Complexity And Clarity in Forecasting Digital Assets: Evaluating Statistical, Singular/Hybrid Deep Learning and AutoML Models

Harry Cook

MSc in Computer Science
The University of Bath
2024

Complexity And Clarity in Forecasting Digital Assets: Evaluating Statistical, Hybrid Deep Learning and NAS Models

Submitted by: Harry Cook

Complexity And Clarity in Forecasting Digital Assets: Evaluating Statistical, Hybrid Deep Learning and NAS Models

Declaration

Department of Computer Science
12-Point Ethics Checklist for UG and MSc Projects

Harry Cook	**Student**
2024	**Academic Year or Project Title**
Rohit Babbar	**Supervisor**

This form must be attached to the dissertation as an appendix.

Does your project involve people for the collection of data other than you and your supervisor(s)? YES / [✓ NO]

If the answer to the previous question is YES, you need to answer the following questions, otherwise you can ignore them.

This document describes the 12 issues that need to be considered carefully before students or staff involve other people ('participants' or 'volunteers') for the collection of information as part of their project or research. Replace the text beneath each question with a statement of how you address the issue in your project.

1. *Will you prepare a Participant Information Sheet for volunteers?* YES / NO

   This means telling someone enough in advance so that they can understand what is involved and why – it is what makes informed consent informed.

2. *Will the participants be informed that they could withdraw at any time?* YES / NO

   All participants have the right to withdraw at any time during the investigation, and to withdraw their data up to the point at which it is anonymised. They should be told this in the briefing script.

3. *Will there be any intentional deception of the participants?* YES / NO

   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

4. *Will participants be de-briefed?* YES / NO

The investigator must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. This phase might wait until after the study is completed where this is necessary to protect the integrity of the study.

5. *Will participants voluntarily give informed consent?* YES / NO

Participants MUST consent before taking part in the study, informed by the briefing sheet. Participants should give their consent explicitly and in a form that is persistent –e.g. signing a form or sending an email. Signed consent forms should be kept by the supervisor after the study is complete. If your data collection is entirely anonymous and does not include collection of personal data you do not need to collect a signature. Instead, you should include a checkbox, which must be checked by the participant to indicate that informed consent has been given.

6. *Will the participants be exposed to any risks greater than those encountered in their normal work life (e.g., through the use of non-standard equipment)?* YES / NO

Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life.

7. *Will you be offering any incentive to the participants?* YES / NO

The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

8. *Will you be in a position of authority or influence over any of your participants?* YES / NO

A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. *Will any of your participants be under the age of 16?* YES / NO

Parental consent is required for participants under the age of 16.

10. *Will any of your participants have an impairment that will limit their understanding or communication?* YES / NO

Additional consent is required for participants with impairments.

11. *Will the participants be informed of your contact details?* YES / NO

All participants must be able to contact the investigator after the investigation. They should be given the details of the Supervisor as part of the debriefing.

12. *Will you have a data management plan for all recorded data?* YES / NO

Personal data is anything which could be used to identify a person, or which can be related to an identifiable person. All personal data (hard copy and/or soft copy) should be anonymised (with the exception of consent forms) and stored securely on university servers (not the cloud).

Complexity And Clarity in Forecasting Digital Assets: Evaluating Statistical, Hybrid Deep Learning and NAS Models

Abstract

"The global user base of cryptocurrencies increased by nearly 190 percent between 2018 and 2020, only to accelerate further in 2022.", Raynor De Best, Statista (2023). In November 2023, there were over 547m cryptocurrency users, and these are just the accounts that have been verified. The digital asset market is only growing and has the potential to outpace traditional markets. Optimising models for predicting future prices of traditional and digital assets is an important area of research in the financial industry. The technological capabilities of today's algorithms and automated learning models are a marvel compared to the origins of machine learning (ML). With the right preprocessing and tuning, the predictive accuracy of these models can exceed traditional methods and achieve outsized profits. But which model is the most suitable? Does complexity beat simplicity? And can they be trusted in the noisy and chaotic realm of digital assets?

Contents

Acknowledgements

I want to thank my friends, colleagues and family for their continued support during my time at the University of Bath.

Complexity And Clarity in Forecasting Digital Assets: Evaluating Statistical, Hybrid Deep Learning and NAS Models

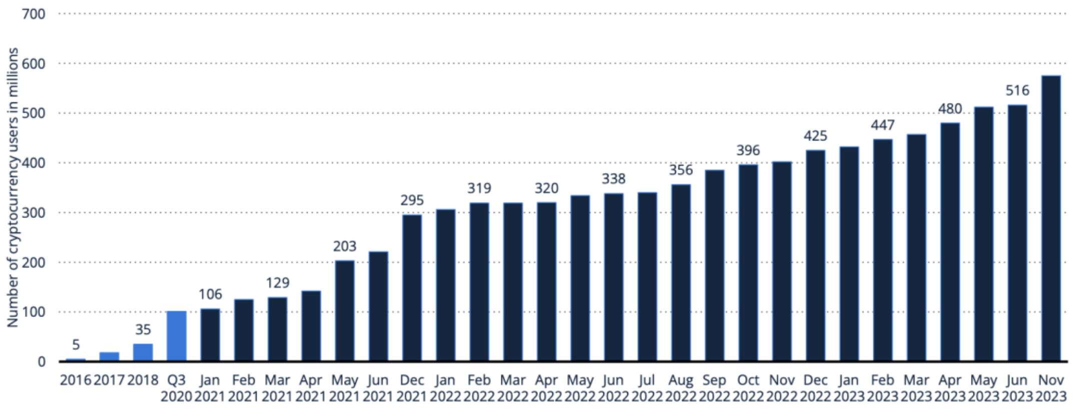
1. Introduction

1.1 Problem Description

The three main categories of research in cryptocurrency trading are market forecasting models, portfolio optimisation and automated trading. The most popular models being Neural Networks and Deep Learning (DL) among trading algorithms and reinforcement learning and fuzzy logic for automated trading, Nguyen and Chan, (2024). Forecasting in traditional and cryptocurrency markets is a difficult task due to the complexity of the data including random noise made by external market participants Lee and Kim, (2020). Financial time series (FTS) forecasting is a key focus within the financial sector, offering insights into potential risks, trends, and entry and exits for investing. Advances in technology and the continued rise in volume of data have led to more complex models for predicting these signals. Small improvements in forecasting can result in substantial gains. Enhancing the accuracy of these predictions and protecting against market volatility is of great benefit to organisations and individual investors involved in the financial market, Tang et al (2022). How complex can these models get and does this add to their predictive power? There is a push and pull over simplicity over complexity in FTS research. Too simple doesn't capture non-linearity in the data and too complex is computationally inefficient and risks overfitting. If a model is effective at predicting the direction of an asset but can't be deployed efficiently, then it can't be used effectively in real world scenarios. The stock market originated to release funds for businesses and expand production. They act as a form of investment for individuals and institutions who believe in the product or service they represent. Stocks and shares are a biproduct of the industrial revolution where money was needed to fund increased production. The investors received a share of the profits and an increase in their capital invested, Shadbolt, J. et al (2002). Thereafter, came government bonds, aggregates of stock prices (S&P 500, FTSE 100) and complex financial instruments such as futures. These are key components of economics, the financial industry, and people's own personal investment strategies, but they come with risk and uncertainty.

"Market uncertainty represents a significant risk in trading assets, characterized by difficulties in assessing current and future market conditions due to high volatility" Sather, (2023). How can predictive models help people and institutions deal with market uncertainty and help mitigate risk? Cruz et al (2021) explains, "The ability to extract knowledge and forecast stock trends is crucial to mitigate investors' risks and uncertainties in the market". Quantitative investing involves mathematical and statistical models used to identify opportunities and manage portfolios. Harry Markowitz' (1991) Modern Portfolio Theory, is still used to construct portfolios with optimal risk-return characteristics. Advancements in DL have accelerated the development and application of ML across various investment and portfolio management strategies, Grajzl & Murrell (2020).

The latest iteration within investing is the Cryptocurrency, or digital assets industry. Emerging in 2009, Bitcoin was the first pure digital asset, initiating a novel era in the realm of fintech. Since then, thousands more have been created. These are referred to as altcoins, the biggest of them all being Ethereum. These assets are underpinned by blockchain technology, which facilitates trustless and near-instantaneous peer-to-peer (P2P) transactions Tschorsch & Scheuermann, (2016). Digital assets have been influential in creating a new dynamic within financial systems and markets, presenting both opportunities and challenges Manjula et al (2022). The application of ML to areas like automated trading (AT) and digital assets is a more recent development, gaining significant traction and continuing to evolve rapidly. A 2022 survey of 146 cryptocurrency trading papers by Fang et al (2022) noted that 85% of all published scientific papers on AT of digital assets have been published since 2018 out of a sampling interval from 2013 to 2021. This reinforces how recent these technological developments are and asserts the rising use of ML in the financial sector. The understanding and development of new methods for predicting traditional and digital assets is paramount to maintaining authority to stay ahead of competitors. Graph 1 shows the growth of crypto users as of November 2023, highlighting the rate of adoption for this new technology.



Graph 1 - Number of identity-verified crypto asset users from 2016 to November 2023 (in millions), Statista (2023).

1.2 Research Aims

This project aims to discern if complexity improves or impedes predictability of the digital asset, Ethereum. This will be done by experimenting with different models of ranging complexity to predict next day prices.

1.3 Objectives

- Conduct a thorough analysis of past research across statistical, DL and AutoML models used in FTS.
- Provide background knowledge of past and current methodologies used in the research space.
- Design a series of models utilising a range of techniques and architectures.
- Compare results based on performance and complexity.

1.4 Contributions

- Literature review on FTS, statistical, ML, DL and AutoML research.
- 11 predictive models of different methodologies.
- Assessment of models and discussion on their success or failure.
- Suggestions for future research.

1.5 Structure

The thesis will continue with a literature review, discussing past and recent studies on FTS and methodologies. Thereafter, we will work through the background knowledge needed to understand the techniques used in this thesis. Then, the design methodology for all models and techniques used, followed by a discussion of the results and finally suggestions for future work.

2.

Literature Review

Digital assets are still in their infancy in the financial sector; however, their influence is growing, and institutional interest is increasing. Spot Exchange Traded Funds (ETFs) for Bitcoin were approved for some of the world's largest funds, Schmitt et al, (2024), with applications on the way for Ethereum ETFs. This removes barriers to entry for people wanting exposure to Bitcoin and represents an effort to make investing in this industry more accessible.

2.1 Market Dynamics Theory

In the past, how the markets can be interpreted and predicted has been governed by a set of theories. One of which is based on the Efficient Market Hypothesis (EMH), by Fama (1965), which suggests that all available information is continuously integrated into asset prices, allowing for the instant incorporation of new information. This implies that it is not possible to systematically outperform the market since all types of information, historical, public, and private, are reflected in current prices. Another is the Random Walk theory which suggests that stock price movements are independent and identically distributed. According to this theory, asset prices move unpredictably, making accurate forecasting impossible, Shynkevich et al (2017). Researchers introduced the Adaptive Market Hypothesis (AMH) to merge elements of the EMH with behavioral finance. This theory considers market prices as perceptions influenced by cognitive biases such as overreaction and overconfidence. An empirical study by Urquhart and Hudson (2013) compared the UK, US, and Japanese stock markets to analyze returns. They found that markets adaptively fluctuate between periods of dependence and independence. This suggests AMH more accurately describes stock behavior than the EMH. In trading, there are two primary philosophies, fundamental and technical analysis. Fundamental analysis evaluates the financial condition of businesses and assets through economic indicators like earnings, debt levels, and economic growth, considering industry and broader economic conditions. Technical analysis (TA) uses historical price and volume to predict future asset prices. It assumes that past behaviors will repeat and employs Technical Indicators (TI) to identify trends and market conditions. The combination of TA and ML forms the basis of this thesis. The use of TIs as input features is designed to help the models recognise complex patterns in the price data and increase forecasting performance.

2.2 Time Series

"A time series is whose mean, frequency, and variance fluctuate over time and frequently display high volatility, trend, and heteroskedasticity is referred to as non-stationary." Murray et al (2023). The study of time series forecasting is integral to financial and economic predictive research, Salomone et al (2019) utilised Bitcoin return data to demonstrate the efficacy of its proposed spectral subsampling Markov Chain Monte Carlo (MCMC) method. Leveraging the Fast Fourier Transform (FFT) for efficient likelihood evaluation in the frequency domain, this method offers a scalable solution for Bayesian inference in time series analysis. Applied to Bitcoin, it showcases its ability to manage the complexities of digital assets. Digital assets are characterised by significant volatility and strong price fluctuations over time. When combined with a large volume of unpredictable factors involved, forecasting is generally considered extremely challenging, resulting in complicated temporal dependencies, Liveries et al (2021). Due to the chaotic and non-linear nature of historical price datasets, such as in stocks and digital assets, ML has been applied in various ways to try to understand if forecasting is plausible. Multivariate time series capture the behaviour and relationships between two or more variables as they change over time. This is common in many fields such as finance, meteorology, Miao et al (2020), economics, Lee and Tshung (2007), and healthcare, Piccialli et al (2021). Each variable in a multivariate time series has its own sequence of observations, and these sequences are aligned in time. This alignment allows for the analysis of not only the individual time-evolving patterns of each variable but also the interdependencies and correlations between them. For example, in a financial context, a multivariate time series might include daily observations of stock prices, trading volumes, and interest rates all together.

2.3 Autoregressive Integrated Moving Average (ARIMA)

ARIMAs can model various time series with trends and seasonality, such as SARIMA, Manchanda et al (2021). Their accuracy can provide some accurate short-term forecasts depending on their parameter tuning. Mittal et al (2018) successfully modelled digital asset prices, averaging 86.424% accuracy across 95% of the assets analyzed. Kazem et al (2013) explains that FTS is non-linear and non-stationary. ARIMA models require the data to be stationary, and Kazem states they are not adequate for forecasting FTS. ARIMAs can only look at historical values and don't take external factors into account such as sentiment or economic indicators, making them less nuanced. Models that are extended beyond to incorporate external features like ARIMA with eXogenous variables (ARIMAX) or Seasonal ARIMAX (SARIMAX).

2.4 Machine Learning and Neural Networks in Finance

The Meese and Rogoff puzzle, (1983), asserts that no economic model outperforms a simple random-walk in forecasting exchange rates. Despite subsequent reviews by the authors, their findings remained unchanged. Alvarez-Diaz, (2008) criticises this for its underlying assumption of linearity, arguing it shouldn't be seen as definitive due to potential nonlinear dynamics in exchange rate movements. The limited predictive power of structural and univariate models is explained in Meese and Rogoff's 1983 study. This demonstrated that most linear models failed to surpass the simple random walk's out-of-sample forecasts. In response, several researchers have explored nonlinear modelling through methods like Markov switching models, however, they have largely failed to enhance exchange rate predictions. An alternative approach is the use of neural network models, which differ from traditional nonlinear methods. They are data-driven and can identify nonlinear relationships without predefined functional forms. Neural networks offer flexibility and can approximate any continuous function with high accuracy.

There are three classifications of ML, Supervised, Unsupervised and Reinforcement learning. Supervised learning involves developing a predictive model based on training data that includes both inputs and their corresponding correct outputs. This method relies on data where each example is paired with an outcome, often determined by an expert, to guide the model's expected performance. Commonly, in financial trading, these outcomes are based on future return predictions for various assets. Unsupervised learning works with data that lacks specific outcomes, aiming to discover underlying patterns or groupings in the data without predefined answers, useful for exploratory analysis and identifying similarities. Reinforcement learning employs software agents that learn to achieve goals by maximizing a utility function, which balances immediate rewards against future gains. This approach is particularly relevant in finance, where trading strategies can be framed as a game seeking to maximize returns over a given period.

ML for developing digital assets trading strategies relies on two main components, input features and objective function, Fang et al (2022). Training uses historical data to learn patterns using input features that will consist of fundamental and technical analysis. Input can divide into several groups of features:

- Economic - Gross domestic product (GDP), Interest rates.
- Social – Google trends, Twitter (X).
- Technical – Price, Volume
- Seasonal – Time of day/week/month/year

Output Function defines the fitness criteria to decide if the model has learnt from the data and completed its objective. Typically, they try to achieve accurate numerical or categorical outcomes. Traditional ML methods have been used in time series analysis and price prediction across multiple asset classes. Yao and Tan (2000) detailed the applicability of neural networks in predicting foreign exchange (forex) rates. The study involved feeding time series data and technical indicators into neural networks. The authors noted that in efficient markets, it is difficult to generate profits using these methods. Kim (2003) concluded Support Vector Machines (SVM) are effective for FTS forecasting, potentially outperforming traditional methods like back-propagation neural networks and case-based reasoning. Galeshchuk, (2016), explored the application of multi-layer perceptron (MLP) models in forecasting forex. Galeshchuk's research demonstrates the superior performance of MLP models over traditional linear models when predicting short-term forex for major currencies. Vasily et al (2021), used Random Forest (RF) and Stochastic Gradient Boosting Machine (SGBM) to compare their effectiveness in forecasting. This revealed that both RF and SGBM exhibit notable accuracy, with a prediction error range of 0.92-2.61% for the assets under consideration. Krauss et al (2017) also utilised RFs in various combinations of ensembles to effectively predict the performance of the S&P 500. RFs can handle non-linear relationships between features and the target variable, where price direction can be influenced by complex interactions of multiple factors. Although, they may struggle to capture the full complexity of digital assets due to relying on historical patterns. RF's provide insights to which features are most important by focusing on significant predictors. Due to the ensemble method of combining multiple decision trees, RFs are generally more robust to overfitting compared to single decision trees when dealing with noisy data. However, understanding the exact decision making is hard to interpret and how the specific features contribute to the prediction. RFs require less preprocessing of data, normalisation and scaling making them a user-friendly option for complex financial datasets.

2.5 Convolutional Neural Networks (CNNs)

CNNs, primarily designed for image processing, have been adapted for FTS. These models are known for their local connectivity, hierarchical representation, and parallel processing capabilities, which allow them to learn data representations at multiple levels. This helps discern both long and short-term dependencies in time series data, Ozbayoglu et al. (2020). They are particularly suited for FTS analysis. CNNs can identify predictors such as trends, cycles, and seasonal patterns. Examples of these could be geopolitical events, economic indicators, and market sentiment, all of which influence stock prices. To effectively apply CNNs for FTS, the data must be restructured to simulate an image, with axes for time and various features or time-lagged values. This restructuring leverages CNNs' strength in feature extraction and offer a level of invariance to data shifts and distortions. When combined with DL models like LSTMs or Deep Belief Networks (DBN), CNNs can further enhance their performance. Chen et al. (2020) explored the use of CNNs with DBN for reducing risk in quantitative investing, and Livieris et al. (2021), found moderate success with a combined CNN-LSTM model.

CNNs have emerged as a groundbreaking architecture in deep learning, achieving success through efficient training and minimising parameters by leveraging spatial relationships. This efficiency enhances performance with standard backpropagation algorithms and requires minimal preprocessing, as noted by Lui et al. (2017). Tang et al (2020) notes there are limitations to the use of CNNs for FTS however, such as the hidden state needing updating at each training step. Therefore, they cannot determine whether memory is saved or discarded. The authors also state that CNNs will find it difficult to capture the sequence-correlation characteristics of FTS data.

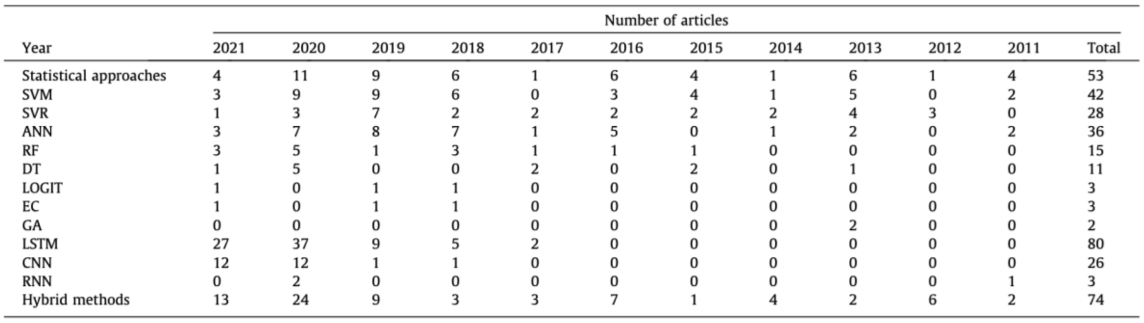
2.6 Long Short Term Memory (LSTM)

In their literature survey, Tang et al (2022) revealed the most used models for prediction were LSTM and hybrid methods. LSTMs excel in sequence modeling and Fischer and Krauss, (2018) deployed their LSTM for predicting out of sample directional movements of stocks from the S&P500 to achieve positive results. The authors found LSTM networks outperform memory-free classification methods such as RF, deep neural nets (DNN), and logistic regression classifier (LOG). Chen et al. (2020b) developed an LSTM model that obtained 67.2% accuracy, outperforming their linear regression (LR) and support vector machine (SVM) models. Cherati et al (2021) also used an LSTM for forecasting the daily close price direction of BTC, obtaining 76.83% accuracy on the testing data.

2.7 Hybrid and Ensemble Models

Liveries et al, (2020b), discussed the shortcomings of using singular deep learning models, displaying only marginal improvements in accuracy of predictions compared to traditional ML methods. The authors expressed the need for more advanced combinations of models and ensembles. Hasson et al (2023), researched ensembling strategies in time series forecasting. They demonstrated how ensemble weights can be dynamically adjusted across different dimensions for improved forecasting accuracy and enhanced prediction models. Another study by Liveieris et al, (2020a) proposed ensemble models with deep learning models as component learners, comprised of combinations of LSTM, Bi-directional LSTM and convolutional layers. The results indicated combining the strengths of different DL models is effective in forecasting digital assets. Yang et al, (2020), investigated the use of a hybrid deep learning framework combining CNN and LSTM networks for predicting stock price movements. The research focused on applying this model to FTS and forecast the direction of stock price movements, not the actual price prediction. It utilised three-dimensional CNNs with time series, technical indicators, and stock index correlations as inputs. They found combined CNN-LSTM models outperform standalone models in predicting the direction of stock price movements. This approach leverages the strengths of both CNNs in identifying local patterns and LSTMs in capturing long-term dependencies in time series data. Ensemble techniques have been applied across multiple sectors such as health, agriculture, energy, oil, gas, and finance. In every application it is reported that ensemble classifiers or regressors are more precise than the discrete classifiers or regressors, Nti et al (2020). Ballings et al. (2015), conducted a comparative study between ensemble methods, including RF, AdaBoost, and Kernel Factory, and single classifier models like Neural Networks, Logistic Regression, SVMs, and K-Nearest Neighbour. They focused on forecasting one year ahead, using data from 5767 publicly listed European companies with RFs outperforming the others. Nti et al, (2020) explored ensemble learning techniques for stock market prediction, focusing on the choice of base models, combination techniques, and the number of models in an ensemble. They concluded that "the results undoubtedly suggest that an innovative study in the domain of stock market prediction ought to include ensemble techniques".

Below is a table extracted from Tang et al's (2022) survey depicting the prediction models used in financial time series research from 2011 to 2021. It shows the latest research is centered around LSTMs, CNNs and hybrid models, highlighting the growing popularity of DL within the research community, see Table 1.



	Number of articles											
Year	2021	2020	2019	2018	2017	2016	2015	2014	2013	2012	2011	Total
Statistical approaches	4	11	9	6	1	6	4	1	6	1	4	53
SVM	3	9	9	6	0	3	4	1	5	0	2	42
SVR	1	3	7	2	2	2	2	2	4	3	0	28
ANN	3	7	8	7	1	5	0	1	2	0	2	36
RF	3	5	1	3	1	1	1	0	0	0	0	15
DT	1	5	0	0	2	0	2	0	1	0	0	11
LOGIT	1	0	1	1	0	0	0	0	0	0	0	3
EC	1	0	1	1	0	0	0	0	0	0	0	3
GA	0	0	0	0	0	0	0	0	2	0	0	2
LSTM	27	37	9	5	2	0	0	0	0	0	0	80
CNN	12	12	1	1	0	0	0	0	0	0	0	26
RNN	0	2	0	0	0	0	0	0	0	0	1	3
Hybrid methods	13	24	9	3	3	7	1	4	2	6	2	74

Table 1 – Survey on published papers for predictive models in FTS 2011-2021, Tang et al (2022).

2.8 Neural Architecture Search (NAS)

NAS is a field of research in ML that focuses on automating the process of designing neural network architecture, also known as AutoML. AutoML automates the pipelines for data collection, feature engineering and model search. This helps produce top quality models, free up expert time and make ML more accessible, Weill et al (2019). Weill notes that the development of powerful hardware and cloud computation services have made AutoML viable in recent years and references ensemble methods as consistently achieving state-of-the-art performances. Weill's AdaNet framework focuses on adaptive structural learning of artificial neural networks. It uses an algorithm that learns the model structure along with the weights of the network. Cortes et al (2016) conducted experiments on CIFAR-10 and the Criteo dataset to demonstrate the effectiveness of this method, showing that it can achieve high accuracy in binary classification tasks.

Alsharef et al (2022) researched the use of AutoML techniques for creating models for digital asset price predictions and did not achieve outstanding results when compared to manually tuned models. Alsharef also examined numerous studies that compared AutoML frameworks, Gijsbers et al (2019), Hanussek et al (2020) and Zöller and Huber (2021). Their analysis revealed significant variability or negligible differences in performance across the models they tested. Paldino et al (2021) evaluated the performance of AutoML platforms like H2O against traditional forecasting methods across several timeseries forecasting tasks. The results indicated AutoML techniques are still in the early stages for effective application for time series. Elsken et al's, (2019) NAS survey elaborates on three pivotal components for automating model architecture creation; the search space, search strategy, and performance estimation strategy. Search space is the set of architectural choices that NAS can explore, (convolutional, recurrent, etc.), their connections, and features. Search strategy determines how to explore the search space, involving various algorithms such as reinforcement learning, evolutionary algorithms, gradient-based methods, and random search.

Assessing how well an architecture performs involves training and evaluating models on a dataset with full training potentially being computationally expensive. Various strategies such as weight sharing, network morphisms, or using smaller subsets of the data, are used to estimate performance more efficiently. Zhang et al (2021), finds that the common practice for developing neural networks involves, "creating a search space for hyper-parameters that determine the architecture of the network, and then identifying the optimal parameters within this space.". The results from NAS are promising but are computationally intensive, requiring significant resources.

2.9 Alternative research

Hu, et al (2018) states that "The stock market is affected by many factors, such as political events, general economic conditions, and traders' expectations. Several examples of research incorporating an array of classes of data for improving accuracy of the models exist. Hu, et al (2018), presented two prediction models with and without Google Trends data. The results showed, including Google Trends data, significantly increases performance. The hit ratios with Google Trends were 86.81% for the S&P 500 Index and 88.98% for the Dow Jones Index. Wang et al. (2012) incorporated market sentiment and introduced an innovative text mining approach that merges ARIMA and Support Vector Regression (SVR) models. Building on the interplay of different sentiment sources, Yang et al (2017) demonstrated a genetic programming optimised trading strategy that capitalised on sentiment feedback between market news and tweets. This yielded great market returns and offered a perspective on combining this data with traditional market indicators like Moving Average Convergence Divergence (MACD) and Relative Strength Index (RSI). Akhtar (2020) presented a unified model for aspect term extraction and aspect sentiment classification. Authors researched BiLSTM and CNN frameworks to streamline the process to achieve competitive results in multiple languages. Song et al (2017) also applied 'learning-to-rank' algorithms to stock portfolio selection, using news sentiment to rank stocks. Their strategies, based on sentiment shock and trend indicators tested over a decade of data, outperformed the S&P 500 index. Garcia and Schweitzer, (2015) blended the use of social sentiment and polarisation with algorithmic trading. The authors concluded that sentiment proceeded price movement, offering an edge compared to conventional prediction methods.

2.10 Feature Selection

Guyon and Elisseeff (2003) offer insights for the construction of predictive models. Their research highlights the importance of selecting appropriate features when developing a deep learning model. They employed techniques such as simplifying complex data and choosing the most significant features, which is vital for effectively managing large and diverse datasets. They describe various methods for selecting features, including the use of filters, wrappers, and embedded methods to aid in identifying the most influential data points. The study also stresses the necessity of balancing the depth and simplicity of the data within the models to prevent them from becoming overly complex or overly simplistic. They also emphasise the importance of understanding the relationships between different data points. This is especially important for narrowing in on the data points that can most accurately forecast future prices. The authors touch on the significance of thoroughly testing these models and ensuring their adaptability to new and unforeseen market conditions.

3.

Background

3.1 ARIMA

ARIMA is a statistical method for forecasting time series data. and can be split into three respective parts. AutoRegressive, p , capturing the relationship between an observation and lagged observations, assuming past values have an influence on future values. Integrated, d , makes the time series stationary by computing the differences between consecutive observations, known as differencing the data. Stationary means it does not depend on the time at which the series is observed, making it easier to model. Moving Average, q , models the error term as a combination of past error terms. It infers past shocks or surprises that influence the trajectory of the time series. The moving average model of order q or MA(q) is shown in Equation (1).

$$X_t = w_t + \sum_{j=1}^{q} 0_j\, w_{t-1} \tag{1}$$

In Equation (i) the terms, $0_1 \ldots 0_j$ represent the model coefficients for the j-th lag. $w_1, \ldots, w_t$ are the error terms, specifically white noise. X_t is the time series value at time t . The index j runs from 1 to q , where q is the order of the moving average model. The general model known as the autoregressive moving average, or ARMA$(p,\ q)$ incorporates both autoregressive and moving average elements, as illustrated in Equation (2).

$$X_t = \sum_{j=1}^{p} \phi_j\, X_{t-j} + \sum_{j=1}^{q} \theta_j\, w_{t-j} + w_t \tag{2}$$

The first summation $\sum_{j=1}^{p} \phi_j\, X_{t-j}$ represents the autoregressive (AR) part of the model. X_{t-j} is the value of the time series at time $t - j$ (the j-th lagged value) and p , as described earlier, is the order of the autoregressive part. The second summation $\sum_{j=1}^{q} \theta_j\, w_{t-j}$ represents the moving average (MA) part of the model as described in Equation (1).

The ARIMA model, a generalisation of the ARMA model, is defined in Equation (3).

$$\left(1 - \sum_{j=1}^{p} \phi_j\, X_{t-j}\right)(1 - \mathrm{B})^d X_t \left(\sum_{j=1}^{q} \theta_j\, w_{t-j}\right) w_t \tag{3}$$

The degree of differencing, d , refers to the count of iterations where previous data points have been subtracted from the dataset. The lag operator, B, is employed to retrieve past data observations.

3.2 Deep Neural Networks (DNNs)

DL is a representation learning method featuring hierarchical representations, achieved through layering multiple nonlinear modules. Each module transforms the representation from one layer (beginning with the input layer) into a higher-level abstract representation for the subsequent layer. Through enough of these transformations, the system can learn highly complex functions. Deep multilayer perceptron's (DMLPs) are the first artificial neural network (ANN) of their kind, consisting of input, hidden and output layers, but having more layers than a multilayer perceptron (MLP). Figure 1 shows the breakdown of a DNN and its components.

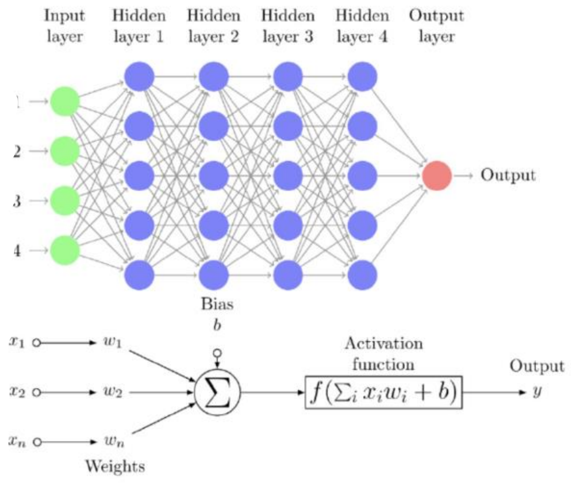


Figure 1. Diagram of a Deep Neural Network, StackExchange (2013).

Where, 1 - 4 are the input nodes, shown as green circles, with $w_{1,} w_2.., w_n$ as the weights. associated with the inputs. $x_{1,} x_2.., x_n$ represent the input values with b , the bias, as the value added to the sum of the weighted inputs to adjust the output of the neuron. f is the activation function that processes the sum of the weighted inputs and bias, $\sum_{i} {}_{i}^{x} w_i + b$, to produce the neuron's output, y . Sigmoid, hyperbolic tangent, Rectified Linear Unit (ReLU), leaky ReLU and softmax are some the most common nonlinear activation functions, Ozbayoglu (2020).

3.3 Random Forest (RF) Regressors

Training RFs with large numbers of trees and high dimensional data can be very computationally intensive and time consuming and not applicable for real-time trading applications. RFs are summarised in Krauss et al's 2017 study, where the authors believed higher returns and directional accuracy were attributed to the large number of decorrelated deep trees. The random feature selection ensures the model's immunity to overfitting, making it very reliable for financial applications. In this study we have not implemented any trading strategies so cannot surmise the models performance in a real-life scenario. However, we can assume based on previous studies that with correct optimisation, RFs can produce profitable results. Findings from Bou-Hamad and Jamali (2020) study on RFs and ANNs show RFs are particularly effective in understanding the long-term dynamics of financial series. From these findings, future work should focus on dynamic methods for persistent time series and adjust the model to make longer predictions. Random Forest is a type of ensemble learning technique used in regression (and classification) that involves constructing multiple decision trees during training and outputting the average prediction of the individual trees to improve accuracy and control over-fitting. RF regressors work by bootstrapping or bagging. They create multiple decision trees, each trained on a random subset of the training data. This is selected with replacement (known as bootstrapping), which means some observations may be repeated in each subset. When building each tree, it randomly selects a subset of features at each split rather than using all features. This helps make the trees more diverse, reducing the correlation among them. Each tree is grown to the maximum depth possible, and no pruning is usually performed. The high depth of trees may lead to overfitting but is mitigated by the ensemble approach. For regression tasks, the prediction is the average of the predictions made by all the individual trees. This averaging reduces variance and improves the robustness of the model against noise.

3.4 Support Vector Machines (SVMs)

SVMs are effective in high-dimensional cases and efficient as they use a subset of training points in the decision function called support vectors. Custom kernels can also be specified for the decision functions. SVMs and SVRs contain several key components that can be mathematically represented as follows.

3.4.1 Objective Function

SVR tries to fit the best line, in a higher-dimensional space, to the data points while keeping the deviations within a certain threshold, Equation (4):

$$\min_{w,\,b} \frac{1}{2}|w|^2 + C\sum_{i=1}^{n}(\varepsilon_i + \varepsilon_i^{'}) \tag{4}$$

Where w = weights of the model, b = bias and C = penalty parameter of the error term. ε_i, $\varepsilon_i^{'}$ = slack variables representing the distances from the actual data points to the margins of the decision boundary for points that are above and below the decision boundary, respectively.

3.4.2 Constraints

For each data point i , the following constraints need to be satisfied, Equations (5), (6), (7):

$$y_i - (w \cdot x_i + b) \leq \epsilon + \varepsilon_i \tag{5}$$

$$(w \cdot x_i + b) - y_i \leq \epsilon + \varepsilon_i^{*} \tag{6}$$

$$\varepsilon_i,\ \varepsilon_i^{*} \geq 0 \tag{7}$$

Where y_i are the target values, x_i are the feature vectors and ϵ is the epsilon-tube within which predictions are considered acceptable without penalty.

3.4.3 Kernel Trick

SVR uses the kernel trick to handle non-linear relationships. This involves mapping input features into higher-dimensional spaces. A common kernel function is the Radial Basis Function (RBF), Equation (8):

$$K(x_i,\ x_j) = \exp\left(-\gamma \left|x_i - x_j\right|^2\right) \tag{8}$$

Where γ is the parameter that needs to be tuned.

3.4.4 Dual Formulation

The problem can also be expressed in its dual form which allows the incorporation of the kernel trick, Equations (9), (10):

$$\max_{\alpha,\ \alpha^{*}} \square - \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^{*})(\alpha_j - \alpha_j^{*})K(x_i,\ x_j) - \epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^{*}) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^{*}) \tag{9}$$

$$\text{subject to: } \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^{*}) = 0 \text{ and } 0 \leq \alpha_i,\ \alpha_i^{*} \leq C \tag{10}$$

$\alpha_i,\ \alpha_i^{*}$ = Lagrange multipliers.

These formulas allow the SVR to perform regression tasks by constructing a function based on the support vectors and the kernel that predicts the next day's Ethereum prices, given historical data. The parameters C, ϵ, and γ will need to be tuned based on the dataset and the performance metrics selected.

3.5 Convolutional Neural Networks (CNN)

3.5.1 Convolutional Layer

The convolutional layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field but extend through the full depth of the input volume. For each filter, the convolution operation is performed between the filter and the input volume to produce a 2D activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input, see Equation (11):

$$Z_{ij}^{k} = \sum_m \sum_n \sum_c K_{mn}^{k} \, X_{(i+m)(j+n)c} + b^k \tag{11}$$

WhereZ_{ij}^{k} = output of the k-th filter applied to position $(i,\ j)$ and K_{mn}^{k} = value at position mn of the k-th filter. $X_{(i+m)(j+n)c}$ = input value at the position shifted by m and n in spatial dimensions, and across all channels and c b^k = bias term for the k-th filter.

3.5.2 Activation Function

After each convolution operation, an activation function is applied to introduce nonlinear properties to the system. A commonly used function is Rectified Linear Unit (ReLu), see Equation (12):

$$A(x) = \max(0,\ x) \tag{12}$$

Where $A(x)$ = activation function, x = input to the function and the function outputs x if $x > 0$; otherwise, it outputs 0. This is applied element-wise to the output of the convolutional layer.

3.5.3 Pooling Layer (Max Pooling)

Pooling layers reduce the spatial dimensions (i.e., width and height) of the input volume for the next convolutional layer. They are used to decrease the computational load, memory usage, and the number of parameters. Max pooling outputs the maximum value of the input portion covered by the filter. See Equation (13):

$$P_{ij} = \max_{a,b \in \text{Region}} Z_{i+a,\ j+b} \tag{13}$$

Where P_{ij} = output of the pooling layer at position $(i,\ j)$, the operation selects a max value $Z_{i+a,\ j+b}$ within a defined region of the filter and a and b iterate over the filter dimensions.

3.5.4 Fully Connected Layer

At the end of a CNN there is typically one of more fully connected layers, or densely connected layers. Neurons in a fully connected layer have full connections to all activations in the previous layer, see Equations (14) and (15):

$$z = Wx + b \tag{14}$$
$$a = \sigma(z) \tag{15}$$

Where x = input layer, W = weight matrix, b = bias vector, σ = activation function and a = output layer.

3.5.5 Loss Function

Finally, to train a CNN, a loss function is needed to measure the discrepancy between predicted outputs and the actual outputs. Commonly used loss functions include cross-entropy loss for classification tasks, see Equation (16):

$$L = -\sum_{c=1}^{M} y_o^{(c)} \log\left(p_o^{(c)}\right) \tag{16}$$

Where $y_o^{(c)}$ = binary indicator (0 or 1) if class label is c is the correct classification for the observation o and $p_o^{(c)}$ = predicted probability that observation o is of class c .

3.6 Long-Short-Term-Memory Networks (LSTM)

LSTMs are a type of RNNs which are designed to process sequential data, making them particularly suited for time series analysis, by maintaining an internal state that updates with each step, allowing them to remember information from previous inputs. This "memory" is key for making predictions based on both current and past data. However, RNNs suffer from vanishing gradient problem during training where gradients become too small, leading to difficulty in learning and adjusting parameters. Thich complicates capturing long-term dependencies in sequences. Bengio et al. (1994). Additionally, the original RNN model does not account for the likelihood that objects distant in time or sequence might be more closely related than those that are closer together, Hull, (2020). Training RNNs, especially on large sequences of financial data, can be computationally expensive and time-consuming, requiring significant resources.

To mitigate these issues, advancements and variations of RNNs have been introduced, such as LSTM networks and Gated Recurrent Units (GRUs). Below follows a simplified mathematical representation of an LSTM network. LSTMs are a type of recurrent neural network (RNN) designed to address the vanishing gradient problem. They process data sequentially and are used for tasks like language modelling and time series prediction. An LSTM unit follows the chain-line structure of a RNN but in its repeating module it includes four interacting neural layers known as gates: forget gate, two input gates and an output gate. The input gates add new information, forget gates remove irrelevant information and the output passes the updated information to the next cell, Murray et al (2023).

Figure 2 displays an example of the LSTM cell structure with activation layers, inputs, outputs and forget gates. For reference, in the image $\otimes$ is multiplication $(\cdot)$ and $\oplus$ is concatenation (+).

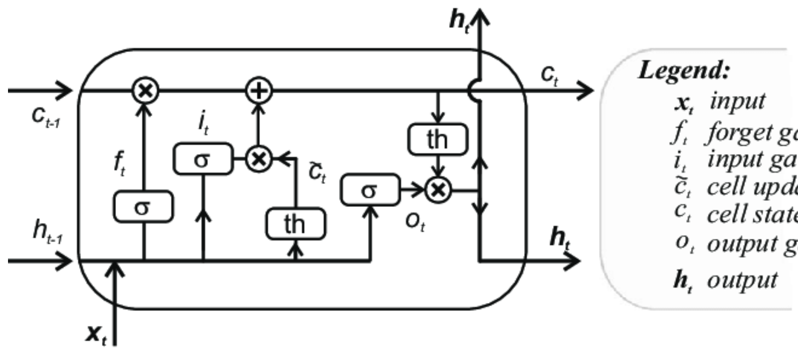


Figure 2 – LSTM Cell and Internal Structure, Hrnjica, B. and Bonacci, O., (2019).

Given an input sequence $x_1,\ x_2,\ \ldots,\ x_n$ the LSTM updates for time step can be represented as follows:

Where σ = sigmoid function, this ensures the gate outputs are between 0 and 1, $\cdot$ = element-wise multiplication and W = weights, b = biases for each gate. $[h_{t-1},\ x_t]$ = concatenation of the previous hidden state and the current input and tanh = the hyperbolic tangent function to provide outputs between -1 and 1.

Forget state, f_t, decides what information to discard from the cell state, Equation (17):

$$f_t\ =\ \sigma\big(W_t \cdot [h_{t-1},\ x_t] + b_f\big) \tag{17}$$

Input gate, i_t, and candidate cell state, $\big(\widetilde{C_t}\big)$, decide what new information to add to the cell state, Equations (18), (19):

$$i_t\ =\ \sigma(W_t \cdot [h_{t-1},\ x_t] + b_i) \tag{18}$$

$$\big(\widetilde{C_t}\big)\ =\ \tanh(W_C \cdot [h_{t-1},\ x_t] + b_C) \tag{19}$$

Update cell state, C_t, updates the old cell state into the new cell state, Equation (20):

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t} \tag{20}$$

Output gate, o_t, and output, h_t, decide the next hidden state and output., Equations (21), (22):

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{21}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{22}$$

3.7 Automated Machine Learning (AutoML) – H2O

H2O AutoML is an automated machine learning platform developed by H2O.ai, (2024) designed to simplify the process of building, tuning and deploying ML models. H2O supports various ML tasks such as classification, regression and time series prediction. Key features include:

- Automated Model Selection and Training
  - H2O AutoML automatically selects the most appropriate machine learning algorithms based on the input dataset and trains multiple models.
- Model Stacking
  - After training, it constructs a stacked ensemble model, which from our literature review has been proven to out-perform single models.
- Hyperparameter Tuning
  - Automatic tuning of hyperparameters to optimise performance, saving tije and resources.
- Ease of use
  - Simple interface to allow users to train high quality models with mininmal effort and expertise.
- Scalability
  - Engineered to efficiently handle large datasets and scale across multi-core systems and distributed environments.

H2O.ai can train RF, GBM, General Linear Models, SVM, XGBoost and Distributed RF among others. When it comes to DL, H2O does not offer a broad array of architectures like CNNs or RNNs. Instead, it focuses on highly customisable multi-layer feed forward neural networks, which are flexible and adaptable to different use cases. H2O is equipped with a range of features for building resilient architectures. It incorporates multiple hidden layers that decipher complex patterns and utilizes activation functions such as Tanh for handling nonlinear data. H2O applies regularization techniques like L1 and L2 to control weight magnitudes and implements dropout to intermittently exclude inputs, helping prevent overfitting. It also supports adaptive learning rates, including ADADELTA and RMSProp, which adjust dynamically during training to facilitate convergence. The use of momentum-based Stochastic Gradient Descent (SGD) optimizes weight adjustments by capitalizing on historical gradients, thereby improving training efficiency and convergence rates. Moreover, strategies such as early stopping minimise overfitting and computational demands, while Mini-batch Gradient Descent enables effective management of large datasets by processing smaller data batches. Tensorflow and Pytorch offer a full suite of DL architectures to choose from, which gives them an edge over H2O for custom architectures. But as we will see later in this thesis, manual hyperparameter tuning, random search and grid search are lengthy processes and require significant knowledge of the dataset and technology. The simple structure of setting up the H2O for automated learning along with a good base for DL techniques offers a lower barrier for entry and efficient use of resources.

H2O uses stacking to ensemble its models for improved performance. Here we will go through its formal representation.

3.7.1 Individual Model Predictions

For a given number of models N with outputs X , each model M_i, where $i \in \{1, 2, \ldots, N\}$, produces a prediction $P_i(X)$.

3.7.2 Stacking Layer

These predictions are used for a meta model, M_{meta}, which is trained to produce a final output Y . The meta models' predictions can be expressed as, Equation (23):

$$Y = M_{\text{meta}}\left(P_1(X),\ P_2(X),\ \ldots,\ P_N(X)\right) \tag{23}$$

3.7.3 Training the Models Using Cross Validation (CV)

Training the meta model using CV requires several steps. For this example, we use a $K -$ fold CV where the dataset is split in to K mutually exclusive subsets (folds), indexed by k . Each fold is used once as a validation set, while the other $K - 1$ folds collectively form the training set.

3.7.4 Base Model Predictions

For each fold k , each base model M_i is trained on the $K - 1$ training folds and then makes predictions on the validation fold. This results in cross-validated predictions for each base model across the dataset, Equation (24):

$$P_{i,\,k} = M_i\left(X_{train_k}\right) \tag{24}$$

Where $P_{i,\,k}$= predictions from model $M_i$for the k validation set X_{train_k}.

3.7.5 Assembling Training Data for Meta-Model

Predictions for each model for each fold are collected to form the features set to train the meta-model, see Equation (25).

$$\text{Features for } M_{\text{meta}} = \bigcup_{k-1}^{K}\left(P_{1,k},\ P_{2,k},\ \ldots,\ P_{N,k}\right) \tag{25}$$

3.7.6 Training the Meta-Model

The meta-model is trained using the cross-validated predictions as features and the corresponding true outcomes Y_k from each validation set as the target, see Equation (26).

$$M_{\text{meta}} = \text{Train}\left(\bigcup_{k=1}^{K}\left(P_{1,k},\ P_{2,k},\ \dots,\ P_{N,k}\right);\bigcup_{k=1}^{K} Y_k\right) \tag{26}$$

3.7.7 Objective Function for Meta-Model

The training objective remains to minimize the loss function $\mathcal{L}$, which in this setting evaluates the difference between the meta-model's predictions and the actual targets over all validation sets, see Equation (27).

$$\min_{M_{\text{meta}}} \mathcal{L}\left(\bigcup_{k=1}^{K} Y_k\ ,\ M_{\text{meta}}\left(\bigcup_{k=1}^{K}\left(P_{1,k},\ P_{2,k},\ \dots,\ P_{N,k}\right)\right)\right) \tag{27}$$

This method for training the meta-model should lead to a final ensemble that is robust and generalises well on unseen data. For a full deep dive into the inner workings of H2O's platform, we recommend reading through their documentation to see the full extent of how it can be utilised, H2O.ai (2024b).

3.8 Performance Metrics

In this study the performance of each model is evaluated based on a series of performance metrics commonly used for regression tasks. The training of each model is executed to reduce the mean squared error and from that the other metrics can be derived. Equations (28) to (32) display Mean Squared Error (MSE), Root Mean Squared error (RMSE), Mean Average Error (MAE), Mean Average Percentage Error (MAPE) and Correlation Coefficient (R^2).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(O_i - T_i)^2 \tag{28}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(O_i - T_i)^2} \tag{29}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|O_i - T_i| \tag{30}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{T_i - O_i}{T_i}\right| \tag{31}$$

$$R^2 = \frac{\left|\sum_{i=1}^{n}(T_i-\overline{T})(O_i-\overline{O})\right|}{\sqrt{\sum_{i=1}^{n}\left(T_i-\overline{T}\right)^2\sum_{i=1}^{n}\left(O_i-\overline{O}\right)^2}} \tag{32}$$

Where, n denotes the number of observations (or data points), O_i is the observed actual values from the data and $\overline{O}$ is the mean average of all observed values. T_i represents the predicted or target values generated by the regression model and $\overline{T}$ is the mean average of all predicted values.

3.9 Fast Fourier Transform (FFT)

FFT is a computational algorithm used to quickly compute the Discrete Fourier Transform (DFT) and its inverse. The DFT converts a sequence of time-domain data into components of different frequencies to reveal the periodic structures and the frequency spectrum of the original time series data. The FFT does this by reducing the complexity of computations needed from $O(N^2)$ to $O(N\log N)$, where N is the number of data points.

DFT can be represented as follows, Equation (33):

$$X|k| = \sum_{n=0}^{N-1} x[n] \cdot e^{-i2\pi k\frac{n}{N}} \tag{33}$$

Where $X|k|$ = complex number representing the amplitude and phase of the signal at the frequency component k . $x[n]$: n^{th} = sample of the input time series (close prices), N = total number of samples in the dataset and k = current frequency.

We compute the FFT of Ethereum closing prices to break the time series into frequencies, each with amplitude and phase. By retaining the first few components and zeroing out the rest, we apply a low-pass filter, smoothing the signal and highlighting longer-term trends. The inverse FFT then transforms these components back, creating reconstructed time series at various smoothing levels. This reduces noise and enhances the efficiency of analyzing large datasets, using frequencies as features to detect cyclic patterns. See Figure 3 for a visual representation of how FFT is applied to the closing prices of Ethereum.

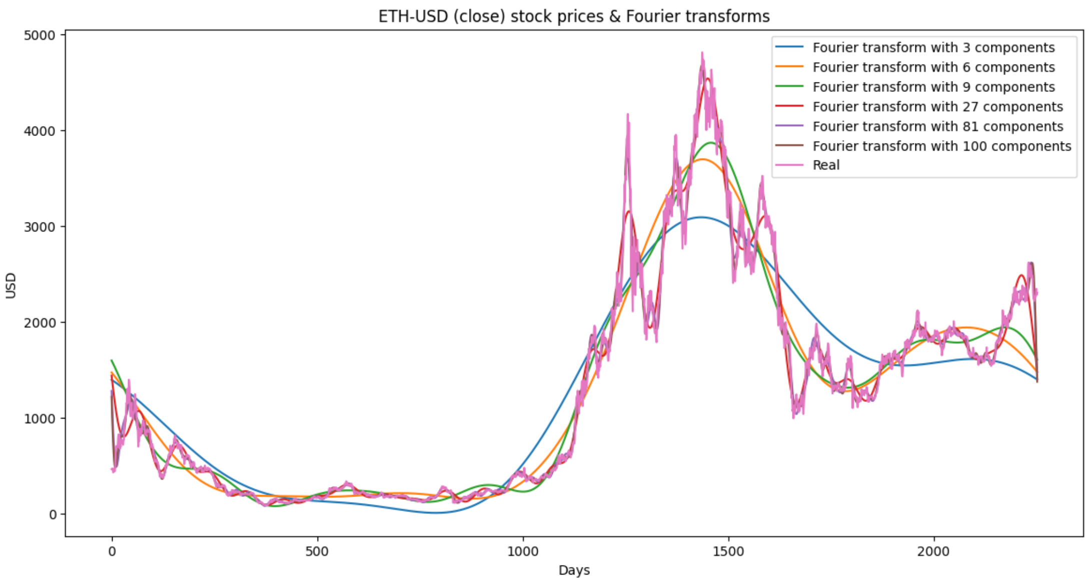


Figure 3 – Fourier Transformations on Ethereum Price Data

3.10 Machine Learning Environment and Libraries

3.10.1 Google Colab

Google Colab is a free cloud service hosted by Google for executing Python code through their browsers and is used in this research. Suited for ML and data analysis, it allows users to host runtime environments and leverage powerful hardware like Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). It seamlessly integrates with Google Drive, facilitating sharing, storage and access and hence is extremely useful for academic purposes. Colab was chosen primarily for its accessibility and computational power. It also eliminates the need for local runtimes; however, we did exercise this as we will discuss later. Colab provides a standardised environment to ensure reproducibility and consistency in results and allows access to superior computational power at no cost and its accessibility was crucial to the project, Google Colab, (2024a).

3.10.2 TensorFlow

TensorFlow is an open-source library developed by google for numerical computation and large-scale ML. It houses a collection of ML and DL models and algorithms, suitable for research and production. TensorFlow uses dataflow graphs for computations, where nodes represent mathematical operations, and the edges represent the tensors communicated between them. It supports both Computer Processing Unit (CPU) and GPU computation and offers great flexibility for designing neural networks, TensorFlow (2024).

3.10.3 Keras

Keras was initially an independent project, is now fully integrated with TensorFlow as `tf.keras`. Keras provides a high-level API for easy building and training of DL models with minimal code such as. layers, activation functions and optimisers. It is user friendly, modular and extensible allowing for fast prototyping and experimentation.

3.10.4 scikit-learn

scikit-learn is a Python library for machine learning that provides simple and efficient tools for data mining and data analysis. It is built on NumPy, SciPy, and matplotlib, and offers a range of state-of-the-art algorithms for regression. Unlike TensorFlow and Keras, which are more focused on deep learning, scikit-learn is mainly focused on supporting traditional ML algorithms. It is known for its clean, uniform, and streamlined API, and has extensive documentation, making it easy to learn and deploy models.

These libraries complement each other in the ML workflow. TensorFlow and Keras facilitate complex model architectures and deep learning and scikit-learn offers robust pre-processing techniques, feature selection, simple and powerful tools for ML models. All of these have been utilised for our experiments along with other notable libraries used for evaluation, plotly, and fetching data, Yahoo finance.

3.10.5 Yahoo Finance

`yfinance` is a Python library that provides a straightforward way to download historical market data directly from Yahoo Finance and is where we will be sourcing our datasets. After installing using the `pip install yfinance` all that is needed is to import the library and select a ticker from the website, Yahoo Finance (2024), see below an extract of the data that can be fetched, see Table 2:

Date	Open	High	Low	Close	Volume
2017-11-09 00:00:00+00:00	308.644989	329.451996	307.056000	320.884003	893249984

Table 2 – Example of data fetched from Yahoo Finance

4.

Design Methodology

4.1 Model Choices

We have designed statistical, ML, DL and AutoML models for predicting the next day closing price of Ethereum. The aim is to assess how complexity in model choice and architecture effects predictive power. We will fetch the price history of Ethereum from Yahoo Finance, extract its TI's and perform some exploratory data analysis to assess their linear relationship with the closing price. Historical price data has also been fetched from several other leading digital assets all of which can be found in Appendix D along with a brief description.

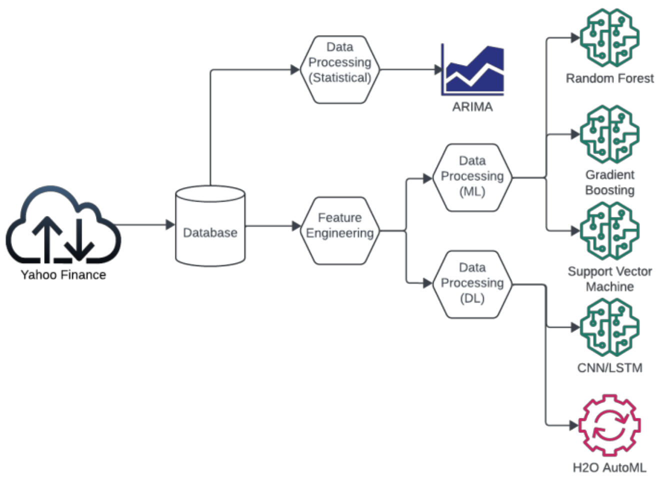


Figure 4 – Workflow from data fetching to models

Six architectures were chosen based on their popularity in current and past literature. ARIMA models have been used for decades for statistical analysis and forecasting stock prices Ayodele and Aderemi (2014), Kumar (2019), Yao and Tan (2000). ARIMAs naturally fit as a benchmark for comparison against more complex and advanced techniques. Random Forest, Support Vector Machine and Gradient Boosting models will be assessed as ML models. These have produced noteworthy results for forecasting both traditional and digital assets in the past and are still currently being used in research, Akyildirim, (2023). A series of LSTM and CNN-LSTM models have been designed as the DL models. Hybrid models have consistently shown in the literature that they outperform most other singular ML or DL models, Hamayel and Owda, (2021), Ptel et al (2020). Finally, we will present an Automated Machine Learning model. This AutoML will be able to take the dataset and automatically iterate through a set of architectures of ranging complexity and their hyperparameters to find the optimal model.

We have already touched on hyperparameter tuning and its importance in the literature review, 2.10. In our experiments, we have adopted some hyperparameter tuning techniques such as grid search, random search and Bayesian optimisation. These techniques have been shown to increase convergence for optimal architectures and boost performance McNally et al (2018), Tripathi and Sharma (2023). See Table 3 for a breakdown of each model and their individual processes. As mentioned, we extracted price data and features from a range of other assets for some of the models. These were used for most of the models, however, to further assess added complexity, some models only included the features extracted from the price data for Ethereum and some feature engineering was omitted entirely. By omitting the extra data, we can assess what added features and external factors (price action of other leading assets) have on performance. ARIMA models do not take in other features, they only look at the sequence data they are predicting. The nomenclature below will be referenced throughout the thesis and in reference to ipynb files attached to this report.

None = No feature engineering
NOT FULL = Just features from Ethereum
FULL = Features from Ethereum and other leading assets.
RS = Random Search,
GS = Grid Search,
GB = Gradient Boosting,
RF = Random Forest,
BO – Bayesian Optimisation,
N/FE = No Feature Engineering,
SVR = Support Vector Regressor,
CNN-BiLSTM = Convolutional Neural Network – Bilateral Long Short Term Memory
D = Default.

Model	Type	Feature Engineering	Parameter Tuning Method
ARIMA-RS	Statistical	None	Random Search
ARIMA-GS	Statistical	None	Grid Search
RF-D	Ensemble Learning	FULL	None
RF-GS	Ensemble Learning	FULL	Grid Search
GB-D	Machine Learning	FULL	None
GB-GS	Machine Learning	FULL	Grid Search
SVR-GS	Machine Learning	FULL	Grid Search
LSTM-BO	Deep Learning	NOT FULL	Bayesian Optimisation
LSTM-BO-N/FE	Deep Learning	None	Bayesian Optimisation
CNN-BiLSTM	Deep Learning	FULL	Manual

Table 3 – Model individual processes

4.2 Data Fetching

As explained previously in 3.10.5, the data will be fetched through the `yfinance` library to directly extract the historical price data of Ethereum and several other assets.

4.3 Feature Engineering

TIs are an essential tool for assessing the status of and prediction of any asset in financial time series and have been used by many researchers, Shynkevich et al (2017). Atsalakis and Valavanis (2009), assessed roughly 20% of the financial market forecasting approaches use technical indicators as input features. Tripathi and Sharma (2023) found that utilising TIs as input features in their DL models outperformed previous models in the literature. Their results predicted with absolute percentage errors as low as 0.28% for next days forecasts and 2.25% for seven days forecasts. ARIMAs are the exception as they are designed specifically for univariate time series forecasting, which means they predict future values based solely on past values of the series. We will list the TIs included in this research and their mathematical representations. There are thousands of indicators available to traders but for our purpose, we will focus on the most prolific and understandable TIs used in technical analysis.

4.3.1 Simple Moving Average (SMA)

SMA takes the arithmetic mean of a given set of prices over the specific number of periods in the data set. Mathematically, it's represented as, Equation (34):

$$\frac{\sum_{i=1}^{n} \text{Price}_i}{n} \tag{34}$$

Where Price_i = the closing price at each period i and n is the total number of periods. For example, if you're calculating a 20-day SMA, you sum up the closing prices of the last 20 days and divide by 20.

4.3.2 Exponential Moving Average (EMA)

EMA is a type of weighted moving average (WMA) that places greater weighting to recent price data, making it more responsive to new information compares to the SMA, see Equation (35):

$$EMA_t = \left(\text{Price}_t \times \frac{s}{1+n}\right) + EMA_{t-1} \times \left(1 - \frac{s}{1+n}\right) \tag{35}$$

Where t = the current time-period and s = smoothing constant typically equal to 2.

4.3.3 Moving Average Convergence/Divergence indicator (MACD)

MACD is a trend-following momentum indicator that shows the relationship between two moving averages of an assets price. It is used to identify potential buy and sell signals through crossovers and divergence from a signal line, Equation (36):

$$EMA_{short}(t) - EMA_{(long)}(t) \tag{36}$$

Accompanied by a signal line, which is an EMA of the MACD itself, typically over 9 periods, Equation (37):

$$\text{Signal} = EMA_9(MACD) \tag{37}$$

EMA_{short} and EMA_{long} are EMAs with different lengths typically 12 and 26 respectively.

4.3.4 Relative Strength Indicator (RSI)

RSI is a momentum indicator used to measure the recent price change to assess if the assets is overbought or oversold. It can be displayed as an oscillator moved between two extremes and read from 0 to 100, see Equation (38):

$$100 - \left(\frac{100}{1+\frac{\text{Average Gain}}{\text{Average Loss}}}\right) \tag{38}$$

Where the Average Gain and Average Loss are average gains and losses of the asset, typically over a period of 14 days.

4.3.5 Average True Range

ATR is a volatility indicator that measures the degree of price volatility by decomposing the entire range of an asset for that period, see Equation (39):

$$ATR_t = \frac{1}{n}\sum_{i=1}^{n} TR_i \tag{39}$$

Where TR_i is the True Range, which is a maximum of; $\text{High}_t - \text{Low}_t$, $\left|\text{High}_t - \text{Close}_{t-1}\right|$ and $|\text{Low}_t - \text{Close}_{t-1}|$.

4.3.6 Bollinger Bands

Bollinger Bands consist of a middle band being an SMA, and two standard deviation bands above and below it, which adjust themselves based on market volatility, helping to identify overbought or oversold conditions, see Equations (40) to (42):

$$\text{Middle Band} = SMA_{close}(n) \tag{40}$$
$$\text{Upper Band} = SMA_{close}(n) + k \cdot SD_{close}(n) \tag{41}$$

$$\text{Lower Band} = SMA_{close}(n) - k \cdot SD_{close}(n) \tag{42}$$

Where SMA = Simple Moving Average, SD = standard deviation of closing prices over n periods and k is typically set to 2.

4.3.7 Raw Stochastic Value

RSV is a component of the Stochastic Oscillator, a momentum indicator used in technical analysis. It measures the position of an asset's closing price relative to the high and low range over a specific period, see Equation (43):

$$RSV = \frac{\text{Close} - \text{Low}_\text{n}}{\text{High}_\text{n} - \text{Low}_\text{n}} \cdot 100 \tag{43}$$

Where Close = closing price, Low_n and High_n are the lowest and highest prices over the last n periods, respectively.

4.3.8 Ichimoku Cloud (IKH)

IKH includes five plots, four based on the average of the high and low over a given period of time. Periods can be adjusted when an indicator is created.

- Tenkan-sen (Conversion Line): (9-period high + 9-period low)/2))
- Kijun-sen (Base Line): (26-period high + 26-period low)/2))
- Senkou Span A (Leading Span A): (Conversion Line + Base Line)/2))
- Senkou Span B (Leading Span B): (52-period high + 52-period low)/2))
- Chikou Span (Lagging Span): Close plotted 26 days in the past

4.3.9 Volume Weighted Average price (VAP)

Calculates the average price of an asset, weighted by volume, over a specific period. VWAP is often used as a trading benchmark especially by day traders and in algorithmic trading, to ensure trading at a favorable price, see Equation (44):

$$VWAP = \frac{\sum \text{Price}_t \cdot \text{Volume}_t}{\sum \text{Volume}_t} \tag{44}$$

Price_t = Transaction rice at time t , Volume_t = number of shares/tokens traded at time t .

4.3.10 Chaiken Money Flow (CMF)

Measures the volume-weighted average of accumulation and distribution over a specified period. It combines price and volume to show how money may be flowing into or out of a stock. Positive values suggest buying pressure, while negative values indicate selling pressure. CMF is calculated in three steps, where n = number of periods CMF is calculated and High_t, Low_t and $\text{Close}_t t$ are the high, low and close prices at time t ,see Equations (45) to (47):

$$\textit{Money Flow Multipler}_t = \frac{\left((\text{Close}_t - \text{Low}_t) - \left(\text{High}_t - \text{Close}_t\right)\right)}{\text{High}_t - \text{Low}_t} \tag{45}$$

$$\textit{Money Flow Volume}_t = \textit{Money Flow Multipler}_t \cdot \textit{Volume}_t \tag{46}$$

$$CMF_t = \frac{\sum_{i=t-period+1}^{t} \textit{Money Flow Volume}_i}{\sum_{i=t-period+1}^{t} \textit{Volume}_i} \tag{47}$$

4.3.11 On Balance Volume (OBV)

OBV uses volume flow to predict changes in price. Volume and price are combined to show how money is flowing in or out of a stock. When the price increases, volume adds to the OBV, and when it decreases, volume is subtracted, see Equation (48):

$$OBV_t = OBV_{t-1} + (\text{Volume}_t \cdot \text{Multiplier}) \tag{48}$$

Where Multiplier = +1 if $\text{Close}_t > \text{Close}_{t-1}$, -1 if $\text{Close}_t < \text{Close}_{t-1}$ and 0 if $\text{Close}_t = \text{Close}_{t-1}$.

4.3.12 Commodity Channel Index (CCI)

Identifies new trends or cyclical conditions, typically used to detect when an asset is reaching a condition of being overbought or oversold. It does this by measuring the variation of a commodity price from its statistical mean, where n = number of periods for the SMA, see Equation (49):

$$CCI = \frac{\text{Typical Price}_t - SMA\ (\text{Typical Price}, n)}{0.015 \cdot \text{Mean deviation}} \tag{49}$$

Where, Typical Price = $(\text{High}_t + \text{Low}_t + \text{Close}_t)$ / 3

4.3.13 Stochastic Oscillator

Stochastic Oscillator is a momentum indicator comparing a particular closing price of an asset to a range of its prices over a certain period of time. The sensitivity of the oscillator to market movements is reducible by adjusting that period or by taking a moving average of the result, see Equation (50):

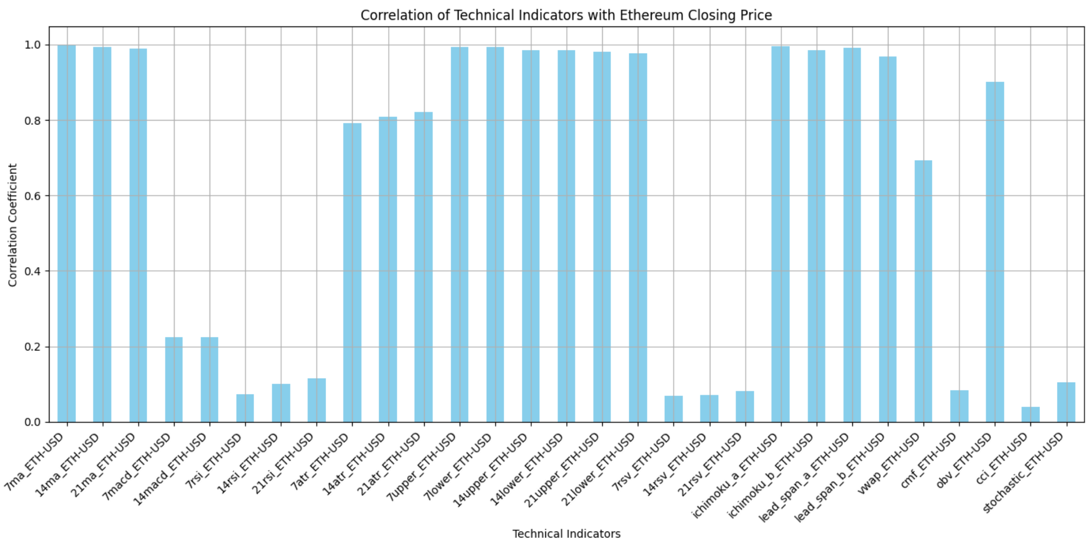
$$\text{Stochastic} = \frac{\text{Close}_t - \text{Low}_n}{\text{High}_n - \text{Low}_n} \cdot 100 \tag{50}$$

4.4 Exploratory Data Analysis

Following feature engineering, the linear correlations between the indicators and the close price of the asset can be analysed. To do this we establish the target variable as Ethereum's close price and scale the data accordingly. The Pearson correlation coefficient helps identify the strength and direction of a linear relationship between two variables. –1 indicates a weak correlation +1 a strong correlation. The formula for Pearsons Correlation r , between two variables x and y is described in Equation (51):

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2]\left[n\sum y^2 - (\sum y)^2\right]}} \tag{51}$$

This is handled within the Pandas library when calling the `.corr()` method which computes the correlation between pairs of columns in a dataframe. See below a graphical representation of the correlation for all the technical indicators and the close price of Ethereum, see Graph 2.



Graph 2 – Linear relationships of technical indicators to closing price of Ethereum

From this chart we can derive that the MACD, RSI, RSV, CMF, CCI and Stochastic Oscillator are showing a poor correlation to the closing price of Ethereum data. We could deduct these from the feature engineering process when preparing the data for training the models. However, correlation does not necessarily mean causation, so how trustworthy can these findings be? OKX.com (2024) and Bybit.com (2023) are both leading exchanges in the digital asset industry and both put most of these indicators (including the low correlated TIs) in their top ten recommended indicators for analysing digital assets. All these TIs have proven to be useful in the literature, Zatwarnicki et al (2023), Huang et al (2019) and therefore we decided to implement all in the experiments.

Epoch/batch size play a significant role in tuning a model. The number of epochs corresponds to the number of times the training process should loop over the full sample. Whereas batch size determines the number of observations used for each increment of the loop, Hull (2020). At each epoch, the model outputs the value of the loss and the accuracy of the predictions, both in the training and validation samples.

4.5 Hyperparameter Tuning

4.5.1 Grid Search

Grid search is a systematic method that defines a grid of hyperparameter values to explore. In terms of time series analysis this could be exploring different units, dropout rates or learning rates. It exhausts all possible combinations set within the boundaries the user has defined and can be computationally expensive and time consuming, however it is simple to implement and interpret. One tactic is to start with a large set of parameters with wide differences in values, then gradually become more precise with the values and tighter in range. This way the users can cover a large set of parameters and converge to an optimal range without lengthy computational times between iterations. We utilised grid search for many of the models

4.5.2 Random Search

Random search randomly iterated through a sample of hyperparameters from a specified distribution, set by the user. Unlike grid search, instead of searching through all combinations, it will randomly select combinations to evaluate. This is useful when computational resources are limited or the impact of individual hyperparameters is not well understood.

4.5.3 Bayesian Optimisation

Bayesian optimisation is a probabilistic model-based approach or global optimisation of black-box functions. This means it uses previous observations to build a probabilistic model of the objective function, such as a Gaussain process. Based on this model, it decides where to sample the next set of hyperparameters. It does this by balancing exploration, sampling in regions where uncertainty about the function is high, and exploitation, sampling where the function is likely optimal.

In summary, grid search is exhaustive but computationally expensive, random search is more efficient for high-dimensional spaces, and Bayesian optimization is effective for complex search spaces with limited evaluations.

4.6 Evaluation

The model's performance will be evaluated via the RMSE detailed previously in 3.8. Their complexity will be assessed by their training time, inference is also included (time taken to make predictions). The dataset date range, technical indicator parameters, train/test splits and scaling have been kept consistent throughout the model designs wherever possible. All computation is undertaken in the Google Colab environment. Specific specs on the CPU/TPUs used can be found in Googles official documentation, Google Cloud (2024b).

5.

Results

In the design process some models were tuned with 'Default' parameters. These served as a base with minimal tuning and used the most common parameters such as 100 trees or 100 estimators. Two experiments were conducted on a singular LSTM model, one with zero feature engineering and one with technical indicators extracted from just Ethereum. Both used Bayesian optimisation for architecture tuning. See 4.1 to recount the nomenclature for labelling the models. The LSTM-BO, due to its suspiciously low RMSE and the ARIMA-GS, due to its substantially high MAPE, have been omitted in this comparison. Further work is needed to assess the processing and parameter selection. The ARIMA models were not expected to achieve outstanding results for the reasons discussed in 2.3. ARIMAs are not designed to handle non-linear data like price history of stocks and cryptocurrency as they require stationary, sequential data to perform well.

Table 4 shows each model ranked in order of ascending RMSE, with the lowest indicating the best performance. Table 5 shows the models complexity ranked by ascending Train Times (s), where the fastest is considered the least complex. We observe that the H2O AutoML exhibits the best performance, yet the execution time is ranked second slowest. This could hinder quick action in a real-life scenario, where seconds count for profitability. The SVR presents the fastest Training Time and could therefore be considered the least complex, however it is one of the worst performing models.

The DL approaches are the top performing models, except the CNN-LSTM, but are in the bottom half regarding training time. The ML models are the most consistent. They hold third, fourth, fifth and sixth place in terms of performance and the top three for fastest models to train, bar the exceptions of the GB-GS and RF-GS. Unsurprisingly the consistently worst performing model is the ARIMA-GS.

Model	RMSE	MAE	MAPE	R2
~~LSTM-BO~~	~~$0.02~~	~~$0.02~~	~~21.40%~~	~~0.87~~
H2O AutoML	$3.50	$1.60	N/A	0.99
~~ARIMA-RS~~	~~$52.50~~	~~$35.30~~	~~122.90%~~	~~0.002~~
LSTM-BO-N/FE	$69.00	$57.90	3.22%	0.95
RF-GS	$72.15	$38.70	3.18%	0.99

RF-D	$79.20	$39.80	3.20%	0.99
GB-D	$80.70	$41.20	3.60%	0.99
GB-GS	$80.70	$41.20	3.56%	0.99
CNN-BiLSTM	$204.70	$147.00	9.50%	0.54
SVR-GS	$213.20	$92.80	18.20%	0.96
ARIMA-GS	$247.50	$198.70	12.40%	0.39

Table 4 Performance of each model based on RMSE

Model	No. Of Parameters/Trees	Train(s)	Inference(s)
SVR-GS	N/A	29	1.102
GB-D	100 Trees	52	0.262
~~**LSTM-BO**~~	~~1,539,297 Parameters~~	~~**93~~	~~1.245~~
RF-D	100 Trees	98	0.304
LSTM-BO-N/FE	3,152,385 Parameters	**146	5.8
CNN-BiLSTM	79241 Parameters	148	0.79
RF-GS	50 Trees	794	0.609
ARIMA-GS	N/A	913	0.017
GB-GS	N/A	1107	1.906
H2O AutoML	N/A	3616	0.239
~~**ARIMA-RS**~~	~~N/A~~	~~16498~~	~~215~~

*Table 5 Complexity of each model based on training time. **Additional compute points purchased to speed up training time.*

5.1 Discussion

All parameters for the models can be found in Appendix A. The H2O AutoML model converged on a Gradient Boosting Regression model and the parameters show the number of estimators at 90, which is considered moderate and the mean leaves of 60.4, considered high. This allows the model to make many decisions to enable more fine-grained partitioning of the data space, good for capturing subtle patterns. The model is therefore considered complex. The LSTM-BO-N/FE model, with an RMSE of $69 and over 3m parameters, is very complex and was slow to train, necessitating additional computational resources purchased from Google. This improved training speed but complicates direct comparisons with other models in the experiment. Despite its large parameter count suggesting potential inefficiency, its Bayesian Optimization (BO) feature is valuable for selecting optimal architectures, enhancing reproducibility across different digital assets. This approach and grid search offer flexibility over hard coding parameters. The ML models, except the SVR, hold the mid-table for performance with RMSEs ranging from $72.15 to $80.7. When considering the current price of Ethereum at $3121 at time of writing, this means on average the RF-GS, RF-D and GB-D have a 2.5% relative error of magnitude, see Equation (52).

$$REOM\% = \left(\frac{RMSE}{\text{Current Price}}\right) \cdot 100 \tag{52}$$

Given how volatile digital assets are this could be considered reasonable. The ML models have the best training times and their collective average MAPE less than 10%, without the SVR this sits lower at 3.3%. MAPE tells us how big the average error is as a percentage of actual values. This counts for accuracy, reliability and error impact (the errors aren't drastically impacting the predictions). The evidence suggests that the ML models are the most consistent in terms of performance and efficiency. Visual inspection of predictions and learning curves in Appendix B can be used to verify correct model training. The ML models do not easily allow for learning curves to be extracted and due to time constraints allowances were made. Instead, the predicted versus the actual values were plotted for the ML models. The relationship between the current price and the previous day's price is expected to be strong in FTS and is displayed for emphasis. The strong linear relationship between the predicted and actual prices from the models indicates they are a good fit for predicting the price of Ethereum. The clustering of points along the line of best fit suggests low variance in prediction errors. The CNN-LSTM Model Learning Curve in Appendix B illustrates training and validation loss over time, revealing sharp initial decreases followed by gradual convergence, indicating effective generalization rather than memorization. The losses stabilize post-initial epochs, suggesting consistent performance without overfitting. A small spike at epoch 60, missed by early stopping, recovers quickly, posing minimal concern. The H2Os predicted versus actual values can also be seen in Appendix C and display exceptional linearity, indicating strong performance. We are also able to view which variables are most influential in predicting the target variable. The LSTM-BO-NF/E graph of actual versus predicted prices in Appendix B showcases the predicted prices closely follow the actual prices in a shifted manner. This behaviour is characteristic of a model that carries over the last known value as the prediction. This could happen for several reasons such as data leakage, where the model was fed future information during training. An ‘echo state’, where the model outputs the last observed data point as the best solution to minimize the training error. Overfitting to the Ethereum data and insufficient feature engineering are likely the cause of the model performing this way. Further techniques to increase generalisation will be needed before this model can be considered for fair comparison.

6.

Conclusions

This thesis aimed to compare a range of regression models to predict next day prices of Ethereum. We explored varying levels of complexity in architecture and its effect on predictive power. From the results we can see that the top performing models utilised both ML and DL techniques exhibiting the most complex architectures. This leads us to believe greater complexity in model choice and architecture design can boost performance for FTS regarding digital assets. The AutoML H2O model is the best performing model and was the most complex and computationally expensive to execute. Retail and institutional traders rely on predictive models to assess the risk of investing in an array of assets. There are those who trade on tighter or looser time scales such as hourly or monthly charts. The work conducted in this thesis would need tuning for these specific needs. Investigation into developing a platform that can actively switch between timeframes and architectures based on the users' needs would be one avenue for future work. The alternative digital assets used as features could have been iterated further. There was no considerable research into which assets were included and this was a flaw in the research. Further research into which assets correlate with each other could yield improvements. If the model could include alternative assets that rotated over different combinations, an optimal selection could be obtained. The work presented in this project involved the design, iteration and execution of eleven models. Due to models that either failed or proved too difficult to optimise, the total number of models including those not presented is far greater. This put considerable strain to present the best work possible for the problem statement. In future work, a more focused, tighter methodology would yield better research.

7. Bibliography

Adithya Balaji and Allen, A., 2018. Benchmarking Automatic Machine Learning Frameworks [Online]. arXiv.org [Online]. Available from: https://doi.org/10.48550/arxiv.1808.06492.

Akhtar, M.S., Garg, T. and Ekbal, A., 2020. Multi-task learning for aspect term extraction and aspect sentiment classification. Neurocomputing [Online], 398, pp.247–256. Available from: https://doi.org/10.1016/j.neucom.2020.02.093.

Akyildirim, E. 2023. Forecasting mid-price movement of Bitcoin futures using machine learning. [Online]. Annals of Operations Research [Online], 330(1/2), pp.553–585. Available from: https://doi.org/10.1007/s10479-021-04205-x.

Alsharef, A., Sonia, S., Kumar, K. and Iwendi, C., 2022. Time Series Data Modeling Using Advanced Machine Learning and AutoML [Online]. Sustainability. [Online], 14(22). Available from: https://doi.org/10.3390/su142215292.

Alvarez-Diaz, M., 2008. Exchange rates forecasting: local or global methods? [Online]. Applied economics [Online], 40(15), pp.1969–1984. Available from: https://doi.org/10.1080/00036840600905308.

anychart.com (2024). AnyChart Documentation, Technical Indicators Mathematical Description. Available from https://docs.anychart.com/Stock_Charts/Technical_Indicators/Mathematical_Description#overview

Atsalakis, G.S. and Valavanis, K.P., 2009. Surveying stock market forecasting techniques – Part II: Soft computing methods [Online]. Expert systems with applications [Online], 36(3), pp.5932–5941. Available from: https://doi.org/10.1016/j.eswa.2008.07.006.

Ayodele A., Aderemi A. 2014. Stock Price Prediction Using the ARIMA Model [Online]. Available from: https://doi.org/10.1109/uksim.2014.67.

Ballings, M., Van den Poel, D., Hespeels, N. and Gryp, R., 2015. Evaluating multiple classifiers for stock price direction prediction. Expert systems with applications [Online], 42(20), pp.7046–7056. Available from: https://doi.org/10.1016/j.eswa.2015.05.013.
https://uk.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html

Basher, S.A. and Sadorsky, P., 2022. Forecasting Bitcoin price direction with random forests: How important are interest rates, inflation, and market volatility? [Online]. Machine learning with applications [Online], 9, p.100355. Available from: https://doi.org/10.1016/j.mlwa.2022.100355.

Bender, G., Kindermans, P.-J., Zoph, B., Vasudevan, V. and Le, Q., 2018. Understanding and Simplifying One-Shot Architecture Search. In: Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80.

Bengio, Y., Simard, P. and Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), pp.157-166.

Bergstra, J., 2012. Random search for hyper-parameter optimization. Journal of machine learning research : JMLR., 13, pp.281–305.

Bińkowski, M., Gautier Marti and Donnat, P., 2017. Autoregressive Convolutional Neural Networks for Asynchronous Time Series. Proceedings of The 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 2018. Available from: https://doi.org/10.48550/arxiv.1703.04122.

Bou-Hamad, I. and Jamali, I., 2020. Forecasting financial time-series using data mining models: A simulation study [Online]. Research in international business and finance [Online], 51, p.101072. Available from: https://doi.org/10.1016/j.ribaf.2019.101072.

Breiman, L., 2001. Random Forests [Online]. Machine learning [Online], 45(1), pp.5–32. Available from: https://doi.org/10.1023/A:1010933404324.

Bukhari, A.H., Raja, M.A.Z., Sulaiman, M., Islam, S., Shoaib, M. and Kumam, P., 2020. Fractional Neuro-Sequential ARFIMA-LSTM for Financial Market Forecasting [Online]. IEEE access [Online], pp.1–1. Available from: https://doi.org/10.1109/ACCESS.2020.2985763.

Bybit.com. 2023. The 11 Best Technical Indicators for Cryptocurrency Trading (2023). Available from https://learn.bybit.com/indicators/best-technical-indicators/.

Carta, S., Ferreira, A., Podda, A.S., Reforgiato Recupero, D. and Sanna, A., 2021. Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. Expert systems with applications [Online], 164, p.113820. Available from: https://doi.org/10.1016/j.eswa.2020.113820.

Chan, E., 2013. Algorithmic Trading [Online]. 1st ed. Newark: Wiley. Available from: https://doi.org/10.1002/9781118676998.

Chen, C., Zhang, P., Liu, Y. and Liu, J., 2020a. Financial quantitative investment using convolutional neural network and deep learning technology [Online]. Neurocomputing (Amsterdam) [Online], 390, pp.384–390. Available from:

https://doi.org/10.1016/j.neucom.2019.09.092.

Chen, Z., Li, C. and Sun, W., 2020b. Bitcoin price prediction using machine learning: An approach to sample dimension engineering. Journal of Computational and Applied Mathematics, 365, p.112395.

Cherati, M., Haeri, A. and Ghannadpour, S.F., 2021. Cryptocurrency direction forecasting using deep learning algorithms. Journal of Statistical Computation and Simulation, 91(12), pp.2475-2489.

Chollet, F and others. 2015. Keras. Version (3.0). Available from https://keras.io/.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M. and Yang, S., 2016. AdaNet: Adaptive Structural Learning of Artificial Neural Networks [Online]. arXiv (Cornell University) [Online]. Available from: https://doi.org/10.48550/arxiv.1607.01097.

Cristianini, N., Ricci, E. (2008). Support Vector Machines. In: Kao, MY. (eds) Encyclopedia of Algorithms. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30162-4_415

Cruz, L.F.S.A. and Silva, D.F., 2021. Financial Time Series Forecasting Enriched with Textual Information [Online]. 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp.33–390. Available from: https://doi.org/10.1109/ICMLA52953.2021.00066.

Dang-Nhu, R., 2020. Adversarial attacks on probabilistic autoregressive forecasting models. 37th International Conference on Machine Learning, ICML 2020, PartF168147-3, pp.2334–2343.

De Jong, K., Fogel, D.B. and Schwefel, H.P. 1997. A History of Evolutionary Computation. In: Th. Bäck, D.B. Fogel and Z. Michalewicz, eds. Handbook of Evolutionary Computation. Oxford: Oxford University Press.

Ding, H., Xiao, Y. and Yue, J. 2005. Adaptive training of radial basis function networks using particle swarm optimization algorithm. Lecture Notes in Computer Science 3610, 119–128.

Elsken, T., Metzen, J.H. and Hutter, F., 2018. Neural Architecture Search: A Survey [Online].Journal of Machine Learning Research 20 (2019) 1-21. Available from: https://doi.org/10.48550/arxiv.1808.05377.

Fang, F., Ventre, C., Basios, M., Kanthan, L., Martinez-Rego, D., Wu, F. and Li, L., 2022. Cryptocurrency trading: a comprehensive survey. Financial innovation (Heidelberg) [Online], 8(1), pp.1–59. Available from: https://doi.org/10.1186/s40854-021-00321-6.

Fischer, T. and Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions [Online]. European journal of operational research [Online], 270(2), pp.654–669. Available from: https://doi.org/10.1016/j.ejor.2017.11.054.

Francq, C. and Zakoian, J.-M., 2010. GARCH Models : Structure, Statistical Inference and Financial Applications. 2nd ed. New York: John Wiley & Sons, Incorporated.

Gad, S.C., 2010. 3.13 - Statistical Methods in Toxicology. In: C.A. McQueen, ed. Comprehensive Toxicology (Second Edition). 2nd ed. Oxford: Elsevier, pp.183-197. Available at: https://www.sciencedirect.com/science/article/pii/B9780080468846003201 [Accessed 13/12/24].

Galeshchuk, S., 2016. Neural networks performance in exchange rate prediction. Neurocomputing [Online], 172, pp.446–452. Available from: https://doi.org/10.1016/j.neucom.2015.03.100.

Gallup. (2023). Share of adults investing money in the stock market in the United States from 1999 to 2023. Statista. Statista Inc.. Accessed: May 01, 2024. https://www.statista.com/statistics/270034/percentage-of-us-adults-to-have-money-invested-in-the-stock-market/

Garcia, D. and Schweitzer, F., 2015. Social signals and algorithmic trading of Bitcoin [Online]. Royal Society open science [Online], 2(9), pp.150288–150288. Available from: https://doi.org/10.1098/rsos.150288.

GeeksforGeeks. (2023). Support Vector Machine Algorithm. Available at: https://www.geeksforgeeks.org/support-vector-machine-algorithm/ (Accessed: 24 April 2024).

Georg M. 2013. Forecastable component analysis. In Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28 (ICML'13). JMLR.org, II–64–II–72.

Gijsbers, P., LeDell, E., Janek, T., Poirier, S., Bischl, B. and Vanschoren, J., 2019. An Open Source AutoML Benchmark [Online]. arXiv.org. Ithaca: Cornell University Library, arXiv.org. Available from: https://doi.org/10.48550/arxiv.1907.00909.

Glaffig, C.H. 2018. 'Some Basic Aspects of Deep Learning for Applications to Financial Trading’. Mutual Funds.

Goodfellow, I, Bengio, Y, & Courville, A. 2016, Deep Learning, MIT Press, Cambridge. Available from: ProQuest Ebook Central.

Google Colab. (2024a). Google Colab: Collaborative notebooking for Python. Available from https://colab.research.google.com/

Google Cloud. (2024b). Cloud TPU Documentation. Mountain View, CA. Available from https://cloud.google.com/tpu/docs

Grajzl, P., Murrell, P. 2020. A machine-learning history of English caselaw and legal ideas prior to the Industrial Revolution I: generating and interpreting the estimates. Journal of Institutional Economics.

Guyon, I. Elisseeff, A,. 2003. An introduction to variable and feature selection. Journal of machine learning research : JMLR., 3, pp.1157–1182.

H2O.ai. (2024). H2O Automated Machine Learning (AutoML) (Version 3.36.0.3) [Software]. Mountain View, CA: https://h2o.ai/

H2O.ai. (2024a). H2O Cryptocurrency Trading [Online]. Mountain View, CA Available from: https://h2o.ai/solutions/use-case/cryptocurrency-trading/

H20.ai. (2024b). Welcome to H20 3. [Documentation]. Mountain View, CA: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html

Han Xu, Pengfei He, Jie Ren, Yuxuan Wan, Zitao Liu, Hui Liu, and Jiliang Tang. 2023. Probabilistic categorical adversarial attack and adversarial training. In Proceedings of the 40th International Conference on Machine Learning (ICML'23), Vol. 202. JMLR.org, Article 1600, 38428–38442.

Hamayel, M.J. and Amani Yousef Owda, 2021. A Novel Cryptocurrency Price Prediction Model Using GRU, LSTM and bi-LSTM Machine Learning Algorithms [Online]. AI (Basel) [Online], 2(4), p.477. Available from: https://doi.org/10.3390/ai2040030.

Hanussek, M., Blohm, M. and Kintz, M., 2020. Can AutoML outperform humans? An evaluation on popular OpenML datasets using AutoML Benchmark [Online]. ACM International Conference Proceeding Series. pp.29–32. Available from: https://doi.org/10.1145/3448326.3448353.

Hasson, H., Maddix, D.C., Wang, Y., Gupta, G. and Park, Y., 2023. Theoretical Guarantees of Learning Ensembling Strategies with Applications to Time Series Forecasting [Online]. Proceedings of the 40th International Conference on Machine Learning. Available from: https://doi.org/10.48550/arxiv.2305.15786.

Hayes, A. 2022. Heteroscedasticity Definition: Simple Meaning and Types Explained. Investopedia. Available from https://www.investopedia.com/terms/h/heteroskedasticity.asp#:~:text=Investopedia / Joules Garcia-,What Is Heteroskedasticity?,periods, are non-constant.

Hilbert, M. and Darmon, D., 2020. How complexity and uncertainty grew with algorithmic trading. Entropy (Basel, Switzerland) [Online], 22(5), p.499. Available from: https://doi.org/10.3390/E22050499.

Hrnjica, B. and Bonacci, O., 2019. Lake Level Prediction using Feed Forward and Recurrent

Neural Networks [Online]. Water resources management [Online], 33(7), pp.2471–2484. Available from: https://doi.org/10.1007/s11269-019-02255-2.

Hu, H., Tang, L., Zhang, S. and Wang, H., 2018. Predicting the direction of stock markets using optimized neural networks with Google Trends. Neurocomputing [Online], 285, pp.188–195. Available from: https://doi.org/10.1016/j.neucom.2018.01.038.

Huang, J.-Z., Huang, W. and Ni, J., 2019. Predicting bitcoin returns using high-dimensional technical indicators [Online]. The Journal of finance and data science [Online], 5(3), pp.140–155. Available from: https://doi.org/10.1016/j.jfds.2018.10.001.

Hull, I. 2020. Machine Learning for Economics and Finance in TensorFlow 2 [Online]. 1st Edition. Berkeley, CA: Apress L. P. Available from: https://doi.org/10.1007/978-1-4842-6373-0.

Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. Neural computation. [Online], 9(8), pp.1735–1780. Available from: https://doi.org/10.1162/neco.1997.9.8.1735.

Huang, Y., Gao, Y., Gan, Y. and Ye, M., 2021. A new financial data forecasting model using genetic algorithm and long short-term memory network [Online]. Neurocomputing [Online], 425, pp.207–218. Available from: https://doi.org/10.1016/j.neucom.2020.04.086.

Kazem, A., Sharifi, E., Hussain, F.K., Saberi, M. and Hussain, O.K., 2013. Support vector regression with chaos-based firefly algorithm for stock market price forecasting [Online]. Applied soft computing [Online], 13(2), pp.947–958. Available from: https://doi.org/10.1016/j.asoc.2012.09.024.

Kelly, B. and Xiu, D., 2023. Financial Machine Learning [Online]. Foundations and trends in finance [Online], 13(3-4), pp.205–363. Available from: https://doi.org/10.1561/0500000064.

Kim, K., 2003. Financial time series forecasting using support vector machines. Neurocomputing [Online], 55(1-2), pp.307–319. Available from: https://doi.org/10.1016/S0925-2312(03)00372-2.

Kim, K. 2007, Electronic and Algorithmic Trading Technology : The Complete Guide, Elsevier Science & Technology, San Diego. Available from: ProQuest Ebook Central.

Kow, Y. (2017). Cryptocurrencies and their potential for large-crowd, cost-effective transactions in peer production. First Monday

Kumar, S., n.d. 2019. FORECASTING CRYPTOCURRENCY PRICES USING ARIMA AND NEURAL NETWORK: A COMPARATIVE STUDY. Journal of Prediction Markets, 13(2), pp.33–45.

Krauss, C., Do, X.A. and Huck, N., 2017. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500 [Online]. European journal of operational research [Online], 259(2), pp.689–702. Available from: https://doi.org/10.1016/j.ejor.2016.10.031.

Ladd, J. 2015. Tackling the risks of algorithmic trading [Online]. European Principal Traders Accusation. Available from: https://www.fia.org/epta/articles/tackling-risks-algorithmic-trading-0 [Accessed 11/01/234].

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning [Online]. Nature (London) [Online], 521(7553), pp.436–444. Available from: https://doi.org/10.1038/nature14539.

Lee, V.C.S. and Tshung Wong, H., 2007. A multivariate neuro-fuzzy system for foreign currency risk management decision making [Online]. Neurocomputing (Amsterdam) [Online], 70(4), pp.942–951. Available from: https://doi.org/10.1016/j.neucom.2006.10.025.

Lee, S.W. and Kim, H.Y., 2020. Stock market forecasting with super-high dimensional time-series data using ConvLSTM, trend sampling, and specialized data augmentation [Online]. Expert systems with applications [Online], 161. Available from: https://doi.org/10.1016/j.eswa.2020.113704.

Li, W., Qi, F., Tang, M. and Yu, Z., 2020. Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification [Online]. Neurocomputing (Amsterdam) [Online], 387, pp.63–77. Available from: https://doi.org/10.1016/j.neucom.2020.01.006.

Liashchynskyi, Petro and Liashchynskyi, Pavlo., 2019. Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS [Online]. arXiv.org [Online]. Available from: https://doi.org/10.48550/arxiv.1912.06059.

Livieris, I.E., Pintelas, E., Stavroyiannis, S. and Pintelas, P. 2020a. Ensemble Deep Learning Models for Forecasting Cryptocurrency Time-Series. Algorithms, 13, 121.

Livieris, I.E., Pintelas, E., Stavroyiannis, S.. Kotsilieris, T. and Pintelas, P. 2020b. Fundamental Research Questions and Proposals on Predicting Cryptocurrency Prices Using DNNs; Technical Report TR20-01; University of Patras: Patras, Greece. Available online: https://nemertes.lis.upatras.gr/jspui/bitstream/10889/13296/1/ TR01-20.pdf [Accessed 16/02/2024].

Livieris, I.E., Kiriakidou, N., Stavroyiannis, S. and Pintelas, P. 2021. An Advanced CNN-LSTM Model for Cryptocurrency Forecasting. Electronics , 10, 287.

Liu, H., Simonyan, K. and Yang, Y., 2018. DARTS: Differentiable Architecture Search [Online]. arXiv (Cornell University) [Online]. Available from: https://doi.org/10.48550/arxiv.1806.09055.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y. and Alsaadi, F.E., 2017. A survey of deep neural network architectures and their applications [Online]. Neurocomputing [Online], 234, pp.11–26. Available from: https://doi.org/10.1016/j.neucom.2016.12.038.

Lo, A.W., 2005. Reconciling efficient markets with behavioral finance: the adaptive markets hypothesis. Journal of investment consulting, 7(2), pp.21-44.

Long, W., Lu, Z. and Cui, L., 2019. Deep learning-based feature engineering for stock price movement prediction. Knowledge-based systems [Online], 164, pp.163–173. Available from: https://doi.org/10.1016/j.knosys.2018.10.034.

Manchanda, H. and Aggarwal, S., 2021. Forecasting cryptocurrency time series using adaboost-based ensemble learning techniques. In Innovations in Cyber Physical Systems: Select Proceedings of ICICPS 2020 (pp. 207-219). Springer Singapore.

Manjula, B.C.,Shilpa, B.S. and Sundaresh. M. 2022. Analysis of Cryptocurrency, Bitcoin and the Future. East Asian Journal of Multidisciplinary Research.

Markowitz, H.M., 1991. Portfolio selection : efficient diversification of investments. 2nd ed. Oxford: Blackwell.

McNally, S., Roche, J. and Caton, S. 2018. "Predicting the Price of Bitcoin Using Machine Learning," 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), Cambridge, UK, 2018, pp. 339-343, doi: 10.1109/PDP2018.2018.00060.

Meese, R.A. and Rogoff, K., 1983. Empirical exchange rate models of the seventies: Do they fit out of sample? [Online]. Journal of international economics [Online], 14(1), pp.3–24. Available from: https://doi.org/10.1016/0022-1996(83)90017-X.

Miao, K., Han, T., Yao, Y., Lu, H., Chen, P., Wang, B. and Zhang, J., 2020. Application of LSTM for short term fog forecasting based on meteorological elements [Online]. Neurocomputing (Amsterdam) [Online], 408, pp.285–291. Available from: https://doi.org/10.1016/j.neucom.2019.12.129.

Montgomery, D.C., Peck, E.A. and Vining, G.G., 2021. Introduction to linear regression analysis / Douglas C. Montgomery, Elizabeth A. Peck, G. Geoffrey Vining. 5th edition. Hoboken, N.J.: Wiley

Moews, B., Herrmann, J.M. and Ibikunle, G., 2019. Lagged correlation-based deep learning for directional trend change prediction in financial time series [Online]. Expert systems with applications [Online], 120, pp.197–206. Available from: https://doi.org/10.1016/j.eswa.2018.11.027.

Murray, K., Rossi, A., Carraro, D. and Visentin, A., 2023. On Forecasting Cryptocurrency Prices: A Comparison of Machine Learning, Deep Learning, and Ensembles [Online]. Forecasting [Online], 5(1), pp.196–209. Available from:

https://doi.org/10.3390/forecast5010010.

Nguyen, D. And Chan, K. 2024. Cryptocurrency trading: A systematic mapping study. International Journal of Information Management Data Insights, 4(2), p.100240. Available at: https://doi.org/10.1016/j.jjimei.2024.100240.

Nti, I.K., Adekoya, A.F. and Weyori, B.A. 2020. A comprehensive evaluation of ensemble learning for stock-market prediction. Journal of Big Data, 7(20).

OKX.com. 2024. The 8 best indicators for crypto trading in 2024. Available from https://www.okx.com/learn/best-crypto-indicators-trading.

Ozbayoglu, A.M., Gudelek, M.U. and Sezer, O.B., 2020. Deep learning for financial applications : A survey [Online]. Applied soft computing [Online], 93, p.106384. Available from: https://doi.org/10.1016/j.asoc.2020.106384.

Paldino, G.M., De Stefani, J., De Caro, F., & Bontempi, G. (2021). Does AutoML Outperform Naive Forecasting? Engineering Proceedings, 5(36). https://doi.org/10.3390/engproc2021005036Balaji,

Patel, M.M., Tanwar, S., Gupta, R. and Kumar, N., 2020. A Deep Learning-based Cryptocurrency Price Prediction Scheme for Financial Institutions [Online]. Journal of information security and applications [Online], 55, p.102583. Available from: https://doi.org/10.1016/j.jisa.2020.102583.

Pham, H., Guan, M.Y., Barret Zoph, Le, Q.V. and Dean, J., 2018. Efficient Neural Architecture Search via Parameter Sharing [Online]. arXiv (Cornell University) [Online]. Available from: https://doi.org/10.48550/arxiv.1802.03268.

Piccialli, F., Giampaolo, F., Salvi, A. and Cuomo, S., 2021. A robust ensemble technique in forecasting workload of local healthcare departments [Online]. Neurocomputing (Amsterdam) [Online], 444, pp.69–78. Available from: https://doi.org/10.1016/j.neucom.2020.02.138.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. Nature (London) [Online], 323(6088), pp.533–536. Available from: https://doi.org/10.1038/323533a0.

Salomone, R., Quiroz, M., Kohn, R., Villani, M. and Tran, M.-N., 2019. Spectral Subsampling MCMC for Stationary Time Series In Proceedings of the 37th International Conference on Machine Learning (ICML'20), Vol. 119. JMLR.org, Article 783, 8449–8458.

Sather, M. 2023. Taking Stock of Market Uncertainty. [Online] Available at: https://www.luc.edu/quinlan/about/newsandevents/archive/taking-stock-of-market-uncertainty.shtml [Accessed 8 Jan 2024].

Schmitt, W., Masters, B. and Chipolina, A. 2024. SEC approves first spot bitcoin ETFs in boost to crypto advocates. Financial Times. 11/01/2024. Available from: https://www.ft.com/content/443b2589-0a4a-48ef-872e-3cd52b1b297d [Accessed 16/02/2024].

Sermpinis, G., Theofilatos, K., Karathanasopoulos, A., Georgopoulos, E.F. and Dunis, C. 2012. Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and Particle Swarm Optimization. European Journal of Operational Research.

Shadbolt, J., Taylor, J. and Taylor, J. G. 2002. Neural Networks and the Financial Markets : Predicting, Combining and Portfolio Optimisation. 1st ed. 2002. London: Springer London : Imprint: Springer (Perspectives in Neural Computing).

Shynkevich, Y., McGinnity, T., Coleman, S.A., Belatreche, A. and Li, Y., 2017. Forecasting price movements using technical indicators: Investigating the impact of varying input window length. Neurocomputing [Online], 264, pp.71–88. Available from: https://doi.org/10.1016/j.neucom.2016.11.095.

Siji, G.C. and Sumathi, B., 2020. Grid Search Tuning of Hyperparameters in Random Forest Classifier for Customer Feedback Sentiment Prediction [Online]. International journal of advanced computer science & applications [Online], 11(9). Available from: https://doi.org/10.14569/IJACSA.2020.0110920.

Song, Q., Liu, A. and Yang, S.Y., 2017. Stock portfolio selection using learning-to-rank algorithms with news sentiment. Neurocomputing [Online], 264, pp.20–28. Available from: https://doi.org/10.1016/j.neucom.2017.02.097.

Srivastava, N., 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of machine learning research : JMLR., 15, pp.1929–1958. (https://dl.acm.org/doi/10.5555/2627435.2670313)

StackExchange. 2013. Available from https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network

Statista. (2023). Number of identity-verified cryptoasset users from 2016 to November 2023 (in millions). Statista. Statista Inc.. Accessed: May 01, 2024. https://www.statista.com/statistics/1202503/global-cryptocurrency-user-base/

Tang, Y., Song, Z., Zhu, Y., Yuan, H., Hou, M., Ji, J., Tang, C. and Li, J., 2022. A survey on machine learning models for financial time series forecasting [Online]. Neurocomputing (Amsterdam) [Online], 512, pp.363–380. Available from: https://doi.org/10.1016/j.neucom.2022.09.003.

TensorFlow. 2024. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Version (2.16). Available from https://www.tensorflow.org/.

Tripathi, B. and Sharma, R.K., 2023. Modeling Bitcoin Prices using Signal Processing Methods, Bayesian Optimization, and Deep Neural Networks [Online]. Computational economics [Online], 62(4), pp.1919–1945. Available from: https://doi.org/10.1007/s10614-022-10325-8.

Tschorsch, F. and Scheuermann, B. 2016. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. IEEE Communications Surveys & Tutorials.

Urquhart, A. and Hudson, R., 2013. Efficient or adaptive markets? Evidence from major stock markets using very long run historic data [Online]. International review of financial analysis [Online], 28, pp.130–142. Available from: https://doi.org/10.1016/j.irfa.2013.03.005.

V. Derbentsev, Vitalina Babenko, K. Khrustalev, H. Obruch and S. Khrustalova, 2021. Comparative Performance of Machine Learning Ensemble Algorithms for Forecasting Cryptocurrency Prices [Online]. International journal of engineering (Tehran) [Online], 34(1). Available from: https://doi.org/10.5829/ije.2021.34.01a.16.

Vapnik, V. N. and Vapnik, Vladimir Naumovich, 1995. The nature of statistical learning theory. New York: Springer.

Virk DS. 2017. Prediction of bitcoin price using data mining. Master's thesis, National College of Ireland

Vlachas, P.R., Byeon, W., Wan, Z.Y., Sapsis, T.P. and Koumoutsakos, P., 2018. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks [Online]. Proceedings of the Royal Society. A, Mathematical, physical, and engineering sciences [Online], 474(2213), pp.20170844–20170844. Available from: https://doi.org/10.1098/rspa.2017.0844.

Wang, B., Huang, H. and Wang, X., 2012. A novel text mining approach to financial time series forecasting. Neurocomputing [Online], 83, pp.136–145. Available from: https://doi.org/10.1016/j.neucom.2011.12.013.

Wang, X., Jin, Y., Schmitt, S. and Olhofer, M., 2023. Recent Advances in Bayesian Optimization [Online]. ACM computing surveys [Online], 55(13s), pp.1–36. Available from: https://doi.org/10.1145/3582078.

Weill, C., 2018. Introducing AdaNet: Fast and Flexible AutoML with Learning Guarantees [Online]. Google Research. Available from: https://blog.research.google/2018/10/introducing-adanet-fast-and-flexible.html [11.03.24].

Weill, C., Gonzalvo, J., Kuznetsov, V., Yang, S., Yak, S., Mazzawi, H., Hotaj, E., Jerfel, G., Macko, V., Adlam, B., Mohri, M. and Cortes, C., 2019. AdaNet: A Scalable and Flexible Framework for Automatically Learning Ensembles [Online]. arXiv.org [Online]. Available from: https://doi.org/10.48550/arxiv.1905.00080.

Weill, C. 2020. adanet/README.md. Google. Available from: https://github.com/tensorflow/adanet/blob/master/README.md

Xu, C. and Xie, Y., 2023. Sequential Predictive Conformal Inference for Time Series. In: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett, eds. Proceedings of the 40th International Conference on Machine Learning. [online] Proceedings of Machine Learning Research, Vol. 202, pp.38707-38727. Available at: https://proceedings.mlr.press/v202/xu23r.html [25/01/24].

Yahoo Finance. 2024. Ethereum USD (ETH-USD). Available from https://finance.yahoo.com/quote/ETH-USD?.tsrc=fin-

Yang, C., Zhai, J. and Tao, G., 2020. Deep Learning for Price Movement Prediction Using Convolutional Neural Network and Long Short-Term Memory. Mathematical problems in engineering [Online], 2020, pp.1–13. Available from: https://doi.org/10.1155/2020/2746845.

Yang, S.Y., Mo, S.Y.K., Liu, A. and Kirilenko, A.A., 2017. Genetic programming optimization for a sentiment feedback strength based trading strategy. Neurocomputing (Amsterdam) [Online], 264, pp.29–41. Available from: https://doi.org/10.1016/j.neucom.2016.10.103.

Yao, J. and Tan, C.L., 2000. A case study on using neural networks to perform technical forecasting of forex. Neurocomputing [Online], 34(1-4), pp.79–98. Available from: https://doi.org/10.1016/S0925-2312(00)00300-3.

Zatwarnicki, M., Zatwarnicki, K. and Stolarski, P., 2023. Effectiveness of the Relative Strength Index Signals in Timing the Cryptocurrency Market [Online]. Sensors (Basel, Switzerland) [Online], 23(3), p.1664. Available from: https://doi.org/10.3390/s23031664.

Zhan, Z.-H., Li, J.-Y. and Zhang, J., 2022. Evolutionary deep learning: A survey [Online]. Neurocomputing (Amsterdam) [Online], 483, pp.42–58. Available from: https://doi.org/10.1016/j.neucom.2022.01.099.
Zhang, X., Liang, X., Zhiyuli, A., Zhang, S., Xu, R. and Wu, B., 2019. AT-LSTM: An Attention-based LSTM Model for Financial Time Series Prediction [Online]. IOP conference series. Materials Science and Engineering [Online], 569(5), p.52037. Available from: https://doi.org/10.1088/1757-899X/569/5/052037.

Zhang, X., He, K. and Bao, Y., 2021. Error-feedback stochastic modeling strategy for time series forecasting with convolutional neural networks [Online]. Neurocomputing (Amsterdam) [Online], 459, pp.234–248. Available from: https://doi.org/10.1016/j.neucom.2021.06.051.

Zhang, J. and Zhang, C., 2022. Do cryptocurrency markets react to issuer sentiments? Evidence from Twitter [Online]. Research in international business and finance [Online], 61, p.101656. Available from: https://doi.org/10.1016/j.ribaf.2022.101656.

Zöller, M.A. and Huber, M.F., 2021. Benchmark and Survey of Automated Machine Learning Frameworks [Online]. The Journal of artificial intelligence research [Online], 70, pp.409–472. Available from: https://doi.org/10.1613/JAIR.1.11854.

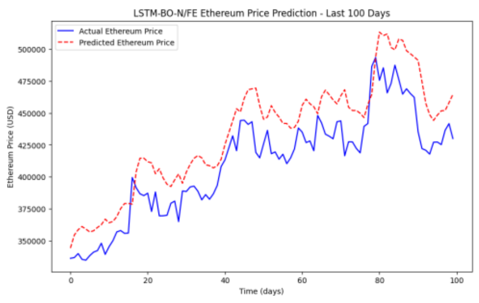
Appendix

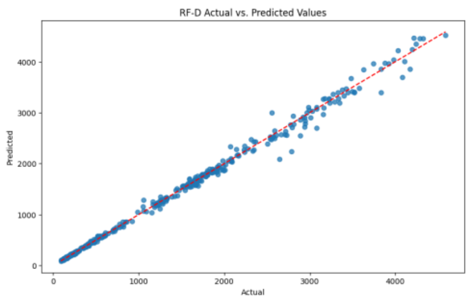
A. Model Parameters

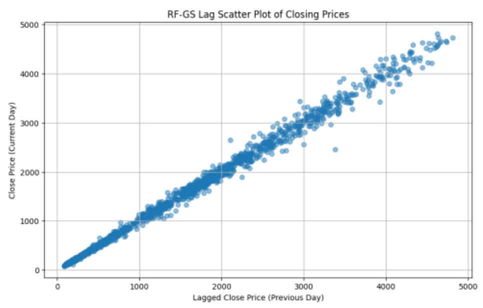
Model			**rs Used**
	sModel		• p:4 • d:3 • q:5
	sModel		• p:4 • d:4 • q:3
- D	t-Learn		• Number of estimators: 100 • Criterion: gini • Max depth: None • Min samples split: 2 • Min samples leaf: 1 • Max features: auto
- GS	t-Learn		• Number of estimators: 100 • Criterion: gini • Max depth: None • Min samples split: 2 • Min samples leaf: 1 • Max features: auto
ing - D	t-Learn		• Number of estimators: 100 • Learning rate: 0.1 • Max depth: 3 • Min samples split: 2
ing - GS	t-Learn		• Number of estimators: 100 • Learning rate: 0.05 • Max depth: 5 • Min samples split: 6
Regression - GS	t-Learn		• C: 50 • Kernel: rbf • Degree: 3 • gamma: scale • Epsilon: 0.1
	orFlow	vo LSTM Layers	• First LSTM layer units: 352 • First dropout rate: 0.0-0.05 • Second LSTM layer units: 352 • Second dropout rate: 0.0-0.05 • Dense layer units: 1 • Optimiser: Adam • Loss function: mse

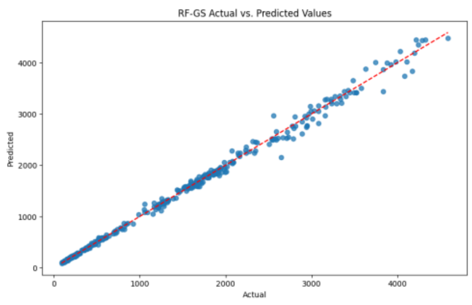
E	ιorFlow	vo LSTM Layers	• First LSTM layer units: 512 • First dropout rate: 0.0-0.05 • Second LSTM layer units: 512 • Second dropout rate: 0.0-0.05 • Dense layer units: 1 • Optimiser: Adam • Loss function: mse
		)ne CNN Layer Bidirectional Layer ıttention Mechanism ree Dense Layers	• First conv kernel size: 3 • First dropout rate: 0.25 • Bidirectional LSTM layer units: 50 • LSTM activation: ReLU • Second dropout rate: 0.25 • First dense layer units: 16 • Second dense layer units: 16 • Learning rate: adaptive (using ReduceLROnPlateau) • Optimizer: adam • Loss function: mse
	2O.ai	nt Boosting Estimator	• Trees: 90 • Min depth: 6 • Max depth: 6 • Min leaves: 42 • Max leaves: 64 • Mean leaves: 60.34

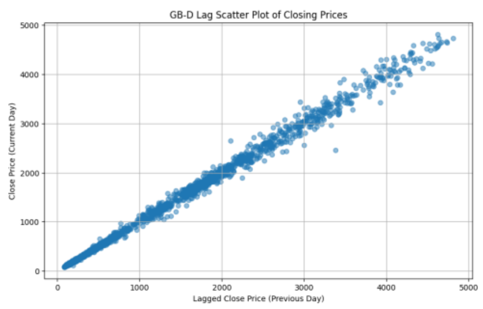
B. Model Plots

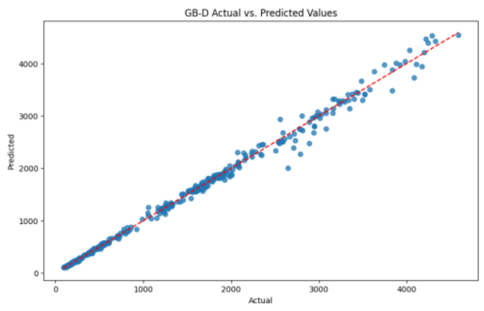


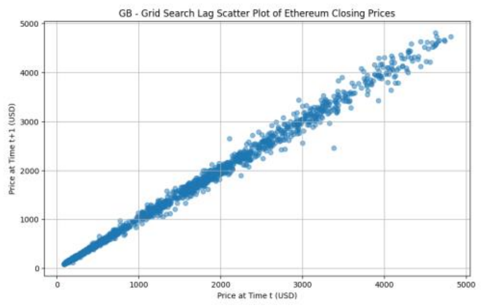


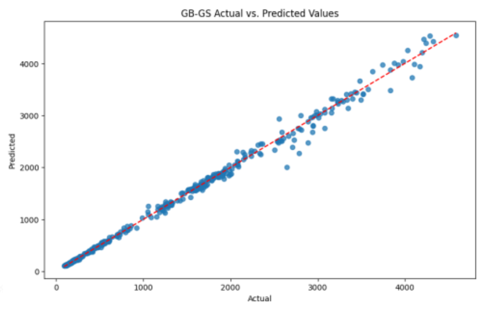

RF-GS Lag Scatter Plot of Closing Prices
Close Price (Current Day)
Lagged Close Price (Previous Day)

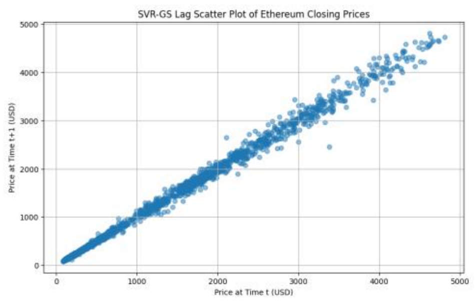

RF-GS Actual vs. Predicted Values
Predicted
Actual

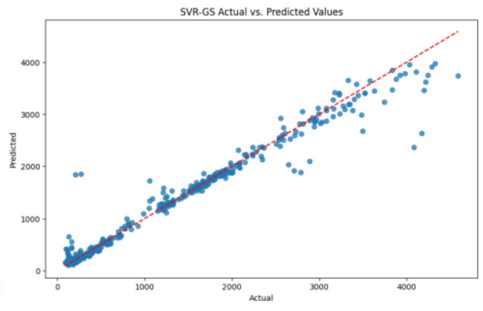

GB-D Lag Scatter Plot of Closing Prices
Close Price (Current Day)
Lagged Close Price (Previous Day)

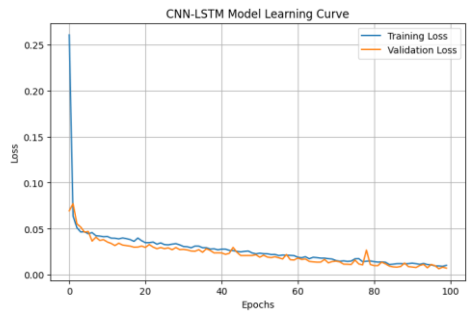

GB-D Actual vs. Predicted Values
Predicted
Actual


GB - Grid Search Lag Scatter Plot of Ethereum Closing Prices
Price at Time t+1 (USD)
Price at Time t (USD)

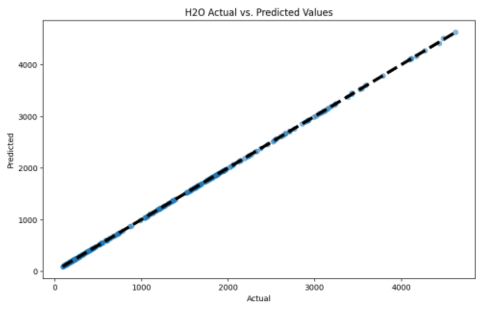

GB-GS Actual vs. Predicted Values
Predicted
Actual

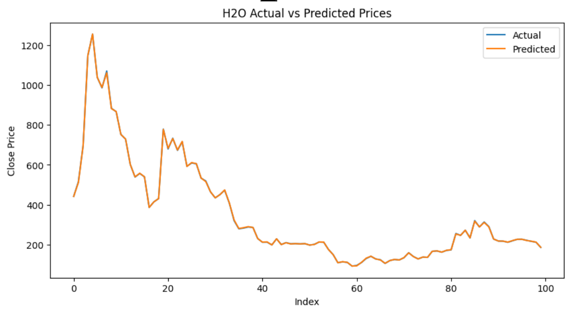

SVR-GS Lag Scatter Plot of Ethereum Closing Prices
Price at Time t+1 (USD)
Price at Time t (USD)

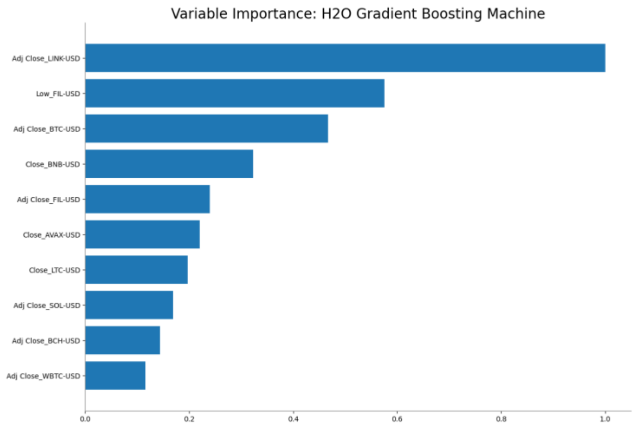

SVR-GS Actual vs. Predicted Values
Predicted
Actual


CNN-LSTM Model Learning Curve
Training Loss
Validation Loss
Loss
Epochs

C. H2O AutoML Plots/Graph







D. Alternative Assets

Asset	Web3 Narrative	Description
BTC-USD (Bitcoin)	Digital Gold	The first cryptocurrency, serving as a decentralized digital currency without a central bank.
BNB-USD (BNB)	Utility and Governance Token	Originally created as a utility token for the Binance exchange, now powers the Binance Smart Chain.
SOL-USD (Solana)	High Performance Blockchain	A high-speed blockchain supporting decentralized apps and crypto-currencies with fast transaction times.
XRP-USD (XRP)	Banking and Payment Bridge	Designed for fast, inexpensive international transactions, aiming to facilitate cross-border payments for banks.
STETH-USD (Lido Staked ETH)	Ethereum Staking Solution	Represents Ethereum staked through the Lido platform, allowing users to earn staking rewards.
ADA-USD (Cardano)	Third-Generation Cryptocurrency	A blockchain platform for smart contracts, known for its scientific approach and emphasis on security.
DOGE-USD (Dogecoin)	Meme to Currency	Initially started as a joke, now a widely recognized cryptocurrency with a strong community.
SHIB-USD (Shiba Inu)	Decentralized Meme Token	A meme token aiming to be an Ethereum-based counterpart to Dogecoin, with a committed and vibrant community.
AVAX-USD (Avalanche)	Layer 1 Blockchain Platform	A blockchain platform designed for scalability, supporting decentralized applications and custom blockchain networks.
DOT-USD (Polkadot)	Interoperability and Scalability	Aims to enable different blockchains to transfer messages and value in a trust-free fashion; supports multiple parallel blockchains.
WTRX-USD (Wrapped TRON)	TRON's Token on Ethereum	Represents TRON's native token on the Ethereum blockchain, facilitating its use in the Ethereum ecosystem.
TRX-USD (TRON)	Decentralized Internet	A blockchain-based operating system designed to make blockchain technology suitable for daily use.
LINK-USD (Chainlink)	Decentralized Oracle Network	A decentralized network of oracles that enables smart contracts to securely interact with external data feeds, events, and payment methods.
MATIC-USD (Polygon)	Ethereum Scaling and Infrastructure Development	A platform for building and connecting Ethereum-compatible blockchain networks, aiming to solve scalability issues.
WBTC-USD (Wrapped Bitcoin)	Bitcoin on Ethereum	Represents Bitcoin in the Ethereum ecosystem, facilitating its use in Ethereum's decentralized applications.
TON11419-USD (Toncoin)	High-Speed, Scalable Blockchain	A blockchain focused on high transaction speeds and scalability, originally initiated by the Telegram team.

BCH-USD (Bitcoin Cash)	Bitcoin Fork for Scalability	A cryptocurrency created as a fork of Bitcoin, with a focus on larger block sizes to process more transactions.
UNI7083-USD (Uniswap)	Decentralized Trading Protocol	A leading decentralized crypto exchange protocol on Ethereum, facilitating automated trading of decentralized finance (DeFi) tokens.
LTC-USD (Litecoin)	Silver to Bitcoin's Gold	A peer-to-peer cryptocurrency, created as a lighter alternative to Bitcoin, aiming for faster transaction times.
ICP-USD (Internet Computer)	Reimagining the Internet	Aims to extend the internet with serverless cloud functionality, allowing smart contracts to run at web speed.
DAI-USD (Dai)	Decentralized Stablecoin	A stablecoin pegged to the US dollar through smart contracts.
FIL-USD (Filecoin)	Decentralized Storage Network	A decentralized storage system aimed at making the web more secure and efficient with a blockchain-based storage network.

E. CNN-LSTM Change Log

Base parameters	
CNN filters	64
Kernels	3
Activation	relu
Reguliser	L1 / L2
Pool size	2
Dropout	0.05
LSTM units	50
Activation	relu
Reguliser	L1 / L2
Dropout	0.05
Dense units	64

Parameter	Change	Reason	Evaluation Results					
			Accuracy	MSE	RMSE	MAE	MAPE	R2
	None	NA	50%	376,955	$613.00	$526.00	28.70%	-3.12
LSTM units	50 -> 100	Increase complexity	53%	376,955	$613.00	$526.00	28.70%	-3.12
CNN filters	64 -> 128	Increase complexity	50%	964,515	$982.00	$872.00	48%	-9.1
LSTM units	100 -> 50							
CNN filters	128 -> 32							
Dense units	64 -> 32	Reduce complexity.	51%	201,306	$448.00	$411.00	22%	-1.2
LSTM layer	Add 1	Increase complexity	50%	231,519	$481.00	$429.00	23%	-1.5
Dropout	0.05 -> 0.1							
LSTM layer	Remove 1	Increase robustness, reset layers.	55%	332,465	$576.00	$537.00	30%	-2.6
n_steps	20 -> 50	Expand historical context for input. Test if model requires longer dependecies	50%	242,418	$492.00	$439.00	23%	-2.29
n_steps	50 -> 16	Reduce historical input. Test if model requires shorter dependencies.	53%	134,438	$366.00	$320.00	17%	-0.4
n_steps	16 -> 10	" "	47%	120,680	$347.00	$299.00	16%	-0.23
LSTM units	50 -> 100							
CNN filters	32 -> 64							
Dense units	32 -> 64	Increase complexity against lower n_inputs for observation.	49%	368,423	$606.00	$567.00	31%	-2.8
LSTM units	100 -> 50							
CNN filters	128 -> 32							
Dense units	64 -> 32							
n_steps	10 -> 20	Revert back to previous settings and further reduce input for observation.	55%	102,866	$320.00	$280.00	15%	-0.025
LSTM units	50 -> 25	Reduce complexity.	55%	117,652	$343.00	$300.00	16%	-0.17
LSTM units	25 -> 50							
Dropout	01 -> 0.25	Increase complexity against further robustness.	50%	42,224	$205.00	$158.00	8%	0.58
Dropout (lstm layer)	0.25 -> 0.5	Follow typical set up with higher dropout for hidden layers.	50%	135,363	$367.00	$318.00	16%	-0.35
Dropout	Set all to 0.5	Revert to previous settings.	45%	1,270,185	$1,127.00	$1,082.00	60%	-11.7
Dropout	Reset all to 0.25							
LSTM units	25 -> 50							
CNN filters	32 -> 16							
Dense units	32 -> 16	Reduce robustness against increased complexity.	51%	41,616	$204.00	$147.00	10%	0.54
LSTM units	50 -> 40							
n_steps	20 -> 30	Reduce complexity set against larger window of inputs.	50%	108,204	$328.00	$292.00	16%	-0.25
CNN kernel size	4 -> 2	Focus on smaller, local features.	52%	54,854	$234.00	$203.00	11%	0.45
Reset CNN Kernel	2 -> 3							
LSTM activation function	relu -> sigmoid	Explore other functions more typically applied to layers.	54%	165,901	$407.00	$379.00	20%	-0.66
Fully connected layer activation functions (respectively)	relu -> sigmoid							
	relu -> tanh	" "	50%	237,955	$487.00	$375.00	19%	-1.4
Activation functions	Reset all to relu							
LSTM units	40 -> 256	Greatly increase complexity, exploratory.	50%	72,328	$268.00	$235.00	12%	0.27
n_steps	30 -> 100	Match increasing complexity with input size.	51%	86,111	$293.00	$254.00	13%	-0.43