Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK

The University of Westminster, Coat of Arms

Stochronetics

Surpassing Time Series Forecasting Limitations using Liquid Time-stochasticity Networks

A dissertation by

Mr. Ammar Raneez

W1761196 | 2019163

Supervised by

Mr. Torin Wirasingha

April 2023

This Dissertation is submitted in partial fulfilment of the requirements for the BSc (Hons) Computer Science degree at the University of Westminster.

ABSTRACT

Despite the significant progress made in Deep Learning (DL) in areas such as Natural Language Processing (NLP) and Reinforcement Learning (RL), Time Series (TS) forecasting has yet to witness a comparable breakthrough due to inherent algorithmic limitations. Consequently, systems developed using these methods are constrained in forecasting performance.

Researchers have proposed various approaches to address this challenge, with neural Ordinary Differential Equations (ODEs) and Liquid Time-constant (LTC) networks among the most promising. While neural ODEs introduce the concept of continuous-time and depth models, their performance has been underwhelming compared to traditional neural networks like Long Short-Term Memory (LSTM). On the other hand, LTC networks have shown impressive results, but their outdated architecture of ODEs limits their usability and performance. In this project, the author proposes a novel algorithm that combines the adaptability of LTC networks with Stochastic Differential Equations (SDEs). The architecture achieved state-of-the-art performance in TS forecasting, as demonstrated by the highly popular Bitcoin (BTC) price prediction problem.

The proposed algorithm, called **Liquid Time-stochasticity** (LTS), combines the adaptability of the LTC network with SDEs. By introducing SDEs, the LTS can handle small noises and immediate changes, resulting in a more stable and robust implementation of continuous-time and depth models. The algorithm uses traditional Backpropagation Through Time (BPTT) to produce accurate results, although it comes at the cost of higher memory usage. Applying LTS on a BTC price forecasting problem yielded a promising result, with a percentage error of **2.5% ± 0.1**. Future research could explore using adjoint sensitivities to push the algorithm's efficiency and scalability.

Keywords: Time Series (TS) forecasting, Liquid Time-constant (LTC) networks, Liquid Time-stochasticity (LTS) networks, Stochastic Differential Equations (SDEs).

Subject Descriptors:

- Theory of computation → Design and analysis of algorithms → Approximation algorithms analysis → Stochastic approximation.
- Mathematics of computing → Probability and statistics → Stochastic processes.
- Computing methodologies → Machine learning → Machine learning algorithms → Neural networks.

PUBLICATIONS

1. **A Review on Breaking the Limits of Time Series Forecasting Algorithms**

   Conference: IEEE CCWC 2023

   Description: Literature review & Liquid Time-stochasticity proposal

   Status: Accepted & Presented

   Venue: Las Vegas, NV, USA

   Date: 8-11 March 2023

   Publication: **https://doi.org/10.1109/CCWC57344.2023.10099071**

DECLARATION

I hereby declare that this dissertation and its associated artefacts are results of my own research efforts, and that none of them have been submitted / currently being submitted to another degree program or related qualification of another affiliation. Any information extracted from other sources are clearly accredited.

Student name – Ammar Raneez

Registration Number – w1761196 | 2019163

Signature: 

Date: April 27, 2023

ACKNOWLEGEMENT

‘*People come and go from your life, but only the most real of them stick with you through thick and thin.’* This is a quote I will always live up to, no matter where I go in the future. It was only made evident recently when certain events led to others, leading to more events. To whoever stayed by my side while I was facing such adversity, I thank you sincerely, as I otherwise would not have been able to bring this research to fruition.

Apart from them, I would like to extend my thanks to my supervisor, Mr. Torin Wirasingha, for his continuous support and guidance whenever necessary, as I would not have been able to be the first person in my batch – up to my knowledge - to have published a research paper. Furthermore, I sincerely appreciate and am eternally grateful to our module leader, Mr. Guhanathan Poravi, for his continuous and relentless advice and valuable insights since the second year's commencement.

Special recognition and appreciation are extended to all the interviewees and evaluators – both within the country and offshore - and survey respondents for their cooperation and valuable feedback. The extent to which I could push the research was only possible by such domain expertise. I would also like to take a moment to express my appreciation to the other lecturers at the university for their persevering hard work in shaping me into the person I am today. Needless to say, my family plays a vital role in every aspect of my life, and words would never be enough to express my gratefulness.

Lastly, I would like to profess my gratitude to a peculiar person who has constantly challenged me to push beyond my limits – never being satisfied with what I do. Even when I doubted myself, their unshakeable faith in me has had a profound impact that I will always treasure.

Thank you all from the bottom of my heart,

Ammar

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence.
API	Application Programming Interface.
AD	Automatic Differentiation.
ARIMA	Autoregressive Integrated Moving Average.
BPTT	Back-Propagation Through Time.
BTC	Bitcoin.
CT-GRU/RNN	Continuous-time Gated Recurrent Unit / Recurrent Neural Network.
DL	Deep Learning.
GPU	Graphics Processing Unit.
LSTM	Long Short-Term Memory.
LTC	Liquid Time-constant.
ML	Machine Learning.
(s)MAPE	Symmetric Mean Absolute Product Error.
MASE	Mean Absolute Scaled Error.
MSE	Mean Squared Error.
MVP	Minimal Viable Product.
N-BEATS	Neural Basis Expansion Analysis for interpretable Time Series.
NER	Named Entity Recognition.
NLP	Natural Language Processing.
POC	Proof-Of-Concept.
REST	Representational State Transfer.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
SOTA	State Of the Art.
SDE; **ODE**	Stochastic Differential Equations; Ordinary Differential Equations.
TS	Time Series.
UI	User Interface.
VADER	Valence Aware Dictionary for Sentiment Reasoning.
XAI	Explainable Artificial Intelligence.

CHAPTER 01. INTRODUCTION

1.1 Chapter overview

Every research begins with an introduction and a broad overview of the subject that it will cover in the following chapters. This chapter aims to provide readers with an overview of current issues in time series forecasting and introduce the concept of a liquid time-stochasticity network and its purpose. Specifically, the author will define the problem, review the literature to identify a research gap and formulate research questions, objectives, and challenges that this study will address that would arise upon embarking on this research journey.

1.2 Problem domain

1.2.1 Time series forecasting

TS forecasting is a critical business concern, and an area ML could significantly influence. It forms the basis of crucial domains such as customer management, inventory control, marketing, and finance. Correspondingly, even slight enhancements in forecasting accuracy can have a substantial financial impact, potentially worth millions of dollars (Jain, 2017; Raneez and Wirasingha, 2023).

Despite the successes of ML and DL in NLP and computer vision, TS forecasting remains a challenge compared to classical statistical methodologies (Makridakis et al., 2018a;b). In fact, in the M4 competition, ensembles of traditional statistical techniques dominated the top-ranking methods. In contrast, traditional ML methods were not competitive (Makridakis et al., 2018b): the winner was a hybrid model of LSTM and classical statistics (Smyl, 2020), who concluded that the only way to improve TS forecasting accuracy was through hybrid models. However, in this research project, the author seeks to challenge this conclusion.

1.2.2 Liquid Time-constant (LTC) networks

LTCs are neural ODEs: hidden layers are not specified; instead, the derivative of hidden states is parameterized by neural networks (Chen et al., 2019). RNNs are successful algorithms for TS data modelling if ODEs determine continuous time-hidden states (Chen et al., 2019). Studies show that existing algorithms such as the CT-RNN (Funahashi and Nakamura, 1993; Rubanova, Chen and Duvenaud, 2019) and CT-GRU (Mozer, Kazakov and Lindsey, 2017) can achieve good

performance. However, they have issues with expressivity and fixed behaviour once trained (Hasani et al., 2020). Therefore, an important question arises: What would happen if unexpected changes to the input characteristics occurred during inference? Additionally, these algorithms show reduced generalization compared to even a simple LSTM network (Hasani et al., 2021), which raises another question, what is the benefit of introducing a seemingly more complex approach if it cannot perform well in real-world applications?

Hasani et al. state that LTCs can *'identify specialized dynamical systems for input features arriving at each time point'* (2020, p1). The ability to exhibit stable and bounded behaviour demonstrates that the proposed approach provides better expressivity than traditional implementations. The LTC state and their respective time constant '*exhibit bounded dynamics and ensure the stability of the output dynamics'*, which is a prominent factor when inputs increase relentlessly (Hasani et al., 2020, p2).

However, it is important to note that the underlying concepts used within the LTC architecture are considered outdated (Duvenaud, 2021) and lack the adaptability needed to handle noisy data and data with high volatility. As a result, the LTS algorithm presented by the author represents an enhancement to the LTC that addresses these limitations.

Due to the highly volatile nature of BTC, it is an ideal candidate for evaluating the proposed algorithm. Cryptocurrencies, including BTC, have demonstrated high levels of volatility in their price movements, making them a challenging domain for TS forecasting. Therefore, BTC serves as an ideal problem domain for rigorously evaluating the effectiveness of the algorithm.

1.2.3 Cryptocurrencies

Cryptocurrency, especially BTC, has been a widely discussed topic recently - it has even come to the point where crypto and BTC are used interchangeably. Cryptocurrencies are a fully decentralized digital currency form that operates using a peer-to-peer system without needing a third party, thus enabling safer online transactions (S. Nakamoto, 2008). BTC is the first and most popular digital currency, piquing many academic researchers' interest (Rahouti et al., 2018).

Recent advances in ML and statistics have shown acceptable results in analyzing and predicting cryptocurrencies. However, these algorithms are still static, and BTC's high volatility challenges investors (Kervanci and Akay, 2020).

1.3 Problem definition

TS forecasting is considered lagging compared to other domains where ML is applied, as highlighted by Makridakis et al. (2018a;b), because existing TS forecasting algorithms cannot adapt to unforeseen data stream changes, as Hasani et al. (2020) noted. As a result, they tend to perform poorly when applied to TS forecasting.

It is worth noting that while the LTC architecture proposed by Hasani et al. (2020) addresses this limitation to some extent, their proposed solution that utilizes ODEs cannot effectively model instantaneous changes or highly volatile environments (Duvenaud, 2021).

1.3.1 Problem statement

TS forecasting algorithms cannot adapt to incoming data streams with unexpected changes and volatile characteristics that differ from the data they were trained on; consequently, they exhibit poor performance, particularly in volatile data modelling.

1.4 Research motivation

The field of AI, particularly neural networks, has experienced significant growth recently, with many interesting research developments. However, as Hasani et al. (2020) have noted, the static network limitation and inability to adapt to varying characteristics pose a challenge for the future of intelligent systems, especially TS. Therefore, this research is expected to contribute to the advancement of the domain by providing insights into overcoming this challenge.

1.5 Research gap

1.5.1 Gap in existing forecasting algorithms

Presently, all available forecasting solutions employ conventional deep neural network techniques, such as LSTM (Hochreiter and Schmidhuber, 1997), which are static and cannot learn and adapt during inference (Hasani et al., 2020), leading to a gradual deterioration in the model's accuracy due to data drift (Poulopoulos, 2021).

1.5.2 Gap in the liquid time-constant network

The proposed LTC architecture employs a series of linear ODEs, currently considered outdated and lack rapid adaptability (Duvenaud, 2021). Recent developments in this field suggest using

SDEs instead, as they provide more flexibility than ODEs. Another issue with ODEs is that they model ‘deterministic dynamics’, meaning that uncertainty and unobserved interactions cannot be modelled, which is unavoidable in TS data (Duvenaud, 2021). Therefore, the way forward is to construct the LTC architecture using SDEs (Raneez and Wirasingha, 2023). As suggested, the algorithm proposed is the author’s own, named ‘Liquid Time-stochasticity’, as it is no longer constant.

1.5.3 Gap in bitcoin forecasting solutions

The literature on BTC forecasting has yet to incorporate exogenous factors that may influence its price (Roy et al., 2018; Rizwan et al., 2019; Fleisher et al., 2022). Consequently, their inability to adapt to such factors is a significant concern, rendering them less robust. For instance, tweet sentiment, tweet volume, and Google Trends have all been shown to correlate with the price of BTC (Abraham et al., 2018). Therefore, future systems forecasting BTC must consider such factors to enhance their predictive capabilities.

Forecasting cryptocurrency prices can be challenging due to the system's open and decentralized nature, making it vulnerable to external factors beyond the control of researchers. However, some factors have been found to correlate with cryptocurrency prices, such as tweet sentiment, tweet volume, and Google Trends, as identified by Abraham et al. (2018). These factors can be taken into account by researchers when developing forecasting models.

Self-reflection on the research gap

The literature review reveals that only one paper is available on the LTC architecture (Hasani et al., 2020), while other related works focus on the family of neural ODEs (CT-RNN (Rubanova, Chen and Duvenaud, 2019) and CT-GRU (Mozer, Kazakov and Lindsey, 2017)) and the secondary problem domain of cryptocurrencies and TS forecasting. Since the LTS is an improvement over the LTC, the lack of available literature on this architecture poses a significant challenge.

1.6 Contribution to the body of knowledge

In a nutshell, the author desires to answer the following question:

- Can the LTS algorithm overcome the limitations of existing TS forecasting methods?

1.6.1 Research domain contribution

The LTS algorithm will be comprehensively designed, developed, and thoroughly documented to facilitate future research. Furthermore, the algorithm will be generalizable, without being specific to any particular problem, so researchers can assess its performance in other domains and determine whether it could significantly advance those fields.

The proposed solution to the current limitations in TS forecasting is a dynamic algorithmic architecture that can adapt and change its underlying mathematical expressions and evaluation strategies in response to changes in incoming data streams. Furthermore, it is necessary to incorporate enhancements to deal with sudden and minute changes that often occur in highly volatile and non-linear TS data. An algorithm capable of adapting to unforeseen changes and is highly expressive can significantly contribute to the TS forecasting community and serve as a foundation for future research.

1.6.2 Problem domain contribution

Based on the above critique, it is evident that the current solutions have limitations. However, creating a more robust forecasting solution that considers the mentioned factors makes it possible to predict the highly volatile market of cryptocurrencies more efficiently.

The author will also provide dedicated scripts to obtain the required data since no direct data source is available for building the model. These scripts will be publicly available to support future research on BTC forecasting.

1.7 Research challenges

1.7.1 Research domain challenges

RC1: There is a prevalent belief in deep neural networks that deeper architectures are usually better; however, the LTC challenges this notation by emphasizing the importance of scaling down (Hasani et al., 2020). Therefore, a challenge arises in identifying the requirement of scaling down and what an LTC and neural ODE aim to solve.

RC2: The LTC represents a novel approach that has thus far only been explored in a single research paper. As it is still in the experimental phase, its proposed solution relies on a distinct formulation that requires further investigation and validation.

RC3: The broader domain of neural ODEs (Chen et al., 2019) is still in its early stages, which can create challenges for future research and the implementation of systems due to the scarcity of references.

RC4: SDEs are an advanced topic in mathematics, and while there have been studies on modelling them as neural SDEs, they were primarily focused on specific applications. As a result, no generic papers are available for neural SDEs, unlike neural ODEs, which makes modelling difficult.

RC5: Currently, existing TS forecasting systems are built using statistical ensemble methods (Makridakis et al., 2018b) or traditional neural net architectures (Hasani et al., 2021), which creates a new challenge where there is no reference implementation.

1.7.2 Problem domain challenges

RC6: The domain of application chosen for this research is an open system. Open system forecasting typically yields poor results, and beating the naïve forecast can be challenging (Gilliland, 2014) since it can depend on any external factor. Hence, there is a risk of discouragement from pursuing the research if the obtained results do not meet expectations.

1.8 Research questions

RQ1: What are the recent advancements in TS algorithms that can be considered when building the algorithm?

RQ2: How can the algorithm implement a TS forecasting system, and how will the challenges faced be overcome?

RQ3: What contributions can be made to the TS forecasting community?

RQ4: What external factors impact BTC prices and must be considered in model construction?

1.9 Research aim

The aim of this research is to design, develop & evaluate the author-proposed LTS algorithm for TS forecasting, which could be the stepping stone for breaking TS forecasting limitations.

Specifically, this research project will produce a TS forecasting system utilizing the LTS algorithm to forecast BTC.

1.10 Research objectives

Research objectives are milestones that the author must achieve for the research to succeed.

Table 1: Research objectives

Objective	Description	Learning Outcomes	Research Questions
Problem Identification	Document and comprehend the identified problem and its novelty. **RO1**: Research on a domain the author is interested in and identify a comprehensive enough issue that requires solving. **RO2**: Delve deeper into the identified problem to obtain a general understanding of how to solve the problem. **RO3**: Split the primary problem into manageable subproblems so it is easier to digest and solve one section at a time. **RO4**: Design a schedule, associated deliverables, and the Gantt chart to follow in this research.	LO1, LO2	**RQ1**
Literature Review	Collate relevant information by reading, understanding, and evaluating previous work. **RO5**: Conduct preliminary studies and investigations on existing TS forecasting systems and algorithms. **RO6**: Analyze the requirement for specialized TS algorithms. **RO7**: Research on neural ODEs, LTCs & SDEs. **RO8**: Obtain deep insights into the architecture behind the LTC. **RO9**: Research and obtain insights on factors affecting the price of BTC.	LO1, LO4, LO5	**RQ1**, **RQ4**

	RO10: Research existing BTC forecasting and related open market systems. **RO11**: Research on necessary ML techniques and evaluation approaches.		
Requirement Elicitation	Collect and analyze project requirements using appropriate tools and techniques. **RO12**: Analyze stakeholders and understand their viewpoints and concerns. **RO13**: Gather the requirements and architectures of neural ODEs, LTCs, and SDEs. **RO14**: Collate the most up-to-date details of BTC and obtain insights into the end users' perspectives. **RO15**: Design the use case and context diagrams to justify the product's specifications.	LO1, LO3	**RQ1**, **RQ4**
Design	Design the architecture and system to solve the identified problems effectively. **RO16**: Design diagrams to grasp the algorithm. **RO17**: Design the LTS formula and analyze its complexities. **RO18**: Choose appropriate techniques for forward and backward propagation of the LTS. **RO19**: Design diagrams to understand the supplementary system being developed. **RO20**: Design a deployment pipeline for seamless deployments.	LO1	**RQ2**, **RQ3**
Implementation	Implement a system that effectively addresses the research gaps. **RO21**: Design, evaluate and select the technologies best suited for the implementation. **RO22**: Develop an efficient LTC implementation. **RO23**: Build on the LTC to implement the LTS.	LO1, LO5, LO6, LO7	**RQ2**, **RQ3**, **RQ4**

	RO24: Integrate the developed algorithm into a TS forecasting application. **RO25**: Incorporate the intelligent system into a client application to display forecasts. **RO26**: Design and implement an automated flow to update the built network with the latest data. **RO27**: Consider any legal, social, ethical, and professional issues upon implementation.		
Evaluation	Effectively test the implemented algorithm, system, and data science model using recommended techniques. **RO28**: Evaluate the developed algorithm and the respective model against the predefined evaluation metrics. **RO29**: Create a test plan and cases and perform functional, non-functional, and integration testing. **RO30**: Solicit feedback from domain and tech experts to understand limitations and future work.	LO4	**RQ2**, **RQ3**
Documentation	Document the progress of the research project and report any encountered challenges. **RO31**: Produce a comprehensive report of new skills acquired and contributions made, ensuring the aforementioned objectives are achieved. **RO32**: Publish a research paper outlining the LTS architecture to provide validation.	LO6, LO8	**RQ3**

1.11 Chapter summary

This chapter provided an overview of the research project, including its significance and the challenges in solving the identified problem. The author also presented the key objectives that need to be achieved for the research to succeed and how these objectives align with the required learning outcomes for the degree.

CHAPTER 02. LITERATURE REVIEW

2.1 Chapter overview

Research is required to prevent reinvention of the wheel. This chapter presents a detailed critique of related work within the domain of TS and BTC forecasting and in the broader realm of open market forecasting to justify the author's chosen research domain wholly. Moreover, the author will provide a background on the chosen field and bring forward research possibilities upon project completion.

2.2 Concept map

To ensure a comprehensive literature review, the author created a concept map that visually represents the key concepts and their relationships within the research domain. The full concept map can be found in **APPENDIX I**.

2.3 Problem domain

2.3.1 Time series forecasting

TS forecasting has attracted numerous researchers to develop robust, scalable, and practical solutions. It plays a crucial role in various real-life problems ranging from forecasting the weather to predicting traffic to predicting EEGs of patients in medical monitoring (Lara-Benítez, Carranza-García and Riquelme, 2021). Essentially, any problem with a temporal component can potentially benefit from TS forecasting, making it highly in demand and impactful in the business world.

With subject-matter expertise, conventional statistical models have been applied for TS forecasting. ML has grown increasingly important in creating forecasting models for the upcoming generation due to the availability of data and computing capacity (Lim and Zohren, 2020). ML offers a technique to learn temporal dynamics in a data-driven manner, in contrast to their conventional counterparts (Ahmed et al., 2010). Particularly, DL has recently experienced considerable popularity growth due to its outstanding achievements in image classification, NLP, and reinforcement learning. DL is a compelling method for TS forecasting because it can learn representations from complex data without explicit engineering (Lim and Zohren, 2020).

2.3.2 Cryptocurrencies (Bitcoin)

Blockchain technology is based on peer-to-peer connectivity and cryptographic security that removes the need for a third-party centralized system, thereby demonstrating transparency and trust compared to traditional monetary systems (Rejeb, Rejeb and G. Keogh, 2021).

After the global financial crisis of 2008, BTC, a peer-to-peer electronic cryptocurrency, was introduced due to the loss of public trust in banking systems (Nakamoto, 2008). Cryptocurrencies were motivated by the need to create a system that allowed simple, fast, and cheap transactions not influenced by a third party (ex: banks) (Kfir, 2020). Moreover, multiple scholars and enthusiasts considered BTC the future of currency (Bouri et al., 2018). Over time, over 1,600 cryptocurrencies (Wilson, 2019) have been introduced to exchange goods and services; this research focuses on BTC.

2.3.2.1 Opportunities of cryptocurrencies

Although cryptocurrencies do not solve all financial problems, they address many issues, such as the lack of trust, instability, and inefficiency in transactions (Nakamoto, 2008). Furthermore, Buhalis et al. (2019) suggest that transactions with cryptocurrencies are much more systematic and straightforward, as they can potentially prevent fraudulent exchanges and payments.

One of the most critical use cases for cryptocurrencies is for online payments. With the rise of cashless payments, including credit cards, cryptocurrencies have emerged as a popular alternative for online transactions (Wang et al., 2019). According to Pournader et al. (2020), companies can benefit from the ability to make instant, commission-free money transfers using cryptocurrencies.

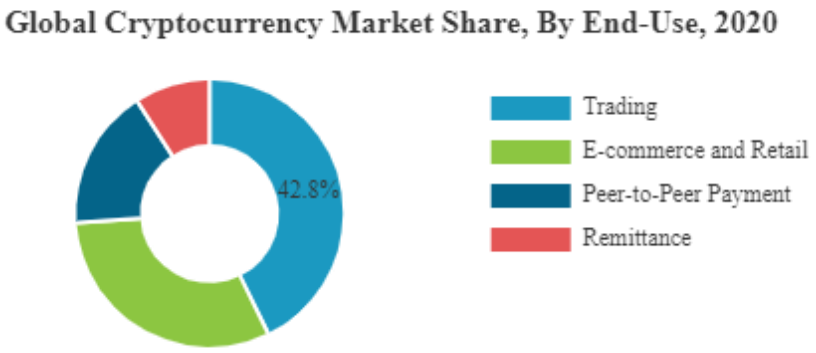


Figure 1: Global market share (Fortune Business Insights, 2021)

BTC's popularity can be attributed to its low transaction fees compared to traditional payment services (Nica et al., 2017). However, this can be sustained only if trading occurs under '*different assumptions that may not hold in specific situations, including quick links between users, low transaction costs, and high liquidity*' (Alonso-Monsalve et al., 2020, p10). According to Nakamoto (2008), BTC's settlement time is much faster than non-cash transactions, which may take days or weeks.

The growing popularity of cryptocurrencies has raised the possibility that they may eventually replace traditional currencies. However, this trend also presents challenges that must be addressed, such as regulatory concerns, price volatility, and technological barriers.

2.3.2.2 Challenges of cryptocurrencies

The rapid growth of cryptocurrencies has introduced certain risks due to the absence of legislation and regulatory standards, which raises concerns about their integration into the financial system (Avdeychik & Capozzi, 2018).

Although cryptocurrencies' decentralized nature has advantages, the lack of a governing authority has also resulted in the development of online black markets. Due to the anonymity provided by cryptocurrencies, it is even more challenging to trace the identity of a specific operation, user, or criminal activity, leading to concerns about using cryptocurrencies illegally (Baldimtsi et al., 2017). According to Kerr (2018), BTC has been used as a tool for conducting business in the black market. Cryptocurrencies have been used to facilitate the sale of drugs, weapons, and other illegal goods, including child pornography (Miller, 2016). This growth in illegal cryptocurrency-related activities could threaten individuals' safety, income, and lives (Scharding, 2019).

2.3.2.3 Why have cryptocurrencies taken the world by storm?

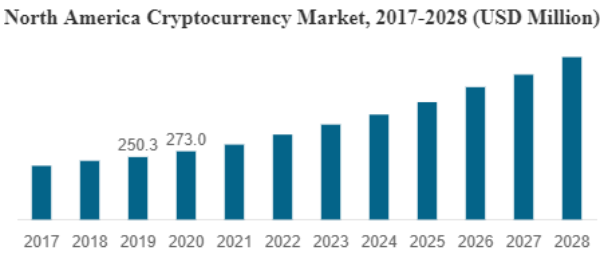


Figure 2: Latest trends (Fortune Business Insights, 2021)

The graph above illustrates the steady growth of the cryptocurrency market over the years, which can be attributed to the increasing popularity of digital currencies. This growing popularity has also gained the trust of central banks, leading them to establish a patent for digital currency projects called the Central Bank Digital Currency (CBDC), resulting in multiple countries, such as China, Thailand, Uruguay, and those in the Caribbean, supporting the CBDC and considering digital currency as a viable means of exchange. As a result, there is potential for an explosion in the popularity of cryptocurrencies in the future (Fortune Business Insights, 2021).

The graph could experience exponential growth with the increased involvement of more countries.

2.3.2.4 Cryptocurrency exchanges

A cryptocurrency exchange is a platform that facilitates the trading, buying, and selling of cryptocurrencies. They also allow the conversion of cryptocurrencies to fiat currency and to withdraw it to a local bank account (Boom, 2021).

Some of the more popular platforms include *Gemini*, *Coinbase* & *Binance*.

2.4 Existing work

2.4.1 Time series forecasting

Multiple methods have been introduced for TS forecasting, ranging from traditional statistics to deep learning models. Since the domain of TS forecasting is more technologically specific, insights on the various approaches are broken down under the **technological review** section of this chapter.

2.4.2 Bitcoin forecasting

BTC forecasting has experienced exponential growth in the past decade, with many websites making historical data publicly available to satisfy stakeholders' need for forecasting models. As a result, research has focused on developing forecasting models to predict BTC prices accurately.

Fleischer et al. (2022) proposed using the LSTM to learn patterns in BTC's historical closing prices to predict future prices. While the approach was shown to be effective in producing accurate one-day predictions, its simplicity raises concerns about its long-term sustainability as a forecasting model. This is because the BTC market is an open and complex system influenced by many factors beyond the closing price. Therefore, while the study's results are promising, more sophisticated forecasting models are needed to capture the multifaceted nature of the BTC market.

Yenidogan et al. (2018) suggested using the Prophet model by Facebook to distinguish it from ARIMA. They found that other features in the dataset (such as open price and volume) were not meaningful for predictions. After removing these features, the authors added a fiat currency price feature, which strongly correlates with BTC prices, resulting in more accurate forecasts.

2.4.2.1 Factors that affect the rate of Bitcoin (BTC)

- A study conducted by Sarkodie, Ahmed and Owusu (2022) found a strong correlation between the number of deaths from COVID-19 and BTC price.
- Tweet sentiment - identified by Kim et al. (2016).
- Tweet volume - identified by Abraham et al. (2018) & Shen, Urquhart and Wang (2019).
- Google Trends – identified by Abraham et al. (2018).

2.4.2.2 Reflection on bitcoin forecasting systems

The above-proposed solutions have limitations in that they do not consider the impact of external factors on the price of BTC. This can be a concern as the cryptocurrency market is highly volatile, and even a single tweet can cause significant price fluctuations. While external factors such as country legislation and laws may be beyond researchers' control, it is crucial to consider those that can be accounted for, including sentiment analysis of tweets, tweet volume, and Google Trends data. However, it is important to note that these factors alone may not always provide accurate predictions as they are subject to change rapidly. Therefore, to build robust forecasting systems, it is essential to consider and account for all relevant external factors that could impact the price.

Abraham et al. (2018) suggested incorporating Twitter and Google Trends data into forecasting models. They discovered that tweet volume and Google Trends were highly correlated with the BTC price, making them important factors to consider. However, they noted that tweet sentiment might not be reliable because tweets about cryptocurrencies tend to be objective. The authors employed a linear model for their predictions, but other studies have demonstrated that non-linear models can yield better performance (Maiti, Vyklyuk and Vukovic, 2020).

2.4.3 Open market forecasting

As cryptocurrencies operate in an open market, it is natural to expect that research in other open market forecasting domains, such as the stock market, could be adapted to cryptocurrency forecasting. However, current research lacks promising results (Li and Bastos, 2020). On the other

hand, there has been relatively more success in utilizing NLP techniques and analyzing large volumes of textual data to forecast cryptocurrency prices.

Picasso et al. (2019) developed a stock market forecasting solution that combines news data and historical prices. By applying sentiment analysis to the news data and integrating it with technical and fundamental analysis, they aimed to create a more robust model. While the results were not outstanding, the approach showed promise and can be adapted to other forecasting domains, such as cryptocurrency.

Kim et al. (2019) presented a hybrid forecasting model for power demand that can be applied to various forecasting problems. The hybrid model incorporated a bivariate learning approach to enhance its ensemble potential. The authors also considered external factors that could influence power demand to improve the model's performance. However, the model is mainly intended for short-term power demand forecasting, limiting its application to medium to long-term forecasting.

2.4.3.1 Reflection on open market forecasting

Forecasting in open systems can be challenging due to the unpredictable nature of external factors. For example, predicting the stock market is notoriously difficult and is often said to be as reliable as palm reading. In such domains, the application of DL models may not be suitable, despite their ability to achieve high performance and reliability that exceeds human capabilities. The reason for this is the uncertainty of future events, which can significantly hinder the effectiveness of such models. Additionally, it is important to note that the relationship between news and the stock market becomes more ambiguous when forecasting long-term trends.

2.4.4 Bitcoin Twitter analysis

As identified above, adding external factors can improve performance drastically, where some of the more impactful factors are Twitter sentiments and tweet volumes. However, creating such models is more complex than including them as another feature in the dataset. Several issues still need solving when using tweet data as a factor in crypto price predictions (Critien, Gatt and Ellul, 2022). Some of them are detailed below:

- Bots often duplicate and automate tweets (Valencia et al., 2019).
- They are often noisy (ex: hashtags, profile mentions, URLs).

- Sarcasm can affect sentiments (Rosenthal et al., 2014).

Therefore, the data must first be preprocessed with such challenges in mind.

A point worth noting is that Valencia et al. (2019) identified that more than Twitter data is needed for the prediction of BTC, but it can be helpful when used in conjunction with other data.

Pant et al. (2018) developed a model for BTC price prediction by incorporating Twitter sentiment analysis. They found a moderate correlation between Twitter sentiment and BTC price and integrated the sentiment scores of tweets with historical data to forecast future prices. However, as Abraham et al. (2018) pointed out, tweet volume is more influential than tweet sentiment in determining the price of BTC, which Pant et al. (2018) did not account for.

Serafini et al. (2020) explored statistical and DL models for the predictions utilizing sentiment in the BTC ecosystem. They identified that rather than financial features, the sentiment has a more significant impact on the overall price, which further justifies that the usage of the volume and open features do not have such significance. They justified that the tweet sentiment and BTC price produced the best performance. The issue here is: although the performance obtained is excellent, the implementation is not robust since, as mentioned by Valencia et al. (2019), the Twitter data alone is insufficient, and, as identified by Abraham et al. (2018), tweets can be objective.

2.4.4.1 Reflection on bitcoin Twitter analysis

The drawbacks of the above two solutions are that tweets are objective, can carry sarcasm, and can be duplicated or generated by bots. Therefore, utilizing the tweet sentiment solely is not robust. Hence, the optimal solution would be to utilize additional features as Abraham et al. (2018) proposed alongside a non-linear model, as identified by Maiti, Vyklyuk and Vukovic (2020).

2.5 Technological review

The sheer number of available algorithmic solutions shows that this ML area still requires a satisfactory settlement. Methodologies range from classical statistical approaches to complex DL algorithms; although particular methods have proven effective, many attempts have been made to improve them. A couple of the more popular and modern techniques among the many available are evaluated below.

2.5.1 Statistical-based forecasting techniques

Statistical forecasting uses statistics to make predictions based on historical data.

2.5.1.1 Autoregressive Integrated Moving Average (ARIMA)

The ARIMA model combines the Moving Average (MA) and Autoregressive (AR) models. Based on historical data, these models forecast future values (Box et al., 2015). Due to its remarkably good performance, this particular model is the most widely used in TS forecasting; however, certain crucial factors that cause these models to make errors must be taken into account.

- It is best suited for stationary time series data. It can be less effective when applied to non-stationary or open systems that are subject to external factors.
- It may not capture complex patterns or non-linear relationships in data.
- Not well-suited for modelling seasonal data.

2.5.1.2 Seasonal Autoregressive Integrated Moving Average (SARIMA)

SARIMA was introduced as an extension to ARIMA to address the issue of seasonality in data. It uses the same autoregressive and moving average components as ARIMA but adds a seasonal component to the model. SARIMA's parameters are more complex than those of ARIMA, and even small adjustments can have a negative impact on performance. However, Valipour's (2015) research has shown that SARIMA outperforms ARIMA in terms of forecasting accuracy.

2.5.1.3 Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

GARCH is a widely used method for evaluating market volatility when predicting prices or rates, which considers the contextual information of the data (Engle, 1982). In contrast to ARIMA, which only considers historical values, GARCH models incorporate the variance of past residuals, which allows them to account for volatility and better capture changes in the data. Bhardwaj et al. (2014) showed that GARCH forecasts were more accurate than ARIMA's when predicting market volatility, especially when the data is highly volatile.

2.5.1.4 Prophet

Facebook presented a method that relies on changing parameters to address the difficulty of forecasting at scale (Taylor and Letham, 2017). It was created to predict daily data that included seasonality on a weekly and yearly basis while considering the impact of vacations (Hyndman et al., 2021). As a result, it functions best with highly seasonal data.

2.5.1.5 Is there a clear better statistical forecasting algorithm?

Research shows that no model outperforms all others regarding forecasting accuracy when considering existing statistical algorithms. The differences in errors and accuracy among models are typically small, and the best-performing model will depend on the specific domain, issue, and dataset being analyzed (Raneez and Wirasingha, 2023).

2.5.2 Deep learning-based forecasting techniques

Studies have shown that certain time series forecasting domains may perform better with statistical models (ex: Zhang et al., 2022), while others may benefit from DL models (ex: Siami-Namini, Tavakoli and Siami Namin, 2018). Therefore, it is important to evaluate both models to determine which is most suitable for a given domain and dataset.

2.5.2.1 Long Short-term Memory (LSTM)

LSTMs were developed by Hochreiter and Schmidhuber (1997) to overcome the limitations of earlier RNNs, which had difficulty retaining information over longer periods. By allowing for the modeling of temporal dependencies over longer horizons without neglecting short-term trends, LSTMs paved the way for more effective RNNs in the future (Lara-Benítez, Carranza-García and Riquelme, 2021). Studies have shown that the ability to extract important information from TS data is the most significant advantage of LSTMs. A prominent study by Sagheer and Kotb (2019) found that the LSTM outperformed ARIMA and GRU in terms of forecasting accuracy.

2.5.2.2 Gated Recurrent Unit (GRU)

The GRU is a variant of the LSTM architecture introduced by Cho et al. (2014). GRUs have fewer parameters than LSTMs, requiring less computation while providing competitive results in various applications (Lara-Benítez, Carranza-García and Riquelme, 2021). While the relative performance of GRUs and LSTMs varies depending on the dataset and the specific task, both architectures are widely used and often compared in TS forecasting research. According to a study by Kuan et al. (2017), GRU performed better than ARIMA and LSTM.

GRUs and LSTMs are effectively similar in performance

It is clear that there is no superior method between GRU and LSTM because Kuan et al. (2017) and Sagheer and Kotb (2019) produced contradictory results. However, these non-linear models outperform statistical models, such as ARIMA, giving the study by Maiti, Vyklyuk and Vukovic (2020) greater validity.

2.5.2.3 Neural Basis Expansion Analysis for interpretable Time Series (N-BEATS)

The architecture of N-BEATS, a relatively new SOTA forecasting algorithm, is focused on tackling univariate point forecasting. It has several qualities, the most important of which is that it can be interpreted and applied across various domains (Oreshkin et al., 2020).

N-BEATS is a pure DL architecture that is straightforward, generic, and expressive. In the M4 competition, it outperformed the 2019 winner - a hybrid of statistical and neural network methods - setting its mark as the 2020 SOTA algorithm. This result challenges the effectiveness of hybrid models in TS forecasting, as proposed by Makridakis et al. (2018b).

2.5.2.4 Temporal Fusion Transformer (TFT)

Google has developed a new attention-based architecture called TFT, which aims to address the problem of multi-horizon forecasting while maintaining a high level of interpretability. TFT is optimized for performance and allows for feature selection and removal, improving it over previous black box solutions (Lim et al., 2019). Experimental results have shown that TFT outperforms previous benchmarks.

The proposed architecture was inspired by existing transformer-based architectures, such as Li et al. (2019). However, unlike these architectures, the TFT was specifically designed to handle different types of inputs in multi-horizon forecasting and to account for scenarios where the required exogenous features are not always available.

2.5.2.5 Are DL models superior to statistical models?

While the idea that statistical models are always better is sometimes justified, the literature mentioned above suggests that this is not always the case. Furthermore, the superiority of non-linear DL models over statistical models cannot be universally inferred. Therefore, when approaching TS problems, it is worth considering both approaches and selecting the model that best performs for the problem at hand (Raneez and Wirasingha, 2023).

2.5.3 Concerns about existing used techniques

2.5.3.1 Issues in statistical models

Based on the literature discussed earlier, the following limitations of statistical-based models can be identified:

- Linearity - statistical models are generally linear, which means they may not be able to capture complex patterns in the data.
- Seasonal data - these models are most suited for data that exhibit seasonal patterns.
- Lack of expressivity and interpretability - statistical models are often seen as a black box.
- Domain expertise - knowledge of the domain is often necessary to tune the parameters.

2.5.3.2 Issues in deep learning models

Despite the remarkable success of DL models in many fields, TS forecasting remains a challenge, as noted by Makridakis et al. (2018b). While the non-linearity of neural networks can improve their performance, it can also make them less interpretable and less expressive compared to statistical models. Therefore, the choice of model depends on the specific requirements of the problem at hand, and a balance must be struck between accuracy and interpretability.

Given the result of the comparison between statistical and DL models that was deliberately left open-ended, the logical question is: **how to proceed?** A limitation of the models above is that they are static and unable to adapt to the dynamic nature of TS data. TS data is often unpredictable, volatile, and ever-changing, which may result in unanticipated changes in its characteristics. This can pose difficulties for fixed statistical models or neural networks, which may have contributed to Makridakis et al.'s (2018b) findings. Therefore, there is a need to explore and develop adaptive models that can dynamically adjust to changes in TS data.

2.5.4 Neural Ordinary Differential Equations (ODEs)

Neural ODEs are a novel DL model class with continuous-depth and constant memory costs. These models are known for their speed but may sacrifice numerical accuracy. One of their distinguishing features is their ability to adapt their evaluation methods based on input changes (Chen et al., 2019).

The following modification to the conventional equation used by RNNs and residual networks was suggested by Chen et al. (2019) in their study of the capacity of RNNs with continuous-time hidden states specified by ODEs to model TS data efficiently.

Equation followed by RNNs & residual networks.

$$h_{t+1} = h_t + f(h_t, \theta_t)$$

The equation proposed by Chen et al. (2019)

$$\frac{dh\,(t)}{dt} = f(h(t), t, \theta)$$

Where $h_t \in \mathbb{R}^D, t \in \{0 \ldots T\}$

Rather than using a series of fixed hidden layers, an ODE is used by a neural network to represent the derivative of the hidden units, allowing for more flexibility in adapting to the input data.

In the field of TS forecasting, the ability to adapt to incoming data streams may be a useful feature of neural ODEs. However, it is worth noting that these models may lack robustness and are not well-equipped to detect uncertainty in predictions, as noted by Anumasa and Srijith (2022).

Upon experimentation, Hasani et al. (2021) observed that the performance of neural ODEs was subpar compared to a simple LSTM network. Based on this finding, the author concluded that there is little justification for exploring other existing neural ODEs for TS forecasting.

2.5.5 Approach proposed by a Liquid Time-constant (LTC)

A remedy was put out by Hasani et al. (2020) to boost the neural ODEs' disappointing performance. The formulation shows 'liquid' qualities that lead to superior performance for TS predictions. The proposed architecture reveals stable and bounded behavior and is extremely expressive, enabling it to capture the dynamics in TS data better.

Additionally, it is hypothesized that the solution can learn both during training and while being used if the underlying formulations are continuously changed to accommodate variations in the input data (Ackerman, 2021). Furthermore, these networks have the advantage of being both resilient and stable while also being able to respond to changes in real-world systems.

The architecture is more detailed while being scaled-down than other neural networks, making it simpler to understand the specific computations within the network's black box, which is an advantage over conventional neural networks that are often criticized for their lack of interpretability.

The proposed formula is an alternative to what Funahashi and Nakamura (1993) proposed for solving differential equations. Unlike the approach taken by Chen et al. (2019), which uses a

neural network to define the derivative, the proposed formula includes an additional term that helps the system reach equilibrium.

The equation proposed by Funahashi and Nakamura (1993)

$$\frac{dX\,(t)}{dt} = -\frac{X(t)}{\tau} + f(X(t), I(t), t, \theta)$$

Where $-\frac{X(t)}{\tau}$ *is the additional term,* $X(t)$ *is the hidden state,* $I(t)$ *is the input, t represents time, and* f *is parameterized by* θ.

The equation proposed by Hasani et al. (2020)

$$\frac{dX(t)}{dt} = -\left[\frac{1}{\tau} + f(X(t), I(t), t, \theta)\right] X(t) + f(X(t), I(t), t, \theta) A$$

The above novel time-continuous RNN declares the hidden state by a system of linear ODEs parameterized by θ *and A.*

Hasani et al. (2020) further conducted experiments to justify their hypothesis producing promising results by beating other SOTA TS algorithms.

After considering the limitations of the previous approaches and the advantages of the LTC architecture, the optimal way to conduct this research is by utilizing the LTC approach. Despite the promising results of the LTC architecture, it is important to note that it uses ODEs, which are considered outdated by researchers (Duvenaud, 2021). Additionally, LTC models may not be suitable for rapidly changing data streams as they cannot quickly adapt to sudden changes in input.

A table summarizing these algorithms and a table describing a few associated studies are presented in ***APPENDIX B***.

2.5.6 Neural Stochastic Differential Equations (SDEs)

Neural ODEs proposed by Chen et al. (2019) utilize ODEs to abstract deterministic models. However, a natural question arises if a state of randomness exists: **What could be used in probabilistic models** (Raneez and Wirasingha, 2023)?

Neural SDEs share similarities with neural ODEs in that they can use an ODE solver, but they have a different focus on ‘stochastic evolution’ rather than ‘deterministic evolution’ (Tzen

and Raginsky, 2019). Additionally, given the expressive power of neural ODEs, Peluchetti and Favaro (2019) proposed that neural SDEs could further enhance the expressive power of these models.

SDEs are particularly useful for modeling data that exhibit sudden, small changes, such as market prices or molecule motion (Duvenaud, 2021). Given that the domain of implementation belongs to this category, it is advisable to focus on SDEs instead of ODEs from here on forward.

2.5.7 The final verdict – SDE-based liquid neural network

After this thorough examination, the exact next steps still need to be clarified. While the LTC architecture proposed by Hasani et al. (2020) introduces liquid adaptability, it cannot model randomness and instantaneous changes and is considered outdated.

A logical next step researchers could look into is drawing inspiration from the LTC architecture, swapping the underlying ODE for an SDE, and manifesting a novel and more flexible liquid neural network capable of handling high volatility (Raneez and Wirasingha, 2023). This algorithm, dubbed the 'Liquid Time-stochasticity' by the author, can be considered the author's core contribution. The design will be presented in **Chapter 6**.

2.5.8 Traditional required techniques

As the system would utilize multiple datasets, other more traditional techniques must also be reviewed.

2.5.8.1 Data preprocessing

Preprocessing is a generic step that must be performed before creating the combined enriched dataset.

Cleaning

Tweet data containing empty text fields must be ignored when calculating the average daily sentiment. Additionally, unnecessary content in text, such as hashtags, URLs, usernames, and links, must be removed (Abraham et al., 2018); however, as proposed by Hutto and Gilbert (2014), other techniques, such as punctuation and stop word removal, are best not performed. Moreover, tweets about stock trades and markets are mostly objective and have no genuine sentiment, and many of these tweets could be created by advertisements and bots, which also must be removed from the dataset (Abraham et al., 2018).

Data collected apart from tweets have a single value for each timestep. Therefore, empty rows cannot be removed, as the progression of time would be interrupted. However, imputation techniques, such as ‘linear interpolation’, could be performed to fill in these fields, as conducted by Mudassir et al. (2020).

Feature scaling

Normalization techniques are crucial to prevent unnecessary bias in each value and ensure they are all on the same scale. Google Trends, historical data, Twitter volume, and the block reward size may have different ranges and units of measurement. Scaling these values to a standard range allows for more meaningful comparisons and accurate modelling.

Feature selection

Feature selection is a crucial step that aims to select the most relevant input features for the model while excluding irrelevant or redundant ones. Correlation analysis is a widely used technique to measure the degree of association between features. The correlation matrix, Pearson correlation coefficient, and p-value are commonly employed to identify highly correlated features with significant impact (Abraham et al., 2018).

The researcher can identify the features that have the strongest correlation with the target variable and exclude those that do not contribute significantly.

2.5.8.2 Hyperparameter tuning

None of the literature the author reviewed included a hyperparameter tuning stage; therefore, the author would attempt it themselves. Hyperparameter tuning is performed to get the maximum out of the model as the hyperparameters become optimal. Multiple techniques are available for tuning ML models in the scikit-learn package, such as ‘grid search CV’ and ‘randomized search CV’. However, as this implementation will be a neural network with DL requirements, these packages cannot be used as they are computationally intensive.

Fortunately, tailored techniques, such as the Keras Tuner and HParams dashboard in TensorBoard, could be used to identify the best set of hyperparameters.

2.5.8.3 Validation

Validation techniques are required to ensure the robustness of the created model by ensuring that the predictions remain accurate in any condition. The literature reviewed by the author most

utilizes the K-fold cross-validation technique (ex: Valencia et al., 2019). However, as the system is TS-based, the Walk-forward cross-validation (Serafini et al., 2020) or cross-validation on a rolling basis (Shrivastava, 2020) should be used, which are specialized versions with the ability to handle temporal data.

Conducting validation tests will ensure that the implemented model is as robust as possible, which is vital, as the chosen application domain is notorious for failures.

2.5.9 NLP techniques that can assist

2.5.9.1 Sentiment analysis

Exogenous features must be present to create a robust system. As identified by Abraham et al. (2018), NLP techniques could be utilized to extract information and obtain meaning from the Twitter tweets. It is worth noting that there may be the presence of duplicate tweets, which libraries such as fuzzy-wuzzy (Pant et al., 2018) could remove.

While recent research has suggested the use of transformer architectures for extracting information and meaning from social media data (Wolf et al., 2020), the author believes that the VADER library is more suitable for sentiment analysis in the context of Twitter (Hutto and Gilbert, 2014; Valencia et al. 2019). Therefore, both VADER and transformer architectures will be evaluated for their performance in sentiment analysis.

It is worth mentioning that specific cleaning steps must not be performed. These steps include the removal of punctuation, capitalization, and the removal of stop words – especially ‘but’. Hutto and Gilbert (2014) developed five heuristics that are utilized within VADER; the most prominent four are described below:

- Punctuation – can increase the magnitude of intensity without modifying the sentiment.
- Capitalization – acts in a similar way to punctuation.
- ‘But’ – could signal a shift in polarity.
- Degree modifiers – act in a similar way to punctuation and capitalization.

2.5.10 Proposed architecture

Upon identifying the requirements to implement the system and considering required exogenous factors, the author proposes the following architecture to fulfil the research’s mentioned aim.

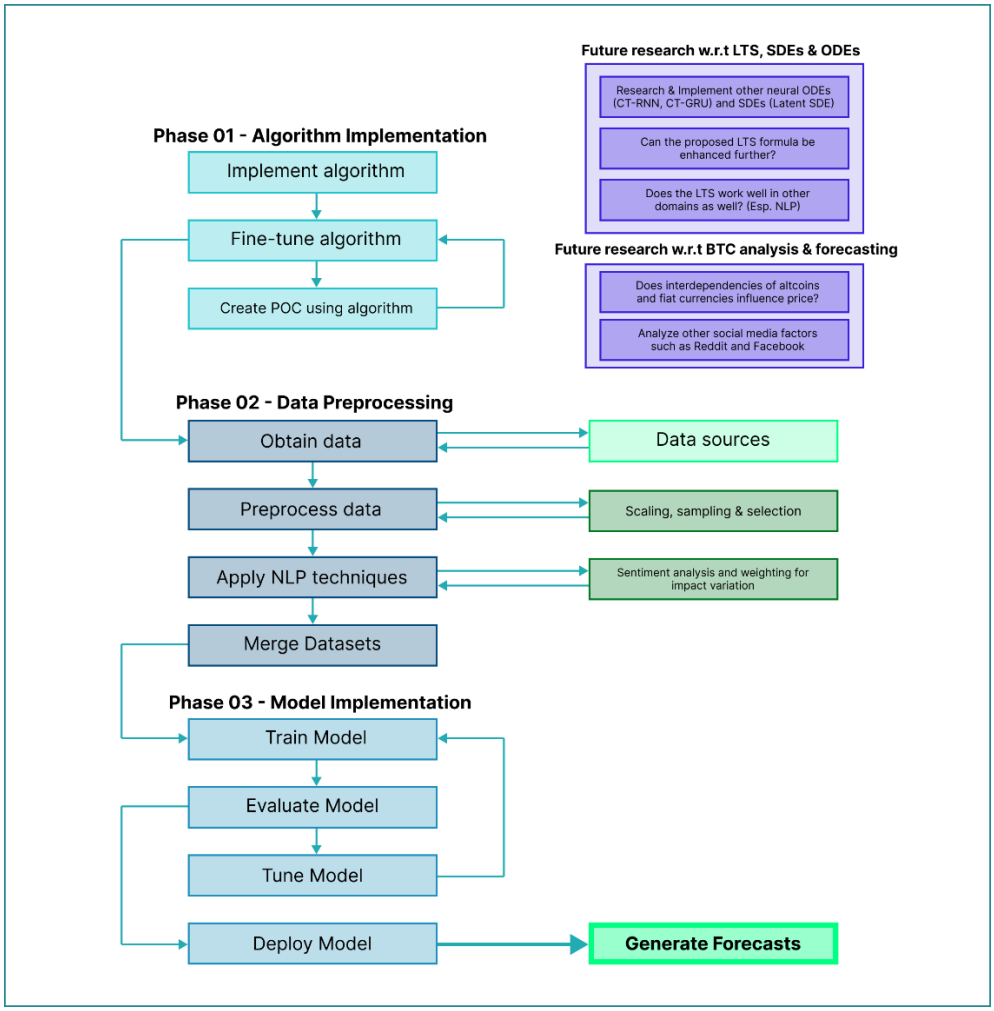


Figure 3: Proposed architecture (*self-composed*)

The aim of studying the above architecture is to fulfill the research's mentioned aim and identify any potential limitations or challenges that may arise during the implementation process. Additionally, the study of the architecture could lead to identifying areas for future research and improvement.

2.6 Evaluation

2.6.1 Evaluation approaches

Validation methodologies and evaluation metrics can evaluate the application by generating forecasts and comparing them against the test data.

The evaluation metrics: MAE, MSE, RMSE, and MAPE (Hyndman et al., 2021) can deduce how well the model performs, where the lower the value obtained, the better the

performance. It is worth noting that it can be misleading when values are close to zero: obtaining error values of zero is implausible and can indicate a problem with the model.

Table 2: Evaluation metrics for TS forecasting systems

Metric	**Description**	**Formulae**
MAE	Represents the average difference between the predicted and actual values and measures the average absolute distance between the predicted and actual values.	$\frac{\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert}{n}$
MSE	Measures the average squared difference between the predicted and actual values and is more sensitive to outliers than MAE.	$\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$
RMSE	Is the square root of MSE and is a popular metric as it is on the same scale as the target variable.	$\sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$
MAPE	Measures the percentage difference between the predicted and actual values and is particularly useful when dealing with variables with different scales. Helpful if the algorithm is evaluated in the M4 competition, as this metric is used widely in competitions since it does not have any units. Therefore, it can be compared across other datasets.	$\frac{100}{n} \sum_{i=1}^{n} \lvert \frac{y_i - \hat{y}_i}{y_i} \rvert$

An additional metric: MASE (Hyndman and Koehler, 2006), can be used as an alternative to percentage errors (ex: MAPE) as MAPE can 'explode'. Helpful when trying to beat the naïve forecast, which produces a MASE value close to 1; therefore, getting a value less than this would mean that the model performs better than the naïve forecast.

Besides these technical metrics, quality-of-service could be validated, such as CPU & memory usage, startup time, and application fluidity.

2.6.2 Benchmarking

2.6.2.1 System benchmarking

A standard dataset is required to conduct a valid benchmarking analysis on the system. As there is no previous system utilizing the LTC and no system combining all mentioned features in a non-linear model, the author will not be able to conduct a benchmarking analysis on the proposed system. However, conducting baselining to capture performance is feasible.

Furthermore, sharing the results publicly will contribute to advancing research in this domain and foster transparency and reproducibility in the TS forecasting community.

2.6.2.2 Algorithm benchmarking

The algorithm could be benchmarked in the M4 competition, as specific benchmarking datasets exist.

2.6.3 Research justification

To thoroughly evaluate the proposed solution, two evaluation factors need to be considered.

- **E**01 - evaluate the application against existing work of cryptocurrency forecasting. This comparison will assess the superiority of the proposed approach and its potential to improve the accuracy of cryptocurrency forecasting.
- **B**01 - benchmark LTS against other SOTA TS algorithms. This comparison will highlight the strengths and limitations of the proposed approach.

As identified, LTC applications do not exist that could be compared. Therefore, the author will compare against the current work in forecasting cryptocurrencies to satisfy **E**01. Additionally, if time permits, the author will attempt to fulfil **B**01 by benchmarking the algorithm in the M4 competition.

2.7 Chapter summary

This chapter presented the comprehensive background research process conducted by the author to propose the identified problem. The research was presented in a birds-eye view by a concept map and was further broken down into subsections that were individually evaluated and critiqued in detail. Additionally, the author highlighted possible future research opportunities upon completing this project.

CHAPTER 03. METHODOLOGY

3.1 Chapter overview

The research methodology must be defined to ensure a streamlined research journey. In this chapter, the author presents the chosen methodologies and discusses their rationale in depth. Additionally, this chapter includes a discussion of project requirements, potential risks and their mitigation plans, schedule, work breakdown plan, and associated deliverables.

3.2 Research methodology

Methodologies that are suitable for the research project have been evaluated and chosen from the predefined Saunders Research Onion Model (Saunders, Lewis and Thornhill, 2007, p102).

Table 3: Research methodology

Philosophy	The **positivism** philosophy was chosen: although the data collected will be a collection of nominal and ordinal data, the data collected will be encoded into numbers. Additionally, it is expected to validate/invalidate the proposed research questions using necessary evaluation comparisons.
Approach	The author chose the **deductive** approach instead of the inductive approach because the final analysis and evaluation would be quantitative in nature.
Strategy	The author chose **archival research** and **action research** as the data collection strategies. Since the research topic is modern, the principal source of data collection would be research documents. Action research will also be included since the research process will likely be an iterative approach of diagnosis, planning, action & evaluation. Additionally, the **survey** was chosen to assist the supplementary research by obtaining insights from end users.
Choice	**Multi-method** will suit the proposed research project most since qualitative analysis would complement the primary quantitative approach. However, it will not be used as a combination; hence the reason for not choosing the mixed method.

Time Horizon	The chosen time horizon for the research project is **cross-sectional** rather than longitudinal. This decision was made because the research will focus on independent data points that do not have an interlinking relationship over time. The latest available data will be obtained regularly to update the model.
Techniques and procedures	To optimize data collection and analysis, the researcher will draw from various sources while being mindful of available resources. Primary mediums include statistics, reports, journals, articles, and observations.

3.3 Development methodology

3.3.1 Life cycle model

A thorough analysis revealed that the Waterfall, V-shaped, and Incremental models were unsuitable. Sequentially completing the stages was unlikely, and revisiting previous stages was inevitable due to the need to develop a new algorithm (Sami, 2012). Partially completed components of the LTS must be independently evaluated and comprehended to ensure that the author can handle the situation. Therefore, **Prototyping** was chosen as the life cycle development methodology.

3.3.2 Design methodology

Structured System Analysis & Design Method (SSADM) was selected as the design methodology for its simplicity and ease of maintenance (madhurihammad, 2020). Given the extended development period, this is crucial. Python and React were chosen as software requirements, which do not inherently promote Object-Oriented Programming (OOP) but rather functions and modules. However, the design paradigm is less critical in the case of a data science project.

3.4 Project management methodology

The author analyzed different methodologies and concluded that Agile, Scrum, and Kanban were unsuitable for this project since they were designed for team-based collaboration. Therefore, the author decided to follow the **PRINCE2** methodology, which allows the project to be developed through subunits using a plan-based approach (Asana, 2022). This methodology was chosen

because it aligns well with the author's individual project needs and enables a more structured approach to project management.

3.4.1 Schedule

3.4.1.1 Gantt chart

The project's schedule is presented as a Gantt chart in **APPENDIX J**.

3.4.1.2 Deliverables

The deliverables and respective dates are specified in the table below.

Table 4: Deliverables & dates

Deliverable	**Date**
Literature Review. Critical analysis of related work & solutions.	27th October 2022
Project Proposal & Ethics Forms. The initial proposal of the research to be conducted.	9th November 2022
Software Requirement Specification. Defines the requirements that must be met to prototype and collect data.	24th November 2022
Proof Of Concept & Implementation Presentation. An initial implementation of the proposed system.	23rd December 2022
Project Specifications Design & Prototype. A prototype of the system with the core features and an accompanying document specifying the design followed & an overview of the implemented algorithm.	16th February 2023
Test & Evaluation Report. Documentation of test findings and evaluations conducted on the prototype.	23rd March 2023
Draft Project Report. A draft submission of the final thesis to get evaluations.	10th April 2023
Final Thesis. Final submission of the thesis with complete documentation of the project's journey.	27th April 2023

3.5 Resources

3.5.1 Software resources

- **Operating System (Windows/macOS**/Linux**) -** Windows will be the default since it provides easy access to the required development environments and tools. macOS will be utilized to create documentation due to its great battery life and low power consumption.
- **Python**/R - to create the network & the respective model. Python is chosen since it has a much simpler learning curve, provides easy integration with other mentioned software, and is optimized for large-scale ML software. Meanwhile, R is more suitable for statistical data analysis (IBM Cloud Team, 2021)
- **TensorFlow/**Torch – provides libraries that facilitate DL in Python & R. TensorFlow is chosen due to its large developer community, seamless integration with Keras, and the ability to provide multiple visualizations using TensorBoard (Dubovikov, 2018).
- **Flask/**Node/Golang – for seamless client and model communication. Flask will be the primary choice since the ML component will use Python, it is incredibly lightweight, and there is only a requirement for a minimal REST API.
- **React**/Vue/Svelte – required to develop the client side. A fast performant library is required to prevent lags and other performance issues. React will be the used due to its fast and performant library and the author's familiarity with it (Boisdequin, 2020).
- **MongoDB/**MySQL – to store the datasets and enable quick querying of the raw data. MongoDB is chosen due to its ability to store and process large amounts of raw data.
- **VSCode/**PyCharm – environment to facilitate application development. VSCode is the primary choice since it provides a general-purpose yet lightweight development experience with multiple plugins making it more developer-friendly. PyCharm will be a secondary option if there are issues with the Python environment or a need for a dedicated Python development environment.
- **Jupyter Notebook /** Kaggle – development environment for building the forecasting model. Jupyter will be the primary choice as it runs locally, which ensures training will not be interrupted in case of power failures. Kaggle will be the backup choice if a GPU is required to train the model or if a large memory capacity is required for loading the datasets.

- **Zotero**/Mendeley – manage references and research artefacts. Zotero is chosen due to the author's preference and easy to use.
- **MS Office** | **Overleaf** – tools to create primary research reports. A combination of MS Office and Overleaf will facilitate the drafting of project submission documents and the creation of professional research papers/surveys/reviews, if necessary.
- **Google Drive** / OneDrive – to backup research artefacts. Google Drive will be the primary option due to the unlimited storage availability provided by the university.
- **Figma** | **Canva** | **Draw.io –** tools to create figures and diagrams. Combining all three will streamline the design process, as each has different strengths. Figma - to design and prototype; Draw.io – to draw flow charts and other associated diagrams; Canva - to prepare attractive presentation slides.
- **GitHub**/Bitbucket/Gitlab – track, version & manage development code and research documents. GitHub will be the choice due to the author's familiarity, integrations with the development environments, and email notifications that could be significant.

3.5.2 Hardware resources

- A processor with a minimum of **Core i5 (8th gen)**, Ryzen 5, or M1 architecture – to handle long-running intensive workloads and manage multiple development environments. While the author has access to an Intel processor, the other options can also be considered.
- **At least 8GB RAM** – to manage model training, multiple development environments, loading large datasets, and performing simultaneous tasks.
- **At least 20GB of disk space** – to store application code & data.

If the available hardware does not meet the required criteria, a cloud-based development environment can be used (ex: GitHub codespaces, Google Colab, Zotero web, Google Drive).

3.5.3 Technical skills

- Design and development of TS forecasting systems using SOTA techniques.
- Expertise in ODEs, SDEs, and numerical solvers for these equations.
- Proficiency in implementing raw neural ODEs and SDEs.
- Ability to create optimized and scalable DL models that can handle complex tasks.
- Creating a seamless deployment pipeline.

- Ability to develop optimized client-side charts & user interfaces that dynamically update.

3.5.4 Data requirements

- **BTC price observations** – from a financial website (investing.com).
- **BTC block reward size**, **Twitter volume** and **Google Trends** - from a website that provides the required data publicly without any authorization (bitinfocharts.com).
- **BTC tweets** – Tweets up to an available date were taken from source 01 and source 02 and combined. The rest are extracted from a scraper that scrapes Twitter tweets for each day.

3.6 Risks & mitigation

The following table identifies possible risks that the author could face and how they could mitigate them.

Table 5: Risk management plan

Risk Item	Severity	Frequency	Mitigation Plan
Lack of required knowledge	5	5	Consult with domain experts and, if needed, the author of the LTC to gain valuable insights.
Corrupted documentation	4	4	Keep all necessary documentation in the cloud and external storage for easy access and prevent loss of important data.
Lose access to development code	5	2	Ensure the code is backed up on source control and cloud storage.
Inability to deliver all expected deliverables	4	2	Prioritize tasks and deliverables to ensure the timely completion of each objective.

3.7 Chapter summary

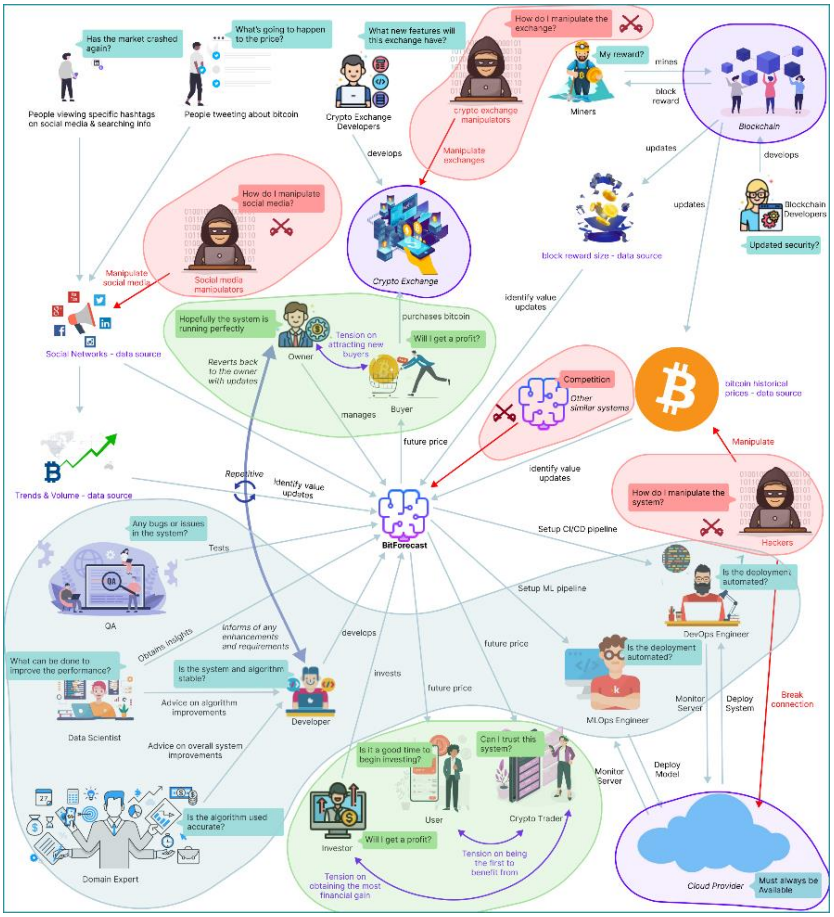
This chapter defined the proposed methodology followed by the author and discussed its reasoning. Then, the author discussed the research, development, and project management methodologies followed. Finally, the project requirements, work breakdown plan, associated deliverables, and the risks the author could face and their mitigation plans were specified.

CHAPTER 04. SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Chapter overview

Requirement gathering is essential to ensure that the research is performed to the best possible quality and to develop software that would be helpful in the real world. In this chapter, the author focuses on identifying the requirements and the steps followed to gather these requirements. In detail, possible stakeholders, their interaction points, and roles are documented using a rich picture diagram and a stakeholder onion model. Furthermore, the requirement-gathering techniques followed and the insights obtained upon analysis are discussed to produce necessary functional and non-functional requirements, system diagrams, and prototype descriptions.

4.2 Rich picture



A clearer version can be found Here

Figure 4: Rich picture diagram (*self-composed*)

The above diagram presents a helicopter view of the wider environment, depicting how particular stakeholders interact with the system and the benefits they derive from it. It also highlights any negative impacts, concerns of stakeholders, geographical localities, and stages of construction. This information can help the researcher identify areas of improvement for the system.

4.3 Stakeholder analysis

The following section presents the key stakeholders involved in the system, their relationships, and their roles. This information is illustrated using the stakeholder onion model, which visually represents the stakeholders' level of involvement with the system. Additionally, the stakeholder viewpoints are discussed in detail to understand their perspectives.

4.3.1 Stakeholder onion model

The following diagram illustrates the key stakeholder interaction in an onion model.

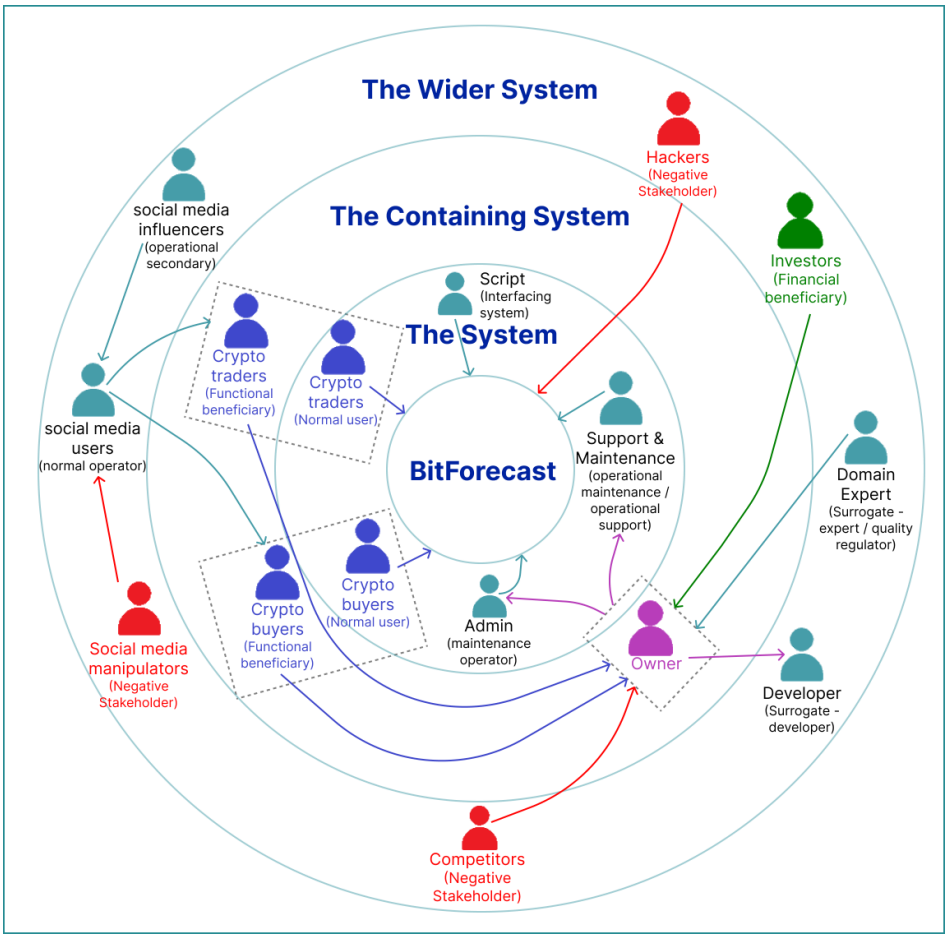


Figure 5: Stakeholder onion model (*self-composed*)

4.3.2 Stakeholder viewpoints

The following table describes the stakeholders, their roles, and associated actions.

Table 6: Stakeholder viewpoints

Stakeholder	**Role**	**Benefits/Description**
Support & Maintenance	Operational – support & Operational - maintenance	Maintains the health of the system and attends to user inquiries.
Admin	Maintenance operator	Monitors and updates the system when necessary.
Script	Interfacing system	Makes sure the system and data are updated.
Owner	Owner & Operational - administration	Manages other operators, listens to feedback, and communicates with other stakeholders.
Crypto trader	Functional beneficiary	More convenient for deciding whether to purchase or sell currently held assets.
Crypto buyer		
Investor	Financial beneficiary	Makes profit by investing in the system, upon marketing, or user subscriptions.
Domain expert	Surrogate – expert & Quality regulator	Provides advice on overall system improvements.
Developer (includes all roles in the rich picture)	Surrogate – developer	Develops (or helps in developing) the system.
Social media influencers	Operational - secondary	Influence users, drive trends, and provide thoughts.
Social media users	Normal operator	Get influenced to invest or sell currently held assets.
Competitors	Negative stakeholder	Build competing products that outperform or have better value.
Social media manipulators		Manipulate set trends and influencer thoughts.
Hackers		Disrupt the system and corrupt data.

4.4 Selection of requirement elicitation methodologies

Researchers can carry out requirement elicitation methodologies to gather requirements. The following table discusses the selected ones and their purpose.

Table 7: Selected requirement elicitation methodologies

Method 01: Literature review
An exhaustive literature review has been conducted to identify a respectable research gap in a cutting-edge research field and a domain of interest. The author studied existing systems to determine limitations and future research. A brief understanding of the implementation methods was also identified, alongside necessary best practices.
Method 02: Observations
Upon conducting the literature review, analysis of similar systems is an added advantage. Validating and evaluating its viability is paramount as the chosen research domain is relatively new. Therefore, existing algorithmic POCs must be studied and thoroughly assessed, as this will provide the author with the necessary insights and techniques to implement.
Method 03: Interview
Interviews can help gather knowledge and insights into more theoretical concepts that will be helpful behind the scenes for implementing the research component and associating with and answering the proposed research questions. The author can interview specific niche experts with knowledge of neural ODEs and SDEs to obtain said intuition, which they cannot acquire by conducting a survey.
Method 04: Survey
Obtaining insights and expectations from end users can be gathered by conducting a survey, specifically, the questionnaire. Upon receiving this prominent information, the author can decide whether the proposed system is helpful for the target audience and understand how the target audience intends to benefit from it. As they are large in sample size, the survey is a powerful choice for data collection.
Method 05: Prototyping
Prototyping will allow the developer to iterate between implementations and improvements; as the architecture is more novel, this procedure will be used abundantly as a straightforward approach to obtaining the optimal performance is unlikely and will take time.

Table 8: Omitted requirement elicitation methodologies

Method	Reason
Self-evaluation	The process of self-evaluation is employed to gain insight into the perspective of the target audience. However, it is deemed unnecessary since a survey can provide a more precise means of gathering requirements.
Brainstorming	The author decided brainstorming was not necessary, as interviews and observations would encompass its findings.

4.5 Discussion of findings

The essential stakeholders were separated into groups, and each group was analyzed in the most suited methodology. The table breakdown of these stakeholders is available in **APPENDIX C.1**.

4.5.1 Literature review

The below table discusses the key findings upon conducting an extensive literature review.

Table 9: Literature review findings

Citation	Discussion of findings
Research domain	
Hasani et al., 2021	Existing solutions in TS forecasting use traditional neural nets or statistics.
Hasani et al., 2020	Although traditional neural ODEs have been used for TS forecasting, they have been found to underperform compared to existing neural networks
Duvenaud, 2021	The LTC architecture uses the ODE, which is outdated and lacks rapid adaptability. By using an SDE instead, flexibility can be improved. Therefore, combining both would produce the optimal architecture.
Problem domain	
Abraham et al., 2018; Valencia et al., 2019	Based on the reviewed literature, work that included multiple exogenous features had not utilized a non-linear model.
Fleischer et al., 2022; Serafini et al., 2020	Work that used a non-linear model had not included the additional features the author aims to include. As implied, a non-linear model with multiple features would produce the optimal solution.

4.5.2 Observations

The below table discusses the criteria of conducing observations and its respective findings.

Table 10: Observations findings

Criteria	Discussion of findings
To find approaches to creating a neural SDE to implement the core research component	The author observed a scarcity of POCs of neural SDEs applied to ML systems, which implies that the proposed solution could pioneer a new approach in this field. Moreover, the research component developed in this work could serve as a foundation for future neural SDE implementations.
To find approaches taken to implement the BTC forecasting model.	Although there are numerous POCs of BTC forecasting systems that use LSTMs and statistical algorithms, they commonly rely only on the closing price as a feature or the closing price with the Twitter sentiment. However, the author observed that these approaches overlook the potential impact of other exogenous features. Thus, the decision was made to prioritize building the primary research component first, which can be utilized to create the BTC forecasting system.

4.5.3 Interviews

Interviews were conducted to obtain domain expertise and any information that the author may have missed, which could be significant. The author interviewed only a few candidates as the research domain is new and unknown; fortunately, they were the most knowledgeable. The author also interviewed a candidate experienced in the problem domain area. The findings were analyzed using thematic analysis and presented below. The participants' affiliations and areas of expertise are available in **APPENDIX C.2** with supporting evidence for each finding.

Table 11: Interview thematic analysis codes, themes & conclusions

Code	Theme	Conclusion
Research component		
Algorithm architecture	Research Problem & Gap	The interviewees confirmed the existence of a research gap and the defined problem. They also appreciated the author's efforts

		in conducting this research, as only a few papers have been published in this domain.
Resource intensive	Requirements	The interviewees expressed concerns regarding the computational cost of ODEs and SDEs, highlighting that this could result in longer processing times and potentially lead to user-unfriendliness. Thus, the author should optimize the model as much as possible to ensure the forecasts can be produced quickly without compromising accuracy.
Obsolete, Inflexible	Advice	The author originally intended to implement only the LTC architecture proposed by Hasani et al. (2020). However, the author could enhance the architecture by using SDEs instead of ODEs, resulting in the development of the LTS. This decision was made due to the potentially significant contribution and greater significance offered by the LTS architecture.
Visualizations, Explainability	Other suggestions	The author concluded that while XAI is prevalent in image classification, there is a lack of literature on its application in the time series domain. Moreover, incorporating XAI into time series modelling may be challenging due to the temporal component. The author also noted a need for research on XAI for SDEs, which they may explore if time allows.
Problem domain		
External features and trends	Robustness	The interview provided additional validation for the data collected in the survey, as the participants suggested utilizing as many exogenous features as possible to make the model more robust. The author will consider this feedback and ensure the model uses the mentioned features.

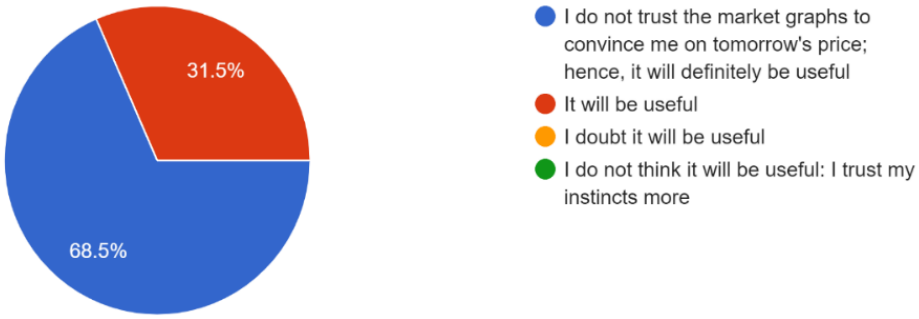
4.5.4 Survey

A survey was carried out to collect requirements from the target audience to identify the functionalities to implement for the developed supplementary product.

Table 12: Survey analysis

Question	How much would a system capable of assuming tomorrow's price benefit you?
Aim of question	To identify whether the system is beneficial in the first place

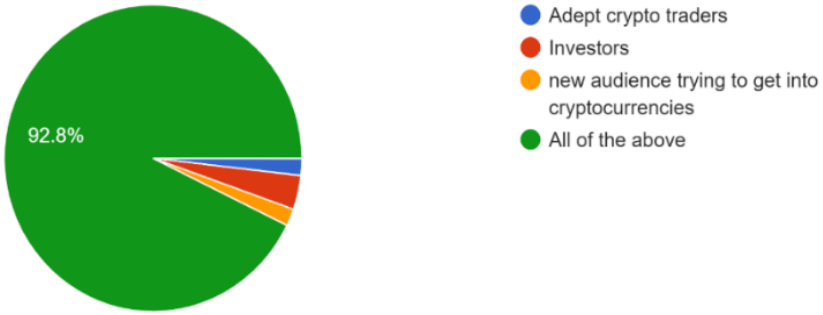
Findings & conclusions



The survey results indicated that all participants perceived the proposed system as highly beneficial, with the majority strongly believing in its usefulness. This finding proves that the supplementary system under development will be valuable. Importantly, none of the participants expressed concerns about the system's potential lack of utility, thereby validating the identified problem domain and enabling the author to proceed confidently.

Question	Who do you think would benefit from this system?
Aim of question	To identify beneficiaries and target audience

Findings & conclusions

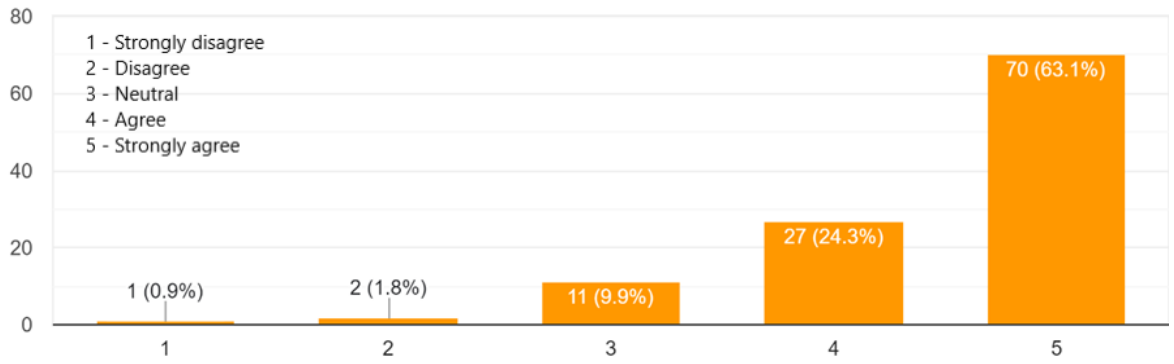


The survey revealed that a significant majority of participants believed that the proposed system would benefit both expert traders and investors, as well as new audience members. However, a small percentage of participants believed the system would primarily benefit those already in the market. This highlights the need for the system to be user-friendly and easy to understand to

attract a newer audience. Additionally, ensuring that the system is simple enough and mature to provide value to expert traders and investors is important.

Question	This system will also benefit people who are not experts in cryptocurrencies
Aim of question	To identify whether non-technical crypto traders would benefit

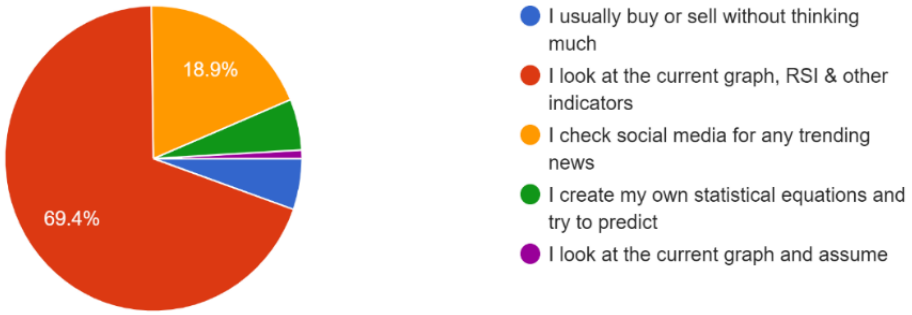
Findings & conclusions



The responses to the previous question confirm that the proposed system can target a broader audience, including individuals, not cryptocurrency experts. It is essential to note that this finding aligns with the author's objective of making the system accessible to a wider range of people rather than solely focusing on a niche audience of current investors and traders.

Question	How do you decide whether to buy or sell assets?
Aim of question	To understand how a buyer/seller proceeds with their decision

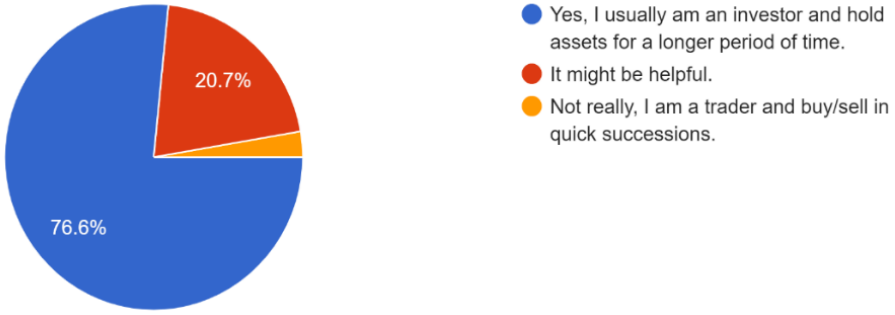
Findings & conclusions



The responses to the above question do not have a specific project-related purpose. However, a significant portion of the respondents has some knowledge of cryptocurrencies. Approximately 70% of the respondents reported having experience trading or investing in cryptocurrencies.

This information provides valuable insights for the author, suggesting that the respondents have specific knowledge that could be leveraged during the evaluation phase.	
Question	Do you think predicting a more future date (ex: a week from now) is as important as tomorrow's price?
Aim of question	To identify whether a greater future date prediction is also necessary

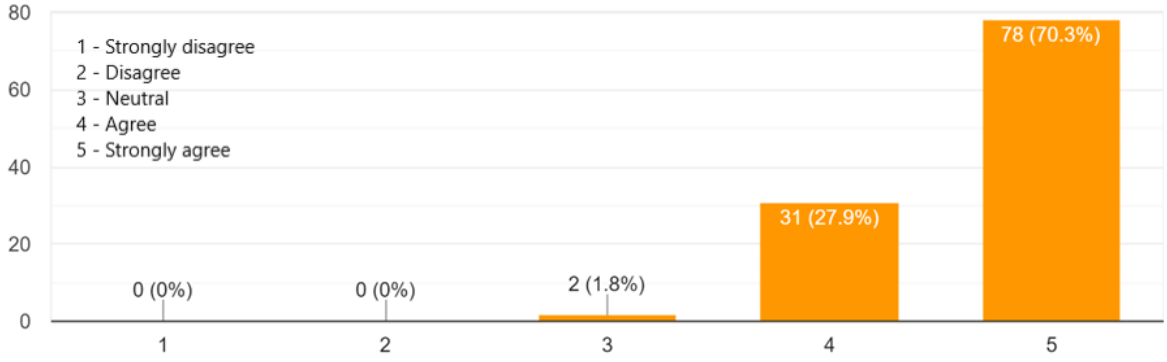
Findings & conclusions



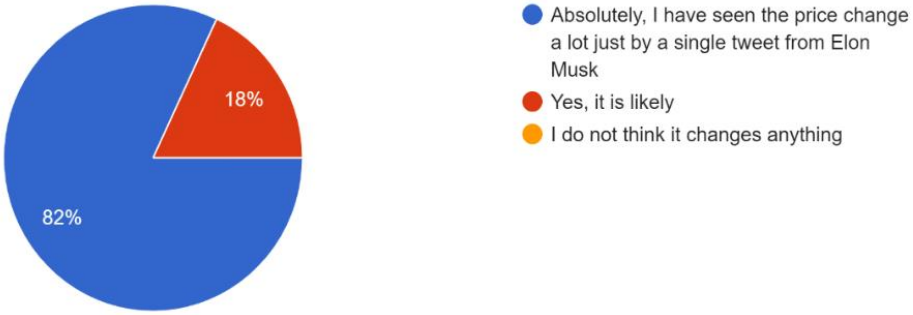
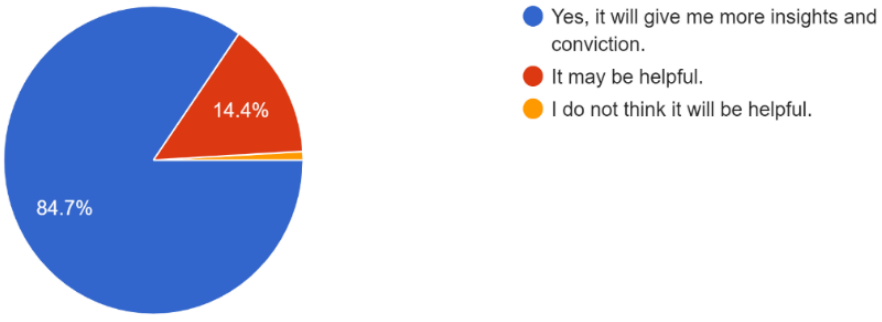
Considering the limited time, the author initially considered only having a single horizon forecast. However, based on the above responses, it is evident that the audience would also expect forecasts for multi horizons. Therefore, the author will additionally aim to implement the ability of multi-horizon forecasting.

Question	Social media trends can impact the price
Aim of question	To identify whether the community believes that social media trends impact the price

Findings & conclusions



The survey results indicate that a significant number of respondents believe that social trends have an impact on cryptocurrency prices. As a result, it is crucial to consider as many relevant

social trends as possible. While the project has a limited scope and timeline, the author has decided to incorporate Twitter volume and Google Trends data. However, Reddit, Facebook, and others could also provide valuable insights and should be considered for future work.	
Question	If a highly influential person tweets about Bitcoin, do you expect the price to tip to the side in favor of their tweets meaning?
Aim of question	To identify whether including Twitter sentiment is beneficial and to confirm the problem domain contribution.
Findings & conclusions  All participants acknowledged that social media somehow impacts the price of cryptocurrencies. Most participants further thought that the tweeter's influence adds additional significance. The responses to the previous two questions validate the contribution of the problem domain. Based on these findings, it is necessary to give more weightage to specific Twitter users' sentiments based on their influence.	
Question	Would it be helpful to obtain a range of prices rather than a point price? (Ex: 10,000 - 15,000 instead of 12,500)
Aim of question	To identify whether including uncertainty estimates is beneficial
Findings & conclusions 	

<table>
<tr><td colspan="2">The author initially decided on only providing a point forecast for the system, as this research aims to develop a novel architecture for TS forecasting. However, based on the responses and during the prototyping phase, it became evident that a single-point prediction may be less valuable than a range of prices. A point prediction may only be partially accurate, which makes the requirement of uncertainty estimates more crucial.</td></tr>
<tr><td>Question</td><td>What functionalities would you expect to have in a bitcoin forecasting system?</td></tr>
<tr><td>Aim of question</td><td>To identify any additional requirements</td></tr>
<tr><td colspan="2">Findings & conclusions
Thematic analysis was conducted to analyze the responses to the open-ended questions, and the results are presented in APPENDIX C.3.

The analysis revealed that the participants expressed a desire for Explainability in the system, and the author will consider incorporating XAI if time permits. The participants also emphasized the importance of utilizing exogenous factors to make the system more robust while keeping it simple for a newer audience. Based on these findings, the author plans to make the use of exogenous features mandatory and maximize Explainability in the system.</td></tr>
<tr><td>Question</td><td>Any extra feedback you would like to provide?</td></tr>
<tr><td>Aim of question</td><td>No specific reason – is mainly used to obtain any additional feedback</td></tr>
<tr><td colspan="2">Findings & conclusions
The respondents provided motivational statements to encourage and inspire the author to perform their best.</td></tr>
</table>

4.5.5 Prototyping

Upon iterative prototyping, challenges that the developer did not expect to arise emerged. Challenges ranged from finding a suitable dataset to implementing the algorithm itself. The below table discusses the criteria for conducting prototyping and its respective findings.

Table 13: Prototyping findings

Criteria	**Discussion of findings**
To assess the viability of creating the primary research component.	Building the algorithm was challenging, as no proper references were available. The author soon realized that implementing the algorithm required a deeper understanding of SDEs, differential solvers, and traditional deep learning theories. As a result, a direct implementation was not possible. Hence, the author decided to implement the LTC architecture proposed by Hasani et al. (2020) as a starting point, and then built upon it to develop the LTS algorithm.
To explore the feasibility of developing the BTC forecasting application.	The author initially planned to use the Twitter API to collect tweet sentiment for specific days. However, they discovered that Twitter had updated its API to only provide tweets from the past seven days. To work around this limitation, the author had to use a third-party library to scrape tweets from dates beyond that point in time. During experimentation, the author realized that a single-point price prediction would be insufficient, and a range of uncertainty estimates would be more helpful. Additionally, the author acknowledged that explainable insights from the neural networks could provide valuable intuition into the forecasting process.

4.6 Summary of findings

The below table summarizes the findings to gather the requirements for the MVP.

Table 14: Summary of findings

ID	Finding	Literature Review	Observations	Survey	Interview	Prototyping
Research domain						
1	Validate research domain and gap.	✓	✓		✓	
2	The novelty of the research hypothesis (an architecture inspired by the LTC).	✓	✓		✓	
3	Neural ODEs are an advancement for TS forecasting.	✓			✓	
4	Innovate by integrating latent SDEs into the LTC architecture, creating a novel algorithm better suited to capture high volatility.				✓	✓
Problem domain						
5	The system will be of use to experts and new audiences.			✓		
6	Social trends are a significant source of impact.	✓		✓		✓
7	Opinions from Well-known influencers have a more drastic impact.	✓		✓		
8	A non-linear model that combines all exogenous features has yet to be explored.	✓				
9	Rather than a single-point prediction, a range of prices increases the system's credibility.			✓		✓
10	The addition of an Explainability component would significantly improve the system's credibility.			✓		✓
11	A system that can adapt its hyperparameters would be highly valuable to experts.			✓		

4.7 Context diagram

The following diagram depicts the system's boundaries and interactions. By identifying them before development, the author gains insights into how information should flow within the system.

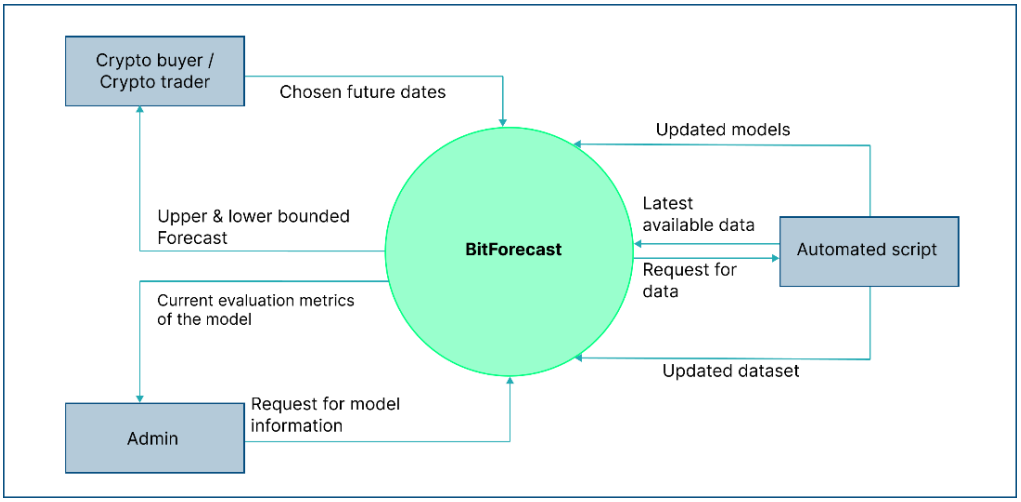


Figure 6: Context diagram (*self-composed*)

4.8 Use case diagram

The diagram presented below illustrates the fundamental use cases of the proposed system through 'sea level', outlining the high-level functionalities that the system will offer to its end-users.

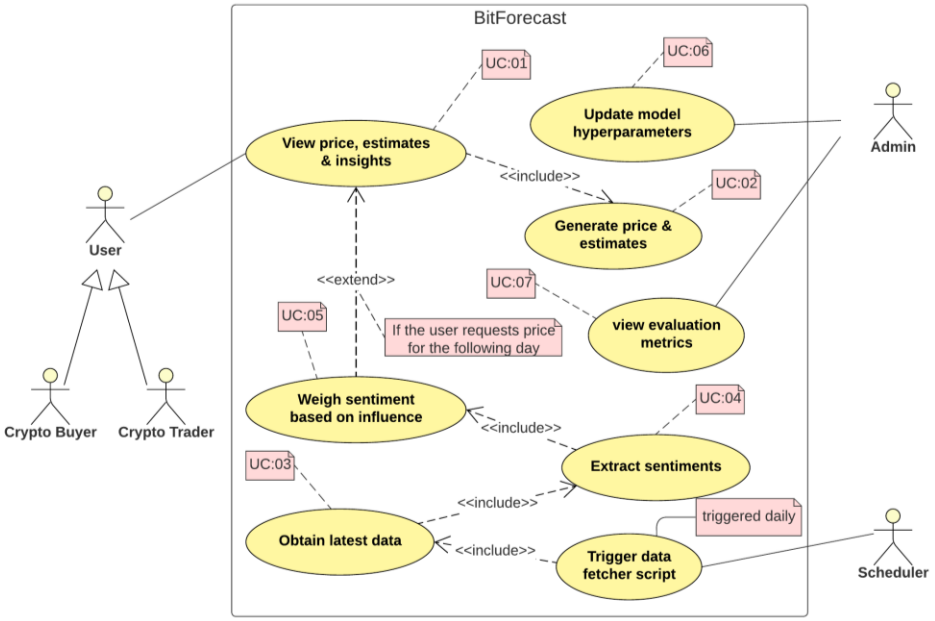


Figure 7: Use case diagram (*self-composed*)

4.9 Use case descriptions

The core use case description is presented below; any sub-descriptions are available in **APPENDIX C.4**.

Table 15: Use case description UC:01

Use case	Display price & estimates
Id	UC:01
Description	Display future prices and their corresponding uncertainty estimations based on the selected date, along with any relevant Explainability insights.
Actor	User
Supporting actor (if any)	Scheduler
Stakeholders	Crypto buyer, crypto trader
Pre-conditions	All the data must be scraped, preprocessed, and the forecast must be generated.
Main flow	MF1. The user requests tomorrow's price. MF2. The system recognizes the need to utilize available exogenous features. MF3. The system verifies that all data is up-to-date. In the event that the data is outdated, the system takes the following steps: 1. Obtains the latest available data. 2. Performs sentiment analysis and self-retrains the model. MF4. The system generates the price and upper and lower estimations. MF5. Display output to the user, including any relevant insights.
Alternative flows	AF1. The user requests the price for a date beyond tomorrow. AF2. The system recognizes the inability to utilize other features. AF3. The system generates price and upper and lower estimations. AF4. Display output to the user, including any relevant insights.
Exceptional flows	EF1. The system could not generate a prediction – display a user-friendly error message.
Post-conditions	The user is displayed with a forecast and necessary insights.

4.10 Requirements

The prioritization of the functional and non-functional requirements was performed using the 'MoSCoW' technique, which is elaborated on in **APPENDIX C.5**.

4.10.1 Functional requirements

Table 16: Functional requirements

FR ID	Description	Priority	Use Case
Research level			
FR1	A robust and scalable implementation of the LTS must follow recommended standards.	M	-
FR2	The developed algorithm must be capable of being used as existing layers and algorithms (ex: LSTM, CNN).	M	-
System level			
FR3	Users must be able to choose a future date.	M	UC:01
FR4	Users must be able to view the point prediction price.	M	UC:01
FR5	The system must generate the point prediction price based on the user's choice of date.	M	UC:02
FR6	The script must obtain the latest data available periodically.	M	UC:03
FR7	The script must extract trends and sentiments from obtained data.	M	UC:04
FR8	Users should be able to view a range of prices along with the single-point price.	S	UC:01
FR9	The system should generate higher and lower bound uncertainty estimations.	S	UC:02
FR10	The GUI should plot the forecast with the current prices in a single graph to show the growth/decline.	S	UC:01
FR11	The script could weigh sentiment based on tweets from influential individuals in the cryptocurrency space.	C	UC:05
FR12	The system could display some insights to the user, such as a highly influential tweet that made it predict the price.	C	UC:01

FR13	Admins could authenticate and update the model with different parameters.	C	UC:06
FR14	Admins could get additional information about a prediction, such as the evaluation metric and accuracy.	C	UC:07
FR15	The system will not produce forecasts for other cryptocurrencies.	W	N/A
FR16	The system will not produce real-time forecasts (ex: hourly).	W	N/A

4.10.2 Non-functional requirements

Table 17: Non-functional requirements

NFR ID	Requirement	Description	Priority
NFR1	Performance	The system should be optimized to generate forecasts efficiently, even when utilizing additional features.	M
NFR2	Performance	The system should update its data only when necessary to avoid unnecessary overhead.	M
NFR3	Usability	The user interface should be intuitive providing clear and concise error messages in case of any issues	M
NFR4	Maintainability	The codebase should be thoroughly documented to ensure future maintainability, especially for the algorithm development repository.	M
NFR5	Quality	The system output should be high-quality, providing valuable insights to users.	S
NFR6	Scalability	The system should be deployed to a scalable cloud environment with ample resources to handle multiple concurrent user requests.	S
NFR7	Security	The system should be designed with security to prevent data manipulation and protect against attackers, especially to avoid tampering with the used datasets.	S
NFR8	Compatibility	Compatibility testing should be performed on browsers and mobile devices to ensure a smooth user experience.	S

NFR9	Availability	A dedicated primary operator should be available to handle critical issues promptly.	C

4.11 Chapter summary

In this chapter, the author identified the essential stakeholders interacting with the system and outlined the nature of their interaction, leveraging a rich picture diagram and Saunder's Onion model. The chapter also delved into various requirement elicitation techniques, explaining their underlying rationale and subsequent outcomes. Lastly, the author defined the use cases, accompanied by their detailed descriptions, and established the system requirements that must be satisfied.

CHAPTER 05. SOCIAL, LEGAL, ETHICAL & PROFESSIONAL ISSUES

5.1 Chapter overview

This chapter presents an overview of any social, legal, ethical, or professional issues that have emerged during this project, along with the measures taken by the author to address these concerns.

5.2 SLEP issues and mitigation

5.2.1 Social

The author took several measures to ensure anonymity and privacy. They excluded the names of interviewees and evaluators in the dissertation to prevent potential bias or prejudice, even though written consent was obtained. Consequently, personal details were not collected during the survey, respondents were kept anonymous, and no IP addresses were tracked. The information gathered has not been stored elsewhere or presented in this thesis to ensure privacy further.

The application is not designed to replace jobs. Its primary focus is developing a theoretical-based approach by formulating the LTS algorithm to overcome the limitations of highly volatile data modeling rather than attempting to achieve the best-performing BTC forecasting solution. Furthermore, to prevent any bias or discrimination, only numerical data was used to develop the model.

In terms of personal relationships, the application is more likely to foster social connections than destroy or replace them as the cryptocurrency community continues to expand. The application could facilitate communication and collaboration between individuals interested in BTC and TS forecasting, creating new connections and opportunities for knowledge exchange.

5.2.2 legal

The datasets used in this research are available and accessible to the public free of charge. The author did not implement FR12 to prevent legal issues; hence, no names, including in Twitter tweets, were recorded. All the utilized tools and technologies are open source or are provided to students free of charge. Furthermore, the codebase is available on GitHub with the open-source MIT license, ensuring that it is accessible by anyone without any legal issues.

The application does not rely on an autonomous machine learning system, and the underlying algorithm is transparent, making it possible to identify potential issues, bias, or discrimination. Additionally, the application is compliant with the General Data Protection Regulation (GDPR) enforced by the European Union.

5.2.3 Ethical

All participants were informed of the data's intended use and obtained consent before collection. The work presented in this report is genuine and not reproduced elsewhere. All ideas taken from other sources, including the author's previous work, are cited and credited to avoid plagiarism. Ethical guidelines were followed, and responses and evaluations were not manipulated or altered.

Understanding the inner workings of AI systems can be challenging as they often function as a black box. However, the LTS algorithm has overcome this limitation by offering transparency and revealing all crucial information, making it easier for users to comprehend the behind-the-scenes process.

5.2.4 Professional

Professional standards recommended in the industry were followed as much as possible, leading to no illegal tools and software being utilized, ensuring professional conduct. The limitations of the project are clearly stated in **Chapter 10**, ensuring readers are aware of the extent of the work.

As stated, the application poses no risk of harm to its users or threatens their job security. To ensure that the necessary expertise was available at every stage of implementation, the author conducted extensive research on the most suitable techniques to apply, regardless of whether they were previously familiar with those methods.

In conclusion, the research had no significant SLEP issues that required addressing, as it strictly adheres to the principles articulated by the Association for Computing Machinery (ACM) of honesty, integrity, privacy, and responsibility (Siqueira De Cerqueira, Acco Tives and Dias Canedo, 2021). Consequently, this research falls under Westminster's Class 1 category, indicating no ethical issues were identified.

5.3 Chapter summary

The chapter detailed the social, legal, ethical, and professional issues associated with this research and the author's decisions to mitigate such risks.

CHAPTER 06. DESIGN

6.1 Chapter overview

Detailed design is required to ensure that implementation is as seamless as possible. The author selects appropriate architectural structures in this chapter based on the gathered requirements. Specifically, the high-level, low-level, and associated design diagrams are presented along with necessary UI wireframes. It also discusses the LTS architecture and presents a new tweet sentiment weighting formula developed for the project.

6.2 Design goals

The design goals that should be achieved by the system are specified in the table below.

Table 18: Design goals of the proposed system

DG ID	Goal	Justification
DG1	Performance	Traditional TS forecasting models require retraining every time a new prediction is made because the data used to train the model may become outdated. However, using multiple features in the proposed system could significantly impact performance due to the need for frequent retraining. To avoid this, the author suggests storing past data and retrieving only the necessary data when required.
DG2	Usability	Based on the analysis obtained during the requirement-gathering phase, it was found that opinions were divided on whether the application would be useful for non-experts in cryptocurrencies. Therefore, it is essential to prioritize the development of a user-friendly system that users of all levels of expertise can easily use.
DG3	Quality	The output quality must meet the highest standards to ensure the system's reliability, which can be achieved by displaying a range of prices instead of a single-point price. As identified in the gathered requirements, this is necessary to increase confidence in the system's predictions.

DG4	Maintainability	As stated by the author, the research project's success depends on delivering two products. The maintainability of the proposed research product is crucial, and the algorithm architecture must be designed to be optimal and independent to serve as a reference for future research.

6.3 System architecture design

6.3.1 Architecture diagram

The high-level architecture design of the system is presented below, which adopts a three-tiered architecture. This architecture was selected due to the clear separation between the presentation, logic, and data layers, ensuring each layer performs its specific function effectively.

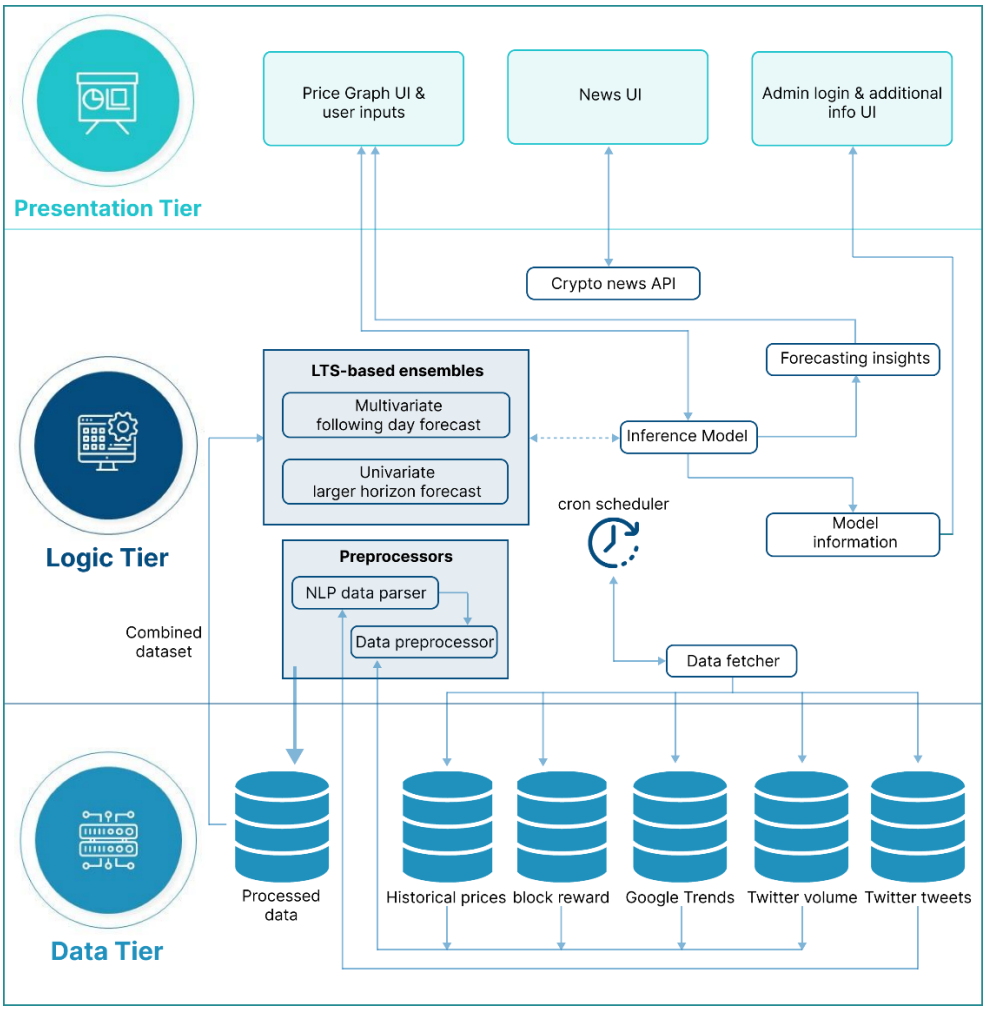


Figure 8: Three-tiered architecture (*self-composed*)

6.3.2 Discussion of tiers of the architecture

Data Tier

The data layer of the proposed system retrieves data from various sources via an API. It stores them as individual collections in MongoDB allowing efficient access to up-to-date data whenever required. The data collections used in the system include:

- Historical prices – historical closing prices of BTC for the past several years.
- Block reward size - block reward obtained for mining BTC.
- Google Trends – daily historical search volume of BTC-related keywords.
- Twitter volume – historical volume of BTC-related tweets posted each day.
- Twitter tweets – historical tweets that are BTC-related.

Logic Tier

The logic tier is responsible for processing the data obtained from the data tier and generating output presented in the presentation tier. It includes the following components:

- Preprocessors – consist of code required to process the raw data fetched from the APIs so that the forecasting model can use it. The preprocessor includes two modules:
    - Data preprocessor – performs general preprocessing steps such as normalization and cleaning.
    - NLP data parser – performs sentiment analysis on the tweet data and gives more weightage to a tweeter's sentiment based on certain factors.
- Data fetcher & cron – a script that automatically runs on a schedule to ensure the data and model are up-to-date.
- Forecasting models – models that provide forecasts.
    - Multivariate following-day forecast – utilized for the following-day forecasts.
    - Univariate greater horizon forecast – utilized for forecasts requested days ahead of the following day.
- Model information – presents extra information about the model that the admin could view, such as the current evaluation metrics.
- Forecasting insights – provides additional information to the user to demonstrate forecasting-related Explainability.

- Crypto news API – an additional third-party API to provide users with daily cryptocurrency news.

Presentation Tier

The point of interaction where the user interacts with the system.

- Price graph UI & user inputs – main UI of the MVP presented to the user. It would display the current pricing graph, provide the user options to choose a future date, and generate a new chart with the inference.
- News UI – a minor sub-feature that will display news about the cryptocurrency world.
- Admin login & additional info UI – a 'could have' feature that will provide an authorized user to obtain information about the current model in use and adjust hyperparameters to retrain the model.

6.4 Detailed design

6.4.1 Choice of design paradigm

As discussed in Chapter 3, the chosen design methodology for this project is **SSADM**. This decision was made because the research objective is to develop a novel architecture with a new algorithm, and extensive experimentation is crucial. Additionally, the selected programming languages do not emphasize OOP but instead promote the use of function-based modules and components.

6.4.2 Data flow diagrams

The author decided to use the 'Yourdon & DeMarco' (DeMarco, 1979) notation for designing the data flow diagrams to ensure consistency throughout the three levels.

The data flow diagrams are depicted using level 0, level 1, and level 2, where level 0 is the **context diagram** presented in the SRS chapter.

6.4.2.1 Level 01 data flow diagram

The level 01 diagram provides a detailed breakdown of the core components proposed in the context diagram, showing how they interact with each other to achieve the system's goals.

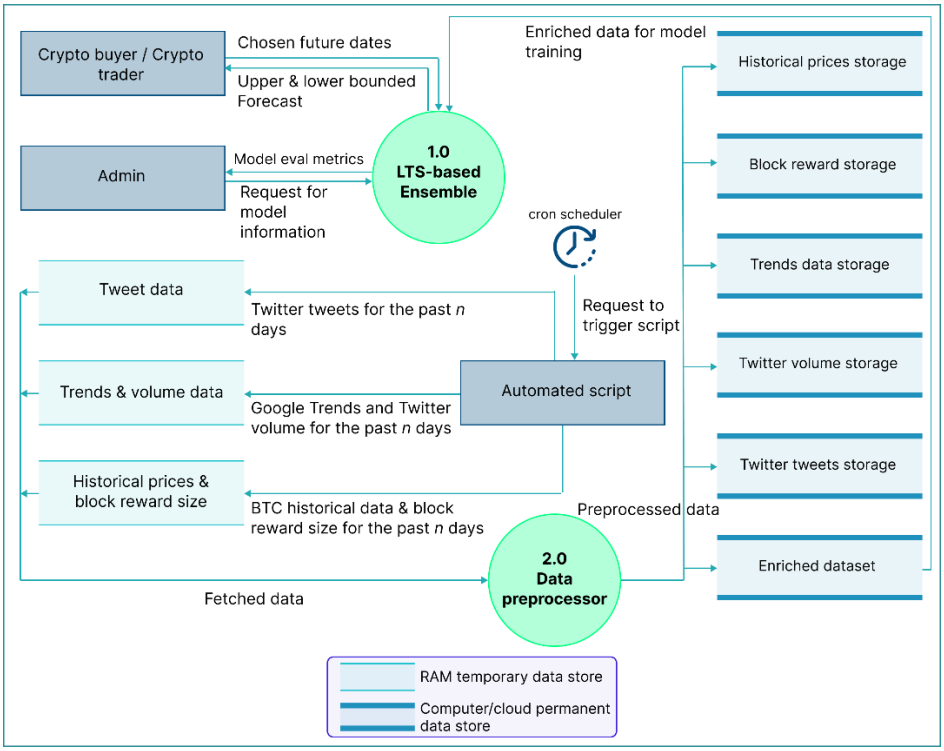


Figure 9: Data flow diagram - level 01 (*self-composed*)

6.4.2.2 Level 02 data flow diagram

The level 02 diagram provides a detailed illustration of the data preprocessor component, which was introduced in level 01 for preprocessing the raw data fetched from the automated script.

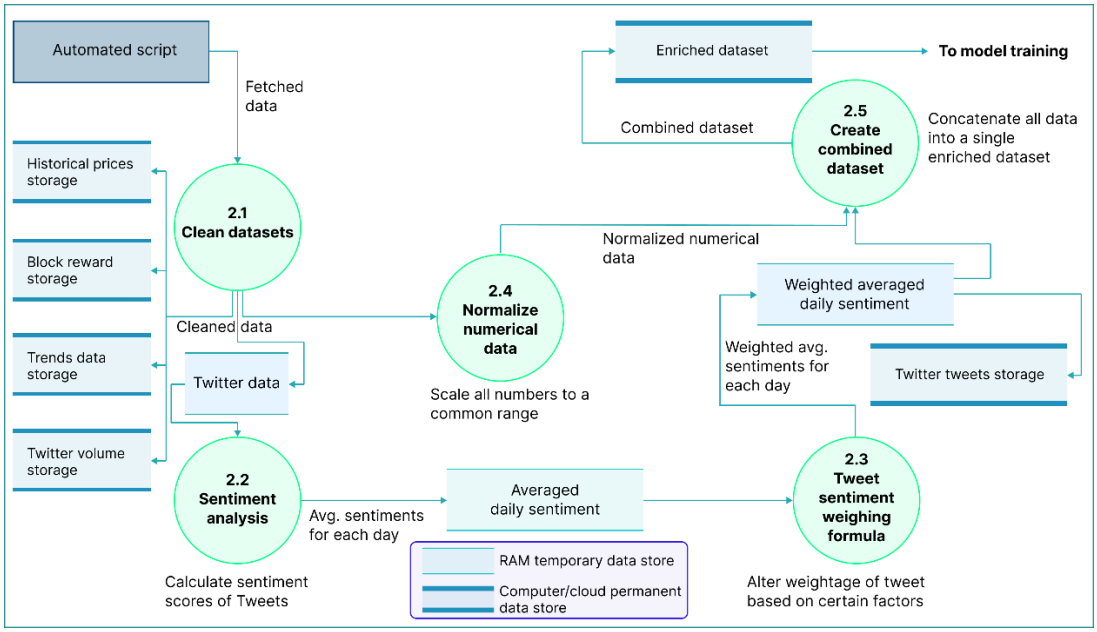


Figure 10: Data flow diagram - level 02 (*self-composed*)

6.4.3 Algorithm design

Research domain

Upon gathering requirements to implement the research component, the author realized they could further enhance the existing LTC architecture by integrating flexible latent SDEs instead of the current ODEs. The author will therefore attempt to design and evaluate a novel algorithmic implementation inspired by the original LTC proposed by Hasani et al. (2020), which can be considered their primary contribution to the body of knowledge (the LTS is the result of this modification). A simple illustration is available in **APPENDIX D.1** to gather intuition alongside an in-depth derivation and explanation of the proposed formula.

Problem domain

Upon analyzing requirements and reading literature on the supplementary forecasting application being implemented, the author noticed no straightforward technique to weigh tweet sentiments based on the tweeter's influence. Therefore, the author will propose another novel formula to weigh these sentiments - this can be considered a secondary contribution.

6.4.3.1 Liquid Time-stochasticity (LTS) algorithm

Upon studying the architecture proposed by Hasani et al. (2020), the author could utilize a linear system of SDEs to declare the flow to manifest a novel algorithm with more flexibility for the instantaneous adaptation of tiny changes. Moreover, this is an excellent enhancement as the additional component being developed belongs to the open market, which can have small instant price changes. The below formula is what the author proposes (Raneez and Wirasingha, 2023):

$$\boxed{\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta) - \sigma B\right] x(t) + f(x(t), I(t), t, \theta) A}$$

Where;

τ	*Time-constant*	f	*Neural network*
$x(t)$	*Hidden state*	θ, A	*Parameters*
$I(t)$	*Input*	B	*Noise (Brownian motion)*
t	*Time*	σ	*Intensity of noise (parameter)*

Algorithm forward propagation by implicit SDE solvers

Hasani et al. (2020) determined that their LTC architecture was ‘stiff equations’. They also found that regular Runge-Kutta was unsuitable for solving LTCs; therefore, they designed a hybrid ODE solver upon combining implicit and explicit Euler solvers (Raneez and Wirasingha, 2023). Combining these solvers is recommended as Press et al. (2007) specified, to achieve further stability; however, researchers can implement baseline stability by only using an implicit solver (Raneez and Wirasingha, 2023). That said, using solely an explicit solver is not recommended.

In the context of this system, SDE solvers must be used, as it uses SDEs instead of ODEs. Therefore, the author will use an SDE solver, which is implicit, and if time permits, create a hybrid SDE solver by integrating an explicit solver within.

According to the author's research, the Euler-Maruyama method is the suggested solution for SDE Euler methods since it can handle all types of noise (Li et al., 2020). Future studies should focus on combining the explicit and implicit Euler-Maruyama solvers to produce a hybrid solver to achieve further stability (Raneez and Wirasingha, 2023).

How to train the network?

Accuracy and memory are traded off while training these networks. The adjoint sensitivity approach was encouraged by Chen et al. (2019) as a way to do reverse-mode AD, which uses less memory. However, this method causes more numerical errors, as identified by Hasani et al. (2020), who instead employed the conventional BPTT approach, which is more accurate but uses more memory. Although there is an adjoints technique designed for SDEs, they cannot be used; Tzen and Raginsky (2019) determined that it cannot be used in this case, necessitating the creation of a custom backpropagation rule (Raneez and Wirasingha, 2023).

For this research, the author will opt for the approach by Hasani et al. (2020) to give more precision and as the author is time constrained to implement a custom backpropagation algorithm. Future research must focus on reverse-mode AD, which is the suggested strategy when memory effectiveness is more crucial. It is also important to note that adopting the BPTT technique has other advantages, such as the ability to be employed as an RNN layer alongside well-known and popular optimization algorithms such as Adam (Kingma and Ba, 2017) and stochastic gradient descent (Raneez and Wirasingha, 2023).

6.4.3.2 Tweet sentiment weighting algorithm

Based on the requirements received from end-users, it is evident that the impact of tweets depends on how influential the tweeter is. Performing this algorithm would change the weight of the sentiments of the tweets based on how influential the tweeter is and the tweet's engagement. The author considered the 'total followers' and 'total listed' metrics of the tweeter and the number of retweets and likes of the specific tweet; the other metrics, such as the number of 'friends', are not directly correlated and were not considered at this point. However, there is no limit to the number of factors that can be integrated in the future. The formula is presented below:

$$influencer_{sum} = \alpha \log_{10}(followers_{count} + 1) + \beta \log_{10}(lists_{count} + 1)$$

$$tweet_{sum} = \gamma \log_{10}(retweets_{count} + 1) + \delta \log_{10}(like_{count} + 1)$$

$$\boxed{weighted_{score} = \frac{tweet_{sum} + influencer_{sum}}{tweet_{sum} + influencer_{sum} + 1} * compound_{score}}$$

Where;

α	*Weight of number of followers* ***Set as 0.5***	γ	*Weight of number of retweets* ***Set as 0.1***
β	*Weight of number of lists* ***Set as 0.3***	Δ	*Weight of number of likes* ***Set as 0.1***

The derivation of this formula can be found in **APPENDIX D.2**.

6.4.4 LTS algorithm analysis

The notable difference between the proposed architecture and the traditional neural ODEs, as Chen et al. (2019) suggested, is the use of traditional BPTT instead of the recommended adjoint sensitivity. The analysis of the complexities of these approaches is demonstrated in **APPENDIX D.3**.

According to the table in the appendix, the traditional BPTT approach yields more accurate results but requires more memory. On the other hand, reverse-mode AD acts oppositely; it has lower accuracy but requires less time and memory. Therefore, the choice of approach depends on the specific trade-offs required by the task, where in this case, obtaining the best possible result is the priority and the reasoning for choosing BPTT over reverse-mode AD.

6.4.5 Deployment pipeline

As stated under the **Research objectives**, one of the objectives is to design a deployment pipeline for seamless deployments of the application. Its design is available in **APPENDIX D.4**.

6.4.6 UI design

The author chose to implement a web application for the forecasting application for convenience. The low-fidelity wireframes, designed to aid the development process, are available in **APPENDIX D.5**.

6.4.7 System process activity diagram

A summarized system flow activity diagram that end-users will follow is presented below.

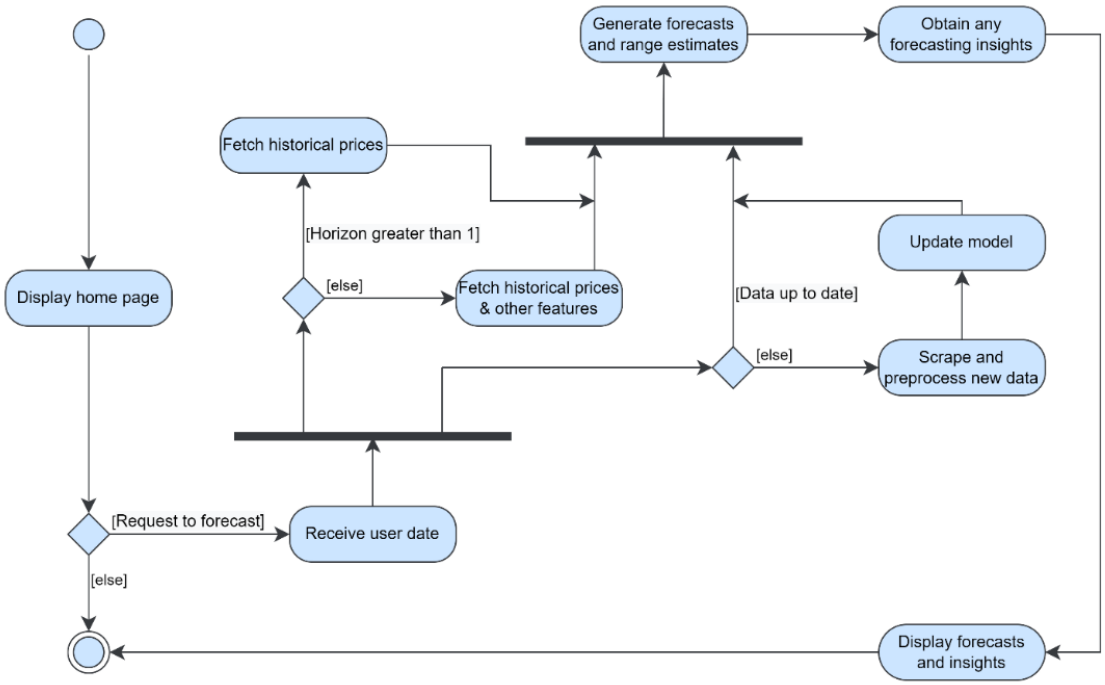


Figure 11: System process activity diagram (*self-composed*)

6.5 Chapter summary

This chapter presented the design of the LTS algorithm architecture, the necessary intuition behind it, and the reasons for taking specific directions over others. Moreover, a novel tweet sentiment weighting algorithm is also presented. Additionally, the chapter illustrated the system's design, architecture, data, system flow diagrams, and wireframes that would demonstrate them in the end application.

CHAPTER 07. IMPLEMENTATION

7.1 Chapter overview

Upon designing the necessary diagrams, the next step is to convert the idea into reality. In this chapter, the author describes the core implementation of the system and the decisions taken to approach that implementation. Moreover, the chosen tools, languages, and technologies are presented with their reasoning for being selected over others.

7.2 Technology selection

7.2.1 Technology stack

The chosen technologies and tools to implement the system are depicted in the diagram below.

Figure 12: Tech stack (*self-composed*)

7.2.2 Selection of data

Table 19: Dataset sources

Dataset	Source
BTC historical data	Extracted from: `http://api.scraperlink.com/investpy/`
BTC block reward size, BTC Twitter volume, BTC Google Trends	Scraped from: `https://bitinfocharts.com/`
BTC tweets	Tweets from 2014-2019 were downloaded from `Kaggle`. The remaining were extracted from a Twitter tweet scraper.

The multivariate single-horizon forecasting model combined the above-mentioned data sources, while the univariate multi-horizon forecasting model solely relied on historical data. Collecting the necessary data was challenging and time-consuming, as it involved more than just downloading available datasets, and API limitations were a hindrance. Dedicated Python scripts were developed to extract the data, store it in MongoDB, and automate data updates to tackle this issue. The author intends to share these scripts to facilitate future research.

7.2.3 Selection of programming languages

The author analyzed programming languages before development, focusing on three main aspects: the client, the data science component, and the API that communicates between the model and the client.

APPENDIX E.1 compares Python and R to determine the ideal language for model development. **Python** was chosen as the most relevant due to its large number of libraries, better community support, and, primarily, the author's familiarity that would streamline the process.

To develop the user interface, the author did not find many competing technologies to analyze. **JavaScript** is the dominant leader in this space and was ultimately chosen due to its dynamic nature. While recent technology has presented the usage of C#, this approach suffers from high latency issues and a lack of community knowledge, making it less suitable for the project.

APIs are required to communicate between the model and the user interface. Multiple technologies are available for API development. The author chose **Python** as their core data science component is also built using Python; therefore, utilizing the same language would reduce the time taken to learn new languages for insignificant reasons. However, technologies such as Nodejs and Golang are ideal substitutes.

7.2.4 Selection of development framework

7.2.4.1 Deep Learning (DL) framework

The author chose Python for developing the core data science component. As the core algorithm and model will be DL-based, DL frameworks must be meticulously analyzed to select the most relevant framework. The two most popular frameworks, TensorFlow and PyTorch, were analyzed in **APPENDIX E.2**. The author opted to use **TensorFlow**, which is more relevant for building a new algorithm, as it provides low-level details with more control and a larger community support.

7.2.4.2 User Interface (UI) framework

As JavaScript was chosen for developing the UI, respective JavaScript frontend frameworks and libraries must be analyzed. There is an ocean of JavaScript libraries - the top four were selected for evaluation: Angular, Vue, Svelte, and React. The author decided to use **React**; the evaluation can be found in **APPENDIX E.3**, as there is no requirement for large-scale app development.

7.2.4.3 Application Programming Interface (API) web framework

As Python was chosen for the API development, respective Python web frameworks must be analyzed to select the more relevant one. Analysis was conducted between Django and Flask, as they are the two most popular frameworks. **APPENDIX E.4** demonstrates the comparison where the author selected **Flask**, as an API is needed only for exposing the models' endpoints.

7.2.5 Other libraries & tools

Other libraries and tools that would facilitate implementation are stated in the table below.

Table 20: Chosen libraries & tools

Library	**Justification**
NumPy	Facilitates mathematical functions and calculations that are immensely required when building the algorithm.
Pandas	To create data frames for analysis, cleaning, transformation, and filtration.
Scikit-learn	To create data splits and feature scaling.
Lingua	To detect the language of the tweets. This project is limited to using only English tweets, so they must first be identified.
Matplotlib + Seaborn	For analysis, visualizations, and dashboarding.

Beautiful Soup	For scraping the block reward size and the Twitter volume from the public dashboard.
VADER	Perform sentiment analysis on the tweets.
Redux	For API requests from the client.
Ant design	Makes creating appealing user interfaces hassle-free.
MongoDB	To store the datasets and retrieve/update them whenever necessary.
AWS S3	Store the models in use.
Docker + Heroku	To deploy the model for client-side communication upon deployment.

7.2.6 Integrated Development Environment (IDE)

IDEs are required to streamline implementation; the chosen IDEs are stated in the table below.

Table 21: Chosen IDEs

IDE	**Justification**
Kaggle	Chosen for its large memory capacity (32GB) allowing easy loading of large datasets. Additionally, it provides easy integration with existing Kaggle datasets and user-uploaded datasets.
Jupyter	Chosen for trials, testing, and model training as it would not be interrupted.
VSCode	Chosen for its lightweight and powerful features, such as shortcuts, extensions, and snippets, that can significantly boost development productivity.

7.2.7 Summary of chosen tools & technologies

The table below summarizes the tools and technologies the author chose to aid in implementation.

Table 22: Summary of chosen tools & technologies

Component	**Tools**
Programming languages	Python, JavaScript
Development framework	Flask, TensorFlow
UI development framework	React
Libraries	Ant design, NumPy, Pandas, Scikit-learn, Beautiful Soup, Lingua, Matplotlib, Seaborn, VADER, Redux, Ant design

IDEs; Version control	Kaggle, Jupyter notebooks, VSCode; Git + GitHub
Other tools	Heroku, AWS, Docker, MongoDB

7.3 Implementation of core functionalities

The project's core functionalities are the novel algorithm, the scripts to fetch the required data, and the preprocessing performed.

7.3.1 Algorithm implementation

The LTC architecture is available at the following repository: **`https://github.com/raminmh/liquid_time_constant_networks`**. However, it was implemented using TensorFlow V1, which is now outdated, especially after the integration of the Keras library. Thus, the author initially developed a modern LTC implementation. Subsequently, the author replicated this architecture and replaced the ODEs with SDEs.

Although some variable names have been adopted from traditional neural networks and Lapicque's (1907) work on *C. Elegans worms*, the code presented in this section is original; to the best of the author's knowledge, it has not been previously published.

```
class LTSCell(tf.keras.layers.Layer):

  def __init__(self, units, **kwargs):
    '''
    Initializes the LTS cell & parameters
    Calls parent Layer constructor to initialize required fields
    '''

    super(LTSCell, self).__init__(**kwargs)
    self.input_size = -1
    self.units = units
    self.built = False

    self._time_step = 1.0
    self._brownian_motion = None

    # Number of SDE solver steps in one RNN step
    self._sde_solver_unfolds = 6
    self._solver = SDESolver.EulerMaruyama
    self._noise_type = NoiseType.diagonal

    self._input_mapping = MappingType.Affine

    self._erev_init_factor = 1

    self._w_init_max = 1.0
    self._w_init_min = 0.01
    self._cm_init_min = 0.5
    self._cm_init_max = 0.5
    self._gleak_init_min = 1
    self._gleak_init_max = 1

    self._w_min_value = 0.00001
    self._w_max_value = 1000
    self._gleak_min_value = 0.00001
    self._gleak_max_value = 1000
    self._cm_t_min_value = 0.000001
    self._cm_t_max_value = 1000

    self._fix_cm = None
    self._fix_gleak = None
    self._fix_vleak = None

    self._input_weights = None
    self._input_biases = None
```

Figure 13: Initialize algorithm (Lapicque, 1907; Hasani et al., 2020)

The above code initializes the algorithm cell with maximum and minimum values necessary for its operation. By inhering the base Keras Layer class, the model can perform input-independent initializations and can be used as a regular LSTM or RNN cell in higher-level layer definitions

```
def build(self, input_shape):
    '''
    Automatically triggered the first time __call__ is run
    '''

    self.input_size = int(input_shape[-1])
    self._get_variables()
    self.built = True
```

Figure 14: Build algorithm (*self-composed*)

The above snippet defines what occurs upon initialization; in other words, it 'builds' the algorithm cell. A helper function is utilized here that defines the variables (sigma, mu, weights, and leakage conductance variables (Lapicque, 1907; Hasani et al., 2020)). The input shape is available within the above function; therefore, the model can initialize the variables used here. The below snippet demonstrates how some of these variables are initialized.

```
# Define sensory variables
self.sensory_mu = tf.Variable(
  tf.random.uniform(
    [self.input_size, self.units],
    minval = 0.3,
    maxval = 0.8,
    dtype = tf.float32
  ),
  name = 'sensory_mu',
  trainable = True,
)
```

```
# Define base stochastic differential equation variables
self.mu = tf.Variable(
  tf.random.uniform(
    [self.units, self.units],
    minval = 0.3,
    maxval = 0.8,
    dtype = tf.float32
  ),
  name = 'mu',
  trainable = True,
)
```

```
# Synaptic leakage conductance variables of the neural dynamics of small species
if self._fix_vleak is None:
  self.vleak = tf.Variable(
    tf.random.uniform(
      [self.units],
      minval = -0.2,
      maxval = 0.2,
      dtype = tf.float32
    ),
    name = 'vleak',
    trainable = True,
  )
else:
  self.vleak = tf.Variable(
    tf.constant(self._fix_vleak, dtype = tf.float32),
    name = 'vleak',
    trainable = False,
    shape = [self.units]
  )
```

Figure 15: Algorithm – sensory, stochastic and leakage variables (*self-composed*)

The final step is the forward computation process that will occur on each epoch, in other words, the forward propagation process.

```
@tf.function
def call(self, inputs, states):
  '''
  Automatically calls build() the first time.
  Runs the LTS cell for one step using the previous RNN cell output & state
  by calculating the SDE solver to generate the next output and state
  '''

  inputs = self._map_inputs(inputs)
  next_state = self._sde_solver_euler_maruyama(inputs, states)
  output = next_state
  return output, next_state
```

Figure 16: Algorithm – forward propagation (*self-composed*)

The above function is run automatically on each epoch. Initially, a helper function defines the weights and biases of the network, as demonstrated below.

```
def _map_weights_and_biases(self, inputs):
  '''
  Initializes weights & biases to be used
  '''

  # Create a workaround from creating tf Variables every function call
  # init with None and set only if not None - aka only first time
  if self._input_weights is None:
    self._input_weights = tf.Variable(
      lambda: tf.ones(
        [self.input_size],
        dtype = tf.float32
      ),
      name = 'input_weights',
      trainable = True
    )

  if self._input_biases is None:
    self._input_biases = tf.Variable(
      lambda: tf.zeros(
        [self.input_size],
        dtype = tf.float32
      ),
      name = 'input_biases',
      trainable = True
    )

  inputs = inputs * self._input_weights
  inputs = inputs + self._input_biases

  return inputs
```

Figure 17: Algorithm – define weights and biases (*self-composed*)

As discussed in **Chapter 6**, the forward computation of SDEs can be performed using the Euler-Maruyama method. The author's implementation of the Euler-Maruyama SDE solver is shown in the code snippet below. In this implementation, Brownian motion is used as the noise term, as suggested by Duvenaud (2021).

```python
@tf.function
def _sde_solver_euler_maruyama(self, inputs, states):
    '''
    Implement Euler Maruyama implicit SDE solver
    '''

    for _ in range(self._sde_solver_unfolds):
        # Compute drift and diffusion terms
        drift = self._sde_solver_drift(inputs, states)
        diffusion = self._sde_solver_diffusion(inputs, states)

        # Compute the next state
        states = states + drift * self._time_step + diffusion * self._brownian_motion
        states = tf.reshape(states, shape=[int(self._time_step), self.units])
```

Figure 18: Algorithm – Euler-Maruyama SDE solver (*self-composed*)

7.3.2 Data fetchers

Data fetchers are required to obtain the required data. As the author discussed before, they are not directly available. The scripts used to extract the required data were written exclusively by the author with no adaptation. The scripts are placed under **APPENDIX E.5**.

7.3.3 Preprocessing

Data fetched from the data fetchers must undergo preprocessing (cleaning, scaling) before being used by the model. The preprocessing scripts were exclusively written by the author with no adaptation and are placed under **APPENDIX E.6**. The data fetchers and preprocessing scripts will be well-documented and publicly available to facilitate future research. This is particularly important, as these steps proved to be lengthier and more arduous than anticipated.

7.3.4 The forecasting models

As previously stated, the LTS can be used as existing Keras layers. The BTC forecasting model was manually constructed as an ensemble to generate a range of prices. The code used to construct the ensemble is available in **APPENDIX E.7**.

7.4 User interface

Screenshots of the final GUI are placed under **APPENDIX E.8**.

7.5 Chapter summary

This chapter focused on defining the technologies and tools that would facilitate development to demonstrate the research. Additionally, the implementation of the core features is shown with accompanying code snippets.

CHAPTER 08. TESTING

8.1 Chapter overview

Once implementation is satisfactory, testing is essential to ensure that the system's functionalities act as expected. This chapter performs detailed testing on the system and the model in use. Testing methodologies utilized include functional, non-functional, integration, and model testing to evaluate the system as much as possible.

8.2 Testing objectives & goals

The ultimate goal in conducting testing is to ensure that the system performs as expected. To meet this goal successfully, a couple of testing objectives must be met:

- Ensure that the models are as performant as they can be.
- Ensure that the functionalities implemented align with the MoSCoW's 'Must have' and 'Should have' techniques.
- Identify any bug fixes/improvements that must/can be applied to the application.
- Identify if the essential non-functional requirements are met.
- Conduct baseline benchmarking so that the system can be used as a benchmark in the future.

8.3 Testing criteria

Prior to conducting testing, a criterion was defined to test the system in two methods.

- Functional testing – focused on determining how well the system performs and meets the functional requirements.
- Non-functional testing – focused on evaluating the achievements of the non-functional requirements and design goals.

8.3.1 Self-reflection on the testing criteria

To further enhance the algorithm's robustness, it would be beneficial to establish a specific testing criterion for evaluating the LTS. There are currently no standardized procedures for testing the algorithm, so defining such criteria would be a crucial next step in development and deployment.

8.4 Model testing & evaluation

8.4.1 Model testing

Two ensemble models were implemented: univariate and multivariate. Both models were tested similarly by setting a 'pseudo-future' from a specific timeframe.

A pseudo-future is a 'fake' future where we know the actual value at that point.

The below graphs illustrate the closing price change with the actual and predicted prices.

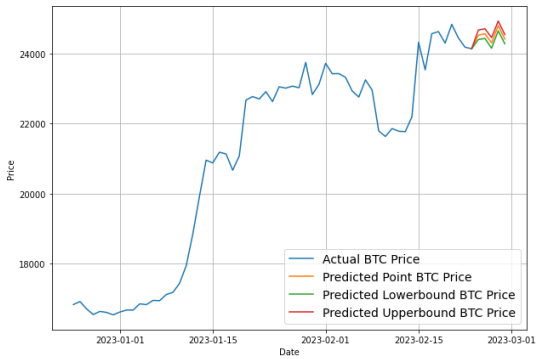


Figure 19: Univariate model testing (*self-composed*)

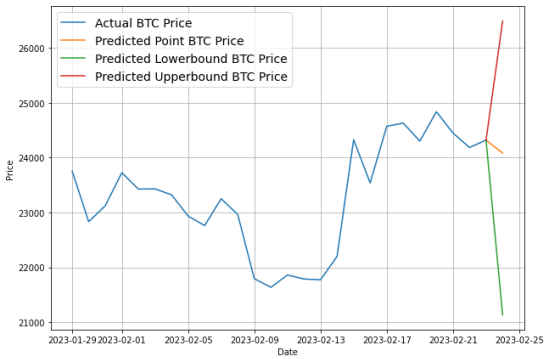


Figure 20: Multivariate model testing (*self-composed*)

Thirty different models were trained and combined to form the ensemble, resulting in 3 predicted prices. The median prediction is used as the predicted point BTC price, while the highest and lowest predictions are used to create the upper and lower bounds, respectively.

Before implementing the final ensemble, a single model was trained to determine whether the architecture was optimal. The loss curve recorded in TensorBoard during the training of a single model showed signs of overfitting – a limitation that requires attention in the future. However, this is expected due to the limited number of data samples.

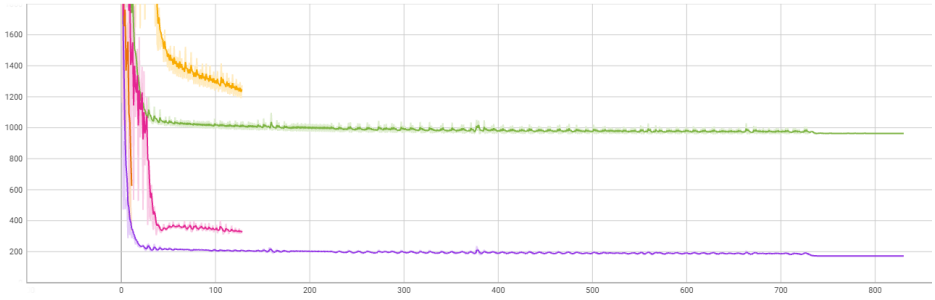


Figure 21: TensorBoard loss curve (*self-composed*)

8.4.2 Model evaluation

The models were evaluated by calculating the evaluation metrics that were recommended having read through the literature and the author's experience in evaluating ML models; these metrics were presented under the **Evaluation** section in **Chapter 2**.

Standard prediction and test set evaluation techniques were applied to both univariate and multivariate ensembles to evaluate the models' performance. The results were compared with the naïve forecast, which was chosen as a benchmark due to its reputation of being notoriously difficult to beat in open market systems (Gilliland, 2014). In addition, other architectures were also evaluated during the univariate phase to ensure that the ensemble architecture was the best performing.

Table 23: Univariate model evaluation

	MAE	MSE	RMSE	MAPE	MASE
Traditional architectures					
Basic dense	1227	2882849	1697	3.06%	1.07
2x dense	1146	2628006	1621	2.98%	1.05
Stacked dense	1147	2604766	1613	2.88%	1.04
Conv1D	1153	2653370	1628	2.90%	1.02
LSTM	1216	2834756	1683	3.07%	1.06
N-BEATS	1142	2614896	1617	2.86%	1.03
Benchmark & LTS Ensemble					
Naïve forecast	951	2021966	1421	2.56%	1.00
Ensemble	950	2013928	1419	2.56%	0.99

As the performance of traditional architectures was not satisfactory, evaluating them for the multivariate model was deemed unnecessary.

Table 24: Multivariate model evaluation

	MAE	MSE	RMSE	MAPE	MASE
Naïve forecast	858	1631648	1277	2.41%	1.00
Ensemble	932	1826788	1351	2.67%	1.00

8.4.3 Self-reflection on the model evaluation

Based on the above evaluation, the multivariate model performs better than the univariate model, attributed to its ability to incorporate multiple exogenous features that provide more information and context for accurate predictions. This makes the model more robust and less reliant on a single input variable, resulting in improved performance. It can also be observed that the naïve forecast performs better in the multivariate model, whereas the opposite is true for the univariate model. This difference in performance could be attributed to the increased complexity of the ensemble architecture used in the multivariate model, as it can add more noise, especially since a more diverse dataset is used, which must be addressed as enhancements in the future.

8.5 Benchmarking

Although extensive benchmarking was not possible due to the algorithm's novelty and the absence of an existing system that utilizes all the features in the application, a baseline was established by benchmarking the naïve model, which can be used as a point of reference for future evaluations.

8.6 Functional testing

Functional testing was performed by evaluating whether the system aligns with the functional requirements specified in **Chapter 4**. **APPENDIX F.1** shows the breakdown of the functional testing, where an **80%** success rate was achieved as FR12 and FR13 were not implemented.

8.7 Module & integration testing

As demonstrated in the high-level architecture diagram presented in **Chapter 6**, the system's logic was modularized. Each module was tested to ensure that it performed as expected.

Table 25: Module & integration testing

Module	**Input**	**Expected result**	**Actual result**	**Status**
Data fetcher	Triggered by a scheduler.	Fetch & update datasets.	Datasets are scraped and stored in the database.	Passed
NLP data parser	Latest scraped data.	Perform sentiment analysis on tweets.	Sentiment analysis is performed, and the scores	Passed

			are weighted and saved in the database.	
Data preprocessor	Latest obtained & parsed data.	Combine all datasets, clean, process, and remove unneeded columns.	Datasets are cleaned and combined into a single dataset and then stored in the database.	Passed
Model information	Type of model.	Return the current evaluation metrics of the specific model.	Returns the current evaluation metrics of the specified model.	Passed

8.8 Non-functional testing

Non-functional testing assessed how well the system aligns with the non-functional requirements outlined in **Chapter 4** and the design goals stated in **Chapter 5**. The breakdown of the exhaustive non-functional testing conducted by the author on all aspects is presented in **APPENDIX F.2**.

8.9 Limitations of the testing process

While the system was tested thoroughly, it was impossible to effectively test the mathematical expressions calculated behind the scenes of the algorithm. This was due to the complexities of the LTS algorithm and the limitations of available testing methods. However, extensive unit testing can be performed in the future to determine its robustness, which the author has not done at this moment in time. Despite this, the author conducted simple debugging, such as printing calculated values on the terminal and comparing them against a calculator. It is worth noting that the research community accepted the formula proposed by the author to rectify the identified limitations.

8.10 Chapter summary

This chapter initially presented the testing objectives and the criterion on which it was performed. Testing was carried out on the core research component by evaluating the used models. The system was evaluated by functional, non-functional, and integration testing, and finally, the limitations of this procedure were specified. Lastly, as future enhancements, it is best to conduct exhaustive unit testing on the LTS layer prior to pushing it into the Keras library to ensure it is industry ready.

CHAPTER 09. EVALUATION

9.1 Chapter overview

Upon implementation, comprehensive evaluation is required to ensure that the system has met the requirements gathered in ***Chapter 4***. This chapter discusses the self-evaluation conducted by the author and feedback obtained from domain and technical experts.

9.2 Evaluation methodology & approach

As the primary focus of this research is the design and implementation of the LTS algorithm, qualitative and quantitative evaluations are necessary to ensure its robustness and performance. Therefore, qualitative analysis was conducted by obtaining feedback from domain and technical experts, while quantitative analysis was performed to assess the numerical output model's performance, as presented in **Chapter 8**. This chapter will present a thematic analysis of the feedback obtained from the qualitative analysis.

9.3 Evaluation criteria

Before evaluation, clear criteria must be defined to ensure that all aspects of the research are assessed. The table below breaks down the criteria the author defined prior to conducting the evaluation.

Table 26: Evaluation criteria

CR ID	Criterion	Purpose
CR1	Choice of research domain	To validate the significance of the chosen domain, topic, and research gap.
CR2	Research contribution	To determine the significance of the observed findings and contributions to the body of knowledge.
CR3	Research novelty	To assess the novelty of the proposed solution.
CR4	Research difficulty	To assess the technical difficulty of the research.

CR5	Research quality	To confirm that adequate literature has been reviewed and the research is well documented with adequate reasoning behind each technique.
CR6	Development approach	To ensure that the approach taken meets industry standards and is best in class.
CR7	Usability	To confirm that the system is user-friendly and convenient.
CR8	Limitations & improvements	To identify any limitations and future research possibilities.

9.4 Self-evaluation

The table below presents the author's self-evaluation according to the abovementioned criteria.

Table 27: Self-evaluation of the author

CR ID	Author's self-evaluation
CR1	The selected domain is relatively new, and only a few experts possess the required technical knowledge, resulting in limited available research. Nevertheless, the domain offers tremendous potential as a foundation for more advanced DL algorithms, given its adaptability in evaluation strategies and non-static nature.
CR2	Given the short duration, the author considers the contributions made as impactful. Contributions range from a novel algorithm within the research domain to a novel algorithm within the problem domain. Furthermore, the research was left open for further enhancements and work to motivate upcoming/seasoned researchers to, at the very least, be curious about the domain of neural ODEs/SDEs & LTCs.
CR3	Prior to the completion of this research, TS algorithms all utilized traditional neural networks (ex: LSTMs, RNNs, GRUs) and had been stagnant owing to the restrictions mentioned in previous chapters. Therefore, the ultimate goal of the LTS designed by the author was to break these limitations to establish a new benchmark in the research community and set the foundation for more advanced TS algorithms.

CR4	Compared to other DL domains with abundant research and literature, the chosen domain has limited resources, with fewer than ten comprehensive papers available for detailed understanding. This lack of research posed a significant challenge, as there was no clear roadmap to follow.
CR5	The author compiled a comprehensive review of existing literature on neural ODEs/SDEs & LTCs, including research papers and recorded video conferences, within this document to ensure that the quality of the research is of the highest standard possible.
CR6	The development process was carried out in a systematic and structured manner, with the algorithm implementation being completed before the system implementation. • The LTS algorithm was designed and built in accordance with recommended practices by TensorFlow, ensuring that it works and behaves similarly to other deep learning algorithms. • The system was developed in line with industry standards and techniques to ensure maximum robustness and alignment with recommended best practices. To incorporate cutting-edge tools and technologies in high demand among companies and employers, they were carefully selected and used in the implementation process. Despite being the work of a single author, the system was developed with the mindset that multiple contributors would be involved. As a result, the GitHub repositories are accompanied by clear documentation, commit messages, and workflows to facilitate collaboration and enhance the system's overall quality.
CR7	The system was developed with a user-centered approach, prioritizing the user experience in every aspect of the process. Additionally, technical jargon is avoided.
CR8	Upon completing the system, the author identified several areas for potential improvement and future work – they are discussed in **Chapter 10**.

9.5 Selection of evaluators

The evaluators were selected based on grouping to ensure that feedback was obtained for all aspects of the project. The table below shows the grouping breakdown.

Table 28: Categorization of selected evaluators

CAT ID	Category
CAT1	Experts in neural ODEs & SDEs.
CAT2	Experts in traditional ML/DL.
CAT3	End users of the application and cryptocurrency experts.

9.6 Evaluation results & expert opinions

As determined previously, the LTS algorithm is the main contribution of this research; therefore, to have a distinct separation in evaluation, two independent components were defined:

- Evaluation of the LTS and its associated architecture by CAT1 and CAT2.
  - **https://youtu.be/dIbBM988zzM**
- Evaluation of the Twitter sentiment weighing algorithm and the application by CAT3.
  - **https://youtu.be/tAfbJ0Iacx8**

Thematic analysis was conducted to analyze the opinions and feedback received. The table below discusses the findings that emerged.

Table 29: Thematic analysis of expert feedback

CR ID	CAT ID	Theme	Conclusion
CR1	CAT1	Gap in neural ODEs/SDEs.	A significant and well-informed gap has been identified that required fulfilment.
	CAT2	Gap in TS forecasting.	The chosen domain is excellent, a clear gap has been identified, and a new technique has been introduced to solve it.
CR2	CAT1	Surpass TS forecasting limitations.	The contribution is important as the LTS is a more robust and stable liquid neural network.
	CAT2		A novel and significant contribution. A new technique/algorithm was developed to overcome issues in TS forecasting.

CR3	CAT1	A technique to solve highly volatile data.	Well-justified novelty: there have yet to be solutions for modelling highly volatile TS data.
	CAT2	A new approach to solve TS forecasting issues.	A couple of conclusions were made: **01**: All TS forecasting systems have been using traditional neural network architectures. **02**: There is little documentation available.
CR4	CAT1	Research difficulty.	The difficulty is respectable as work had to be done from researching to training the network with the most appropriate technique.
	CAT2	Algorithm development difficulty.	Technical issues were inevitable but had been overcome.
CR5	CAT1	Training & forward propagation justification.	Adequate research was conducted. The choice of forward and backward propagation was well justified and explained.
CR6	CAT2	Keras library extension.	An interesting approach had been taken as a new layer had been built, which can be added to the Keras ML library.
	CAT3	Well-built and convenient.	A few conclusions were drawn: **01**: The application is well-built and understandable. **02**: The Twitter sentiment weighing algorithm is significantly accurate. **03**: The range of prices produced is more realistic and convincing and gives the required insights.
CR7	CAT3	User-friendly & straightforward.	The application looks user-friendly and hides technical information making it ideal for amateurs.
CR8	CAT1	Analyze other SDE solvers.	The choice of BPTT was justified; however, there is a concern about scalability. Additionally, to further

			justify the selection of an SDE solver, compare it with others.
	CAT2	Further evaluation.	A good research approach was followed, and evaluation was conducted to compare with the current TS forecasting algorithms. However, further evaluation should be performed.
	CAT3	Consider more factors.	Although a few extraneous had been considered, further factors can be included to make it significantly more accurate.

9.7 Limitations of evaluation

Obtaining expert opinions was challenging due to the scarcity of experts in the research domain. During the requirement-gathering phase, the author contacted numerous ML/DL domain experts to gather more insights; however, only a few could provide such information, highlighting the difficulty of finding experts in niche domains. Therefore, only the experts specified in **APPENDIX G.1** were selected to provide constructive feedback.

9.8 Evaluation of functional requirements

The completion breakdown of the functional requirements is presented in **APPENDIX G.2**.

9.9 Evaluation of non-functional requirements

The completion breakdown of the non-functional requirements is presented in **APPENDIX G.3**.

9.10 Chapter summary

This chapter provided a comprehensive evaluation of the implemented system. The criteria were established beforehand to ensure that all aspects of the system were thoroughly evaluated. The author performed self-evaluation and received feedback from evaluators with the necessary expertise, which was then analyzed using thematic analysis. Finally, the evaluation of functional and non-functional requirements and the achievement of the proposed design goals were presented.

CHAPTER 10. CONCLUSION

10.1 Chapter overview

This chapter concludes the author's research journey by marking its final remarks. This concluding chapter summarizes the author's research journey, highlighting their contributions to the field, the achievement of stated objectives, deviations from the proposed scope, challenges faced, and possibilities and limitations for future research.

10.2 Achievement of research aim & objectives

10.2.1 Achievement of the research aim

The aim of this research is to design, develop & evaluate the author-proposed LTS algorithm for TS forecasting, which could be the stepping stone for breaking TS forecasting limitations.

The aim was successfully attained by designing and developing the proposed LTS architecture so that it could be used as other existing Keras layers. This algorithm was then utilized in a BTC forecasting application - through an ensemble architecture to obtain upper and lower bounds - to evaluate its performance and to determine whether it broke limitations in TS forecasting algorithms.

10.2.2 Achievement of objectives

All objectives stated in **Chapter 01** were met. The status of each objective has been marked with the objective itself in **APPENDIX H.1**.

10.3 Utilization of knowledge from the degree

The proposed research required tremendous knowledge, and the author was fortunate to have gained a solid foundation through various modules completed during the degree. Among them, a few were particularly beneficial (see **APPENDIX H.2** for their justification):

- Mathematics for Computing
- Object Oriented Programming
- Software Development Group Project
- Algorithms: Theory Design and Implementation

10.4 Use of existing skills

- **Full-stack development** – the author has a few years of experience in full-stack development, having done their internship at 99x working on web and mobile applications. Additionally, the author has worked for over two years at Niftron as a full-stack developer.
- **ML/DL** – the author has done a few freelancing projects that included ML & DL. Extra knowledge was gained by following courses on Coursera and YouTube.
- **Calculus** – the author had foundational calculus knowledge, having completed their A levels in mathematics.

10.5 Use of new skills

- **ML deployment** – creating an ML notebook can be considered only 50% of the work in the deployment pipeline. The author had to learn a few techniques to create APIs to serve the created models, and host the model in the cloud to avoid needing to run it locally.
- **Data scraping & mining** – the datasets were not readily available and had to be fetched, scraped, cleaned, and condensed. The author had to learn and get a lot of practice on these techniques to ensure seamless updating of the used datasets.
- **Advanced calculus** – the author had to get familiar with college-level calculus to implement the LTS as extensive knowledge on differential equations was vital. MIT OpenCourseWare was used heavily to understand and learn these techniques.
- **Neural network building blocks** – the author had to gain comprehensive knowledge and insights into the underlying theoretical and mathematical concepts within neural networks to comprehend and make specific decisions, especially due to the lack of literature.

10.6 Achievement of learning outcomes

The achievement of the learning outcomes is presented in **APPENDIX H.3**.

10.7 Problems and challenges faced

Challenges are inevitable in any research. The following table describes the challenges faced during this study and the strategies used to address them.

Table 30: Problems and challenges faced

Problem/Challenge	**Mitigation**
Research domain	
There are hardly any experts in neural ODEs, SDEs, and LTCs.	Although there are few experts in this domain and only a few evaluators, the interviewees were the best and provided valuable insights.
Steep learning curve in mathematics.	The final formula was designed by reading through centuries-old documentation, which made it challenging to understand the initial mathematical and scientific concepts. However, the author utilized available resources and sought help from experts to overcome this challenge.
Unclear direction due to the lack of SDE documentation.	The author read through available literature and made a considerable effort to understand it as clearly as possible. Additionally, the author published a paper with the proposed formula as soon as possible to verify its correctness and proceed with implementation.
Problem domain	
Twitter API rate limitations provided access to tweets for only the past seven days.	The author developed a dedicated scraper to fetch the data. Only 500 tweets were collected each day to avoid rate limitations and as this was not the primary focus of the research.
The datasets did not satisfy the author's requirements.	The author developed scrapers to scrape a public dashboard that provided the required data free of charge.

10.8 Deviations

The author partially automated the deployment pipeline, automating the scraping of datasets. However, automating the model deployment process proved challenging due to the need for dedicated cloud servers. As free-tier or hobby servers tend to timeout, the author could only trigger model deployment locally.

The project proposal outlined a set of ‘in-scope’ features (see **APPENDIX A.2**), and the author adhered to this plan without any major deviations. However, none of the ‘desirables’ were implemented due to time constraints; nevertheless, they do not affect the core functionality.

10.9 Limitations of the research

The limited time frame of the project resulted in certain compromises, which ultimately led to some limitations that require attention.

- The application only generates predictions for specific days instead of providing real-time forecasts, which could have been more beneficial for traders.
- Scraping only 500 tweets daily to obtain sentiment values is a considerable limitation. Considering more tweets could have significantly improved the model's robustness as it would be a more realistic averaged sentiment.
- The application only considers tweets with the tag ‘bitcoin’, which limits the accuracy of the sentiment analysis. Twitter consists of a substantial number of handles that speak about BTC. Considering them would significantly improve the accuracy of the overall sentiment leading to a more robust forecasting model.
- The application plots the daily forecast alongside hourly data making the graph seem unrealistic or even misleading as it loses its natural progression over time.
- The need for manual training and deployment of the model highlights the incomplete CI/CD pipeline, a moderate limitation that makes deployment tedious. While migrating to a dedicated cloud server can overcome this limitation, it may not be cost-effective.

Mitigating these limitations could enhance the model's performance and make it more applicable to real-world scenarios. Nevertheless, it is crucial to acknowledge that this model's predictions should not be taken as financial advice, as market forecasting is inherently unpredictable and influenced by a multitude of external factors. Despite the sophistication and advancements of deep learning architectures, they are still limited to certain domains.

10.10 Future enhancements

This thesis has room for future improvements in both of its components.

10.10.1 Research domain enhancements

- The LTS uses the Euler-Maruyama SDE solver, which handles all noise. Nevertheless, other SDE solvers (ex: Heun's method) must be evaluated to determine if there would be a better-performing option; however, the used solvers must be stiff, as the LTS is a stiff equation.
- The LTS can use a hybrid SDE solver that combines the implicit and explicit Euler-Maruyama solvers rather than the utilized implicit Euler-Maruyama (Press et al., 2007) to achieve further stability.
- LTS with reverse-mode AD must be evaluated instead of the proposed BPTT approach to determine memory and time efficiency. However, researchers cannot use the same reverse-mode AD for SDEs (Tzen and Raginsky, 2019): there is no straightforward way. Fortunately, Duvenaud's (2021) approach could work as expected.
- Benchmark the LTS in the M4 competition to determine whether it is a SOTA TS forecasting algorithm. Although the architecture's performance was better compared to other networks that the author evaluated, it is not officially a SOTA algorithm; applying this in the M4 competition could set its mark as the '*next big thing'* in TS forecasting and be the benchmark to beat in the future.
- XAI in neural ODEs/SDEs still needs to be researched. However, it is vital to note that it can be tricky because of the temporal component.
- Research and attempt to overcome gradient explosion in neural ODEs/SDEs. As stated in **Chapter 7**, it is a common weakness. The author overcame this by adding an LSTM layer, as demonstrated in **Chapter 7**; however, there must be a better approach.

Implementing these enhancements could yield significant findings in the field of neural ODEs/SDEs. Being recognized as a SOTA algorithm in TS forecasting could garner the attention of researchers, leading to further investigation and advancement.

10.10.2 Problem domain enhancements

- Enhance the Twitter sentiment weighting formula and the application by incorporating additional factors (ex: Facebook, Reddit). Researchers can consider various factors to improve the formula and application further. However, it is advisable to exclude factors that typically have little or no impact, such as the number of people a person follows.

- Conduct extensive testing on the weighting formula to determine whether the score calculated aligns with sentiment analysis guidelines. The current testing process may only partially validate the final result. While the weighting process is generally accurate, it may still lead to imprecise results if not adequately tested and validated.
- The third-party tweet scraper library can be unpredictable due to frequent updates in the Twitter API. Therefore, developing a dedicated library designed explicitly for scraping tweets is recommended to avoid any rate limitations or unforeseen errors.

10.11 Achievement of the contribution to the body of knowledge

Upon completion, the author managed to achieve contributions to the research domain of DL and TS forecasting, the problem domain of BTC, and the general software engineering domain.

10.11.1 Research domain contribution

- SDE-based liquid neural network – a novel algorithm that draws inspiration from the LTC that exhibits adaptability to instantaneous changes, surpassing TS forecasting limitations.

10.11.2 Problem domain contributions

- A formula to weight Twitter sentiments based on influencer metrics.
- A robust implementation of a BTC forecasting model that considers multiple exogenous features that affect the historical price, outperforming the naïve forecast in open markets.

10.11.3 Technical/additional contributions

- Data extraction and preprocessing scripts for various sources, including Twitter tweets, Twitter volume, block reward size, Google Trends, and BTC historical prices.
- Custom Keras layers of the LTC and LTS that can be integrated within the Keras library.

10.12 Outcome of the proposed research questions

The table summarizing the key findings of the research questions proposed in **Chapter 1** is in **APPENDIX H.4**, which provides a holistic overview of the results obtained in this research.

10.13 Looking back on the research journey

The author asserts that any decisions taken throughout the research were carefully considered and deemed appropriate. However, certain compromises were made at various stages due to the stipulated time frame. The author had intended to evaluate other neural ODEs before the LTC. Unfortunately, time constraints and unforeseen challenges hindered this effort. Evaluating them would have streamlined the development procedure, as similarities between the LTC and other neural ODEs would lead to a more seamless development experience.

The author encountered limitations regarding participant availability during the interview process, resulting in feedback from a relatively small number of individuals. Despite the best efforts to solicit feedback, some participants were unresponsive, while others had busy schedules that made scheduling interviews challenging. The author could have reached out to them at an earlier time.

Regarding project planning, the author diligently followed the scheduled Gantt chart to the best of their abilities; however, minor deviations were inevitable. Nevertheless, the author proactively communicated each *checkpoint* with their supervisor to obtain feedback and ensure alignment between both parties.

While the implementation of the LTS yielded promising results, the author acknowledges that further analyses could have been conducted to bolster the study's findings. For example, alternative SDE solvers could have been evaluated and compared to Euler-Maruyama to determine whether it was indeed the best choice for this particular application.

As a final piece of advice, the author suggests that exploring more rigorous and reliable methods for evaluating Keras layers beyond the current practice of manual debugging and unit testing, as currently suggested and recommended by Keras, would be a worthwhile endeavor.

10.14 Concluding remarks

This marks the end of researching, designing, and developing a novel algorithmic solution to address a significant challenge in TS forecasting that hindered its performance compared to other applications of DL. Its application was demonstrated in a BTC forecasting application - BTC being very volatile made it an ideal candidate for effective evaluation - exhibiting promising performance that can be improved to apply in other domains as well.

REFERENCES

Abraham, J., Higdon, D., Nelson, J. and Ibarra, J. (2018). Cryptocurrency Price Prediction Using Tweet Volumes and Sentiment Analysis. *SMU Data Science Review*: Vol. 1: No. 3, Article 1. Available at: **https://scholar.smu.edu/datasciencereview/vol1/iss3/1**

Ackerman, D. (2021). "Liquid" machine-learning system adapts to changing conditions. *MIT News | Massachusetts Institute of Technology*. Available from **https://news.mit.edu/2021/machine-learning-adapts-0128** [Accessed 17 October 2022].

Ahmed, NK., Atiya, AF., Gayar, NE. and El-Shishiny, H. An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews*. 2010;29(5-6):594–621.

Alonso-Monsalve, S. et al. (2020). Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems with Applications*, 149, 113250. Available from **https://doi.org/10.1016/j.eswa.2020.113250** [Accessed 22 October 2022].

Anumasa, S. and Srijith, P.K. (2022). Latent Time Neural Ordinary Differential Equations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (6), 6010–6018. Available from **https://doi.org/10.1609/aaai.v36i6.20547** [Accessed 17 October 2022].

Asana (2022). Project Management Methodologies: 12 Best Frameworks [2022] • Asana. *Asana*. Available from **https://asana.com/resources/project-management-methodologies** [Accessed 23 April 2023].

Avdeychik, V., & Capozzi, J. (2018). SEC's Division of Investment Management Voices Concerns Over Registered Funds Investing in Cryptocurrencies and Cryptocurrency Related Products. *Journal of Investment Compliance*, 19(2), 8–12.

Baldimtsi, F., Kiayias, A., & Samari, K. (2017). Watermarking Public-Key Cryptographic Functionalities and Implementations. In Nguyen, P. Q., & Zhou, J. (Eds.). *Information Security*, 173–191. Berlin: Springer.

Bhardwaj, S.P. et al. (2014). An Empirical Investigation of Arima and Garch Models in Agricultural Price Forecasting. *Economic Affairs*, 59 (3), 415. Available from **https://doi.org/10.5958/0976-4666.2014.00009.6** [Accessed 17 October 2022].

BI4ALL (2021). Supervised Machine Learning in Time Series Forecasting. *BI4ALL – Turning Data into Insights*. Available from **https://www.bi4all.pt/en/news/en-blog/supervised-machine-learning-in-time-series-forecasting/** [Accessed 12 October 2022].

Boisdequin, H. (2020). React vs Vue vs Angular vs Svelte. *DEV Community* 👩‍💻👨‍💻. Available from **https://dev.to/hb/react-vs-vue-vs-angular-vs-svelte-1fdm** [Accessed 18 October 2022].

Boom, V.D. (2021). Want to Buy Crypto? Here's What to Look for In a Crypto Exchange. *Time, 11 June*. Available from **https://time.com/nextadvisor/investing/cryptocurrency/what-are-cryptocurrency-exchanges** [Accessed 23 October 2022].

Bouktif, S. et al. (2018). Optimal Deep Learning LSTM Model for Electric Load Forecasting using Feature Selection and Genetic Algorithm: Comparison with Machine Learning Approaches †. *Energies*, 11 (7), 1636. Available from **https://doi.org/10.3390/en11071636** [Accessed 28 December 2022].

Bouri, E., Gupta, R., Lau, C. K. M., Roubaud, D., & Wang, S. (2018). Bitcoin and global financial stress: A copula-based approach to dependence and causality in the quantiles. *The Quarterly Review of Economics and Finance*, 69, 297–307.

Buhalis, D. et al. (2019). Technological disruptions in services: lessons from tourism and hospitality. *Journal of Service Management*, 30 (4), 484–506. Available from **https://doi.org/10.1108/JOSM-12-2018-0398** [Accessed 22 October 2022].

Chaman L. Jain. Answers to your forecasting questions. *Journal of Business Forecasting*, 36, Spring 2017.

Chen, R.T.Q. et al. (2019). Neural Ordinary Differential Equations. Available from **https://doi.org/10.48550/arXiv.1806.07366** [Accessed 25 September 2022].

Cho, K. et al. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Available from **https://doi.org/10.48550/ARXIV.1406.1078** [Accessed 9 April 2023].

Critien, J.V., Gatt, A. and Ellul, J. (2022). Bitcoin price change and trend prediction through twitter sentiment and data volume. *Financial Innovation*, 8 (1), 45. Available from **https://doi.org/10.1186/s40854-022-00352-7** [Accessed 16 October 2022].

DeMarco, T. (1979). Structure Analysis and System Specification. In: Broy, M. and Denert, E. (eds.). *Pioneers and Their Contributions to Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 255–288. Available from **https://doi.org/10.1007/978-3-642-48354-7_9** [Accessed 24 April 2023].

Dubovikov, K. (2018). PyTorch vs TensorFlow — spotting the difference. *Medium*. Available from **https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b** [Accessed 18 October 2022].

Duvenaud, D (2021). Directions in ML: Latent Stochastic Differential Equations: An Unexplored Model Class. *YouTube*. Available from **https://www.youtube.com/watch?v=6iEjF08xgBg**. [Accessed on 30 Sep. 2022].

Engle, R.F. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflation. *Econometrica*, 50 (4), 987. Available from **https://doi.org/10.2307/1912773** [Accessed 28 December 2022].

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270 (2), 654–669. Available from **https://doi.org/10.1016/j.ejor.2017.11.054** [Accessed 14 February 2023].

Fleischer, J.P. et al. (2022). Time Series Analysis of Cryptocurrency Prices Using Long Short-Term Memory. *Algorithms*, 15 (7), 230. Available from **https://doi.org/10.3390/a15070230** [Accessed 26 September 2022].

Fortune Business Insights (2021). Cryptocurrency Market Size, Growth & Trends | Forecast [2028]. *Fortune Business Insights*. Available from

https://www.fortunebusinessinsights.com/industry-reports/cryptocurrency-market-100149 [Accessed 23 October 2022].

Funahashi, K. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6 (6), 801–806. Available from **https://doi.org/10.1016/S0893-6080(05)80125-X** [Accessed 14 October 2022].

G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, Time series analysis: forecasting and control (John Wiley & Sons, 2015).

Gilliland, M. (2014). A naive forecast is not necessarily bad. *The Business Forecasting Deal*. Available from **https://blogs.sas.com/content/forecasting/2014/04/30/a-naive-forecast-is-not-necessarily-bad/** [Accessed 15 October 2022].

Hasani, R. et al. (2020). Liquid Time-constant Networks. Available from **https://doi.org/10.48550/arXiv.2006.04439** [Accessed 25 September 2022].

Hasani, R. et al. (2021). Liquid Neural Networks. *YouTube*. Available from **https://www.youtube.com/watch?v=IlliqYiRhMU&t=350s**. [Accessed on 30 Sep. 2022].

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9 (8), 1735–1780. Available from **https://doi.org/10.1162/neco.1997.9.8.1735** [Accessed 25 September 2022].

Hutto, C., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of social media Text. *Proceedings of the International AAAI Conference on Web and social media*, 8(1), 216-225

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688

Hyndman, R.J., & Athanasopoulos, G. (2021). *Forecasting: principles and practice*, 3rd edition, OTexts: Melbourne, Australia. Available from **https://otexts.com/fpp3/**. [Accessed on 30 Sep. 2022].

IBM Cloud Team (2021). Python vs. R: What's the Difference? *IBM*. Available from **https://www.ibm.com/cloud/blog/python-vs-r** [Accessed 18 October 2022].

InterviewBit (2021). Flask Vs Django: Which Python Framework to Choose? *InterviewBit*. Available from **https://www.interviewbit.com/blog/flask-vs-django/** [Accessed 12 December 2022].

Kerr, J. (2018). How Can Legislators Protect Sport from the Integrity Threat Posed by Cryptocurrencies? *The International Sports Law Journal*, 18(1), 79–97.

Kervanci, I. sibel and Akay, F. (2020). Review on Bitcoin Price Prediction Using Machine Learning and Statistical Methods. *Sakarya University Journal of Computer and Information Sciences*. Available from **https://doi.org/10.35377/saucis.03.03.774276** [Accessed 25 September 2022].

Kfir, I. (2020). Cryptocurrencies, national security, crime and terrorism. *Comparative Strategy*, 39 (2), 113–127. Available from **https://doi.org/10.1080/01495933.2020.1718983** [Accessed 22 October 2022].

Kim, M. et al. (2019). A Hybrid Neural Network Model for Power Demand Forecasting. *Energies*, 12 (5), 931. Available from **https://doi.org/10.3390/en12050931** [Accessed 16 October 2022].

Kim, Y.B. et al. (2016). Predicting Fluctuations in Cryptocurrency Transactions Based on User Comments and Replies. *PLOS ONE*, 11 (8), e0161197. Available from **https://doi.org/10.1371/journal.pone.0161197** [Accessed 16 October 2022].

Kingma, D.P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. Available from **https://doi.org/10.48550/arXiv.1412.6980** [Accessed 27 December 2022].

Kuan, L. et al. (2017). Short-term electricity load forecasting method based on multilayered self-normalizing GRU network. *2017 IEEE Conference on Energy Internet and Energy System Integration* (EI2). November 2017. Beijing: IEEE, 1–5. Available from **https://doi.org/10.1109/EI2.2017.8245330** [Accessed 17 October 2022].

Kurama, V. (2022). PyTorch vs. TensorFlow: 2022 Deep Learning Comparison | Built In. *Built In*. Available from **https://builtin.com/data-science/pytorch-vs-tensorflow** [Accessed 12 December 2022].

Lapicque, L. 1907. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej* 9: 620–635.

Lara-Benítez, P., Carranza-García, M. and Riquelme, J.C. (2021). An Experimental Review on Deep Learning Architectures for Time Series Forecasting. *International Journal of Neural Systems*, 31 (03), 2130001. Available from **https://doi.org/10.1142/S0129065721300011** [Accessed 16 October 2022].

Li, A.W. and Bastos, G.S. (2020). Stock Market Forecasting Using Deep Learning and Technical Analysis: A Systematic Review. *IEEE Access*, 8, 185232–185242. Available from **https://doi.org/10.1109/ACCESS.2020.3030226** [Accessed 16 October 2022].

Li, S. et al. (2019). Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. Available from **https://doi.org/10.48550/ARXIV.1907.00235** [Accessed 17 October 2022].

Li, X. et al. (2020). Scalable Gradients for Stochastic Differential Equations. Available from **http://arxiv.org/abs/2001.01328** [Accessed 18 January 2023].

Lim, B. and Zohren, S. (2020). Time Series Forecasting With Deep Learning: A Survey. Available from **https://doi.org/10.1098/rsta.2020.0209** [Accessed 21 October 2022].

Lim, B. et al. (2019). Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. Available from **https://doi.org/10.48550/ARXIV.1912.09363** [Accessed 17 October 2022].

madhurihammad (2020). Difference between Structured and Object-Oriented Analysis. *GeeksforGeeks*. Available from **https://www.geeksforgeeks.org/difference-between-structured-and-object-oriented-analysis/** [Accessed 23 April 2023].

Maiti, M., Vyklyuk, Y. and Vuković, D. (2020). Cryptocurrencies chaotic co-movement forecasting with neural networks. *Internet Technology Letters*, 3 (3). Available from **https://doi.org/10.1002/itl2.157** [Accessed 16 October 2022].

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. (2018a). Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13 (3), e0194889. Available from **https://doi.org/10.1371/journal.pone.0194889** [Accessed 25 September 2022].

Makridakis, S., Spiliotis, E. and Assimakopoulos, V. (2018b). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34 (4), 802–808. Available from **https://doi.org/10.1016/j.ijforecast.2018.06.001** [Accessed 25 September 2022].

Miller, P. (2016). Chapter 1—The Cryptocurrency Enigma. In Sammons, J. (Ed.), *Digital Forensics*, 1–25. Syngress. **https://doi.org/10.1016/B978-0-12-804526-8.00001-0**

Mozer, M.C., Kazakov, D. and Lindsey, R.V. (2017). Discrete Event, Continuous Time RNNs. Available from **https://doi.org/10.48550/ARXIV.1710.04110** [Accessed 14 October 2022].

Mudassir, M. et al. (2020). Time-series forecasting of Bitcoin prices using high-dimensional features: a machine learning approach. *Neural Computing and Applications*. Available from **https://doi.org/10.1007/s00521-020-05129-6** [Accessed 3 December 2022].

Nica, O., Piotrowska, K., & Schenk-Hoppé, K. R. (2017). Cryptocurrencies: Economic Benefits and Risks. SSRN Scholarly Paper ID 3059856.

Oreshkin, B.N. et al. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. Available from **http://arxiv.org/abs/1905.10437** [Accessed 26 September 2022].

Pan, C. et al. (2019). Very Short-Term Solar Generation Forecasting Based on LSTM with Temporal Attention Mechanism. *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. December 2019. Chengdu, China: IEEE, 267–271. Available from **https://doi.org/10.1109/ICCC47050.2019.9064298** [Accessed 14 February 2023].

Pant, D.R. et al. (2018). Recurrent Neural Network Based Bitcoin Price Prediction by Twitter Sentiment Analysis. 2018 IEEE 3rd International Conference on Computing, *Communication and Security (ICCCS)*. October 2018. Kathmandu: IEEE, 128–132. Available from **https://doi.org/10.1109/CCCS.2018.8586824** [Accessed 16 October 2022].

Patadiya, J. (2021). Angular vs React | Angular vs Vue | React vs Vue - Know the Difference. *Radixweb*. Available from **https://radixweb.com/blog/angular-vs-react-vs-vue** [Accessed 12 December 2022].

Peluchetti, S. and Favaro, S. (2019). Infinitely deep neural networks as diffusion processes. Available from **https://doi.org/10.48550/ARXIV.1905.11065** [Accessed 2 December 2022].

Picasso, A. et al. (2019). Technical analysis and sentiment embeddings for market trend prediction. *Expert Systems with Applications*, 135, 60–70. Available from **https://doi.org/10.1016/j.eswa.2019.06.014** [Accessed 16 October 2022].

Poulopoulos, D. (2021). Is "Liquid" ML the answer to autonomous driving? *Medium*. Available from **https://towardsdatascience.com/is-liquid-ml-the-answer-to-autonomous-driving-bf2e899a9065** [Accessed 25 September 2022].

Pournader, M. et al. (2020). Blockchain applications in supply chains, transport and logistics: a systematic review of the literature. *International Journal of Production Research*, 58 (7), 2063–2081. Available from **https://doi.org/10.1080/00207543.2019.1650976** [Accessed 22 October 2022].

Press, W.H. (ed.). (2007). *Numerical recipes: the art of scientific computing*, 3rd ed. Cambridge, UK; New York: Cambridge University Press.

Rahouti, M., Xiong, K. and Ghani, N. (2018). Bitcoin Concepts, Threats, and Machine-Learning Security Solutions. *IEEE Access*, 6, 67189–67205. Available from **https://doi.org/10.1109/ACCESS.2018.2874539** [Accessed 25 September 2022].

Raneez, A. and Wirasingha, T. (2023). A Review On Breaking the Limits of Time Series Forecasting Algorithms. *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. 8 March 2023. Las Vegas, NV, USA: IEEE, 0482–0488. Available from **https://doi.org/10.1109/CCWC57344.2023.10099071** [Accessed 24 April 2023].

Rejeb, A., Rejeb, K. and G. Keogh, J. (2021). Cryptocurrencies in Modern Finance: A Literature Review. *ETIKONOMI*, 20 (1), 93–118. Available from **https://doi.org/10.15408/etk.v20i1.16911** [Accessed 22 October 2022].

Rizwan, M., Narejo, S. and Javed, M. (2019). Bitcoin price prediction using Deep Learning Algorithm. *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*. December 2019. Karachi, Pakistan: IEEE, 1–7. Available from **https://doi.org/10.1109/MACS48846.2019.9024772** [Accessed 26 September 2022].

Rosenthal, S. et al. (2014). SemEval-2014 Task 9: Sentiment Analysis in Twitter. Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). 2014. *Dublin, Ireland: Association for Computational Linguistics*, 73–80. Available from **https://doi.org/10.3115/v1/S14-2009** [Accessed 16 October 2022].

Roy, S., Nanjiba, S. and Chakrabarty, A. (2018). Bitcoin Price Forecasting Using Time Series Analysis. *2018 21st International Conference of Computer and Information Technology (ICCIT)*. December 2018. Dhaka, Bangladesh: IEEE, 1–5. Available from **https://doi.org/10.1109/ICCITECHN.2018.8631923** [Accessed 25 September 2022].

Rubanova, Y., Chen, R.T.Q. and Duvenaud, D. (2019). Latent ODEs for Irregularly-Sampled Time Series. Available from **https://doi.org/10.48550/ARXIV.1907.03907** [Accessed 18 October 2022].

S. Nakamoto, (2020). *Bitcoin: A peer-to-peer electronic cash system*. Available from **https://bitcoin.org/bitcoin.pdf** [Accessed 25 September 2022].

Sagheer, A. and Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*, 323, 203–213. Available from **https://doi.org/10.1016/j.neucom.2018.09.082** [Accessed 17 October 2022].

Sami, M. (2012). Software Development Life Cycle Models and Methodologies. *Mohamed Sami*. Available from **https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/** [Accessed 23 April 2023].

Sarkodie, S.A., Ahmed, M.Y. and Owusu, P.A. (2022). COVID-19 pandemic improves market signals of cryptocurrencies–evidence from Bitcoin, Bitcoin Cash, Ethereum, and Litecoin. *Finance Research Letters*, 44, 102049. Available from **https://doi.org/10.1016/j.frl.2021.102049** [Accessed 16 October 2022].

Saunders, M.N.K., Lewis, P. and Thornhill, A. (2007). *Research methods for business students*, 4th ed. Harlow, England; New York: Financial Times/Prentice Hall.

Scharding, T. (2019). National Currency, World Currency, Cryptocurrency: A Fichtean Approach to the Ethics of Bitcoin. *Business and Society Review*, 124(2), 219–238.

Serafini, G. et al. (2020). Sentiment-Driven Price Prediction of the Bitcoin based on Statistical and Deep Learning Approaches. *2020 International Joint Conference on Neural Networks (IJCNN)*. July 2020. Glasgow, United Kingdom: IEEE, 1–8. Available from **https://doi.org/10.1109/IJCNN48605.2020.9206704** [Accessed 16 October 2022].

Shen, D., Urquhart, A. and Wang, P. (2019). Does twitter predict Bitcoin? *Economics Letters*, 174, 118–122. Available from **https://doi.org/10.1016/j.econlet.2018.11.007** [Accessed 16 October 2022].

Shrivastava, S. (2020). Cross Validation in Time Series. *Medium*. Available from **https://medium.com/@soumyachess1496/cross-validation-in-time-series-566ae4981ce4** [Accessed 12 October 2022].

Siami-Namini, S., Tavakoli, N. and Siami Namin, A. (2018). A Comparison of ARIMA and LSTM in Forecasting Time Series. *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. December 2018. Orlando, FL: IEEE, 1394–1401. Available from **https://doi.org/10.1109/ICMLA.2018.00227** [Accessed 17 October 2022].

Siqueira De Cerqueira, J.A., Acco Tives, H. and Dias Canedo, E. (2021). Ethical Guidelines and Principles in the Context of Artificial Intelligence. *XVII Brazilian Symposium on Information Systems*. 7 June 2021. Uberlândia Brazil: ACM, 1–8. Available from https://doi.org/10.1145/3466933.3466969 [Accessed 27 April 2023].

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36 (1), 75–85. Available from **https://doi.org/10.1016/j.ijforecast.2019.03.017** [Accessed 25 September 2022].

Taylor, S.J. and Letham, B. (2017). Forecasting at scale. *PeerJ Preprints*. Available from **https://doi.org/10.7287/peerj.preprints.3190v2** [Accessed 17 October 2022].

Tzen, B. and Raginsky, M. (2019). Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit. Available from **http://arxiv.org/abs/1905.09883** [Accessed 2 December 2022].

Ugurlu, U., Oksuz, I. and Tas, O. (2018). Electricity Price Forecasting Using Recurrent Neural Networks. *Energies*, 11 (5), 1255. Available from **https://doi.org/10.3390/en11051255** [Accessed 14 February 2023].

Valencia, F., Gómez-Espinosa, A. and Valdés-Aguirre, B. (2019). Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning. *Entropy*, 21 (6), 589. Available from **https://doi.org/10.3390/e21060589** [Accessed 16 October 2022].

Valipour, M. (2015). Long-term runoff study using SARIMA and ARIMA models in the United States: Runoff forecasting using SARIMA. *Meteorological Applications*, 22 (3), 592–598. Available from **https://doi.org/10.1002/met.1491** [Accessed 16 October 2022].

Wang, Y. et al. (2019). Making sense of blockchain technology: How will it transform supply chains? *International Journal of Production Economics*, 211, 221–236. Available from **https://doi.org/10.1016/j.ijpe.2019.02.002** [Accessed 22 October 2022].

Wang, Y., Liao, W. and Chang, Y. (2018). Gated Recurrent Unit Network-Based Short-Term Photovoltaic Forecasting. *Energies*, 11 (8), 2163. Available from **https://doi.org/10.3390/en11082163** [Accessed 28 December 2022].

Wilson, C. (2019). Cryptocurrencies: The Future of Finance? In: Yu, F.-L.T. and Kwan, D.S. (eds.). *Contemporary Issues in International Political Economy*. Singapore: Springer Singapore, 359–394. Available from **https://doi.org/10.1007/978-981-13-6462-4_16** [Accessed 22 October 2022].

Wolf, T. et al. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. October 2020. Online: Association for Computational Linguistics, 38–45. Available from **https://doi.org/10.18653/v1/2020.emnlp-demos.6** [Accessed 19 October 2022].

Yenidogan, I. et al. (2018). Bitcoin Forecasting Using ARIMA and PROPHET. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. September 2018. Sarajevo: IEEE, 621–624. Available from **https://doi.org/10.1109/UBMK.2018.8566476** [Accessed 16 October 2022].

Zhang, R. et al. (2022). Comparison of ARIMA and LSTM for prediction of hemorrhagic fever at different time scales in China. *PLOS ONE*, 17 (1), e0262009. Available from **https://doi.org/10.1371/journal.pone.0262009** [Accessed 17 October 2022].

APPENDIX A – INTRODUCTION

A.1. Prototype feature diagram

The following diagram illustrates the prototype feature diagram that was proposed in the original project proposal document.

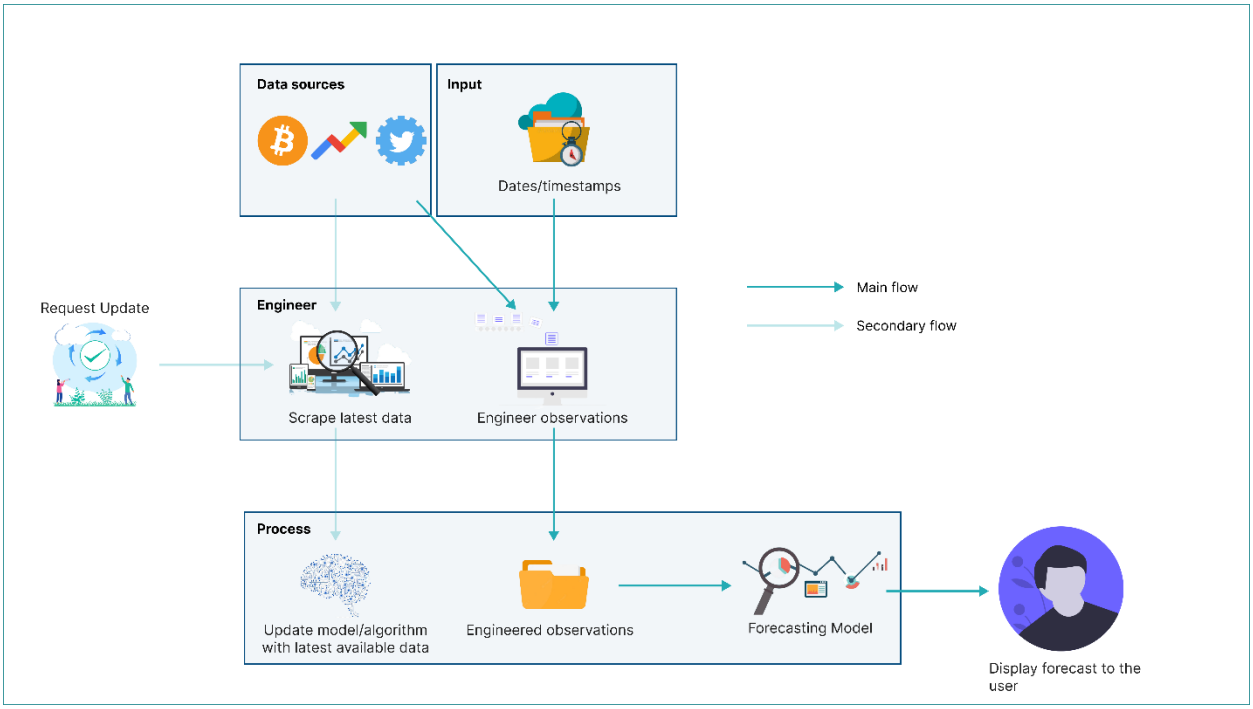


Figure 22: Prototype feature diagram (*self-composed*)

A.2. Project scope

In scope

- Implementing a novel LTC architecture capable of being used as currently existing solutions and the corresponding creation of a system.
- Periodic updates of the model with the latest available data.
- Evaluate and compare the implemented system against existing solutions.
- Ability to display a range of predictions for the chosen horizon.
- By combining them with the BTC historical data, consider Twitter sentiment, volume, and the block reward size as external factors.

Out scope

- Application of the algorithm implemented in other domains to justify whether it could be an advancement in those domains.
- Forecast multiple different cryptocurrencies.
- Use of live, on-demand data instead of daily data & incremental learning.

Desirables

- Benchmark implementation against the M4 competition to further justify the future of TS forecasting algorithms.
- Evaluate other neural ODEs (CT-RNN, CT-GRU, Latent ODE) and SDEs (Latent SDE).
- XAI for neural SDEs and neural ODEs.

APPENDIX B – LITERATURE REVIEW

B.1. Analysis of forecasting algorithms

Table 31: Analysis of forecasting algorithms (Raneez and Wirasingha, 2023)

Ref.	Brief	Contribution	Limitations
Statistical-based forecasting algorithms			
Box et al., 2015	**ARIMA**. An analytical statistical model for comprehending the dataset or forecasting future trends. This model employs lagged moving averages to smooth the data and relies on past values to forecast the future.	Improved accuracy of TS predicting data that correlates with future values.	Cannot handle long-term forecasting and non-linear data well. Additionally, it cannot capture data volatility and works best with univariate analysis.
Engle, 1982	**GARCH**. A modelling approach with a focus on forecasting data volatility.	Captures dataset volatility and claims a considerable increase in performance over previous statistical forecasting systems.	Improved flexibility and interpretability are required.
Taylor and Letham, 2017	**Prophet**. an interpretable modular regression model with parameters. Like	Solves scale-based forecasting, where scale is one of three categories. 1) A large number of people forecasting. 2) A wide	It performs substantially worse than ARIMA and uses basic and weak assumptions. Additionally, it doesn't

	ARIMA, domain experts can modify these parameters based on the problem.	range of issues. 3) The production of several forecasts.	model the connections between the past and the future.
DL-based forecasting algorithms			
Hochreiter and Schmidhuber, 1997	**LSTM**. A method for learning to minimize temporal lags by requiring continual error flows. It picks up knowledge quickly, generates more successful runs, and can complete previously unsolved complex jobs.	Enhanced short-sequence prediction performance. Overcame back-flow issues in traditional BPTT, where they tended to explode or disappear.	Long sequence performance is constrained by prediction capacity, where the MSE and RMSE increase intolerably. Therefore, there are more effective methods for predicting the far future. They also tend to overfit.
Cho et al., 2014	**GRU**. Similar architecture to LSTMs, except instead of the three gates found in LSTMs, the 'reset' and 'update' gates are created by combining the 'forget' and 'input' gates.	Solved the vanishing gradient problem in RNNs more efficiently, faster, and consume less memory.	Appropriate for issues with smaller datasets but tends to be less precise for datasets with bigger sequences.
Oreshkin et al., 2020	**N-BEATS**. An architecture to solve the univariate point forecasting problem. It has some advantages, like being simple to understand, easily adaptable, and fast to train.	Beat last year's M4 competition champion and enhance the statistical benchmark forecast.	They are designed specifically only for univariate TS analysis.

Lim et al., 2019	**TFT**. An attention model that handles multi-horizon forecasting without compromising interpretability.	Demonstrates performance improvements for many datasets.	Costly and requires a fair number of resources, training, and inference times. Hardware improvements can decrease these.
Hasani et al., 2020	**LTC**. An innovative neural ODE architecture. Has exceptional expressivity that may adjust to accommodate unforeseen circumstances.	Beat conventional DL and statistical models and outperformed other neural ODE architectures.	It is computationally demanding and unable to model uncertainty.

B.2. Studies associated with these algorithms

Table 32: Studies associated with these algorithms (Raneez and Wirasingha, 2023)

Ref.	Technology	Outperforms	Findings
Statistical-based forecasting algorithms			
Zhang et al., 2022	ARIMA	LSTM	ARIMA beat the LSTM model for monthly and weekly forecasts, but LSTM outperformed it for daily forecasts. Therefore, they remarked that there is no obvious victor.
Yenidogan et al., 2018	Prophet	ARIMA	Despite anomalies and missing data, Prophet remains robust. It is also better suited to business forecasts with internal trends, seasonality, and non-linear data growths than ARIMA.
DL-based forecasting algorithms			
Kuan et al., 2017	GRU	LSTM, traditional GRU	To handle vanishing gradients, scaled exponential linear units were developed. These models performed noticeably better than LSTM and conventional GRU models.
Ugurlu, Oksuz and Tas, 2018	GRU	MLP, LSTM	A more streamlined version of LSTM, performed far better than LSTM and trained more quickly.

Wang, Liao and Chang, 2018	GRU	LSTM, ARIMA	Groups were clustered using K-Means clustering, and the most crucial attributes were then extracted using Pearson coefficient before the model was trained.
Sagheer and Kotb, 2019	LSTM	ARIMA, GRU	An improved LSTM architecture was developed using genetic algorithms.
Bouktif et al., 2018	LSTM	MLP, Linear regression	Genetic algorithms were used to determine the time lags and layer count for an LSTM that were most effective.
Fischer and Krauss, 2018	LSTM	MLP, Logistic regression	Looked within the LSTM to uncover common stock trends in noisy data.
Pan et al., 2019	LSTM	MLP, traditional LSTM	Used a brand-new attention-based LSTM architecture to enhance the functionality of conventional LSTMs.
Oreshkin et al., 2020	N-BEATS	Competition winner	surpassed the M4 competition's previous champion with a substantial performance improvement. Found that hybrid models are not always the most effective.
Hasani et al., 2020	LTC	LSTM, GRU, CT-RNN, CT-GRU	A more reliable neural ODE implementation that was created by declaring the network's flow using a linear system of ODEs.

APPENDIX C – SRS

C.1. Requirement elicitation methodologies

Table 33: Stakeholder groups

Group	Stakeholders	Reason	Instrument
G1	Domain experts (neural ODE/SDE and blockchain/crypto)	Collect insights and knowledge from relevant research to address the research questions and identify any areas the author missed.	Interview
G2	End users (trader & buyer)	Gather requirements for the implementation of the application that utilizes the LTS.	Survey
G3	Competitors	Analyze existing systems and literature in the research and problem domain.	LR/Observations
G4	Developers	Ensure the successful completion of the project within the stipulated time frame.	Prototyping

C.2. Interview analysis

Table 34: Interview participant details

ID	Affiliation	Expertise related to the research
P1	Google Brain visiting researcher and Associate Professor at University of Toronto.	Neural ODEs and SDEs.
P2	Research scientist at Deepmind.	Neural ODEs and SDEs.
P3	Research scientist at Meta AI.	Probabilistic DL and differential equations.
P4	PhD candidate at University of Nottingham.	XAI
P5	Chief Product Officer at Niftron.	Blockchain and cryptocurrencies.

Table 35: Interview thematic analysis themes, conclusions & evidence

Theme	**Conclusion**	**Evidence**
Research Problem & Gap	The interviewees confirmed the existence of a research gap and the defined problem. They also appreciated the author's efforts in conducting this research, as only a few papers have been published in this domain.	'*Yes, there are many TS forecasting algorithms; however, many are obsolete.*' '*Yes, the chosen field of architectures can be considered an advancement.*' '*As per my knowledge, I have not seen a system using the basic LTC architecture itself, so this new architecture will be novel*.'
Requirements	The interviewees expressed concerns regarding the computational cost of ODEs and SDEs, highlighting that this could result in longer processing times and potentially lead to user-unfriendliness. Thus, the author should optimize the model as much as possible to ensure the forecasts can be produced quickly without compromising accuracy.	'*They are expensive to compute.*' '*It can be resource-intensive.*'
Advice	The author originally intended to implement only the LTC architecture proposed by Hasani et al. (2020). However, the author could enhance the architecture by using SDEs instead of ODEs, resulting in the development of the LTS. This decision was made due to the potentially significant contribution and greater significance offered by the LTS architecture.	'*I think latent ODEs are obsolete.'* '*You should look into latent SDEs instead.'* '*Latent SDEs are more flexible, you could try applying LTC architectures to those more flexible models instead'*

Other suggestions	The author concluded that while XAI is prevalent in image classification, there is a lack of literature on its application in the time series domain. Moreover, incorporating XAI into time series modelling may be challenging due to the temporal component. The author also noted a need for research on XAI for SDEs, which they may explore if time allows.	'*Yea, in the domain of TS I have not seen many explainable AI research conducted.'* '*Explainable AI is flourishing in image classification but I have not seen it in TS.'* '*Integrating explainable AI might not be straightforward as other domains.'*
Robustness	The interview provided additional validation for the data collected in the survey, as the participants suggested utilizing as many exogenous features as possible to make the model more robust. The author will consider this feedback and ensure the model uses the mentioned features.	'*It is best if you try to include as many features as possible.*' '*It is not practical to forecast with only historical prices.*'

C.3. Survey analysis

Table 36: Survey thematic analysis codes, themes, conclusions & evidence

Code	**Theme**
Exogenous factors	Robustness
Explainability, Insights	Reliability
Simplicity, Convenience	User-friendly
Tuning	Editability
On-demand	Future consideration

Theme	**Conclusion**	**Evidence**
Robustness	Participants suggested that incorporating additional factors, such	'*Use previous trends in the past.'*

	as social media trends and sentiment, is necessary to make the system robust and high-performing and not solely rely on historical prices for prediction.	'*Consider all possible external factors.'*
Reliability	Almost all respondents requested that the system provide an Explainability component so that the insights obtained can be reliable as the inference becomes as transparent as possible.	'*Insights about the forecast will be beneficial.'* '*Provide as much Explainability to make the prediction as credible as possible.'* '*The rate of success of the prediction would be useful.'*
User-friendly	A few participants suggested incorporating cryptocurrency news into the system to facilitate the inference process and make it more user-friendly.	'*Show some news about the current cryptocurrency world in the platform, so it's convenient for the users.'* '*Make the steps from choosing a date to forecasting as simple as possible.'*
Editability	An ML-knowledgeable participant mentioned that it would be ideal if the system could tune the hyperparameters of the model in use, which could be an excellent enhancement to the system.	'*Coming from a machine learning point of view, I think it'll be a good idea if there's a functionality to change the hyperparameters used.'*
Future considerations	A couple of participants mentioned some additional features the author believes they will be unable to cover, given the time allotted.	'*Predict the market for any given time duration.'* '*Ability to identify a pump and dump scenario compared to an actual increase in the price of stock/crypto.'*

C.4. Use case descriptions

Table 37: Use case description UC:03

Use case	Obtain latest data
Id	UC:03
Description	Scrape the latest available data.
Actor	Scheduler
Supporting actor (if any)	None
Stakeholders (if any)	None
Pre-conditions	None
Main flow	MF1. A cron job fetches the latest historical prices, tweets, Twitter volume, Google Trends, and block reward size data. MF2. Twitter volume, Google Trends, and block reward size are scaled and cleaned. MF3. Database collections are updated.
Alternative flows	None
Exceptional flows	EF1. The script could not fetch recent data – retry a few days later or alert Admin for manual overhaul.
Post-conditions	Data of each required feature is updated.

Table 38: Use case description UC:04

Use case	Extract sentiments
Id	UC:04
Description	Perform sentiment analysis on the scraped tweet data.
Actor	None
Supporting actor (if any)	None

Stakeholders (if any)	None
Pre-conditions	The latest available data must be scraped and available.
Main flow	MF1. Tweets undergo sentiment analysis to determine current speculation. MF2. The tweets collection in the database is updated.
Alternative flows	None
Exceptional flows	EF1. Failure to obtain a sentiment value - ignore the particular tweet when calculating the average daily sentiment value.
Post-conditions	A new enriched dataset with the features is generated.

Table 39: Use case description UC:05

Use case	Weigh sentiment based on influence
Id	UC:05
Description	Change the sentiments' value obtained from UC:04 based on certain factors.
Actor	None
Supporting actor (if any)	None
Stakeholders (if any)	None
Pre-conditions	Sentiment analysis is performed on the obtained tweets.
Main flow	MF1. The sentiments are weighted based on certain factors. MF2. Features are combined to create an enriched dataset.
Alternative flows	None
Exceptional flows	EF1. Failure to combine datasets due to missing rows for specific dates – impute the missing row with the specific feature's mean.
Post-conditions	A new enriched dataset with the features is generated.

Table 40: Use case description UC:06

Use case	Update model hyperparameters
Id	UC:06
Description	Manually change the hyperparameters used by the model.
Actor	Admin
Supporting actor (if any)	None
Stakeholders (if any)	None
Pre-conditions	All the data must be scraped and preprocessed.
Main flow	MF1. Admin authorizes themselves. MF2. Admin can change the hyperparameters in use to a set of predefined values. MF3. The system ensures data available is up-to-date (must be in this case, as the script will run periodically automatically). If not: 1. The system fetches the latest available data. 2. Sentiment analysis is performed. MF4. The system retrains the model.
Alternative flows	None
Exceptional flows	None
Post-conditions	The model is updated with the chosen hyperparameters.

Table 41: Use case description UC:07

Use case	View evaluation metrics
Id	UC:07
Description	View the current evaluation status of the used models.
Actor	Admin

Supporting actor (if any)	None
Stakeholders (if any)	None
Pre-conditions	None
Main flow	MF1. Admin authorizes themselves. MF2. Admin views current evaluation metrics.
Alternative flows	None
Exceptional flows	None
Post-conditions	None

C.5. Functional requirements

Table 42: 'MoSCoW' technique of requirement prioritization

Priority level	**Description**
M (Must have)	The author must implement requirements with this priority for the project to succeed.
S (Should have)	Requirements that would be of value but are not necessary.
C (could have)	Features that are optional and have no significant impact. It is desirable to implement them if time permits.
W (Will not have)	Requirements that will not be a part of the implementation at this point.

APPENDIX D – DESIGN

D.1. LTS algorithm intuition

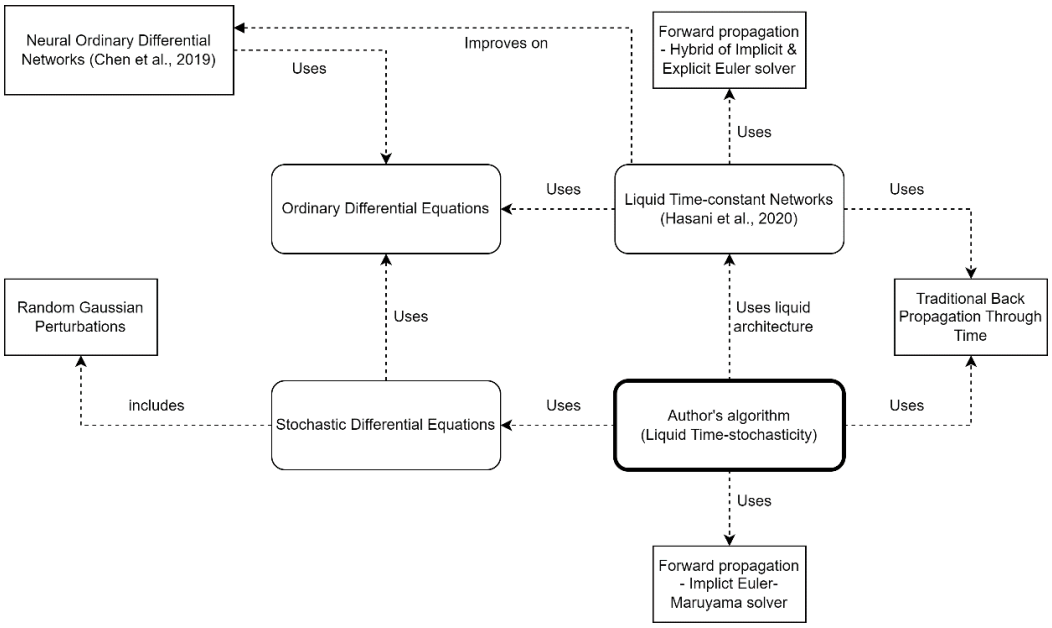


Figure 23: Algorithm intuition (*self-composed*)

What exactly an SDE solves compared to an ODE?

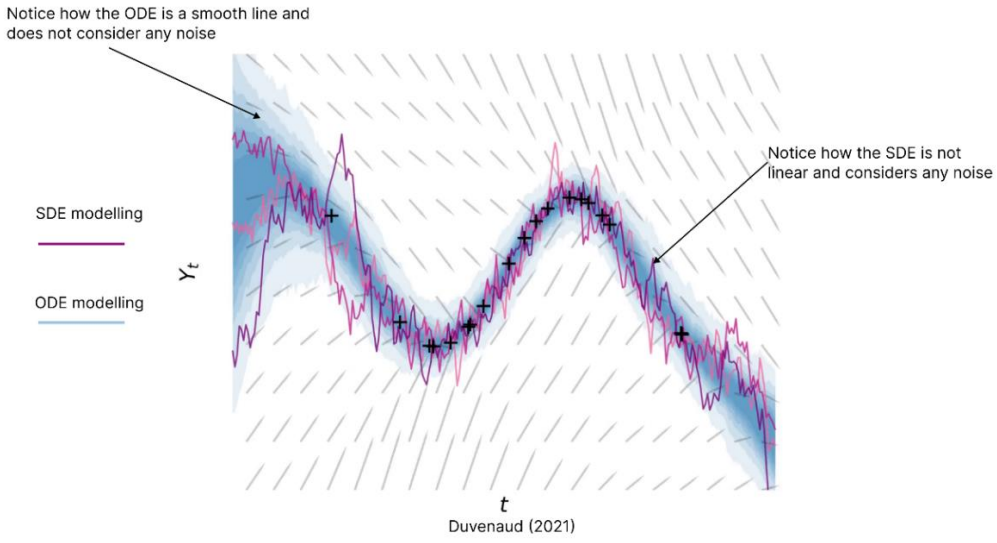


Figure 24: Understanding what an SDE solves

Considering this, SDEs would more accurately model domains that have high volatility/noise.

Formulation

Step 01 – Evolving from an ODE to an SDE

An SDE can be used by models to represent uncertainty because it is simply an ODE with more noise introduced at each step.

$$Assume\ an\ ODE\ is\colon \frac{dx}{dt} = f(x); which\ obtains\ the\ expected\ slope\ of\ x(t)$$

The expected slope of the above ODE may be calculated; however, the 'realized' slope deviates from the 'expected' due to presence of random noise (also known as random Gaussian perturbations or Gaussian white noise). Taking it into account, the following conclusion can be drawn:

$$An\ SDE\ is\colon \frac{dx}{dt} = f(x) + \mathcal{E}_{t+dt}; where\ \mathcal{E}_{t+dt}\ is \sim N(0,1)$$

$$Where\ N(0,1)\ is\ a\ Gaussian\ 0{,}1\ random\ variable$$

However, noise can have varying intensities (some sounds may be very loud, while others may be very quiet). Taking into account this fluctuating intensity, the above generic SDE can be further represented as follows:

$$\frac{dx}{dt} = f(x) + g(x) * \mathcal{E}_{t+dt}; where\ g(x)\ is\ the\ intensity$$

As mentioned, the absence of stochastic transition dynamics (i.e., a noise for each timestep - essential to describe the minute unobserved interactions) is the key component in the existing design made up of ODEs. The equation above considers any potential tiny unobserved interactions and uncertainties; this is crucial when discussing TS data because the initial state of the data is unlikely to be definite.

Step 02 – Integrating neural networks into SDE dynamics

The noise indicated in the preceding step can be regarded as Brownian motion, which is a generalized version of the Gaussian noise, according to Duvenaud's (2021) discoveries. Researchers can generate the following results by including Brownian motion into the formula established in the preceding phase.

$$dx = f\big(x(t)\big)dt + \sigma\big(x(t)\big)dB(t)$$

The system can be solved using a neural network, which can be incorporated into the previous equation to produce the following equation.

$$dx = f_\theta(x(t))dt + \sigma_\theta\big(x(t)\big)dB(t)$$

$$where\ f\ is\ usually\ a\ tiny\ neural\ network\ and\ \theta\ are\ its\ parameters$$

Step 03 –Formulating the LTS

Moving back to the core issue, the author can create a new formula by utilizing the equation discovered in the previous stage.

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t)$$

The above equation, which is a linear system of ODEs, was initially proposed by Lapicque (1907). The formula would be as follows if the uncertainty noise were included in the equation:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma\big(x(t)\big)B$$

Instead of deterministic evolution, the above equation describes a stochastic process. Researchers can predict any minute unseen interactions as a result.

Finally, by combining the formulas, the following can be derived:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma\big(x(t)\big)B$$

$$Replace\ S(t)\ with\ the\ nonlinearity\ proposed,$$

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + f(x(t), I(t), t, \theta)(A - x(t)) + \sigma\big(x(t)\big)B$$

$$Expand\ out\ the\ equation,$$

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} - f(x(t), I(t), t, \theta)x(t) + \sigma\big(x(t)\big)B + f(x(t), I(t), t, \theta)A$$

$$Lastly, refactor\ the\ equation\ into\ the\ format\ of\ the\ original\ LTC$$

$$\boxed{\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta) - \sigma B\right]x(t) + f(x(t), I(t), t, \theta)A}$$

The above derivation of the formula was adopted from Raneez and Wirasingha (2023)

D.2. Tweet sentiment weighting algorithm intuition

The proposed formula is a linear combination of the metrics considered by the author.

Initially, the metrics are specified and defined as follows:

$$followers_{count}, lists_{count}, retweets_{count}, like_{count}$$

Alpha, beta, gamma, and delta are weighting factors applied to each metric based on their observed impact. These factors are used to calculate a linear combination of the metrics, which results in a final score or prediction. The formula for this calculation will be as follows:

$$\alpha followers_{count}, \beta lists_{count}, \gamma retweets_{count}, \delta like_{count}$$

The author wanted to provide more weightage to the tweeter themselves rather than the specific tweet, as the tweet's impact is more correlated with the tweeter rather than its likes and retweets. As such, the author proposes the values of **0.5**, **0.3**, **0.1** and **0.1** for alpha, beta, gamma and delta respectively.

A logarithm can be applied to prevent a specific metric from dominating the score. This is particularly important when the number of followers is significantly larger than the number of retweets, for instance. By applying a logarithm, all metrics can be normalized and balanced to ensure a fair representation of their true impact on the score. To avoid mathematical errors that may arise when the metric value is 0, the author will add 1 to each metric before obtaining the logarithmic value. Therefore, the terms can be written as follows:

$$\alpha \log_{10}(followers_{count} + 1), \beta \log_{10}(lists_{count} + 1), \gamma \log_{10}(retweets_{count} + 1), \delta \log_{10}(like_{count} + 1)$$

As the formula is a linear combination of these metrics, they are summed up to provide a tweet sum and an influencer sum.

$$influencer_{sum} = \alpha \log_{10}(followers_{count} + 1) + \beta \log_{10}(lists_{count} + 1)$$

$$tweet_{sum} = \gamma \log_{10}(retweets_{count} + 1) + \delta \log_{10}(like_{count} + 1)$$

The obtained sums do not necessarily fall within a specific range, and it may be desirable to normalize them to ensure that the final score is between 0 and 1. Normalization can be applied as follows:

$$influencer_{score} = \frac{tweet_{sum} + influencer_{sum}}{tweet_{sum} + influencer_{sum} + 1}$$

Finally, this influencer score is applied to the vanilla compound score - initially obtained upon conducting sentiment analysis – to obtain a weighted compound score.

$$\boxed{weighted_{score} = influencer_{score} * compound_{score}}$$

D.3. LTS algorithm complexity analysis

Table 43: Complexities of BPTT and adjoint sensitivity (Raneez and Wirasingha, 2023)

Note: L = number of steps

	BPTT	**Adjoint sensitivity**
Time	O(L)	**O(LlogL)**
Memory	O(L)	**O(1)**
Forward accuracy	High	High
Backward accuracy	**High**	Low

D.4. Deployment pipeline

The below diagram illustrates the deployment pipeline, and how various components interact with each other.

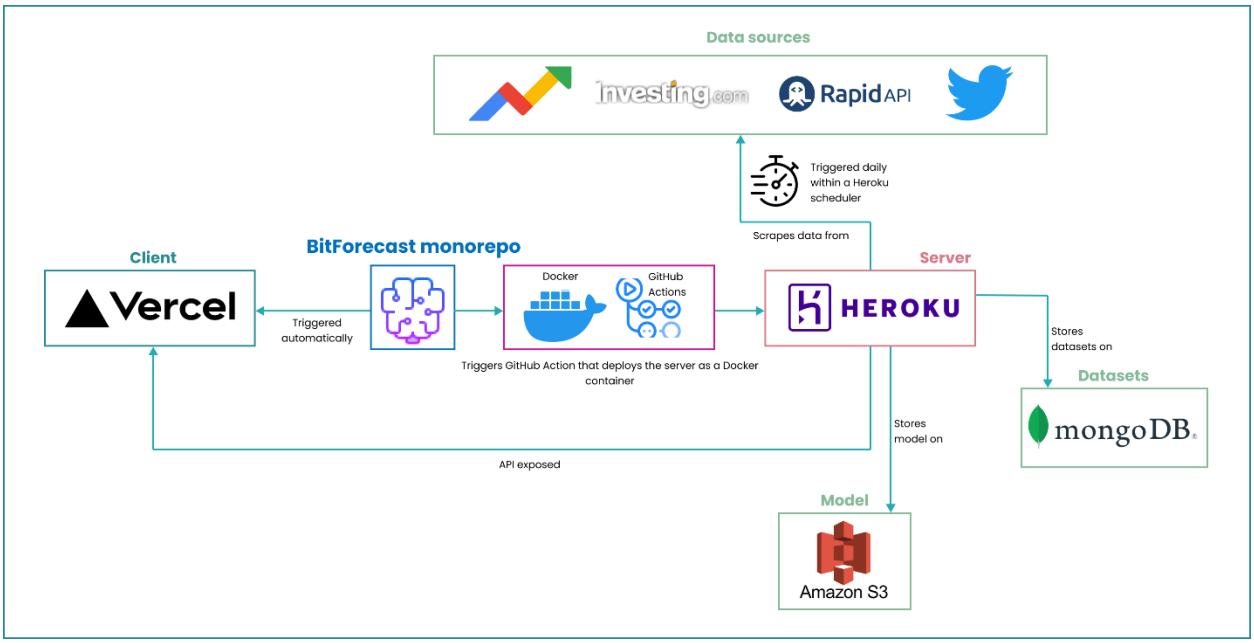


Figure 25: Pipeline diagram (*self-composed*)

D.5. UI wireframes

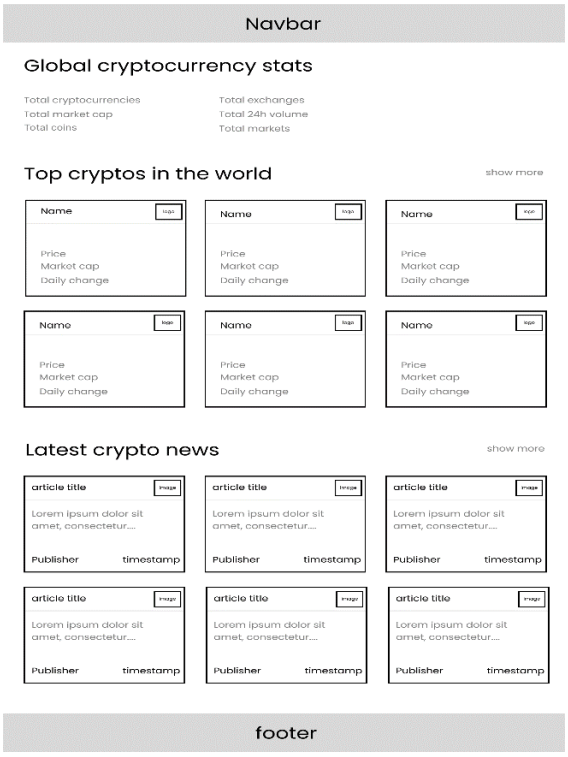


Figure 26: UI wireframes – Home (*self-composed*)

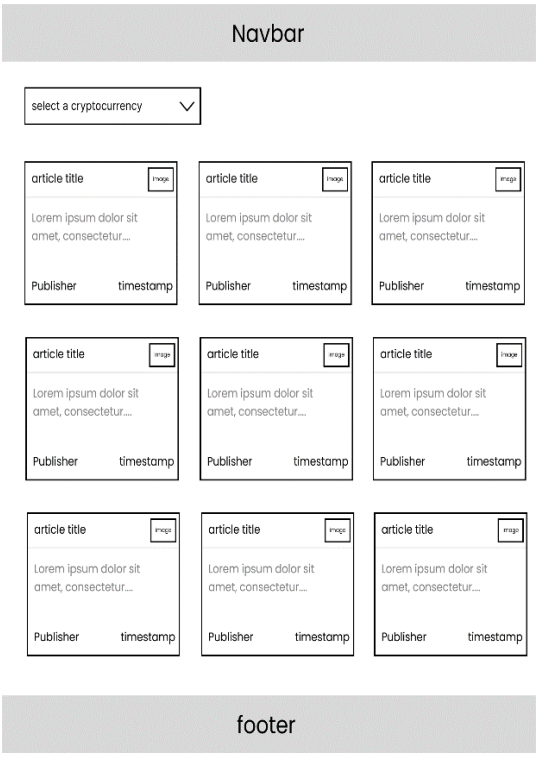


Figure 27: UI wireframes – News (*self-composed*)

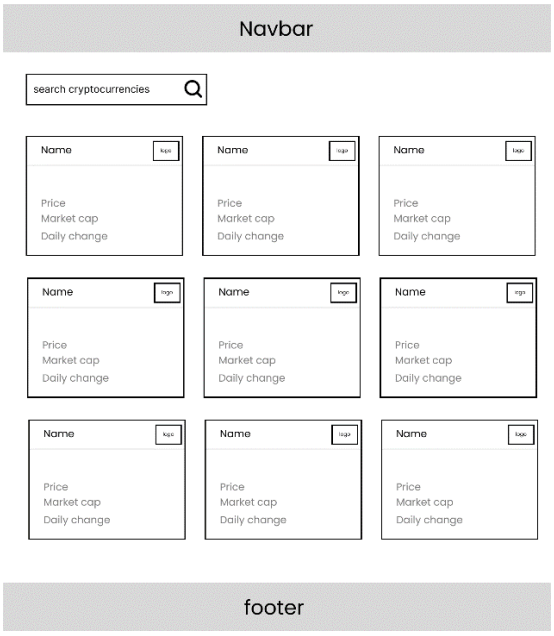


Figure 28: UI wireframes – Cryptocurrencies (*self-composed*)

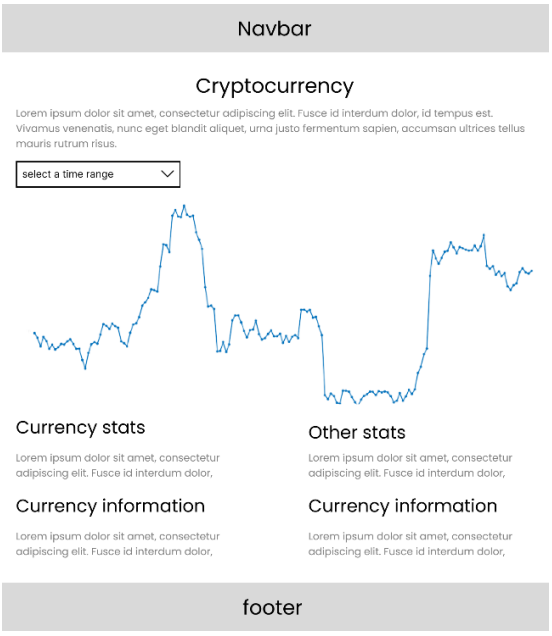


Figure 29: UI wireframes – Cryptocurrency (*self-composed*)

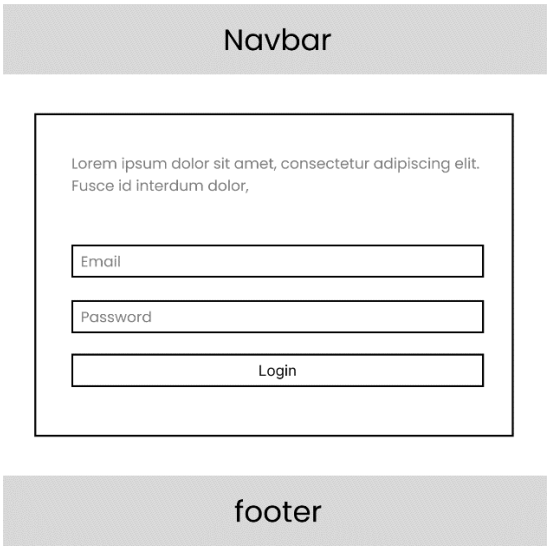


Figure 30: UI wireframes – Admin login (*self-composed*)

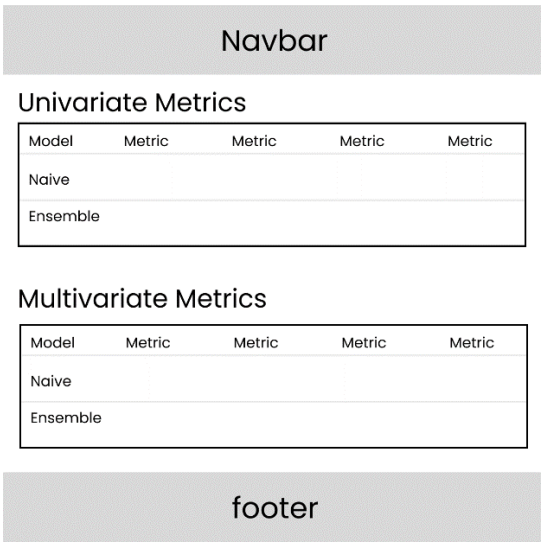


Figure 31: UI wireframes – Admin metrics (*self-composed*)

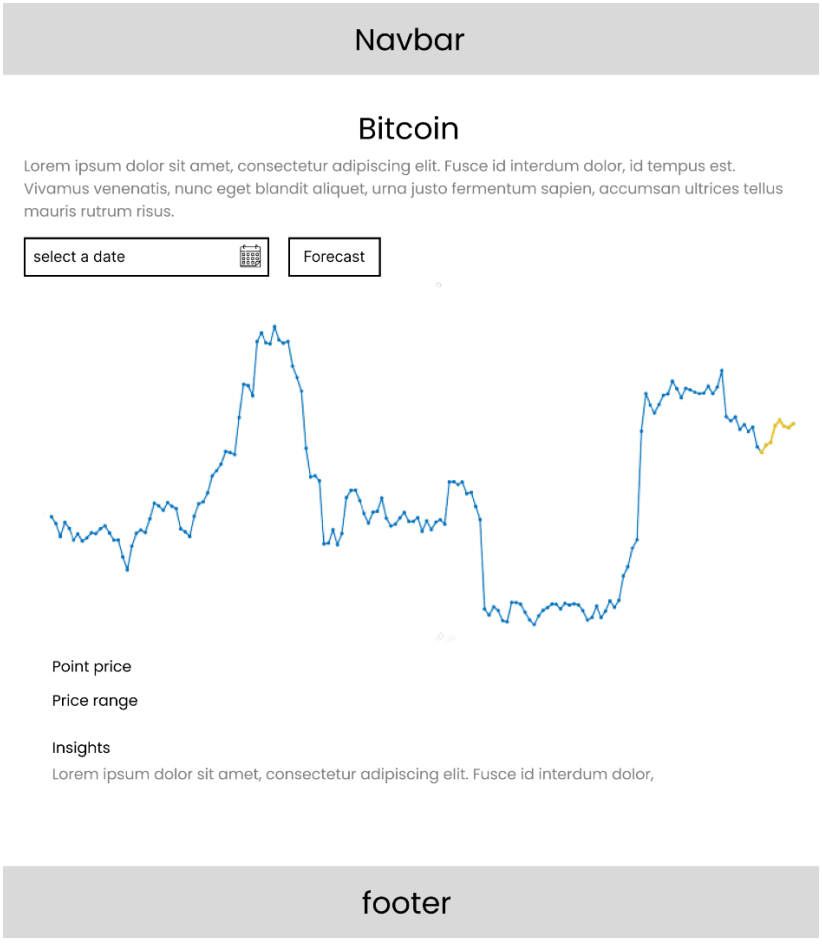


Figure 32: UI wireframes – Forecast (*self-composed*)

APPENDIX E – IMPLEMENTATION

E.1. Selection of programming language

The below table summarizes the analysis of the programming languages for the data science component. Each option was given a score within H – High, M – Medium, and L – Low.

Table 44: Selection of data science language

Data science Two of the most popular languages used widely for data science were analyzed to implement the core data science components.			
Aspect	**Relevance**	**Python**	**R**
Availability of libraries.	A language supporting multiple libraries is paramount, as the author would require numerous techniques to gather the necessary data and streamline the model and algorithm development.	H	M
Author familiarity and ease of implementation.	Implementing the algorithm, the mathematical intricacies, and the respective model should be as simple as possible. It is also beneficial if the author has hands-on experience with the chosen language.	H	M
Learning curve	The difficulty of the chosen language must not be a hindrance, as the goal is to utilize the tool to implement a system rather than spend time learning the language.	L	M
Community and documentation.	Community support and well-written documentation are important, as the author will not have time to debug trivial issues.	H	M
Conclusion Based on the analysis, the author decided to use **Python**, as it was more relevant.			

E.2. Selection of Deep Learning (DL) framework

Table 45: Selection of DL framework

Framework	Description
TensorFlow	A widely used and popular DL framework. Used for production-level applications, has detailed documentation and community support, and handles large datasets. It also provides better visualization options, making it easy to debug and monitor training, which is vital as a novel algorithm is being built, and no comparison is present.
PyTorch	It is more lightweight and developer-friendly, as it provides a higher-level development. Therefore, it has a much smaller learning curve, easier to get started, and feels more intuitive as it is simpler to build models.
Conclusion The author opted to use **TensorFlow**, as although it is more complicated, the higher-level API, Keras, is now officially a part of TensorFlow, making model development much more straightforward. Additionally, building the algorithm requires more low-level details (Kurama, 2022).	

E.3. Selection of User Interface (UI) framework

Table 46: Selection of UI framework

Framework	Description
Angular	A framework with a modular architecture that enables scalable development for large-scale applications. It can be less performant than other frameworks and has a steep learning curve.
Vue	A lightweight and intuitive framework with a growing community and a well-documented API. It has a companion tool called Vue CLI that provides a streamlined development experience.
Svelte	A lightweight and reactive framework that can be more performant than others due to its compilation approach. It has gained significant traction in recent years and has an active community.

React	A customizable and widely used framework that promotes code reusability via components. It has a large and active community, and the React developer tools are handy. While it can be used to build SEO-friendly applications, additional effort and consideration are required.

Conclusion

The author chose **React** as the framework for developing the user interface based on the analysis. React's customizable and reusable components, large community, and SEO-friendliness make it a suitable choice for this project. Additionally, React's developer tools provide helpful features. While other frameworks, such as Angular and Vue, have their advantages, they are better suited for larger-scale applications, which is not required for this project (Patadiya, 2021).

E.4. Selection of Application Programming Interface (API) framework

Table 47: Selection of web framework

Framework	**Description**
Flask	A lightweight framework that offers minimal features, making it ideal for developing simple ML APIs. Its lightweight nature allows for fast and efficient development.
Django	A more robust framework suitable for larger applications that require extensive functionalities. It is more rigid and less flexible than Flask but offers more features out-of-the-box, making it a better choice for complex web applications. However, its more demanding nature can lead to slower development times.

Conclusion

The author chose **Flask** as it provides only the necessities for exposing an ML model - the luxury features provided by Django (ex: authentication) were not required, which made it the preferred choice (InterviewBit, 2021).

E.5. Fetch data

Fetch historical prices

```python
# API reference: http://api.scraperlink.com/investpy/
BASE_URL = 'http://api.scraperlink.com/investpy/?email=ammarraneez@gmail.com&type=historical_data&product=cryptos&symbol=BTC&key=474f2c8d88ee117dc1408e97d03c6a24a745db9c'

def get_crypto_data(start, end):
    '''
    Scrape data current solution
    Possible to break in future, therefore must create a dedicated scraper, if time permits
    '''

    response = requests.request(
        'GET',
        f'{BASE_URL}&from_date={start}&to_date={end}'
    )

    return response.json()['data']

def create_dataframe(prices):
    '''
    Create dataframe of fetched prices
    '''

    return pd.DataFrame(prices)

def clean_data(prices):
    '''
    Clean data and remove unneeded columns
    '''

    df = create_dataframe(prices)
    df.drop(['direction_color', 'rowDateRaw', 'last_close', 'last_open', 'last_max', 'last_min', 'volume', 'change_precent'], axis=1, inplace=True)
    df.rename(columns={ 'volumeRaw': 'volume', 'last_closeRaw': 'close', 'last_openRaw': 'open', 'last_maxRaw': 'max', 'last_minRaw': 'min', 'change_precentRaw': 'change_percent' }, inplace=True)
    df['date'] = pd.to_datetime(df['rowDate'])
    df.drop(['rowDate', 'rowDateTimestamp'], axis=1, inplace=True)
    df.sort_values(['date'], inplace=True)
    df['date'] = df['date'].astype(str)
    df['close'] = df['close'].astype(float)
    # df.set_index('date', inplace=True)
    return df

def export_data(df):
    '''
    Save data
    '''

    # Store datasets in mongodb for any requirements in production
    df.index = df.index.astype(str)
    df.sort_values(['date'], inplace=True)
    df_dict = df.to_dict('index')
    dataset_db = init_mongodb()
    dataset_db[BTC_PRICES_COLLECTION].delete_many({})
    dataset_db[BTC_PRICES_COLLECTION].insert_one(df_dict)
    print('Saved data to MongoDB')
```

Figure 33: Fetch historical prices (*self-composed*)

The script above defines functions that fetch BTC's latest historical price data and store it in a MongoDB collection. A third-party API was utilized since existing APIs are no longer available. The model can later access the fetched data when necessary.

Fetch Twitter volume, Google Trends & block reward size

```
def parse(string_list):
    '''
    parse list of strings within the script tag
    [date, volume]
    '''

    clean = re.sub('[\[\],\s]', '', string_list)
    splitted = re.split("[\'\"]", clean)
    values_only = [s for s in splitted if s != '']
    return values_only

def process_scripts():
    '''
    Scrape URL script tag and extract tweet volume & respective date
    '''

    dates = []
    tweets = []

    for script in scripts:
        if 'd = new Dygraph(document.getElementById("container")' in script.text:
            str_lst = script.text
            str_lst = '[[' + str_lst.split('[[')[-1]
            str_lst = str_lst.split(']]')[0] +']]'
            str_lst = str_lst.replace('new Date(', '').replace(')', '')
            data = parse(str_lst)

    for each in data:
        if (data.index(each) % 2) == 0:
            dates.append(each)
        else:
            try:
                tweets.append(float(each))
            except:
                tweets.append(None)

    return dates, tweets

def create_dataframe():
    '''
    Create dataframe from scraped twitter volume and dates
    '''

    dates, tweets = process_scripts()
    df = pd.DataFrame(list(zip(dates, tweets)), columns=['Date', 'Tweet Volume'])
    return df

def export_data(df):
    '''
    Save data
    '''

    # Store datasets in mongodb for any requirements in production
    df.index = df.index.astype(str)
    df.sort_values(['Date'], inplace=True)
    df_dict = df.to_dict('index')
    dataset_db = init_mongodb()
    dataset_db[TWITTER_VOLUME_COLLECTION].delete_many({})
    dataset_db[TWITTER_VOLUME_COLLECTION].insert_one(df_dict)
    print('Saved data to MongoDB')
```

Figure 34: Fetch Twitter volume (*self-composed*)

```
def process_scripts():
    '''
    Scrape URL script tag and extract block reward & respective date
    '''

    dates = []
    sizes = []

    for script in scripts:
        if 'd = new Dygraph(document.getElementById("container")' in script.text:
            str_lst = script.text
            str_lst = '[[' + str_lst.split('[[')[-1]
            str_lst = str_lst.split(']]')[0] +']]'
            str_lst = str_lst.replace('new Date(', '').replace(')', '')
            data = parse(str_lst)

    for each in data:
        if (data.index(each) % 2) == 0:
            dates.append(each)
        else:
            try:
                sizes.append(float(each))
            except:
                sizes.append(None)

    return dates, sizes

def create_dataframe():
    '''
    Create dataframe from scraped block reward sizes and dates
    '''

    dates, sizes = process_scripts()
    df = pd.DataFrame(list(zip(dates, sizes)), columns=['Date', 'Block Reward Size'])
    return df

def export_data(df):
    '''
    Save data
    '''

    # Store datasets in mongodb for any requirements in production
    df.index = df.index.astype(str)
    df.sort_values(['Date'], inplace=True)
    df_dict = df.to_dict('index')
    dataset_db = init_mongodb()
    dataset_db[BLOCK_REWARD_COLLECTION].delete_many({})
    dataset_db[BLOCK_REWARD_COLLECTION].insert_one(df_dict)
    print('Saved data to MongoDB')
```

Figure 35: Fetch block reward size (*self-composed*)

```
def parse(string_list):
    '''
    parse list of strings within the script tag
    [date, volume]
    '''

    clean = re.sub('[\[\],\s]', '', string_list)    You, 6 days ago • feat: save data
    splitted = re.split("[\'\"]", clean)
    values_only = [s for s in splitted if s != '']
    return values_only

def process_scripts():
    '''
    Scrape URL script tag and extract trends & respective date
    '''

    dates = []
    trends = []

    for script in scripts:
        if 'd = new Dygraph(document.getElementById("container")' in script.text:
            str_lst = script.text
            str_lst = '[[' + str_lst.split('[[')[-1]
            str_lst = str_lst.split(']]')[0] +']]'
            str_lst = str_lst.replace('new Date(', '').replace(')', '')
            data = parse(str_lst)

    for each in data:
        if (data.index(each) % 2) == 0:
            dates.append(each)
        else:
            try:
                trends.append(float(each))
            except:
                trends.append(None)

    return dates, trends

def create_dataframe():
    '''
    Create dataframe from scraped trends and dates
    '''

    dates, trends = process_scripts()
    df = pd.DataFrame(list(zip(dates, trends)), columns=['Date', 'bitcoin_unscaled'])
    return df

def export_data(df):
    '''
    Save data
    '''

    # Store datasets in mongodb for any requirements in production
    df.index = df.index.astype(str)
    df.sort_values(['Date'], inplace=True)
    df_dict = df.to_dict('index')
    dataset_db = init_mongodb()
    dataset_db[TRENDS_COLLECTION].delete_many({})
    dataset_db[TRENDS_COLLECTION].insert_one(df_dict)
    print('Saved data to MongoDB')
```

Figure 36: Fetch Google trends (*self-composed*)

The above scripts scrape data for Twitter volume, Google Trends, and block reward size from a publicly available website. As no authentication or authorization is required, a simple web scraping tool can extract the required information.

Fetch tweet data

```
def scrape_tweets(dates):
    '''
    Scrape tweets of the specified dates
    '''

    tweets_list = {}
    for i, date in tqdm(enumerate(dates)):
        print(f'Trying date: {date} | Currently at index: {i}')
        try:
            next_day = pd.Timestamp(date) + datetime.timedelta(days=1)
            for j, tweet in tqdm(enumerate(sntwitter.TwitterSearchScraper(
                f'#bitcoin -filter:retweets since:{dt(date).strftime("%Y-%m-%d")} until:{dt(next_day).strftime("%Y-%m-%d")}'
            ).get_items())):
                if j > 500:
                    break
                if not tweets_list.get(date):
                    tweets_list[date] = []
                tweets_list[date].append([tweet.date, tweet.user, tweet.retweetCount, tweet.likeCount, tweet.rawContent])
        except Exception as e:
            print(f'Error: {e}')

    return tweets_list
```

Figure 37: Scrape tweets (*self-composed*)

The author used a third-party library to obtain tweet data, as the Twitter API only provides access to tweets from the past week. Due to time, performance, and storage constraints, only 500 tweets were fetched daily, and only tweets containing the hashtag '#bitcoin' were included. Initially, tweets were collected up to a specific time, but in the future, the above script could be used to scrape tweets for specific dates corresponding to the days in the collection up to the day the script is run. It should be noted that obtaining tweet data requires a more laborious process than other data sources.

```python
def clean_tweets(dates):
    '''
    Clean tweets that have empty records and non-english tweets
    '''

    print('Scraping tweets ... \n')
    tweets_list = scrape_tweets(dates)
    print('Tweets scraped\n')
    scraped_dfs = process_tweets(tweets_list)

    print('Cleaning tweets ... \n')
    for i, df in enumerate(tqdm(scraped_dfs)):
        if df.iloc[0].get('timestamp'):
            filename = str(df.iloc[0]['timestamp'])
        else:
            filename = str(df.iloc[0]['date'])

        print(f'Currently at df: {i+1} | {filename}')
        df.dropna(subset=['user', 'timestamp', 'text', 'user_total_followers', 'user_total_listed', 'tweet_retweets', 'tweet_likes'], inplace=True)
        L = []
        for row in df['text']:
            # Use lingua to remove any non-english observations
            if len(row) != 0:
                L.append(detector.detect_language_of(row))
            else:
                L.append(None)

        df['lang'] = L
        df_filtered = df.loc[df.loc[:, 'lang'] == Language.ENGLISH].copy(deep=True)
        df_filtered.drop(['lang'], axis=1, inplace=True)

    print('Tweets cleaned\n')
    return scraped_dfs
```

Figure 38: Clean tweets (*self-composed*)

As this research is currently limited to only English, the tweets are filtered, and non-English tweets are removed using Lingua – a language detector library.

E.6. Preprocessing

Tweet sentiment analysis

As identified in previous chapters, the first step in preprocessing the data is to conduct sentiment analysis on the retrieved tweets - performed using the VADER sentiment analyzer.

```python
def calculate_sentiment(sentence):
    '''
    Calculate the sentiment of a single tweet (sentence)
    '''

    sid_obj = SentimentIntensityAnalyzer()
    try:
        sentiment_dict = sid_obj.polarity_scores(sentence)
        return sentiment_dict['neg'], sentiment_dict['neu'], sentiment_dict['pos'], sentiment_dict['compound']
    except Exception as e:
        print(f'Something went wrong with this sentence: {sentence}')
        return e

def analyze_sentiment(dfs):
    '''
    Updates all dfs with respective sentiment columns
    '''

    sentiment_analyzed_dfs = []
    for i, df in tqdm(enumerate(dfs)):
        # Certain files have timestamp, certain have date
        if df.iloc[0].get('timestamp'):
            df_filename = str(df.iloc[0]['timestamp'])
        else:
            df_filename = str(df.iloc[0]['date'])

        print(f'Currently at df: {i+1} | {df_filename}')
        negative_scores = []
        positive_scores = []
        neutral_scores = []
        compound_scores = []

        for j in range(df.shape[0]):
            try:
                neg, neu, pos, compound = calculate_sentiment(preprocess(df.iloc[j]['text']))
            except:
                neg, neu, pos, compound = None, None, None, None

            negative_scores.append(neg)
            positive_scores.append(pos)
            neutral_scores.append(neu)

            # Weigh the compound score here based on the proposed formula
            weighted_compound_score = compound
            alpha, beta, gamma, delta = 0.5, 0.3, 0.1, 0.1
            followers, lists, retweets, likes = df.iloc[j]['user_total_followers'], df.iloc[j]['user_total_listed'], df.iloc[j]['tweet_retweets'], df.iloc[j]['tweet_likes']
            weighted_followers = alpha * math.log10(followers + 1)
            weighted_lists = beta * math.log10(lists + 1)
            weighted_retweets = gamma * math.log10(retweets + 1)
            weighted_likes = delta * math.log10(likes + 1)
            weighted_sum = weighted_followers + weighted_lists + weighted_retweets + weighted_likes
            influencer_score = weighted_sum / (weighted_sum + 1)
            weighted_compound_score *= influencer_score
            compound_scores.append(weighted_compound_score)

        df['negative_score'] = negative_scores
        df['positive_score'] = positive_scores
        df['neutral_score'] = neutral_scores
        df['compound_score'] = compound_scores
        sentiment_analyzed_dfs.append(df)

    return sentiment_analyzed_dfs
```

Figure 39: Analyze sentiments (*self-composed*)

The above script performs sentiment analysis using the VADER sentiment library on the tweets and concatenates the negative, positive, neutral, and compound scores into the existing tweet dataset. The compound score is weighted beforehand by utilizing the proposed sentiment weighting formula in **Chapter 6** (as only the compound score is used subsequently, there is no requirement to weigh the negative, positive, and neutral scores). They can then be condensed down to create an average score for a single day.

Tweet dataset condensation

```python
def read_mongo_df():
    '''
    Load all existing condensed df
    '''

    db = init_mongodb()
    sentiments = db[TWITTER_SENTIMENTS_COLLECTION].find_one()
    del sentiments['_id']
    df = pd.DataFrame.from_dict(sentiments, orient='index')
    return df

def condense(dfs):
    '''
    Condense tweet dfs into a single df of averaged sentiment values for each date
    '''

    condensed_df = None
    existing_df = read_mongo_df()

    for i, df in tqdm(enumerate(dfs)):
        # Certain files have timestamp column, certain have date
        if df.iloc[0].get('timestamp'):
            df_filename = str(df.iloc[0]['timestamp'])
        else:
            df_filename = str(df.iloc[0]['date'])
        print(f'Currently at df: {i+1} | {df_filename}')

        # Get the average values for each date
        averages = list(df[['negative_score', 'neutral_score', 'positive_score', 'compound_score']].mean())
        data = {
            'date': [df_filename],
            'negative_score': averages[0],
            'neutral_score': averages[1],
            'positive_score': averages[2],
            'compound_score': averages[3],
        }

        tweet_df = pd.DataFrame(data, index=None)
        if condensed_df is not None:
            condensed_df = pd.concat([condensed_df, tweet_df])
        else:
            condensed_df = pd.DataFrame(data, index=None)

    condensed_combined_df = pd.concat([existing_df, condensed_df])
    condensed_combined_df.date = condensed_combined_df.date.apply(dt)
    condensed_combined_df.sort_values(['date'], inplace=True)

    # Remove duplicate dates
    condensed_combined_df = condensed_combined_df[~condensed_combined_df.date.duplicated(keep='first')]
    condensed_combined_df['date'] = condensed_combined_df['date'].astype(str)
    condensed_combined_df.reset_index(inplace=True, drop=True)

    # Filter only required columns
    condensed_combined_df_required = condensed_combined_df[['date', 'negative_score', 'neutral_score', 'positive_score', 'compound_score']]
    return condensed_combined_df_required
```

Figure 40: Combine and condense tweets (*self-composed*)

While the other data can be directly combined into a single data frame, the tweet data is fetched as separate data frames for each date. Therefore, the above script condenses the tweet data into a single data frame by averaging the sentiment scores for each day

Final dataset creation

```python
def create_combined_dataset(
    prices,
    block_reward,
    trends,
    tweet_volume,
    tweets
):
    '''
    Create and clean the final combined dataset
    '''

    exogenous_features = get_exogenous_datasets(
        block_reward,
        trends,
        tweet_volume,
        tweets
    )

    filtered_prices = get_prices(prices)

    combined_df = filtered_prices.copy(deep=True)

    # Combine datasets together and add NaN to empty date rows
    for i in exogenous_features:
        combined_df = pd.merge(
            combined_df,
            i,
            on=['date'],
            how='left'
        )

    # Impute missing values with the respective columns mean
    combined_df.fillna(combined_df.mean(numeric_only=True), inplace=True)
    combined_df['date'] = pd.to_datetime(combined_df['date'])
    return combined_df

def export_data(df):
    '''
    Save data
    '''

    # Store datasets in mongodb for any requirements in production
    df.index = df.index.astype(str)
    df.sort_values(['date'], inplace=True)
    df_dict = df.to_dict('index')
    dataset_db = init_mongodb()
    dataset_db[FINAL_DATASET_COLLECTION].delete_many({})
    dataset_db[FINAL_DATASET_COLLECTION].insert_one(df_dict)
    print('Saved data to MongoDB')
```

Figure 41: Combine all datasets (*self-composed*)

The script above is used to generate the final dataset the model uses. It fetches all the necessary datasets and merges them into a single data frame. A helper function is then applied to remove any columns that are not needed, as determined through correlation testing. Any missing values for each feature of specific dates are imputed using the mean of their respective columns. Finally, the combined dataset is stored in a MongoDB collection to be accessed by the model.

E.7. The forecasting models

```
for i in range(num_models):
  for loss_fn in loss_fns:
    print(f'Model loss: {loss_fn} | model number: {i}')
    model = tf.keras.Sequential([
      tf.keras.layers.Input(
        shape=(window_size)
      ),
      tf.keras.layers.Lambda(
        lambda x: tf.expand_dims(x, axis=1)
      ),
      tf.keras.layers.RNN(
        LTSCell(16),
        time_major=True,
        return_sequences=True
      ),
      tf.keras.layers.LSTM(
        16,
        activation='relu'
      ),
      tf.keras.layers.Dense(
        128,
        kernel_initializer='he_normal',       #Required for the prediction intervals
        activation='relu'
      ),
      tf.keras.layers.Dense(
        128,
        kernel_initializer='he_normal',
        activation='relu'
      ),
      tf.keras.layers.Dense(horizon)
    ])

    model.compile(
      loss=loss_fn,
      optimizer=tf.keras.optimizers.Adam(),
      metrics=['mae', 'mse']
    )

    model.fit(
      dataset_all,
      epochs=num_epochs,
      verbose=0,
    )

    ensemble.append(model)
```

Figure 42: The forecasting model architecture (*self-composed*)

The above code snippet does not possess novel qualities, as it is the conventional approach for building models in TensorFlow and Keras. Upon analysis, the ‘LTSCell’ within an RNN layer can be noticed; this was empowered upon extending the base Layer class. It should be emphasized that an LSTM immediately follows the ‘LTSCell’, as the LTS is susceptible to gradient explosion, a weakness also shared by other neural ODEs and SDEs that require further investigation in the future.

Several models are built and trained using the Adam optimizer (Kingma and Ba, 2017) and added to an ensemble list. The upper-bound, lower-bound, and point forecasts are generated from this list by extracting the highest, lowest, and median forecasts, respectively.

E.8. User interface

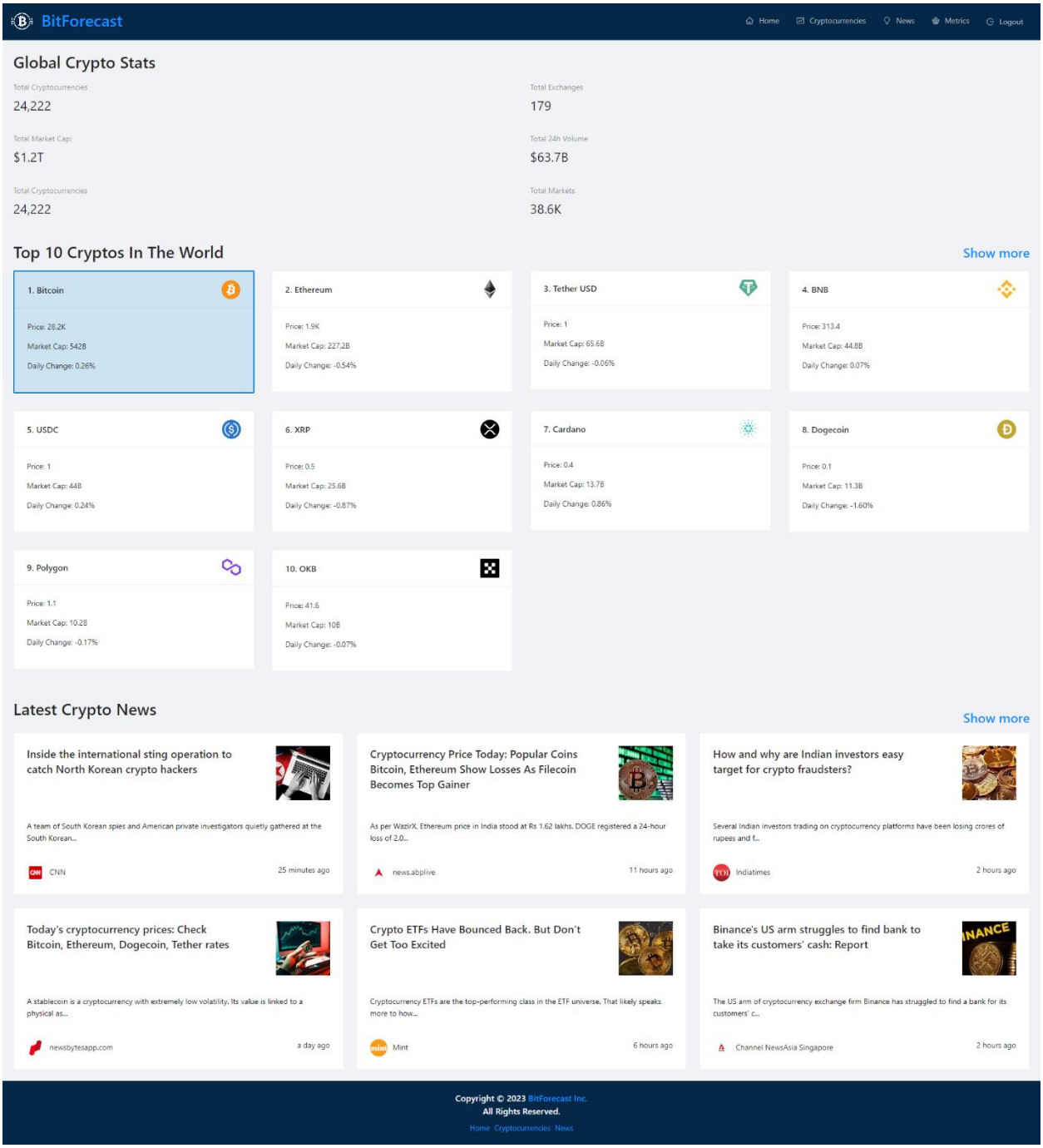


Figure 43: GUI - Home (*self-composed*)

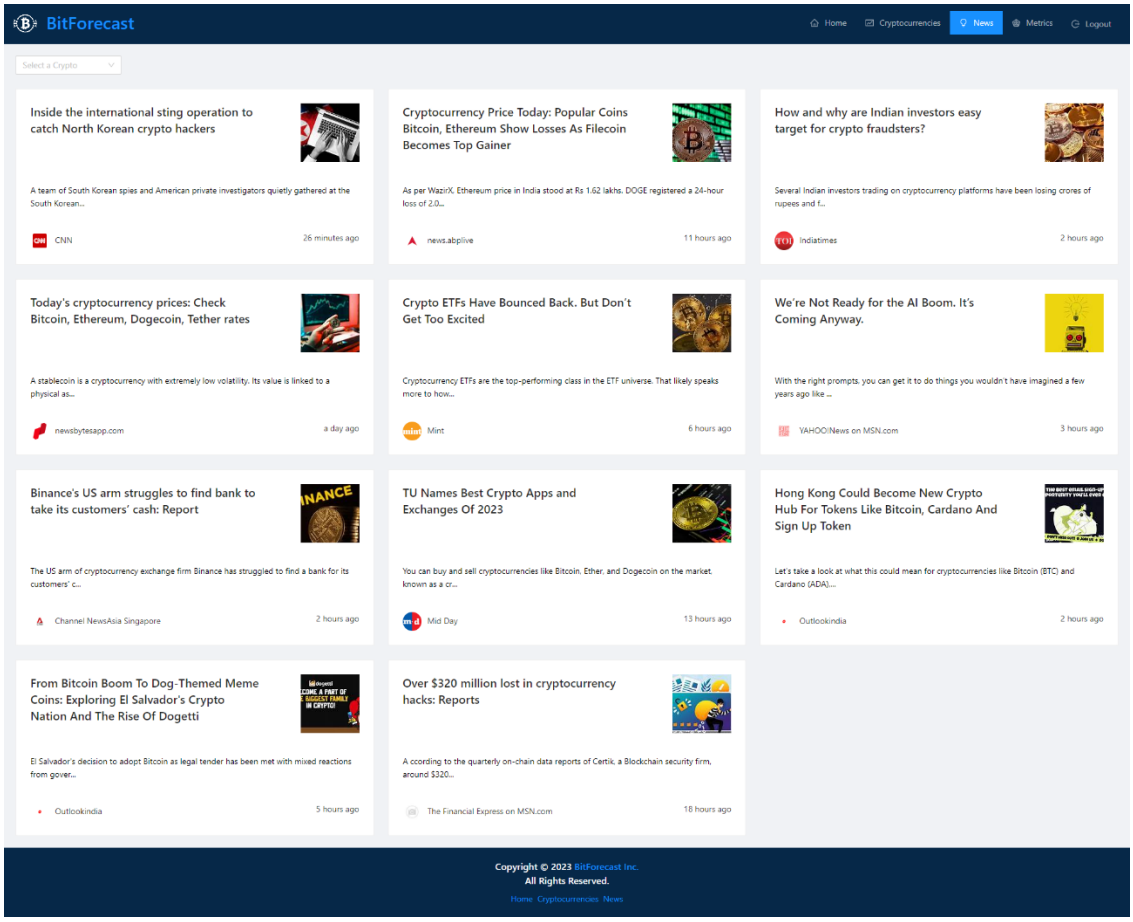


Figure 44: GUI - News (*self-composed*)

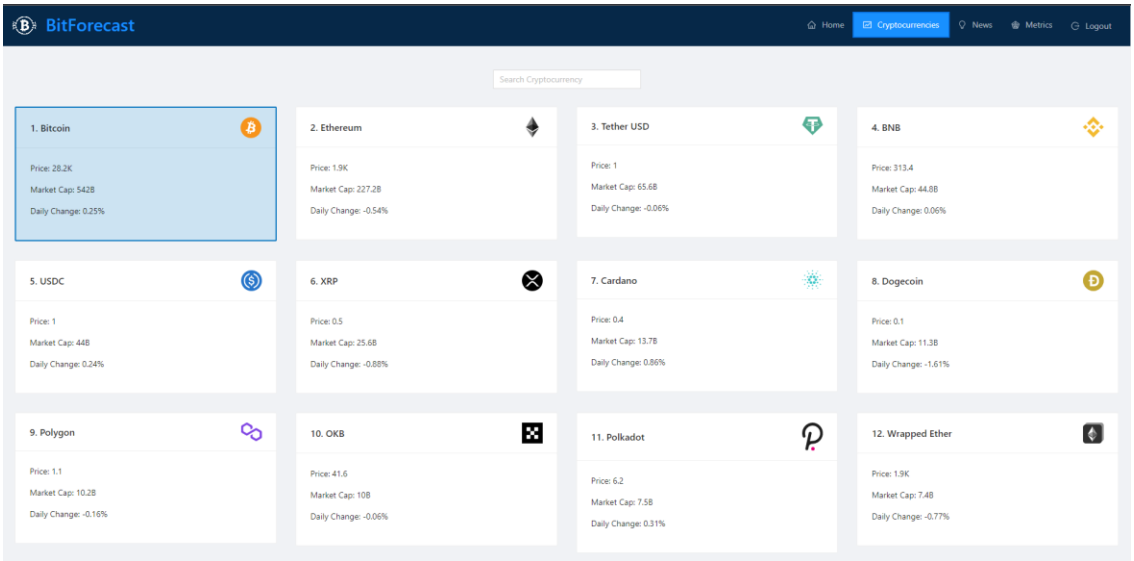


Figure 45: GUI - Cryptocurrencies (*self-composed*)

Figure 46: GUI - Cryptocurrency (*self-composed*)

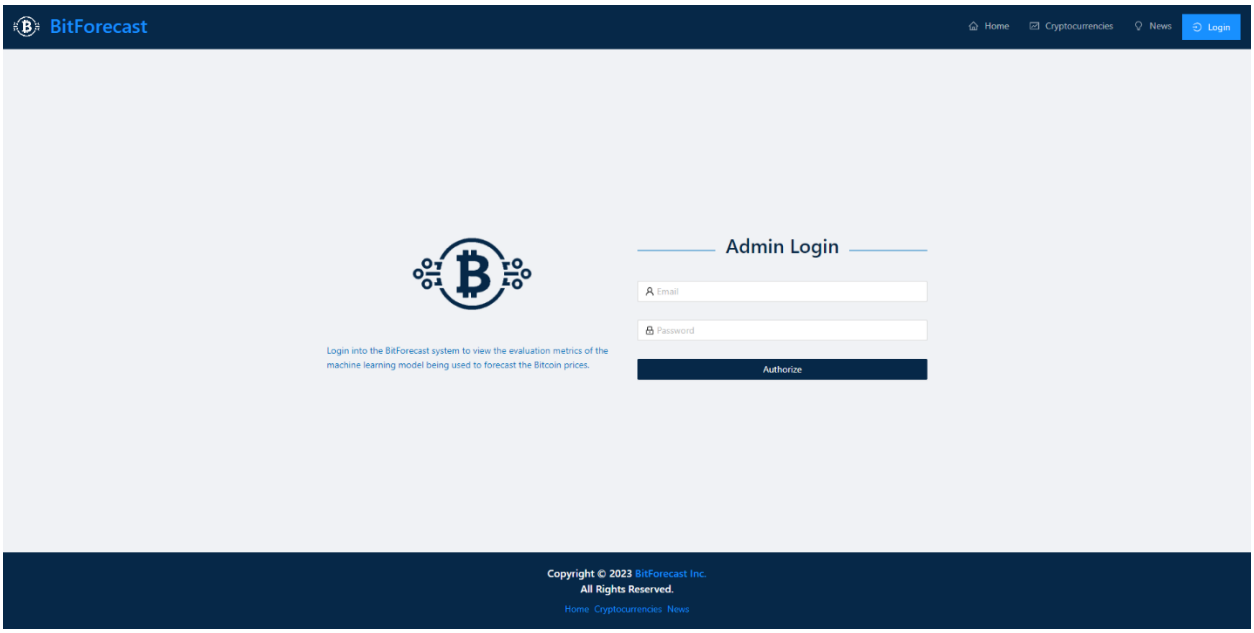


Figure 47: GUI - Admin login (*self-composed*)

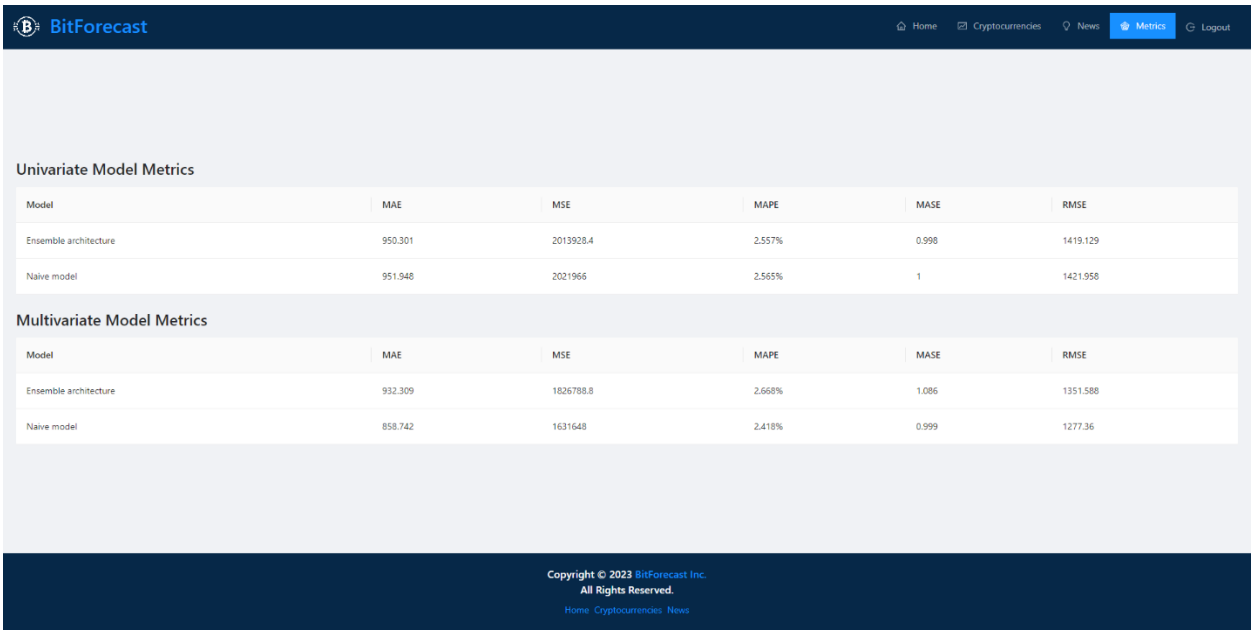


Figure 48: GUI - Admin metrics (*self-composed*)

Figure 49: GUI - Forecast (*self-composed*)

APPENDIX F – TESTING

F.1. Functional testing

Table 48: Functional testing

Test case	FR ID	Action	Expected result	Actual result	Status
Research level					
1	FR1	-	The LTS follows recommended standards to be scalable and built upon.	The architecture was built as a Keras layer, so it handles behind-the-scenes techniques that need to happen.	Passed
2	FR2	-	The LTS can be used as existing layers, such as Conv1D and LSTM.	Building the LTS as a Keras layer provided this functionality.	Passed
System level					
3	FR3, FR4, FR5	Users choose the future dates.	The user can view the prices of the chosen dates.	The deployed endpoint is triggered, and the model's response to the chosen dates is returned to the user.	Passed
4	FR3, FR8, FR9		The user can view the price ranges of the chosen dates.	The deployed endpoint is triggered, and the models' responses to the chosen dates are returned to the user.	Passed

5	FR6, FR7	Cron script is triggered periodically.	The latest data are available is stored in the database.	The exogenous features are scraped/extracted, processed, condensed, combined, and saved into the database.	Passed
6	FR10	Upon receiving the responses, the user views the updated graph.	The graph is updated with the predictions and plotted alongside the past prices.	The GUI is updated with more data points that are future predictions.	Passed
7	FR11	Sentiments extracted from scraped data.	Sentiments are weighted based on influencer scores by the proposed formula.	The sentiments undergo a weighting stage, changing the score based on specific metrics.	Passed
8	FR12	Users choose the future dates.	Explainability insights are provided to the user to justify the forecast further	Not implemented	N/A
9	FR13	Not implemented	Models updated based on the hyperparameter update.	Not implemented	N/A
10	FR14	Admins log into the system.	Technical information about the models is shown.	The evaluation metrics of the two models are displayed.	Passed

F.2. Non-functional testing

The author conducted various testing methods to ensure the system met the non-functional requirements and design goals. These include performance, GUI, security, maintainability, compatibility testing, and several test cases.

Performance testing

The author had deployed the API and model to a server; therefore, there is no requirement to have a high GPU and CPU power machine. Docker, GitHub Actions, and Heroku with basic Dynos were utilized for deployment purposes, capable of serving requests for small-scale applications. However, for large-scale purposes, it is recommended that the Dynos be scaled up, as the application would not handle multiple concurrent requests due to limitations in the deployment environment. It is also worth mentioning that as the system is developed using TensorFlow, initial load times can take some time.

GUI testing

The requirement-gathering phase revealed the critical need for a simple and effective GUI. To ensure that the GUI meets performance and accessibility requirements, Google Lighthouse testing was conducted. The following diagrams depict the test results.

(The rest of this page is intentionally left blank)

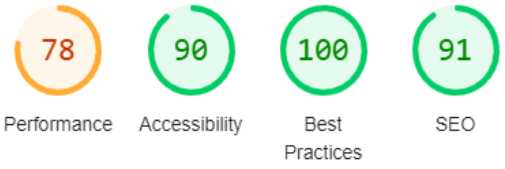


Figure 50: Lighthouse - Home (*self-composed*)

Figure 51: Lighthouse - News (*self-composed*)

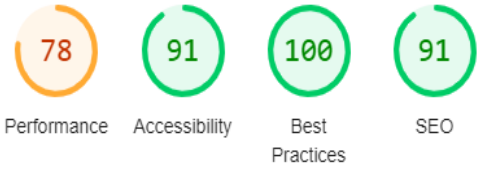


Figure 52: Lighthouse - Cryptocurrencies (*self-composed*)

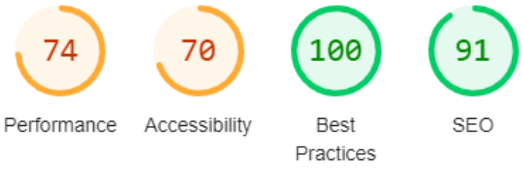


Figure 53: Lighthouse - Cryptocurrency (*self-composed*)

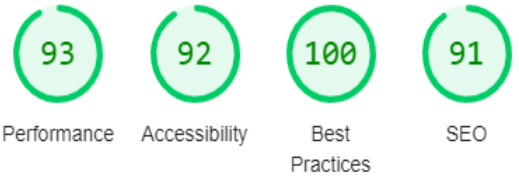


Figure 54: Lighthouse - Admin login (*self-composed*)

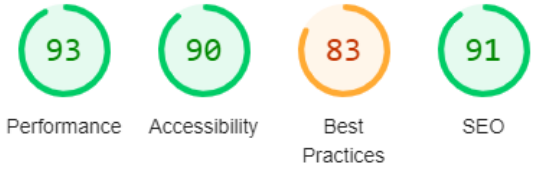


Figure 55: Lighthouse – Admin metrics (*self-composed*)

The results show variation across different pages. This may be due to the use of third-party APIs to render information, which can impact performance. However, the testing process provides valuable insights for improving the GUI's performance and accessibility.

Maintainability testing

Code maintainability was considered an essential factor to ensure that the developed algorithm and the system could be maintained and improved in the future. CodeFactor was utilized to assess the quality of the repositories, ensuring that the code was well-documented and adhered to best practices.

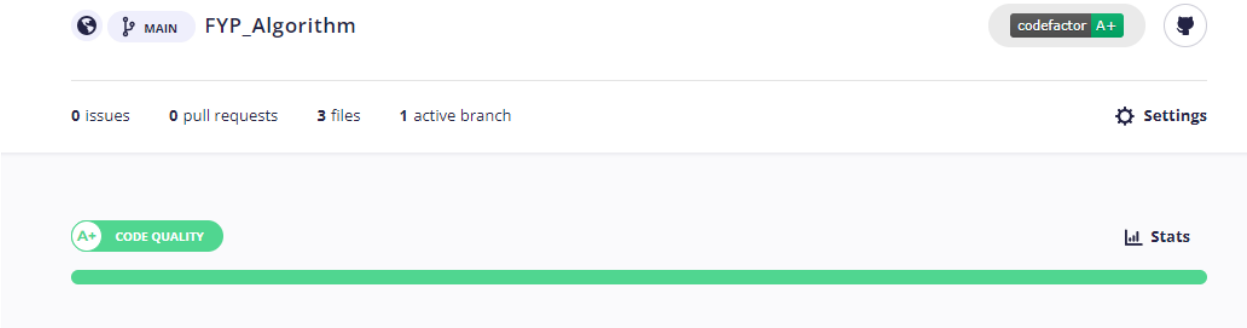


Figure 56: CodeFactor - Algorithm repository (*self-composed*)

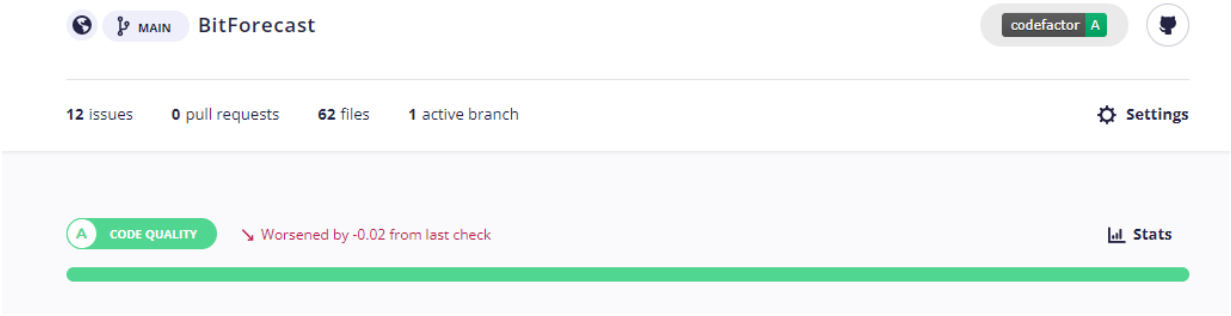


Figure 57: CodeFactor - Application repository (*self-composed*)

Security testing

Despite the absence of sensitive information or data collection from the application, ensuring its security is paramount to prevent any malicious attacks that could jeopardize its availability and integrity. Therefore, the author performed a comprehensive vulnerability analysis on the code repositories to identify and mitigate potential security threats that attackers may exploit. To achieve this, the author utilized CodeQL, a powerful tool that helped detect and resolve any vulnerabilities in the codebase.

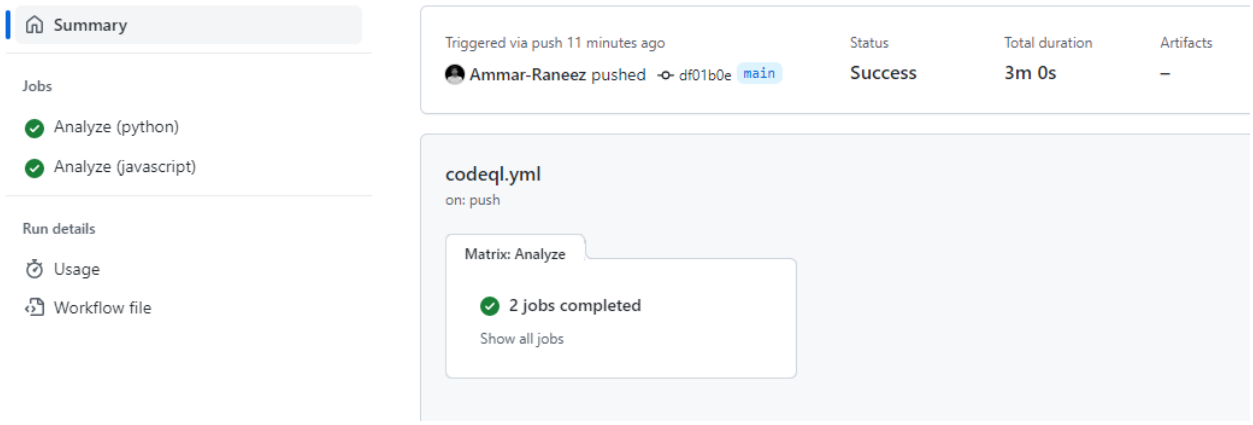


Figure 58: CodeQL - Algorithm repository (*self-composed*)

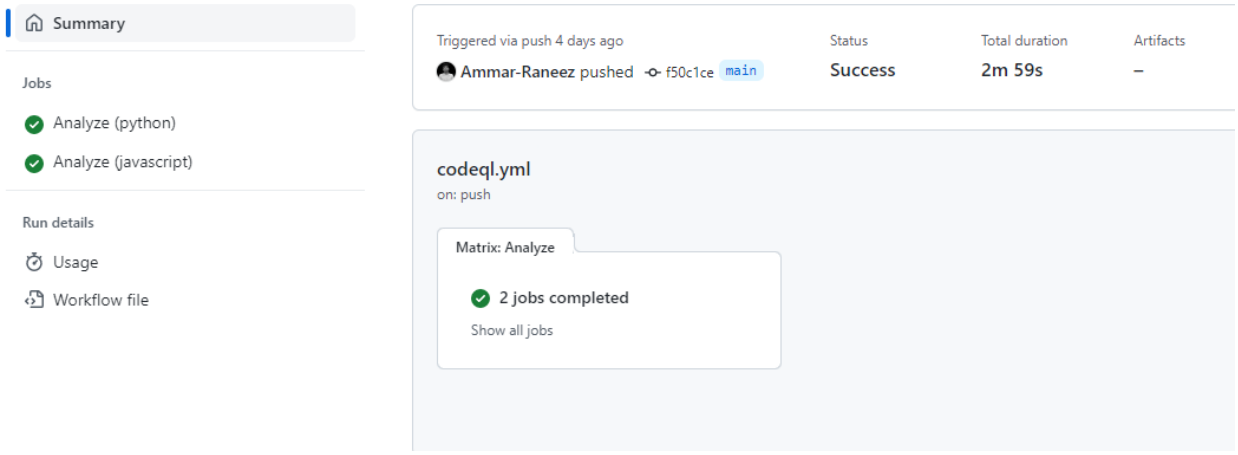


Figure 59: CodeQL - Application repository (*self-composed*)

Personal steps taken to avoid vulnerabilities

As the author utilized AWS, Rapid API, MongoDB, and Firebase, it was essential to ensure that sensitive information, such as secret and access keys and URIs, were securely stored and not exposed in the code repositories. To achieve this, the author used environment and config variables to inject the keys when required while keeping them hidden from the code. This measure was put in place to prevent any potential security breaches and minimize the risk of unforeseen charges or unauthorized database tampering.

Compatibility testing

To ensure effortless accessibility of the application on multiple devices, the author adopted a mobile-first development approach. This approach ensured the application's compatibility with various screen sizes, browsers, and operating systems, resulting in a user-friendly and adaptable interface. The screenshots below demonstrate the application's interface on different devices, validating its responsiveness and adaptability; it may be noticed that the author has chosen to hide the graph on smaller screens, as displaying it on such screens would be impractical and difficult to interpret. The author utilized the browser developer tools to streamline this process.

(The rest of this page is intentionally left blank)

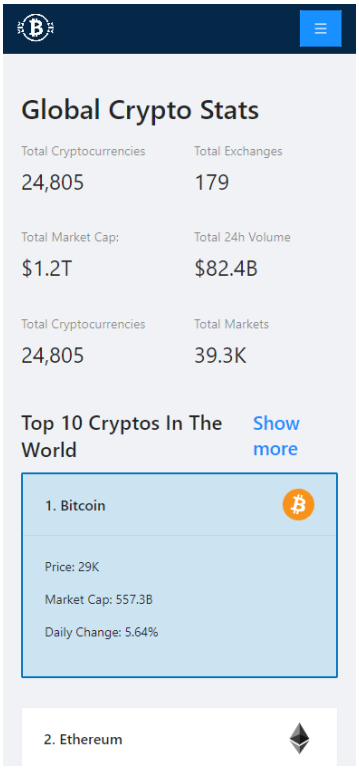


Figure 60: iPhone 12 Pro (*self-composed*)

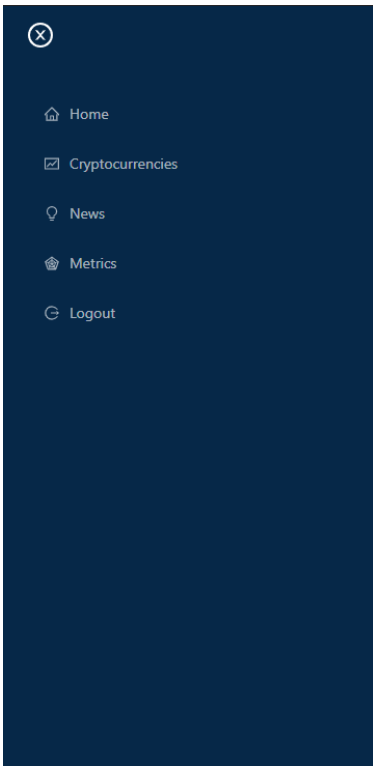


Figure 61: Galaxy S8+ (*self-composed*)

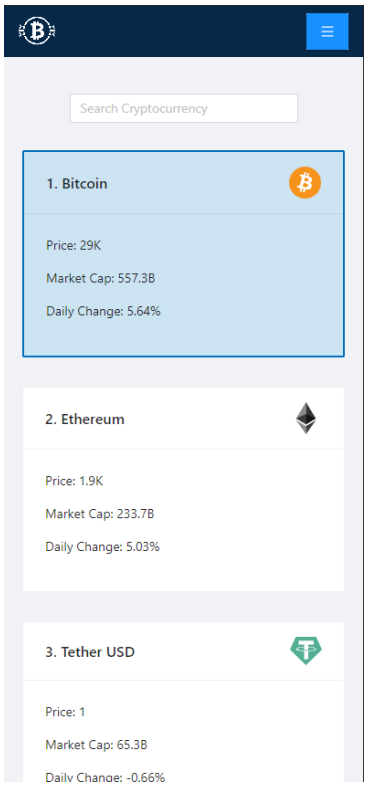


Figure 62: Pixel 5 (*self-composed*)

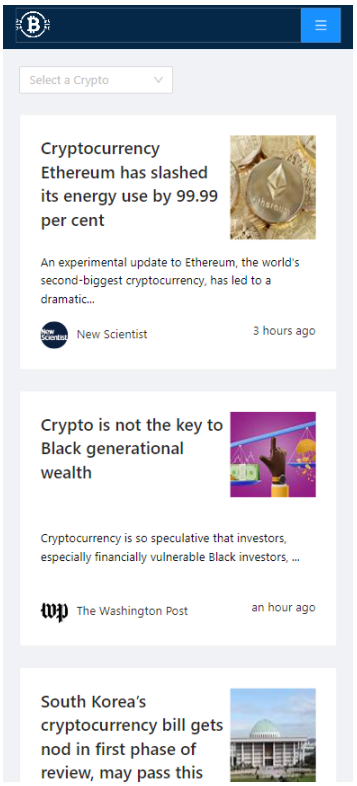


Figure 63: S20 Ultra (*self-composed*)

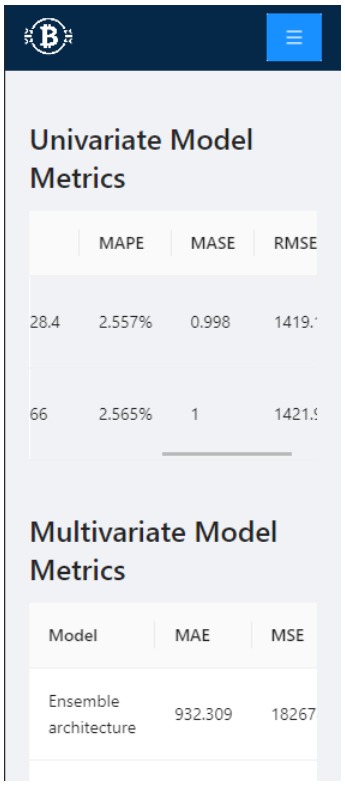


Figure 64: Galaxy Fold (*self-composed*)

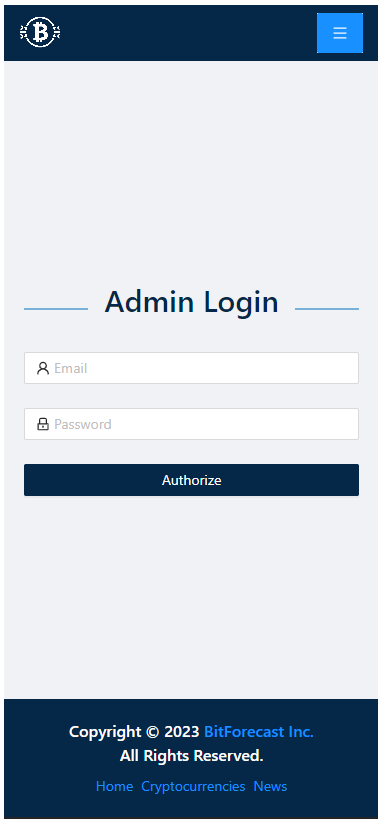


Figure 65: iPhone X (self-composed)

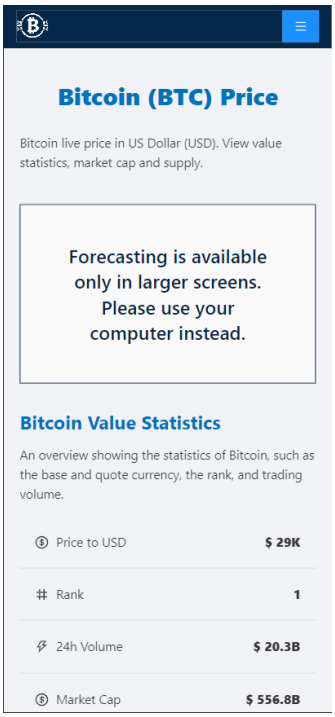


Figure 66: iPhone XR (*self-composed*)

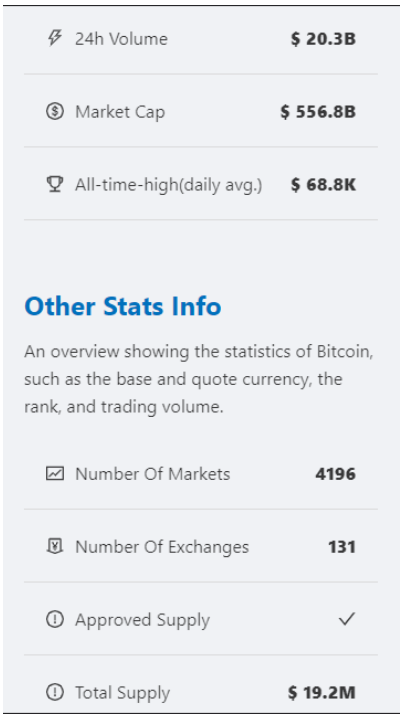


Figure 67: Moto G4 (*self-composed*)

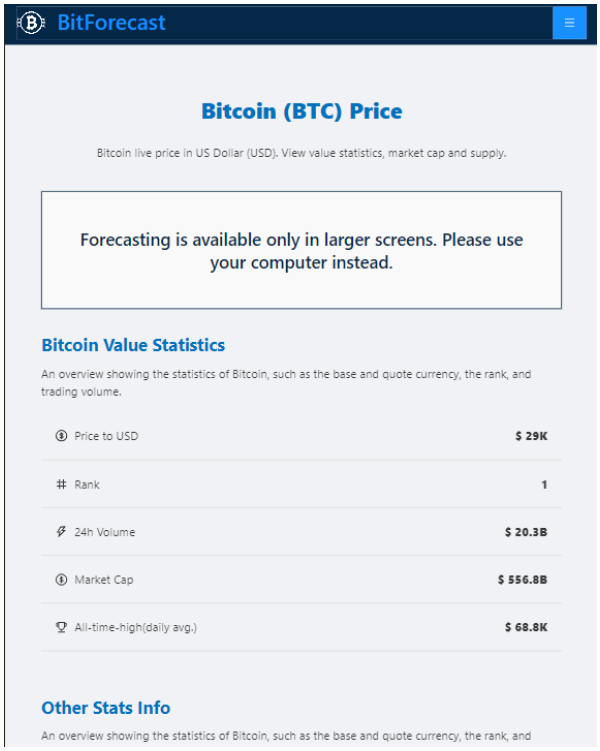


Figure 68: iPad Air (*self-composed*)

Figure 69: Surface Duo (*self-composed*)

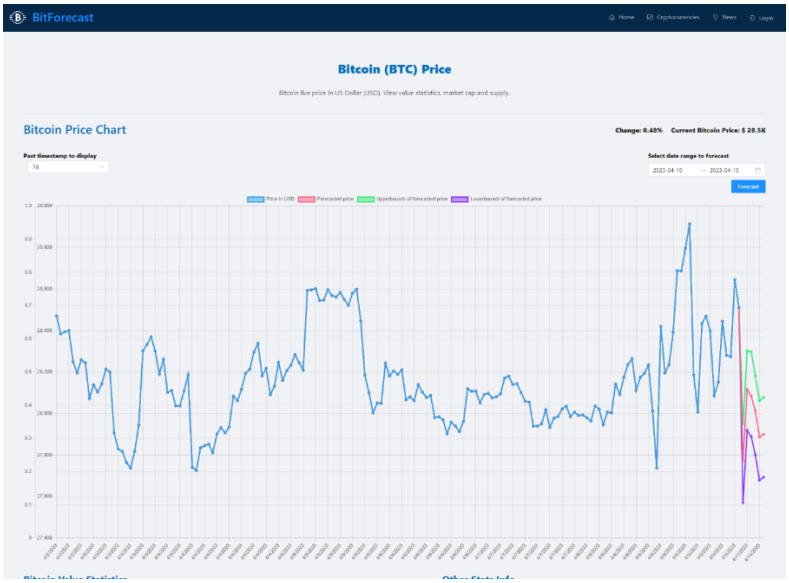


Figure 70: Google Chrome (*self-composed*)

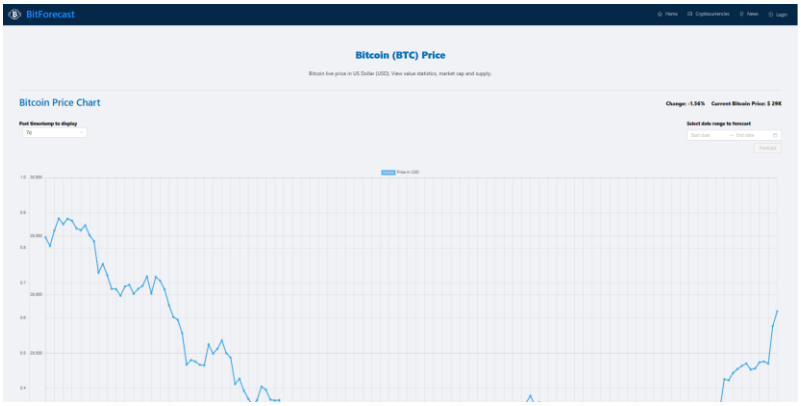


Figure 71: Microsoft Edge (*self-composed*)

Figure 72: Safari (*self-composed*)

Repositories' status

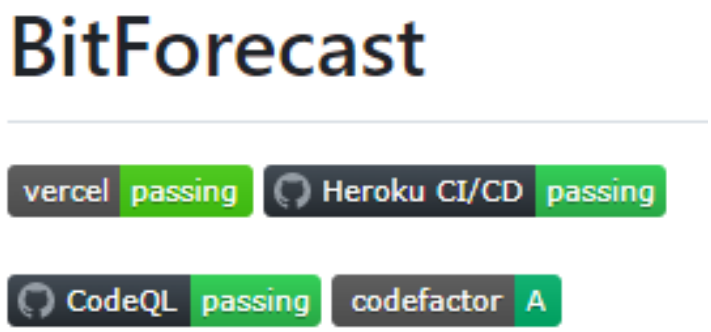


Figure 73: BitForecast repository status (*self-composed*)

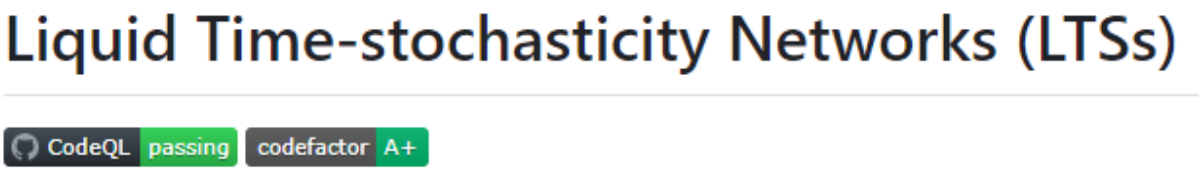


Figure 74: LTS repository status (*self-composed*)

As shown above, *BitForecast* has been deployed to Vercel and Heroku, ensuring accessibility from any device. Additionally, a comprehensive vulnerability analysis was conducted on the repositories to identify and rectify any potential vulnerabilities attackers could exploit. The use of CodeQL facilitated the detection of any such vulnerabilities. Moreover, Codefactor rated the code quality as excellent, which is essential for ensuring maintainability and facilitating future enhancements to the system.

Test cases

Table 49: Non-functional testing

Non-functional requirements		
Test case	**NFR ID**	**Result**
1	NFR1	The system gives responses within 1-2 minutes.
2	NFR2	Only the data for specific dates are scraped and updated whenever necessary.
3	NFR3	The developed GUI is simple, easy to follow, and attractive. Additionally, technical information is displayed only for admins. Validation for this claim was obtained in **Chapter 9**.
4	NFR4	Docstrings were added for each method and comments wherever necessary.
5	NFR5	The responses are plotted within the graph to show a growth/decline.
6	NFR6	The system was deployed to Heroku by Docker and GitHub Actions but will not scale as traffic increases: the utilized Dynos are the most basic version.
7	NFR7	The model is stored in AWS S3 for backup and requests, the data is stored in MongoDB, and Firebase handles admin authentication. Therefore, security is integrated to some extent.
8	NFR8	The application is responsive across a wide range of screen sizes to an extent.
Design goals		
9	DG1	The data are stored in MongoDB and fetched whenever necessary. When the available data is not up-to-date, the data fetcher script fetches data only for the missing dates and updates the database.
10	DG2	The system is built to be as user-friendly as possible, with zero information on any technical jargon. Validation for this claim was obtained in **Chapter 9**.

11	DG3	The forecast is plotted alongside the existing price chart using a different color to differentiate between them. Two more lines are plotted to demonstrate the uncertainty estimations / upper and lower bound prices that display the range of prices.
12	DG4	Docstrings were added for each method and comments wherever necessary. Analysis using Codefactor produced a grade of A+ for the algorithm repository, which is the maximum grade possible.

APPENDIX G – EVALUATION

G.1. Expert evaluators & feedback

Table 50: Selected expert evaluator details

EVAL ID	CAT ID	Affiliation	Expertise related to the research
Research domain			
EV1	CAT1 CAT2	Google Brain visiting researcher and Associate Professor at University of Toronto.	Neural ODEs and SDEs.
EV2	CAT1 CAT2	Professor at the University of Melbourne.	Artificial Intelligence.
EV3	CAT2	Professor at the University of Colombo.	Artificial Intelligence.
Problem domain			
EV4	CAT3	Blockchain and Web 3.0 developer	Blockchain and cryptocurrencies.
EV5	CAT3	Medical doctor & crypto evangelist	Cryptocurrencies and crypto exchanges.
EV6	CAT3	Avid BTC trader and statistician	Cryptocurrencies, market trading and statistics.
EV7	CAT3	Prefer not to say	Cryptocurrencies and market trading.

Table 51: Evaluator feedback

EVAL ID	Feedback
Research domain	
EV1	*'The LTS architecture seems to be more robust and stable than the LTC. It is great to see the researcher picking a domain which I myself am working on. I consider the contribution to be respectable and significant especially since this is for an undergraduate project. The difficulty is up there as it's a novel neural network therefore different computation and considerations must be taken. However, the technique of back propagation is a concern whether it is scalable due to it being memory and time intensive. It is true that till the domain of NODEs came time series was stagnant therefore it is evident that this algorithm is a stepping stone and definitely in the right direction for breaking these limitations. I suggest to include a hybrid SDE solver instead of the implicit solver to achieve more stability and efficiency in forward propagation.'*
EV2	*'The researcher contacted me initially during his requirement gathering phase to obtain certain insights and this research domain seems to be quite interesting and peculiar as it's not something that you see every day like for instance NLP and computer vision. The research gap identified is well informed and significant as not everyone will come up with a new algorithm that would solve a demanding task. This research is quite difficult as the researcher had to implement a lot of the work by themselves starting from designing the algorithm to determining the optimal way of performing forward and backward propagation. The novelty of the project is well justified as till date there is very few research in this domain aiming to solve such intricate problems; it is great to see that he took up such research for their undergraduate program.'*
EV3	*'Chosen Research Domain is very good. The student has identified a research gap and introduced a new method for that. There were technical difficulties but he has overcome the problems. He has discovered a new technique to overcome issues with LTS demonstrated it better than LSTM, GRU methods for time series*

	forecasting. He has developed a new layer to be used in the Keras machine learning library. Need to evaluate the results further. He has a new technique using LTS for time series forecasting. Has followed a good research approach.
Problem domain	
EV4	*'Yea, the factors chosen to predict the price was well justified. The usage of influencer formula supports the accuracy, since these are being determined based on the real-world value. As a web3 dev and a crypto enthusiastic, the user experience was understandable. Implementing trading curves and volumes would allow amateur to determine the market performance of the coin. Yea, since predicting the price to point is nearly impossible. The range of prices given would give an understanding of the currency's performance. The application has come up with a solution that can be used as a trump card for crypto investors.'*
EV5	*'Yes, he has. He has provided ample detail of the multiple possible external factors that may affect the course of cryptocurrencies, and bitcoin in particular, justifying his cause. Significantly accurate, as it takes into account the multiple extraneous factors that have a role, and assign them particular weightage, which makes it statistically possible to evaluate the effect of the tweet on the course of BTC. The application is very user friendly, with the interface being very simple and straightforward, without several jargon which makes it easier for first time users to have a more wholesome understanding of cryptocurrencies. Majority of the technical jargon and concepts are not visible to the amateur who is looking into the initial steps of investing, and the interface is very user friendly. Yes, he has. The idea behind the range of prices is straightforward as it is relatively impossible to predict a single set value of the future of bitcoin but a predicted range seems far more reasonable and probable. Overall, I think the application is well built, with the convenience of newcomers into the crypto-space in mind, and overall, well thought out using factors like Google trends and tweets into the equation, making sure the range of values are as close to realistic as possible. The news section*

	provides sufficient information about the newest in the industry, making sure investors are informed of any sudden proceedings.'
EV6	*'Yes, he has, crypto currently is significantly volatile and does not revolve around a single factor, therefore, taking into account as many factors as possible makes the forecasting significantly more accurate. The weightage is significantly accurate taking account recent tweets and activities of influential people in the field and its relation with the prices of particular crypto currency. However, the significance of the tweeter in the crypto setting needs to be taken into account. Comparably to other applications, this is very user friendly as the other applications are very complicated to beginners, and may scare traders away, however this interface is more welcoming and appealing to amateurs and first-time investors. Yes, it can as its very straightforward, uncomplicated and user friendly. It avoids throwing too much information at the user right as they enter to the landing page. The minimalistic design stands out in comparison than to more well-known crypto resources. Yes, he has. The range of values with upper and lower bounds gives the user a ballpark range as to what to expect to avoid following a single predicted value as crypto currencies are volatile albeit predictable using the suggested algorithm. It is very user friendly, straightforward, extremely easy to use for a beginner. The predicted ranges would help the traders to make a more informed decision.'*
EV7	*'Yes, he has. it makes it very clear and adding a new section regarding the latest updates about crypto is extremely useful too when it comes to making a decision. I think it's quite accurate, seems like all the relevant factors have been considered. it seems to be very user-friendly. and convenient to use. application seems very straightforward. can definitely be used by an amateur, especially since they can inform themselves with latest updates within the app itself, instead of separately searching for it, which can save a lot of time. yes. since the market fluctuates so often, giving a range seems to be the best way to predict an outcome, instead of 1 fixed price, and there's a less margin of error. I think that it's a very beneficial application. it's very convenient and will certainly help prevent many losses.'*

G.2. Evaluation of functional requirements

Table 52: Evaluation of the implementation of functional requirements

FR ID	Description	Priority	Use Case	Evaluation
Research level				
FR1	A robust and scalable implementation of the LTS must follow recommended standards.	M	-	Implemented
FR2	The developed algorithm must be capable of being used as existing layers and algorithms (ex: LSTM, CNN).	M	-	Implemented
System level				
FR3	Users must be able to choose a future date.	M	UC:01	Implemented
FR4	Users must be able to view the point prediction price.	M	UC:01	Implemented
FR5	The system must generate the point prediction price based on the user's choice of date.	M	UC:02	Implemented
FR6	The script must obtain the latest data available periodically.	M	UC:03	Implemented
FR7	The script must extract trends and sentiments from obtained data.	M	UC:04	Implemented
FR8	Users should be able to view a range of prices along with the single-point price.	S	UC:01	Implemented
FR9	The system should generate higher and lower bound uncertainty estimations.	S	UC:02	Implemented
FR10	The GUI should plot the forecast with the current prices in a single graph to show the growth/decline.	S	UC:01	Implemented
FR11	The script could weigh sentiment based on tweets from influential individuals in the cryptocurrency space.	C	UC:05	Implemented

FR12	The system could display some insights to the user, such as a highly influential tweet that made it predict the price.	C	UC:01	Not-considered
FR13	Admins could authenticate and update the model with different parameters.	C	UC:06	Not-considered
FR14	Admins could get additional information about a prediction, such as the evaluation metric and accuracy.	C	UC:07	Implemented
FR15	The system will not produce forecasts for other cryptocurrencies.	W	N/A	Not-considered
FR16	The system will not produce real-time forecasts (ex: hourly).	W	N/A	Not-considered
Functional requirement completion percentage = $\frac{12}{16} * 100 = 75\%$				

G.3. Evaluation of non-functional requirements

Table 53: Evaluation of the implementation of non-functional requirements

NFR ID	Requirement	Description	Priority	Evaluation
NFR1	Performance	The system should be optimized to generate forecasts efficiently, even when utilizing additional features.	Important	Implemented
NFR2	Performance	The system should update its data only when necessary to avoid unnecessary overhead.	Important	Implemented
NFR3	Usability	The user interface should be intuitive providing clear and concise error messages in case of any issues	Important	Implemented
NFR4	Maintainability	The codebase should be thoroughly documented to ensure future maintainability, especially for the algorithm development repository.	Important	Implemented

NFR5	Quality	The system output should be high-quality, providing valuable insights to users.	Desirable	Implemented
NFR6	Scalability	The system should be deployed to a scalable cloud environment with ample resources to handle multiple concurrent user requests.	Desirable	Implemented (~50%)
NFR7	Security	The system should be designed with security to prevent data manipulation and protect against attackers, especially to avoid tampering with the used datasets.	Desirable	Implemented
NFR8	Compatibility	Compatibility testing should be performed on browsers and mobile devices to ensure a smooth user experience.	Desirable	Implemented
NFR9	Availability	A dedicated primary operator should be available to handle critical issues promptly.	Desirable	Not-implemented
Non-functional requirement completion percentage = $\frac{7.5}{9} * 100 = 83.3\%$				

Table 54: Evaluation of the achievement of design goals

DG ID	Goal	Evaluation
DG1	Performance	Achieved
DG2	Usability	Achieved
DG3	Quality	Achieved
DG4	Maintainability	Achieved
Design goals achievement percentage $\frac{4}{4} * 100 = 100\%$		

APPENDIX H – CONCLUSION

H.1. Status of research objectives

Table 55: Status of research objectives

Objective	Description	Status
Problem Identification	Document and comprehend the identified problem and its novelty. **RO1**: Research on a domain the author is interested in and identify a comprehensive enough issue that requires solving. **RO2**: Delve deeper into the identified problem to obtain a general understanding of how to solve the problem. **RO3**: Split the primary problem into manageable subproblems so it is easier to digest and solve one section at a time. **RO4**: Design a schedule, associated deliverables, and the Gantt chart to follow in this research.	Completed
Literature Review	Collate relevant information by reading, understanding, and evaluating previous work. **RO5**: Conduct preliminary studies and investigations on existing TS forecasting systems and algorithms. **RO6**: Analyze the requirement for specialized TS algorithms. **RO7**: Research on neural ODEs, LTCs & SDEs. **RO8**: Obtain deep insights into the architecture behind the LTC. **RO9**: Research and obtain insights on factors affecting the price of BTC. **RO10**: Research existing BTC forecasting and related open market systems. **RO11**: Research on necessary ML techniques and evaluation approaches.	Completed
Requirement Elicitation	Collect and analyze project requirements using appropriate tools and techniques.	Completed

	RO12: Analyze stakeholders and understand their viewpoints and concerns. **RO13**: Gather the requirements and architectures of neural ODEs, LTCs, and SDEs. **RO14**: Collate the most up-to-date details of BTC and obtain insights into the end users' perspectives. **RO15**: Design the use case and context diagrams to justify the product's specifications.	
Design	Design the architecture and system to solve the identified problems effectively. **RO16**: Design diagrams to grasp the algorithm. **RO17**: Design the LTS formula and analyze its complexities. **RO18**: Choose appropriate techniques for forward and backward propagation of the LTS. **RO19**: Design diagrams to understand the supplementary system being developed. **RO20**: Design a deployment pipeline for seamless deployments.	Completed
Implementation	Implement a system that effectively addresses the research gaps. **RO21**: Design, evaluate and select the technologies best suited for the implementation. **RO22**: Develop an efficient LTC implementation. **RO23**: Build on the LTC to implement the LTS. **RO24**: Integrate the developed algorithm into a TS forecasting application. **RO25**: Incorporate the intelligent system into a client application to display forecasts. **RO26**: Design and implement an automated flow to update the built network with the latest data. **RO27**: Consider any legal, social, ethical, and professional issues upon implementation.	Completed

Evaluation	Effectively test the implemented algorithm, system, and data science model using recommended techniques. **RO28**: Evaluate the developed algorithm and the respective model against the predefined evaluation metrics. **RO29**: Create a test plan and cases and perform functional, non-functional, and integration testing. **RO30**: Solicit feedback from domain and tech experts to understand limitations and future work.	Completed
Documentation	Document the progress of the research project and report any encountered challenges. **RO31**: Produce a comprehensive report of new skills acquired and contributions made, ensuring the aforementioned objectives are achieved. **RO32**: Publish a research paper outlining the LTS architecture to provide validation.	Completed

H.2. Utilization of knowledge from the degree

Table 56: Knowledge utilized from the degree

Modules	**Knowledge utilized**
Mathematics for Computing	This module provided in-depth mathematical knowledge required to build the algorithmic formula. It laid the foundation for learning more advanced mathematical concepts.
Object Oriented Programming	Experience in creating full-stack applications, which are useful in the real world, was gained through this module. Building a backend and integrating an API with a client were essential skills to create the MVP.
Software Development Group Project	This module motivated the author to undergo a challenging project. From pitching an idea to presenting the final product in

	competitions, it laid the groundwork for future research, design, development, and evaluation.
Algorithms: Theory Design and Implementation	The knowledge gained from this module was paramount when analyzing the performance and complexities of the developed algorithm.

H.3. Achievement of learning outcomes

Table 57: Achievement of learning outcomes

Description	**Learning outcome(s)**
The project was divided into two subproblems: algorithm implementation and application development. They were then further broken down into digestible units to understand and implement one at a time by applying appropriate techniques recommended by analyzing literature and requirements.	LO1
The units identified were placed into a Gantt chart as milestones to achieve within a set timeframe and complete the project within the given timescale.	LO2
Requirements were gathered from academic researchers and end users to obtain insights into developing the LTS algorithm and prioritize the features necessary for the BTC forecasting application.	LO3
Literature was read and analyzed, including both recent and some over a century old, to gain a thorough understanding of the mathematical concepts.	LO4
After gathering requirements and acquiring necessary insights and knowledge, the author divided the project into two subproblems: algorithm implementation and application development. These subproblems were further broken down into manageable units, and appropriate techniques were applied to implement one unit at a time. The author regularly met with the supervisor throughout the	LO5, LO6, LO7

implementation process to ensure that milestone deliverables were being produced. Additionally, any SLEP issues were considered and documented.	
The research progress was documented and presented to the supervisor as milestone deliverables, with each chapter being updated based on feedback from the supervisor and module leader. Two document artefacts, including the project proposal and PSDP, were submitted to ensure the project's success. Moreover, the author presented a paper at a conference to justify their proposed LTS architecture.	LO8

H.4. Outcome of the proposed research questions

Table 58: Outcome of the research questions

Research Question	Conclusion
RQ1	Neural ODEs, SDEs & the LTC were studied thoroughly to obtain insights into implementing the LTS. Although these algorithms have not been used in TS forecasting, the underlying architecture had to be thoroughly assessed, as implied in previous chapters. Therefore, the LTS is a novel approach to TS forecasting that builds on insights gained from the study of these algorithms.
RQ2	The algorithm was designed as a custom Keras layer, enabling it to be used as existing layers and generalized and ready to be integrated into the Keras library. Challenges ranged from implementing the LTS to obtaining Tweets, as discussed in previous sections; these challenges were mitigated through a meticulous literature review and a rigorous analysis of libraries and code repositories.
RQ3	The contributions are broken down in detail in **Chapter 10**.
RQ4	The Twitter sentiment, Twitter volume, block reward size, historical closing price, and Google Trends were considered to construct the model. The volume, open, high, and low features available in historical prices were not included as correlation tests resulted in a high correlation

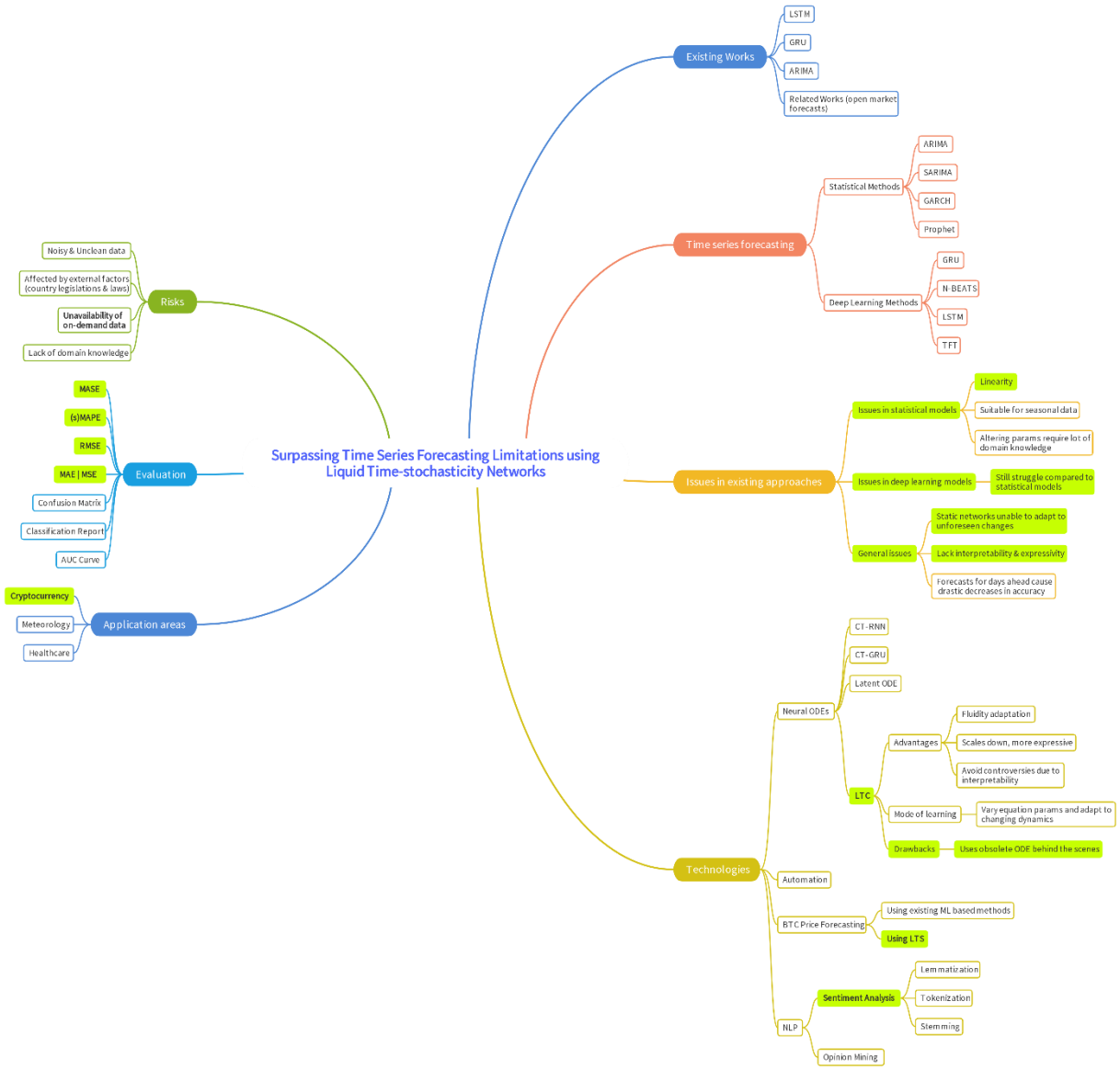
	between them and the closing price. Therefore, they would not bring additional information, making the model more complex. While other factors such as Facebook, Fiat currency, Reddit, and other cryptocurrencies could have an impact, they were not considered in this research. However, analyzing their impact and incorporating them into a more comprehensive dataset could be an area for future research.

H.5. Implementation code

The code written and associated research work conducted is in GitHub for convenience.

- Algorithm trials and testing repository – **`http://bit.ly/3HYOqBB`**.
- Application implementation repository - **`http://bit.ly/3HXDtQu`**.
- Research documentation repository - **`http://bit.ly/3jxjf6V`**.

APPENDIX I – CONCEPT MAP



A clearer version can be found [Here]

APPENDIX J – GANTT CHART

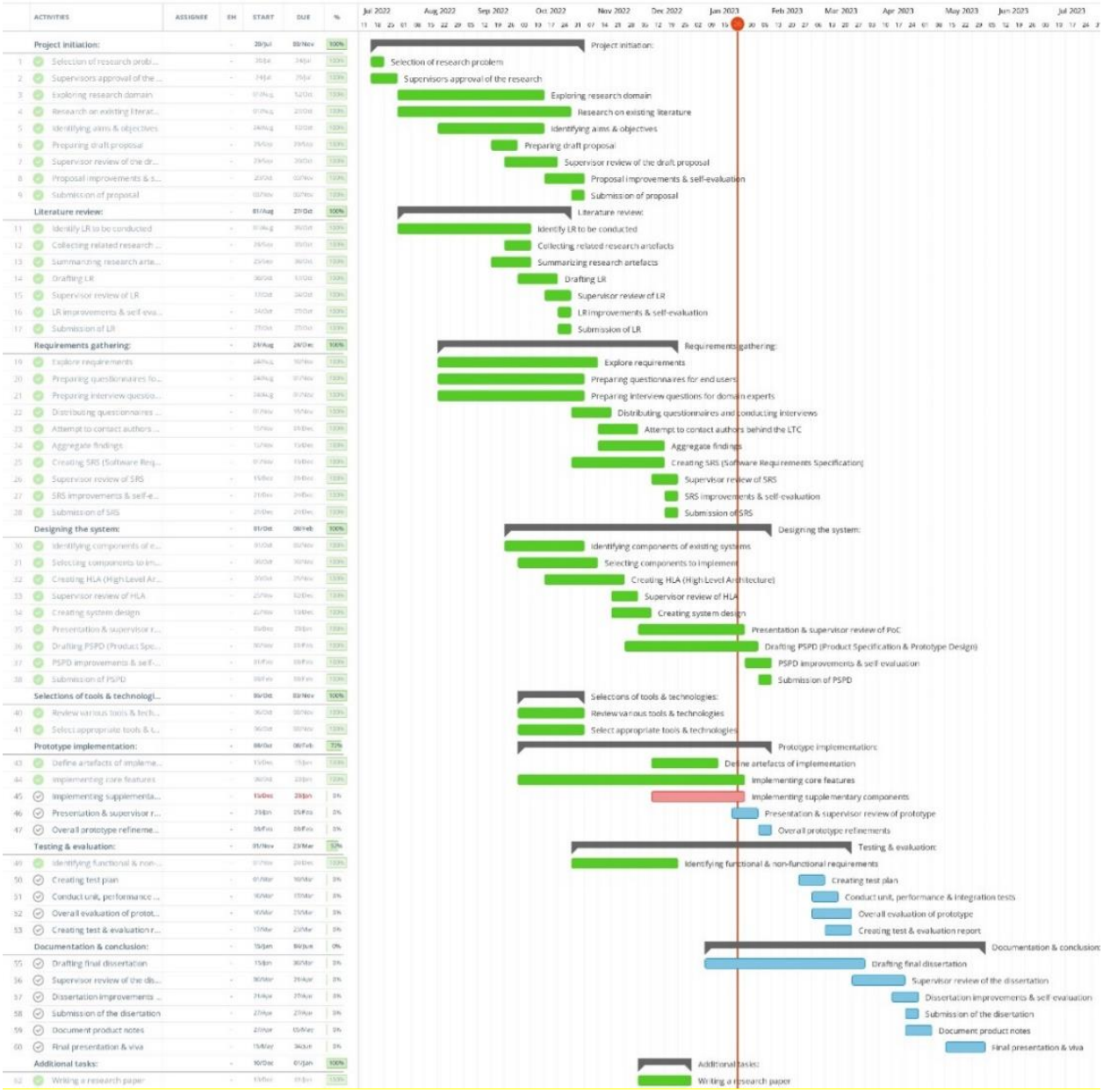


Figure 75: Gantt chart (*self-composed*)

A clearer version can be found Here

APPENDIX K – EXTENDED REVIEW PAPER

K.1. Acceptance notification

Registration instructions for: IEEE CCWC 2023 Inbox ×

[redacted] 2:46 PM (16 minutes ago)

Hi authors,

Congratulations, Well done! Your paper has been accepted for virtual presentation at the **IEEE CCWC 2023 which will be held during 8-11 March 2023.**

Please refer to the below instructions for registration and full-paper submission to the conference. **Check the https://ieee-ccwc.org/registration/ page for the amount details. The early bird registration date is 21st February 2023.**

The required steps are listed below. Follow the steps.

- Register for IEEE CCWC 2023. Registration can be done at EDAS via https://edas.info/listConferencesRegister.php?c=30482. Note that registration is completed if only payment is sent via EDAS. Registration is a prerequisite for the next steps, so please register as soon as possible. **For a single accepted paper, one author registration of 200$ (for IEEE members) and 250$ (for Non-IEEE members) is required.**
- Authors Have To Complete The IEEE Copyright Process Through The "Copyright" Column from EDAS. This Must Be Completed For Your Paper To Be Included In The Conference. The scanned copyright form is not allowed.
- Upload the **camera-ready/final paper** to EDAS. Please check the similarity score **(must be within 15%)** of the paper prior to uploading **and please follow the proper IEEE paper format.**
- **Ensure the appropriate copyright from https://ieee-ccwc.org/submission/ is added to the bottom of the first page of the camera-ready paper. Details about using PDF Express can be obtained from the Submissions page.** The code for the conference (PDF eXpress) id is **57344X. The final paper should be IEEE PDF eXpress certified and written with the Copyright number. The last date is 22nd February 2023.**
- Upload your **video PPT presentation** to EDAS. This is only for backup purposes. (Regarding the mode of the presentation given on the submission page).

The final uploaded manuscript will be sent to IEEE for publication. So you are requested to abide by all the necessary guidelines as stated above. Please contact us for any clarification or any assistance related to registration (also can contact EDAS help). Cooperation in this regard is highly solicited.

Thanks and Regards,

[redacted]

Technical Co-Chair, IEEE CCWC 2023

K.2. Extended review paper

A Review On Breaking the Limits of Time Series Forecasting Algorithms

Ammar Raneez
Computer Science and Engineering
University of Westminster
London, UK
ammarraneez@gmail.com

Torin Wirasingha
Department of Computing
Informatics Institute of Technology
Colombo, Sri Lanka
torin.w@iit.ac.lk

***Abstract*—Time Series (TS) forecasting has stagnated owing to algorithm restrictions, therefore systems developed using these methods can only perform so well. TS remains a challenge despite recent advances in Deep Learning (DL) in Natural Language Processing (NLP) and Reinforcement Learning (RL). This paper reviews the literature on these algorithms, highlights studies using them, and shows their limits. Neural Ordinary Differential Equations (NODEs) with continuous-time and continuous-depth tackle TS forecasting issues. Liquid Time-Constant (LTC) networks, a more advanced and reliable implementation of these NODEs, provides fluidity. We propose a new design that uses the LTC's liquid adaptability and is more adaptable to manage immediate changes. These algorithms are more steady, adaptive, and versatile than DL, which may help overcome its TS forecasting shortcomings.**

Keywords*—*Time Series (TS) Forecasting, Neural Ordinary Differential Equations (NODEs), Liquid Time-constant Networks (LTCs), Ordinary Differential Equations (ODEs), Stochastic Differential Equations (SDEs)

I. Introduction

In this ever-changing world, the need for accurate forecasting has become more and more demanding. Time series (TS) forecasting has been studied for several decades, even before Artificial Intelligence (AI) became popular. Various approaches have been proposed throughout the years to produce effective forecasts; however, most have become obsolete.

Until recently, upon introducing neural ordinary differential equations (NODEs) [21], systems have all been utilizing these traditional and obsolete approaches for their forecasting requirements. This new family of Deep Learning (DL) algorithms has demonstrated promising results in their application in TS forecasting, piquing many academic researchers' interest in deviating from the traditional Deep Neural Network (DNN) architectures to implement more flexible, adaptable, and efficient models.

Although these models have produced great accomplishments, it has been noticed that they could be more stable and fail in modeling uncertainties [22]. Attempts were made to solve stability issues, the most influential and groundbreaking being the Liquid Time-Constant Network (LTC) [23]. However, the issue of modeling uncertainty still lingers.

In this paper, we initially delve into the vast ocean of TS forecasting algorithms and conduct a summarized review of the most popular ones. We then propose a novel architecture inspired by the LTC that could solve the issue of modeling uncertainty while also being robust and stable.

II. Background

A. Time Series Forecasting

Many academics have been drawn to TS forecasting to provide a reliable, scalable, and practical solution. It may be extremely useful in solving various real-world issues, such as forecasting the weather, traffic, or patient EEGs in medical monitoring [1]. Any problem with a temporal component can be considered a potential domain for applying TS forecasting, which is one of many reasons for so much demand and significant impact on business.

TS forecasting is a significant business issue and an area where Machine Learning (ML) could create an impact. It is the foundation for contemporary business practices, including pivotal domains like customer management, inventory control, marketing, and finance. As a result, it has a comprehensive financial impact, with millions of dollars for subtle improvements in forecasting accuracy [2].

Until recent advancements in ML, traditional statistical models have been used with the help of domain expertise. However, with increasingly available data and computing power, ML has become vital in creating forecasting models for the next generation [3]. ML provides a means of learning temporal dynamics in a data-driven manner compared to their traditional counterparts [4].

In particular, DL has gained tremendous popularity in recent times for its remarkable accomplishments in image classification, Natural Language Processing (NLP), and Reinforcement Learning (RL), as it can learn representations from complex data without needing explicit engineering [3]. However, DL has hit a plateau in performance upon application in TS forecasting.

B. Statistical-based Forecasting Techniques

Statistical forecasting is an application of statistics to historical data to forecast what could happen in the future.

1. ARIMA

The Autoregressive Integrated Moving Average (ARIMA) model is a combination of the Autoregressive (AR) and Moving Average (MA) models. The ARIMA model predicts future values based on the past [5]. This specific model is the most popular in TS forecasting due to its considerably good performance; however, specific vital points that cause these models to produce inaccuracies need to be considered: it bases its future values on its past, therefore, can be inaccurate when used in open systems; it is pretty complicated and hence lacks expressivity; and, it cannot be used for seasonal data.

2. SARIMA

Seasonal Autoregressive Integrated Moving Average (SARIMA) is an extension to ARIMA that was introduced to

handle seasonality in data. Research performed by [6] has proven that SARIMA produced better performance than ARIMA; however, its parameters are more sensitive to changes where even a slight change could result in poor performance.

3. GARCH

Generalized Autoregressive Conditional Heteroskedasticity (GARCH) is an approach to estimating market volatility while being more contextual than others when predicting prices or rates [7]. Research conducted by [8] identified that the forecasts attempted by GARCH were more accurate than ARIMA's as ARIMA cannot capture volatility in the dataset.

4. PROPHET

Facebook introduced an approach to solving the challenge of forecasting at scale via adjusting parameters [9]. It was designed to forecast daily data consisting of weekly and yearly seasonality, considering the effects of holidays [10]. Therefore, it performs best for highly seasonal data.

Is there a clear better choice? Upon understanding current statistical algorithms, research suggests that there is no transparent superior model as the differences in errors and accuracy are low, which signifies that the better-performing model will change according to the domain, the problem, and the dataset.

C. Deep Learning-based Forecasting Techniques

Studies have shown that certain TS forecasting domains perform better on statistical models (ex: [11]). At the same time, other studies have shown that DL models have outperformed statistical models (ex: [12]). Therefore, it is also expected to be worth evaluating DL models.

1. LSTM

Long Short-Term Memory (LSTMs) was introduced by [13] to solve the issues in previous RNNs of the inability to store information. They paved the path for more performant future RNNs as they remember short-term patterns, which made modeling temporal dependencies for larger horizons possible [1]. Studies conducted to identify the impact of LSTMs on TS proved multiple advantages, the most prominent being the ability to extract meaningful information from TS data.

2. GRU

Gated Recurrent Units (GRUs) are, in effect, a simplification of an LSTM where the 'forget' and 'input' gates are combined. GRUs achieve similar results to LSTMs while taking less computational time [1]. There is no clear distinction between GRUs and LSTMs on which one performs better; therefore, generally, both are attempted.

GRUs and LSTMs are similar in performance. Another stalemate arises as there is a dispute in the works of [14] and [15]: [14] demonstrated that the GRU performance was superior to ARIMA and LSTM; meanwhile, [15] observed better performance in LSTMs compared to ARIMA and GRU. Therefore, it is evident that there is no superior between GRU and LSTM. However, these non-linear models perform better than statistical models, in this case, ARIMA, which bestows more credibility on the study conducted by [16].

3. N-BEATS

Neural Basis Expansion for Interpretable Time Series (N-BEATS) is a relatively new state-of-the-art forecasting algorithm. The proposed architecture aimed at solving univariate point forecasting that carries multiple properties, the most impactful being interpretable and generalizable to multiple domains [17]. N-BEATS overcame the M4 competition's previous winner, a hybrid statistical and neural net model, while being a pure DL architecture that is simple, generic, and expressive. This finding inspired the authors to challenge the claim of [18]: the future of TS forecasting is hybrid architectures of statistical and neural net models.

4. TFT

Temporal Fusion Transformer (TFT) is a novel attention-based architecture proposed by Google to solve the challenge of multi-horizon forecasting optimized explicitly for outstanding performance and interpretability – as existing solutions are typically a 'black box.' The architecture has specialized features that make it possible to choose required features and remove unnecessary ones [19]. In experimentation, it was identified that there were significant performance improvements compared to existing benchmarks. The proposed architecture was influenced upon reviewing existing transformer-based architectures (ex: [20]) that had yet to consider different types of inputs present in multi-horizon forecasting or assumed the required exogenous features are always available.

Are DL models superior to statistical models? Based on the literature, the belief that statistical models are superior at all times is only partially justifiable. It even cannot be deduced that non-linear DL models are superior. Therefore, it is worth investigating both schools of thought when solving TS problems and choosing the model that demonstrates better performance for that problem.

Please refer to the supplementary material S1 for a more detailed breakdown of these algorithms.

III. Concerns on Existing Used Techniques

A. Issues in statistical techniques

Based on the literature, the following drawbacks can be identified with statistical-based models.

- Linearity
- Suitability for mostly seasonal data
- Lack of interpretability and expressivity
- Altering model parameters requires more in-depth domain knowledge

B. Issues in Deep Learning techniques

As identified by [18], in contrast to other domains of NLP and computer vision, DL models still struggle in TS forecasting. Additionally, it is worth noting that although the non-linearity introduced by neural networks improves performance, they lack expressivity.

A general issue with the above models is that they are static and lack adaptability. TS data is volatile, ever-changing, and unpredictable; they can get unexpected characteristic changes to their inputs; therefore, a fixed statistical model or neural network has the possibility of struggling, which could be a reason for the identification of [19].

Given the open-ended conclusion on the comparison between statistical and DL models, the natural question is **how to proceed**.

IV. Neural Ordinary Differential Equations

NODEs are a new family of DL models that are continuous-depth, have constant memory cost, implicitly trade numerical precision for speed, and, most importantly, can adapt the evaluation strategy based on inputs [21].

Paper [21] claimed that RNNs with continuous-time hidden states determined by ODEs are effective algorithms for modeling TS data, proven by their ability to adapt computation depending on the input data. They proposed the following change to the classical equation followed by RNNs and residual networks.

Equation followed by traditional residual networks

$$h_{t+1} = h_t + f(h_t, \theta_t) \quad (1)$$

Equation proposed

$$dh(t)/dt = f(h_t, t, \theta_t) \quad (2)$$

Where $h_t \in \mathbb{R}^D$, $t \in \{0 \ldots T\}$

The equation proposed utilizes an ODE specified by a neural network to "*parameterize the derivative of the hidden units*" instead of specifying a sequence of hidden layers.

The ability to adapt to incoming data streams could be a promising application in the domain of TS forecasting. However, these ODEs fail to identify uncertainties in predictions and are not robust [22].

A. Liquid Time-Constant Networks

Research [23] proposed a solution to improve the performance of NODEs, as experiments performed on other NODE architectures produced underwhelming performance compared to traditional LSTM architectures. The proposed architecture exhibits stable and bounded behavior alongside being highly expressive. The alternate formulation demonstrates 'liquid' features which give rise to better performance for TS predictions.

It is also speculated that the solution can learn during training and while in use by continuously changing the underlying formulations to adapt to the changes in incoming inputs [24]. What is impactful is that these networks can adapt to the changes in real-world systems while also being robust, stable, and more interpretable.

The architecture is richer in information while being scaled down than other neural networks, which adds another advantage of it being easy to glance into the 'black box' of the underlying network to understand the reasoning behind certain computations – as realized, is another drawback of standard neural networks.

The formula is an alternative to what [25] proposed. Instead of using a neural network to define the derivative, an additional term assists the system in reaching equilibrium [21].

V. Neural Stochastic Differential Equations

ODEs can be used to abstract deterministic models, as utilized by [21], to propose neural ODEs. Hence, the natural question arises: what **could be used if a state of randomness exists?**

SDEs are ODEs with additional noise, which can be used to model uncertainty. Neural SDEs are similar to NODEs in that an ODE can be used as the solver; however, the additional focus is the 'stochastic evolution' instead of the 'deterministic evolution' [26]. Considering NODEs and their expressive power, SDEs can enhance the expressive power even further, as proposed by [27].

The issue is that data prone to tiny and rapid changes cannot be modeled well by NODEs, as these types of data are bound to have noise. SDEs are suitable for fitting such data as they can model instantaneous changes via random noise.

VI. The Ultimatum - LTCs with SDE Flexibility

Based on the above analysis, it is unclear what exactly would be the next step or even the end proposition. The LTC architecture proposed by [23] utilizes the more obsolete ODE, which cannot model randomness and instantaneous changes. However, the usage of SDEs is more flexible as it handles randomness.

An intuitive next step that a researcher could investigate is obtaining inspiration from the LTC architecture, replacing the underlying ODE with an SDE instead, which manifests a novel and more flexible implementation capable of handling random noise.

A. Formulation

Considering the formulation proposed by the original LTC, the structure can be tweaked by utilizing a linear system of SDEs to declare the flow of the network instead of ODEs. The proposed linear ODE system by [29] can be improved by adding uncertainty to produce the following: $dx(t)/dt = -x(t)/\tau + S(t) + \epsilon(x(t))B$.

This can then be plugged into the LTC formulation instead of the ODE system to manifest the following novel formula. *The supplementary material S2 provide more in-depth proof.*

$$\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta) - \sigma B\right] x(t) + f(x(t), I(t), t, \theta) A \quad (3)$$

The sole difference from the original LTC is the additional "$-\sigma B$" term which can model uncertainty.

B. Forward propagation with implicit SDE solvers

Paper [23] determined that their LTC architecture that uses a linear system of ODEs was 'stiff equations.' They also found that regular Runge-Kutta was not suitable for solving LTCs; therefore, they designed a custom ODE solver by combining both implicit and explicit Euler methods.

As this system uses SDEs, SDE solvers must be used. As [23] determined, the architecture is a system of stiff equations. Therefore, as [29] decided, researchers must use an implicit solver to ensure stability. Additionally, researchers can combine an explicit solver to achieve further stability. For this research, the authors will use an implicit SDE solver – creating a custom SDE solver by fusing an explicit solver within is something researchers should explore in the future.

To solve the state of these SDEs, we propose using implicit Euler-Maruyama solver, as it can handle all forms of noise and it is immensely popular. Having mentioned this, evaluating other implicit SDE solvers is recommended to determine whether there are more suitable options.

Note: L = number of steps

TABLE I. COMPLEXITIES OF BPTT AND ADJOINT SENSITIVITY

	BPTT	**Adjoint sensitivity**
Time	O(L)	**O(LlogL)**
Memory	O(L)	**O(1)**
Forward accuracy	High	High
Backward accuracy	**High**	Low

C. Training the network with BPTT

Training these networks has a trade-off between accuracy and memory. [21] promoted the use of the adjoint sensitivity method to perform reverse-mode Automatic Differentiation (AD), which is more memory efficient. Paper [23] mentioned that this method introduced more numerical errors and opted to use the traditional Back Propagation Through Time (BPTT) approach, which is more accurate but consumes more memory.

Although a technique of adjoints exists specifically for SDEs, they cannot be used, as determined by [26], and hence requires a custom-built backpropagation rule.

The authors suggest the BPTT approach to maintain high accuracy with the sacrifice of lower memory consumption, as it can be determined whether said architecture is more accurate. Researchers must investigate reverse-mode AD in the future as it is the recommended approach when memory efficiency is more important.

It is worth noting that using the BPTT approach carries added benefits, such as being able to be used as a Recurrent Neural Network (RNN) layer alongside popular optimization algorithms such as stochastic gradient descent (SGD) and Adam [30].

Table 1 summarizes the time and memory complexities of the vanilla BPTT and adjoint sensitivity approaches.

VII. CONCLUSION AND FUTURE WORK

We aimed to summarize existing TS forecasting algorithms and their respective applications, highlighting the issues of forecasting systems utilizing these algorithms. We then dive into a new family of DL algorithms that attempt to fix certain issues by providing continuous-time architectures while introducing other issues and not providing the most effective solution. We further dive into a more cutting-edge algorithm that attempts to fix these issues in NODEs; however, the drawback is that the internal system uses obsolete technology. We finally propose a new architecture heavily inspired by the LTC but by modifying the underlying linear system and proposing respective forward and backward propagation algorithms while also mentioning some compromises that require future research.

The domain of TS forecasting has been stagnant and hindered by the abovementioned more traditional statistical and DL techniques. However, recent advancements in DL have introduced NODEs – a new family of DL architectures that have attempted to surpass these limitations – some being extremely close. As this domain is relatively new, it is important to focus on this research area as the next big thing could be soon.

REFERENCES

[1] P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme, "An experimental review on deep learning architectures for time series forecasting," *International Journal of Neural Systems*, vol. 31, iss. 3, pp. 2130001, Mar 2021.

[2] L. J. Chaman, "Answers to your forecasting questions," *Journal of Business Forecasting*, vol. 36, iss. 2, Summer 2017.

[3] B. Lim and S. Zohren, "Time series forecasting with deep learning: a survey," *Phil. Trans. R. Soc.*, vol. 379, iss. 2194, Feb 2021.

[4] N. K. Ahmed, A. F. Atiya, N. E. Gayar and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, iss. 5-6, pp. 594–621, Sep 2010.

[5] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*, 5th ed. John Wiley & Sons, 2015.

[6] M. Valipour, "Long-term runoff study using SARIMA and ARIMA models in the United States: runoff forecasting using SARIMA," *Meteorological Applications*, vol. 22, iss. 3, pp. 592–598, Jul 2015.

[7] R. F. Engle, "Autoregressive Conditional Heteroscedasticity with estimates of the variance of United Kingdom inflation," *Econometrica*, vol. 50 iss. 4, pp. 987, Jul 1982.

[8] S. P. Bhardwaj, R. K. Paul, D. R. Singh and K. N. Singh, "An empirical investigation of Arima and Garch models in agricultural price forecasting," *Economic Affairs*, vol. 59, iss. 3, pp. 415, Jan 2014.

[9] S. J. Taylor and B. Letham, "Forecasting at scale," PeerJ Preprints. Rep. 5, 2017.

[10] R. J. Hyndman and G. Athanasopoulos. (2021, May 31). *Forecasting: principles and practice* (3rd ed)[Online]. Available: https://otexts.com/fpp3/.

[11] R. Zhang, et al., "Comparison of ARIMA and LSTM for prediction of hemorrhagic fever at different time scales in China," *PLOS ONE*, vol. 17, iss. 1, pp. e0262009, Jan 2022.

[12] S. Siami-Namini, N. Tavakoli and A. Siami Namin, "A comparison of ARIMA and LSTM in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Orlando, FL, 2018, pp. 1394-1401.

[13] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9 iss. 8, pp. 1735–1780, Nov 1997.

[14] L. Kuan, et al., "Short-term electricity load forecasting method based on multilayered self-normalizing GRU network," in *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, Beijing, 2017, pp. 1-5.

[15] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep LSTM recurrent networks," *Neurocomputing*, vol. 323, pp. 203–213, Jan 2019.

[16] M. Maiti, Y. Vyklyuk and D. Vuković, "Cryptocurrencies chaotic co - movement forecasting with neural networks," *Internet Technology Letters*, vol. 3 iss. 3, May 2020.

[17] B. N. Oreshkin, D. Carpov, N. Chapados and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *arXiv* [Online], Feb 20 2020. Available: https://arxiv.org/abs/1905.10437.

[18] S. Makridakis, E. Spiliotis and V. Assimakopoulos, "The M4 competition: results, findings, conclusion and way forward," *International Journal of Forecasting*, vol. 34 iss. 4, pp. 802–808, Oct 2018.

[19] B. Lim, S. O. Arik, N. Loeff and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, iss. 4, pp. 1748-1764, Jun 2021.

[20] S. Li, et al., "Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting," in *2019 33rd Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, 2019.

[21] R. T. K. Chen, Y. Rubanova, J. Bettencourt and D. Duvenaud, "Neural Ordinary Differential Equations," *arXiv* [Online], Dec 14 2019. Available: https://arxiv.org/abs/1806.07366.

[22] S. Anumasa and P. K. Srijith, "Latent Time Neural Ordinary Differential Equations," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36 iss. 6 pp. 6010–6018, Jun 2022.

[23] R. Hasani, M. Lechner, A. Amini, D. Rus and R. Grosu, "Liquid Time-constant networks," *arXiv* [Online], Dec 14 2020. Available: https://arxiv.org/abs/2006.04439.

[24] MIT News | Massachusetts Institute of Technology. (2021, Jan. 28). *"Liquid" machine-learning system adapts to changing conditions* [Online]. Available: https://news.mit.edu/2021/machine-learning-adapts-0128.

[25] K. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, iss. 6, pp. 801–806, Jan 1993.

[26] B. Tzen and M. Raginsky, "Neural Stochastic Differential Equations: deep latent Gaussian models in the diffusion limit," *arXiv* [Online], Oct 28 2019. Available: https://arxiv.org/abs/1905.09883.

[27] S. Peluchetti and S. Favaro, "Infinitely deep neural networks as diffusion processes," *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, vol. 108, pp. 1126-1136, 2020.

[28] L. Lapicque, "Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization," *Journal de Physiologie et de Pathologie Generalej* vol. 9, pp. 620–635, Oct 1907.

[29] W. H. Press, *Numerical recipes: the art of scientific computing*, 3rd ed. Cambridge, UK; New York: Cambridge University Press, 2007.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv* [Online], Jan 30 2017. Available: https://arxiv.org/abs/1412.6980.

[31] K. Cho, et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation, Sep 2014.

[32] S. Bouktif, A. Fiaz, A. Ouni and M. Serhani, "Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: comparison with machine learning approaches †," *Energies*, vol. 11 iss. 7, pp. 1636, Jun 2018.

[33] Y. Wang, W. Liao and Y. Chang, "Gated Recurrent Unit network based short-term photovoltaic forecasting," *Energies*, vol. 11, iss. 8, pp. 2163, Aug 2018.

[34] I. Yenidogan, A. Cayir, O. Kozan, T. Dag and C. Arslan, "Bitcoin forecasting using ARIMA and PROPHET," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sarajevo, 2018, pp. 621–624.

[35] YouTube (2021, Jun. 2). *Directions in ML: Latent Stochastic Differential Equations: An Unexplored Model Class* [Online]. Available: https://www.youtube.com/watch?v=6iEjF08xgBg.

[36] U. Ugurlu, I. Oksuz and O. Tas, "Electricity price forecasting using recurrent neural networks," *Energies*, vol. 11, iss. 5, pp. 1255, May 2018.

[37] C. Pan, J. Tan, D. Feng and Y. Li, "Very shortterm solar generation forecasting based on LSTM with temporal attention mechanism," in *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2019, pp. 267-271.

[38] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research* vol. 270, iss. 2, pp. 654-669, Oct 2018.

SUPPLEMENTARY MATERIALS

S1. Summarized analysis

TABLE II. ANALYSIS OF FORECASTING ALGORITHMS

Ref.	Brief	Improvements/Contribution	Limitations/Future work
Statistical-based forecasting algorithms			
[5]	**ARIMA**. A statistical analysis model for understanding the dataset or predicting future trends. This model depends on past values to predict the future and uses lagged moving averages to smoothen the data.	Improved performance for TS forecasting data that correlate with values ahead of time.	Does not handle well with nonlinear data and long-term forecasting. Furthermore, it performs best on univariate analysis and cannot capture data volatility.
[7]	**GARCH**. A modeling technique that specializes in predicting volatility in data.	Captures volatility in datasets and boasts significant performance improvements in the family of statistical forecasting algorithms.	Needs to improve interpretability and adaptability.
[9]	**Prophet**. A modular regression model with interpretable parameters. These parameters can be adjusted according to the problem by domain experts, similar to ARIMA.	Solves forecasting at scale, where scale refers to three types. 1) A large number of people forecasting. 2) A large variety of problems. 3) A large number of forecasts being created.	It uses simple and weak assumptions and produces much poorer performance than ARIMA. And it does not model relationships between the past and future.
DL-based forecasting algorithms			
[13]	**LSTM**. An algorithm that learns to bridge minimal time lags by enforcing constant error flows. It learns much faster, creates more successful runs, and can solve complex tasks that have not been solved before.	Improved performance for short-sequence predictions. Overcame error back-flow problems present in conventional BPTT, where they tended to blow up or vanish.	Prediction capacity limits long sequence performance, where the MSE and RMSE rise unacceptably. Therefore, there are better solutions for predictions of the distant future. They are also prone to overfitting.
[31]	**GRU**. Similar architecture to that of LSTMs but combine the 'forget' and 'input' gates to create two gates, 'reset' and 'update,' instead of the three found in LSTMs.	Solve the vanishing gradient problem in RNNs as LSTMs, but also consume less memory and run faster.	Suitable for problems with smaller datasets and tend to be less accurate for datasets with larger sequences.
[17]	**N-BEATS**. An architecture that solves the univariate time series point	Outperformed the M4 competition winner of the previous year and	Tailored specifically for univariate TS analysis, therefore, would

	forecasting problem. It carries some benefits, some of which are being understandable, easily applicable to multiple other fields, and being fast to train.	improved the statistical benchmark forecast.	perform poorly on multivariate analysis. Additionally, Meta-learning is speculated to be a reason for the performance and must be investigated.
[19]	**TFT**. An attention-based architecture that solves multi-horizon forecasting with interpretability of the used inputs.	Demonstrate significant performance improvements over set benchmarks for a variety of datasets.	Training and inference times are expensive and require moderately extensive resources. Hardware optimizations can reduce these.
[23]	**LTC**. A novel formulation of the NODE architecture. Boasts superior expressivity that is capable of adapting to unforeseen changes.	Surpassed traditional DL and statistical models and overcame the underwhelming performance of other NODE architectures.	It cannot model uncertainty and is computationally intensive.

TABLE III. FEW STUDIES ASSOCIATED WITH THESE ALGORITHMS

Ref.	Technology	Outperforms	Findings
Statistical-based forecasting algorithms			
[11]	ARIMA	LSTM	ARIMA outperformed the LSTM model for monthly and weekly forecasts, while LSTM performed better for daily forecasts. Additionally, they mentioned that there is no clear superior.
[34]	Prophet	ARIMA	Prophet is strong to outliers and missing data. It is also optimized for business forecasts with trends and seasonality within and nonlinear data growths, which ARIMA cannot handle.
DL-based forecasting algorithms			
[14]	GRU	LSTM, traditional GRU	Scaled exponential linear units were proposed to deal with vanishing gradients. These architectures significantly outperformed LSTM and traditional GRU models.
[36]	GRU	MLP, LSTM	Performed much better than LSTM and trained faster as it is simply a simpler form of LSTM.
[33]	GRU	LSTM, ARIMA	K-Means clustering and Pearson coefficient were used to cluster groups and extract the most important features, respectively, which were then used to train the model.
[15]	LSTM	ARIMA, GRU	Genetic algorithms were used to create an optimized LSTM architecture.
[32]	LSTM	MLP, Linear regression	The most optimal time lags and number of layers for an LSTM was found using genetic algorithms.
[38]	LSTM	MLP, Logistic regression	Peeked into the internals of the LSTM to find common stock patterns in noisy data.
[37]	LSTM	MLP, traditional LSTM	Utilized a novel attention-based LSTM architecture to improve the performance of traditional LSTMs.
[17]	N-BEATS	Competition winner	Surpassed the previous winner of the M4 competition with a significant difference in performance. Concluded that hybrid models are only sometimes the best-performing.
[23]	LTC	LSTM, GRU, (Continuous time) CT-RNN, CT-GRU	A more stable implementation of the NODE can be built by using a linear system of ODEs to declare the network's flow.

S2. Proof of Proposed Formula

1. Transitioning from an ODE to an SDE

In simple terms, an SDE is an ODE with additional noise added at each step, which the model can use to model uncertainty.

Assume an ODE is: $\frac{dx}{dt} = f(x)$; *which obtains the expected slope of x(t)* (4)

The above ODE can be used to calculate the 'expected' slope, whereas the 'realized' slope differs from the 'expected' due to random noise, also called random Gaussian perturbations or Gaussian white noise. With that in consideration, the following can be derived:

$$\text{An SDE is: } \frac{dx}{dt} = f(x) + \mathcal{E}_{t+dt}\text{; where } \mathcal{E}_{t+dt} \text{ is } \sim N(0,1) \tag{5}$$

Where $N(0,1)$ is a Gaussian 0,1 random variable

However, noise can be of varying intensities (some could be high, some could be low). Considering this varying intensity, the SDE can be further expressed as follows:

$$\frac{dx}{dt} = f(x) + g(x) * \mathcal{E}_{t+dt}\text{; where g(x) is the intensity} \tag{6}$$

As implied, the missing factor in the existing architecture that consists of ODEs is the absent stochastic transition dynamics (i.e., a noise for each timestep – which is vital to model the tiny unobserved interactions). The above equation considers the small unobserved interactions and uncertainties that could occur; this is further important in the context of TS data, as the initial state of data is unlikely to be certain.

2. Adding neural networks into SDE dynamics

Based on the findings of [35], the noise mentioned in the previous step can be considered as Brownian motion, a generalized form of the Gaussian noise. Researchers can produce the following by plugging Brownian motion into the equation determined in the previous step.

$$dx = f(x(t))dt + \sigma(x(t))dB(t) \tag{7}$$

A neural network can be integrated into the above equation to solve the system, resulting in the following equation:

$$dx = f_\theta(x(t))dt + \sigma_\theta(x(t))dB(t) \tag{8}$$

where f is usually a tiny neural network and θ are its parameters

3. Integrating the above equation into the LTC architecture

Moving back to the main problem at hand, the author can now construct a new formula by using the equation determined in the previous step.

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) \tag{9}$$

As the above equation is a linear system of ODEs initially proposed by [28], the author could add the uncertainty noise to the equation to produce the following:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma(x(t))B \tag{10}$$

The above equation now defines a stochastic process instead of deterministic evolution. Therefore, researchers can model any tiny unobserved interactions. Finally, the following could be derived by applying this to the LTC formula:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma(x(t))B \tag{11}$$

Replace S(t) with the nonlinearity proposed,

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + f(x(t), I(t), t, \theta)(A - x(t)) + \sigma(x(t))B \tag{12}$$

Expand out the equation,

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} - f(x(t), I(t), t, \theta)x(t) + \sigma(x(t))B + f(x(t), I(t), t, \theta)A \tag{13}$$

Lastly, refactor the equation into the format of the original LTC

$$\boxed{\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta) - \sigma B\right]x(t) + f(x(t), I(t), t, \theta)A} \tag{14}$$