CincoCrypto - A Cryptocurrency Price Forecasting Tool For Everyone

Project Report

Gonzalo Lara de Leyva

Aalborg University
Electronics and IT

Electronics and IT
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
CincoCrypto - A Cryptocurrency Price Forecasting Tool For Everyone

Theme:
Cryptocurrency Price Prediction

Project Period:
Spring Semester 2023

Project Group:
4.01

Participant(s):
Author 1
Gonzalo Lara de Leyva

Supervisor(s):
Supervisor 1
Ashutosh Dhar Dwivedi

Copies: 1

Page Numbers: 63

Date of Completion:
June 1, 2023

Abstract:

In the last decade, cryptocurrencies have become ever more popular but so have cryptocurrency investment scams. In order for investors to make safe, well-informed investments in the cryptocurrency market, this project proposes a price forecasting tool, named *CincoCrypto*, which is developed to be a simple, free, open-source alternative targeted to all audiences. Four Machine Learning models are trained in the tool, using financial data from 1-year datasets to make 1-day ahead predictions in United States Dollars. The results show that the Space Vector Regression model is the fastest whereas the Convolutional Neural Network model is the most accurate of the four.

This Master Thesis is dedicated to my mother, who has always supported me throughout the best and worst times of my life, and to my supervisor for having encouraged me since the beginning of this project. Thank you AAU for having accepted me into the MSc Cybersecurity and preparing me for my professional career.

Contents

List of Figures

List of Tables

List of CincoCrypto Listings

Abbreviations

R^2	Coefficient of Determination
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Autoregressive Integrated Moving Average
BNB	Binance (Binance network cryptocurrency)
BTC	Bitcoin (Bitcoin network cryptocurrency)
CNN	Convolutional Neural Network
CWD	Current Working Directory
DL	Deep Learning
ETH	Ether (Ethereum network cryptocurrency)
GRU	Gated Recurrent Unit
GUI	Graphical User Interface
LSTM	Long Short-Term Memory
LTC	Litecoin (Litecoin network cryptocurrency)
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
NaN	Not a Number
OHLC	Open High Low Close
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SARIMA	Seasonal ARIMA
SSR	Sum of Squares due to Regression
SST	Sum of Squares Total
SVM	Space Vector Machine
SVR	Space Vector Regression
tanh	Hyperbolic tangent function
USD	United States Dollar
USDC	USDC (USD Coin network cryptocurrency)
USDT	Tether (Tether network cryptocurrency)
ZEC	Zcash (Zcash network cryptocurrency)

Preface

This project has been undertaken to acquire a solid foundation of Machine Learning and cryptocurrency price prediction.

Aalborg University, June 1, 2023

Author
Gonzalo Lara de Leyva
glarad21@student.aau.dk

Chapter 1

Introduction

Cryptocurrencies are becoming more entangled with society as each year goes by in this ever, increasingly digitised society. However, given its use as a value exchange, it means cryptocurrency-related scams are alarmingly increasing as well. According to the FBI's most recent Internet Crime Report, investment opportunities represented the top cybercrime by victim loss worldwide (i.e. 3.31 billion USD) in 2022 for which 77% (2.57 billion USD) was due to cryptocurrency investment fraud. Compared to 2021 when this type of cyberscam caused 907 million USD in victim losses, this represents an unprecedented increase of 183% from one year to the next. As a result, numerous victims (all in the ages 30 and 49) have required to cover their losses by becoming indebted with massive amounts [**1**]. Both the FBI and Europol [**2**] agree on cryptocurrency investment scams being sutile and distinct and, thus, can be classified in different categories. Some of the categories worth mentioning are:

- Liquidity Mining: Victims are told by scammers to connect their cryptocurrency wallet with a fake liquidity mining software. This enables scammers to retrieve the victims' funds without being notified or consented to do so.
- Business Opportunity: Scammers get in touch with victims and show them fraudulent cryptocurrency websites, convincing the victims to invest. Victims are left powerless to withdrawn their funds once they have invested in these websites.
- Phishing: Similar to Business Opportunity, scammers advertise cryptocurrency investment on social media platforms. Victims who are enticed by these adverts will then provide their contact details, allowing scammers to manipulate them into investing.

Cryptocurrency-related scams have generated enough attention to become a subject of study as in [**3**] and [**4**] as well as requiring long-standing financial institutions such as Santander Bank [**5**] and HSBK UK [**6**] to raise awareness of this issue.

Therefore with these issues at hand, this project presents a cryptocurrency price forecasting tool (referred throughout the project as *CincoCrypto*) which analyses OHLC and trading information from the current (as of the 6^{th} May 2023) top five cryptocurrencies (or coins) to develop model in which their closing USD price can be forecasted in a transparent, user-friendly manner to those who have little to no trading or technical background.

1.1 Motivation

To ensure investors can make well-informed, honest, trading in the cryptocurrency market and mitigate fraudulent investment to the extent possible, this project aims to develop a simple, free, open-source tool for cryptocurrency closing price forecasting using the latest available financial information at the time of writing this Thesis. To this end, the tool has been published on GitHub for it to be available to all audiences[**7**]. *CincoCrypto* is trained and tested on using 1-year (06/05/2022 - 06/05/2023) OHLC and trading volume information from the top five cryptocurrencies at the time of writing (i.e. BTC, ETH, USDT, BNB and USDC) as stated by the world's leading sources of cryptocurrency price information: Binance [**8**], CoinMarketCap [**9**] and Yahoo! Finance [**10**]. To make predictions on the *Close* price as accurate as possible, four models are generated within *CincoCrypto* and their performance is reviewed to determine which one is best for a given coin dataset being analysed.

1.2 Scope

The breadth and limitations of the project provide a brief description of the relevance of the results for the proposed tool. The areas deliberately taken into account during the development of *CincoCrypto* are the following:

- Cryptocurrency daily financial data
- 1-year long datasets
- Regression and Deep Learning algorithms
- Analysis of the current top 5 cryptocurrencies
- Model performance evaluation

The limitations in development of *CincoCrypto* are the following:

- Cryptocurrency sentiment and network information not considered
- Analysis of more than five crypto datasets is not enabled
- Datasets from CryptoCompare can only be analysed
- Algorithm hyperparameters are left as default and not assessed
- Feature selection algorithms are not implemented

Therefore the scope can be defined as:

- How can a simple, easy-to-use tool be developed to predict the closing cryptocurrency exchange price?
- What is the best overall (or per-coin) model for predicting the closing price?
- How should model performance be evaluated to choose the best one for a given coin?

Having defined the scope, the problem formulation can be stated as:

How can a cryptocurrency price forecasting tool be developed for an audience with little to no financial and technical background?

1.3 Contributions

The contributions of this project aim to be twofold. Firstly, the software developed in this Thesis project is meant to be free, open source program to be used as a secondary tool to be used with more information (e.g. sentiment and network) and other more advanced programs to make a prediction regarding cryptocurrency closing value. In this manner, the author hopes he can contribute to the Cybersecurity efforts of ensuring safe and well-informed cryptocurrency investments.

Secondly, as far as the author knows, there are no studies which use datasets as current as the ones analysed in this project, therefore, the results gathered from their evaluation can be used to provide more information to the ML community on the performance of each algorithm and arrangement.

1.4 Report Structure

This chapter has briefly explained the context around which this project has revolved, laying out its full breadth as well as the relevance of the proposed tool, *CincoCrypto*, and its results. In **Chapter 2**, relevant key pertaining cryptocurrencies, price forecasting and the ML algorithms employed are explored. **Chapter 3** covers the criteria used to find academic articles from peer-reviewed journals, and the relevant papers for the project are described. **Chapter 4** describes the development tools employed in the creation of *CincoCrypto*, a high-level overview of the tool before delving deeper into the project stages. **Chapter 5** lays out the evaluation metrics used to assess model performance as well as their results for a single run of the tool. **Chapter 6** concludes the report and **Chapter 7** explains future work which could be carried on to improve the performance of *CincoCrypto*. In **Appendix A** the plots for the datasets from four coins is given, and a table containing the values from the output file generated by the tool can be found in **Appendix B**.

The introduction chapter has served to explain the motivation, scope and contribution of this project to create a free, open source tool (*CincoCrypto*) whose tentative results can be used as secondary information to make cryptocurrency predictions. The following section provides information on the background required to develop the tool. Any date-based statistics or exchanges mentioned from this section onwards have been taken on the 6^{th} May 2023 unless it is stated otherwise.

Chapter 2

Background

The key concepts and algorithms employed in this Thesis are explained in this chapter and, together with the Introduction, it should allow the reader to understand the models in *CincoCrypto* as well as the significance of the results obtained.

2.1 Cryptocurrencies

According to the US Internal Revenue Service (more commonly known as IRS), digital currency represents value in the digital realm and, as any kind of currency, it can be used as a unit of account, storage of value and exchange medium. Cryptocurrencies are a type of digital currency which use cryptography to ensure secure transactions and record said transactions on a digital distributed ledger (e.g. blockchain) [**11**]. An essential program or application used by cryptocurrency users is the digital wallet, which holds the keys to the coins owned by the user. However, should the keys get lost or stolen, the user would be unable to recover them and, thus, their coins would be irretrievable. Digital wallets can be one of two types: hot (they are connected to the internet) or cold (they are hardware wallets or private key printouts) [**12**]. Hot wallets (e.g. mobile wallets) are appropriate for operational accounts which make regular transactions whereas cold wallets are more appropriate to store value [**13**].

The history of the cryptocurrency revolution dates back to the release of BTC in 2008 [**14**] after Satoshi Nakamoto published their infamous White Paper which detailed the workings behind said coin [**15**]. Their paper revolutionised the concept of currencies as it solved the issue of double-spending (i.e. the same token in a user's wallet is spent multiple times [**16**]) by using a peer-to-peer network with a Proof-of-Work consensus mechanism to validate transactions. The hash of these transactions are given a timestamp and published in a distributed, public blockchain which serves to validate them, thereby, requiring no central institution (e.g. a bank) or party to oversee the Bitcoin network (see **Figure 2.1**). This system works as long as 51% of all nodes in the network cooperate for the benefit of the network, thus, a long chain is created which cannot be outpaced by the smaller percentage of attacking nodes. As of the 29^{th} May 2023 13:24:24 UTC, there is a grand total of 25,027 active cryptocurrencies [**9**] and despite them using different consensus mechanisms and/or network implementations, their foundations are all based one way or another on

the pioneering work from Satoshi Nakamoto.

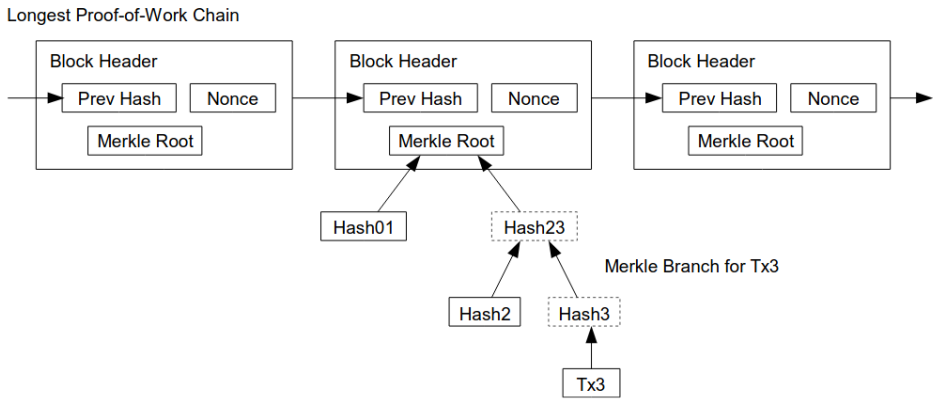


Figure 2.1: Blockchain concept of the Bitcoin network [15]

Despite having met major scepticism and curbs for a number of reasons, such as not being backed and regulated by any public or private entities [**17**], cryptocurrencies have progressively become more adopted worldwide. They are currently used to buy a number of products on e-commerce websites such as Tech giants like Microsoft and AT&T [**18**]. In addition, to ecommerce purchasing, the Central African Republic and El Salvador are, as of the 29^{th}) May 2023, the only two countries worldwide which have enabled cryptocrurrencies as legal tenders (i.e. officially recognised currency), showing their potential as an alternative,if not a replacement, to physical currencies [**19**]. Furthermore, cryptocurrencies have taken an active role in the ongoing Russia-Ukraine war conflict via donations to Ukraine, totalling more than $200 million as of the 24^{th} February, proving cryptocurrencies can be used as a means for large-scale donations [**20**].

In this project, the top 5 cryptocurrencies are used to train the models developed for *CincoCrypto*. The Bitcoin network (as explained above) is the backbone of BTC, the first ever successful cryptocurrency which holds the largest market cap dominance (45.01%) [**21**]. Below the networks of the other four coins are briefly explored in the order from largest to lowest market cap value:

1. Ethereum - Launched in 2015, it is a programmable network which allows for the deployment of smart contracts. Ethereum uses a Proof-of-Stake consensus algorithm [**22**]. Its cryptocurrency, (ETH), holds the second largest market cap dominance (18.9%) [**21**].
2. Tether - Unlike Bitcoin and Ethereum, this is the first stablecoin to be ever made. Therefore, the exchange from fiat currency (USD) to digital currency, and vice versa, is done 1-to-1, offering price stability [**23**]. Tether, whose coin is USDT, uses a Proof-of-Reserves consensus algorithm since the reserves used to back the coin will always be the same if not greater than the number of Tether coins in circulation. The cryptocurrency was launched in 2014 [**24**] and holds the third largest market cap dominance (6.47%) [**21**].
3. Binance - It is another programmable network due to having been forked from *go-ethereum* back in 2017. Binance enables smart contracts to be used and employs

a Proof-of-Staked Authority consensus algorithm **[25]**. Its cryptocurrency, (BNB), holds the fourth largest market cap dominance (4.06%) **[21]**.

4. USD Coin - It is another stablecoin based on the *usd*. According to its official website, USD Coin, whose coin is USDT, is backed by fiat reserves with high liquidity, thus, the cryptocurrency uses a Proof-of-Reserves consensus algorithm **[26]**. The cryptocurrency (USDC) holds the fifth largest market cap dominance (2.39%) **[21]**.

2.2 Cryptocurrency Trading and Investment

In 2010, shortly after being made public, Bitcoin was valued for the first time when a user decided to sell their 10,000 BTC for two pizzas **[27]**. Those same 10,000 BTC would be worth today around 289 million USD **[28]**. As shown by CoinMarketCap in **Figure 2.2**, cryptocurrencies have had massive fluctuations over the past years, showing their highly volatile nature but also their potential as a trading (short-term) or investment (long-term) opportunity. According to **[29]** the main advantages of trading or investing in cryptocurrencies is the fact that they have emerged as a new asset class, being considered as a potential for risk diversification due to their different behaviour compared to traditional investments and their high immunity ot the rising inflation. Besides, their relatively young age means cryptocurrencies can still show a great upside potential as new changes arise (e.g. stablecoins) and they become more widely adopted worldwide.

Figure 2.2: Total Cryptocurrency Market Cap (30/04/2013 - 06/05/2023) [30]

However, by the end of 2022, cryptocurrencies suffered some major blows and scandals, the main ones being the collapse of the digital exchange giant FTX, and the collapse of the two Terra coins, reducing the global cryptocurrency marketcap value significantly (see **Figure 2.2**) **[31]**. Therefore, cryptocurrencies are considered a speculative market which can be greatly affected by sentiment like media hype and celebrities' opinion **[32]**. As stated in Forbes Advisor, the main risks of investing in cryptocurrencies consist of the loss

of capital due to price fluctuations, lack of government policies to regulate the use of crypto assets, fraud (as in the case of FTX) and hacks ($3.2 billion worth tokens stolen in 2021) [**33**]. Moreover, an important entry-barrier to cryptocurrencies is their technical complexity as crypto-assets can be difficult to comprehend for non-technical users, therefore, putting them off from trading or investing [**32**].

Despite these disadvantages, Binance assures cryptocurrency investment can be safe as long as investors use common sense by doing their own research beforehand, starting small and diversifying, and staying constantly involved in the cryptocurrency landscape [**34**]. In addition, to these basic guidelines, The Times newspaper [**35**] mentions some of the main strategies followed by traders in order to obtain return for their investment:

- Day Trading - Cryptocurrencies are bought and sold within a day to make use of short-term price movements.
- Hedging - One investment has its risk of loss partially if not totally mitigated by another investment (e.g. contracts).
- HODling - Tokens are bought and held as long as possible.
- Trend Trading - Depending on the rising or falling trend, investors buy or sell their tokens, respectively.

2.3 Time Series Forecasting

A time series consists of sequential data points which are evenly spaced out throughout a period of time (i.e. the independent variable) [**36**]. By using past and current values, time series allow the influence of one (univariate) or more (multivariate) factors/series on another variable/s to be studied over time. As stated in [**37**], the predicted value ($\hat{x}$) of a one-dimensional, discrete timeseries (x) is defined mathematically as:

$$\hat{x}(t + \Delta_t) = f(x(t - a), x(t - b), x(t - c), ...) \tag{2.1}$$

Since time is involved, time series forecasting requires a different approach than the typical random-based splitting used in conventional Machine Learning. This alternative approach is called time-based splitting, which essentially separates data based on the timestamp. Time series can be found everywhere from Finance to E-commerce and Business, and they have three main components:

- Trend - Movement change over a period of time
- Seasonality - Periodic behaviour (i.e. spikes or drops) due to a number of factors (e.g. natural occuring patterns or social behaviour-defined events).
- Residual - Unpredictable fluctuations [**38**]

Machine Learning algorithms are commonly used in Time Series Forecasting given their ability to process and analyse large quantities of data with high speed and efficiency in an automated fashion.

2.4 Machine Learning

Artificial Intelligence encompasses numerous fields amongst which ML is one of them (see **Figure 2.3**). This subfield is vaguely defined as a machine trying to mimic human intelligence. Therefore, machines are able to learn by themselves through experience **[39]**. An ML model has a life cycle consisting of a number of stages which range from design all the way to the deployment: problem formulation, data collection, data processing, model implementation, evaluation, and deployment (and maintenance)**[40]**. Furthermore, there are three functions a ML can have depending on the tasks it has been set to do: descriptive (data is used to explain occurrences), predictive (data is used to make predictions) and prescriptive (data is used to develop suggestions about what decision should be made) **[41]**.

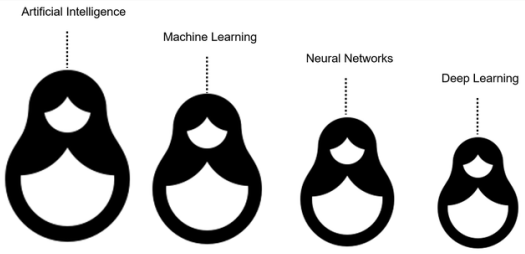


Figure 2.3: Russian nesting doll concept of AI and its subfields [42].

According to **[43]**, one of the most important factors to take into account when developing Machine Learning algorithms is how they are trained, since this is the defining trait which classifies them:

- Supervised Learning - Labelled datasets are used to train the algorithm. For example, an image classification algorithm could be given images of different types, such as dogs and cats, and the ML scientist would make the program know which one is which.
- Unsupervised Learning - The algorithm is given unlabelled data to identify patterns which humans cannot see or find on their own. For example, an algorithm could be given unlabelled sales data and identify the various types of clients involved in the purchases.
- Reinforcement Learning - By using a reward mechanism, the algorithm is trained through trial and error to learn what actions to take as time progresses. For example, autonomous vehicles would use reinforcement training to learn how to drive.

As seen in **Figure 2.4** from MathWorks, depending on the learning technique, ML can be used to solve problems belonging to one of three categories: Clustering, Classification, and Regression.

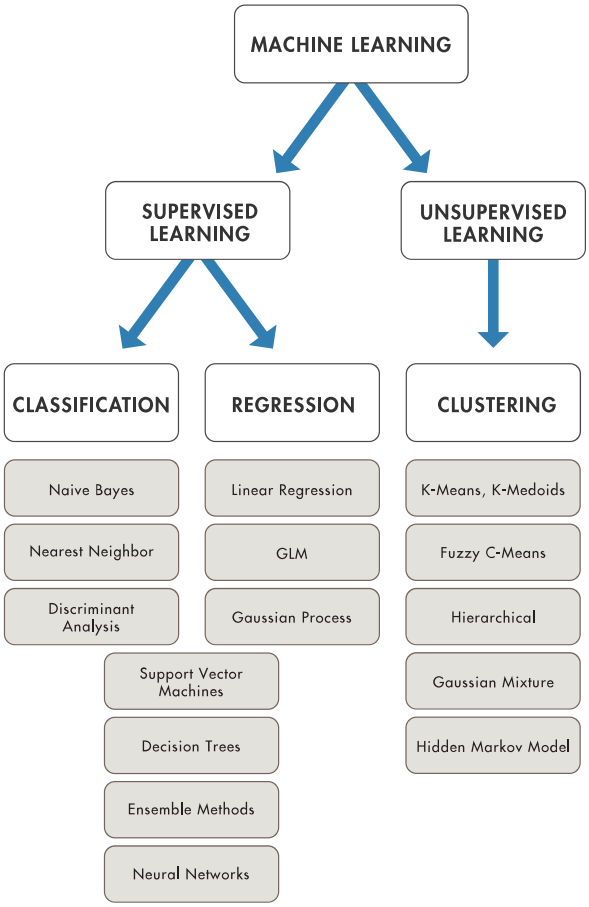


Figure 2.4: ML taxonomy [44]

The amount of values in the datasets, and the amount of data generated by the algorithm is required for the model to be able to learn, predict and make decisions. If the data is scarce, ML will not be of use [**39**]. In addition to data quantity, another important aspect is how data is split to build a reliable model. A model cannot use the same data for training and testing as its real performance would not be known and its behaviour would be unpredictable when used on real data [**45**]. There are three types of sets which can result from splitting the data:

- Training Set - The model is trained using this set, and allows any hidden features or patterns in the data to be identified.
- Validation Set - During training, this set is used to evaluate model performance. The results obtained from this set can be used to tune the hyperparameters and settings in the model.
- Test Set - It is used to test the model once training has been completed, acting as an unbiased source of information.

There is no absolute rule for the splitting ratio but there are some common, standard

splits used such as 80% training : 10% validation : 10% testing, and even 80% training : 20% testing [**46**].

Once a model is trained, it is essential to look at the performance metrics from the test set to check if the model is able to generalise to new data and is not overfitting. A model which overfits fits perfectly on its training data and learns the noise, making the model unable to generalise to new samples and thus preventing it from predicting or classifying. Therefore, a test set is always required to check if the model overfits by looking at the error rates and variance, and ensuring they are not too low nor too high, respectively. The causes for overfitting are many and varied such as long training time (i.e. too many epochs), small training set, data far too clean (no noise), and no feature selection employed. On the opposite side if data is not trained for a reasonable amount of time and the model complexity becomes far too simple, underfitting may occur and the model will not be able to develop a significant relationship between the input and output variables. Underfitting can be identified by looking at the performance metrics from the test set; high bias and low variance indicate the model underfits [**47**].

2.4.1 Deep Learning

A subset of ML which is currently very popular is DL. The algorithms under this area are trained to mimic the behaviour of the human brain via a ANN with as many as 150 hidden layers **Figure 2.5**.

For a network to classify as a DL algorithm, it requires a depth of more than three layers (inclusive the input and output layers) [**42**]. Within each layer, the nodes, or artificial neurons, are connected to the nodes of the adjacent layer\s, being assigned a weight and threshold for each interconnection. Data can only be passed from layer to layer if the output of a sending node is greater than the bias of a receiving node. Hence, DL algorithms require training data to learn and raise their accuracy [**48**].

DL is very attractive due to having achieved outstanding results which have even outperformed human-level performance in certain tasks. Its models have become highly successful as of late thanks to the large abundance of labelled data required to train them in the first place, and to the increase in computing power compared to a few decades ago [**49**]. In addition, DL algorithms enable automation for much of the feature extraction procedure, being capable of analysing large datasets (structured or unstructured) with little need for human interaction [**42**]. Therefore, DL has shown great promise to forecast values from time series data as shown in the study from [**50**].

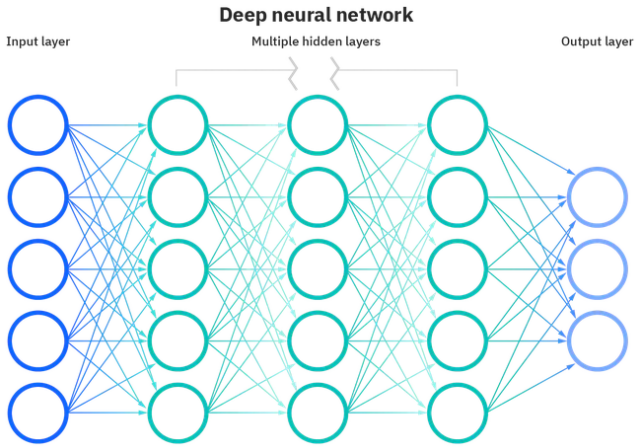


Figure 2.5: Concept diagram of a DL algorithm [42].

Within DL, RNN is an algorithm in which the output of a neuron in the hidden layer can be passed to any other neuron connected to it (i.e. a neuron in the previous, current or following layer). Therefore, they excel at remembering contexts and taking them into account when making predictions. However, RNN have an important drawback in that they only have a short-term memory which stores only the most recent data. When trained with the gradient method, the RNN gradient takes either infinitesimal values close to zero (vanishing gradient) or large values near infinity (exploding gradient). This prevents the neuron's weights from being changed at all during backpropagation [**51**]. Since its conception, a number of algorithms based on the RNN structure (e.g. LSTM, and GRU) have been designed to fix this issue. A simple RNN structure with a single tanh layer can be found in **Figure 2.6** below:

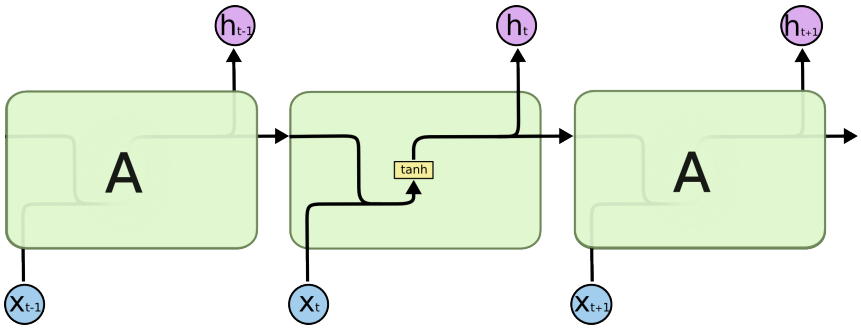


Figure 2.6: Simple RNN cell chain (tanh layer) [52]

2.4.2 Trained Algorithms

Since this project aims to forecast the following day's closing price (i.e. regression problem), four supervised learning models are developed. The first model uses a SVM algorithm (SVR) whereas the remaining three use DL algorithms (LSTM, GRU, and CNN).

Support Vector Regression

SVMs are a type of ML algorithm which can accurately predict time series values in non-linear, dynamic and undefined system. SVMs have shown higher performance than other non-linear methods such as multi-layer perceptrons. When applied to general regression analysis, SVMs are known as (SVR). The latter performs regression by using an estimating function whose curve fits the given time series values. Unlike other models, there is no actual model and the data is what steers the prediction since the curve function is determined via the observed data, which is also used to train the algorithm.

The key concept of how SVM/SVR works is the kernel function, which maps non-linear input space $x(i)$ to a, probably, higher dimension feature space $\phi(x(i))$ to which linear regression can be applied. Taking into account that SVR requires the dot products of its input values, a kernel function must satisfy a certain set of of conditions, known as Mercer's conditions, to be generated using the formula:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \tag{2.2}$$

Some of the common kernel functions which satisfy the mathematical conditions required by SVR to converge on a solution are Gaussian (the most common one), polynomial and hyperbolic tangent. The free parameters of the kernel function are very important to its performance, and consist of the constant C and the "tube size" or ϵ (i.e. the precision the function must approximate to). Both parameters are selected by the user through empirical computation.

In addition to higher performance, the main advantages of SVR compared to other algorithms, such as ANN, is that it is independent from the model and linear stationary processes as well as being an algorithm which ensures convergence to the most optimal solution using a small number of free parameters whilst being computationally efficient. However, the main disadvantage of SVR is that it requires high manual intervention (e.g. selection of the kernel function and its free parameters as well as the optimisation techniques among others). Still SVR is a noteworthy algorithm for time series prediction from which financial forecasting is the most researched application of said algorithm **[37]**.

Long Short-Term Memory

LSTM fixes the problem with RNN by storing certain information in long-term memory, which is known as Cell State. Besides, LSTM is also provided with a short-term memory, which is known as hidden state, to store the result from the previous calculation [**51**].

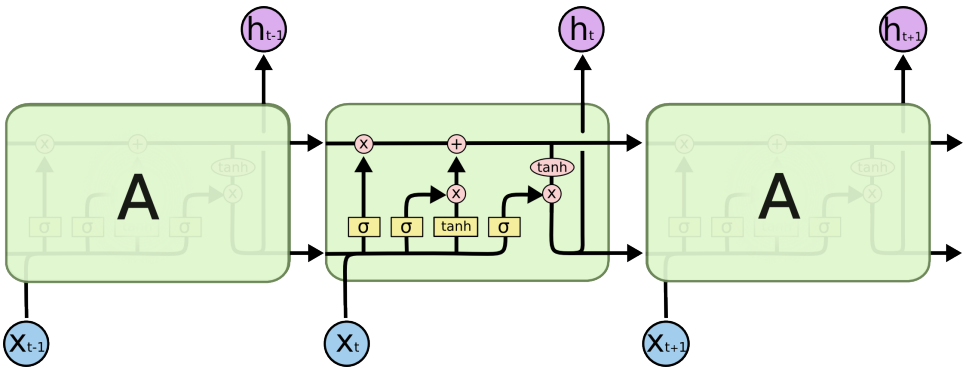


Figure 2.7: Chain of LSTM cells [52]

As seen in [**53**], the LSTM architecture uses a gating mechanism designed to regulate the memorisation process (see **Figure 2.8**). These gates store information as analogue values and can have one of two activation functions: tanh (non-linear, values between -1 and 1) and sigmoid (non-linear, values between 0 and 1). The former is used to regulate the state of the network whereas the latter is used to update/forget data.

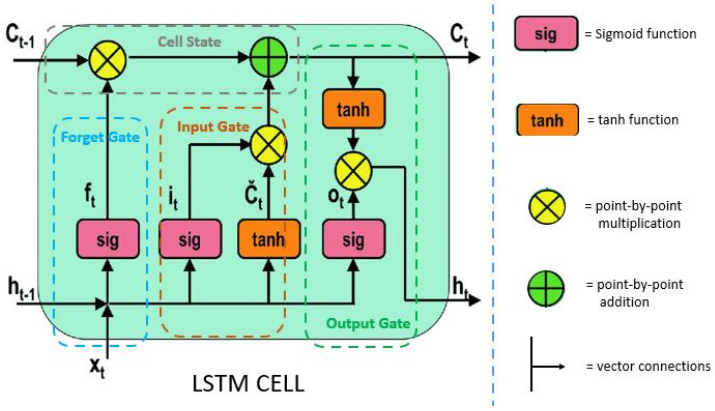


Figure 2.8: LSTM cell [53]

A typical LSTM cell consists of four core processes:

1. Forget Gate (f_t) - Prioritises information by deciding what information should be considered or ignored. Both, the current input (x_t) and the hidden state h_{t-1} are fed into the sigmoid (σ) function, which determines if the information from the previous output is required. Should the previous information not be required, the output of σ is tends towards 0. The equation for the forget gate vector is given as:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.3}$$

Where t is the timestep, W_f is the weight matrix between the forget gate and the input gate, and b_f is the connection bias at the forget gate.

2. Input Gate (i_t) - Updates the cell status by passing the current state and previous hidden state, in parallel, into the sigmoid and tanh functions. The sigmoid function gives an output between 0 (relevant) and 1 (irrelevant), whereas the tanh function will create a vector ($\tilde{C}_t$) to regulate the state of the network. The output of both functions undergo a point-by-point multiplication. The equations for the input gate and ($\tilde{C}_t$) vectors are given as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.4}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C \tag{2.5}$$

Where W_i is the weight matrix of the sigmoid operator between the input and output gates, b_i is the connection bias at the input gate, W_C is the weight matrix of the tanh operator between the cell state and the network output, and b_C is the bias vector with respect to W_C

3. Cell State (C_t) - With the information from the forget and input gates having been generated, it must be decided now if this information (i.e. new state) is to be stored in the cell state. The product between the previous cell state (C_{t-1}) and f_t is calculated, and the values are dropped should the result give 0. Then another product operation is performed between i_t and $\tilde{C}_t$. The new cell state is calculated by performing point-to-point addition on the results from the two multiplications:

$$C_t = f_t * C_{t_1} + i_t * \tilde{C}_t \tag{2.6}$$

4. Output Gate (o_t): It calculates the LSTM output, which is equivalent to the next hidden state value h_t. x_t and h_{t-1} are fed into the third sigmoid function. The result is used to calculate h_t by multiplying it with $tanh(C_t)$. After the operation is complete, the new hidden state and the new cell state are forwarded to the following step. The LSTM output equation is given as follows:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.7}$$

$$h_t = o_t * tanh(C_t) \tag{2.8}$$

Gated Recurrent Unit

A type of RNN, GRU is one of the latest generation in DL algorithms. Like LSTM, it also uses a gated mechanism which uses the sigmoid and tanh functions as activation functions. However, it is more lightweight as it does not have an input gate nor a cell state and merges the input and forget gate into one update gate, using the hidden state to transfer information. Therefore, GRU is less complex and much faster than LSTM.

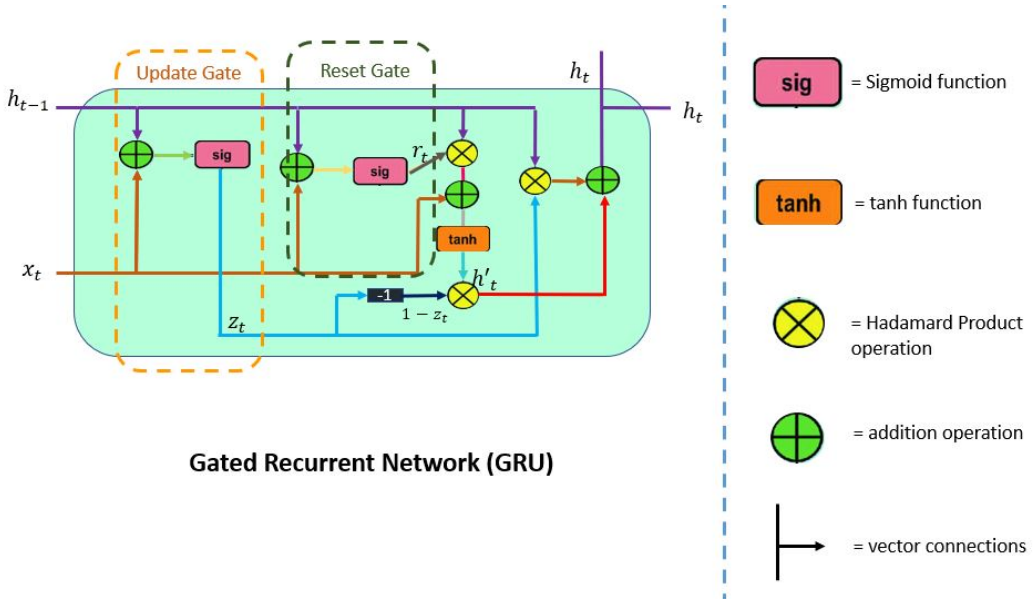


Figure 2.9: GRU cell [54]

As stated by [**54**], only two gates are used in a GRU unit:

1. Update Gate (z_t) - Prioritises information by deciding how much of the previous information stored in the hidden state ($h_t - 1$) should be forwarded to the next state. The equation for this gate is given as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{2.9}$$

   Where W_z is the update gate weight matrix, x_t is the input vector, and b_z is the connection bias at the update gate.

2. Reset Gate (r_t) - On the contrary to the update gate, the reset gate determines how much of the previous information should be neglected. It has the same formula as the update gate:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{2.10}$$

   Where W_r is the reset gate weight matrix, and b_r is the connection bias at the reset gate.

According to the comparative study made by [**55**], there are no conclusive evidences on whether GRU outperforms LSTM or vice versa.

Convulational Neural Networks

A type of DL algorithm, CNNs convolve identified features with input data, using convolutional layers. For this very reason, CNNs tend to be used the most in problems involving image recognition, but they can be used with time series as well to make predictions. In CNNs, manual feature extraction is not required since they determine the relevant features as the model is trained, using between tens or hundreds of layers to improve model sophistication but at the cost of greater complexity [**49**]. CNNs are different from regular ANNs in that they use parameter sharing, where all the nodes in the layer are connected to one another and the weights have an associated fixed value, making CNN less computationally taxing. As explained in [**56**], the algorithm is also very attractive as it looks at an information segment at a time (e.g one patch of an image at a time), requiring less neurons with less parameters compared to other conventional algorithms.

CNN uses a series of layers which are explained below:

1. Convolutional layer - It applies a convolution filter or kernel to the data to identify features. Said kernel is initialised with random values and placed in every possible position within the input matrix. Element-wise multiplication is performed between kernel-sized data matrix and the input data, and the results are added up. This layer is the most computationally-intensive as it enables the discovery of features across the input data space. The resulting matrix or feature map is usually passed through the ReLU activation function to provide non-linearity to the model by replacing all the negative values with zero.

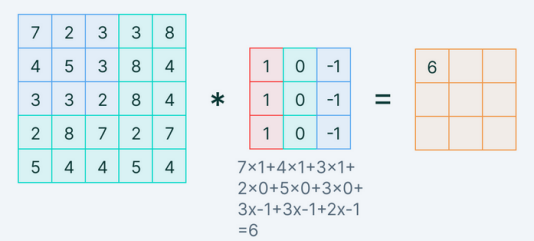


Figure 2.10: Convulution operation [56]

2. Pooling Layer - It is placed between two convolutional layers to reduce the spatial size of the data representation, reducing the computation and making the model more resistant against distortions and variations. Two important hyperparameters of this layer are the window size and the stride. Depending on the type of pooling applied, the maximum value (which provides better performance) or the average value of the window is taken to resize the input.

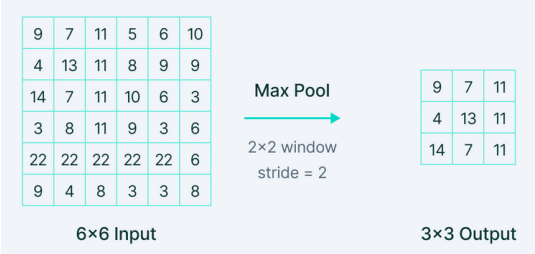


Figure 2.11: Pooling Operation [56]

3. Normalisation Layer - It is placed between the convolution and pooling layers to normalise the output of the previous layer. This allows the algorithm to be more independent and avoids overfitting. The normalisation layer tends to be used only on relatively simple architectures as it makes no effective contribution to the learning process.

Once these layers are assembled in the desired configuration, their arrangement can be replicated as many times as required to increase model perceptiveness when identifying finer details but at the cost of increased complexity and computational power. At the final stage of a CNN mode, the output of the CNN part is flattened and fed into a regular neural network.

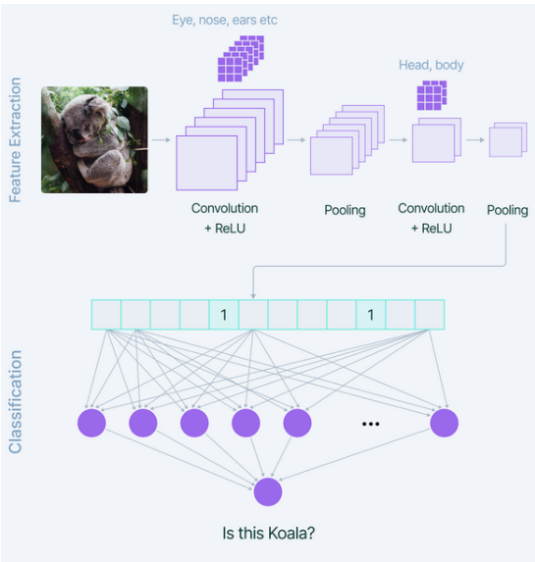


Figure 2.12: CNN model example [56].

The background behind the core concepts of the project have all been explained in this section, allowing the reader to understand the topic being studied as well as the algorithms employed in *CincoCrypto*. In order to determine the relevant model hyperparameters and understand the significance of the results gathered from the tool, a literature review of peer-reviewed journals and articles is required to be conducted.

Chapter 3

Literature Review

This chapter explores the methodology used to find relevant academic articles. Having already described the background of the project in the previous chapter, the methods and results from relevant papers are briefly discussed to understand the configurations used in developing the models for *CincoCrypto* as well as the results obtained from them.

3.1 Search Criteria

Throughout the project, the Google Scholar database has been used to retrieve results from well-known peer-reviewed journals, but the amount of results was often overwhelming and polluted with papers from predatory journals, requiring to submit the same search requests on the four journals of interest: IEEE Xplore, Wiley Online Library, ScienceDirect, and ACM Digital Library.

Initially, broad, vague searches regarding cryptocurrency and price prediction were made until suitable papers could be found. The search terms employed were *"Blockchain AND Machine Learning"*, *"Blockchain AND Price Prediction"*, *"Cryptocurrency AND Machine Learning"*, and *"Cryptocurrency AND Price Prediction"*.

Once the algorithms of interest were identified, two types of searches were made: the algorithm name ANDed with *cryptocurrency forecast* (e.g. *"LSTM AND cryptocurrency forecast"*), and the algorithm name ANDed with *price prediction* (e.g. *"LSTM AND price prediction"*.

3.2 Model Algorithm Papers

This section groups the papers found by relevance to the Machine Learning algorithms employed in the price forecasting tool models.

3.2.1 Support Vector Regression

In the work published by [**57**], the effect kernel choice has on the performance of trading volume forecasting is studied. Using the Bayesian optimisation method to tune the hyper-

parameters, three common kernel types are investigated: linear, polynomial, and RBF. The data from thirty cryptocurrencies is splitted into a training (80%) set and testing (20%) set, and the output from the SVR algorithm is benchmarked against the ARIMA, Lasso and Gaussian processes. The metrics employed are RMSE and MAE. The results show that Bayesian Optimisation kernel is best for making next day predictions whereas the polynomial kernel is best at predicting the next-week volume. In both cases, SVR outperforms the other processes due to its good generalisation when analysing a small dataset, robustness to outliers, and its nature as a non-linear predictive system.

A model comparison between LSTM and SVR with RBF kernel is carried out in **[58]**. The LSTM model is developed with one dense layer, a ReLU activation function and an Adam optimisation function. Only OHLC features are analysed from three cryptocurrency datasets: BTC (8 years), ETH (5 years), and LTC (7 years). The evaluation metrics (i.e. RMSE and MAPE) shows that SVR with RBF kernel performs better than the proposed LSTM model.

Parameter optimisation with Grid Search Method is used in **[59]** on SVR with Radial Basis Function. The model predicts the price of the next five weeks by using weekly BTC price data from January 2018 to March 2020. The data is normalised using min-max normalisation as it keeps data relationships. MAPE is used to evaluate the results, giving the least error (10.738%) to an SVR model with $C = 5$, $\epsilon = 0.004$, and $\gamma = 0.07$.

3.2.2 Long Short-Term Memory

A comparison study between LSTM and other ML and DL models (namely SVR, GRU and CNN) is carried out in **[60]**, where a 3-day prediction window is used on four cryptocurrencies datasets: BTC, BTC, LTC and ETH. The datasets range from the 15^{th} September 2016 to 5^{th} November 2018 and the metrics used are RMSE, MAE, MAPE, and R^2. The LSTM model is shown to have the lowest error and highest R^2 score.

In **[61]**, an LSTM-GRU hybrid model is proposed where LTC and ZEC, as well as their interdependency to the parent coin, are analysed. The datasets include only OHLC data from 2016 to 2018, and are splitted into a training and testing set. Various forecasting windows are used (i.e. 1 day, 3 days, 7 days and 30 days) and MSE is used as the main metric to evaluate the result. The model is trained for 50 epochs and shown to predict the values of ZEC accurately except when the window value used is 30 days.

3.2.3 Gated Recurrent Unit

Various time series architectures are compared in **[62]**. The architectures compared are, namely, ARIMA, SARIMA, GRU, RNN, LSTM, and Facebook Prophet. A four year dataset (2018 to 2022), comprising OHLC and volume data,is splitted into three datasets (i.e. 80% training, 10% validation and 10% testing) and used to predict the Closing value. The predictions are assessed using MSE and RMSE, showing that GRU outperforms the other models used in the study.

The work from **[63]** compares LSTM and GRU, keeping the hyperparameters fixed whilst analysing model performance based on the number of epochs used from 1 to 30. Three cryptocurrency sets (BTC, ETH, and LTC) are used. Model performance is measured using RMSE and MAE, and the results indicate that GRU is better at predicting the downward trend whereas LSTM is better at analysing the upward trend.

3.2.4 Convolutional Neural Network

The paper from [**64**] makes a comparative study between CNN, RNN and LSTM, showing that the LSTM model is the best performing one out of the three. Four performance indicators (i.e. MAE, MAPE, RMSE and R^2) are used on 4-year long dataset (2017 - 2021) with OHLC, volume and market cap values. The data is normalised using the MinMaxScaler function. The CNN model designed has a 1-Dimesional convolutional (*Conv1D*) layer to extract the features, one pooling layer (*MaxPooling1D*) to reduce the size of the *Conv1D* output, a Flatten to reshape the data, and a *Dense* layer to shape the output of the model. The model is run for 1000 epochs, and later, 3000 epochs to predict the 60^th day Close value

The study published by [**65**] analyses two CNN and three LSTM models for Open stock market prices from fifty companies (2008-2020). The models predict the price of the following week, using RMSE as the main evaluation metric. The results show that the univariate CNN using a window of one week is the fastest model whereas the LSTM model with a window of two weeks is the most accurate one.

Alternatively, a similar study proposed by [**66**] compares LSTM, RNN and CNN for stock prediction as well but in this case, CNN is shown to be the best performing algorithm, being able to predict based on instant values unlike the other two models.

3.3 State of The Art

Apart from SVM and RNN-based models, there are other studies which explore other algorithms. In [**67**], three boosting algorithms (i.e. XGBoost, AdaBoost and CatBoost) are used to forecast the market value of BTC and Ripple, showing that the algorithms employed can make accurate predictions.

Furthermore, in [**68**], a stochastic ANN is proposed to simulate market volatility. Unlike the previously mentioned studies, this one analyses daily market statistics (i.e. Low, High and trading volume), blockchain network information (e.g. hash rate, transaction fee, etc.), and social sentiment (e.g. google trends and tweet volume). The model is trained for 700 epochs to predict the 8^{th} day price, using a window value of 7 days. A two-year dataset (from 2017 until 2019) is used and splitted into a training (75%) and testing (25%) datasets. The evaluation metrics employed are MAPE, MAE, RMSE and MSE, and show that almost all the stochastic models used outperform the deterministic ones, thus, the stochastic neural network is effective at predicting market volatility.

Similarly, [**69**] shows that external factors such as USD/EUR exchange rate and the intricate interdependency between certain coins can have an important effect in cryptocurrency prices.

The search criteria and the relevant papers found in the early stages of the project have been discussed. The papers have been divided according to the model algorithm they describe or compare, and latest forefront research on cryptocurrency forecasting. The methods and results from these studies can be used to understand the criteria used when developing *CincoCrypto* and analysing its performance.

Chapter 4

Procedure

Having covered the background behind the core concepts and algorithms as well as the relevant papers for cryptocurrency price forecasting using ML, it can be described next how the price forecasting tool has been designed to require minimal input from the user (i.e. adding the dataset names into a list) and predict the next day price for the top five cryptocurrencies.

4.1 Development Tools

4.1.1 Programming Language, Libraries and Modules

Python is used as the base programming language for the project given its beginner-friendly, open-source and general-purpose nature [**70**]. This high-level programming language is amongst the most popular languages with plenty of online support via communities and numerous tutorials, making it relatively straightforward to learn as well as being versatile for developing third party modules for a wide range of applications [**71**].

ML is based upon automation, requiring programs to be platform-independent, simple and consistent wherever and whenever possible. The advantages offered by Python is what makes it one of the most powerful languages for ML, which already has specialised Python libraries [**72**]. Therefore, taking all these factors into account, Python is used to develop *CincoCrypto*. Furthermore, seven main libraries and modules are used for the four main stages of the program:

Data Collection & Preprocessing Libraries

These two program stages go hand in hand, requiring the CSV datasets to be uploaded, pertinent features extracted and data put into the right format to be fed into the ML algorithms. Therefore, the libraries used happen to be common for both:

- **os** - It is a module which enables the use operating system dependent functionalities [**73**]. In the price forecasting tool, *os.getcwd()* is used to retrieve the absolute path to the current working directory, which contains the datasets of the five cryptocurrencies to be analysed.

- **pandas** - This library allows for data processing and analysis of different file formats (e.g. CSV and text files). Its main feature are the *dataframe* objects, which can hold multiple data types within a two-dimensional, tabular-like structure and enable fancy indexing. Therefore, said objects are ideal for data manipulation, allowing highly granular actions such as data alignment, merging and reshaping as well as handling of missing data amongst many others. In addition, the library is highly efficient and fast given that it is party written in C [**74**], [**75**].
- **numpy** - Array computing allows for large scale mathematical operations in a highly efficient manner by means of vectorised code, which is already precompiled in C. At the core of the library is the *ndarray* object, which, unlike standard Python sequences (e.g. lists), it requires arrays to have a fixed size upon creation where all values have an homogeneous data-type [**76**]. In Python, numpy is considered the de facto library for array computing as it enables developers to use arrays and matrices of any size and apply advanced mathematical functions to them. Therefore, array computing is the basis for highly mathematical fields such as image processing, artificial intelligence, and machine learning [**77**].

Model Implementation Libraries

In order to develop ML models capable of price forecasting, two libraries are required:

- **sklearn** - Its full name being scikit-learn [**78**], this library holds a great variety of algorithms and tools for data preprocessing as well as model creation, fitting and evaluation [**79**]. In the project, sklearn is used to implement the algorithms for SVR, data normalisation, and evaluation metrics.
- **keras** - It is a deep learning API built on top of the TensorFlow platform made to be simple (*developer cognitive load* reduction), flexible (*progressive disclosure of complexity principle adoption*) and powerful (*high performance*). Keras is highly scalable, allowing for a straightforward deployment in a vast number of systems and devices (such as Android, iOS and embedded devices). Furthermore, the API provides critical abstractions and building blocks to design models with high running and debugging speed and easy maintainability, thus, being ideal for ML-based applications [**80**]. The DL algorithms (i.e. LSTM, GRU, and CNN) are all developed using this library.

Model Evaluation Libraries

After the models are implemented, it is required to compare and evaluate the predictions against the closing values for each coin. Therefore, evaluation metrics (from sklearn library) and model fitting time are used as indicators to assess model performance.

- **time** - It is a module containing time-related functions of which most invoke their name-equivalent C library functions. The only function used from this library is the time() function, which returns the number of seconds since the epoch (i.e. January, 1, 1970, 00:00:000 UTC time) [**81**].
- **matplotlib** - It is a Python library, enabling the creation of high quality, static and customisable plots which can be embedded both in JupyterLab and GUIs. Amongst

the various plot types available, the main ones are basic (e.g. x, y plots), arrays and fields (e.g. contour maps), statistics (e.g. histograms), unstructured coordinates (e.g. tricontour maps) and 3D (e.g. 3D surface graphs).

4.1.2 Development Environment

Using Python and the libraries mentioned above, the ML models are developed in JupyterLab (also known as IPython), which is a flexible, web-based, dynamic environment. JupyterLab enables the usage of different kernels which run code in many programming languages as well as environments. The most interesting feature offered by JupyterLab is the Jupyter notebook. These notebooks merge live, executable code with interactive features (such as LaTeX equations) **[82]** whilst using Jupyter's language-agnostic behaviour, uniting various open source communities towards exploiting the advantages of this tool **[83]**. Therefore, *CincoCrypto* is stored in a Jupyter notebook.

4.1.3 Hardware

Throughout the project, *CincoCrypto* has been run and tested in a Lenovo ThinkPad T570 W10DG laptop which comes with a 7^{th} generation Intel Core i7 processor, 8GiB DDR4 memory and a 238,5Gb NVMe.

4.2 CincoCoin Implementation

The previous section has given a detailed explanation of the programming languages, libraries and environment used to develop *CincoCrypto*. The laptop machine specifications in which the tool has been executed and tested have also been covered. Therefore, this section gives first a brief explanation of the datasets used and a high level outline of the program before delving deeper into the design and implementation of the four main stages of the tool.

4.2.1 Datasets

The program uses 1-year long datasets (06/05/2022 - 06/05/2023) of the top five cryptocurrency datasets: BTC **[84]**, ETH **[85]**, USDT **[86]**, BNB **[87]**, and USDC**[88]**. The datasets (in CSV format) are downloaded from CryptoCompare, which is a global data provider for the cryptocurrency market founded in 2014 **[89]**. *CincoCrypto* has been developed using datasets from CryptoCompare, and, thus, it can only analyse datasets from said data provider due to the number features, and the way they are arranged within the CSV file. The datasets include OHLC and trading information within 8 columns as shown below in **Table 4.1**.

Table 4.1: BTC dataset sample (last five rows)

time	timeDate	close	high	low	open	volumefrom	volumeto
1682985600000	2023-05-02	28694.85	28896.62	27897.95	28086.19	32635.23	927167494.21
1683072000000	2023-05-03	29041	29274.35	28155.84	28694.85	40240.91	1150094643.14
1683158400000	2023-05-04	28866.54	29370.41	28703.83	29041	27555.47	798614954.67
1683244800000	2023-05-05	29550.84	29697.23	28853.1	28866.54	35357.4	1037025191.1
1683331200000	2023-05-06	28845.45	29857.86	28459.77	29550.84	23039.88	670625654.68

The *time* column is discarded as only *timeDate* is used as the dependent variable to forecast the closing price of the next day. The other columns are left unmodified as their purpose is to be used as features for the models.

4.2.2 High-Level Program Outline

CincoCrypto is programmed to take only five datasets from CryptoCompare whose CSV files must be in the same CWD as the notebook, and calculates the next day price by looking at the information from the day prior. The program is made with simplicity in mind as the user is only required to manually write the names of the datasets (without the .csv extension) into the *cryptos* list at the top of the program. When the program is run, the datasets are uploaded one by one, the *time* column is removed and, instead, another one is added for the independent variable (*1_day_price_forecast*). Most studies in **Chapter 3** only use OHLC data, but [**57**] states there is a relationship between trading volumes and closing price, thus, *OHLC* and trading columns are kept as features for the models. In addition, the datasets are splitted into a training (80%) and testing set (20%) as done in [**57**].

Once a dataset is fully preprocessed, it is fed into the four models (i.e. SVR, LSTM, GRU, and CNN) and the evaluation metrics (i.e. MAPE, MSE, R^2) are calculated for each model. This is repeated for the other four datasets until all models have been evaluated and their results are stored in *Metrics_Results.csv*. Finally, charts are drawn to examine the performance of the models on all five coins for the user to choose the best model; the charts are saved in the CWD.

4.2.3 Program Stages

Below the program stages followed by *CincoCrypto* from the moment the user writes the names of the cryptocurrency datasets to be assessed up until the model evaluation results are stored into an output csv file and the appropriate graphs are drawn.

Data Collection

The first three program stages and part of the fourth one are all executed inside a main for loop which uploads the datasets one by one. Initially, five cryptocurrency datasets from CryptoCompare are downloaded and stored in the same CWD as that of the notebook. Then, the notebook is opened in the environment of choice and the names of the five datasets (without the .csv extension) are added into the *cryptos* list for the program to identify the files it must upload. (The *window* variable is left as 1 for the program to calculate next day predictions). Once the names are added, the program can be run. The

for loop is designed to use *cryptos* with a *crypto* iterator, thus, in one loop all four models are trained and evaluated for a given coin dataset as specified by *cryptos*. For a dataset to be uploaded, *link* is declared in every for loop iteration to store the absolute path to a cryptocurrency dataset. Next, the user-defined function *preprocess()* is called and a dataset is uploaded, using *link*, and stored into the *Dataset* dataframe.

```
directory = os.getcwd()

cryptos = ['Bitcoin', 'Ethereum', 'Tether', 'Binance', 'USD Coin']

result = {}
allCrypto = []
y_plot_predvals = {}

for crypto in cryptos:
    link = directory + '/' + crypto + '.csv'

    preprocess()

    y_plot_predvals[f'{crypto}'] = {}

    #Model Implementation code further down in loop
```

Listing 4.1: Data Collection stage [7]

4.2.4 Data Preprocessing

Within *preprocess()*, the *time* column in *Dataset* is removed and a new column (*1_day_price_forecast*) is added instead whose values are the same as in the *Close* column but shifted by one day to act as the independent variable of the models (see **Table 4.2**). Moreover, the index of *Dataset* is changed to be the dates within the *timeDate* column, which is the dependent variable, to ease chart plotting later in the Model Evaluation stage.

Table 4.2: BTC *Dataset* after modification (last five rows)

time	timeDate	close	high	low	open	volumefrom	volumeto	1_day_price_forecast
1682985600000	2023-05-02	28694.85	28896.62	27897.95	28086.19	32635.23	927167494.21	29041
1683072000000	2023-05-03	29041	29274.35	28155.84	28694.85	40240.91	1150094643.14	28866.54
1683158400000	2023-05-04	28866.54	29370.41	28703.83	29041	27555.47	798614954.67	29550.84
1683244800000	2023-05-05	29550.84	29697.23	28853.1	28866.54	35357.4	1037025191.1	28845.45
1683331200000	2023-05-06	28845.45	29857.86	28459.77	29550.84	23039.88	670625654.68	NaN

Given *preprocess()* is called five times to upload and preprocess the five CSV files, the values in the *close* column are all stored in the *AllDatasets* list to be used later when making the charts in the Model Evaluation stage of the program. The forecasting horizon is set to "1" inside a *window* variable at the top of the program as only information from a given day is used to predict the next day closing price.

A *train_index* variable is created to be used as an index to split the data into the training and testing sets whilst removing the row from *Dataset* as it has a NaN in *1_day_price_forecast* due to there not being more data after the 6^{th} of May 2023 (see **Table 4.2**). In addition, the next day closing values of the testing region are stored in *uy_test* to be used later to calculate the evaluation metrics.

Next, as done in **[64]** and **[59]**, the data is normalised using the MinMaxScaler function (range 0 to 1) to ensure data relationships are maintained, and, using *train_index*, the training and testing sets are created. The training set consists of *X_test* and *y_train*, and the testing set consists of *X_test*, where *X* and *y* are the feature space and the independent variable, respectively. This can be graphically shown in **Figure 4.1** below.

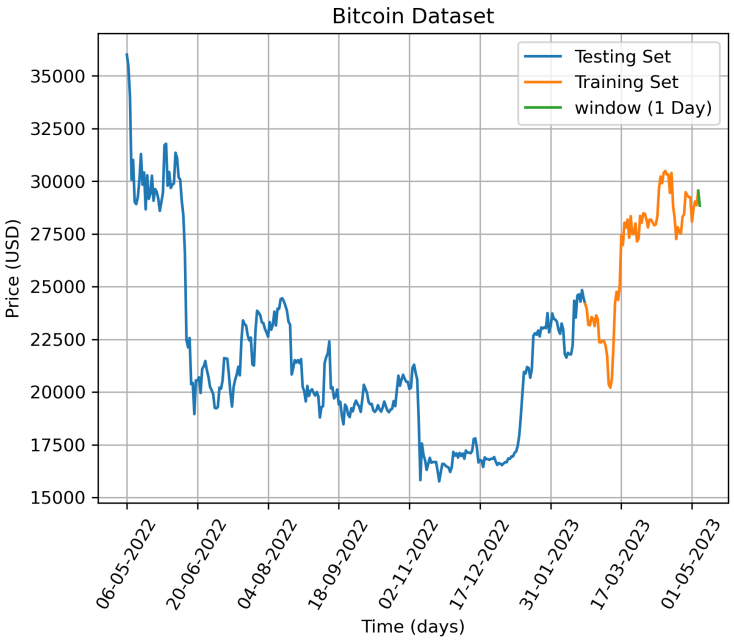


Figure 4.1: BTC dataset plot

Where window (1 day) is the forecasting horizon and the number of end rows (NaN) which have to be removed from *Dataframe*. The plots for the other four datasets can be found in **Appendix A**.

```
def preprocess():

    Dataset = read_csv(link, usecols = [1,2,3,4,5,6,7])
    Dataset[f"{window}_day_price_forecast"] = Dataset[['close']].shift(-window)
    Dataset.index = pd.to_Datetime(Dataset['timeDate'], format = '%Y-%m-%d')

    AllDatasets[f'{crypto}'] = Dataset['close'].to_numpy()

    train_index = int(((len(Dataset))-window) * 0.8)

    X - Dataset.iloc[:,1:7]
    X = X[:-window]
    y = pd.DataFrame(Dataset[f'{window}_day_price_forecast'].copy())
    y = y[:-window]

    uy_test = pd.Dataframe(y.copy())
    uy_test = uy_test[train_index:]
    uy_test = uy_test.to_numpy()

    scalerX = MinMaxScaler(feature_range=(0,1))
    scalery = MinMaxScaler(feature_range=(0,1))

    X_train = scalerX.fit_transform(X)[:train_index]
    X_test = scalerX.fit_transform(X)[train_index:]
    y_train = scalery.fit_transform(Y)[:train_index]
    y_train = y_train.ravel()
```

Listing 4.2: Data Preprocessing stage [7]

4.2.5 Model Implementation

The program exits *preprocess()* and returns back to the main for loop where the four models are implemented. To hold the various results from each of the models and metrics for each coin, the following dictionaries and list are declared before the for loop (see **Listing 4.1**):

- *result* - This dictionary is used for holding the "metadata" of a given model and is then emptied in the Model Evaluation stage (see **Listing 4.8**) before being used in the next model. The keys are named after the data they hold: 'Model' (name of the model), 'Training Time' (time to fit the model), 'Pred Time' (time to predict the closing values in the test set), 'MAPE' (MAPE score), 'MSE' (MSE score), and 'R2' (R^2 score).
- *allCrypto* - A list used for storing the values of *result* before it is emptied, thus, holding the metadata of all four models for all five coins being analysed.
- *y_plot_pred_vals* - It is a nested dictionary which holds the values predicted by all four models in the testing set for the five coins being analysed. The nested dictionary is initialised (and cleared) at the start each for loop iteration (see **Listing 4.1**)

With the pertinent lists and dictionaries created, the program begins to execute each model sequentially: SVR, LSTM, GRU and CNN. The hyperparameters of the models are predefined and remain constant throughout run time to maintain a uniform program architecture for each coin being analysed.

Space Vector Regression

The SVR model is most simple of all models to implement, requiring just a few lines of Python code. The hyperparameters are not set manually, relying instead on the default values assigned by the sklearn library to reduce complexity and that the documentation states the default kernel type is RBF [**90**], which has already shown positive results against LSTM in [**58**].

```
result['Model'] = 'SVR'
model1 = SVR()
start_time = time.time()
model1.fit(X_train, y_train)
result['Training Time'] = (time.time())-start_time
testModel(model1)
y_plot_predvals[f'{crypto}']['SVR'] = y_pred
```

Listing 4.3: Model Implementation stage (SVR model) [7]

The following three DL models are based from [**91**] and require the feature space to be reshaped into a 3-Dimensional matrix where the first dimension represents the number of samples used, and since all values in the sets are used, the number of samples (or rows) of each set is passed. The second dimension represents the number of time steps, which is equivalent to the number of days being used to predict the future price, thus, the *window* parameter is passed. The third dimension represents the number of features being use to predict the next day closing price and since all features (i.e. 8) are used, the total number of columns (i.e. 8) is passed.

```
X_train = np.reshape(X_train, (X_train.shape[0], window, X_train.shape[1]))
X_test = np.reshape(X_test, (X_test.shape[0], window, X_test.shape[1]))
```

Listing 4.4: Train & testing set reshape [7]

Long Short-Term Memory

A Sequential layer is used followed by an InputLayer, which has a shape tuple passed as an argument, consisting of the number of time steps being used (i.e. *window*) and the number of features [**92**]. The following layer is followed by the LSTM layer whereas the last three ones are Dense layers. In *compile()*, the loss function is set as the *MeanSquaredError()*, and the optimiser is of type 'Adam' with a learning rate of 0.0001 to ensure convergence towards the minimum point. The model fitting is set to 50 epochs as in [**61**] to ensure optimum convergence whilst not taxing the laptop CPU and lengthening too much the execution of the program.

```
result['Model'] = 'LSTM'
model2 = Sequential()
model2.add(InputLayer((window, 6)))
model2.add(LSTM(64))
model2.add(Dense(32, activation = 'relu'))
model2.add(Dense(32, activation  = 'relu'))
model2.add(Dense(1))
print(f"Summary of model is:\n {model2.summary()}\n")
model2.compile(loss = MeanSquaredError(),
                optimizer = Adam(learning_rate = 0.0001))
start_time = time.time()
model2.fit(X_train, y_train, epochs = 50)
result['Training Time'] = (time.time())-start_time
testModel(model2)
y_plot_predvals[f'{crypto}']['LSTM'] = y_pred
```

Listing 4.5: Model Implementation stage (LSTM model) [7]

model2.summary() prints the number of parameters developed for model2, showing that LSTM is the architecture with the most parameters of all the DL models trained:

Table 4.3: LSTM Parameters

Layer (type)	Output Shape	Param #
LSTM	(None, 64)	18176
Dense	(None, 32)	2080
Dense	(None, 32)	1056
Dense	(None, 1)	33
Total params: 21,345		
Trainable params: 21,345		
Non-trainable params: 0		

Gated Recurrent Unit

The GRU model is almost identical as the one employed in the LSTM model with the only difference being that the LSTM layer is replaced instead with a GRU layer.

```
result['Model'] = 'GRU'
model3 = Sequential()
model3.add(InputLayer((window, 6)))
model3.add(GRU(64))
model3.add(Dense(32, activation = 'relu'))
model3.add(Dense(32, activation = 'relu'))
model3.add(Dense(1))
print(f"Summary of model is:\n {model3.summary()}\n")
model3.compile(loss = MeanSquaredError(),
                optimizer = Adam(learning_rate = 0.0001))
start_time = time.time()
model3.fit(X_train, y_train, epochs = 50)
result['Training Time'] = (time.time())-start_time
testModel(model3)
y_plot_predvals[f'{crypto}']['GRU'] = y_pred
```

Listing 4.6: Model Implementation stage (GRU model) [7]

Table 4.4 prints the number of parameters developed for model3, which are shown to be considerably reduced compared to LSTM. This is to be expected given the reduced complexity of GRU as stated in **[54]**.

Table 4.4: GRU Model Parameters

Layer (type)	Output Shape	Param #
GRU	(None, 64)	13824
Dense	(None, 32)	2080
Dense	(None, 32)	1056
Dense	(None, 1)	33
Total params: 16,993		
Trainable params: 16,993		
Non-trainable params: 0		

Convolutional Neural Network

The architecture is similar to that of LSTM and GRU as well. However, after the InputLayer, a Conv1D layer is introduced followed by two dense layers (the first one with a 'relu' optimiser and the last one with a 'linear' optimiser). The *compile()* has the same parameters as in the previous two models, and hte number of epochs in *fit()* is also set to 50 for the same reasons as described for LSTM.

```
result['Model'] = 'CNN'
model4 = Sequential()
model4.add(InputLayer((window, 6)))
model4.add(Conv1D((64, kernel_size = 1)))
model4.add(Flatten())
model4.add(Dense(8, activation = 'relu'))
model4.add(Dense(1, activation = 'linear'))
print(f"Summary of model is:\n {model4.summary}()\n")
model4.compile(loss = MeanSquaredError(),
                optimizer = Adam(learning_rate = 0.0001))
start_time = time.time()
model4.fit(X_train, y_train, epochs = 50)
result['Training Time'] = (time.time())-start_time
testModel(model4)
y_plot_predvals[f'{crypto}']['GRU'] = y_pred
```

Listing 4.7: Model Implementation stage (CNN model) [7]

As shown in **Table 4.5**, this is the model with least parameters from the three DL models, using only a total of 977 parameters. This minimum amount of parameters whilst retaining a similar architecture and hyperparameters as LSTM might explain how CNN is able to outperform it in both **[65]** and **[66]**.

Table 4.5: CNN Model Parameters

Layer (type)	Output Shape	Param #
Conv1D	(None, 1, 64)	448
Flatten	(None, 64)	0
Dense	(None, 8)	520
Dense	(None, 1)	9
Total params: 977		
Trainable params: 977		
Non-trainable params: 0		

4.2.6 Model Evaluation

Each model calls upon a user-defined function *testmodel()* in which the *sy_array* is created to hold the predicted values for a given model being tested. Since the values predicted are still scaled, the values in the array must be unscaled by first reshaping the array into a column vector and performing the inverse transform.

```
def testModel(model):

    start = time.time()
    sy_pred = model.predict(X_test)
    result['Pred Time'] = (time.time())-start

    y_pred = scalery.inverse_transform(sy_pred.reshape(-1,1))

    evaluate_model()

    result['Crypto'] = crypto
    allCrypto.append(result)
    result = {}
```

Listing 4.8: Model Evaluation stage - testmodel() [7]

Once the values are unscaled, the user-defined function *evaluate_model()* is called to calculate the MAPE, MSE and R^2 scores between *y_pred* and *uy_test*. The results are stored in their corresponding key within the *result* list, which is then appended to the *allCrypto* dictionary before the list is emptied. *allCrypto* is then saved in the *df* dataframe to be stored as the CSV file *Metrics_Result.csv* in the CWD.

In the last cells of the notebook, the graphs and charts for the unmodified original datasets, the model predictions, and the evaluation metrics are drawn.

The development tools used to develop *CincoCrypto* have been discussed, and a high-level outline of the program has been provided before delving deeper into the specifics of each stage of the program. Therefore, the results obtained during run time can be now discussed in the next chapter.

Chapter 5

Results

The structure of *CincoCrypto* has been now laid out and the development tools used to develop it have also been briefly described. In this chapter, the metrics results in *Result_Metrics.csv* (see **Appendix B**) obtained during one run are analysed and compared by means of graphical representations.

5.1 Evaluation Metrics Description

Throughout this project, five evaluation metrics are used to measure the performance of each model for all the datasets being analysed. These metrics are widely popular in academic literature related ML and cryptocurrency price forecasting, being used in the papers mentioned in **Chapter 3**. A key parameter when evaluating model performance is the error (E) or amount of difference between the observed or predicted value (P) and the actual value (A):

$$E = A - P \tag{5.1}$$

5.1.1 Training Time

The training time is used to measure model training time and it is done through the *time()* function by invoking it before *train()* and storing the time in *start_time*. Immediately after *train()*, the *time()* function is invoked again and the time difference between this call and *start_time* is calculated, and the training time stored in *result['Training Time']*.

```
1  start_time = time.time()
2  model1.fit(X_train, y_train)
3  result['Training Time'] = (time.time())-start_time
```

Listing 5.1: model1 fitting time calculation [7]

5.1.2 Mean Absolute Percentage Error

As explained in [93], MAPE is a measure of the extent of the error in terms of percentages, being usually applied as a loss function. Furthermore, MAPE cannot be used on data with actual zeroes, has no upper-limit for high predictions, and is non-symmetrical (it is negatively affected if the difference between predicted and real value is too high). The main advantages of MAPE are, first, the use of absolute error to prevent positive and negative errors from adversely affecting the summation, and, second, the independence between relative errors and the scale of the dependent variable allow MAPE to compare time series with different scales) [94]. According to [93], the formula for MAPE is given as:

$$MAPE = \frac{100}{n} \sum_{n}^{i=1} \frac{|E_i|}{|A_i|} \tag{5.2}$$

5.1.3 Mean Squared Error

MSE is a measure of the average square of the errors [93]. This metric is dependent on the scale and amount of data used as well as being heavily affected by outliers. The formula for MSE is given as:

$$MSE = \frac{\sum_{i=1}^{n} {E_i}^2}{n} \tag{5.3}$$

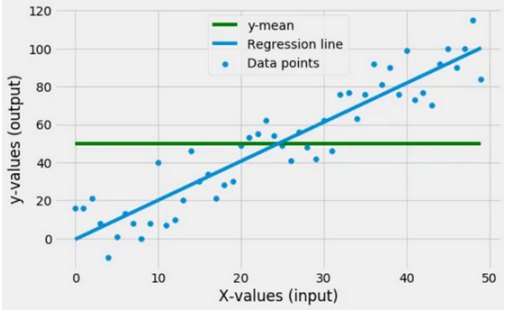
Coefficient of Determination

R^2 is a statistic measure, usually given as a percentage, describing the amount of variation (i.e. movement) in the independent variable due to an index or dependent variable. A high R^2 lies between 70% - 100% and shows that the movements in the dependent variable are completely determined by movements in the independent variable. A low R^2 (70% or less) shows that movements of the dependent variable are not generally related to the independent variable. It should be noted that what constitutes as a high or low R^2 is relative to the context in which the metric is being used. In Finance, a value greater than 0.7 is considered as a high correlation whereas a value below 0.4 is considered a low correlation; this criteria is used when evaluating the R^2 scores of the models. Furthermore, a high value of R^2 does not necessarily mean that the model is good or bad, it depends again on the context [95], but it can be used as measure of goodness of the model fit. Should there be more than one dependent variable, the *Adjusted* R^2 should be used instead [96].

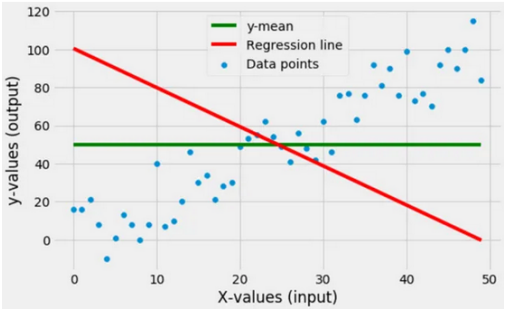
According to [96] and [93], the R^2 formula is given as:

$$R^2 = 1 - \frac{SSR}{SST} = 1 - \frac{\sum_{i=1}^{n}(P_i - A_i)^2}{\sum_{i=1}^{n}(A_i - A_{mean})^2} \tag{5.4}$$

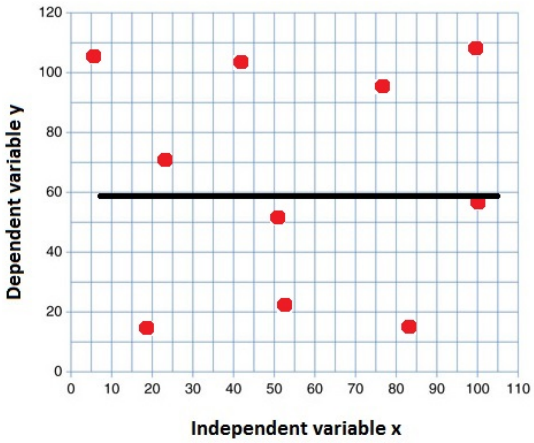
In [97], it is explained the possibility for the actual value of R^2 to be negative and outside $[0, 1]$. In the context of regression, this can be understood as a regression line which does not fit (i.e. underfitting) any of the points in the dataset, thus, the independent variable accounts for no variation in the dependable variable at all.



(a) Regression line for positive R^2 [97].



(b) Regression line for negative R^2 [97].



(c) Regression line for zero R^2 [96].

Figure 5.1: Regression line for positive, zero, and negative R^2

5.2 Analysis

Having explained the metrics used for assessing model performance, the results obtained from **Appendix B** and their corresponding graphs can be now examined to determine any overall patterns in model efficiency.

5.2.1 Bitcoin

Prediction Plot

From Figure 5.2a, it can be seen that the GRU model excels at predicting the closing price compared to the other models, following upward and downward trends very accurately in a consistent manner. The LSTM model follows next with a graph with the same characteristics as the GRU model but with a slight offset which makes its predictions lower than GRU by an almost negligible amount. The CNN model is the third closest model to predict the closing value as it follows the closing price shape very closely but with a greater offset and less capable of following gradual price changes (i.e. less steep gradient

and less accurate predictions around 09-03-2023 and 08-04-2023), making it less accurate than the LSTM and GRU models. The SVR model can be seen to be the worst by far of all models; it draws a curve which tries to follow the shape of the closing price, making predictions which are generally lower than the other graphs and more pronounced peaks when trying to predict price spikes and drops.

Training Time

In **Figure 5.2b** The SVR model is shown to be the fastest by large to train compared to the the other models (0.0020) which makes sense given the SVR is a lighter model, requiring no layers and individual weight optimisation in contrast to DL models. The second fastest model is CNN (2.1557) followed by LSTM (5.3043 seconds). Lastly, GRU is the slowest model with 7.3346 seconds.

Mean Absolute Percentage Error

As expected after looking at **Figure 5.2a**, the bar chart at **Figure 5.2c** confirms that GRU is the model with the least error (0.0263) followed closely by LSTM (0.0294). CNN also comes in closely as the third model (0.0329) whilst the largest MAPE is the one from SVR (0.0505).

Mean Squared Error

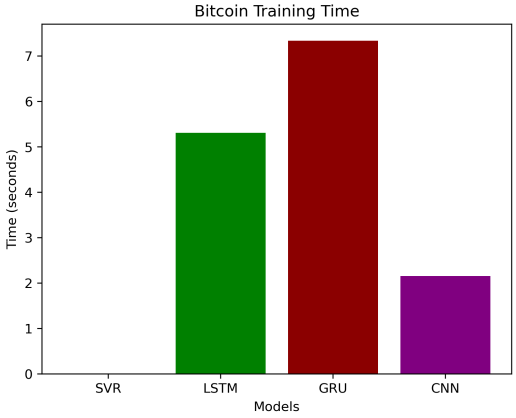
Similarly to the previous metric discussed, **Figure 5.2d** follows the same pattern given that GRU is shown to produce the least squared error (777,953.8180) followed by LSTM (944,782.5172), CNN (1,124,888.5063) and SVR (2,459,684.3586).

Coefficient of Determination

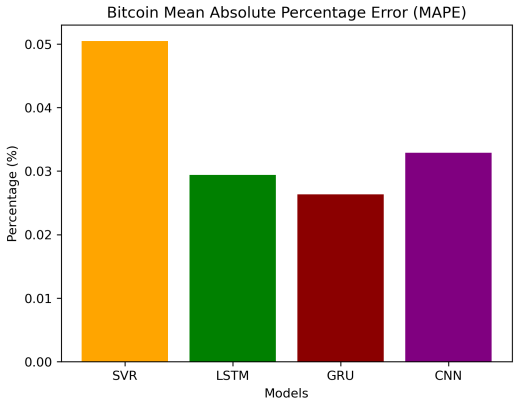
Figure 5.2e shows high correlation for all that all three models, which makes sense given that the three of them follow the closing price very closely as seen in Figure 5.2a. GRU shows the strongest correlation (0.8898) followed by LSTM (0.8589), CNN (0.8302). On the other hand, SVR (0.6743) has a weak correlation which is to be expected given its graph has more pronounced peaks and drops and its prediction is on average lower than the other models.

Summary

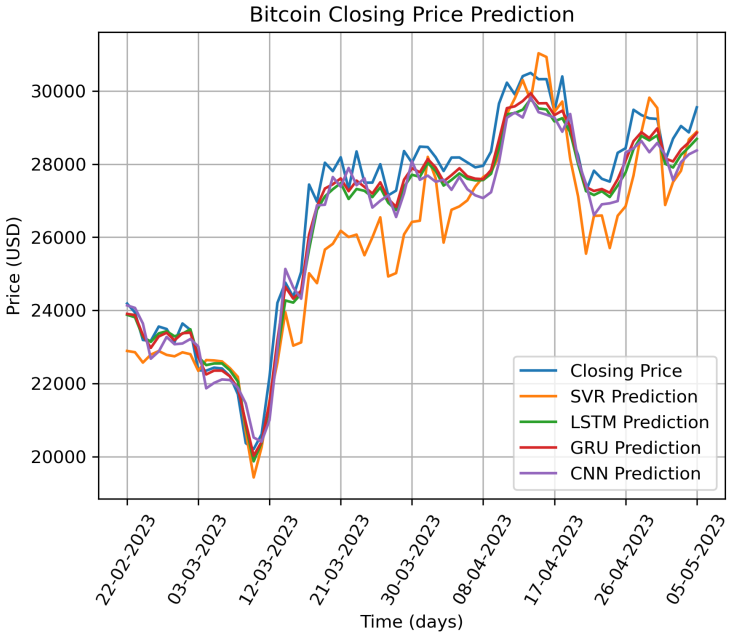
The prediction graphs at and the metrics bar charts in **Figure 5.2** show that for the BTC dataset, GRU is the best performing model but it is also the slowest one to be trained. The second best performing model is LSTM, which has the third slowest training time, followed by CNN (second slowest), and SVR, which is the least performing model but also the fastest one.



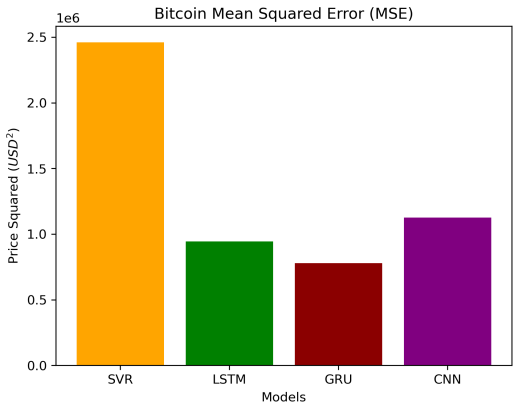
(b) BTC Training Time



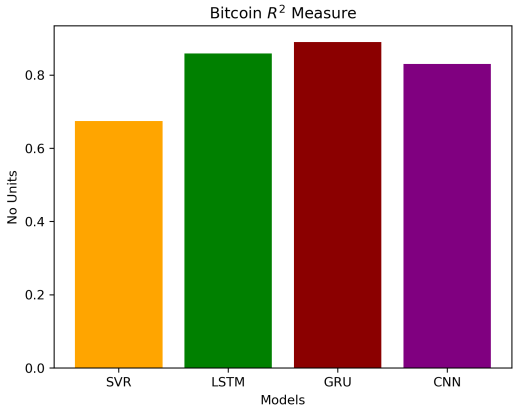
(c) BTC dataset



(a) BTC closing price prediction



(d) BTC MSE



(e) BTC R^2

Figure 5.2: BTC charts

5.2.2 Ether

Prediction Plot

In **Figure 5.3a**, it can be seen that SVR seems to predict the closest to the closing price albeit the prediction graph seems to be slightly skewed towards the right, having little to no accuracy when the closing price experiences spikes and drops. On the other hand, the DL models seems to be better in general at following the upward and downward trends, thus, having a shape which resembles more closely that from the closing value. The CNN is the best of the DL models at predicting the closing value, followed by the GRU and LSTM models. All models are able to predict the lowest closing fairly accurately, but after that point, the gap between the DL models and the closing value increase significantly.

Training Time

As already seen before, **Figure 5.3b** the training training time follows the same trend in which SVR model keeps being the fastest model to be trained (0.0017). This is followed by CNN (1.9289) and LSTM (4.8639). The slowest model, by a marginal amount, is GRU (4.9420).

Mean Absolute Percentage Error

In **Figure 5.3c**, it can be seen that SVR model is now the best model with the least error (0.0278) followed by CNN (0.0359), GRU (0.0423), and LSTM (0.0524).

Mean Squared Error

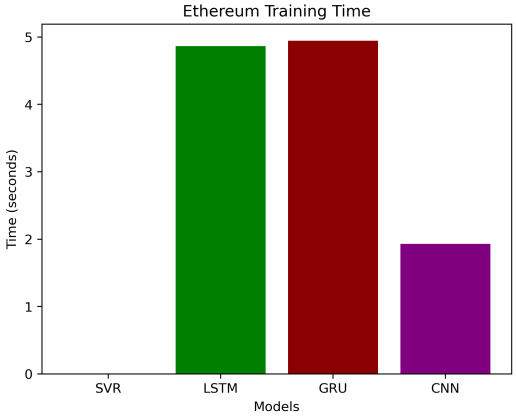
Similarly to MAPE, the models follow a same fashion where SVR has the least error (4140.3413) as shown in **Figure 5.3d**. The next models also fit the pattern where CNN has 5822.1763, GRU obtains 7912.2718, and LSTM has the greatest a square error of 11346.1712.

Coefficient of Determination

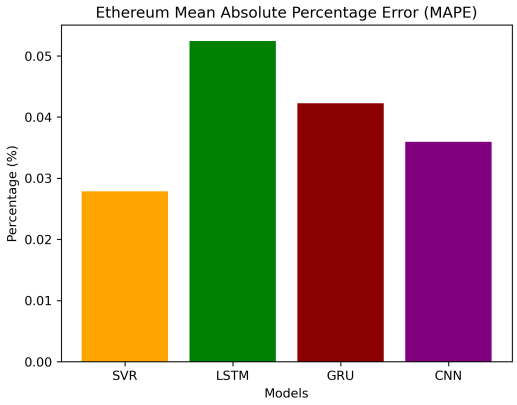
Figure 5.3e shows that the LSTM model (0.1238) is the one out of the four with the least correlation which is not unexpected given its predictions have a large offset as seen in **Figure 5.2a**. Next, GRU has a score (0.4641) and is followed by CNN (0.6739). The model with the strongest correlation is shown to be SVR (0.8004) whose graph intersects the most with the closing price.

Summary

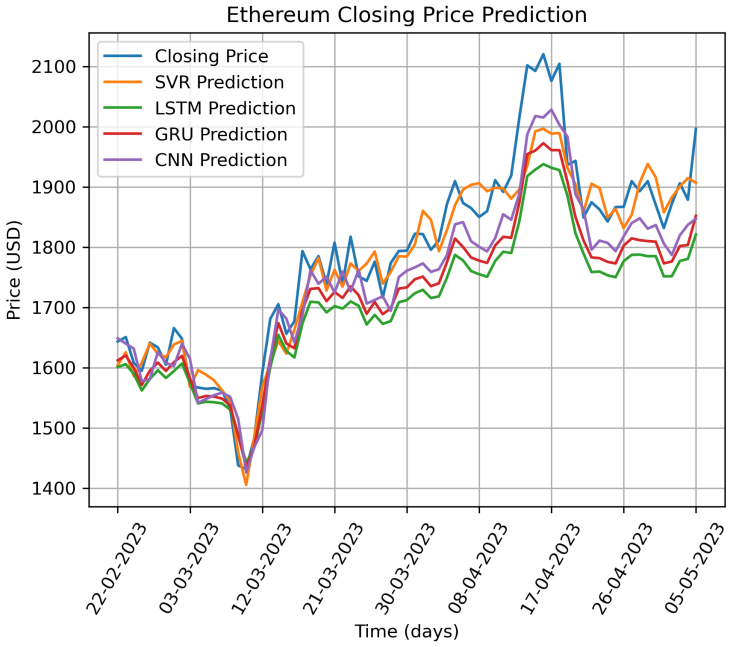
As expected from having seen **Figure 5.3**, SVR is the best performing model out of all four as not only does it have the least MSE and MAPE but it also shows a strong correlation. The model has the added advantage of being the fastest. The second best performing model is CNN, which has the second slowest training time, followed by GRU, which happens to the slowest model. LSTM, is the least performing model and third slowest one.



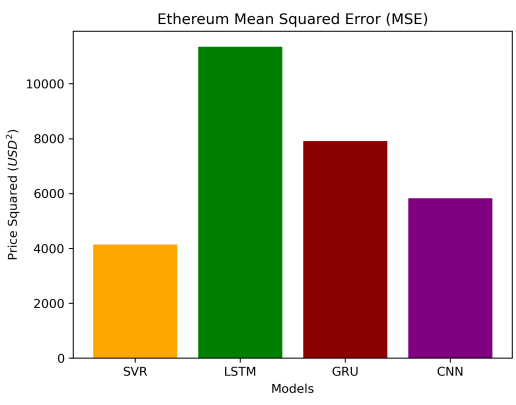
(b) ETH training time



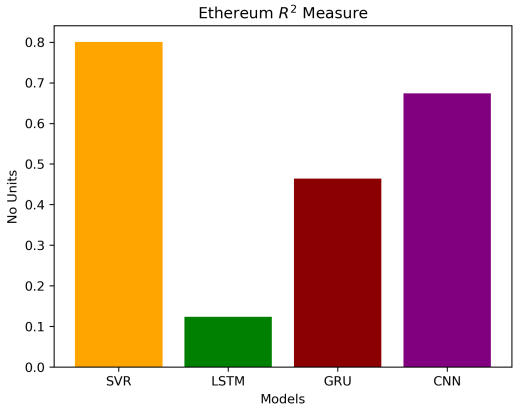
(c) ETH MAPE



(a) ETH closing price prediction



(d) ETH MSE



(e) ETH R^2

Figure 5.3: ETH charts

5.2.3 Tether

Prediction Plot

In **Figure 5.4a**, it can be seen that CNN seems to predict the closest to the closing price, being the best at following the rising and falling trends better and, thus, having a shape which is similar to that of the closing price. The model seems not to be able to correctly follow prices which are stable for a few days after there has been significant rises and falls (i.e. 11-03-2023 and 16-03-2023). Next, the GRU is shown to be the second best performing model alas with a heavy dampening at the steepest closing values from the 08-03-2023 up to 26-03-2023, and it seems not to be able to predict correctly the minor price drop on the 06-04-2023 nor the subsequent price spikes. This behaviour is also seen in the LSTM model but this model predicts values much lower than GRU. Lastly, the SVR model seems to make the worst predictions, fluctuating around the average value of 1.000 USD and being unable to correctly predict the price increase from the 08-03-2023 up to 26-03-2023.

Training Time

As already seen before, Figure 5.4b the training training time follows the same trend in which SVR model keeps being the fastest model to be trained (0.0008). This is followed by CNN (3.1143) and GRU (4.2563). The slowest model, by a marginal amount, is LSTM (4.2943).

Mean Absolute Percentage Error

Figure 5.4c shows CNN having the lowest error (0.0005) followed by GRU (0.0009), LSTM (0.0009) and SVR (0.0010).

Mean Squared Error

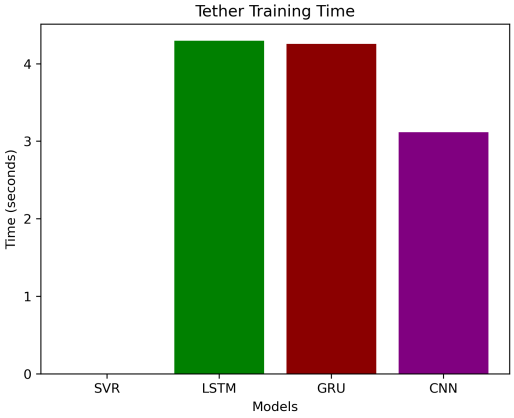
Similarly to MAPE, the models follow a same fashion in **Figure 5.4d** where CNN ($1.2948e^{-06}$) has the lowest squared error followed by GRU ($2.5437e^{-06}$), LSTM ($2.6714e^{-06}$) and SVR ($3.4462e^{-06}$).

Coefficient of Determination

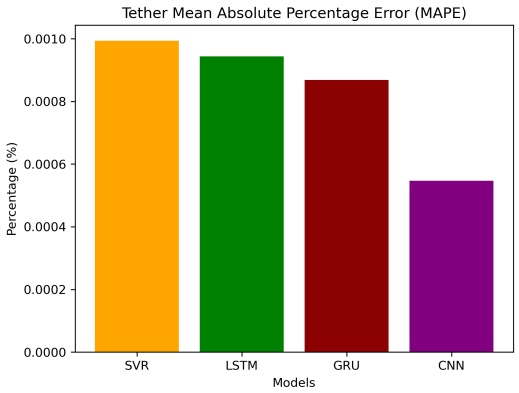
Figure 5.4e shows that all models are negative and with a magnitude greater than 1. Since USDT is a stablecoin, it means that its average value closing price is always at 1 USD to ensure a 1:1 exchange, thus, most of the data is around 1.000 USD and has little spread over the *Price* axis. By looking at Equation (5.4), it can be seen that for the fractional component of R^2 to be less than 1, SST must be greater than SSR, and for this to occur, the actual closing price values must be larger than actual mean. However, this is not possible as most of the closing price values fluctuate around 1.000 USD and, hence, SST will always have a value close to zero for a stablecoin, causing the fractional component to be extremely large and, consequently, make the value of R^2 to be large and negative.

Summary

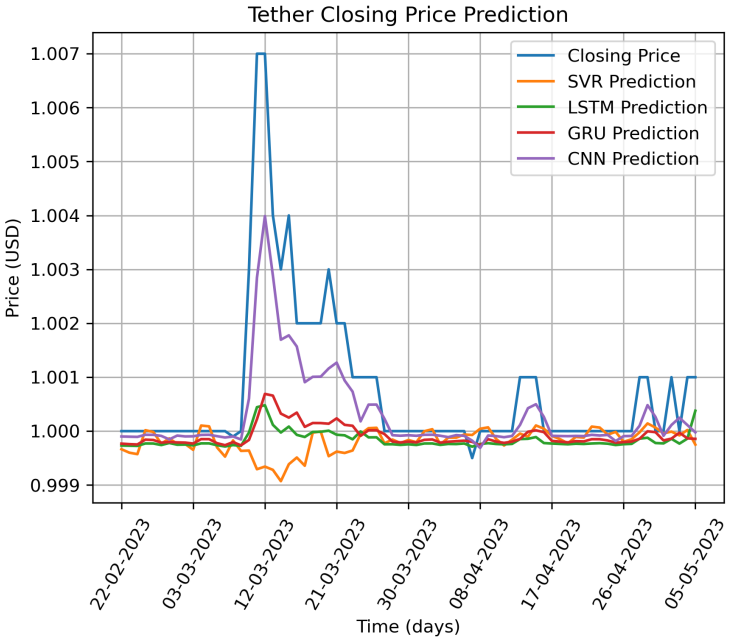
As expected from having seen Figure 5.4, CNN is the best performing model as it has the least MSE and MAPE as well as being the second fastest model to train. In addition, CNN is shown the be the best model at predicting the close price spikes but it is quite slow in doing so as the predictions are always made a day later, usually, when the real closing value starts to go back down, showing that model's response is lagging in the face of upward and downward trends. The next best performing models are GRU and LSTM where both models were the third and fourth fastest models, respectively. Both models show similar performance with each other and seem to also lag when predicting upward and downward trends as with CNN. However, both models are shown to make significantly lower price predictions with values not fluctuating too far away from the average (1 USD). The least performing and slowest model is SVR which predicts the highest closing price on the 12-03-2023 as a price drop instead. In general, the DL models seem to resemble the closing price shape better than SVR, which predicts a drop rather than a spike at strong price spikes.



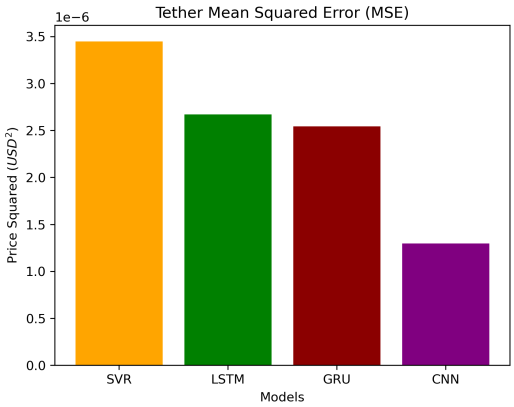
(b) USDT training time



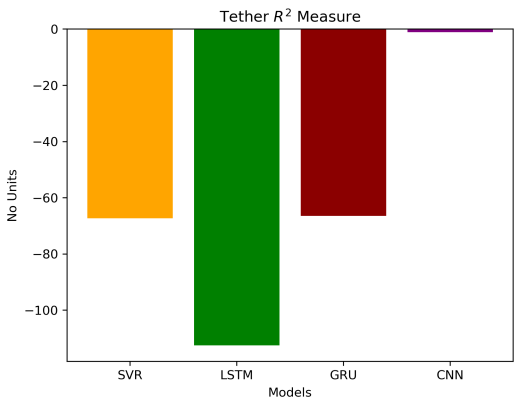
(c) USDT MAPE



(a) USDT closing price prediction



(d) USDT MSE



(e) USDT R^2

Figure 5.4: USDT charts

5.2.4 Binance Coin

Prediction Plot

All four models seem to be able to follow the closing price curve to a high degree, being to predict the rising and falling trends in BNB (see **Figure 5.5a**). They all predict accurately the minimum closing point at 10-03-2033.

The SVR model seems to make the best predictions as it is the model which crosses the closing price curve the most whilst remaining close to it. It is at its closest just before minimum closing price.

The second model which makes the best predictions appears to be CNN as it makes the makes closest predictions given that it makes predictions very similar, if not the same at times, to those made by SVR. However, unlike SVR, the CNN model is shown to better at predicting major price spikes (e.g. 19-03-2023, 17-04-2023 and 25-04-2023).

Next comes the GRU model which draws a curve with values very consistent with those of the CNN model but with a less defined shape when it comes to a large succession of spikes and drops as shown in the region between 23-04-2023 and 27-04-2023.

Lastly, the LSTM model makes the lowest price predictions as seen in the large gap between its predictions the actual closing price.

Training Time

SVR model is the fastest model to be trained (0.0030) as seen in **Figure 5.5b**. This is followed by CNN (3.1850) and LSTM (4.4052). The slowest model is GRU (6.9808).

Mean Absolute Percentage Error

Figure 5.5c shows SVR (0.0265) and CNN (0.0267) having very similar MAPEs. Next the GRU model has an error value 0.0297, and is followed by the LSTM (0.0511).

Mean Squared Error

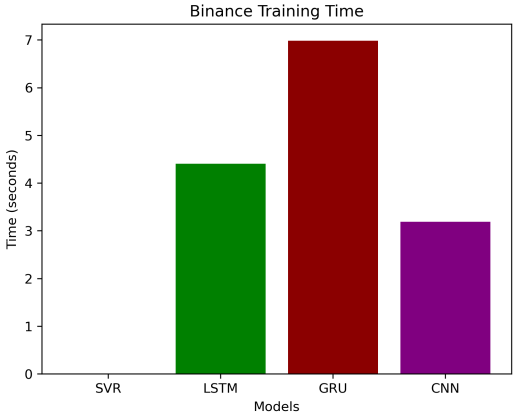
Similarly to MAPE, the models follow a same fashion where SVR (110.1347) has the lowest squared error followed by CNN (113.1334) GRU (136.0487), and LSTM (316.1564).

Coefficient of Determination

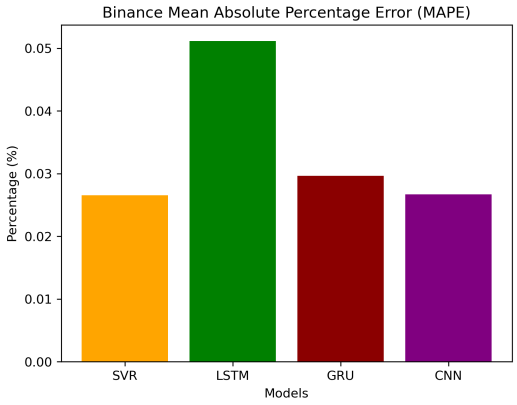
Figure 5.5e shows all R^2 values are negative except for the CNN model (0.1989). It makes sense that the CNN model has the high score (0.1989) but so should the SVR (-0.0894) given that both graphs are show similar curves, which have similar shape characteristics to the closing price curve but predict lower values. Therefore, in this case, the R^2 shows that the models are underfitted and require more training time compared to the previous coins analysed.

Summary

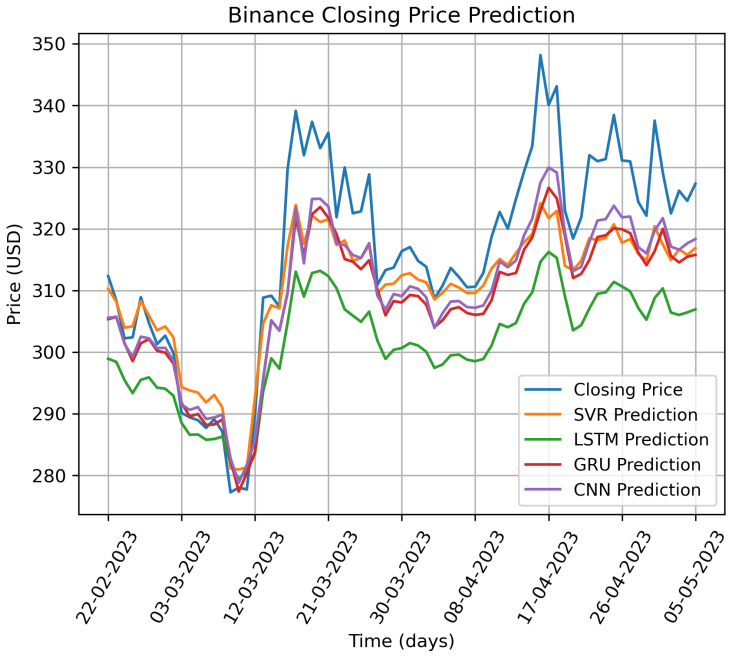
From the metrics analysed it can be seen that SVR performed the best and was followed closely by CNN, both happening to be the fastest and second fastest models respectively. GRU shows a similar performance with little difference in its error metrics compared the two best performing models. LSTM on the other hand shows the least accuracy in all models as shown in **Figure 5.5a** by the wide gap between its predictions and the closing price and as demonstrated in the metrics charts. All four models produce prediction curves which have similar shape to that of the closing value. Nonetheless, it can be appreciated that predictions curves suffer from underfitting as shown as well in **Figure 5.5e,** however, it is believed that at least SVR could have a positive value as it has a similar performance as CNN, which has the only positive score of R^2 score of 0.1989.



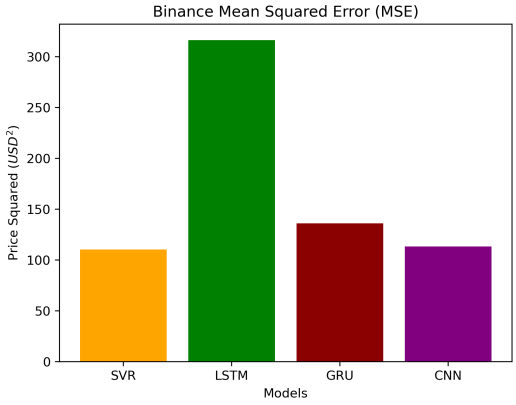
(b) BNB training time



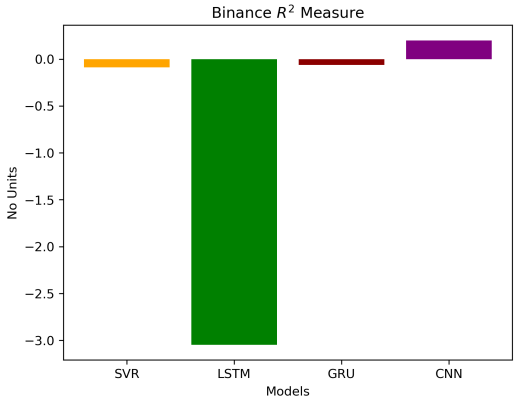
(c) BNB MAPE



(a) BNB closing price prediction



(d) BNB MSE



(e) BNB R^2

Figure 5.5: BNB charts

5.2.5 USD Coin

Prediction Plot

It is clear from **Figure 5.6a** that the closing values of USDC have an average of 1.000 USD from which they almost do not deviate at all given that USDC is a stablecoin. All the DL models correctly predict that the minimum closing price on the 11-03-2023, being able to follow the falling and rising trends correctly. However, the DL models present a substantial amount of noise which accentuates at the corner between 09-03-2023 and 17-03-2023.

The SVR model is shown to remain at the average value for the whole length of the testing set without showing any noise or fluctuations even when making predictions in the minimum closing price region.

Out of the three DL models, CNN shows the best performance by being able to predict the minimum closing value with high accuracy from the beginning of the price fall until its return back to stability.

GRU is also successful in predicting that there is a minimum closing price at 11-03-2023 albeit its prediction is too short and when it tries to return to stability, it crosses closing price curve at 13-03-2023 until becoming stable at around 22-03-2023.

LSTM shows a similar behaviour to GRU but predicts a smaller closing price at 11-03-2023 and rises above GRU when crossing the price curve at 11-03-2023.

Training Time

As already seen in all coins, SVR model is the fastest model to be trained (0.0007) from what it is shown in **Figure 5.6b**. This is followed by CNN (2.1030) and LSTM (4.3682). The slowest model is GRU (4.4679).

Mean Absolute Percentage Error

GRU has the least error (0.0009) with LSTM (0.0010) and SVR (0.0010) following as second and third most accurate model, respectively (see **Figure 5.6c**). In addition, CNN is the model which has the greatest MAPE by far with a value of 0.0267.

Mean Squared Error

Similarly to MAPE, the values obtained for MSE in **Figure 5.6d** follow the same pattern where GRU (1.5646^{-05}) has the least error followed by SVR ($1.5902e^{-05}$), LSTM ($1.6093e^{-05}$), and CNN ($2.3330e^{-05}$).

Coefficient of Determination

As explained above for USDT, USDC is a stablecoin whose average value aims to be at 1.000 USD to ensure 1:1 exchange, therefore, all actual values are close to the mean, making the SST tobe close to zero, resulting in large, negative R^2 scores as seen in **Figure 5.6e**. However, in the case of SVR, the score does show that the model is underfitting since the prediction stays at 1.000 throughout the test, being unable to predict the price drop at 11-03-2023 (see Figure 5.6a) and having a excessively large ($-3.2252e^{026}$) to be printed

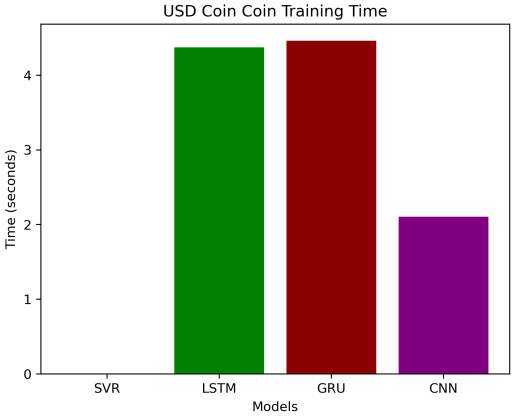
alongside the other models in Figure 5.6e. In addition, it is worth noting that CNN has the least negative score of R^2, being explained by the fact that it is able to correctly predict the price drop at 12-03-2023, matching the downward and upward trends better than the other models, thus, showing a greater degree of fit than the other three models. In any case, the R^2 score should be used with caution when dealing with datasets with a low variance (i.e. stablecoins) as the values tend to gather close to the mean value, causing the score to become large and negative.

Summary

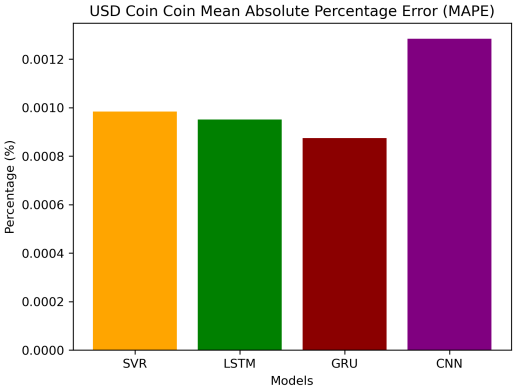
Despite what it can be seen in the metrics from Figure 5.6, it can be said that CNN is the best prediction model for USDC given it that it predicts the closing value accurately and follows the the upward and downward trend of the coin more accurately than the other models, achieving the most positive R^2 score and being the second fastest model overall.

The other DL models have similar performances, being GRU the second best performing model overall as well as the second fastest model to train. The model shows the least fluctuations in the stable regions, leading to it having the least MAPE and MSE scores out of the four models. LSTM also makes predictions close to those made by GRU albeit more small.

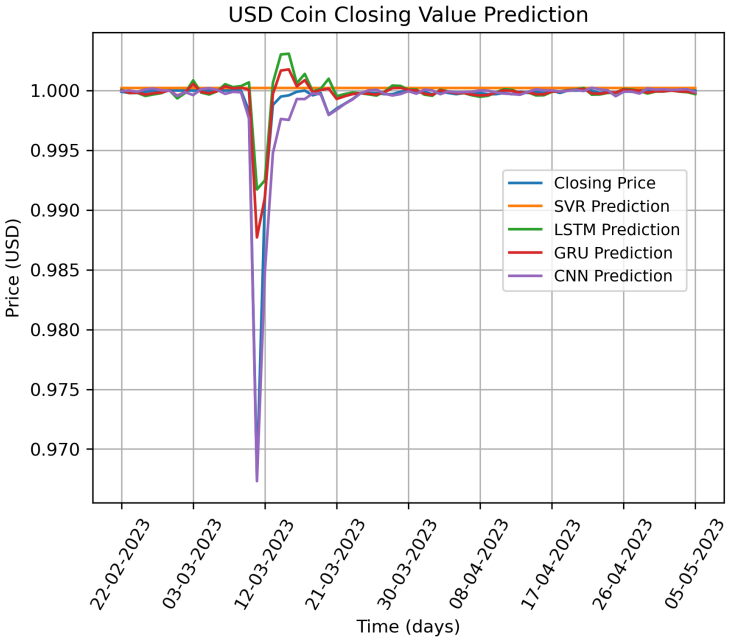
On the other hand, SVR is the least performing model and the fastest one to train. The model does not attempt to predict the minimum closing price, staying flat at the average value for the whole test.



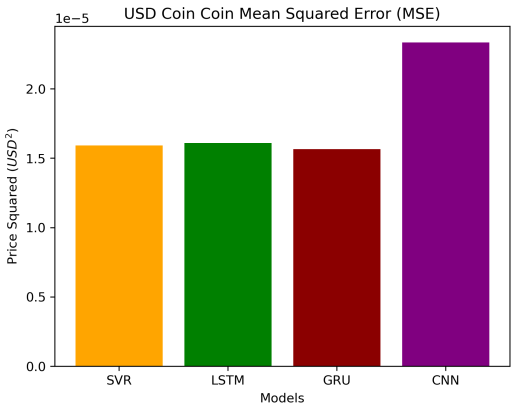
(b) USDC training time



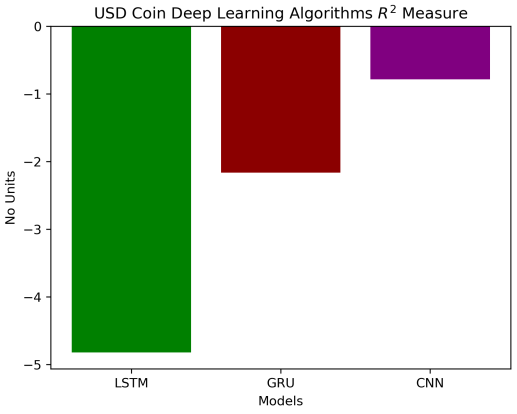
(c) USDC MAPE



(a) USDC closing price prediction



(d) USDC MSE



(e) USDC R^2 (DL models only)

Figure 5.6: USDC charts

5.2.6 Evaluation

Having described the results from the models for each coin individually, a series of highlights and patterns can be made.

BTC is the coin with the highest R^2 scores with prediction curves being close to fit the data. However, it can be appreciated that all models underfit for the coins being analysed, and they would all benefit from longer training times to improve their accuracy. Even so, **Table 5.1** shows that for ETH, *CincoCrypto* is capable of producing better predictions with smaller MAPE values than the multi-forecast horizon study proposed in [**98**] and with similar R^2 values which could possibly be improved should the training time be increased increase.

It is shown in **Figure 5.4** that USDT is the fastest dataset to be tested, and, it presents, overall, the least MAPE and MSE. However, it can be seen that USDT presents the most negative R^2 scores from all five coins despite CNN being able to follow the shape of the closing curve very accurately. This shows that R^2 should be used with caution, in particular, when dealing with stablecoins as their SST is too large, making the R^2 score be negative for models predicting on this type of cryptocurrency.

In general, it is shown that SVR model is the fastest model to train due to having less complexity (no layers and weight optimisation) as in the DL models. Nonetheless, the model greatly underperforms when analysing stablecoins as seen in **Figure 5.4** and **Figure 5.6**. Furthermore, the DL models are shown to best at predicting the closing price as their graphs as the ones which resemble the closing curves the most. In three out of five coins (i.e. ETH, USDT, BNB), the CNN model is shown to be the best performing DL algorithm, making predictions which can correctly determine when a strong price spike will happen, specially, for stablecoins (i.e. **Figure 5.4** and **Figure 5.6**). LSTM and GRU models tend to have similar performances with one another (except with USDT), but for in the other four coins, GRU is the best performing of the two models. Nonetheless, it is suprising to see that GRU is slower to train than LSTM when, unlike LSTM (**Table 4.3**), GRU has significantly less parameters (see **Table 4.4**) to be trained, making one to assume it would cause a drastic drop in training time as in the case of CNN, which is the DL model with the least parameters (see **Table 4.5**).

Table 5.1: ETH results comparison

(a) *CincoCrypto* results

Models	MAPE	R^2
SVR	0.0278	0.8004
LSTM	0.0524	0.1238
GRU	0.0423	0.4641
CNN	0.0359	0.6739

(b) [98] results

Models	MAPE	R^2
SVR	6.37	0.876
LSTM	9.35	0.789
GRU	7.73	0.843
CNN	5.45	0.894

The metrics used for evaluating the performance of the models have been described and the results obtained for each coin have been assessed. The overall evaluation of the model has described the main highlights and patterns observed for the coins and models before moving onto the final chapters of the report.

Chapter 6

Conclusion

This report introduces *CincoCrypto*, a new free, open-source cyrptocurrency price forecasting tool built with simplicity on mind to allow people with little to no financial and/or technical background to make safe investments in the cryptocurrency market. At a minimum, the user is only required to write the dataset names in the *cryptos* list and run the program. The program has been uploaded to GitHub, ready to be used, and free of charge so that it can reach all audiences [**7**]. Only five cryptocurrency datasets are taken from CryptoCompare to forecast the next day closing price using four models (i.e. SVR, LSTM, GRU and CNN) from which the user can choose the best performing one for the coins whose prices are to be predicted. In this manner, the author hopes to curve the alarming rise of scams in cryptocurrency investment. To comprehend the design criteria and the significance of the results, a thorough description is done on the background behind the key concepts and algorithms, being immediately followed by a literature review of the most relevant peer-reviewed papers relevant for the design of the program. The program is then described and its evaluation metrics discussed and assessed. The results show that the models underfit for all coins being analysed yet *CincoCrypto* is shown to be capable of producing good results (very low MAPE values) and somewhat decent R^2 scores as shown in **Table 5.1**. Overall, it is indicated by the results that SVR is the fastest model whereas CNN tends to outperform the other algorithms in terms of metrics and prediction curve shape.

Chapter 7

Future Work

The results obtained show that the models underfit for the coins being analysed, thus, a thorough analysis would be required by means of loss and variance curves, and an increase in the number of epochs specified for the models. In addition, automatic hyperparameter tuning would be an effective method to increase model performance as the parameters could be automatically selected and applied based on the datasets to be fed into the model as done for SVR in [**57**], but at the cost of greater model complexity which could be more taxing on machines with no dedicated graphical processing units. Furthermore, the program can be made more interactive and functional by allowing the users to upload datasets stored in a directory different from the program CWD as well as select the directory in which to store the results. In addition, the functionalities could be increased by removing the five cryptocurrency datasets from CryptoCompare constraint, placing the user more in control by being able upload as many datasets as they want from other sources and different number of features amongst many other changes which could be made. In terms of increasing closing price prediction accuracy, the use of sentiment and network data as well as external factors as done in [**69**] could be investigated. Furthermore, different price forcasting horizons could be analysed to measure model performance when having to make predictions further ahead into the future (e.g. predicting 5 days into the future). Last but not least, other ML models could be implemented (such as Adaboost or Catboost [**67**]) to allow the users to have more models to choose.

Bibliography

[1] Internet Crime Complaint Center, "Internet Crime Report 2022," Federal Bureau of Investigation, Washington, D.C., USA, report, Mar. 2023. [Online]. Available: https://www.ic3.gov/Media/PDF/AnnualReport/2022_IC3Report.pdf (visited on 05/29/2023).

[2] EUROPOL. "Crypto investment scams - infographic," Medium. (), [Online]. Available: https://www.europol.europa.eu/cms/sites/default/files/documents/EP_Scenario%5C%20Crypto%5C%20Scams%5C%20infographic_EN.pdf (visited on 05/26/2023).

[3] M. Bartoletti, S. Lande, A. Loddo, L. Pompianu, and S. Serusi, "Cryptocurrency scams: Analysis and perspectives," *IEEE Access*, vol. 9, pp. 148 353–148 373, 2021. DOI: 10.1109/ACCESS.2021.3123894.

[4] I. Vakilinia, "Cryptocurrency giveaway scam with youtube live stream," in *2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2022, pp. 0195–0200. DOI: 10.1109/UEMCON54665.2022.9965686.

[5] Santander. "Cryptocurrencies: bait for investment scams," Santander. (), [Online]. Available: https://www.santander.com/en/stories/cryptocurrency-scam (visited on 05/20/2023).

[6] HSBC. "Beware of Cryptocurrency scams," HSBC. (), [Online]. Available: https://www.privatebanking.hsbc.com/about-us/fraud-and-security/beware-of-cryptocurrency-scams/ (visited on 05/12/2023).

[7] *CincoCrypto*, 2023, ARealGent. Accessed: May 31, 2023. [Online]. Available: https://github.com/ARealGent/CincoCrypto.

[8] Binance. "Cryptocurrency Prices Today," Binance. (), [Online]. Available: https://www.binance.com/en/price (visited on 05/20/2023).

[9] CoinmarketCap. "Today's Cryptocurrency Prices by Market Cap," CoinMarketCap. (), [Online]. Available: https://coinmarketcap.com/ (visited on 05/20/2023).

[10] Yahoo! Finance. "Top Cryptos by Market Cap," Yahoo! Finance. (), [Online]. Available: https://finance.yahoo.com/u/yahoo-finance/watchlists/crypto-top-market-cap/ (visited on 05/20/2023).

[11] IRS. "Frequently Asked Questions on Virtual Currency Transactions," IRS. (), [Online]. Available: https://www.irs.gov/individuals/international-taxpayers/frequently-asked-questions-on-virtual-currency-transactions (visited on 05/17/2023).

[12] Santander. "Guide to understanding cryptocurrencies," Santander. (), [Online]. Available: https://www.santander.com/en/stories/guide-to-understanding-cryptocurrencies (visited on 05/17/2023).

[13] Deloitte. "The rise of using cryptocurrency in business," Deloitte. (), [Online]. Available: https://www2.deloitte.com/us/en/pages/audit/articles/corporates-using-crypto.html/%5C#top (visited on 05/17/2023).

[14] Coinbase. "What is cryptocurrency?" Coinbase. (), [Online]. Available: https://www.coinbase.com/es/learn/crypto-basics/what-is-cryptocurrency (visited on 05/17/2023).

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized business review*, pp. 1–9, 2008.

[16] Coinbase. "Double spend," Coinbase. (), [Online]. Available: https://help.coinbase.com/en/coinbase/getting-started/crypto-education/glossary/double-spend- (visited on 05/17/2023).

[17] Santander. "Joint press statement by the CNMV and the Banco de España on "cryptocurrencies" and "initial coin offerings" (ICOs)," Santander. (), [Online]. Available: https://www.bde.es/f/webbde/GAP/Secciones/SalaPrensa/NotasInformativas/18/presbe2018_07en.pdf (visited on 05/29/2023).

[18] Karspersky. "Bitcoin," Karspersky. (), [Online]. Available: https://www.kaspersky.com/resource-center/definitions/what-is-cryptocurrency (visited on 05/17/2023).

[19] CoinMarketCap. "Countries which allow cryptocurrency as legal tender," CoinMarketCap. (), [Online]. Available: https://coinmarketcap.com/legal-tender-countries/ (visited on 05/17/2023).

[20] S. Chipolina, "The two sides of crypto in Ukraine war," Feb. 24, 2023. [Online]. Available: https://www.ft.com/content/a3b59f3b-d0b3-4047-af71-c8ef61aa8d58 (visited on 05/29/2023).

[21] CoinGecko. "Global Cryptocurrency Market Cap Charts," CoinGecko. (), [Online]. Available: https://www.coingecko.com/en/global-charts (visited on 05/20/2023).

[22] Ethereum. "What is Ethereum?" Ethereum. (), [Online]. Available: https://ethereum.org/en/what-is-ethereum/ (visited on 05/20/2023).

[23] Tether. "What are Tether tokens and how do they work?" Tether. (), [Online]. Available: https://tether.to/en/how-it-works/ (visited on 05/20/2023).

[24] Kriptomat. "What is cryptocurrency Tether (USDT) and how does it work?" Kriptomat. (), [Online]. Available: https://kriptomat.io/cryptocurrencies/tether/what-is-tether/ (visited on 05/20/2023).

[25] Binance Research. "BNB (BNB)," Kriptomat. (), [Online]. Available: https://research.binance.com/en/projects/bnb (visited on 05/20/2023).

[26] Circle. "USD Coin," Circle. (), [Online]. Available: https://www.circle.com/en/usdc (visited on 05/20/2023).

[27] B. Marr. "A Short History Of Bitcoin And Crypto Currency Everyone Should Read," Forbes. (), [Online]. Available: https://help.coinbase.com/en/coinbase/getting-started/crypto-education/glossary/double-spend- (visited on 05/17/2023).

[28] CryptoCompare. "Bitcoin," CryptoCompare. (), [Online]. Available: https://www.cryptocompare.com/coins/btc/overview/USD (visited on 05/17/2023).

[29] C. Team. "Is Cryptocurrency a Good Investment?" CFI. (), [Online]. Available: https://corporatefinanceinstitute.com/resources/cryptocurrency/is-cryptocurrency-a-good-investment (visited on 05/17/2023).

[30] CoinMarketCap. "Should You Invest In Crypto?" CoinMarketCap. (), [Online]. Available: https://coinmarketcap.com/charts/ (visited on 05/17/2023).

[31] J. Tidy, "After the ftx chaos, is crypto down and out after a torrid 2022?," Dec. 16, 2022. [Online]. Available: https://www.bbc.com/news/technology-63990215 (visited on 05/29/2023).

[32] moneysmart.gov.au. "Cryptocurrencies," moneysmart.gov.au. (), [Online]. Available: https://moneysmart.gov.au/investment-warnings/cryptocurrencies (visited on 05/17/2023).

[33] M. Adams. "Should You Invest In Crypto?" Forbes Advisor. (), [Online]. Available: https://www.forbes.com/advisor/investing/cryptocurrency/should-you-invest-in-crypto/ (visited on 05/17/2023).

[34] Binance Academy. "What is Cryptocurrency?" Binance Academy. (), [Online]. Available: https://academy.binance.com/en/articles/what-is-a-cryptocurrency%5C#How-to-Safely-Invest-in-Crypto (visited on 05/17/2023).

[35] T. Rodgers and H. Smith, "How does cryptocurrency work?," Apr. 12, 2023. [Online]. Available: https://www.thetimes.co.uk/money-mentor/article/how-cryptocurrency-works/ (visited on 05/17/2023).

[36] A. Hayes. "What Is a Time Series and How Is It Used to Analyze Data?" Investopedia. (), [Online]. Available: https://www.investopedia.com/terms/t/timeseries.asp (visited on 05/18/2023).

[37] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: A survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 24–38, 2009. DOI: 10.1109/MCI.2009.932254.

[38] A. P. Jain. "Time Series Forecasting: Data, Analysis, and Practice," neptune:ai. (), [Online]. Available: https://neptune.ai/blog/time-series-forecasting (visited on 05/18/2023).

[39] S. Brown. "Machine learning, explained," IT Management Sloan School. (), [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained (visited on 05/18/2023).

[40] A. Margunov. "The Life Cycle of a Machine Learning Project: What Are the Stages?" neptune:ai. (), [Online]. Available: https://neptune.ai/blog/time-series-forecasting (visited on 05/18/2023).

[41] T. W. Malone, D. Rus, and R. Laubacher, "ARTIFICIAL INTELLIGENCE AND THE FUTURE OF WORK," Massachusetts Institute of Technology, Cambridge, MA, USA, report 17, Dec. 2020. [Online]. Available: https://workofthefuture.mit.edu/wp-content/uploads/2020/12/2020-Research-Brief-Malone-Rus-Laubacher2.pdf (visited on 05/29/2023).

[42] E. Kavlakoglu. "AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?" Youtube. (), [Online]. Available: https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks (visited on 05/26/2023).

[43] IBM. "What is machine learning?" IBM. (), [Online]. Available: https://www.ibm.com/topics/machine-learning (visited on 05/18/2023).

[44] MathWorks. "What is Machine Learning?" MathWorks. (), [Online]. Available: https://uk.mathworks.com/discovery/machine-learning.html (visited on 05/18/2023).

[45] Google for Developers. "Training and Test Sets: Splitting Data," Google for Developers. (), [Online]. Available: https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data (visited on 05/20/2023).

[46] P. Baheti. "Train Test Validation Split: How To & Best Practices [2023]," V7. (), [Online]. Available: https://www.v7labs.com/blog/train-validation-test-set (visited on 05/20/2023).

[47] IBM. "What is overfitting?" IBM. (), [Online]. Available: https://www.ibm.com/topics/overfitting (visited on 05/20/2023).

[48] IBM. "What are neural networks?" IBM. (), [Online]. Available: https://www.ibm.com/topics/neural-networks (visited on 05/26/2023).

[49] MathWorks. "What is Deep Learning?" MathWorks. (), [Online]. Available: https://uk.mathworks.com/discovery/deep-learning.html (visited on 05/18/2023).

[50] K. Benidis, S. S. Rangapuram, V. Flunkert, *et al.*, "Deep learning for time series forecasting: Tutorial and literature survey," *ACM Computing Surveys*, 2021. [Online]. Available: https://www.amazon.science/publications/deep-learning-for-time-series-forecasting-tutorial-and-literature-survey.

[51] Data Base Camp. "Long Short-Term Memory Networks (LSTM)- simply explained!" Data Base Camp. (), [Online]. Available: https://databasecamp.de/en/ml/lstms (visited on 05/19/2023).

[52] C. Olah. "Understanding lstm networks," colah's blog. (), [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on 05/19/2023).

[53] G. Singhal. "Introduction to LSTM Units in RNN," Pluralsight. (), [Online]. Available: https://www.pluralsight.com/guides/introduction-to-lstm-units-in-rnn (visited on 05/19/2023).

[54] G. Singhal. "LSTM versus GRU Units in RNN," Pluralsight. (), [Online]. Available: https://www.pluralsight.com/guides/lstm-versus-gru-units-in-rnn (visited on 05/19/2023).

[55] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv: 1412.3555*, 2014.

[56] P. Baheti. "A Comprehensive Guide to Convolutional Neural Networks," V7. (), [Online]. Available: https://www.v7labs.com/blog/convolutional-neural-networks-guide (visited on 05/19/2023).

[57] S. Lahmiri, S. Bekiros, and F. Bezzina, "Complexity analysis and forecasting of variations in cryptocurrency trading volume with support vector regression tuned by bayesian optimization under different kernels: An empirical comparison from a large dataset," *Expert Systems with Applications*, vol. 209, p. 118 349, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.118349. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422014683.

[58] C. Phasook, J. Polpinij, and B. Luaphol, "A study of comparative methods for closed-price cryptocurrency prediction," in *2022 6th International Conference on Information Technology (InCIT)*, 2022, pp. 399–403. DOI: 10.1109/InCIT56086.2022.10067551.

[59] I. Fadil, M. A. Helmiawan, and Y. Sofiyan, "Optimization parameters support vector regression using grid search method," in *2021 9th International Conference on Cyber and IT Service Management (CITSM)*, 2021, pp. 1–5. DOI: 10.1109/CITSM52892.2021.9589028.

[60] I. Nasirtafreshi, "Forecasting cryptocurrency prices using recurrent neural network and long short-term memory," *Data & Knowledge Engineering*, vol. 139, p. 102 009, 2022, ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2022.102009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169023X22000234.

[61] S. Tanwar, N. P. Patel, S. N. Patel, J. R. Patel, G. Sharma, and I. E. Davidson, "Deep learning-based cryptocurrency price prediction scheme with inter-dependent relations," *IEEE Access*, vol. 9, pp. 138 633–138 646, 2021. DOI: 10.1109/ACCESS.2021.3117848.

[62] L. G. N. De Leon, R. C. Gomez, M. L. G. Tacal, J. V. Taylar, V. V. Nojor, and A. R. Villanueva, "Bitcoin price forecasting using time-series architectures," in *2022 International Conference on ICT for Smart Society (ICISS)*, 2022, pp. 1–6. DOI: 10.1109/ICISS55894.2022.9915199.

[63] J. Kim, S. Kim, H. Wimmer, and H. Liu, "A cryptocurrency prediction model using lstm and gru algorithms," in *2021 IEEE/ACIS 6th International Conference on Big Data, Cloud Computing, and Data Science (BCD)*, 2021, pp. 37–44. DOI: 10.1109/BCD51206.2021.9581397.

[64] T. K. Tran, T. T. T. Le, T. T. Bui, V. Q. Dang, and R. Senkerik, "Constructing a cryptocurrency-price prediction model using deep learning," in *2022 International Conference on Engineering and Emerging Technologies (ICEET)*, 2022, pp. 1–6. DOI: 10.1109/ICEET56468.2022.10007138.

[65] S. Mehtab and J. Sen, "Stock price prediction using cnn and lstm-based deep learning models," in *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 2020, pp. 447–453. DOI: 10.1109/DASA51403.2020.9317207.

[66] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using lstm, rnn and cnn-sliding window model," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643–1647. DOI: 10.1109/ICACCI.2017.8126078.

[67] S. Swati and A. Mohan, "Cryptocurrency value prediction with boosting models," in *2022 International Conference on Intelligent Innovations in Engineering and Technology (ICIIET)*, 2022, pp. 183–188. DOI: 10.1109/ICIIET55458.2022.9967540.

[68] P. Jay, V. Kalariya, P. Parmar, S. Tanwar, N. Kumar, and M. Alazab, "Stochastic neural networks for cryptocurrency price prediction," *IEEE Access*, vol. 8, pp. 82 804–82 818, 2020. DOI: 10.1109/ACCESS.2020.2990659.

[69] O. Angela and Y. Sun, "Factors affecting cryptocurrency prices: Evidence from ethereum," in *2020 International Conference on Information Management and Technology (ICIMTech)*, 2020, pp. 318–323. DOI: 10.1109/ICIMTech50083.2020.9211195.

[70] Coursera. "Coursera." (), [Online]. Available: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (visited on 05/10/2023).

[71] Python Software Foundation. "Python." (), [Online]. Available: https://www.python.org/about/ (visited on 05/10/2023).

[72] CFI Team. "Python (in Machine Learning)," CFI. (), [Online]. Available: https://corporatefinanceinstitute.com/resources/data-science/python-in-machine-learning/ (visited on 05/11/2023).

[73] Python Software Foundation. "os - Miscellaneous operating system interfaces," Python. (), [Online]. Available: https://docs.python.org/3/library/os.html (visited on 05/23/2023).

[74] pandas. "About pandas," pandas. (), [Online]. Available: https://pandas.pydata.org/about/ (visited on 05/11/2023).

[75] nvidia. "Pandas," nvidia. (), [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/pandas-python/ (visited on 05/11/2023).

[76] NumPy. "What is NumPy?" NumPy. (), [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html%5C#why-is-numpy-fast (visited on 05/12/2023).

[77] NumPy. "ARRAY COMPUTING," NumPy. (), [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/pandas-python/ (visited on 05/12/2023).

[78] scikit learn. "Frequenly Asked Questions," scikit learn. (), [Online]. Available: https://scikit-learn.org/stable/faq.html (visited on 05/12/2023).

[79] nvidia. "Scikit-learn," nvidia. (), [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/scikit-learn/ (visited on 05/12/2023).

[80] Keras. "Keras," Keras. (), [Online]. Available: https://keras.io/ (visited on 05/12/2023).

[81] Python Software Foundation. "time-Time access and conversions," Python. (), [Online]. Available: https://docs.python.org/3/library/time.html (visited on 05/12/2023).

[82] JupyterLab. "Overview," JupyterLab. (), [Online]. Available: https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html (visited on 05/12/2023).

[83] M. Mota. "JupyterLab is the data science UI we have been looking for," Medium. (), [Online]. Available: https://towardsdatascience.com/jupyterlab-you-should-try-this-data-science-ui-for-jupyter-right-now-a799f8914bb3 (visited on 05/12/2023).

[84] CryptoCompare, *Bitcoin*, CryptoCompare, May 6, 2023. [Online]. Available: https://www.cryptocompare.com/coins/btc/analysis/USD?period=1Y (visited on 05/30/2023).

[85] CryptoCompare, *Ethereum*, CryptoCompare, May 6, 2023. [Online]. Available: https://www.cryptocompare.com/coins/eth/analysis/USD?period=1Y (visited on 05/06/2023).

[86] CryptoCompare, *Tether*, CryptoCompare, May 6, 2023. [Online]. Available: https://www.cryptocompare.com/coins/usdt/analysis/USD?period=1Y (visited on 05/06/2023).

[87] CryptoCompare, *Binance coin*, CryptoCompare, May 6, 2023. [Online]. Available: https://www.cryptocompare.com/coins/bnb/analysis/USD?period=1Y (visited on 05/06/2023).

[88] CryptoCompare, *USD Coin*, CryptoCompare, May 6, 2023. [Online]. Available: https://www.cryptocompare.com/coins/usdc/analysis/USD?period=1Y (visited on 05/06/2023).

[89] CryptoCompare. "About Us," CryptoCompare. (), [Online]. Available: https://www.ibm.com/topics/overfitting (visited on 05/24/2023).

[90] scikit-learn. "sklearn.svm.SVR," scikit learn. (), [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR (visited on 05/29/2023).

[91] G. Hogg. "Multivariate Time Series Forecasting Using LSTM, GRU & 1d CNNs," Youtube. (), [Online]. Available: https://www.youtube.com/watch?v=kGdbPnMCdOg%5C&t=1775s (visited on 05/26/2023).

[92] TensorFlow. "tf.keras.layers.InputLayer," TensorFlow. (), [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/InputLayer (visited on 05/12/2023).

[93] M. Naser and A. H. Alavi, "Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences," *Architecture, Structures and Construction*, pp. 1–19, 2021. DOI: 10.1007/s44150-021-00015-8.

[94] Oracle. "MAPE." (), [Online]. Available: https://docs.oracle.com/cd/E40248_01/epm.1112/cb_statistical/frameset.htm?ch08s01.html (visited on 05/26/2023).

[95] J. Fernando. "R-Squared: Definition, Calculation Formula, Uses, and Limitations," Investopedia. (), [Online]. Available: https://www.investopedia.com/terms/r/r-squared.asp (visited on 05/26/2023).

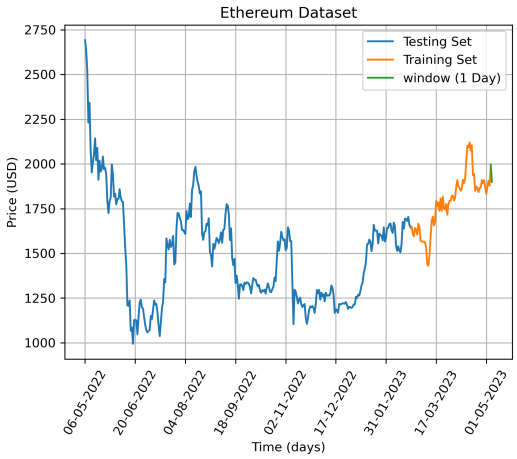
[96] Newcastle University. "Coefficient of Determination, R-squared," Newcastle University. (), [Online]. Available: https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html (visited on 05/26/2023).

[97] A. Marathe. "Negative value of r-square in the data science world... Is it a myth?" Medium. (), [Online]. Available: https://medium.com/data-science-insights-and-predictions/a-negative-value-of-r-square-in-the-data-science-world-is-it-a-myth-fa6488e69efc (visited on 05/26/2023).

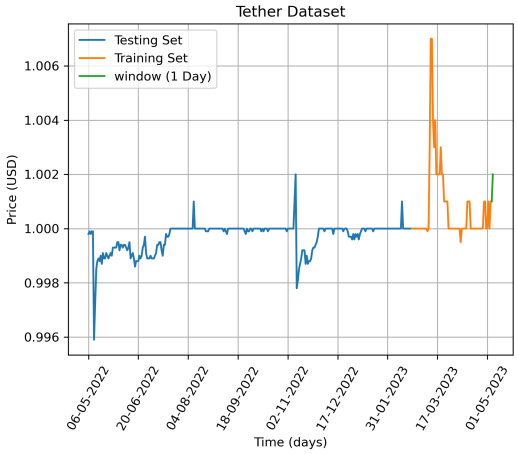
[98] Z. Zhang, H.-N. Dai, J. Zhou, S. K. Mondal, M. M. García, and H. Wang, "Forecasting cryptocurrency price using convolutional neural networks with weighted and attentive memory channels," *Expert Systems with Applications*, vol. 183, p. 115 378, 2021, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115378. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421008046.

Appendix A

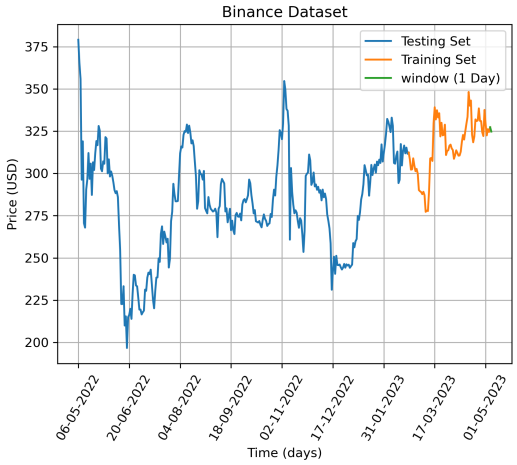
Appendix A - Coin Dataset Charts



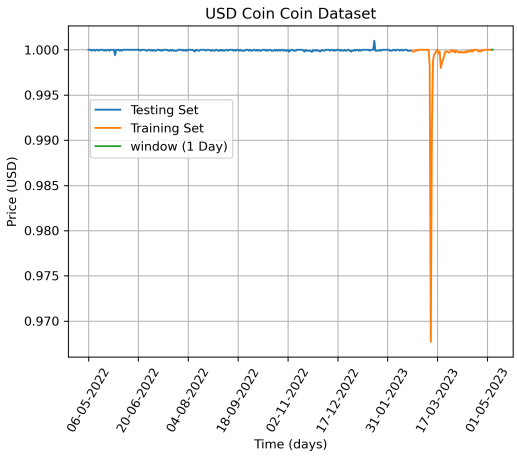
(a) ETH dataset plot



(b) USDT dataset plot



(c) BNB dataset plot



(d) USDC dataset plot

Figure A.1: Coin Dataset Plots

Appendix B

Appendix B - Output CSV File

Table B.1: Metrics_Results.csv

	Model	Training Time	Pred Time	MAPE	MSE	R2	Crypto
0	SVR	0.002018451690673828	0.0003876686096191406	0.0504696902991206	2459684.3586585657	0.6743075282528055	Bitcoin
1	LSTM	5.3043272495269775	0.7390542030334473	0.029383193375235896	944782.5171756691	0.8589234632891902	Bitcoin
2	GRU	7.334591865539551	0.3565669059753418	0.026340116436574353	777953.8180326717	0.8898258661101472	Bitcoin
3	CNN	2.1556577682495117	0.1020052433013916	0.03291028683989652	1124888.5062743116	0.8301562544838689	Bitcoin
4	SVR	0.0016543865203857422	0.0002281665802001953	0.027836961976540196	4140.34125280257	0.8003583649909457	Ethereum
5	LSTM	4.86388087272644	0.4191739559173584	0.05241754797559006	11346.171195337085	0.12381032666319391	Ethereum
6	GRU	4.942010402679443	0.4025247097015381	0.04226630966316376	7912.271755179049	0.46407892396339223	Ethereum
7	CNN	1.9289462566375732	0.6902077198028564	0.035936705143524775	5822.176331392246	0.6739285876046022	Ethereum
8	SVR	0.0008196830749511719	0.0001728534698486328	0.0009931470649536813	3.446231416445051e-06	-67.32176705628947	Tether
9	LSTM	4.294309854507446	0.4241507053375244	0.0009433767714089219	2.671374658384471e-06	-112.57929259783367	Tether
10	GRU	4.256277561187744	0.38173532485961914	0.0008682655450180871	2.5436906381243517e-06	-66.48729340153241	Tether
11	CNN	3.114292621612549	0.1326618194580078	0.0005461618193308908	1.2947820689271233e-06	-1.1579009176921997	Tether
12	SVR	0.002954721450805664	0.0004019737243652344	0.02654513913354446	110.13466288894043	-0.08939754030832492	Binance
13	LSTM	4.405181169509888	0.4783942699432373	0.05113814294999337	316.156385644618	-3.046864098585492	Binance
14	GRU	6.980800151824951	0.38898372650146484	0.029653247139795592	136.04871754472578	-0.062115659810120016	Binance
15	CNN	3.1849536895751953	0.1352834701538086	0.02669722858747477	113.13338355102192	0.19889403776005643	Binance
16	SVR	0.0007381439208984375	0.00011444091796875	0.0009833649708414144	1.5901917808219124e-05	-3.225292104699031e+26	USD Coin
17	LSTM	4.368239879608154	1.1178078651428223	0.0009508531429233711	1.609342810037662e-05	-6.775211030089833	USD Coin
18	GRU	4.457851886749268	0.3710942268371582	0.0008748145199426846	1.564576735625326e-05	-3.8894116221682777	USD Coin
19	CNN	2.1029508113861084	0.10541772842407227	0.0012837194443462467	2.333045899150681e-05	-0.32750181224560304	USD Coin