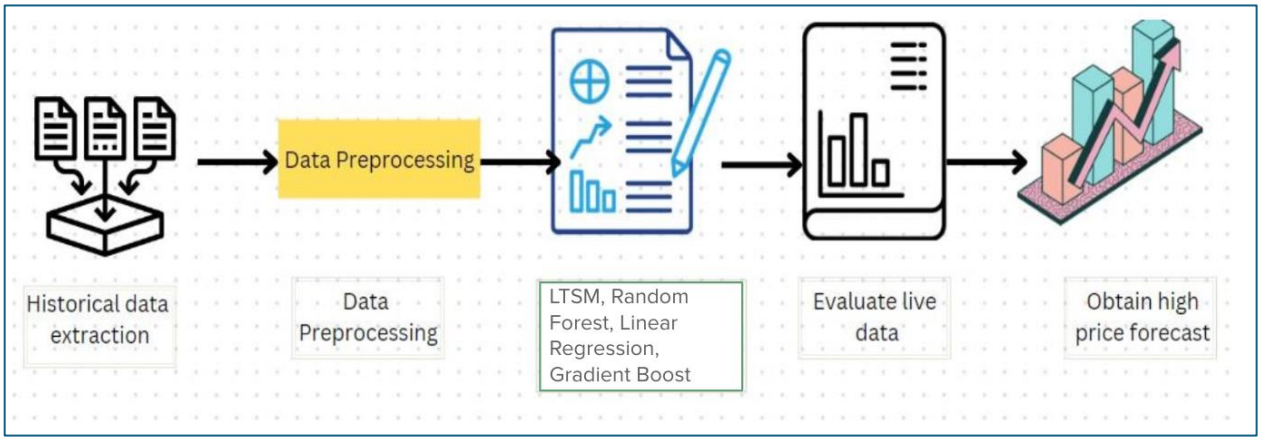
1.1.2 Applications of this project:

With this project comprising of different phases as shown in Figure 1, novice users and businesses can integrate the provided price predictions into their financial strategies. The project serves as an educational tool for cryptocurrency enthusiasts, providing insights into the demographics of cryptocurrency markets and the various factors influencing price movements.

High-price predictions can signal traders to buy or sell specific cryptocurrencies. Traders may execute trades based on forecasted high prices to capitalise on potential profits or to exit positions before anticipated downturns.

Figure 1

Project Architecture



Note. Different Phases of the Project Architecture

1.2. Project Requirements

The project aims to predict future high prices for selected cryptocurrencies. This requires historical price data on these cryptocurrencies. The sample data used in the project include past price information for cryptocurrencies like Bitcoin, Ethereum, Dogecoin, Ripple and Aave. We can get this data from cryptocurrency exchanges such as Binance, Coinbase, or Kraken. For this project, the data collected for each coin is as follows: Bitcoin has around 11 years of data, Aave has four years of data, Dogecoin has 11 years of data, Ethereum has nine years, and Ripple has 11 years of data.

A System Requirement Specification is an organised collection of details that outline the system's necessities. It explains all information and functional and behavioural needs for the software being made or developed.

1.2.1 The Hardware Requirements

The hardware requirements of the project consist of a Processor- 1.9 gigahertz (GHz) x86- or x64-bit dual-core processor with an SSE2 instruction set. RAM- 4 GB or more and Hard disk – 8 GB or more

1.2.2 The Software Requirements

The software requirements of the project comprise of Operating System- Windows 7 or above, Programming Language- Python 3, Libraries- Scikit learn, SqlAlchemy, Request, Crypto CMD, IDE- Visual Studio Code and APIs- Coin gecko, Coin Market Cap

1.2.3 The Functional Requirements

The things the system must do are functional requirements. Some of these are gathering data, preparing it for use, making models using machine learning, checking how accurate the models are, and showing the predictions nicely. First, the system needs to gather past data on certain digital currencies from different places, like exchanges, market data providers, and website scraping. After that, the data needs to be cleaned up and ready for analysis by removing copies and errors and setting it in a standard way. Then, the system must make models using machine learning methods to guess the high prices of certain digital currencies. Once that's done, it must check how well the models use MSE and R-squared. Finally, it needs to show the guesses nicely and clearly. Non-functional needs are about how well the system works. Some needs for this project include How fast the system can make sound predictions, how easy it is for people to use, how safe the system is for people's information, how often the system works well, how much the system can handle without issues and the users say what they need, and the system should do that.

1.3 Project Deliverables

We have divided our project deliverables into phases, ensuring every step is noticed and noticed. The phases are described below.

1.3.1 Project Scope Document:

Project scope, including the specific cryptocurrencies to be analyzed and predicted.

1.3.2 Data Collection and Preprocessing Report

Document the data sources used for historical cryptocurrency prices and relevant market data. Describe the techniques applied, such as handling missing values, outliers, and data normalization for data preprocessing.

1.3.3 Exploratory Data Analysis (EDA) Report

Visualizations and statistical analysis of the collected data and Identification of trends, Analysis of feature importance and selection of the most relevant features for price prediction patterns, and correlations in the cryptocurrency prices and market indicators

1.3.4 Feature Engineering and Selection Report

Documentation of the features engineered from the collected data, Analysis of feature importance and selection of the most relevant features for price prediction.

1.3.5 Machine Learning Model Development Report

Description of the machine learning algorithms and architectures explored for price prediction. Hyperparameter tuning and model optimisation techniques were applied, and evaluation metrics and performance comparisons of different models were used.

1.3.6 Prototype Application

A working prototype of the cryptocurrency price prediction system. Integration of the trained machine learning model into the prototype

1.3.7 Model Evaluation and Testing Report

A comprehensive evaluation of all the model's performance on unseen data. Analyse the models' accuracy, precision, recall, and other relevant metrics. documentation of the testing methodology and results

1.3.8 Final Machine Learning Model Report

Fully trained and optimized machine learning model for cryptocurrency price prediction. Serialized model files and necessary dependencies for deployment

1.3.9 Production Ready Application

A deployable application that integrates the final machine learning model. Documentation for setting up, running, and maintaining the application.

1.4.0 Project Documentation

Detailed documentation of the entire project lifecycle. Code repository with well-commented source code.

1.4. Technology and Solution Survey

Previous research has presented numerous solutions for addressing each of the steps outlined in our problem statement. Suhwan Ji, Jongmin Kim and Hyeonseung Im have conducted a comprehensive study and solution by comparing different state-of-the-art deep learning techniques, including Long Short-Term Memory (LSTM) models, Convolutional Neural Networks (CNN), Deep Neural Networks (DNN), Deep Residual Networks (ResNet, their combinations and some ensemble models for Bitcoin Price Prediction. Their study evaluated these models in terms of both classification (determining the price directions (up or down)) and regression (determining the actual price of the Bitcoin). The results of their study indicated that LSTM-based models slightly outperformed the other models for regression problems. For

regression problems, LSTM-based models slightly outperformed other models, highlighting their capability to capture the temporal dynamics of Bitcoin prices.

On the other hand, for classification problems, DNN-based models performed the best, indicating their effectiveness in capturing relevant features for directional movement. An important observation is that the classification models were more accurate than regression models for algorithmic trading. This research also explored the impact of data split methods, sequence size and normalisation techniques on model performance, giving valuable insights into preprocessing steps that impact the outcomes.

Research work performed by Patel Jay, Vasu Kalariya, Puspendra Parmar, Sudeep Tanwar, Neeraj Kumar, Mamoun Alazab suggested the use of a stochastic module that acquires the market's reaction to new information, which is incorporated into the prediction model. Their study integrated stochastic behaviour into Neural networks, and mathematics is formulated for a layer-wise stochastic walk, which is usually applied in financial markets to model stock prices. It combines layer-wise randomness in neural network feature activations to simulate market volatility. Furthermore, they have also introduced the algorithm for stochastic forward propagation in neural networks. The data is taken from coinmarketcap and stochastic Multi-Layer Perceptron (MLP), and Long Short-Term Memory (LSTM) models are used to predict the cryptocurrency prices. Their comprehensive study achieved an average relative improvement of 4.6% on MAPE using stochasticity.

Another comprehensive study conducted by E. Pintelas, I.E. Livieris, S. Stavroyiannis, T. Kotsilieris and P. Pintelas investigated the capability of Deep learning (DL) techniques in predicting 3 main cryptocurrency prices such as Bitcoin (BTC), Ethereum (ETH), and Ripple (XRP). This research work evaluated various DL models, including Long Short-Term Memory

(LSTM), Bidirectional Long Short-Term Memory (BiLSTM), CNN-LSTM and CNN-BiLSTM first to extract features using CNN and then processed temporal dependencies using LSTM or BiLSTM against traditional machine learning models including Support Vector Regressor (SVR), 3- Nearest Neighbors (3NN), and Decision Tree Regressor (DTR) for cryptocurrency price prediction. This study reported that using advanced Deep Learning models, specifically CNN-LSTM and CNN-BiLSTM, slightly overperformed the traditional Machine Learning Models in cryptocurrency price prediction. This performance improvement was, however, marginal. CNN-LSTM and CNN-BiLSTM models have shown the best performance regarding RMSE, but CNN-BiLSTM performed the best regarding F1 scores and Accuracy over different forecasting horizons (4, 9, 12, and 16 hours). This study suggested that the price prediction of cryptocurrency might need to be more coherent and complex even for advanced Deep Learning Techniques to solve it effectively. This indicates a need for more sophisticated prediction models, advanced ensemble techniques, feature engineering techniques and validation metrics. This also showcased the importance of preferring the correct performance and validation metrics for evaluating price-prediction models.

The research work by Takuya Shintate and Lukáš Pichl indicated a novel framework for trend analysis and prediction classification in the non-stationary and highly volatile environment of cryptocurrency trading, majorly focusing on Bitcoin prices. The Random Sampling Method (RSM framework) is based on deep learning and deals with the challenges encountered because of the non-stationarity of financial time series data. This RSM is used here to compare the performance of the deep learning approach to base models for predicting Okcoin market Bitcoin prices. The primary intent of using RSM is to mitigate the class imbalance problem noticed in cryptocurrency markets by using this novel sampling technique and to exceed the performance of

a uniform random process in trend classification, thereby demonstrating the possibility of extracting deterministic patterns for market prediction.

Regarding F-measure, the RSM showcased the best classification performance compared with the other baseline methods, including Multilayer Perceptron and Long Short-Term Memory (MLP and LSTM). This indicated RSM's ability to analyse and predict trends in the highly chaotic and unpredictable Bitcoin market.

Erdinc Akyildirim, Ahmet Goncu, and Ahmet Sensoy have conducted a research study and investigated the predictability of 12 highly liquid cryptocurrencies at minute and daily-level frequencies using Machine Learning classification models. This research focuses on whether trends in cryptocurrency markets can be used to predict the price to some extent.
This research employed various machine learning classification techniques, including Support Vector Machines (SVM), Logistic Regression, and Artificial Neural Networks (ANN), specifically Multilayer Perceptrons and Random Forests. These models are used here to predict cryptocurrency price movements (up or down) based on historical price information and other technical indicators.

The performance of these algorithms is evaluated based on the accuracy, comparing the predicted price movement with actual movements. SVMs showed the best and most consistent results across different cryptocurrencies and timelines, and ANNs also showcased the effectiveness of predicting price movements with varying degrees of accuracy. This study also reports that the classification accuracy of the models consistently exceeds 50% across all cryptocurrency price movements. Machine learning models reached about 55-65% prediction accuracy on average, indicating that deterministic patterns for trend classification and market prediction are, to some extent, extractable from historical data.

1.5. Literature Survey

This literature survey culminates various papers on determining cryptocurrency trends using machine learning models. Today's cashless and digital transactions have paved the way for the rise in cryptocurrencies, which work as a medium of exchange. (Pintelas E, Livieris ., 2020).The cryptocurrency has gained momentum since early 2017 and skyrocketed in Jan 2018. (A.Elbaharaxy., 2017)

Considering the growth of cryptocurrencies, machine learning algorithms are the best fit for forecasting the process of cryptocurrencies as the models are trained from data using mathematical functions, decreasing the programming logic to execute a specific task. (Bonneau, J., 2015)

This paper, published by Laura And team, has applied machine learning algorithms to predict daily cryptocurrency prices from Nov 2015 to April 2018. The dataset extracted from the Coin Market Cap website was used from November 11, 2015, to April 24, 2018. The team has used XGboost and the LTSM algorithm. They have implemented the three machine learning models to test and compare three supervised models where the first two methods rely on XGBoost, and the third method is based on extended short-term memory networks. The team focused on predicting the return on investment for all currencies with the help of method 1 and built a different model for each currency that used information on the behaviour of the whole market to predict a single currency. Lastly, in method 3, they implemented a different model for each currency, where the prediction was based on previous currency prices. Per their analysis, methods 1 and 2 worked the best when the projections were based on short-term windows for 5-10 days. LTSM recurrent neural networks worked best when predictions were based on 50 days

of data. However, this project has considered a limited dataset from 2015 to 2018 and needs to represent the trend over a more extended period. (Alessandretti, L., 2018)

Mahir Iqbal and the team have focused on predicting the market price and stability of the cryptocurrency (Bitcoins) market using machine learning techniques such as ARIMA, FBProphet, and XG Boosting. The parameters used to evaluate the models are Root Mean Square Error(RMSE), Mean Absolute Error (MAE), and R^2. The team extracted the dataset from Kaggle, an open-source website. As per their analysis, the performance of the machine learning model of ARIMAX has proven to be the best, with an RSME of 322.4, and FBProp and XGBoost algorithms have gained RSME scores of 22.95 and 369, respectively. The team has admitted that the study can be further improved by Hypertuning the parameters of time-series algorithms to improve the RSME value. This paper has shown promising results for the machine learning model ARIMA, which can be helpful to novice users of the time series model. However, the paper has a limited dataset extracted from Kaggle and cannot be updated in real time. The predictions are based only on one cryptocurrency (Bitcoin) (Iqbal, M., 2021)

The paper published by Erdinc and the team is very informative, and they have predicted the most liquid twelve cryptocurrencies and analysed them at daily and minute levels frequencies using machine learning classification algorithms, including support vector machines, logistic regression, random forests and artificial neural networks with the past price information. As per their analysis, the machine learning classification algorithms reach about 55-65% predictive accuracy on average at the daily or minute level frequencies, whereas support vector machines continue to demonstrate the best and most consistent results in terms of predictive accuracy compared to logistic regression, artificial neural networks, and random forest classification algorithms. The paper published by this team has focused on the Question of whether

cryptocurrency prices are predictable and whether the Efficient Market Hypothesis (EMH) holds for cryptocurrencies. The efficient market hypothesis (EMH), alternatively known as the efficient market theory, is a hypothesis that states that share prices reflect all available information and consistent alpha generation is impossible. (Investopedia. (n.d.). Efficient Market Hypothesis). Alpha is used in finance as a performance measure, indicating when a strategy, trader, or portfolio manager has managed to beat the market return or other benchmark over some period. (Investopedia. (n.d.). Alpha) Per their findings and project analysis, the best performing and robust models for prediction are support vector machines with consistently above 50% fit and low variation across all products. The team acknowledges that with extra fine-tuning of the model selection step, the machine learning algorithms and space of features can quickly achieve over 70% predictive accuracy. This paper has once again proved that machine learning is promising for short-term forecasting trends in the cryptocurrency markets. (Awan, S.E., 2020)

Do-Hyung Kwon and the team have applied long short-term memory (LTSM) and Gradient Boost model models to classify the cryptocurrency price time series. They have collected the historical cryptocurrency price time series data from Bithumb. (n.d.). (which is no longer accessible). The project solely focuses on seven available cryptocurrencies. As per their analysis, the LTSM model outperforms the gradient boosting model. The LTSM model showed an improvement of about 7%. The f1-score values of the LSTM model are between 0.63 and 0.68, whereas those of the GB model are between 0.59 and 0.63.

The model proposed by Zeinab and Yung to present the exchange rate of cryptocurrency is based on applying the machine learning XGboost algorithm. The team has analysed that XGBoost performs the highest prediction output. The prediction model's performance was evaluated based on MAE, RMSE, and MAPE. In addition, the XGBoost model contains a minor

record of RMSE compared to others, where the statistical analysis is as follows: 0.68, 0.765, and 0.005. The project is limited to only one machine-learning model, and the provided data source is DigitalCoinPrice. (n.d.). With a limited time, frame from 2018 to 2021.

2. Data & Project Management Plan

2.1 Data Management Plan

The dataset chosen for this project is the Coinmarket cap. This dataset consists of data on coins, i.e., Bitcoin, Ethereum, Aave, Dogecoin and Ripple. We will fetch data suitable for the project requirements from the inception of each of these coins. We can consider data from previous years, which differ for each coin. For example, the Bitcoin data can be fetched from approximately 2013.

This dataset is accessed by an external Python library, 'cryptocmd'. The data can be collected based on the required coin. The repository managed by (guptarohit. n.d.) emphasises and illustrates an example of the data collection method. Currently, the data is in CSV format and stored in a GitHub repository (gudur, N., n,d.). A new repository is created solely for the project, where the data and code files are stored.

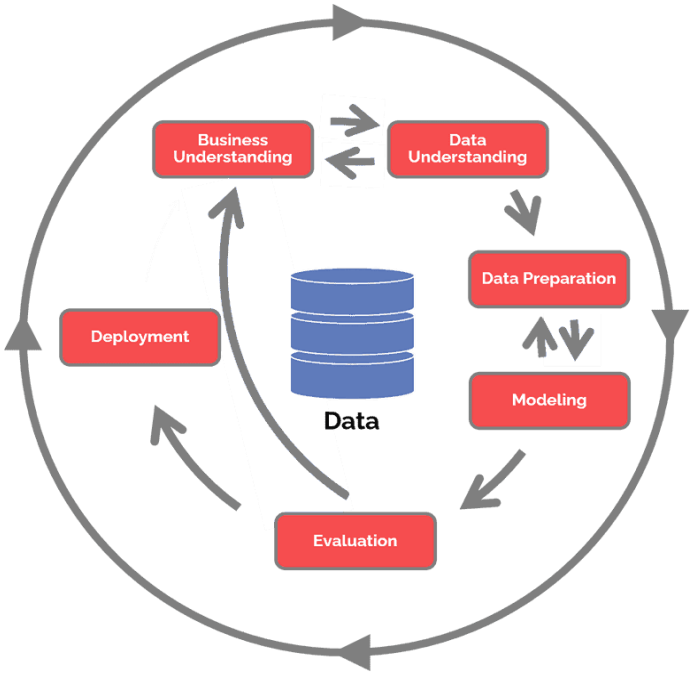
It contains the Date, Open, High, Low, Close, Volume, Market Cap, Time Open, High, Time Low, and Time Close for each coin mentioned above. This data will be used to build the machine learning models and must be thoroughly cleaned and handled; overlooking this will result in inaccurate results. The preliminary checks show no need for data labelling. However, we plan to incorporate them if needed.

2.2 Project Development Methodology

We have chosen the CRISP-DM methodology among the available methods. The CRISP-DM is most suitable due to its structured approach to data mining; through data preparation, modelling to evaluation, and deployment, we believe this model provides a detailed roadmap for understanding the project's objective and, most importantly, the data. This model explained in Figure 2 helps us plan each step methodically and effectively and best suits data science projects.

Figure 2

CRISP-DM



Note. The six phases of the CRISP-DM methodology

2.2.1 Business understanding

Considering the importance of cryptocurrency in today's digital economy, assets like Bitcoin and Ethereum have become increasingly popular. They are now serving as alternative digital assets and means of exchange. Considering the growth of cryptocurrency, investors, traders, and financial institutions resort to understanding the price fluctuations to make a well-informed decision. Modelling accurate prediction models for cryptocurrency prices offers valuable insights for investment strategies, risk management and portfolio diversification. The substantial profits in cryptocurrency are helpful for both seasoned traders and newcomers. The primary focus is the need for reliable tools and methods to navigate the volatile market. By creating a robust price prediction model, researchers and analysts can contribute to the

comprehension and progress of cryptocurrencies, implementing informed decision-making and promoting growth and stability in the growing cryptocurrency world. Furthermore, cryptocurrency price prediction is crucial in addressing challenges connected to market efficiency and price discovery, ensuring that market efficiency and price discovery, and ensuring that the market operates effectively and accurately reflects the inherent value of these digital assets. These are highly decentralised, often traded across various exchanges, and influenced by many factors, including market sentiment, regulatory development, and technological advancements. Predicting their prices can accurately contribute to the efficient functioning of the market by reducing information asymmetry and improving liquidity. This project aims to forecast the “high” cost of selected cryptocurrencies with the utmost accuracy, providing the user with market insights and acting as a resourceful aid in their investment journey.

2.2.2 Data Understanding

As claimed by the authors (Roosenboom, P., 2020). CoinMarketCap is a leading website for tracking exchange-traded cryptocurrencies. The initial exploration found that the dataset has specific null values; we are currently working on achieving a suitable dataset for modelling. To better understand the data, we perform summary statistics and a few exploratory data analyses using data visualisation packages like Matplotlib and Seaborn. This helps us understand various aspects of the data.

We consider each coin's data from its inception for the project requirement. This data is collected to understand and determine cryptocurrency trends and ensure we have enough to prepare for the model.

2.2.3 Data Preparation

After understanding the dataset and the current business trends, we prepared the data for modelling. In the Data Preparation phase, we begin by cleaning the data. The next step involves ensuring that the dataset has no inconsistencies or errors, that missing values and duplicates are handled, and that the data is normalized or standardized to secure uniform scales. It is necessary to establish Data integrity before modelling, as missing out on doing so might result in bias and compromise the model's performance. Jupyter and VS code are the primary tools. The interactive nature of Jupyter Notebooks allows for enriched visualizations, while VS code extensions help in project development.

To proceed further with data modelling, we will split the training dataset into an 80-20 ratio of train and validation set; the validation set will be made before running the test data, which can be improvised if necessary.

2.2.4 Modeling

After preparation, the data is set for the modelling phase. We first proceed with the linear regression model, using the historical data available in the dataset to determine the coefficients for each cryptocurrency parameter. The model uses the coefficients to predict outcomes for new, unseen data. The other machine learning models are XGBoost, Random Forests, and LSTM.

The features are extracted using the correlation matrix; the features with high correlation values are considered. We aim to optimise the developed models by tuning the hyperparameter. After training the models, we consider k-fold cross-validation to optimise them further and ensure they aren't overfitting the training data.

2.2.5 Evaluation

The evaluation metrics used to evaluate the machine learning models are R^2 and MSE. The evaluation report and the most suitable model will be generated based on the different scores obtained during the machine learning models. The model with the best scores will be chosen to determine cryptocurrency trends.

2.2.6 Deployment

The machine learning models are deployed on GitHub. The live data is automated using GitHub actions as part of the data preparation. A step-by-step manual will be generated for the automated processes. All the manuals, performance reports, and code files are added to the GitHub repository for timely access and concurrent availability to all the team members. The team members can collaborate on their work in the GitHub repository.

2.3 Project Organization Plan

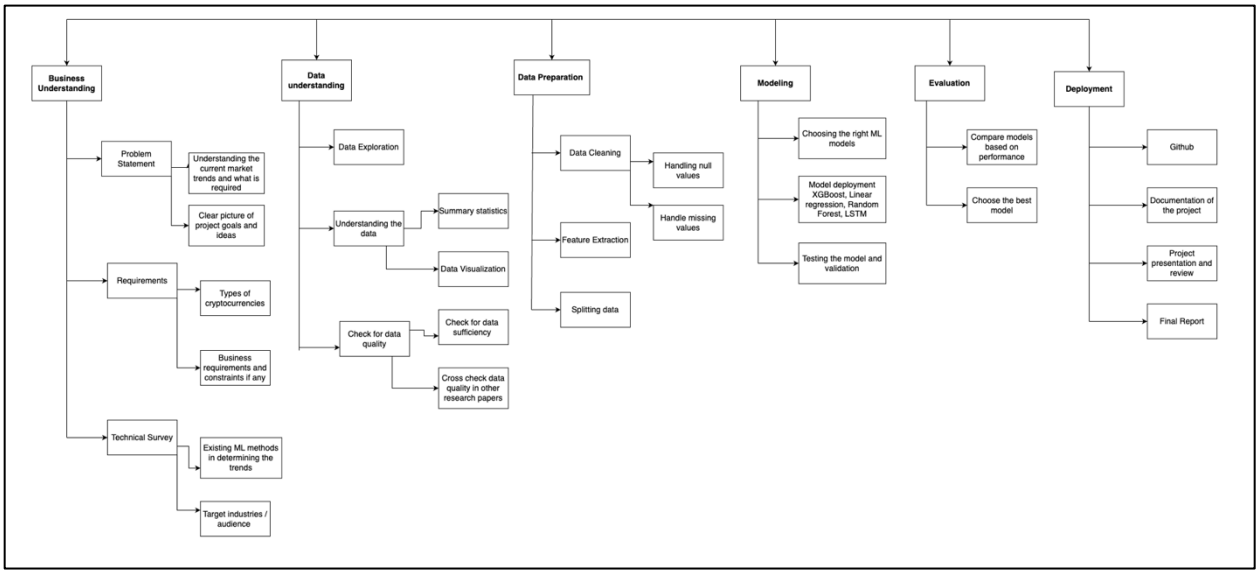
The Project Organization plan mainly consists of a Work Breakdown Structure. As mentioned above, CRISP-DM is well suited for data science projects, and for analytics models, we have chosen the same model to represent the work Breakdown Structure. Fig 2 shows the Work Breakdown Structure, which includes knowledge of the Problem Statement, Requirements to execute the project, and a Survey of existing methods for determining trends and exploring the target industries and audience.

We will be using the Coinmarket cap dataset. To understand the dataset better, we will plot the necessary graphs and perform summary statistics to understand the dataset. After/ Post checking for the quality of the dataset, we proceed with Data preparation, which involves cleaning the data and choosing the right features for extraction.

The project's next phase comprises modelling. We have chosen four machine learning models to implement, test, and validate. We evaluate the models according to project requirements and compare their performance to select the best one for determining cryptocurrency trends. As per the project requirements, we compare the models based on performance, and the best model is chosen to determine the trends of the Cryptocurrency. The last phase of the project comprises deployment. We are considering deploying the model in GitHub, followed by step-by-step documentation, project presentation, and review. The detailed steps in the Work Breakdown Structure are explained in Figure 3.

Figure 3

Work Breakdown Structure



Note: The Work Breakdown Structure is illustrated above

2.4 Project Resource Requirement

As part of the project resource planning, the data loading and processing are done using tools such as VS code and Jupyter Notebook for an extensive range of extensions and enhanced visuals, respectively. Visualisation tools such as Tableau build interactive dashboards to

understand critical takeaways. A GitHub repository is created solely for collaboration and has concurrent access to the stored code files and data. The team members can access and collaborate on the GitHub repository. As a part of the data storing process, AWS S3 is also considered. Machine learning packages such as Scikit Learn, tensor flow, and machine learning models such as AWS SageMaker can help us build, train, and deploy. They are also highly cost-effective and are used to implement machine learning models. The API interface Cryptocmd is used to fetch the data. A detailed list of various software and hardware tools used is given in Table 1.

Table 1

Resources and Cost Estimation

Utility	Resource	Tools	Duration	Unit Cost
Data Storage	Software	AWS S3	2 months	0$ (free tier)
Visualisation Tool	Software	Tableau	2 months	0$ (Student ID exception)
Machine Learning Models	Software	AWS SageMaker	2 months	0$ (free tier)
Machine learning packages	Software	Scikit Learn, tensor flow	2 months	0$
Data Loading and processing	Software, Interface	VS code, jupyter notebook, Crypto CMD, Coinmarketcap	2 months	0$

Local Machines	Hardware	64-bit machine	2 months	1500$
Collaboration	Software	Github	2 months	0$

Note. All resources and cost estimations are as per project requirements.

2.5 Project Schedule

The Gantt chart helps us evaluate completed tasks, organise tasks, and plan according to a stipulated timeline. The Figure 4 depicts the timeline of various phases of the project. This helps in timely collaboration and ensures timely deliverables. For the completion of the project, the project is divided into tasks. These tasks are further divided into subgroups. The Project planning, which is the first phase, has the project background, the project requirements, project deliverables, Technology and Solution Survey and the Literature Survey. These tasks are executed by the team, for nearly a month.

The second stage is the Data and Project Management Plan. This step includes the Data Management Plan, Project Development Methodology, Project Organization Plan, Project Resource Requirement, and the Project Schedule. This process took about 11 days to complete.

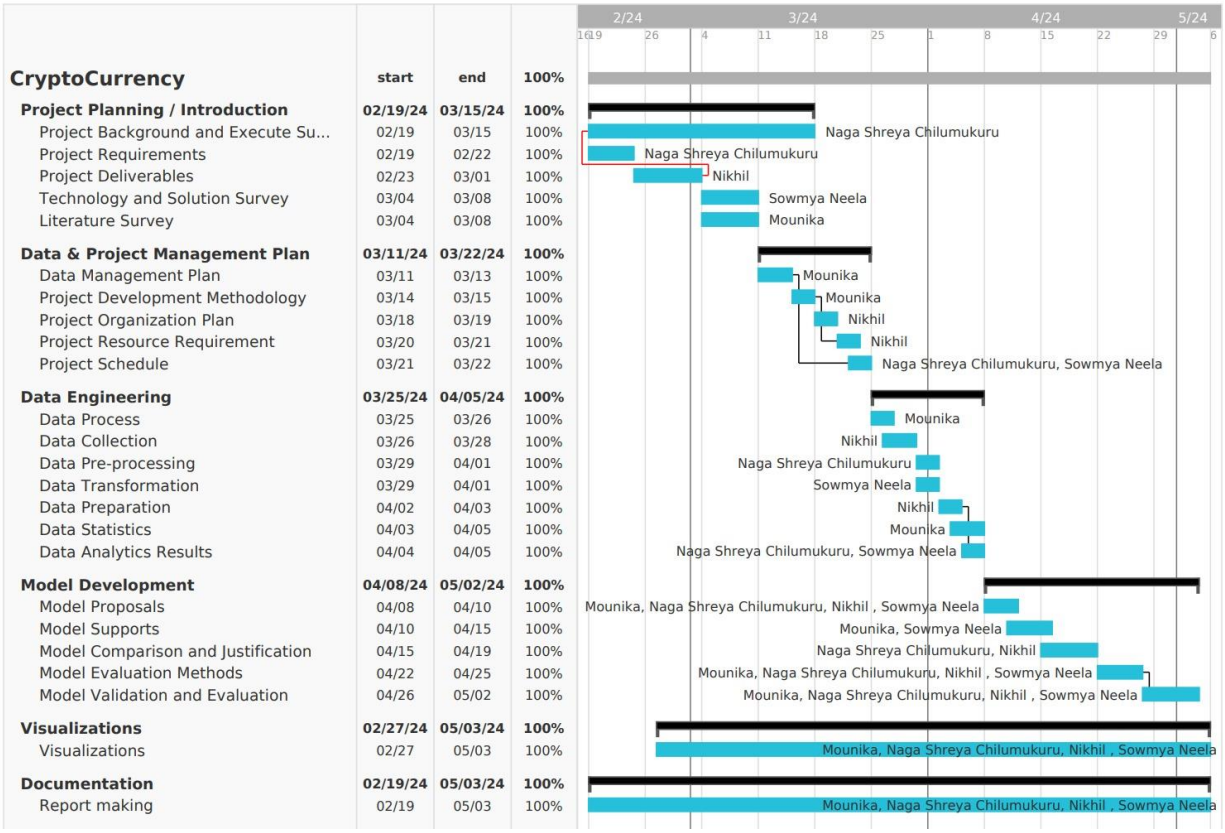
The third phase is the Data Engineering. It includes the Data Process, Data Collection, Data Pre-processing, Data Transformation, Data Preparation, Data Statistics and Data Analytics Results.

The fourth phase is the Model Development which has the Model Proposals, Model Supports, Model Comparison and justification, Model Evaluation Methods and Model Validation and Evaluation.

After the evaluation is done, visualizations are made for easier analysis. The entire process is documented. The same is represented in the form of Gantt Chart in Figure 4.

Figure 4

Gantt Chart



Note. The timelines of different phases are illustrated in the above Gantt Chart.

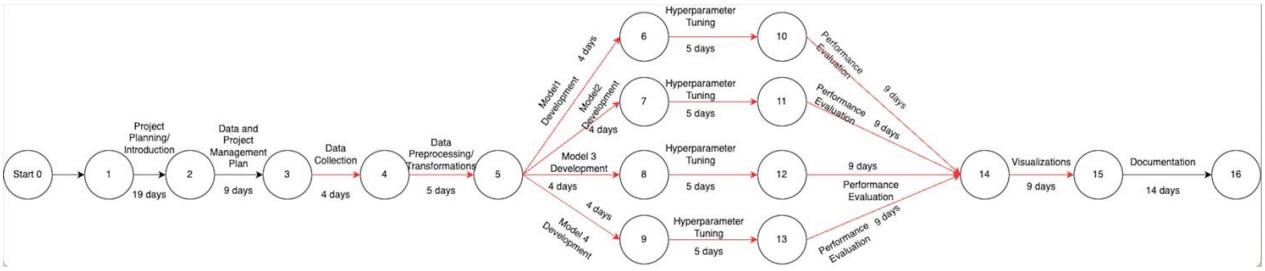
The Pert chart as shown in Figure 5 is categorized based on the activities. The activities begin with 0 and are as follows:

0 - Start of the project,1 - Objectives, Requirements and deliverables are finalised after the start of the project along with the Collection and examination of research papers 2 -Developed a management plan and methodology and conducted resource requirements. 3, 4- data collection begins with an analytical phase to understand the data. The stored data is then cleaned, i.e. scrubbed of errors, duplicates, and redundant information. 5- The data is prepared, transformed, and available for predictive modelling and model development. 6 -9 - The transformed data is then partitioned into training, validation, and test sets in preparation for model training. 10-13 -

Each model performs hyperparameter tuning to optimise its performance. These steps are independent and can be performed simultaneously. 14 - This indicates that all models have been trained and possibly tuned. Trained models are tested against various metrics and refined based on performance. This step is based on the completed training of all models. 15 - Visualizations - Interactive and visually appealing dashboards are built to help understand trends, patterns, and predictions. 16 - Documentation of the entire process and results is completed, which can be done after the model has been thoroughly evaluated and refined.

Figure 5

Pert Chart



Note. The different stages are illustrated in the above Pert Chart.

3. Data Engineering

3.1 Data Process

To determine the trends of cryptocurrency, we would need the dataset which captures the information such as opening price of the cryptocurrency of the day, the closing price of the cryptocurrency of the day, the lowest price, and the closing price etc. In the initial exploration of the dataset, we have found various datasets which align with the requirements of our problem statement like Coinmarket cap, cryptocurrency historical data from data world, every cryptocurrency daily market price from Kaggle etc. Given various datasets Coinmarketcap is best suited to address our problem statement which consist of various types of information such as Date, Open, High, Low, Close, Volume, Marketcap and the timestamps of Open, Close, Low and High prices. This dataset is well suited as it comprises of historical cryptocurrency data and the free crypto API which is available can be used to get the best, most accurate real time data for various coins. This dataset can be accessed using Cryptocmd. Cryptocmd is a Web Scraper written in Python language to scrape the crypto currency data from CoinMarketCap.

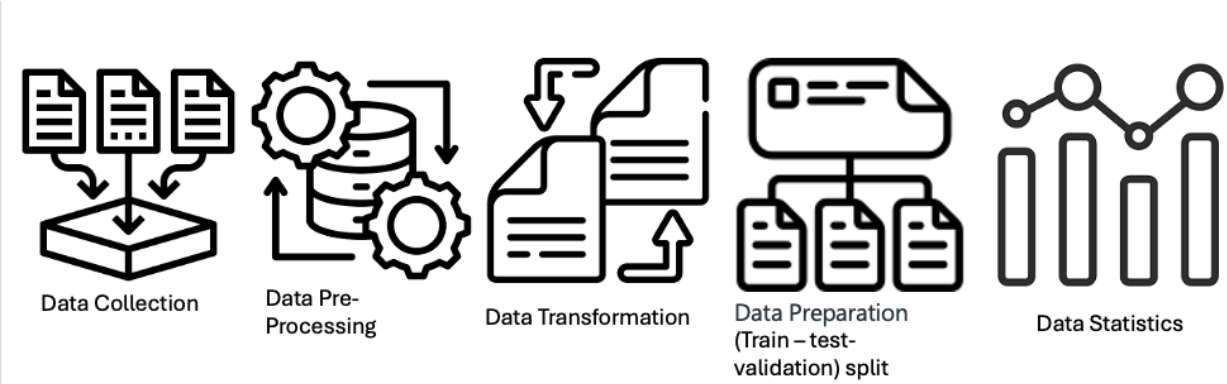
As part of the intial data process, we have identified the number and type of coins we are going to use in addressing the problem statement. We have identified five major coins i.e., Bitcoin, Doge, Etherium, Ripple and Aave. The coins are chosen as these coins are most prominent in the market and have gained significant attention through market capitalization.

Post selecting the coins, we proceed with the initial data exploration and data preprocessing to identify the inconsistencies and address if there are any null or missing values. Check for outliers, Post Preprocessing the data we proceed with Transformation and Preparation where we used Min Max scalar to normalize the feature between 0 and 1, Post transformation the data is split into multiple training and validation sets using K-fold cross validation method. The

collection, preprocessing, transformation, and preparation are briefly explained in the following sections. Figure 6 depicts different phases of data process and engineering.

Figure 6

Data Process and Engineering Diagram



Note. Different phases of Data process and Data Engineering Diagram

3.2 Data Collection

Reflecting on our problem statement, we have used CryptoCMD which is used to fetch the live data. In the data collection process, we use the web Scrapper CryptoCMD which is written in Python to scrape the cryptocurrency data from coin market Cap Figure 6. Depicts the python code which uses CryptoCMD library to fetch and save the cryptocurrency data. We have ensured a logging statement to log the activity of fetching data for a particular ‘coin_name’. this is useful for debugging purposes. The Python code is useful to fetch the data for the selected cryptocurrencies and keep a record of these fetches. A timely record is generated from CryptoCmd for each selected coin. The CryptoCmd acts as a flexible gateway providing a pandas data frame directly or can also be used to export the dataset in the csv format. We are making sure to record all the csv generated files to be incorporated in the Github repository to main concurrent collaboration. A snippet for collection of data is depicted in Figure 7.

Figure 7

Web Scrapping with Python

```python
def get_coin_data_csv(coin_code: str, coin_name: str) -> None:

    can_fetch, fetch_record = can_fetch_data(coin_name)

    if can_fetch:
        log.info(f"Fetching data for {coin_name}")

        scraper = CmcScraper(coin_code=coin_code, coin_name=coin_name)

        scraper.export("csv", name=f"./coin_datasets/{coin_code}_all_time")

        with open(FETCH_RECORD_FILE, "w", encoding="utf-8") as f:
            fetch_record[coin_name] = time.ctime(time.time())
            json.dump(fetch_record, f)


if __name__ == "__main__":
    get_coin_data_csv("btc", "bitcoin")
    get_coin_data_csv("eth", "ethereum")
    get_coin_data_csv("aave", "aave")
    get_coin_data_csv("doge", "dogecoin")
    get_coin_data_csv("xrp", "xrp")
```

Note. Python code snippet for collection of data

A sample dataset using CrytpoCMD containing various columns like Volume, Market Cap, Open close are depicted in Figure 8. The sample dataset is of the Bitcoin, similar datasets are generated for the selected coins, i.e, Ripple, Etherium, Aave and Dogecoin. The merged dataset as shown in Figure 9 of the selected 5 coins comprises of approximately 17k rows 11 columns. As part of collection data process, the latest record of the data is updated as Cryptocmd provides the pandas data frame directly. The dataset of all the selected coins is handled and collected similarly. The sample Bitcoin Dataset comprises of the columns which captures accurate information about the date, Open which the price of the cryptocurrency at the beginning of the day, High which is the highest price of cryptocurrency within the trading day, Low which is the lowest price of the cryptocurrency within the trading day and so on.

Figure 8

Sample Bitcoin Dataset

Date	Open	High	Low	Close	Volume	Market Cap	Time Open	Time High	Time Low	Time Close
22-03-2024	65489.93	66623.75	62355.37	63778.76	41401116964.	12544576006	2024-03-22T	2024-03-22T	2024-03-22T	2024-03-22T
21-03-2024	67911.58	68199.99	64580.92	65491.39	4448035056	12875745619	2024-03-21T	2024-03-21T	2024-03-21T	2024-03-21T
20-03-2024	61930.15	68115.26	60807.79	67913.67	6679263438	13351319595	2024-03-20T	2024-03-20T	2024-03-20T	2024-03-20T
19-03-2024	67556.13	68106.93	61536.18	61912.77	74215844794	12174790936	2024-03-19T	2024-03-19T	2024-03-19T	2024-03-19T
18-03-2024	68371.31	68897.13	66594.23	67548.59	49261579492	13280150484	2024-03-18T	2024-03-18T	2024-03-18T	2024-03-18T
17-03-2024	65316.34	68845.72	64545.32	68390.62	44716864318	13439782054	2024-03-17T	2024-03-17T	2024-03-17T	2024-03-17T
16-03-2024	69392.49	70046.27	64801.4	65315.12	46842198370	12838702891	2024-03-16T	2024-03-16T	2024-03-16T	2024-03-16T
15-03-2024	71387.87	72357.13	65630.69	69403.77	78320453975	13639288411	2024-03-15T	2024-03-15T	2024-03-15T	2024-03-15T
14-03-2024	73079.37	73750.07	68563.02	71396.59	5959460569	14030929538	2024-03-14T	2024-03-14T	2024-03-14T	2024-03-14T
13-03-2024	71482.12	73637.47	71334.09	73083.5	48212536929	14362718226	2024-03-13T	2024-03-13T	2024-03-13T	2024-03-13T
12-03-2024	72125.12	72825.66	68728.85	71481.29	6255443452	14048103116	2024-03-12T	2024-03-12T	2024-03-12T	2024-03-12T
11-03-2024	69020.55	72850.71	67194.89	72123.9	65716656764	14173807477	2024-03-11T	2024-03-11T	2024-03-11T	2024-03-11T
10-03-2024	68500.26	70005.2	68239.98	69019.79	35683977531	13562844931	2024-03-10T	2024-03-10T	2024-03-10T	2024-03-10T
09-03-2024	68299.26	68673.06	68053.13	68498.88	21609650379	13460146149	2024-03-09T	2024-03-09T	2024-03-09T	2024-03-09T
08-03-2024	66938.09	70083.05	66230.45	68300.1	59202881172.	13419928939	2024-03-08T	2024-03-08T	2024-03-08T	2024-03-08T
07-03-2024	66099.74	68029.92	65655.53	66925.48	46989543159	13151820560	2024-03-07T	2024-03-07T	2024-03-07T	2024-03-07T
06-03-2024	63776.05	67637.93	62848.67	66106.8	68750229073	12986351843	2024-03-06T	2024-03-06T	2024-03-06T	2024-03-06T

Note. Samples of raw data from Bitcoin dataset

Figure 9

Sample merged dataset of all the coins.

```
merged_df.head()
```

	Date	Open	High	Low	Close	Volume	Market Cap	Time Open	Time High	Time Low	Time Close	name
0	2024-05-12	60793.502025	61818.154883	60632.602445	61448.394672	1.380046e+10	1.210436e+12	2024-05-12T00:00:00.000Z	2024-05-12T16:22:00.000Z	2024-05-12T07:35:00.000Z	2024-05-12T23:59:59.999Z	btc
1	2024-05-11	60793.357007	61451.153731	60492.625374	60793.709599	1.384227e+10	1.197453e+12	2024-05-11T00:00:00.000Z	2024-05-11T15:19:00.000Z	2024-05-11T11:18:00.000Z	2024-05-11T23:59:59.999Z	btc
2	2024-05-10	63055.190995	63446.742515	60208.781547	60792.776791	2.780495e+10	1.197421e+12	2024-05-10T00:00:00.000Z	2024-05-10T12:05:00.000Z	2024-05-10T17:52:00.000Z	2024-05-10T23:59:59.999Z	btc
3	2024-05-09	61191.200531	63404.915301	60648.075424	63049.959257	2.545334e+10	1.241942e+12	2024-05-09T00:00:00.000Z	2024-05-09T22:53:00.000Z	2024-05-09T11:05:00.000Z	2024-05-09T23:59:59.999Z	btc
4	2024-05-08	62332.642508	62986.084242	60877.127524	61187.941401	2.608817e+10	1.205202e+12	2024-05-08T00:00:00.000Z	2024-05-08T04:22:00.000Z	2024-05-08T23:24:00.000Z	2024-05-08T23:59:59.999Z	btc

```
merged_df.tail()
```

	Date	Open	High	Low	Close	Volume	Market Cap	Time Open	Time High	Time Low	Time Close	name
16286	2013-08-08	0.004397	0.004424	0.004175	0.004254	0.0	3.325863e+07	2013-08-08T00:00:00.000Z	2013-08-08T00:06:03.000Z	2013-08-08T15:21:04.000Z	2013-08-08T23:59:59.999Z	xrp
16287	2013-08-07	0.004669	0.004682	0.004333	0.004417	0.0	3.453412e+07	2013-08-07T00:00:00.000Z	2013-08-07T04:01:18.000Z	2013-08-07T12:46:19.000Z	2013-08-07T23:59:59.999Z	xrp
16288	2013-08-06	0.005637	0.005661	0.004629	0.004680	0.0	3.659101e+07	2013-08-06T00:00:00.000Z	2013-08-06T01:26:06.000Z	2013-08-06T20:51:06.000Z	2013-08-06T23:59:59.999Z	xrp
16289	2013-08-05	0.005875	0.005980	0.005613	0.005613	0.0	4.387916e+07	2013-08-05T00:00:00.000Z	2013-08-05T03:21:05.000Z	2013-08-05T23:56:05.000Z	2013-08-05T23:59:59.999Z	xrp
16290	2013-08-04	0.005874	0.005927	0.005874	0.005882	0.0	4.598358e+07	2013-08-04T00:00:00.000Z	2013-08-04T23:41:04.000Z	2013-08-04T18:51:05.000Z	2013-08-04T23:59:59.999Z	xrp

Note. Samples of raw data from merged dataset

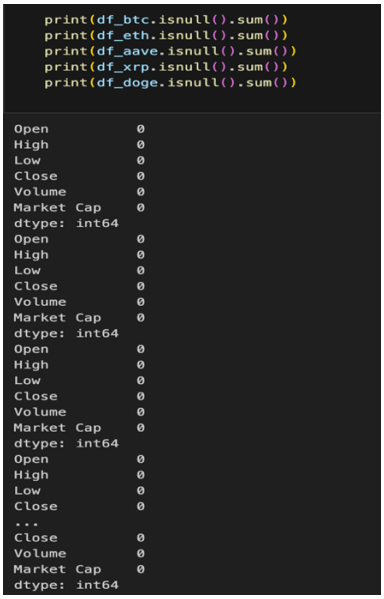
3.4 Data Pre-processing

Post collection of the data we proceed with the next step of Data Engineering which is Data Pre-processing which is a crucial step. This is the first step and plays a vital role while creating a machine learning model.

We have checked for Null values present in each coin dataset, from Figure 10. It is observed that the dataset has no null values. The null values are checked for all the datasets and merged in Figure 11 for better understanding. Given that a new record is generated everyday no duplicates are observed as well. Figure 12 which is a consolidated from the datasets of the selected coins verifies that these datasets do not have any duplicates. Figure 13 depicts the presence of no duplicates in the merged dataset.

Figure 10

Check for null values in the dataset.



Note. Null values in the dataset of the selected coins

Figure 11

Check for null values in merged dataset.

```
merged_df.isnull().sum()
```

```
Date          0
Open          0
High          0
Low           0
Close         0
Volume        0
Market Cap    0
Time Open     0
Time High     0
Time Low      0
Time Close    0
name          0
dtype: int64
```

```
merged_df.isna().sum()
```

```
Date          0
Open          0
High          0
Low           0
Close         0
Volume        0
Market Cap    0
Time Open     0
Time High     0
Time Low      0
Time Close    0
name          0
dtype: int64
```

Note. Null values in the dataset of the merged dataset coins

Figure 12

Check for duplicates in the dataset.

```
if num_duplicates > 0:
    print(f"There are {num_duplicates_bitcoin} duplicate rows in the dataset.")
else:
    print("No duplicate rows found in the dataset.")
if num_duplicates > 0:
    print(f"There are {num_duplicates_aave} duplicate rows in the dataset.")
else:
    print("No duplicate rows found in the Aave dataset.")
if num_duplicates > 0:
    print(f"There are {num_duplicates_ethereum} duplicate rows in the dataset.")
else:
    print("No duplicate rows found in the Ethereum dataset.")
if num_duplicates > 0:
    print(f"There are {num_duplicates_dogecoin} duplicate rows in the dataset.")
else:
    print("No duplicate rows found in the Dogecoin dataset.")
if num_duplicates > 0:
    print(f"There are {num_duplicates_ripple} duplicate rows in the dataset.")
else:
    print("No duplicate rows found in the Ripple dataset.")
```

```
No duplicate rows found in the dataset.
No duplicate rows found in the Aave dataset.
No duplicate rows found in the Ethereum dataset.
No duplicate rows found in the Dogecoin dataset.
No duplicate rows found in the Ripple dataset.
```

Note. Duplicate values in the dataset of the selected coins

Figure 13

Check for duplicates in the merged dataset.

```
merged_df.duplicated().sum()
```

```
0
```

```
merged_df.duplicated()
```

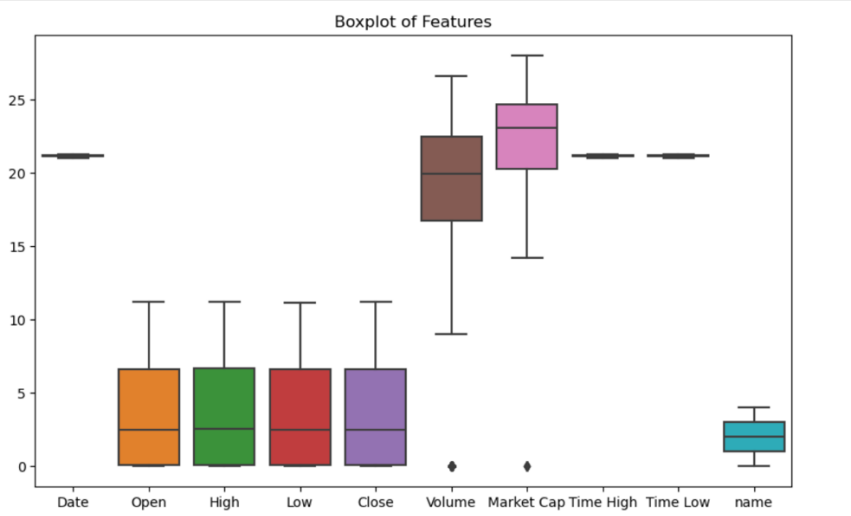
```
0        False
1        False
2        False
3        False
4        False
         ...
16286    False
16287    False
16288    False
16289    False
16290    False
Length: 16291, dtype: bool
```

Note. *Note. Duplicate values in the merged dataset of the coins*

As a part of our data preprocessing, we have checked for the outliers in the merged dataset. Figure 14 shows the presence of outliers in the merged dataset. The IQR method is used to identify the outliers. The bottom and top of the box are the first Q1 and third Q3 quartiles, the band inside the box is the second quartile Q2 which is the median. The outliers are the points which lie outside the range of whiskers.

Figure 14

Boxplot of the merged dataset



Note. Boxplot showing the presence of outliers in the merged dataset.

To handle the outliers as mentioned earlier, the IQR method is used Figure 15 shows the code snippet and the box plot depicting the handling of outliers.

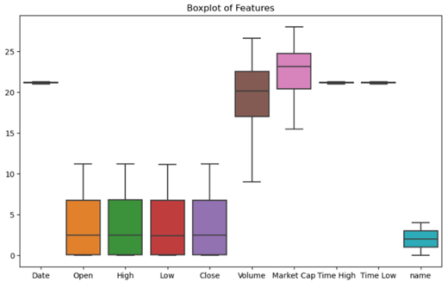
Figure 15

Code snippet and box plot.

```
# dropping outliers and running the model
cols = ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap', 'Time High', 'Time Low','name']
Q1 = merged_log[cols].quantile(0.25)
Q3 = merged_log[cols].quantile(0.75)
IQR = Q3 - Q1

# Define the boundaries for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter the DataFrame to remove outliers
condition = ((merged_log[cols] >= lower_bound) & (merged_log[cols] <= upper_bound)).all(axis=1)
merged_filtered = merged_log[cols][condition]

# merged_filtered = merged_log.loc[(merged_log[num_cols] >= lower_bound) & (merged_log[num_cols] <= upper_bound).all(axis=1)]

# Print the shape of the original and filtered DataFrame to see the effect of outlier removal
print("Original DataFrame shape:", merged_log[cols].shape)
print("Filtered DataFrame shape:", merged_filtered.shape)
```

```
Original DataFrame shape: (16286, 10)
Filtered DataFrame shape: (15883, 10)
```

```
# print number of outliers -16276-15873 = 403 = 2.47% removed
plt.figure(figsize=(10, 6))
sns.boxplot(data=merged_filtered)
plt.title('Boxplot of Features')
plt.show()
```



Box plot for various features post handling the outliers.

Post checking for null values we have generated a correlation matrix. A correlation matrix is used to evaluate the relationship between two variables in a data set. Each cell in the table shows the correlation between two variables. The value is in the range of -1 to 1. From Figure 16, we observe that the volume has low correlation coefficient. Figure 17 depicts the correlation matrix for the merged dataset

Figure 16

Correlation Matrix

	Open	High	Low	Close	Volume	Market Cap
Open	1.000000	0.999535	0.999211	0.998930	0.709292	0.998478
High	0.999535	1.000000	0.999070	0.999531	0.714290	0.998928
Low	0.999211	0.999070	1.000000	0.999431	0.700284	0.999201
Close	0.998930	0.999531	0.999431	1.000000	0.708068	0.999557
Volume	0.709292	0.714290	0.700284	0.708068	1.000000	0.702288
Market Cap	0.998478	0.998928	0.999201	0.999557	0.702288	1.000000

Note. Correlation matrix from the bitcoin dataset

Figure 17

Correlation Matrix of the merged dataset

```
correlation_matrix = merged_df.corr()
```

```
correlation_matrix
```

	Date	Open	High	Low	Close	Volume	Market Cap	Time Open	Time High	Time Low	Time Close	name
Date	1.000000	0.284846	0.284179	0.285886	0.284887	0.332294	0.380784	1.000000	1.000000	1.000000	1.000000	-0.122949
Open	0.284846	1.000000	0.999703	0.999488	0.999307	0.702066	0.950232	0.284846	0.284854	0.284848	0.284846	-0.317515
High	0.284179	0.999703	1.000000	0.999406	0.999697	0.705409	0.950607	0.284179	0.284188	0.284179	0.284179	-0.317444
Low	0.285886	0.999488	0.999406	1.000000	0.999638	0.696069	0.950561	0.285886	0.285895	0.285886	0.285886	-0.317801
Close	0.284887	0.999307	0.999697	0.999638	1.000000	0.701250	0.950908	0.284887	0.284898	0.284886	0.284887	-0.317585
Volume	0.332294	0.702066	0.705409	0.696069	0.701250	1.000000	0.748778	0.332294	0.332302	0.332288	0.332294	-0.184767
Market Cap	0.380784	0.950232	0.950607	0.950561	0.950908	0.748778	1.000000	0.380784	0.380797	0.380783	0.380784	-0.228379
Time Open	1.000000	0.284846	0.284179	0.285886	0.284887	0.332294	0.380784	1.000000	1.000000	1.000000	1.000000	-0.122949
Time High	1.000000	0.284854	0.284188	0.285895	0.284898	0.332302	0.380797	1.000000	1.000000	1.000000	1.000000	-0.122958
Time Low	1.000000	0.284848	0.284179	0.285886	0.284886	0.332288	0.380783	1.000000	1.000000	1.000000	1.000000	-0.122941
Time Close	1.000000	0.284846	0.284179	0.285886	0.284887	0.332294	0.380784	1.000000	1.000000	1.000000	1.000000	-0.122949
name	-0.122949	-0.317515	-0.317444	-0.317801	-0.317585	-0.184767	-0.228379	-0.122949	-0.122958	-0.122941	-0.122949	1.000000

Note. Correlation matrix from the merged dataset

To further ensure the quality and improve the usability of the dataset, we have performed data merging i.e., the datasets as shown in Figure 18 of the selected coins were merged to create a combined comprehensive dataset.

Figure 18

Sample dataset of merged coins.

```
[2]: scraper_btc = CmcScraper(coin_code="btc", coin_name="bitcoin")
     df_btc = scraper_btc.get_dataframe()
     df_btc['name'] = 'btc'

     scraper_eth = CmcScraper(coin_code="eth", coin_name="ethereum")
     df_eth = scraper_eth.get_dataframe()
     df_eth['name'] = 'eth'

     scraper_aave = CmcScraper(coin_code="aave", coin_name="aave")
     df_aave = scraper_aave.get_dataframe()
     df_aave['name'] = 'aave'

     scraper_doge = CmcScraper(coin_code="doge", coin_name="dogecoin")
     df_doge = scraper_doge.get_dataframe()
     df_doge['name'] = 'doge'

     scraper_xrp = CmcScraper(coin_code="xrp", coin_name="xrp")
     df_xrp = scraper_xrp.get_dataframe()
     df_xrp['name'] = 'xrp'

     merged_df = pd.concat([df_btc, df_eth, df_aave, df_doge, df_xrp], ignore_index=True)

[3]: merged_df.shape

[3]: (16291, 12)
```

Note. Merged dataset of 5 coins.

A new column “name” is added to the dataset to represent the detail of the coin in the merged dataset. Further the date formats are standardized to UNIX format. The samples of before and after data pre-processing is depicted in Figure 19 and Figure 20

Figure 19

Sample dataset before pre-processing.

```
merged_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16291 entries, 0 to 16290
Data columns (total 12 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        16291 non-null  datetime64[ns]
 1   Open        16291 non-null  float64
 2   High        16291 non-null  float64
 3   Low         16291 non-null  float64
 4   Close       16291 non-null  float64
 5   Volume      16291 non-null  float64
 6   Market Cap  16291 non-null  float64
 7   Time Open   16291 non-null  object
 8   Time High   16291 non-null  object
 9   Time Low    16291 non-null  object
 10  Time Close  16291 non-null  object
 11  name        16291 non-null  object
dtypes: datetime64[ns](1), float64(6), object(5)
memory usage: 1.5+ MB
```

Note. Types of columns before pre-processing

Figure 20

Dataset after pre-processing.

```
# Convert datetime to Unix timestamp
date_columns = ['Time Open', 'Time High', 'Time Low', 'Time Close']
for column in date_columns:
    merged_df[column] = pd.to_datetime(merged_df[column]).astype(int) // 10**9  # Convert nanoseconds to seconds

# Now all date and time columns are converted to numerical format
print(merged_df.dtypes)
```

```
Date            int64
Open          float64
High          float64
Low           float64
Close         float64
Volume        float64
Market Cap    float64
Time Open       int64
Time High       int64
Time Low        int64
Time Close      int64
name           object
dtype: object
```

Note. Date and time columns are converted to numerical format.

3.4 Data Transformation

As part of Data Transformation, we have conducted Feature Engineering, Data Normalization, Feature Transformation and Data Regularization.

As part of Feature Transformation, the time-based columns such as time, open, low were converted into seconds to maintain consistency across all time-related data.

To convert categorical text data into a numerical format we have performed label encoding, to the ‘Name’ column, which is depicted in Figure 21

Figure 21

Label encoding to the name column.

```
cols = ['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap', 'Time High', 'Time Low']
scaler = MinMaxScaler()
scaled_coins_df = pd.DataFrame(scaler.fit_transform(merged_filtered[cols]), columns=merged_filtered[cols].columns, index=merged_filtered[

scaled_coins_df['name'] = merged_filtered['name']
```

```
# run the model including scaling
scaled_coins_df
```

	Date	Open	High	Low	Close	Volume	Market Cap	Time High	Time Low	name
0	1.000000	0.983565	0.983723	0.985248	0.983560	0.815869	0.985486	0.999988	0.999883	1
1	0.999778	0.986826	0.986574	0.984828	0.983559	0.855594	0.985484	0.999736	0.999722	1
2	0.999556	0.984147	0.986516	0.985478	0.986814	0.850561	0.988397	0.999614	0.999437	1
3	0.999334	0.985797	0.985924	0.985815	0.984137	0.851964	0.986000	0.999221	0.999329	1
4	0.999112	0.986979	0.987892	0.987863	0.985796	0.851619	0.987474	0.999038	0.999113	1
...	...	...	...	...	...	...	...	...	...	...
16136	0.068254	0.002430	0.002433	0.002378	0.002400	0.115884	0.296593	0.068419	0.068148	4
16137	0.067980	0.002376	0.002434	0.002375	0.002411	0.123431	0.296972	0.068198	0.067835	4
16138	0.067705	0.002398	0.002416	0.002344	0.002390	0.109397	0.296264	0.067835	0.067763	4
16139	0.067431	0.002419	0.002465	0.002344	0.002397	0.162172	0.296515	0.067551	0.067483	4
16140	0.067157	0.002147	0.002380	0.002073	0.002378	0.164131	0.295847	0.067406	0.067061	4

Note. Code snippet of label encoding to the ‘Name’ column

In our scenario, Feature Engineering is an important step in our dataset preprocessing and transformation processes in Machine Learning. In this step we have transformed our raw data

into a format which is more suitable for our analysis and predictive modeling. Figure 22 represents the mathematical formula for using min-max scaling. The minimum and maximum values of the feature are given by Xmin and Xmax. All numerical features except ‘name’ column are scaled using the Min-Max scaler to normalize the data from the range 0 to 1.

Figure 22

Mathematical formula for min-max scaling

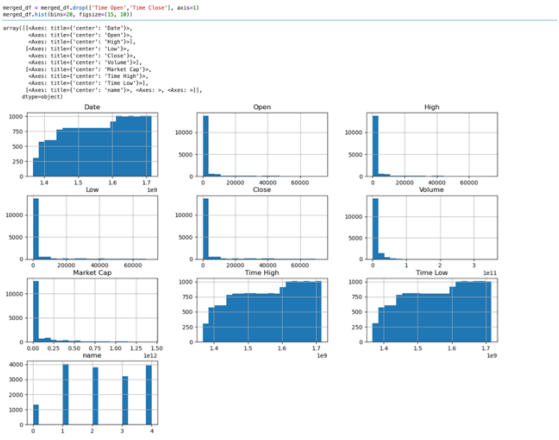
$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Note. The mathematical formula with Xmax and Xmin

Additionally, features like ‘time open’, ‘time close’ which are similar columns were dropped. Exploring the data further, a histogram is plotted for the dataset as shown in Figure 23, which showed the dataset is right-skewed distribution.

Figure 23

Data distribution of the dataset



Note. Histogram depicting the right-skewed data.

Log transformation is applied to address the skewness of the data, Figure 24 depicts the transformed dataset. Normalization is effective only to the “volume” and “market cap” columns.

Figure 24

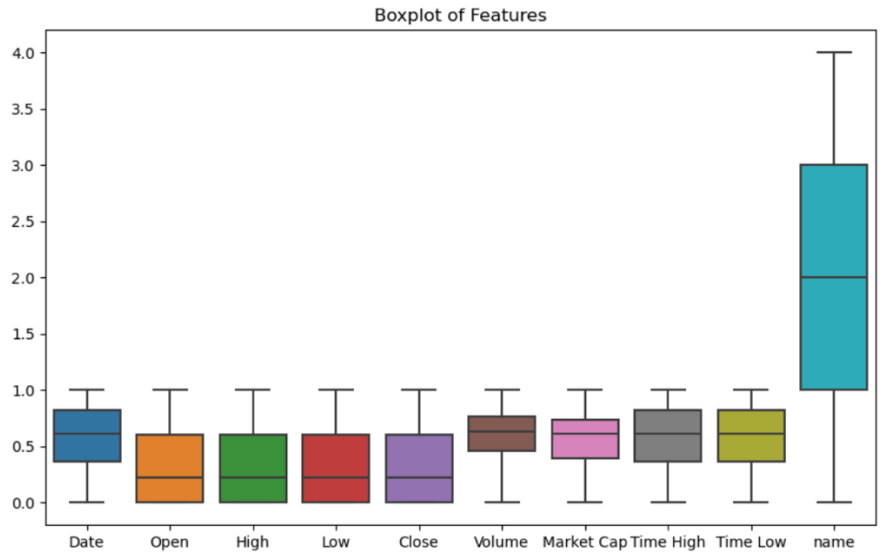
Data distribution after normalization

Note. Histogram depicting the dataset post log-transformation.

Figure 25 depicts the box plot of all the features after min-max scaling is done.

Figure 25

Box plot of all the features



Note. Box plot post scaling.

3.5 Data Preparation

Splitting the transformed data into train, validation and test dataset is very important to avoid overfitting, hyperparameter tuning and improving the performance of machine learning models. The transformed dataset was split into features and target variable such as features(X) which includes all columns except 'high', target (Y) where the 'high' column was used as the target variable. Here, we are using K-fold cross validation technique to do this and try to improve the performance of our model by avoiding overfitting.

K-fold cross validation is a machine learning technique which divides our transformed dataset into K subsets or bins or folds. This is a technique which is useful for evaluating our predictive models. Here, these subsets or folds make sure that every data point in our transformed dataset gets to be in the test dataset exactly once and gets to be in the training dataset in exactly 'K-1' times.

Below are the steps to make this work for our transformed dataset:

3.5.1 Split the data: K is an integer for specifying the number of subsets or folds you want to split the dataset. Now, this transformed dataset is exactly split into K equally sized folds. Each fold or Subset acts as a validation set once and gets to be in the training set for 'K-1' times.

3.5.2 Train and Validate in Loops: For each distinct group, we have a training phase and validation phase. The model is trained first on the K-1 folds and the model is tested on remaining part of the data, which is like a validation/test dataset for that iteration respectively.

3.5.3 Results of performance: Average the results for performance evaluation of the model where R-squared is computed for each iteration and an average of this considered at the end. The Figure 26 shows the code snipped which defines a function that evaluates our machine learning model using K-fold validation. As per the code snippet, we define a function evaluate_model to perform the K-fold cross validation. Kfold object is instantiated and the number of K_Folds are specified. The shuffling is also enabled that our data will be randomized before splitting. The cross-validation scores are calculated for R-squared metrics. Figure 27 and Figure 28 show the images of the two folds dataset. Similarly, it is present for the other folds. Figure 28 depicts the validation output for K-fold cross validation.

Figure 26

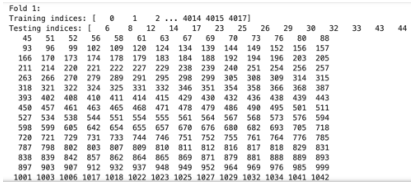
Code snippet of K-fold cross validation

```
# Function to perform K-fold cross-validation
def evaluate_model(X, y, model):
    kf = KFold(n_splits=K_FOLDS, shuffle=True, random_state=42)
    scores = cross_val_score(model, X, y, cv=kf, scoring='r2')
    return scores.mean()
```

Note. Function describing the K-fold cross validation.

Figure 27

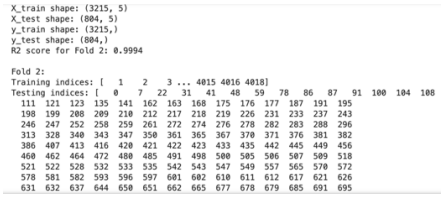
Validation output



Note. K-fold cross validation output.

Figure 28

Validation output

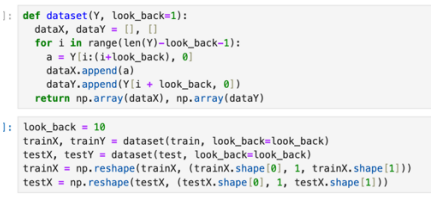


Note. K-fold cross validation output.

In our context, since we are using the K-fold cross validation technique, there is no separate validation. Dataset outside of those formed during each iteration of the above function. Each section or part of the transformed dataset is effectively taken as a validation dataset and is not part of the training dataset. This method is useful when we want all the available data for training to dataset size limitations, or when we need a proper assessment of model stability across various subsets of our transformed data.

Figure 29

Code snippet of Data preparation for LSTM



Note. The python code snippet of data preparation form LSTM machine learning model

The “dataset” function, shown in Figure 29, is written because LSTM needs our input data in batch/sequences, so that this data can be structured accordingly. This helped us in creating a

sequence-based structure where LSTM can learn the patterns in the past and make accurate predictions for the future. A loop_back parameter is set to 10 which is used in preparing both our training and test datasets, also our input data is reshaped to fit the requirements of LSTM.

3.6 Data Statistics

As the final step of the data engineering process, we have performed few data statistics. We have visualized the data pre and post processing to understand the evaluate various methodologies applied. We have fetched the summary statistics of each coin using the describe () function. The sample summary statistics of coins such as bitcoin, ripple are given in the Figure 30 and Figure 31 respectively. The summary statistics has columns such as min, max, mean, count and standard deviation The summary statistics are given only for numeric columns.

Figure 30

Summary statistics of Bitcoin

	Open	High	Low	Close	Volume	Market Cap
count	3982.000000	3982.000000	3982.000000	3982.000000	3.982000e+03	3.982000e+03
mean	13533.969582	13851.982602	13195.252277	13548.848498	1.481073e+10	2.540414e+11
std	16676.586181	17079.606121	16235.485338	16691.810868	1.887319e+10	3.195075e+11
min	68.504997	74.561096	65.526001	68.431000	0.000000e+00	7.784112e+08
25%	587.901749	597.970505	578.798004	588.773010	5.563648e+07	7.889165e+09
50%	6847.105810	7059.060098	6698.722334	6855.385010	6.717410e+09	1.190981e+11
75%	21857.928481	22570.075017	21436.203337	22090.680980	2.503488e+10	4.230336e+11
max	73079.373379	73750.073850	71334.092382	73083.501328	3.509679e+11	1.436272e+12

Note. Summary statistics of numerical columns of Bitcoin dataset

Figure 31

Summary statistics of Ripple

	Open	High	Low	Close	Volume	Market Cap
count	3914.000000	3914.000000	3914.000000	3914.000000	3.914000e+03	3.914000e+03
mean	0.329940	0.343167	0.315830	0.330073	1.427508e+09	1.496877e+10
std	0.352672	0.374516	0.330745	0.352644	2.812730e+09	1.581227e+10
min	0.002809	0.003082	0.002802	0.002810	0.000000e+00	2.196991e+07
25%	0.008595	0.008854	0.008342	0.008609	8.971218e+05	2.577862e+08
50%	0.275578	0.283302	0.267429	0.275745	6.373924e+08	1.164238e+10
75%	0.494933	0.508859	0.480856	0.496375	1.676656e+09	2.416879e+10
max	3.363570	3.841940	3.117340	3.377810	3.695518e+10	1.308535e+11

Note. Summary statistics of numerical columns of Ripple dataset

Figure 32 depicts the summary statistics of the merged dataset.

Figure 32

Summary statistics of merged dataset

```
merged_df.describe()
```

	Date	Open	High	Low	Close	Volume	Market Cap	Time High	Time Low	name
count	1.629100e+04	16291.000000	16291.000000	16291.000000	16291.000000	1.629100e+04	1.629100e+04	1.629100e+04	1.629100e+04	16291.000000
mean	1.561190e+09	3724.360257	3812.567365	3629.660888	3728.002214	6.020875e+09	9.447296e+10	1.561230e+09	1.561230e+09	2.270149
std	9.627966e+07	10621.261851	10871.293349	10347.117321	10629.624624	1.218163e+10	2.088856e+11	9.628105e+07	9.628037e+07	1.288902
min	1.367107e+09	0.000000	0.000089	0.000000	0.000087	0.000000e+00	0.000000e+00	1.367110e+09	1.367154e+09	0.000000
25%	1.480118e+09	0.060361	0.061648	0.058753	0.060443	1.877418e+07	6.612212e+08	1.480127e+09	1.480175e+09	1.000000
50%	1.568074e+09	11.333200	11.680000	11.007400	11.349500	4.724415e+08	1.083560e+10	1.568079e+09	1.568145e+09	2.000000
75%	1.645142e+09	769.215485	782.525513	743.235504	770.982513	5.673720e+09	5.079016e+10	1.645169e+09	1.645204e+09	3.000000
max	1.715472e+09	73079.373379	73750.073850	71334.092382	73083.501328	3.509679e+11	1.436272e+12	1.715531e+09	1.715549e+09	4.000000

Note. Summary statistics of merged dataset.

The info () function helps us retrieving the information of the dataset such as the column type, the count of the non-null attributes. Figures 33 and 34 represent the information of bitcoin and ripple respectively.

Figure shows the summary statistics of the merged dataset.

Figure 33

Information from Bitcoin dataset

```
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        3982 non-null   object
 1   Open        3982 non-null   float64
 2   High        3982 non-null   float64
 3   Low         3982 non-null   float64
 4   Close       3982 non-null   float64
 5   Volume      3982 non-null   float64
 6   Market Cap  3982 non-null   float64
 7   Time Open   3982 non-null   object
 8   Time High   3982 non-null   object
 9   Time Low    3982 non-null   object
 10  Time Close  3982 non-null   object
dtypes: float64(6), object(5)
memory usage: 342.3+ KB
```

Note. Bitcoin info()

Figure 34

Information from Ripple dataset

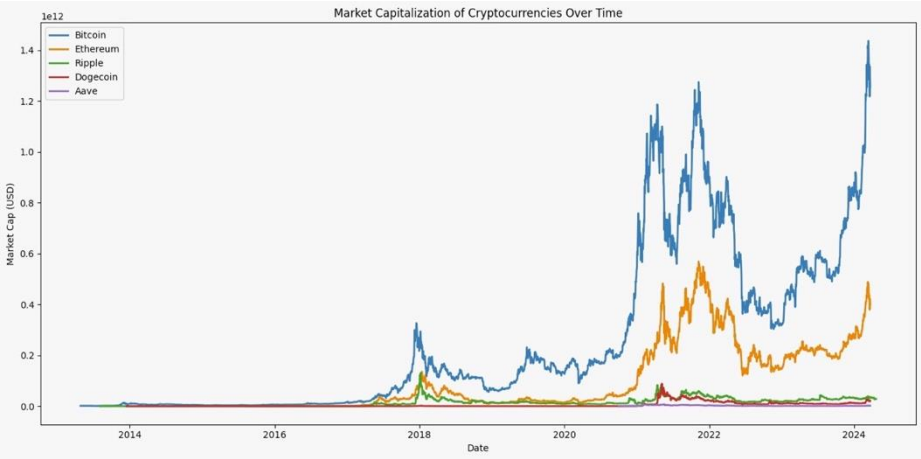
```
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Date        3914 non-null   object
 1   Open        3914 non-null   float64
 2   High        3914 non-null   float64
 3   Low         3914 non-null   float64
 4   Close       3914 non-null   float64
 5   Volume      3914 non-null   float64
 6   Market Cap  3914 non-null   float64
 7   Time Open   3914 non-null   object
 8   Time High   3914 non-null   object
 9   Time Low    3914 non-null   object
 10  Time Close  3914 non-null   object
```

Note. Ripple info()

We have also visualized Market cap pre and post scaling, Figures 35 and 36 depict the Market cap before and after scaling.

Figure 35

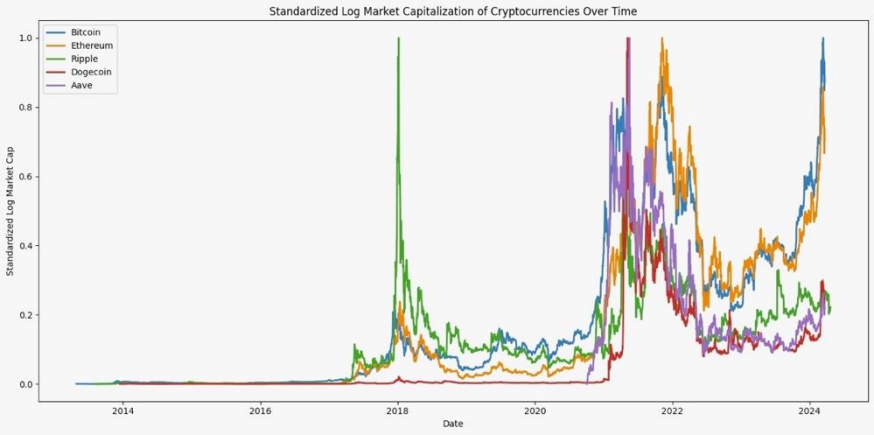
Market cap pre scaling



Note. The image denotes the Market cap column before scaling.

Figure 36

Market cap post scaling

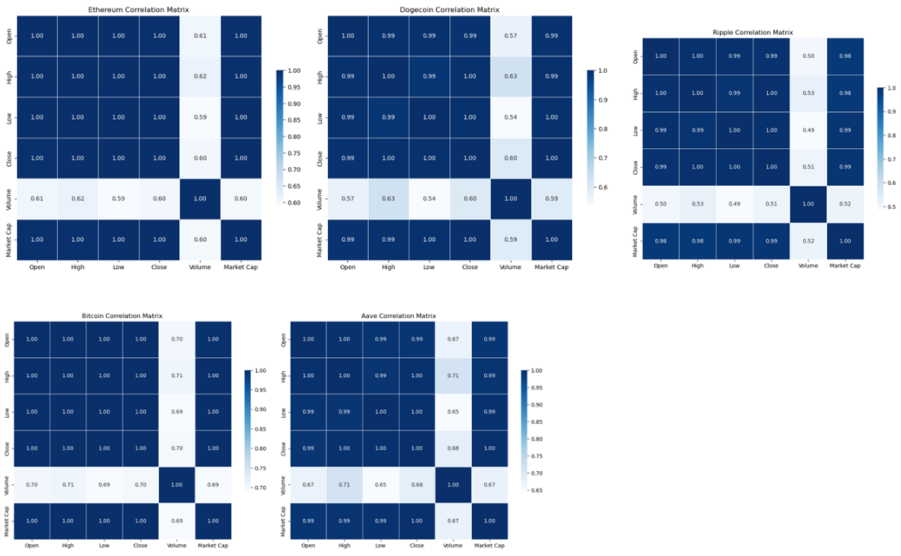


Note. The image denotes Market cap column post scaling.

We have plotted the correlation matrix, based on the correlation matrix depicted in Figure 37, and its dependencies we have considered the columns Market cap, Low, Close, Volume and open of individual datasets and Figure 38 depicts the correlation matrix of the merged dataset

Figure 37

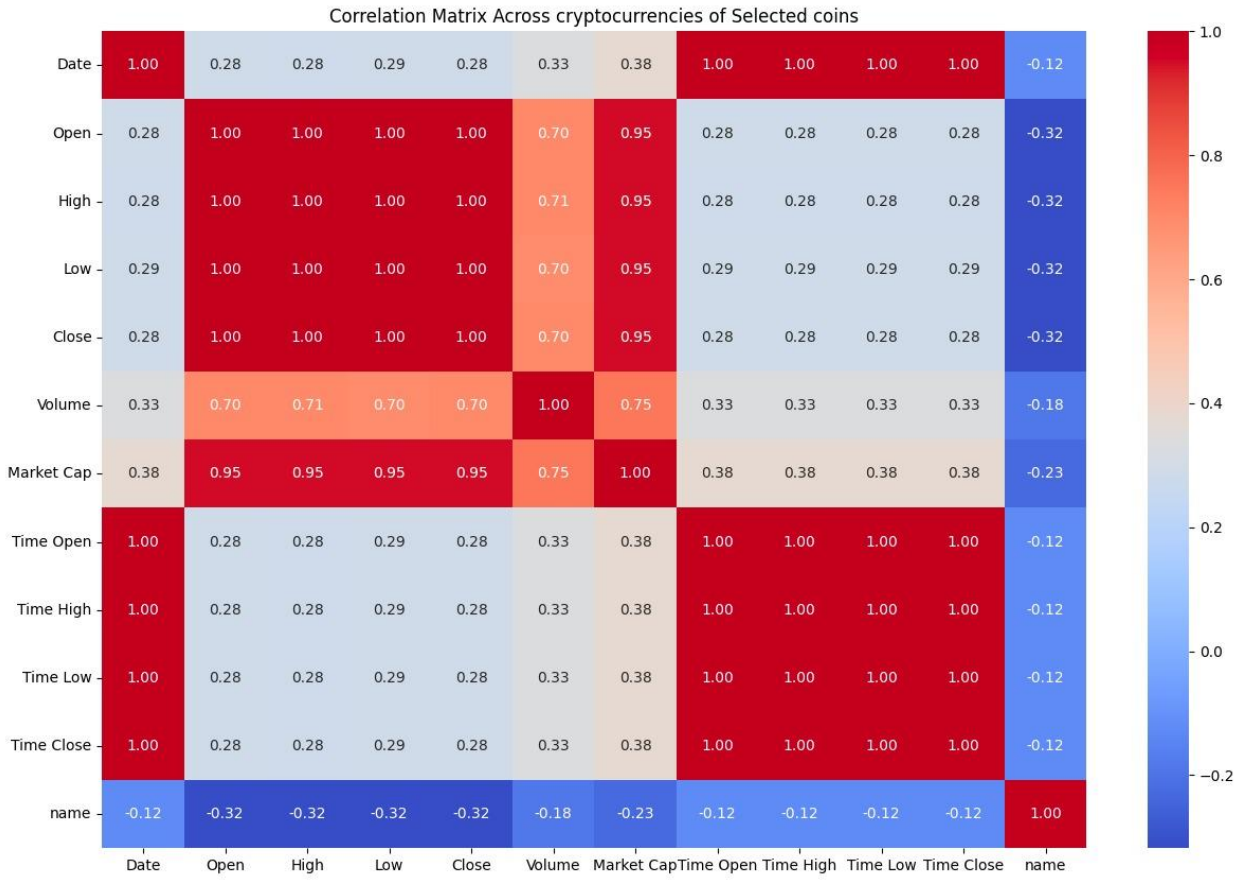
Correlation Matrix



Note. Correlation matrix for selected 5 coins.

Figure 38

Correlation Matrix of merged dataset

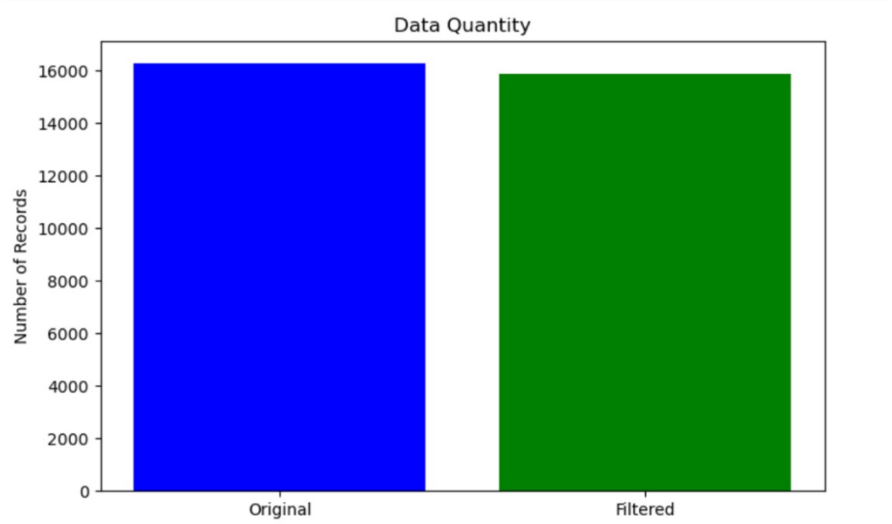


Note. Correlation matrix for selected 5 coins.

Furthermore, as mentioned in data preprocessing, we are considering the columns open, high, low, close, market cap columns for the model implementation. Figure 39 depicts the Quantity of data required to build the machine learning model.

Figure 39

Quantity of data



Note. original and filtered DataFrame to see the effect of outlier removal.

4. Model Development

4.1 Model Proposals

This project aims to build a supervised regression model that can predict the high price of the selected cryptocurrency coin. The selected coins are Bitcoin, Dogecoin, Aave, Ethereum and Ripple. The dataset is extracted using a scrapper, CmcScraper that scrapes the live data from Coin Market Cap. Each coin has nearly 4000 rows, each row represents the record of one day, the open price, close price, market cap, low, high, volume and the timing of each of these features. Figure 40 shows the columns for each dataset.

Figure 40

Columns for the selected five coins

```
df_btc.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap',
       'Time Open', 'Time High', 'Time Low', 'Time Close'],
      dtype='object')
```

```
df_eth.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap',
       'Time Open', 'Time High', 'Time Low', 'Time Close'],
      dtype='object')
```

```
df_aave.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap',
       'Time Open', 'Time High', 'Time Low', 'Time Close'],
      dtype='object')
```

```
df_doge.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap',
       'Time Open', 'Time High', 'Time Low', 'Time Close'],
      dtype='object')
```

```
df_xrp.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Market Cap',
       'Time Open', 'Time High', 'Time Low', 'Time Close'],
      dtype='object')
```

Note. The columns present in the data frames of the selected five coins

The dataset is collected from the inception of each of these coins, Figure 41 depicts the number

of rows and columns in each of the dataframe represented by the shape() attribute and Figure 42 depicts the dataframe of the merged dataset

Figure 41

Dataset from inception of the coins

```
df_btc.shape
```

```
(4025, 11)
```

```
df_eth.shape
```

```
(3194, 11)
```

```
df_aave.shape
```

```
(1311, 11)
```

```
df_doge.shape
```

```
(3794, 11)
```

```
df_xrp.shape
```

```
(3927, 11)
```

Note. The selected five coins dataset till present day

Figure 42

Dataset of merged coins

```
scraper_btc = CmcScraper(coin_code="btc", coin_name="bitcoin")
df_btc = scraper_btc.get_dataframe()
df_btc['name'] = 'btc'

scraper_eth = CmcScraper(coin_code="eth", coin_name="ethereum")
df_eth = scraper_eth.get_dataframe()
df_eth['name'] = 'eth'

scraper_aave = CmcScraper(coin_code="aave", coin_name="aave")
df_aave = scraper_aave.get_dataframe()
df_aave['name'] = 'aave'

scraper_doge = CmcScraper(coin_code="doge", coin_name="dogecoin")
df_doge = scraper_doge.get_dataframe()
df_doge['name'] = 'doge'

scraper_xrp = CmcScraper(coin_code="xrp", coin_name="xrp")
df_xrp = scraper_xrp.get_dataframe()
df_xrp['name'] = 'xrp'

merged_df = pd.concat([df_btc, df_eth, df_aave, df_doge, df_xrp], ignore_index=True)
```

```
merged_df.shape
```

```
(16291, 12)
```

Note. The merged dataset till present day

As per the problem statement, LSTM (Long Short-Term Memory) is chosen. This is because this

model is for time series or sequential data. These networks capture the long-term dependencies in sequential data. Linear regression is the most simple and effective model, and hence, chosen as a base model for this project. Another model highly favored for its predictive value of a high price is Random Forest. That is modelled on a method that builds a bunch of decision trees. XGBoost is an optimized distributed gradient boosting library that aims to be efficient, flexible, and portable. It constructs a sequence of trees such that each one corrects the errors of its previous tree. LSTM (Long Short-Term Memory) is a specific RNN architecture that models the sequential data not to fall under the vanishing gradient. It is a statistical practice-based natural language processing model that can capture long-term dependencies in time series data. Each of the models comes with some advantages and disadvantages. The weakness of this statistical technique is that the relationships are linear and, therefore, simple. On the other hand, Random Forest and XGBoost aim at capturing non-linear relationships and interactions between features but are prone to overfitting because too many trees are grown in the model. While LSTMs did not get the same popularity, they offer a powerful and well-explained solution to sequence data modeling, on the basis of much data and computation that has to train.

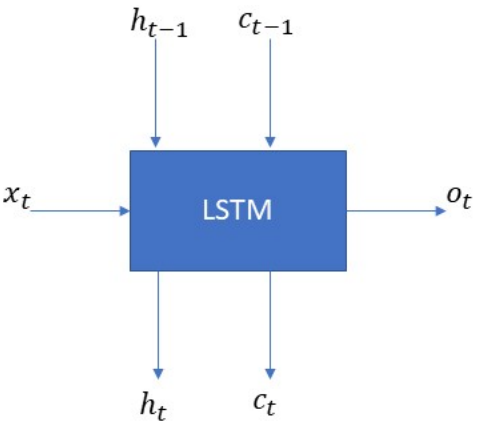
4.1.1 *Long Short-Term Memory*

LSTM stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) architecture intended to capture the long-term dependencies existing within the sequences of data. With LSTM models being sequence-aware, they would be put at work smoothly in any time series forecasting task, say predicting the price of a cryptocurrency. Features can include historical price data, open, close, low, volume, market cap. Simple recurrent networks, however, present some kind of complexity since their cells have memory units able to keep information over time. They may then make use of mechanisms of selective forgetting and remembering so

that, according to the input and past context, they can take the long-term dependencies in the sequential data. The LSTM networks are trained using a backpropagation through time (BPTT) algorithm in the sense that the recurrent gradients are computed and updated over many time steps. The networks can be trained with optimization algorithms like stochastic gradient descent (SGD) and adaptive optimization methods like Adam. We have used an optimizer known as "Adam" to optimize the model after hyperparameter tuning. The formula for LSTM is given in Figure 43.

Figure 43

Formula for LSTM



$$f_t = \sigma_g\,(W_f \times x_t + U_f \times h_{t-1} + b_f)$$
$$i_t = \sigma_g\,(W_i \times x_t + U_i \times h_{t-1} + b_i)$$
$$o_t = \sigma_g\,(W_o \times x_t + U_o \times h_{t-1} + b_o)$$
$$c'_t = \sigma_c\,(W_c \times x_t + U_c \times h_{t-1} + b_c)$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$$
$$h_t = o_t \cdot \sigma_c(c_t)$$

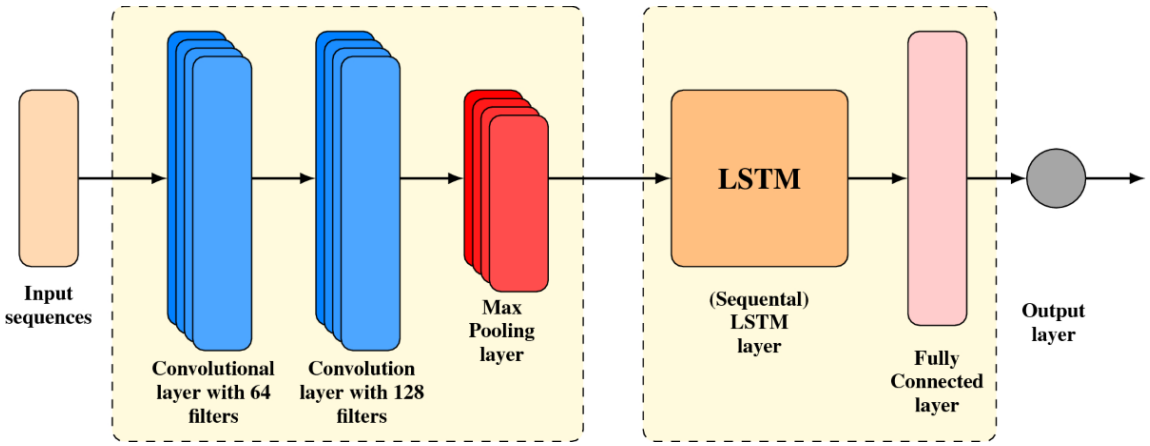
f_t *is the forget gate*
i_t *is the input gate*
o_t *is the output gate*
c_t *is the cell state*
h_t *is the hidden state*

Note. The architecture and operation for LSTM unit

Further, Figure 44 depicts the architecture of LSTM is implemented.

Figure 44

Architecture of LSTM



Note. Architecture and implementation

Linear regression is a simple and very commonly used statistical technique applied to model the linear relationship between a dependent variable and one independent variable. This form of modeling assumes the linear relationship of the variables. It is a method to model the relationship between the independent variable x and dependent variable y, forming a linear equation of the data to the input space; which can be represented as $y=\beta 0 +\beta 1\ x1 +\beta 2\ x2 +\ldots+\beta n\ xn$, where y is the target variable, $x1$,$x2$,…,xn are the independent variables, and $\beta 0$,$\beta 1$,…,βn are the coefficients. The linear regression algorithms calculate the coefficients that would minimize the error between the actual and predicted values.

The Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the average prediction of the individual trees. It's like linear regression, and its features are composed of historical price data. The "Random Forest" is a collection of decision trees. Each tree is built using a random subset of training data and features. The decision trees are then built using methods like bagging and feature randomness to improve the performance and generalization power of the model. The best descriptive feature is

determined by the algorithm through the calculation of the Information Gain (IG) from the features of the training dataset.

XGBoost is an open-source software library providing a gradient boosting framework. Like Random Forest, it is flexible enough to support high, low, open, close, and volume so that the features contain historical pricing and market cap. XGBoost sequentially builds a series of decision trees where each successive tree corrects the errors committed by its predecessors. Gradient boosting optimizes in the gradient descent to minimize a loss function, which helps in better predictive accuracy of the model. This uses the Gradient Boosting algorithm to minimize the loss function by greedily adding new trees to the ensemble. It also uses regularization techniques for the avoidance of overfitting.

4.2 Model Supports

The models were trained using scikit learn, we have used Mac OS, M2 16GB RAM resource to build the machine learning models. The data is fetched at every instance which is a flexible way to avoid any storage requirements. Given the nature of the machine learning models and the dataset we have chosen various tools like Jupiter notebook, VScode and widely used Python programming to understand, preprocess, implement and build the machine learning models. Given a wide range of Python libraries and various packages which facilitate the visualizations gives us an added advantage in the implementation of the project. Table 2 provides a list of libraries used for implementing the machine learning models.

Table 2

Libraries used for model development.

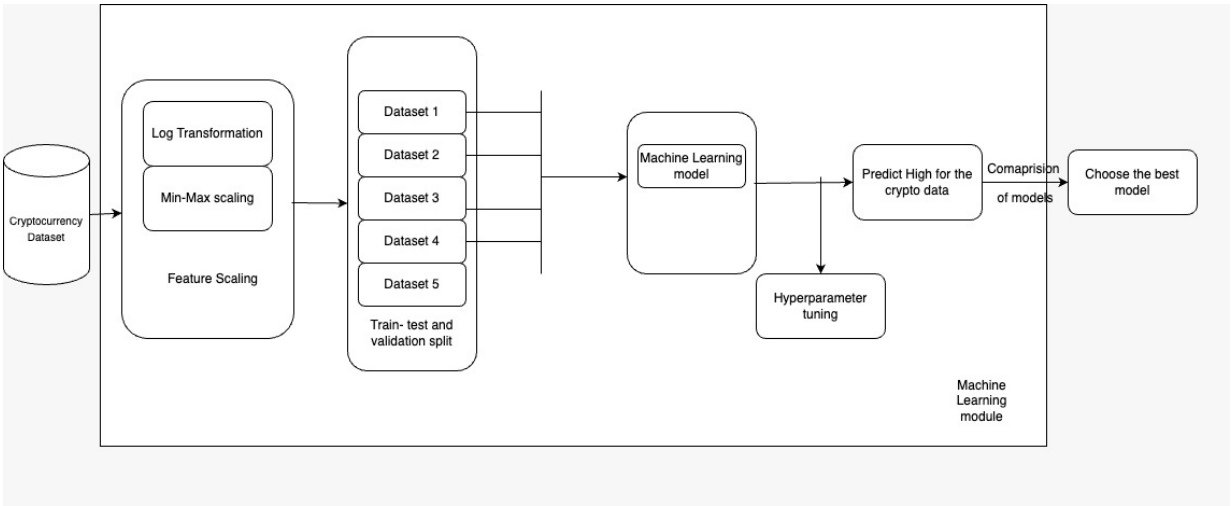
Lib	Module	Method	Usage
Scikit-learn	sklearn.preprocessing	*MinMaxScaler*	Normalizing the data by scaling features

Scikit-learn	sklearn.metrics	mean_squared_error	Calculate the mean squared error
Scikit-learn		r2_score	calculate the coefficient
Scikit-learn	sklearn.model_selection	*KFold*	Split the dataset into K consecutive
Scikit-learn		*Parameter Grid*	Conducts a grid search on hyper parameters
Scikit-learn	sklearn.pipeline	*Pipeline*	Sequentially applies a list of transforms and a final estimator
keras	keras.models	Sequential	Creates a linear stack of neural network layers for deep learning models
keras	keras.layers	*LSTM*	Implement LSTM
keras	keras.layers	Dense	Regular
cryptocmd	cryptocmd	*CmcScraper*	To scrape the dataset
Matplotlob	pyplot		Plotting various metrics
Numpy		expm1	Calculate exponent - 1
Numpy		mean	To calculate the mean
Numpy		sqrt	Calculate the square root
Numpy		log1p	For logarithmic transformation

Figure 45 depicts the system architecture of machine learning model and the models implemented. The architecture depicts the sequence of processing, and the different phases of a machine learning models to determine the trends of cryptocurrency.

Figure 45

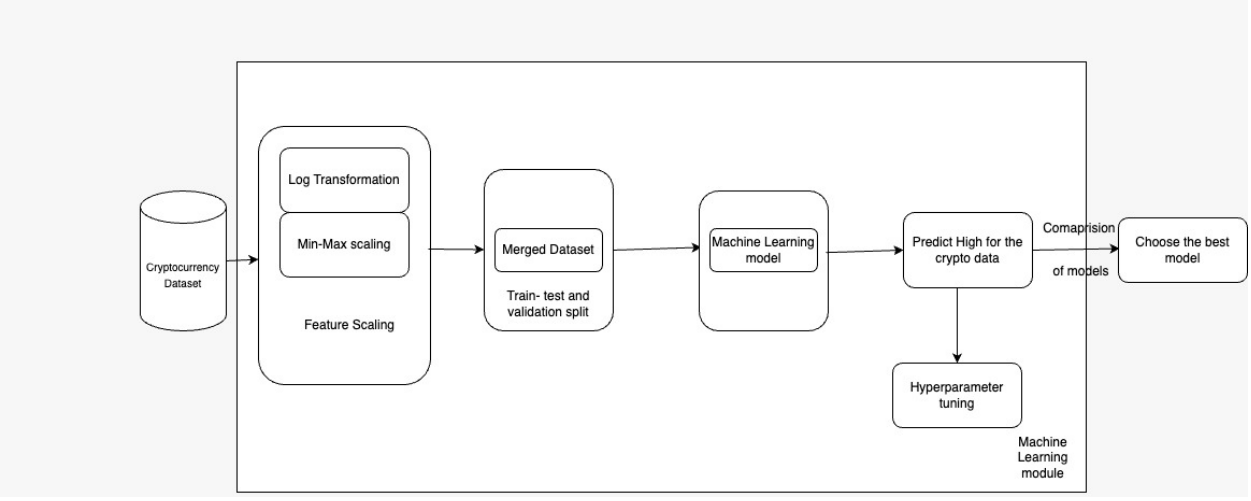
System Architecture and Model Support Diagram



Note. System Architecture of Machine learning module for selected coin datasets

Figure 46

System Architecture and Model Support Diagram



Note. System Architecture of Machine learning module for merged coin datasets

Further, the comprehensive process for training and evaluating a model is given in a data process model which is depicted in Figure 46. The K-fold validation is used to ensure the model generalizes well on unseen data. The results are further evaluated using hyper parameter tuning, to improve the performance.

4.3 Model Comparison and Justification

Table 3 lists the advantages and disadvantages of all the proposed models. These details show the differences and similarities between the models.

Table 3

Model Comparison

Models	Advantages	Disadvantages
Linear Regression	• Simple, interpretable, and fast to train • Works well for linear relationships between features and target variable	• Capturing complex non-linear relationships • Leads to underperformance when relationship is nonlinear. • Sensitive to outliers
Random Forest	• Can handle both numerical and categorical features. • Can capture complex nonlinear relationships in the data. • Robust to overfitting and handles large number of features. • Provides feature importance rankings	• Requires more computational resources and tuning. • May overfit noisy data if not properly tuned
XGBoost	• Can handle both numerical and categorical features. • Highly scalable, efficient and provides state-of-art performance on tabular/structured data and can capture complex patterns.	• Requires careful tuning of hyperparameters. • May not perform well on small datasets. • Less interpretable compared to linear models

	• Can handle missing values and feature interactions.	
LSTM	• Can handle sequential data and captures temporal dependencies, captures patterns over time. • Suitable for time series forecasting tasks. • Can handle variable-length sequences	• It is computationally intensive and suffers from vanishing or exploding gradients. • Prone to overfit on small datasets • Requires extensive hyperparameter tuning

The choice of model depends on factors such as nature of data, complexity of relationships, computational resources, and the need for interpretability. Linear Regression may be suitable for simple problems with linear relationships, while Random Forest and XGBoost are more versatile and can handle complex nonlinear data. LSTM is specifically designed for sequential data and is well-suited for time series forecasting tasks.

4.4 Model Evaluation Methods

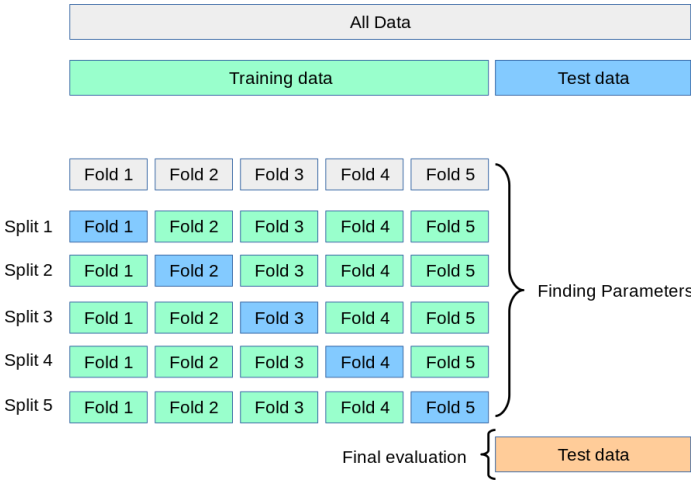
Detailed insights about the validation metrics are given below, we have used techniques like K-Fold Cross Validation, Mean Squared Error (MSE) and R-Squared(R2)

4.4.1 *K-Fold Cross-Validation*

The evaluation and optimization of the algorithm with 5 folds were investigated using the K-Fold cross-validation technique. K-fold cross-validation is an improved approach and just a replacement for the hold-out method in assessing a model's performance in its training. The technique divides the training dataset into k equal subsets, and then k separate evaluations are made. This way, every datum gets tested exactly once, belonging to the training set k-1 times. The data was split into the training-testing ratio of 80:20. Figure 47 represents the 5-fold cross validation.

Figure 47

K-Fold Cross-Validation



Note. K-Fold cross validation with 5 folds

4.4.2 Mean Squared Error (MSE)

Mean Squared Error measures the amount of error in statistical models, It calculates the average squared difference between the observed and predicted values. Figure 48 depicts the mathematical formula for MSE.

Figure 48

Mean Squared Error

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Mean Error Squared

Note. Mathematical formula for MSE

MSE is sensitive to outliers, as it squares the errors, giving higher weight to large errors.

Lower MSE values indicate better model performance, with zero representing a perfect fit. Error is the difference between actual and predicted values of target carriable for the ith data point.

4.4.3 R-Squared (R2)

R2 is, in essence, just a measure of the proportion of the variance in the dependent variable that can be accounted for or predicted by the independent variable. It is the ratio of the explained variance to the total variance in the dependent variable. R2 values are between 0 and 1. A value of 1 indicates a perfect fit, while a value of 0 indicates that the model does not fit the variabilities of the dependent variable any better than a horizontal line would. It indicates that the model fits much worse than a horizontal line, and negative values of R^2 shouldn't exist. This is expressed in mathematical form in Figure 49.

4.4.4 RMSE

The Root Mean Squared Error (RMSE) is one of the two main performance indicators for a regression model. It measures the average difference between values predicted by a model and the actual values. It provides an estimation of how well the model is able to predict the target value.

Figure 49

R-squared

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n} (Y_i - \overline{Y}_i)^2}$$

Note. Mathematical formula for R-squared

Here the Y bar is the mean of the actual values Y*i.* Y cap is the predicted value of target variable and Yi is the actual value for ith data point. These metrics provide insight into how well the

model fits the data and how much variance it explains. Lower MSE and higher R2 values generally indicate better model performance.

4.5 Model Validation and Evaluation

4.5.1 *Model evaluation*

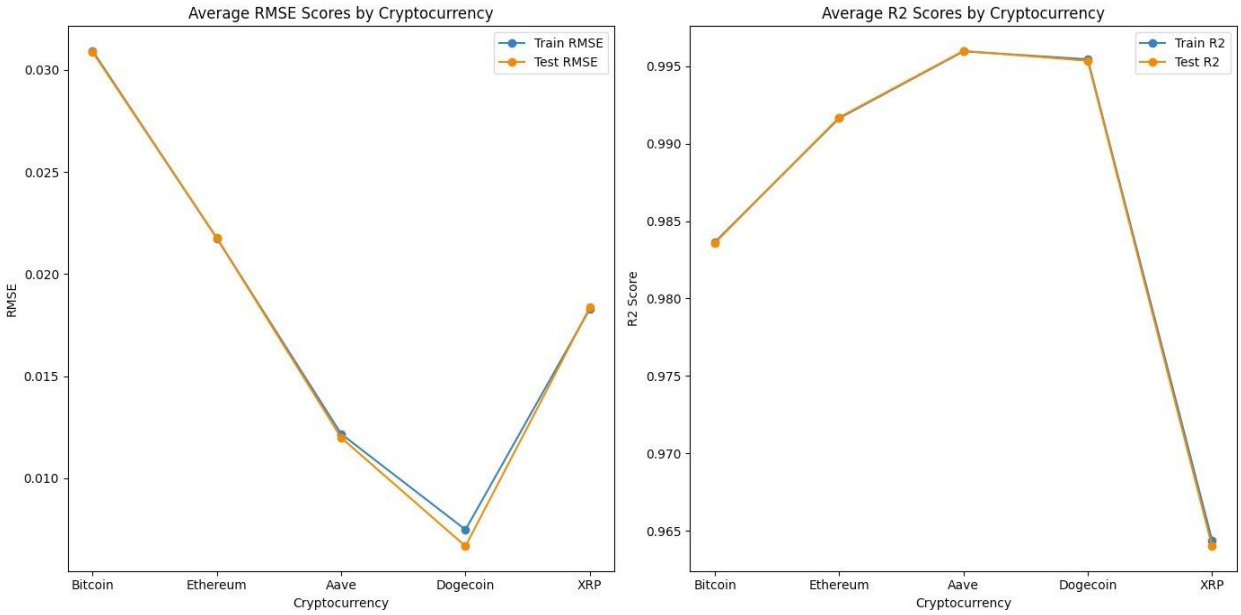
The model is validated and evaluated and is elaborated in the following section.

Long Short-Term Memory Model (LSTM)

The baseline model for LSTM is used in our case by converting the time series data into supervised machine learning data. Our cryptocurrency coin datasets which are log transformed and scaled using min-max scaler are used to train the model with default parameters. The transformed datasets are split into train, test and validation sets using K-fold cross validation with (n_splits = 5) and the random_state = 42 (default). The LSTM model is built using a Keras Sequential API consisting of an LSTM layer with 50 units and then a Dense Layer with 1 unit. The model is compiled for MSE and Adam Optimizer. Predictions are made on both train and test datasets. RMSE and R2 are calculated for both of our train and test datasets. The average RMSE and R2 scores are calculated for all the folds which is depicted in figure 50.

Figure 50

Average RSME and R2 scores



Note. Average RSME and R2 scores

RMSE

The RMSE scores for Dogecoin shows the least, which means it performs very well. The RMSE for Bitcoin and Ethereum shows higher RMSE values for both train and test datasets. Overall the model uses transformed dataset, to ensure the error is less. The RSME train and test scores are enlisted in Table 4

Table 4

RSME train and test scores

Cryptocurrency	RMSE - train	RMSE- test
Bitcoin	0.030936	0.030876

Ethereum	0.021752	0.021765
Aave	0.012173	0.011988
Xrp	0.007484	0.006675
Dogecoin	0.018291	0.018369

R2 score

This shows that Aave and Dogecoin gave the highest R2 score, which means the model was able to predict the high price of these coins accurately. Ripple shows a drop in R2 score for the test dataset which is shown in Table 5.

Table 5

R2 train and test scores

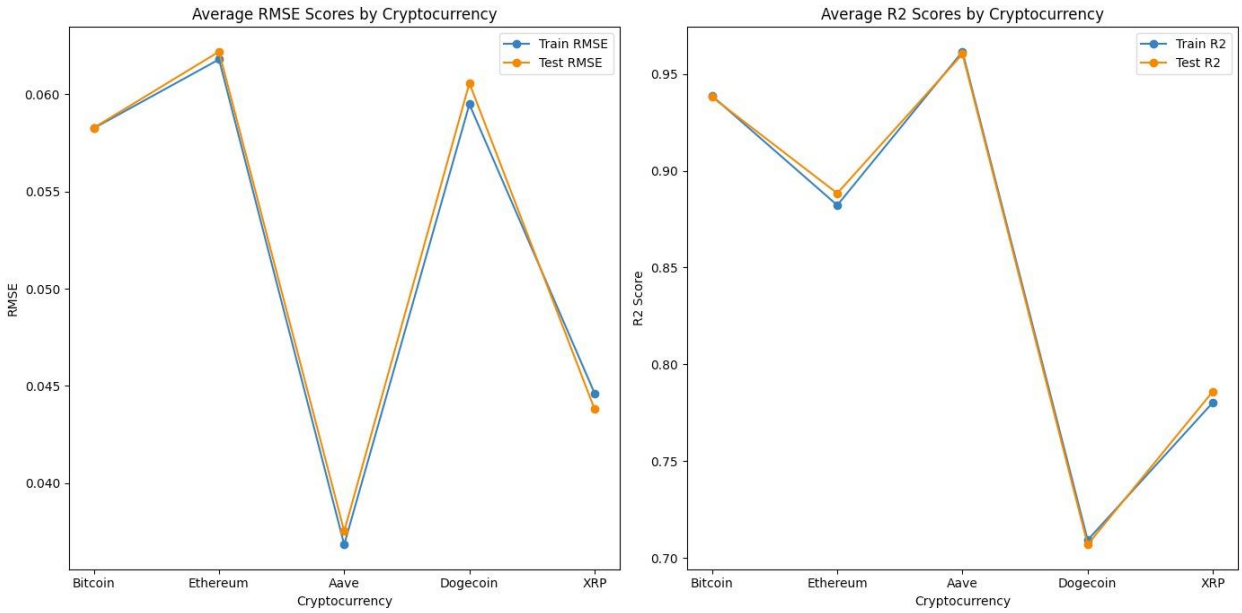
Cryptocurrency	**R2 - train**	**R2- test**
Bitcoin	0.983654	0.983597
Ethereum	0.991649	0.991856
Aave	0.995967	0.995997
Xrp	0.995458	0.995362
Dogecoin	0.964388	0.963999

Hyper Parameter Tuning

The average RSME and R2 scores are depicted in Figure 51

Figure 51

Average RSME and R2 scores



Note. Average RSME and R2 scores

RMSE

The RSME train and test scores are given in Table 6

Table 6

RSME train and test scores

Cryptocurrency	RMSE - train	RMSE- test
Bitcoin	0.064075	0.064261
Ethereum	0.061470	0.062776

Aave	0.025813	0.026399
Xrp	0.059458	0.057808
Dogecoin	0.028818	0.029213

R2 score

The R2 scores are enlisted in Table 7

Table 7

R2 train and test scores

Cryptocurrency	R2 - train	R2- test
Bitcoin	0.929466	0.928748
Ethereum	0.875983	0.869128
Aave	0.980995	0.979915
Xrp	0.707825	0.720212
Dogecoin	0.904885	0.899838

Long Short-Term Memory Model (LSTM) for Merged dataset

The baseline model for LSTM is used in our case by converting the time series data into supervised machine learning data. Our cryptocurrency coin datasets which are log transformed and scaled using min-max scaler are used to train the model with default parameters. The transformed dataset are split into train, test and validation sets. The LSTM model is built using a Keras Sequential API consisting of an LSTM layer with 50 units and then a dropout rate of 0.2. The model is compiled for MSE and R-squared using Adam Optimizer. Predictions are

made on both train and test datasets. RMSE and R2 are calculated for both of our train and test datasets. The RMSE and R2 scores are calculated which is depicted in figure 52.

Figure 52

RSME and R2 scores

```
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Define the LSTM model
def create_lstm_model(units=50, dropout_rate=0.2, optimizer='adam', num_layers=1, look_back=10):
    model = Sequential()
    model.add(LSTM(units=units, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])))
    model.add(Dropout(dropout_rate))
    for _ in range(num_layers - 1):
        model.add(LSTM(units=units, return_sequences=True))
        model.add(Dropout(dropout_rate))
    model.add(LSTM(units=units))
    model.add(Dropout(dropout_rate))
    model.add(Dense(units=1))
    model.compile(optimizer=optimizer, loss='mean_squared_error')
    return model

# Create and train the LSTM model
model = create_lstm_model()
model.fit(X_train, Y_train_reshaped, epochs=25, batch_size=32, verbose=0, validation_data=(X_val, Y_val_reshaped))

# Evaluate the model on the test set
Y_pred = model.predict(X_test)
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print("MSE:", mse)
print("R-squared:", r2)
```

```
/Users/shreyac/anaconda3/lib/python3.11/site-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
MSE: 0.00015770938296721007
R-squared: 0.9985908802696906
```

Note. RSME and R2 scores

RMSE

The RMSE scores for Dogecoin shows the least, which means it performs very well. The MSE for Bitcoin and Ethereum shows higher RMSE values for both train and test datasets. Overall the model uses transformed dataset, to ensure the error is less. The MSE train and test scores are enlisted in Figure 53

Figure 53

MSE and R2 train and test scores

```
Training Data Evaluation Metrics:
aave - MSE: 5.635926039506119e-05, R-squared: 0.9848315180313939
btc - MSE: 4.104554809050524e-05, R-squared: 0.9984342712570934
doge - MSE: 5.1470987497793845e-06, R-squared: 0.8788054123200105
eth - MSE: 1.3675521128846699e-05, R-squared: 0.9996152392721037
xrp - MSE: 1.558244199055515e-05, R-squared: 0.9667786211141963

Testing Data Evaluation Metrics:
aave - MSE: 5.667538941977646e-05, R-squared: 0.982718626029958
btc - MSE: 4.02025553922127e-05, R-squared: 0.9984571608197897
doge - MSE: 5.338673370787412e-06, R-squared: 0.8427332284540333
eth - MSE: 1.669147505371982e-05, R-squared: 0.9995352952556935
xrp - MSE: 1.73959828265408e-05, R-squared: 0.9648795471937436

Overall Training MSE: 2.2201791579618097e-05
Overall Training R-squared: 0.9998039017814306
Overall Testing MSE: 2.3329965536391755e-05
Overall Testing R-squared: 0.9997915487707437
```

This shows that Bitcoin and Ethereum gave the highest R2 score, which means the model was able to predict the high price of these coins accurately. Dogecoin shows a drop in R2 score for the test dataset which is shown.

Hyper Parameter Tuning

The parameters taken in the grid for units is 50,100, dropout_rate is 0.2, 0.5, for num_layers I considered 1,2 and for look_back is it 10,20. The optimizer is read through the given 4 which are adam, rmsprop, sgd, adagrad.

Figure 54

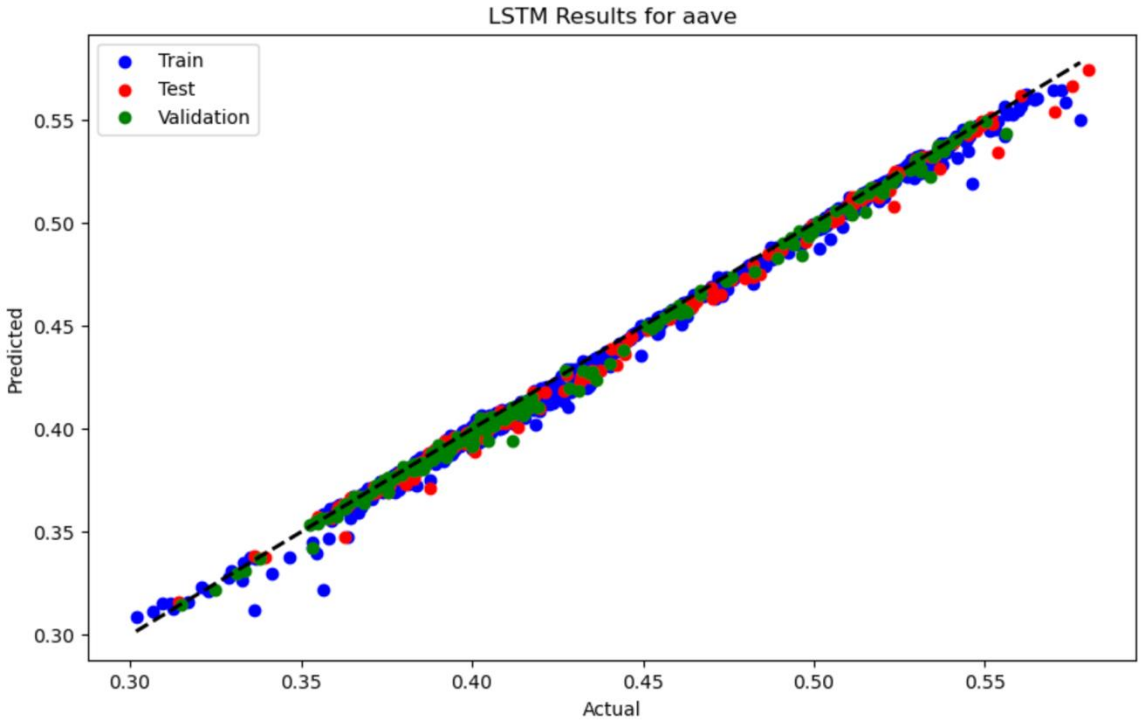
Best parameters and train-test evaluation scores

```
Evaluation metrics for aave using LSTM:
Best MSE (Validation): 1.7321139666478085e-05
Best R-squared (Validation): 0.9953940764453855
Best parameters: Not applicable as no hyperparameter tuning was done here
Train MSE: 2.232252976469912e-05
Train R-squared: 0.9939143307325125
Test MSE: 2.2663314422059325e-05
Test R-squared: 0.9941649987391284
Evaluation metrics for bitcoin using LSTM:
Best MSE (Validation): 5.149638500719514e-05
Best R-squared (Validation): 0.997984686172783
Best parameters: Not applicable as no hyperparameter tuning was done here
Train MSE: 4.6354377757951897e-05
Train R-squared: 0.9982327684523398
Test MSE: 4.724406992762998e-05
Test R-squared: 0.9982512025011171
Evaluation metrics for dogecoin using LSTM:
Best MSE (Validation): 1.1503476182049017e-06
Best R-squared (Validation): 0.9756741423463704
Best parameters: Not applicable as no hyperparameter tuning was done here
Train MSE: 9.04146233433831e-07
Train R-squared: 0.9775546200129561
Test MSE: 1.4967576038719044e-06
Test R-squared: 0.964902131668849
Evaluation metrics for ethereum using LSTM:
Best MSE (Validation): 0.00010554292323611225
Best R-squared (Validation): 0.9973024315403455
Best parameters: Not applicable as no hyperparameter tuning was done here
Train MSE: 0.00010701655434790148
Train R-squared: 0.996926407747002
Test MSE: 0.00010076923713975378
Test R-squared: 0.9971486968571015
Evaluation metrics for ripple using LSTM:
Best MSE (Validation): 2.131214018778828e-06
Best R-squared (Validation): 0.9957009781422247
Best parameters: Not applicable as no hyperparameter tuning was done here
Train MSE: 1.9771857406327364e-06
Train R-squared: 0.9957039488958703
Test MSE: 2.009567729051613e-06
Test R-squared: 0.9955994898509799
```

Note: Shows the best parameters and train-test mse and r2 scores

Figure 55

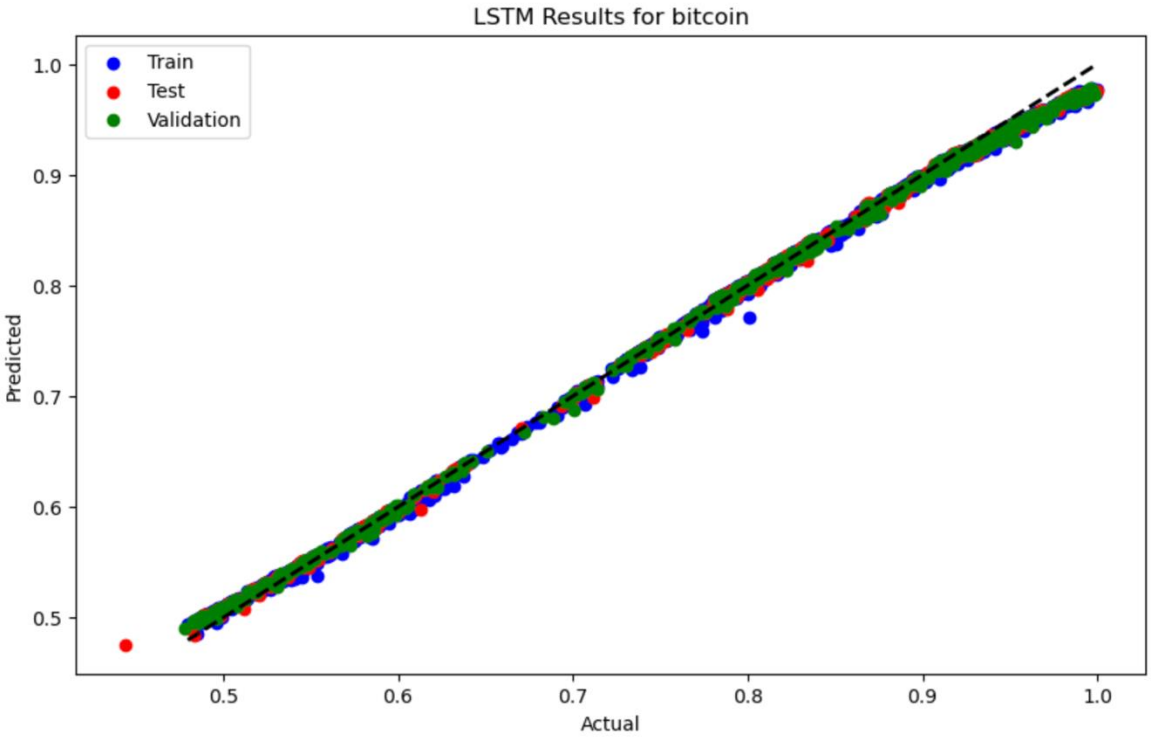
The actual and predicted values of aave coin



Note: actual and predicted values of aave coin

Figure 56

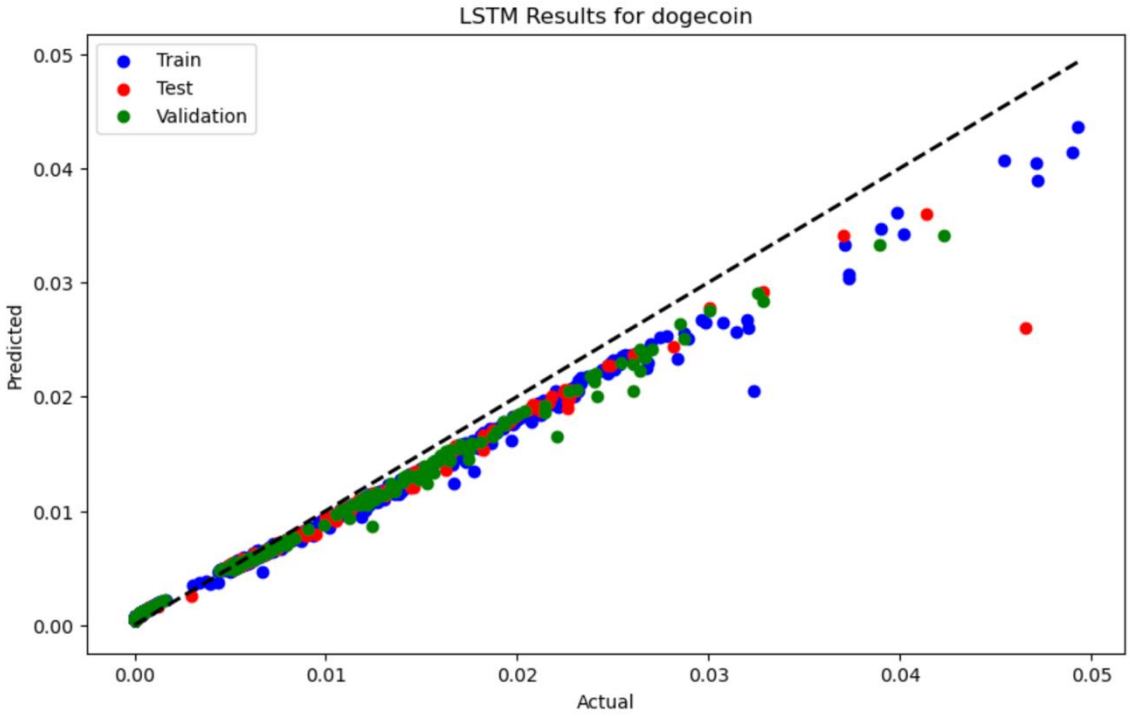
The actual and predicted values of bitcoin



Note: actual and predicted values of bitcoin

Figure 57

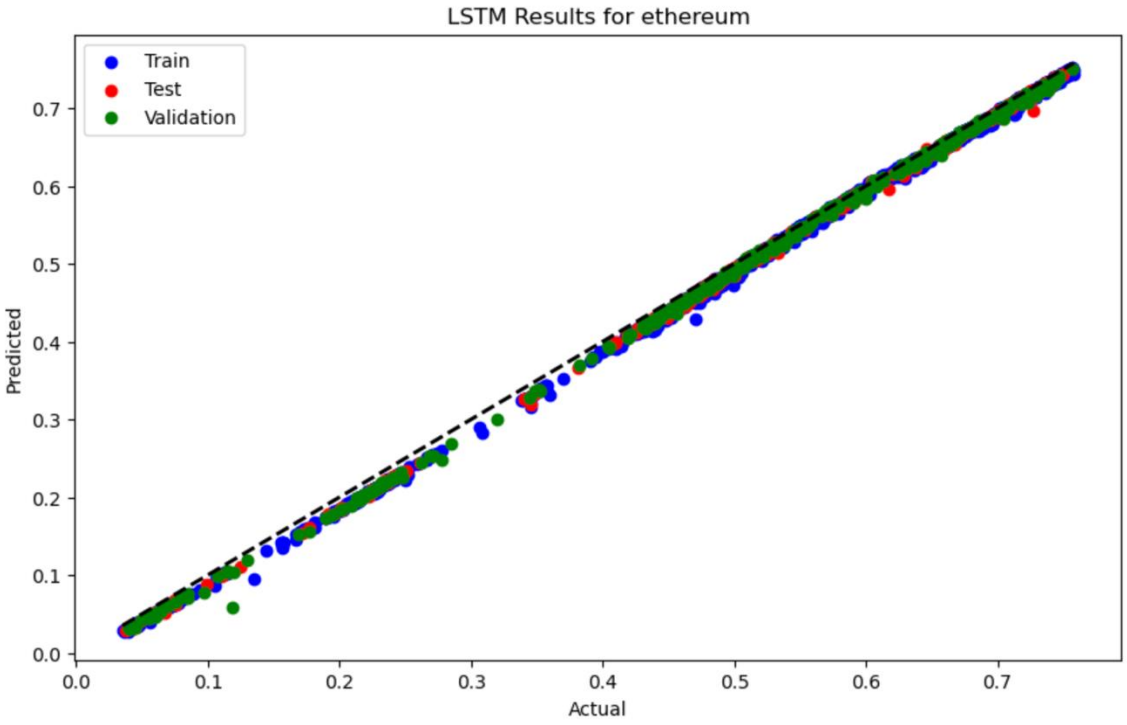
The actual and predicted values of doge coin



Note: actual and predicted values of doge coin

Figure 58

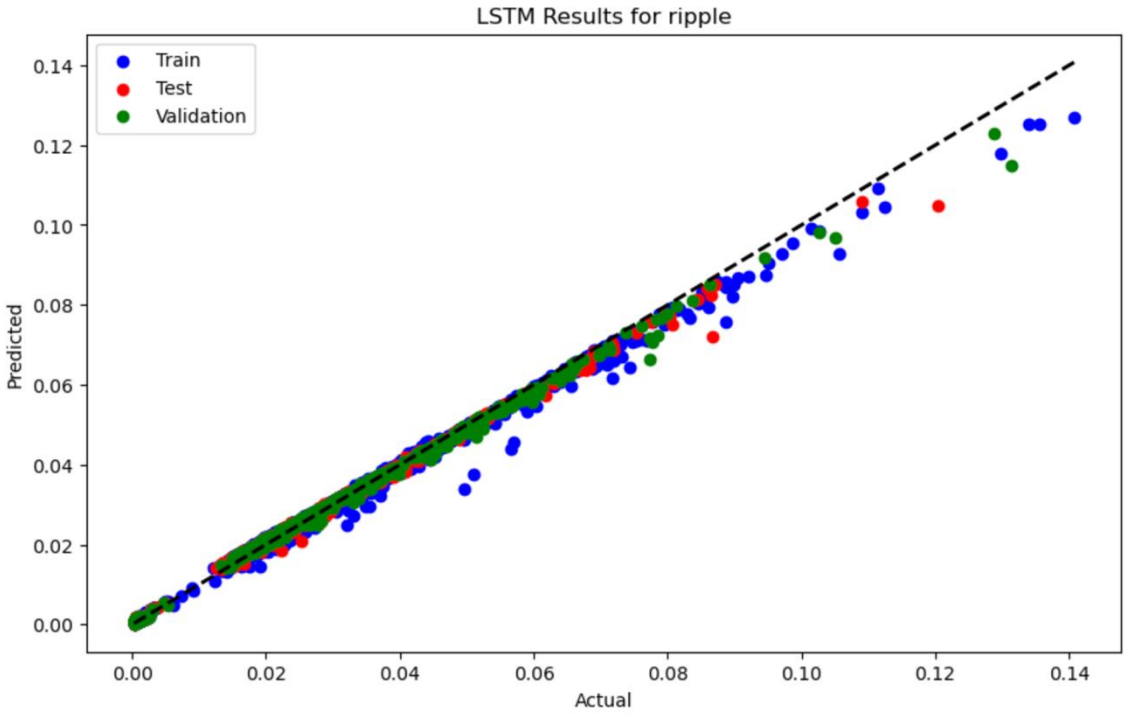
The actual and predicted values of ethereum coin



Note: actual and predicted values of ethereum coin

Figure 59

The actual and predicted values of ripple coin



Note: actual and predicted values of aave coin

Figures 55, 56, 57, 58, 59 show the actual and predicted values of Aave, Bitcoin, Dogecoin, Ethereum and Ripple respectively.

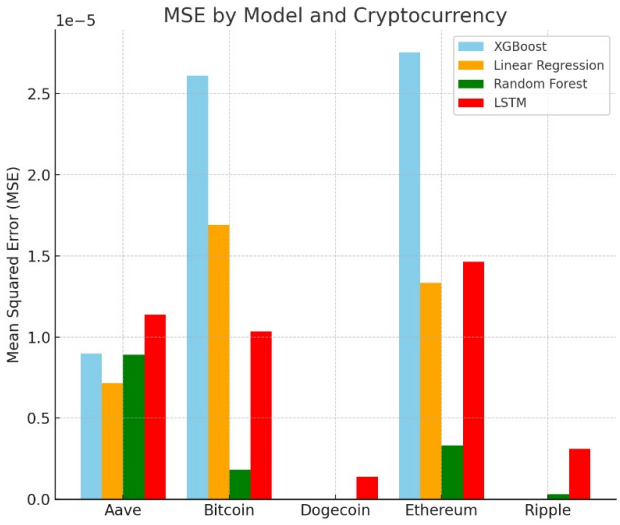
4.5.2 Model performance

Figure 60 shows the bar plot shows the test Root Mean Square Error (RMSE) scores of our various cryptocurrencies datasets across different predictive models which we have used: Random Forest, Linear, XGBoost, and LSTM.

RMSE

Figure 60

Test MSE scores by Model and Cryptocurrency

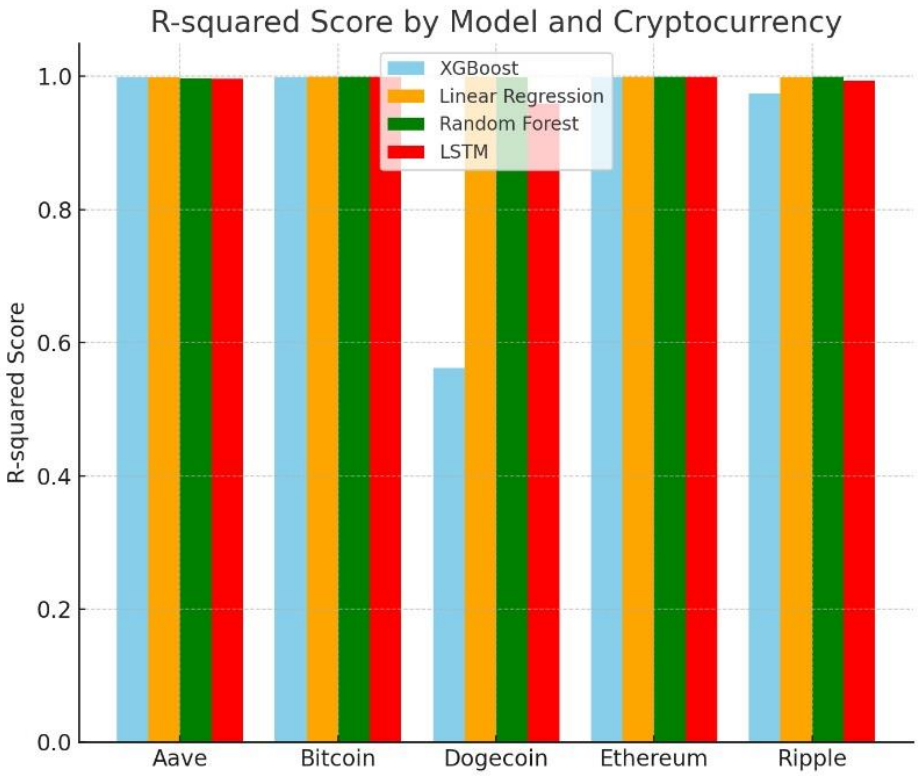


Note. Test MSE scores

Overall, we can see that Dogecoin and Ripple show extremely low MSE scores, meaning good accuracy. Random Forest shows moderate performance for all models.

Figure 61

R2 Test values for different models



Note. Values of different models

All models generally achieve very high R-squared values, often close to 1, indicating a strong fit to the data. Notably, the Random Forest model achieves exceptionally high R-squared values for Bitcoin and Ethereum, highlighting its effectiveness for these cryptocurrencies.

We see from the above that Random Forest gives the best prediction.

R-squared and MSE Chart Analysis

1. R-squared Score Analysis

The chart of the R-squared score is a clear illustration of how the models are good at predicting the data in relation to the real data. The value of R-squared relates to the amount of variance the model tries to explain. On your chart, in most cases, LSTM is performing just as well as, or

slightly better than, the other models for cryptocurrencies such as Bitcoin and Ethereum. This means it is capable of capturing temporal patterns in these time series data. But for coins like Dogecoin, the LSTM model does not perform very well, indicating that its performance varies with the characteristics of the data.

2. Mean Squared Error (MSE) Analysis

MSE measures the average squared difference between the estimated values and the actual value. Thus, the smaller the value of MSE, the better. Thus, the MSE chart would suggest that in these datasets, LSTM supports effective handling by having lower error rates for Bitcoin and Ethereum than most other models. For Aave and Ripple, LSTM also competes well, which is an indication of the robustness among scales of data. It was noted that the important error decrease in complex and highly volatile data, such as Bitcoin, emphasizes the advantage of LSTM in capturing non-linear dependencies and trends over time.

Why LSTM Might Do Better

Temporal Dynamics: Indeed, LSTM models are designed to work on sequence prediction problems. They can learn order dependence in problems of sequence prediction which is of prime importance because the financial time series data has past information being a strong indicator of future trends.

Prevents Overfitting: With the presence of proper regularization, for example, dropout in LSTM networks, such models are less prone to overfitting in cases where high volatility or irregularity in the data is likely to be prevalent.

Long-Term Dependencies: LSTM's ability to hold long-term dependencies is crucial for cryptocurrency prediction, as the markets can be influenced by events that occurred quite far back in time.

Contextual Suitability

Data Nature: Cryptocurrency, by its nature, is rather volatile and non-linear, something which is better modeled by LSTM due to its recurrent nature, as opposed to models like linear regression that assume linearity.

Feature Interactions: As opposed to classical models which consider all input features as independent, by its very nature LSTM has the capability of learning the interaction between different features over time. In financial data sets, the interaction over time between past prices and volumes is an important feature.

Conclusion

While LSTM might be hopeful with a subset of cryptocurrencies and evaluation metrics, the important issues are general context and specific needs. In general, LSTM models are much more computationally expensive and complex to train and tune than models such as linear regression, or even tree-based models, like Random Forest and XGBoost. Therefore, the use of LSTM should be justified by performance gains with respect to simpler or equally sophisticated models, considering both the precision requirements and the computational resources.

References

Pintelas E, Livieris I.E, Stavroyiannis S, Kotsilieris T, Pintelas

"Fundamental research questions and proposals on predicting cryptocurrency prices using DNNs.", *Technical report, TR20-01*, University of Patras, vol. 3, no. 5, pp.97– 106, 2020.

Bonneau, J., Miller, A., Clark, J. Narayanan, A.

"Cryptocurrencies: A Brief Review".*IEEE Security and Privacy*, 13, vol. 4, no.2, pp.18-28,2015. Doi: 10.1109/MSP.2015.61.

A.ElBahrawy, L. Alessandretti, A. Kandler, R. Pastor-Satorras, and A. Baronchelli

"Evolutionary dynamics of the cryptocurrency market," *Royal Society Open Science*, vol. 4, no. 11, November, 170623, 9 pages, 2017.

Alessandretti, L., ElBahrawy, A., Aiello, L. M., & Baronchelli, A. (2018).

Anticipating Cryptocurrency Prices Using Machine Learning. Complexity, 2018, Article ID 8983590, 16 pages. https://doi.org/10.1155/2018/8983590

Iqbal, M., Iqbal, M. S., Jaskani, F. H., Iqbal, K., & Hassan, A. (2021).

Time-Series Prediction of Cryptocurrency Market using Machine Learning Techniques. EAI Endorsed Transactions on Creative Technologies, 8(28)**.** https://doi.org/10.4108/eai.7-7-2021.170286

Edwin Sin and Lipo Wang.

"Bitcoin price prediction using ensembles of neural networks". In: 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNCFSKD) (July 2017).

E. Sin and L. Wang

"Bitcoin price prediction using ensembles of neural networks," 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Guilin, China, 2017, pp. 666-671, doi: 10.1109/FSKD.2017.8393351.

Akyildirim, Erdinc & Göncü, Ahmet & Sensoy, Ahmet. (2018).

Prediction of Cryptocurrency Returns using Machine Learning.

Shintate, Takuya, and Lukáš Pichl. 2019.

"Trend Prediction Classification for High Frequency Bitcoin Time Series with Deep Learning" Journal of Risk and Financial Management 12, no. 1: 17. *https://doi.org/10.3390/jrfm12010017*

Ji, Suhwan, Jongmin Kim, and Hyeonseung Im. 2019.

"A Comparative Study of Bitcoin Price Prediction Using Deep Learning" Mathematics 7, no. 10: 898. *https://doi.org/10.3390/math7100898*

P. Jay, V. Kalariya, P. Parmar, S. Tanwar, N. Kumar and M. Alazab,

"Stochastic Neural Networks for Cryptocurrency Price Prediction," in IEEE Access, vol. 8, pp. 82804-82818, 2020, doi: 10.1109/ACCESS.2020.2990659.

Investopedia. (n.d.). *Efficient Market Hypothesis (EMH).* Retrieved from

https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp

Investopedia. (n.d.). *Alpha. Retrieved* from

https://www.investopedia.com/terms/a/alpha.asp

Awan, S. E., & Sohail, M. K. (2020).

Cryptocurrency price prediction and trading strategies using support vector machines. Annals of Operations Research. https://doi.org/10.1007/s10479-020-03575-y

Kwon, D.-H., Kim, J.-B., Heo, J.-S., Kim, C.-M., & Han, Y.-H. (2019).

Time Series Classification of Cryptocurrency Price Trend Based on a Recurrent LSTM Neural Network. Journal of Information Processing Systems, 15(3), 694–706. https://doi.org/10.3745/JIPS.03.0120

Shahbazi, Z., & Byun, Y.-C. (2022). *Knowledge Discovery on Cryptocurrency Exchange Rate Prediction Using Machine Learning Pipelines*. Sensors, 22(5), 1740. https://doi.org/10.3390/s22051740

Roosenboom, P., van der Kolk, T., & de Jong, A. (2020). *What determines success in initial coin offerings? *Venture Capital**, 22(2), 161-183. https://doi.org/10.1080/13691066.2020.1741127

DigitalCoinPrice. (n.d.). Homepage. Retrieved (2021), from https://digitalcoinprice.com/

gudur, N. (n.d.). data-270-project [Software]. GitHub. Retrieved from https://github.com/nikhilgudur/data-270-project

Bithumb. (n.d.). N.A https://www.bithumb.com/u1/US127