Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK

The University of Westminster, Coat of Arms

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

Time Series (TS) forecasting has stagnated owing to algorithm restrictions; therefore, systems developed using these methods are also limited in performance. TS remains a challenge despite recent advances in Deep Learning (DL) in Natural Language Processing (NLP) and Reinforcement Learning (RL).

Multiple approaches have been taken to solve this problem, the more promising being Neural Ordinary Differential Equations (NODEs) that introduce the concept of continuous-time and depth models. Although the idea seems promising, the results are underwhelming compared to traditional neural networks such as Long Short-Term Memory (LSTM). Considering these limitations, MIT researchers introduced Liquid Time-constant (LTC) networks that surpassed performance expectations. However, these networks carry problems of their own: the obsolete architecture used behind the scenes. This research project proposes a new algorithm that uses the LTC's liquid adaptability with a more modern architecture that uses Stochastic Differential Equations (SDEs). Furthermore, its application is demonstrated on a TS forecasting problem that has piqued the interest of many audiences worldwide – forecasting the price of Bitcoin (BTC).

The LTC with SDEs, dubbed "**Liquid Time-stochasticity**" by the author, produced a more stable and robust implementation of continuous-time and depth models due to its ability to handle small noises and manage immediate changes. The solution utilized **traditional Backpropagation Through Time (BPTT)** to produce more accurate results with the trade-off of consuming more memory, which created a promising result in its application in the chosen domain - a percentage error of $\mathbf{2.5\% \pm 0.1}$. It is worth noting that, utilizing adjoint sensitivities instead must be researched in the future as it is recommended.

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.

CONTENTS

.....

LIST OF TABLES

LIST OF FIGURES

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.
SGD	Stochastic Gradient Descent.
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 the it would cover over in the following chapters. In this chapter, the author aims to identify and provide the reader with an overview of the current issues in time series forecasting and highlight what a liquid time-stochasticity network is and what it aims to solve. In detail, the author will define the problem and evaluate the necessary literature to arrive at a novel research gap capable of producing significant contributions, respective research questions and objectives, and challenges that would arise upon embarking this research journey.

1.2 Problem domain

1.2.1 Time series forecasting

TS forecasting is a significant business issue and an area where 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 (Jain, 2017).

Having said that, although ML and DL have outperformed classical approaches for NLP and computer vision, the domain of TS still seems to be a struggle compared to classical statistical methodologies (Makridakis et al., 2018a;b). Most of the top-ranking methods in the M4 competition were ensembles of traditional statistical techniques (Makridakis et al., 2018b) - where the winner was a hybrid model of LSTM and classical statistics (Smyl, 2020) - while regular ML methods were nowhere near. Furthermore, they claimed that the only way to improve the accuracy of TS forecasting was by creating such hybrid models, which the author aspires to challenge in this research project.

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 there exist continuous time-hidden states determined by ODEs (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) produce such performance. However, they have issues with expressivity and having fixed behaviour once trained (Hasani et al., 2020). Therefore, the question arises: what would happen if there were unexpected changes to the characteristic of the inputs during inference? Moreoever, these algorithms lose in generalization compared to even a simple LSTM network (Hasani et al., 2021), which raises another question, what is the point of defining a different and "*fancy*" approach if they cannot work 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 yields 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 architecture is obsolete (Duvenaud, 2021) and lacks adaptability to noisy data and data with high volatility. The LTS algorithm presented by the author is an enhancement to the LTC considering these limitations.

1.2.3 Cryptocurrencies

The word 'crypto' has been an enormous buzzword recently, especially BTC. It has even come to the point where crypto and BTC are used interchangeably.

Cryptocurrencies are a fully decentralized digital currency form; it is a form of a peer-to-peer system without the need for a third party, thus enabling safer online transactions (S. Nakamoto, 2008). BTC is the first and the most popular digital currency to date, piquing many academic researchers' interest (Rahouti et al., 2018).

However, being at the forefront of the digital currency world, it has developed high volatility, making it difficult to predict future rates and a disadvantage for multiple investors (Kervanci and Akay, 2020). Despite that, recent advances in ML and statistics have shown

acceptable results in the analysis and prediction of cryptocurrencies, yet the root cause of these algorithms persists: they are static.

1.3 Problem definition

Most applications of ML available do perform quite well (ex: image classification, transfer learning, NLP), yet, as mentioned, the field of TS forecasting is subpar (Makridakis et al., 2018a;b). Existing TS forecasting algorithms cannot adapt to unforeseen changes in data streams (Hasani et al., 2020); consequently, they perform relatively poorly when applied to TS forecasting.

It is worth noting that the neural ODEs (Chen et al., 2019), and notably the LTC architecture proposed by Hasani et al. (2020) rectify this to some extent; however, their proposed solution does not have the ability to model instantaneous changes. Therefore, they perform poorly when applied to areas of high volatility (Duvenaud, 2021) (in this case: the forecasting of BTC).

1.3.1 Problem statement

Existing TS forecasting systems cannot adapt to incoming data streams with unexpected changes and characteristics that are different from the data on which they were trained.

1.4 Research motivation

The field of AI, particularly neural networks, has been growing exponentially recently, alongside intriguing research. However, as mentioned by Hasani et al. (2020), the issue of networks being static and unable to adapt to varying characteristics could prove to be a limitation for the future of intelligent systems, TS in particular. Therefore, this research is expected to facilitate further exploration by trying to aid in driving the domain forward.

1.5 Research gap

The literature defines only a single paper for the LTC (Hasani et al., 2020): every other work is not directly related to the LTC but is to 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. As the LTS is an enhancement to the LTC, the lack of documentation is a significant hindrance.

Gap in existing forecasting algorithms

Existing forecasting solutions are all implemented using traditional deep neural net approaches (ex: LSTM (Hochreiter and Schmidhuber, 1997)) that are static and hence have the limitation of not being able to learn and adapt during inference (Hasani et al., 2020), which results in the model's accuracy degrading overtime – a '*data drift*' (Poulopoulos, 2021).

Gap in LTC

The proposed LTC architecture uses a sequence of linear ODEs, which are now considered obsolete and lack rapid adaptability (Duvenaud, 2021). Recent advancements in this field suggest the usage of SDEs instead, as they are more flexible. An additional issue is that ODEs model 'deterministic dynamics' – uncertainty, or any unobserved interactions cannot be modelled, which is inevitable in TS data. Therefore, the way forward is to build the LTC architecture with SDEs instead. As implied, the algorithm proposed is the author's own, giving it the name of "Liquid Time-stochasticity", as it is no longer constant.

Gap in bitcoin forecasting solutions

The work available on BTC forecasting has not considered exogenous factors that could have an impact (Roy et al., 2018; Rizwan et al., 2019; Fleisher et al., 2022). Therefore, a significant concern is that they cannot adapt well; in other words, they are not robust. Factors that could influence the price are as follows:

- Tweet sentiment & volume
- Google Trends

Cryptocurrency forecasting is as reliable as palm reading since it is an open system. Hence, uncontrollable factors such as a country's law can affect the price. Despite this, researchers can consider certain factors, such as the ones mentioned above. Abraham et al. (2018) identified that the above factors correlate with the price of BTC; therefore, it is vital to consider them when building such systems in the future.

1.6 Contribution to the body of knowledge

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

- Would a novel architecture built by the LTS algorithm be an advancement for TS forecasting?

1.6.1 Research domain contribution

An implementation of the LTS algorithm will be developed, following the proposed architecture, to facilitate the model creation. Additionally, the algorithm built will be generalized without being problem-specific so that researchers can apply it elsewhere to evaluate its performance and identify whether it would also be an advancement to those domains.

Having understood the issues in the current literature, a solution is a dynamic algorithmic architecture that can adapt and change its underlying mathematical expressions and evaluation strategies based on changes in the incoming data streams. Additionally, further enhancements are required to avoid sudden and tiny changes that are common in TS data. Being able to adapt to unforeseen changes and being highly expressive could be the stepping stone within the TS forecasting community to aid in future research.

1.6.1 Problem domain contribution

Based on the above critique, creating a more robust forecasting solution considering the mentioned factors (Twitter, Google Trends) could mean that the highly volatile market of cryptocurrencies could be predicted much more efficiently and be the way forward for investors.

Furthermore, there is no direct data source that the author can utilize for building the model; therefore, dedicated scripts are developed to obtain the said data. These scripts will be available for public access to support future research on BTC forecasting.

1.7 Research challenge

Existing architectures scale up, and the LTC scales down - with more expressive nodes (Hasani et al., 2020). Having adapted to the “deeper is usually better” mindset of deep neural nets, a challenge opens up in **identifying the requirement of scaling down and what a neural ODE aims to solve**.

LTCs are a new approach with only a single research paper regarding its proposed solution. Currently, it is only in the experimental stage and utilizes a novel formulation compared to other existing neural ODEs (Hasani et al., 2020).

The broader domain of neural ODEs (Chen et al., 2019) **is also relatively new**; hence the scarcity of references could create more challenges for further research or implementation of systems.

SDEs are an advanced topic in mathematics, and modelling them as neural SDEs have had a couple of research conducted; however, they were primarily for specific purposes. Therefore, **no generic papers exist for neural SDEs, unlike neural ODEs, which makes modelling difficult**.

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**.

The chosen domain of application is an open system. **Open system forecasting is usually poor and generally difficult to beat the naïve forecast** (Gilliland, 2014) since it can depend on any external factor. Therefore, there is the possibility of discouragement from continuing the research if the results are not as expected.

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 be used to implement a TS forecasting system, and how will the challenges faced be overcome?

RQ3: What contributions can be made to the chosen domain?

1.9 Research aim

The aim of this research is to design, develop & evaluate a novel algorithm (LTS) inspired by the LTC so that it can build intelligent systems by developing a novel architecture for TS forecasting, which could be the stepping stone to be further expanded to other domains as well.

Specifically, this research project will produce a TS forecasting system utilizing the LTS algorithm, focused on the forecasting of 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	Understand and document the identified problem and provide reasoning on what makes it novel. **R**O1: Conduct research on a domain of interest and identify a comprehensive enough issue that requires solving. **R**O2: Delve deeper into the identified problem to obtain a general understanding on how to approach solving the problem. **R**O3: Split the problem down into manageable subsections so it is easier to digest and to solve one section at a time. **R**O4: Design a respective schedule, associated deliverables, and the Gantt chart.	LO1, LO2	**R**Q1
Literature Review	Collate relevant information by reading, understanding, and evaluating previous work. **R**O5: Conduct preliminary studies and investigations on existing TS forecasting systems and algorithms. **R**O6: Analyze the requirement for specialized TS algorithms. **R**O7: Conduct research on neural ODEs, LTCs & SDEs. **R**O8: Obtain deep insights into the architecture behind the LTC.	LO1, LO4, LO5	**R**Q1

	RO9: Research and obtain insights on factors affecting the price of BTC. **R**O10: Research on existing BTC forecasting and related open market systems. **R**O11: Research on necessary ML techniques and evaluation approaches.		
Requirement Elicitation	Collect and analyze project requirements using appropriate tools and techniques. **R**O12: Analyze stakeholders and understand their viewpoints and concerns. **R**O13: Gather the requirements and architectures of LTCs and SDEs. **R**O14: Collate the most up-to-date details of BTC and obtain insights on the perspectives of the end users. **R**O15: Design necessary diagrams to justify the product's specification.	LO1, LO3	**R**Q1
Design	Design the architecture and a corresponding system capable of effectively solving the identified problems. **R**O16: Design necessary diagrams required to understand the algorithm. RO17: Design diagrams required to understand the supplementary system being developed. **R**O18: Design the novel algorithm and analyze its complexities.	LO1	**R**Q2, **R**Q3
Implementation	Implement a system that is capable of addressing the research gaps. **R**O19: Design, evaluate and pick necessary technologies best-suited for the implementation. **R**O20: Develop an efficient LTC implementation.	LO1, LO5, LO6, LO7	**R**Q2, **R**Q3

	RO21: Build on the LTC to implement the LTS. **R**O22: Integrate the algorithm developed into a TS forecasting application. **R**O23: Integrate the intelligent system into a client application to display forecasts. **R**O24: Design and implement an automated flow to update the built network with the latest data. **R**O25: Design and implement a pipeline for easy deployments. **R**O26: Consider any legal, social, ethical & professional issues upon implementation.		
Evaluation	Effectively test the algorithm implemented, the system, and the respective data science model using recommended techniques. **R**O27: Evaluate the developed algorithm and the respective model against the evaluation metrics researched in the literature review. **R**O28: Create a test plan & test cases and perform functional, non-functional, and integration testing.	LO4	**R**Q2, **R**Q3
Documentation	Document the progression of the research project and inform about any challenges faced. **R**O29: Create a coherent report of new skills obtained, evaluations, contributions etc., and ensure that all the above-stated objectives are met.	LO6, LO8	**R**Q3

1.11 Chapter summary

In this chapter, the author provided an overview of the research project carried out, respective reasons for the research and problem to be a novelty, and the challenges they can face upon solving it. Furthermore, the necessary goals that must be aimed to consider the research successful were proposed and mapped to the learning outcomes that must be attained by the chosen degree.

CHAPTER 02. LITERATURE REVIEW

2.1 Chapter overview

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

2.2 Concept map

Upon researching available literature, the scope of this literature review was broken down by a concept map. The map was designed to identify the required literature to be covered; it can be found in **APPENDIX I**.

2.3 Problem domain

2.3.1 Time series forecasting

TS forecasting has attracted multiple researchers to develop a robust, scalable, and usable solution. It can play 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). In a nutshell, 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.

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 (Lim and Zohren, 2020). ML provides a means of learning temporal dynamics in a data-driven manner compared to their traditional counterparts (Ahmed et al., 2010). In particular, DL has gained tremendous popularity in recent times for its remarkable accomplishments in image classification, NLP, and reinforcement learning, as it can learn representations from complex data without needing 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). The creation of cryptocurrencies was motivated by the need to create a system that allowed simple, fast and cheap transactions that were 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 as a means of exchange of 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 inefficient transactions (Nakamoto, 2008). Buhalis et al. (2019) stated that transactions are much more methodical and straightforward as they can be used to prevent fraudulent exchanges and payments.

Of many, the most critical use of cryptocurrencies is for online payments. The usage of credit cards and other cashless payments has given rise to the emergence of cryptocurrencies, which are becoming the most popular form of payment on the internet (Wang et al., 2019). Pournader et al. (2020) state that companies can make instant money transfers by utilizing this, thereby reducing commission.

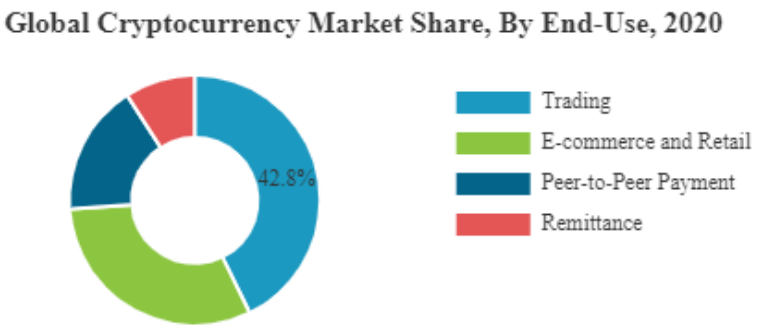


Figure 1: Global market share (Fortune Business Insights, 2021)

Specifically, Nica et al. (2017) state that the popularity of BTC can be credited to the low transaction fees compared to traditional payment services. However, for this to be maintained, Alonso-Monsalve et al. mention that trading must occur on a "*basis of different assumptions that may not hold in specific situations, including quick links between users, low transaction costs, and high liquidity*" (2020, p10). Nakamoto (2008) further mentions that the settlement time of BTC is much faster than any non-cash transactions, which may take days or weeks.

The use of cryptocurrencies is increasing daily, and it may come to the point where it replaces traditional currencies as credibility grows. However, it comes with its own fair number of challenges.

2.3.2.2 Challenges of cryptocurrencies

Given the absence of legislation and regulatory standards, certain risks have been introduced upon the rapid growth of cryptocurrencies that raise several concerns regarding virtual currencies' integration into the financial system (Avdeychik & Capozzi, 2018).

Although being decentralized has advantages, the absence of a governing authority has given rise to the development of online black markets. Due to its anonymity, it is even more challenging to trace the identity of a specific operation, user, or criminal activity (Baldimtsi et al., 2017). Kerr (2018) stated that BTC had been used as a tool for conducting business in the black market. Miller (2016) mentioned that cryptocurrencies had been used to assist in selling drugs, weapons, other unlawful goods, and even child pornography. Therefore, the growth could threaten certain activities, people's incomes, and even their lives (Scharding, 2019).

Nevertheless, it is worth mentioning that upon the exchange of cryptocurrencies for fiat, source detection is possible.

2.3.2.3 Why have cryptocurrencies taken the world by storm?

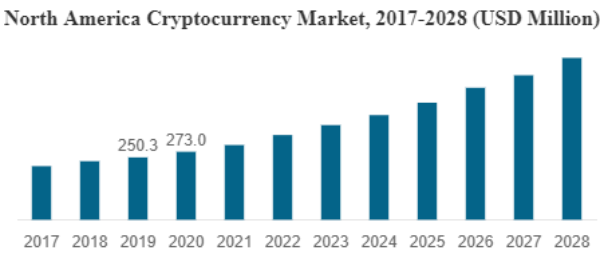


Figure 2: Latest trends (Fortune Business Insights, 2021)

As depicted in the above figure, the market of cryptocurrencies seems to increase each year steadily with the increasing popularity of digital currencies. The increasing popularity, in turn, gained the trust of the central bank, which established a patent, the Central Bank Digital Currency (CBDC), for digital currency projects across many developed countries. Multiple countries, including China, Thailand, Uruguay, and the Caribbean, have grown to support the CBDC for utilizing digital currency as a means of exchange, which could signify the explosion in popularity (Fortune Business Insights, 2021).

The above growth could grow exponentially after more developed countries get involved.

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 a 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 the general domain of 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

The domain of BTC forecasting has been gaining exponential traction within the past decade. Websites are making their historical data available to the public, which signifies stakeholders' need for forecasting models. Over time, research has been done to develop a SOTA forecasting model.

Fleischer et al. (2022) proposed a solution utilizing the LSTM neural network to learn patterns in close prices to predict the future and determine whether the closing price is enough to predict future prices. The resulting forecasts were accurate; however, the approach chosen was somewhat naïve because it was only a one-day prediction. Accurate forecasts could mean that

using only the close price for forecasts is a viable base model. However, as this is an open system, it is improbable that it is sustainable.

Yenidogan et al. (2018) proposed a solution utilizing the Prophet model by Facebook to distinguish between Prophet and ARIMA. It was identified that the other available features (ex: open, volume) in the used dataset could not be used as it would yield meaningless predictions. Having removed these other features, the authors proposed adding a fiat currency price feature due to its relationship with BTC, which resulted 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) identified that the deaths from COVID-19 and the price of BTC had a very high correlation.
- 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).

The above-proposed solutions do not consider external factors that could impact the price of BTC. Therefore, the primary concern is that they will adapt poorly to the ever-changing and unpredictable market, where a single tweet could affect the price drastically. Although some of the factors that could influence are out of the control of researchers (ex: country legislations and laws), the factors mentioned above can cause a significant impact. It is, therefore, essential to consider these factors when building such systems.

Abraham et al. (2018) proposed a solution considering Twitter and Google Trends data. They found that the tweet volume and Google Trends were highly correlated with the price and must be included in future systems. They also identified that the tweet sentiment are not very reliable as tweets about cryptocurrencies are objective. However, they utilized a linear model for predictions, which, as identified by Maiti, Vyklyuk and Vukovic (2020), yields less performance compared to non-linear models.

2.4.3 Open market forecasting

As cryptocurrencies fall under the open market, it is expected to help evaluate work under other open market forecasting domains (ex: the stock market). What could be determined is that there is a lack of research on this domain, which signifies that it does not show promising results (Li and

Bastos, 2020). However, relatively better success has been identified with the advancements in NLP and the large volumes of textual data available.

Picasso et al. (2019) proposed a solution for the stock market forecast utilizing the news data alongside historical prices. By applying sentiment analysis on the news data and combining it with the historical prices, they were able to create unison between the two schools of thought: technical analysis and fundamental analysis, which did result in a more robust solution – albeit not with an outstanding performance.

Kim et al. (2019) proposed a hybrid solution for forecasting power demand applicable to general forecasting problems. The hybrid model was implemented for ensemble potential via a bivariate learning approach. The authors also considered the external factors influencing power demand to generate a reasonable performance. It, however, is aimed at short-term power demand forecasting and is therefore limited in its application for the medium and long term.

Considering the available literature: open system forecasting is notoriously tricky, as it can depend on any external factor – forecasting the stock market is as reliable as palm reading. This is an example of one of those areas in that DL cannot be applied. Even though they are powerful algorithms that can produce performance and reliability more than humans, the obscurity of future events is a significant hindrance. It is also worth noting that the correlation between news and the stock market becomes more ambiguous when the future is considered.

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) proposed a solution using Twitter sentiment analysis for price prediction. They identified a decent correlation between Twitter sentiment and BTC's price and combined tweets' sentiment scores with historical data to predict the future price. However, as Abraham et al. (2018) identified, the tweet sentiment does not contribute as much as the tweet volume to the price of BTC, which the authors still need to include in their prediction model.

Serafini et al. (2020) explored statistical and DL models for the predictions utilizing sentiment available 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 not sufficient, and, as identified by Abraham et al. (2018), tweets are objective.

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 quite robust. The author believes that the optimal solution would be to utilize additional features as proposed by Abraham et al. (2018) 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 is an application of statistics to historical data to forecast what could happen in the future.

2.5.1.1 Autoregressive Integrated Moving Average (ARIMA)

The ARIMA model is, in a nutshell, a combination of the Autoregressive (AR) and Moving Average (MA) models. These models predict future values based on the past (Box et al., 2015). 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.
- It cannot be used for seasonal data.

2.5.1.2 Seasonal Autoregressive Integrated Moving Average (SARIMA)

SARIMA is an extension to ARIMA that was introduced to handle seasonality in data. Valipour's (2015) research 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.

2.5.1.3 Generalized Autoregressive Conditional Heteroskedasticity (GARCH)

GARCH is an approach to estimating market volatility while being more contextual than others when predicting prices or rates (Engle, 1982). Research conducted by Bhardwaj et al. (2014) identified that the forecasts attempted by GARCH were more accurate than ARIMA's as ARIMA cannot capture volatility in the dataset.

2.5.1.4 Prophet

Facebook introduced an approach to solving the challenge of forecasting at scale via adjusting parameters (Taylor and Letham, 2017). It was designed to forecast daily data consisting of weekly and yearly seasonality, considering the effects of holidays (Hyndman et al., 2021). 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.

2.5.2 Deep learning-based forecasting techniques

Studies have shown that certain TS forecasting domains perform better on statistical models (ex: Zhang et al., 2022). At the same time, other studies have shown that DL models have outperformed statistical models (ex: Siami-Namini, Tavakoli and Siami Namin, 2018). Therefore, it is also expected to be worth evaluating DL models.

2.5.2.1 Long Short-term Memory (LSTM)

LSTMs were introduced by Hochreiter and Schmidhuber (1997) to solve the issues in previous RNNs of not being able to store information. They paved the path for more performant future RNNs as they do not forget short-term patterns, which made modelling temporal dependencies for larger horizons possible (Lara-Benítez, Carranza-García and Riquelme, 2021). 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.

A prominent study among the many conducted by Sagheer and Kotb (2019) observed better performance compared to ARIMA and GRU.

2.5.2.2 Gated Recurrent Unit (GRU)

GRUs, introduced by Cho et al. (2014), 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 (Lara-Benítez, Carranza-García and Riquelme, 2021). There is no clear distinction between GRUs and LSTMs on which one performs better; therefore, generally, both are attempted.

A study by Kuan et al. (2017) demonstrated that the GRU performance was superior to ARIMA and LSTM.

GRUs and LSTMs are effectively similar in performance

As there is a contradiction in the work of Kuan et al. (2017) and Sagheer and Kotb (2019), 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 Maiti, Vyklyuk and Vukovic (2020).

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

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 (Oreshkin et al., 2020).

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 author to challenge the claim of Makridakis et al. (2018b): the future of TS forecasting is a hybrid of statistical and neural net models.

2.5.2.4 Temporal Fusion Transformer (TFT)

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 (Lim et al., 2019). 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: Li et al., 2019) 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 above literature, it can be deduced that 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.

2.5.3 Concerns about existing used techniques

2.5.3.1 Issues in statistical models

Based on the above literature, the following drawbacks can be identified with statistical-based models.

- Linearity
- Suitable for seasonal data
- Lack of interpretability and expressivity
- Altering the parameters requires domain knowledge

2.5.3.2 Issues in deep learning models

As identified by Makridakis et al. (2018b), 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 Makridakis et al. (2018b).

Given the open-ended conclusion on the comparison between statistical and DL models, the natural question is **how to proceed.**

2.5.4 Neural Ordinary Differential Equations (ODEs)

Neural ODEs 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 (Chen et al., 2019).

Chen et al. (2019) claimed that RNNs with continuous-time hidden states determined by ODEs are effective algorithms for modelling 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 RNNs & residual networks

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

Equation proposed by Chen et al. (2019)

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

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

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.

It is identified that 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 (Anumasa and Srijith, 2022).

Hasani et al. (2021) mentioned upon experimentation that the performance obtained by these neural networks is underwhelming compared to a simple LSTM network. Given this fact, the author has deduced that the requirement for review of other existing neural ODEs is relatively insignificant.

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

Hasani et al. (2020) proposed a solution to improve the underwhelming performance of neural ODEs. The proposed architecture exhibits stable and bounded behaviour 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 to incoming inputs (Ackerman, 2021). 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 Funahashi and Nakamura (1993) proposed. Instead of using a neural network to define the derivative (Chen et al., 2019), an additional term assists the system in reaching equilibrium.

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.

Equation proposed by Hasani et al. (2020)

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

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. Having understood the drawbacks of the other approaches and the proposed solution by Hasani et al. (2020), the optimal way to conduct this research is by utilizing the LTC approach.

However, the LTC architecture too has its limitations: it uses ODEs which are now considered obsolete (Duvenaud, 2021). Furthermore, they cannot adapt to rapid, instantaneous changes in incoming data streams.

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

2.5.6 Neural Stochastic Differential Equations (SDEs)

ODEs can be used to abstract deterministic models, as utilized by Chen et al. (2019) to propose neural ODEs. Hence, the natural question arises: **what could be used if there exists a state of randomness – in probabilistic models?**

Neural SDEs are similar to neural ODEs in that an ODE can be used as the solver; however, the additional focus is the 'stochastic evolution' instead of the 'deterministic evolution' (Tzen and Raginsky, 2019). Considering neural ODEs and their expressive power, SDEs can enhance the expressive power even further, as proposed by Peluchetti and Favaro (2019).

The important point worth noting about SDEs is that they are suitable for fitting data that can have an instantaneously little change (ex: market prices, molecule motion) (Duvenaud, 2021). As the chosen implementation domain belongs to this category, it is best to look into SDEs rather than ODEs going forward.

The bottom line – LTCs with SDE flexibility: Liquid Time-stochasticity (LTS)

Based on this detailed analysis, it is now unclear what the exact next step is. The LTC architecture proposed by Hasani et al. (2020) 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 with significant impact. This algorithm, dubbed the Liquid Time-stochasticity by the author, can be considered as the author's core contribution. The design will be presented in **chapter 4**.

2.5.7 Traditional required techniques

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

2.5.7.1 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. Tweets about stock trades and markets are mostly objective and have no genuine sentiment. Additionally, 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. 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 required to prevent unnecessary bias in each value. The Google Trends, historical data, Twitter volume, block reward size, and sentiments are numerical values that must be scaled to a standard range.

Feature selection

Feature selection can be conducted to select the optimal features to be an input for the model, ensuring that unneeded features with little to no impact can be ignored. Correlation is a technique that can be used to measure how much would a specific feature be of impact. Of the many available methods, the correlation matrix, 'Pearson R' – Pearson correlation coefficient, and the p-value are the primary methods of determining it (Abraham et al., 2018).

By conducting correlation tests, the author can determine whether specific features must be ignored when constructing the model, as they would not contribute as much and act as noise.

2.5.7.2 Hyperparameter tuning

None of the literature the author reviewed included a hyperparameter tuning stage; therefore, the author will 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, specifically tailored techniques such as the Keras Tuner and HParams dashboard in TensorBoard, could be used to identify the best set of hyperparameters.

2.5.7.3 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.

2.5.8 NLP techniques that can assist

2.5.8.1 Sentiment analysis

It is evident that 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 them alongside sentiment analysis. It is also worth noting that there may be the presence of duplicate tweets, which libraries such as fuzzywuzzy (Pant et al., 2018) could remove.

Recent research has recommended the usage of transformer architectures to apply these techniques (Wolf et al., 2020). However, the author believes that to extract the sentiment from the data, the VADER library is superior, as it can effectively extract sentiment in the context of social media (Hutto and Gilbert, 2014), and it is furthermore more tailored for Twitter data (Valencia et al. 2019). Nevertheless, both will be evaluated.

It is worth mentioning that specific cleaning steps must not be performed. These steps include the removal of punctuation, capitalization, and 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.

2.5.8.2 Named Entity Recognition (NER)

Upon conducting sentiment analysis, NER can be performed to adjust the weight (impact) of each tweet's sentiment depending on how influential the tweeter is or any organization being mentioned (ex: a tweet of Elon Musk could be given more weightage as a single tweet from him could cause a drastic impact).

*This technique was abandoned by the author in favor of a new formula they came up with (presented in **Chapter 6**).*

2.5.9 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.

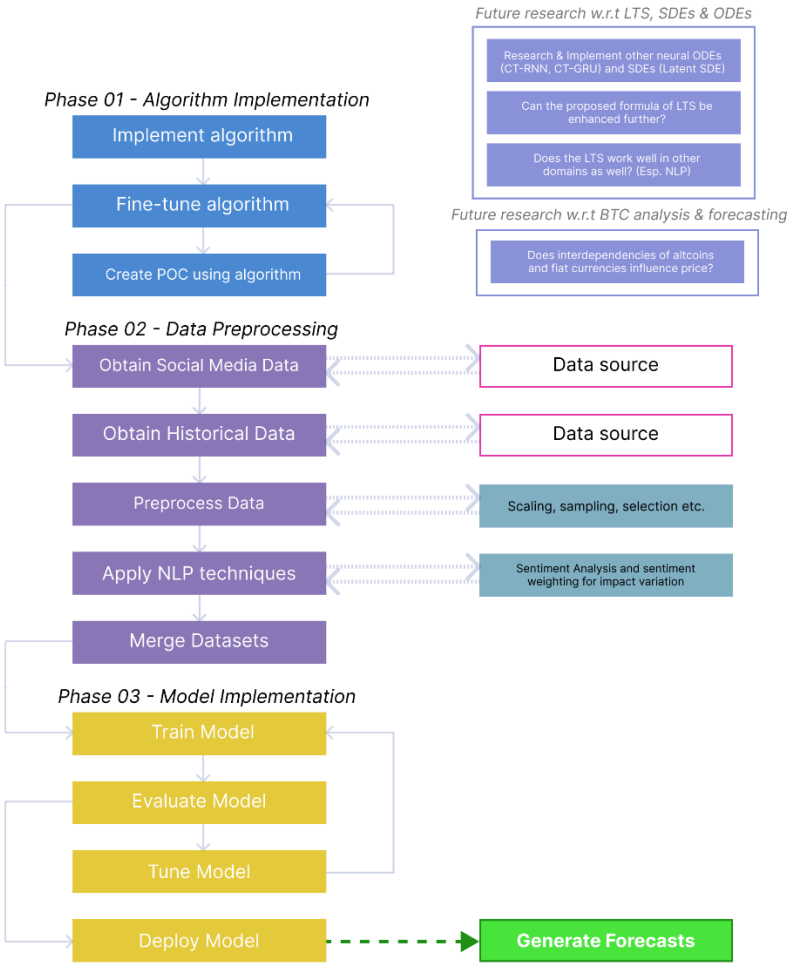


Figure 3: Proposed architecture (*Self-Composed*)

As identified, the architecture will use social media data to make the forecasting model as robust as possible; the data would be fetched from an API which could be Google Trends, Twitter tweets & volume, and the block reward size.

2.6 Evaluation

2.6.1 Evaluation approaches

Evaluation of the application can be conducted by validation methodologies and evaluation metrics based on 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 is implausible to get values of 0 – obtaining such errors is a clear distinction that something has gone wrong.

Table 2: Evaluation metrics for TS forecasting systems

Metric	**Description**	**Formulae**
MAE	The absolute difference between the generated forecast and the ground truths. Since it is simple to understand, it will give the author an idea of how inaccurate the forecast is.	$\frac{\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert}{n}$
MSE	Similar to MAE but computes the squared differences. This metric gives more emphasis to more significant errors and also considers outliers.	$\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$
RMSE	The square root of MSE.	$\sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$
MAPE	The ratio of the forecast to the average absolute difference between the forecast and ground truths. 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, can be used, which is an alternative to percentage errors (ex: MAPE) proposed by Hyndman & Koehler (2006) as there is a possibility for MAPE to '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.

Apart from 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.

Moreover, the results obtained will be publicly available to aid future research and benchmarking analysis.

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 wholly justify the author's proposed solution, two factors of evaluation must take place:

- **E**01 - evaluate the application against existing work of cryptocurrency forecasting.
- **B**01 - benchmark LTS against other SOTA TS algorithms.

As identified, LTC applications do not exist that could be compared. Therefore, the author will compare against the existing work done in forecasting cryptocurrencies to fulfil **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 complete background research process conducted by the author that was necessary to propose the problem identified. In detail, the research was provided in a birds-eye view by a concept map, broken down into subsections that were individually evaluated and critiqued in detail and broken down further, if necessary. Additionally, the author presented possible future research that open up upon the conclusion of this project.

CHAPTER 03. METHODOLOGY

3.1 Chapter overview

The methodology of conducting research must be defined to ensure that the research journey is as streamlined as possible. In this chapter, the author defines the methodologies chosen and discusses their reasoning in depth. Furthermore, the project requirements, risks that can be faced and their mitigation plans, schedule, work breakdown plan, and associated deliverables are presented.

3.2 Research methodology

Methodologies 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 **deductive** approach was chosen over the inductive since the final analysis and evaluation will be quantitative.
Strategy	**Archival research** and a**ction research** were chosen as the data collection strategy. 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. To assist the supplementary research taking place, **survey** was chosen to obtain insights from end users.
Choice	**Multi-method** will suit the proposed research project most since qualitative analysis would complement to the primary quantitative approach. However, it will not be used as a combination.

Time Horizon	The **cross-sectional** time horizon was chosen over the longitudinal time horizon. Even though the latest available data will have to be obtained often to update the model, there will be no interlinking between the times when the data is gathered as they will be independent.
Techniques and procedures	As a form of **data collection & analysis**, as many sources as possible will be used since there are finite resources. The primary mediums will be statistics, reports, journals, articles, and observations.

3.3 Development methodology

3.3.1 Life cycle model

Prototyping was chosen as the life cycle development methodology since heavy iterative building, development and evaluation are required.

3.3.2 Requirement elicitation methodology

The author will utilize **surveys**, **interviews**, and the knowledge obtained from available **literature** to gather requirements to implement the associated software produced.

3.3.3 Design methodology

Structured System Analysis & Design Method (SSADM) was chosen as the design methodology since it is easier to understand and maintain. These are essential factors given that the project will be developed over a considerably long period. Additionally, the selected software requirements (Python & React) do not promote Object Oriented Programming (OOP) in nature, but rather, functions and modules. It is also worth mentioning that it will have little discernible benefit in the case of a data science project.

3.3.4 Software development methodology

Structured programming will accompany the chosen design methodology to facilitate development using modules and functions.

3.4 Project management methodology

The author will follow a combination of PRINCE2 and Agile. The project will require many iterations and improvements since the implementation is novel and no reference exists. Alongside

multiple iterations, it is best implemented by being divided into multiple chunks and focusing on each chunk at a time with a plan-based approach.

3.4.1 Schedule

3.4.1.1 Gantt chart

The project's schedule is presented as a Gantt chart below.

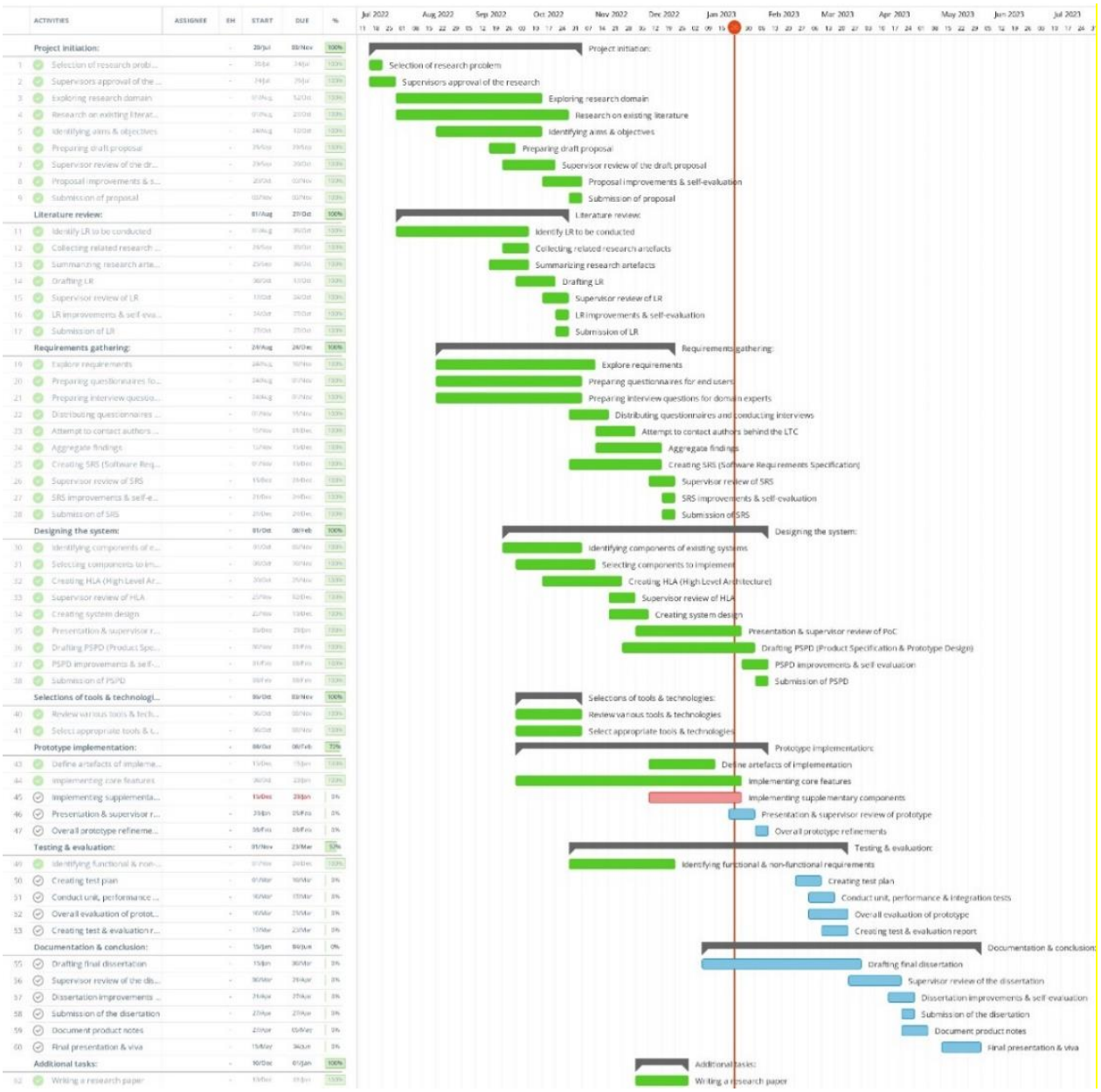


Figure 4: Gantt chart (*Self-Composed*)

A clearer version can be found [Here]

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 requirements

- **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 outstanding battery life and low power consumption.

- **Python /** R - to create the network & the respective model. Python, 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, due to its large developer community and seamless integration with Keras for higher-level API development. Additionally, multiple visualizations will be required, which is made simple by TensorBoard (Dubovikov, 2018).
- **Flask /** Node / Golang – for seamless communication between the client and the model. 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 option because of the author's familiarity and large community. (Boisdequin, 2020).
- **MongoDB** – to store the datasets and enable quick and efficient querying of the 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 /** Google Colab – development environment for building the forecasting model. Jupyter will be the primary choice as it has less risk: it runs locally. Therefore, in case of power failures, training would not be interrupted. Colab will be the backup choice if there is a requirement for a GPU to train the model.
- **Zotero** / Mendeley – manage references and research artefacts. Zotero is chosen due to the author's preference and is easy to use.
- **MS Office** | **Overleaf** – tools to create primary research reports. A combination of MS Office and Overleaf will facilitate the drafting of the 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 its advantages. Figma - to design & 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 & 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 requirements

- **Core i5 Processor (8th gen) /** Ryzen 5 / M1 – for long-running intensive workloads and managing multiple development environments. The author has access to an intel processor; therefore, it will be the CPU choice. However, the other options would suffice just as much.
- **8GB Ram or above** – to manage model training, multiple development environments & multitasking. Moreover, it is required to load large datasets and multitask.
- **Disk space of approx. 20GB** – 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

- Creation of TS forecasting systems.
- Knowledge of ODEs & respective solvers.
- Implementation of a raw neural ODE and SDE.
- Ability to create optimized & scalable DL models.
- 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 (ex: investing.com, cmcmarkets.com, finance.yahoo.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** – 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	Get insights from domain experts and, if necessary, the author of the proposed algorithm.
Corrupted documentation	4	4	Store all necessary documentation on the cloud as well as external storage.
Lose access to development code	5	2	Backup code on source control and cloud storage.
Inability to deliver all expected deliverables	4	2	Follow a list of priorities and deliver accordingly.
Invalid research assumptions	3	2	Continue researching since the final output is a research contribution regardless.

3.7 Chapter summary

This chapter defined the proposed methodology followed by the author and discussed its reasoning. In detail, the author discussed the research, development, and project management methodologies followed. Finally, the project requirements, work breakdown plan, associated deliverables, and the risks that 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

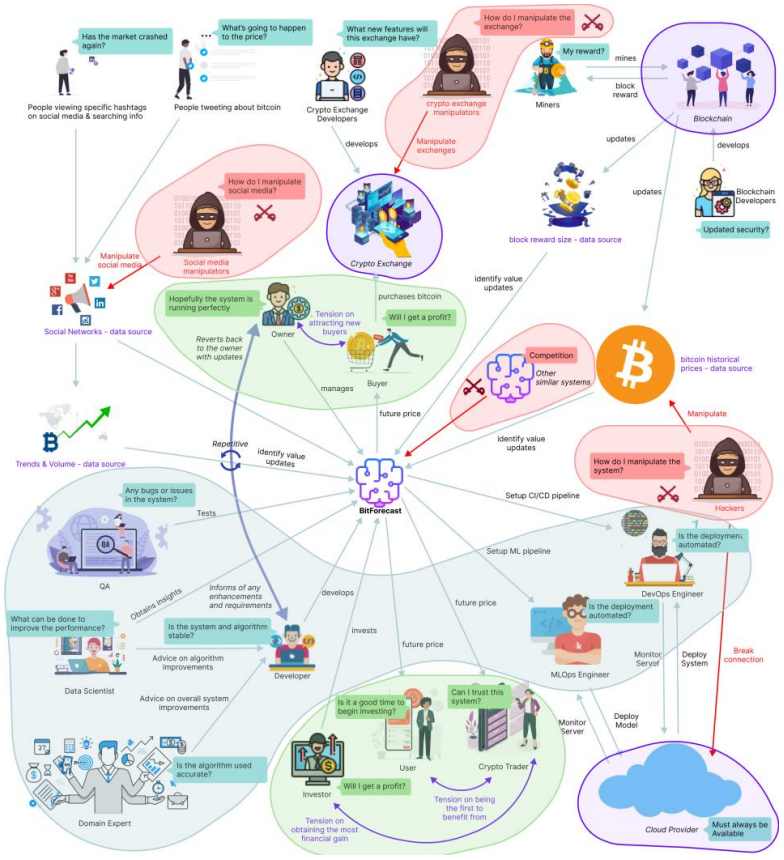


Figure 5: Rich picture diagram (*Self-Composed*)

A clearer version can be found [Here]

The above diagram illustrates a helicopter view of the wider environment, how specific stakeholders would interact with the system, and how they would benefit. Furthermore, the negative impacts, stakeholders' concerns, geographical localities, and stages of construction are identified, with knowledge the researcher could receive to improve the system.

4.3 Stakeholder analysis

The following section recognizes key stakeholders associated with the system, their relationships, and their respective roles. The stakeholder onion model depicts this information, and the stakeholder viewpoints further detail it.

4.3.1 Stakeholder onion model

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

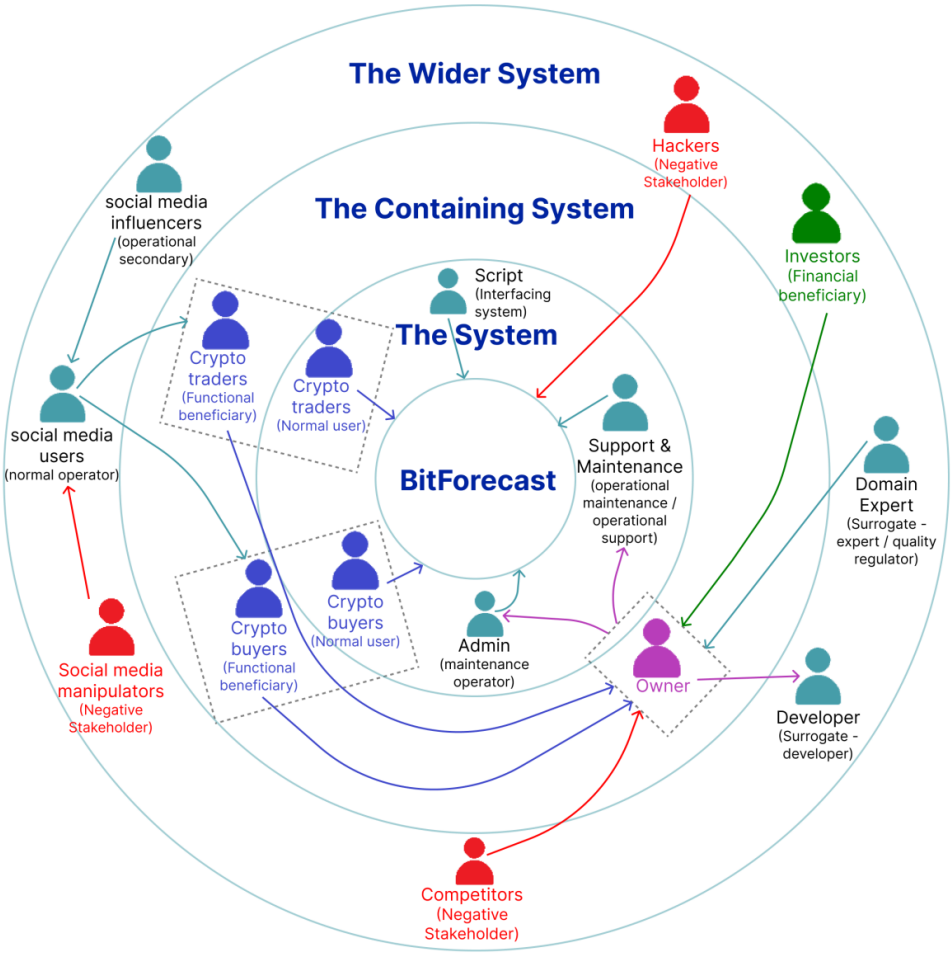


Figure 6: Stakeholder onion model (*self-Composed*)

4.3.2 Stakeholder viewpoints

The following table describes the stakeholders, their role 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	Fetches data and makes sure system is 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 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: 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. 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.

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 8: 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	Traditional neural ODEs were underwhelming in performance compared to existing neural nets.
Duvenaud, 2021	The proposed architecture by Hasani et al. (2020) uses the obsolete ODE, which lacks rapid adaptability - using an SDE instead can improve flexibility further. 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 9: Observations findings

Criteria	**Discussion of findings**
To find approaches to creating a neural SDE	The author noticed that POCs of neural SDEs are available sparingly and had yet to be utilized in an ML system like the proposed solution.

to implement the core research component	It is also safe to assume that building the research component could be later used as a baseline for future neural SDE implementations.
To find approaches taken to implement the additional component of BTC forecasting.	Although POCs of BTC forecasting systems that use LSTMs and statistical algorithms are available in abundance, what was noticed is that they all naively utilize only the closing price as a feature or the closing price with the Twitter sentiment. Considering this, the author decided to build the primary research component first so that the algorithm could be used to build ML systems and create the additional BTC forecasting system utilizing as many exogenous features as possible that can be of effect.

4.5.3 Interviews

Interviews were conducted to obtain domain expertise and any information that the author may have missed and 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 expertise areas are available in **APPENDIX C.2** with necessary evidence for the respective finding.

Table 10: Interview thematic analysis codes, themes & conclusions

Code	**Theme**	**Conclusion**
Research component		
Algorithm architecture	Research Problem & Gap	The interviewees validated the research gap and the defined problem. They were also happy that the author had been conducting this research, as only a few papers were published in this domain.
Resource intensive	Requirements	The interviewees were concerned that ODEs and SDEs could be expensive to compute and hence could take some time, which can be an issue given that the forecasts must be produced quickly. Therefore, the author must optimize the model as much as possible to avoid user-unfriendliness.

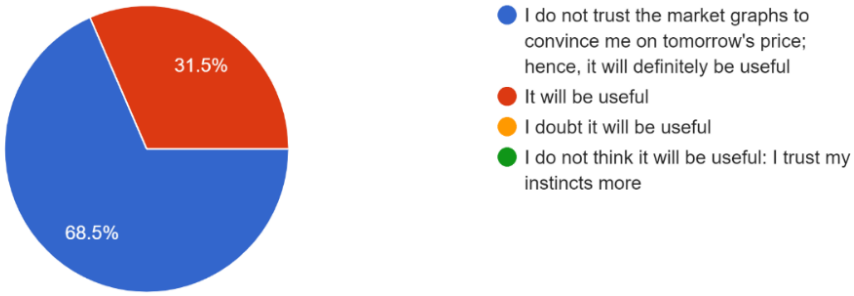
Obsolete, Inflexible	Advice	The author had initially planned on only creating an implementation of the LTC architecture proposed by Hasani et al. (2020). However, the author could further improve the architecture by using SDEs instead (the base LTC uses ODEs), (which manifested into the LTS), which is the author's current aim as it carries more significance and a potentially more outstanding contribution.
Visualizations, Explainability	Other suggestions	What was concluded here was that XAI is primarily present for image classification, and there needs to be more literature on the TS domain. However, XAI integration into TS modelling could be confusing and complicated due to the temporal component. Additionally, XAI for SDEs needs to be researched, which the author could look into if time permits.
Problem domain		
External features and trends	Robustness	The interview was an additional validation for the data collected in the survey. Most suggestions were to use as many extra features as possible to make the model robust. Therefore, the author will ensure that they utilize the mentioned exogenous features.

4.5.4 Survey

A survey was conducted to gather requirements from the target audience to infer functionalities to implement for the supplementary product developed.

Table 11: Survey analysis

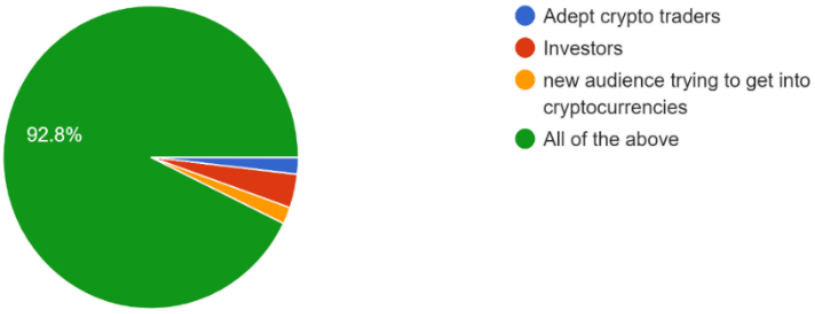
Question	How much would a system capable of assuming tomorrow's price benefit you?
Aim of question	To identify whether the system is beneficial in the first place
Findings & conclusions	



All the participants believed that the proposed system would be beneficial – where the majority had a greater belief than others. Having obtained this information, it is evident that the supplementary proposed system will be helpful. As identified, not a single participant thought that the system would not be beneficial. Notably, this validates the problem domain and gives the author the 'green light' to go ahead.

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

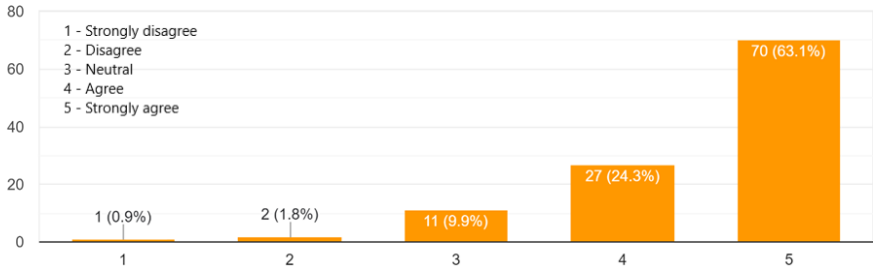
Findings & conclusions



The majority of the participants believed that the system would be beneficial for expert traders, investors as well as a new audience. However, what can be identified, is that a minute portion of participants assumed that the system would be helpful primarily for people who are already involved in the market – this is some evidence that the system must be made as simple as possible to attract a newer audience. It is also identified to help only a new audience – this is evidence that the system must not be immature.

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

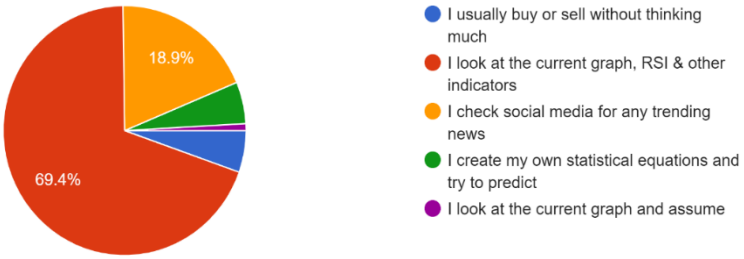
Findings & conclusions



The responses to the above question show that the system will also apply to audiences who are not cryptocurrency experts. This question goes hand in hand with the previous question to confirm whether the system can target a newer audience of people to get into cryptocurrencies rather than just focusing on a niche audience who are experts or current investors/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

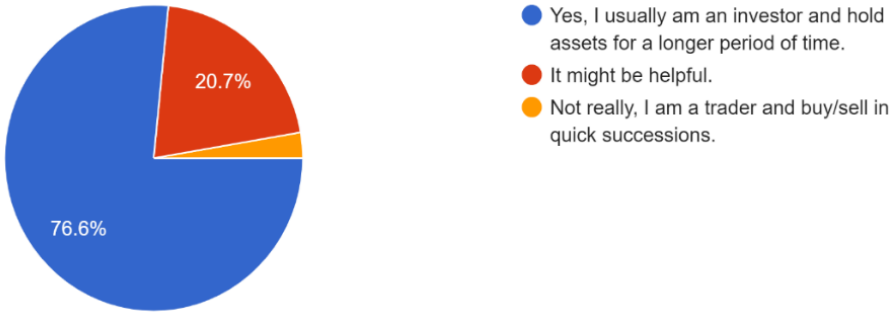
Findings & conclusions



The responses to the above question are more of a 'Know Your Customer' question with no specific project-related purpose. Nevertheless, what can be identified is that most of the respondents have some knowledge of cryptocurrencies, where almost 70% are experienced in trading/investing cryptocurrencies – a great insight as nearly all the respondents have specific knowledge. Therefore, the author could use this to reach out to the respondents (whom they gathered requirements from) 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

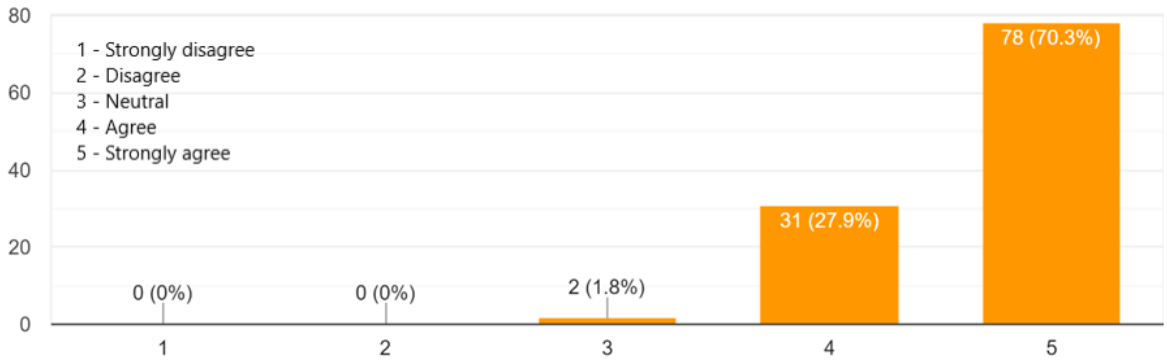
Findings & conclusions



The author initially considered only having a single horizon forecast, considering the limited time. 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

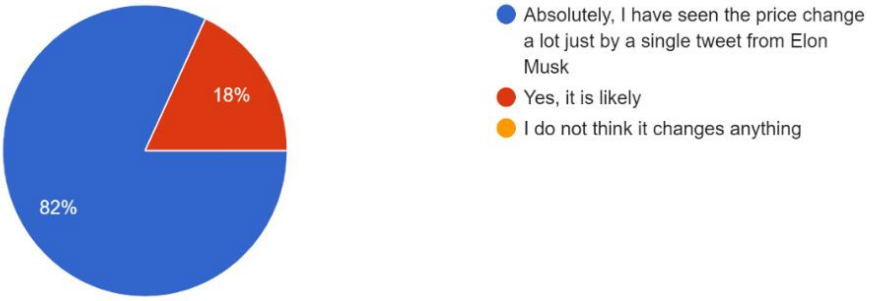
Findings & conclusions



The majority of the respondents believe that social trends impact the price. Therefore, it is necessary to consider as many trends as possible. Considering the project's limited time and scope, the author has decided to use Twitter volume and Google Trends; however, Reddit, Facebook, and others would also provide insights and could be considered 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.

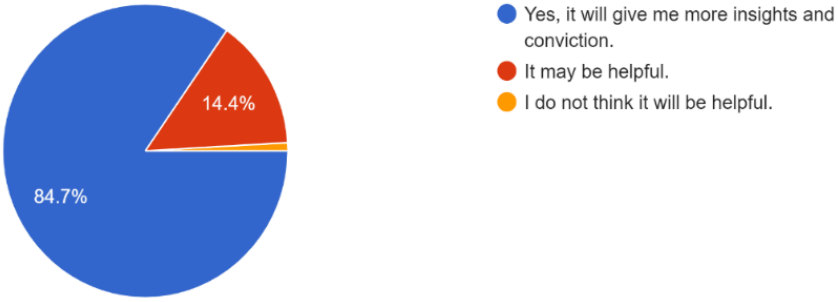
Findings & conclusions



All participants believe that the current thoughts on social media affect the price in one way or another. Most participants further thought that the tweeter's influence adds additional significance. Considering this and the previous question, it is apparent that the mentioned social factors contribute to price changes, which validates the problem domain contribution. Additionally, based on the responses, the requirement to give more weightage to specific tweeter's sentiments (based on their influence) is apparent.

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



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 while conducting prototyping, it became evident that a single-point prediction is likely to be less valuable than a range of prices. A point prediction is implausible to be accurate, which makes the requirement of uncertainty estimates more vital.

Question	What functionalities would you expect to have in a bitcoin forecasting system?
Aim of question	To identify any additional requirements

Findings & conclusions To analyze opened ended questions, the author can perform thematic analysis. The analysis, including the theme and related codes, is available in **APPENDIX C.3**. Based on the analysis conducted, it is evident that the participants would appreciate some Explainability. Including XAI is an addition that the author could look into if time permits. The participants also mentioned that the system would be better performant and robust if it utilized as many exogenous factors while making it as simple as possible. Based on these findings, the author will aim to include as much Explainability as possible and make it mandatory to use the mentioned exogenous features.	
Question	Any extra feedback you would like to provide?
Aim of question	No specific reason – is mainly used to obtain any additional feedback
Findings & conclusions The respondents submitted a few motivational sentences to inspire and motivate the author to perform their best.	

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 conducing prototyping and its respective findings.

Table 12: Prototyping findings

Criteria	Discussion of findings
To explore the feasibility of creating the primary research component.	Building the algorithm was intimidating, as no proper reference exists. The author realized that, alongside traditional DL theories, implementing the algorithm required more profound knowledge and understanding of SDEs and differential solvers. As such a direct implementation was not possible. The author therefore implemented the LTC architecture first proposed by Hasani et al (2020) and then built upon it to develop the LTS algorithm.
To explore the feasibility of	The author had depended on the Twitter API to get tweet sentiment of specific days; however, this was impossible as Twitter had updated the

creating the BTC forecasting application.	API only to provide tweets of the past seven days. Fortunately, there were public datasets available up to a certain point in time; therefore, they had to use a third-party library to scrape tweets of dates ahead of that point in time. Moreover, upon experimentation, they gained an epiphany that solely the point price prediction would be useless; instead, a range of uncertainty estimations that provide a range of values would be more helpful. Furthermore, any explainable insights from the networks can be valuable to provide intuition into the forecast generation.

(The rest of this page is intentionally left blank)

4.5.6 Summary of findings

The below table summarizes the findings of all the techniques performed to gather the requirements for the MVP.

Table 13: 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	Try to integrate latent SDEs into an LTC architecture for a novel algorithm implementation instead of using the same obsolete latent ODE.				✓	✓
Problem domain						
5	The system will be of use to experts and new audiences.			✓		
6	Social trends can be a source of impact.	✓		✓		✓
7	Well-known influencers' opinions cause a more drastic impact.	✓		✓		
8	A system combining all exogenous features in a non-linear model has yet to be explored.	✓				
9	Including a range of prices than a point price is an added advantage and can produce more credibility.			✓		✓
10	Implementing an Explainability component will drastically make the system more credible.			✓		✓
11	A system capable of changing its hyperparameters would make it worthwhile for experts.			✓		

4.6 Context diagram

The following diagram depicts the system's boundaries and interactions. Determining them before the development will provide the author insight into how the information should flow.

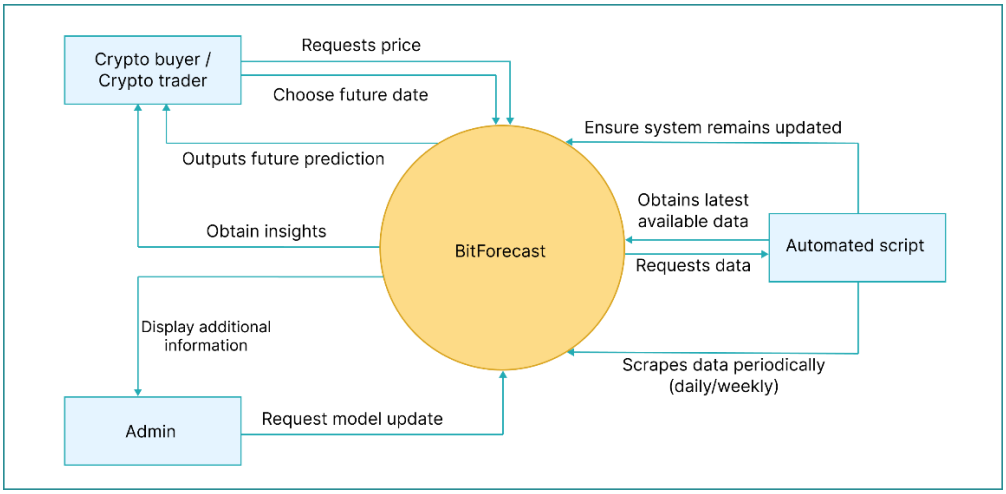


Figure 7: Context diagram (*Self-Composed*)

4.7 Use case diagram

The below diagram demonstrates the "sea level" use cases of the proposed system, describing the functionalities at a high level the system will provide end users with.

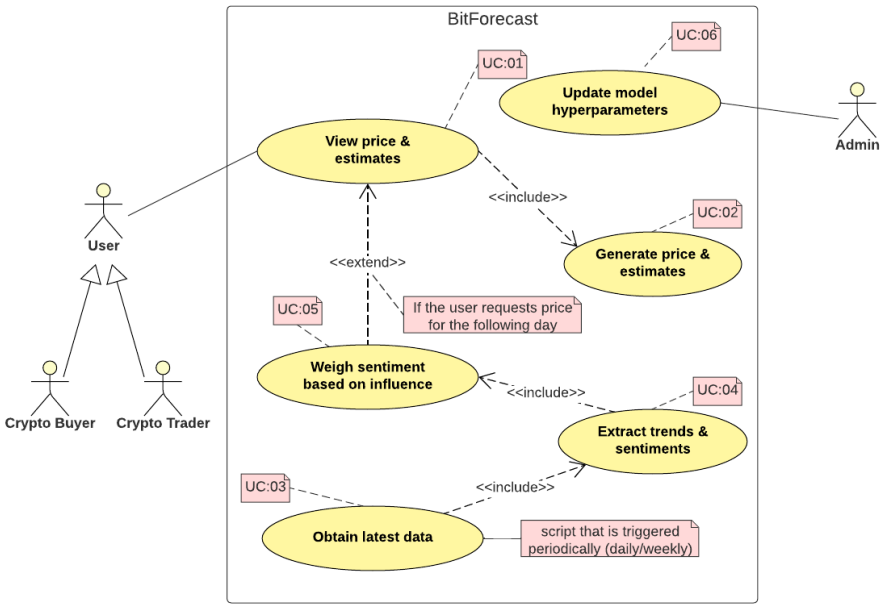


Figure 8: Use case diagram (*Self-Composed*)

4.8 Use case descriptions

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

Table 14: Use case description UC:01; UC:02

Use case	Display price & estimates
Id	UC:01; UC:02
Description	Display future prices and their respective uncertainty estimations based on the user's choice of date, alongside any Explainability insights.
Actor	User
Supporting actor (if any)	None
Stakeholders	Crypto buyer, crypto trader
Pre-conditions	All the data must be scraped and preprocessed, and the forecast should have been generated.
Main flow	MF1. User requests tomorrow's price. MF2. The system recognizes the need to utilize available exogenous features. 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. Obtains the latest available data. 2. Performs sentiment analysis and self-retrains. MF4. The system generates price and upper and lower estimations. MF5. Display output to the user along with any insights.
Alternative flows	AF1. The user requests the price for a date ahead of 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 along with any 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.

4.9 Requirements

4.9.1 Functional requirements

The functional requirements were determined based on priority using the 'MoSCoW' technique, which is detailed in **APPENDIX C.5**.

Table 15: Functional requirements

ID	Description	Priority	Use Case
Research level			
FR1	A robust and scalable implementation of the novel algorithm must follow recommended standards.	M	-
FR2	The developed algorithm must be able to be 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:03
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:04
FR7	The script must extract trends and sentiments from obtained data.	M	UC:05
FR8	Users should be able to view a range of prices along with the single-point price.	S	UC:03
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:03
FR11	The script could weigh sentiment based on any influential personnel's tweet.	C	UC:06
FR12	The system could display some insights to the user, such as a highly influential tweet that made it predict the price.	C	UC:03

FR13	Admins could authenticate and update the model with different parameters.	C	N/A
FR14	Admins could get additional information about a prediction, such as the evaluation metric and accuracy.	C	N/A
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

4.9.2 Non-functional requirements

The author prioritized the non-functional requirements based on the following two levels:

- Important – best to have them.
- Desirable – better to have them.

Table 16: Non-functional requirements

ID	Requirement	Description	Priority
NFR1	Performance	The system must take little time to generate a forecast, given that a couple of extra features are in use.	Important
NFR2	Performance	The system must not unnecessarily keep updating its data.	Important
NFR3	Usability	The user interface must be simple and effective and provide user-friendly errors if any occur.	Important
NFR4	Maintainability	The author must document the codebase well in case of future reference, mainly the algorithm development repository.	Important
NFR5	Quality	The output must be of good quality so that it provides vital insights.	Desirable
NFR6	Scalability	The system must be deployed to a cloud with no scaling issues and good resources for efficient and optimal performance, especially as there could be multiple concurrent active user requests.	Desirable
NFR7	Security	The system must be resilient to attackers, specifically to prevent data manipulation.	Desirable

NFR8	Compatibility	To ensure compatibility, the developer must test the system on most browsers and mobile phones.	Desirable
NFR9	Availability	In critical failures, the primary operator must be available and solve issues as soon as possible.	Desirable

4.10 Chapter summary

In this chapter, the author defined necessary stakeholders interacting with the system and described how the interaction would occur, visualizing this using a rich picture diagram and Saunder's Onion model. Additionally, requirement elicitation techniques, their reasoning, and their respective findings were discussed and presented. Finally, the author specified the use cases, associated descriptions, and system requirements.

CHAPTER 05. SOCIAL, LEGAL, ETHICAL & PROFESSIONAL ISSUES

5.1 Chapter overview

This chapter outlines any social, legal, ethical or professional issues that arose during the journey of this project and the actions that were taken to mitigate them.

5.2 SLEP issues and mitigation

The below table breaks down any identified SLEP issues and how they were mitigated.

Table 17: SLEP issues & mitigation

Social	**Legal**
Names of the interviewees are not included in the dissertation. Personal details were not collected by the conducted survey, and the respondents were kept anonymous. Information obtained are not saved elsewhere.	The datasets used are all available freely for the public and no names were recorded for the case of Twitter tweets. Utilized tools and technologies are open source or are provided to students free of charge. The codebase is available in GitHub with the open-source MIT license.
Ethical	**Professional**
Participants were clearly informed on what would the data be used for. The work presented in this report is genuine and not reproduced elsewhere. Any ideas taken are clearly cited and given credit.	Professional standards were followed as much as possible. Responses and evaluations received are not tampered with and the limitations are clearly stated.

5.3 Chapter summary

The chapter detailed the social, legal, ethical and professional issues the author faced while conducting this research and the steps they took to solve them.

CHAPTER 06. DESIGN

6.1 Chapter overview

Prior to implementation, detailed designing is required to ensure that implementation is as seamless as possible. In this chapter, the author focuses on selecting suitable architectural structures for implementation, considering the gathered requirements. Specifically, high-level, low-level, and associated design diagrams are presented alongside necessary UI wireframes. Moreover, the LTS architecture and a novel tweet sentiment weighing formula are also presented.

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

ID	Goal	Justification
DG1	Performance	A typical flow in TS forecasting requires retraining the model whenever a prediction is made, as the data the model had been trained on could be outdated. However, as multiple features are being used in the proposed system, this can severely hinder performance. The author can avoid this by storing past data and only fetching needed data when necessary; as a further step, the data can be fetched periodically.
DG2	Usability	Based on the analysis obtained during the requirement-gathering phase, there were mixed thoughts on whether the application would benefit people who are not experts in cryptocurrencies. Therefore, this requirement is mandatory as it is crucial to create a system that is as user-friendly as possible to be used by users across all levels of expertise.
DG3	Quality	The output must be of the highest possible quality. Also, as identified in the gathered requirements, the system must display a range of prices to provide more conviction.

DG4	Maintainability	As implied by the author, the research must yield two products for the project to be successful. The goal of maintainability is solely for the research product proposed. The architecture of the algorithm must be optimal and independent to be able to be used as a reference for future research.

6.3 System architecture design

6.3.1 Architecture diagram

The system's high-level architecture design is depicted below. The author chose a three-tiered architecture because of the distinct separation of concerns of the presentation, logic, and data layers.

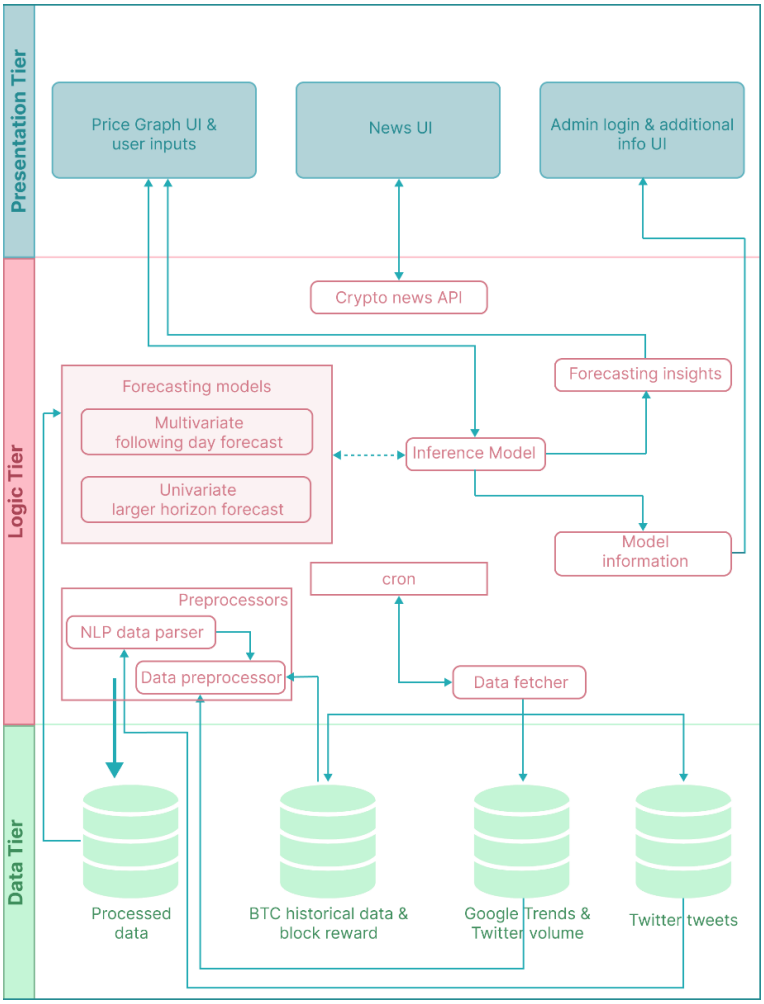


Figure 9: Three-tiered architecture (*Self-Composed*)

6.3.2 Discussion of tiers of the architecture

Data Tier

All data in this layer are fetched from an API and stored in individual collections in MongoDB to ensure updated data is available whenever necessary.

- BTC historical data & block reward – historical data of BTC closing prices of the past several years and the associated block reward obtained for mining BTC.
- Google Trends - historical data of the number of searches made each day that are BTC-related.
- Twitter volume – historical data of the number of tweets posted each day that are BTC-related.
- Twitter tweets - historical data of the tweets posted that are BTC-related.

Logic Tier

The logic tier consists of the base logic performed on the data in the data tier to provide an output in the presentation tier.

- Preprocessors – consist of code required to process the raw data fetched from the API's so that the forecasting model can use it.
    - Data preprocessor – required for general preprocessing steps such as normalization and cleaning.
    - NLP data parser – required to perform sentiment analysis on the tweet data and give more weightage to a specific tweeter's sentiment.
- Data fetcher & cron – the automated scheduler that the script will run periodically to ensure that the data and model are up-to-date.
- Forecasting models – models that will be used to 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 – extra information of the model that the admin could view (ex: accuracy, no. of epochs).

- Forecasting insights – additional information presented 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 that is 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, further, provide the ability to retrain the model by adjusting hyperparameters in use.

6.4 Detailed design

6.4.1 Choice of design paradigm

As identified in previous chapters, the choice of design paradigm is SSADM. To re-elaborate, as this research is primarily focused on developing a novel architecture with a novel algorithm, extensive experimentation is paramount. Furthermore, the selected programming languages do not promote OOP; instead, they encourage using function-based modules and components.

6.5 Design diagrams

6.5.1 Data flow diagrams

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.

6.5.1.1 Level 01 data flow diagram

The level 01 diagram is an extensive breakdown of the core components proposed in the context diagram.

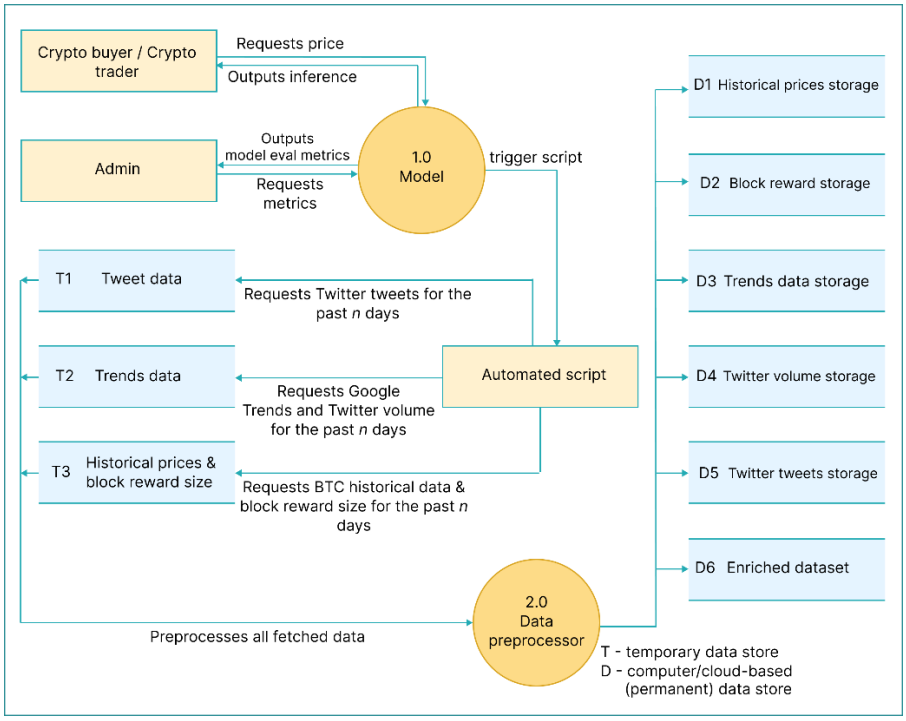


Figure 10: Data flow diagram - level 01 (*Self-Composed*)

6.5.1.2 Level 02 data flow diagram

The level 02 diagram is a more extensive breakdown of the core data preprocessor component proposed in level 01.

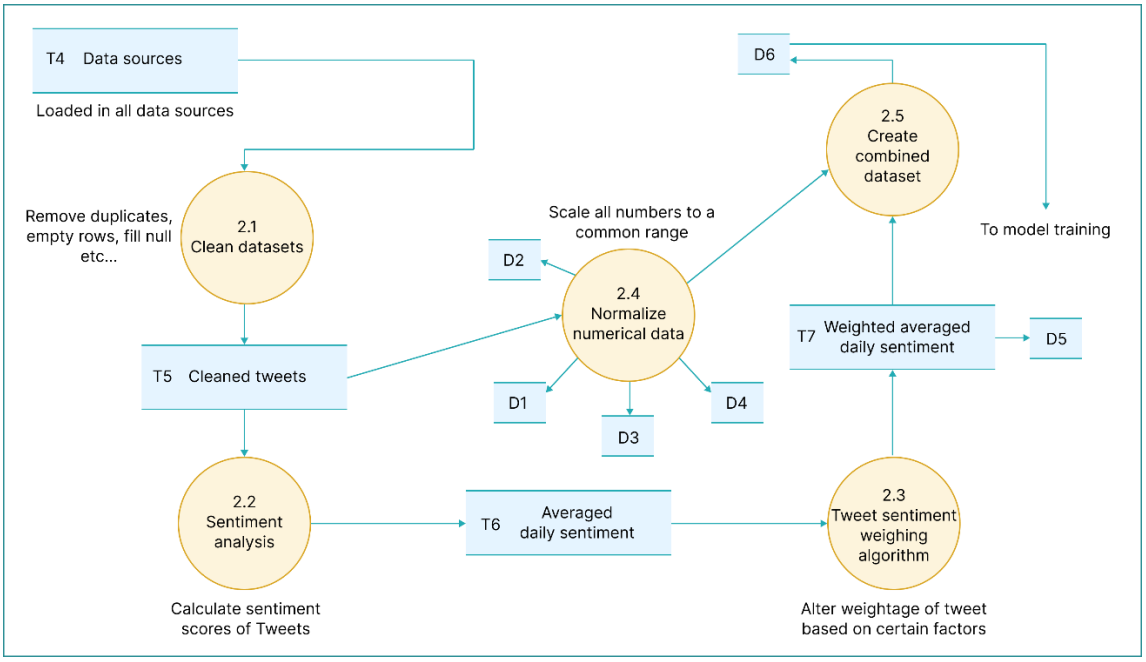


Figure 11: Data flow diagram - level 02 (*Self-Composed*)

6.5.2 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 that there was no clear technique to weigh tweet sentiments based on the tweeter's influence. The author therefore will propose another novel algorithm to weigh these sentiments - this can be considered as a secondary contribution.

6.5.2.1 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 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:

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

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 SDE solvers

Hasani et al. (2020) 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 Hasani et al. (2020) determined, the architecture is a system of stiff equations. Therefore, as Press et al. (2007) decided, researchers must use an implicit solver to ensure stability. Additionally, researchers can combine an explicit solver to achieve further stability. Therefore, the author will use an SDE solver, which is implicit, and if time permits, create a further enhanced custom SDE solver by fusing an explicit solver within.

Based on the author's research, the SDE equivalent for ODE Euler methods is the Euler-Maruyama method; this is the recommended solver as it can handle all forms of noise (Li et al., 2020). Combining the explicit Euler-Maruyama solver within to create a custom solver is something researchers should explore in the future.

How to train the network?

Training these networks has a trade-off between accuracy and memory. Chen et al. (2019) promoted the use of the adjoint sensitivity method to perform reverse-mode AD, which is more memory efficient. Hasani et al. (2020) mentioned that this method introduced more numerical errors and opted to use the traditional BPTT approach, which is more accurate but consumes more memory. Although there exists a technique of adjoints specifically for SDEs, they cannot be used, as determined by Tzen and Raginsky (2019), and hence requires a custom-built backpropagation rule.

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. Researchers must investigate reverse-mode AD in the future as it is the recommended approach when memory efficiency is more important. It is also worth noting that using the BPTT approach carries added benefits, such as being able to be used as an RNN layer alongside the popular optimization algorithms that are very familiar (ex: Adam, SGD) (Hasani et al., 2020).

6.5.2.2 Tweet sentiment weighing algorithm

Based on the requirements received from end-users, it is evident that the impact of tweets depend 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 engagement of the tweet itself. 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 are not directly correlated and as such were not considered at this point in time. 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**.

6.5.3 LTS algorithm analysis

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

The traditional BPTT approach yields more accurate results, with the trade-off of consuming more memory, whilst reverse-mode AD acts the opposite (BPTT has **high** backward accuracy, but **O(L)** time and memory complexities compared to **O(LlogL)** time complexity and **O(1)** memory complexity in reverse-mode AD). Therefore, to obtain the best result possible, the author chose the approach of the traditional BPTT.

6.5.4 UI design

The author had decided to implement a web application for the supplementary application being built due to convenience. The low-fidelity wireframes designed to be of use are available in **APPENDIX D.4**.

6.5.5 System process activity diagram

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

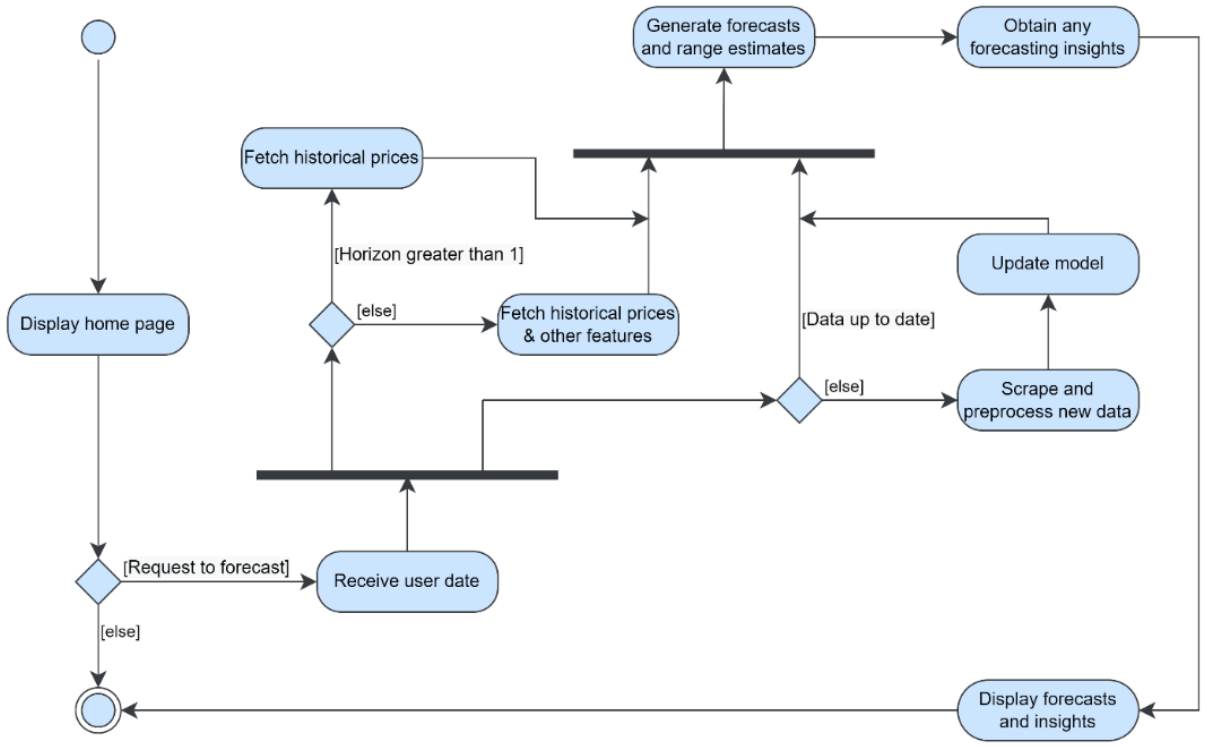


Figure 12: System process activity diagram (*Self-Composed*)

6.6 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 weighing algorithm is also presented. Additionally, the chapter illustrated the system's design, architecture, data and system flow diagrams, and wireframes that would demonstrate them in the end application.

CHAPTER 07. IMPLEMENTATION

7.1 Chapter overview

Upon designing 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 necessary decisions taken to approach that implementation. Moreover, the chosen tools, languages, and technologies are presented with their reasoning for being chosen 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.

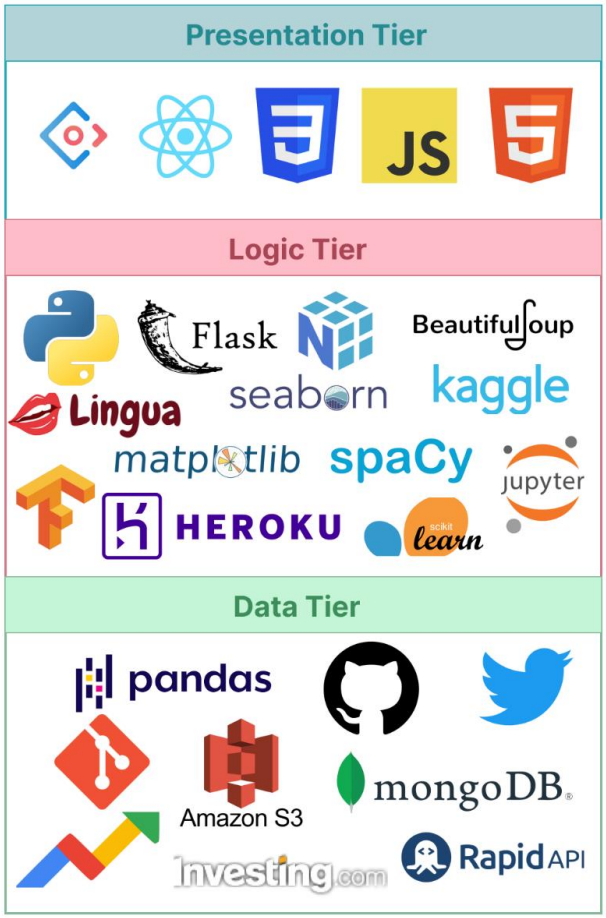


Figure 13: Tech stack (*Self-Composed*)

7.2.2 Selection of data

Table 19: Dataset sources

Dataset	Source
BTC historical data	From a third-party investing.com API.
BTC block reward size, BTC Twitter volume, BTC Google Trends	From a public dashboard that provides multiple different information about a specific cryptocurrency.
BTC tweets	Tweets from 2014-2019 were downloaded from Kaggle – the remaining dates were extracted from a Twitter tweet scraper.

The multivariate single-horizon forecasting model combined the above data, while the univariate multi-horizon forecasting model solely used the historical data. Gathering the data was long and arduous as it was not as simple as downloading available datasets, and specific APIs being rate-limited. Dedicated python scripts were written to extract the data, store it into a MongoDB database, and to streamline updating data. The author will publicize these scripts and the data to facilitate future research.

7.2.3 Selection of programming language

Programming languages were analyzed before development. Specifically for three main aspects: the client, the data science component, and the API communicating between the model and the client. Based on the analysis conducted in **APPENDIX E.1**, the author decided to use **Python**, as it was more relevant.

To develop the user interface, not much competition is present to analyze. **JavaScript** is the stand-alone leader and is the author's choice, as it is dynamic and can handle user interactions seamlessly. Although recent technology has presented the usage of C# for frontend development, high latency issues and lack of community knowledge are a downfall.

APIs are required to set up the communication 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.

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**, where the author opted to use **TensorFlow**, as it provides low-level details.

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 decided to use **Flask**, as it is used for only exposing the models.

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 dataframes to perform analysis, cleaning, transformations, filtration, etc., on the datasets.
Scikit-learn	To create data splits and feature scaling.
Lingua	To detect the language of the tweets. As this project is limited to using only English tweets, they must first be identified.
SpacCy	To perform NER to extract entities that could be within the pre-defined impactful index.

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 whenever necessary.
AWS S3	Store the models in use.
Heroku	To host the Flask API.

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	Consists of 32GB of RAM; therefore, all datasets can be loaded and processed at once without needing to process sections of data at a time. Additionally, it provides easy integration with existing Kaggle datasets and user-uploaded datasets.
Jupyter	For local trials, testing, and model training.
VSCode	Lightweight and extremely powerful. It consists of multiple 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 sentiment analyzer, Redux, Ant design, MongoDB, Heroku
IDEs; Version control	Kaggle and Jupyter notebooks; VSCode. Git + GitHub

7.3 Implementation of core functionalities

The novel algorithm, the scripts to fetch the required data, and the preprocessing performed can be considered the core functionalities of the project.

7.3.1 Algorithm implementation

The author initially implemented the LTC architecture since there is no modern reference utilizing recommended best practices and approaches. The author then built on this architecture, replacing the underlying ODEs with SDEs.

```
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 14: Initialize algorithm (*Self-Composed*)

The above code snippet initializes the algorithm cell with the necessary variable maximum and minimum values. In the above method, the built model can perform input-independent initializations. By inheriting from the base Keras Layer class, the ability to be used in the higher level of the model's layer definition is obtained (as existing LSTM and RNN cells).

```
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 15: 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 (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 16: 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 17: 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_inputs(self, inputs):
  '''
  Maps the inputs to the sensory layer
  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 18: Algorithm – define weights and biases (*Self-Composed*)

As determined in chapter 6, the optimal way of performing the forward computation of SDEs is to use the Euler-Maruyama method. The below code snippet is an implementation of the Euler-Maruyama SDE solver used by the author utilizing Brownian motion as the noise, as demonstrated by Duvenaud (2021).

```
@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 19: Algorithm – Euler-Maruyama SDE solver (*Self-Composed*)

7.3.2 Data fetchers

The scripts that are used to extract the data. The scripts are placed under **APPENDIX E.5**.

7.3.3 Preprocessing

Preprocessing steps are required to prepare the data fetched from the data fetchers before being used by the model. The preprocessing scripts are placed under **APPENDIX E.6**.

7.4 User interface

Screenshots of the final GUI are placed under **APPENDIX E.7**.

7.5 Chapter summary

This chapter focused on defining the technologies and tools that facilitate the software development that would 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. In this chapter, detailed testing is performed 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 model's performance is as performant as it can be.
- Ensure that the functionalities implemented align with the MoSCoW's technique of "Must have" and "Should have".
- Identify any bug fixes/improvements that must/can be applied to the application.
- Identify if the important non-functional requirements are met.
- Conduct baseline-benchmarking so that the system can be benchmarked against in the future.

8.3 Testing criteria

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

- Functional testing – focused on determining how well the system performs and meets the functional requirements.
- Structural testing – focused on evaluating up to what extent were the non-functional requirements met.

8.4 Model testing & evaluation

8.4.1 Model testing

Two ensemble models were implemented: univariate and multivariate. Both models were tested in a similar fashion by setting a "pseudo-future" from a specific time frame.

A pseudo-future is a "fake" future where we know what the actual value is at that point in time.

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

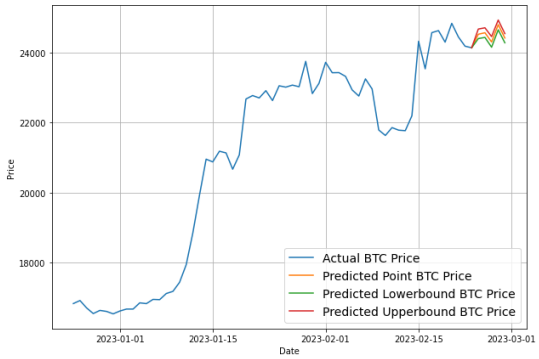


Figure 20: Univariate model testing (*Self-Composed*)

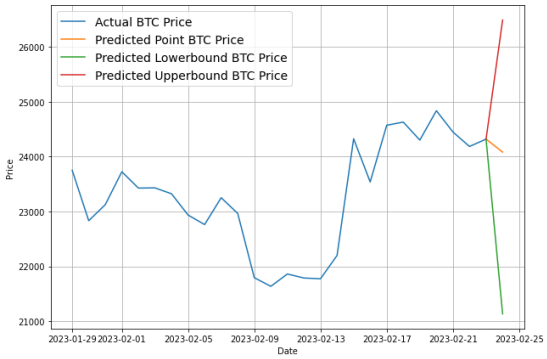


Figure 21: Multivariate model testing (*Self-Composed*)

The three predicted lines are from the implemented ensemble. 30 different models were created and trained on, and in turn 30 different predictions were proposed. The "predicted point BTC price" comes from the model with the median prediction and the upper-bound and lower-bound comes from the models with the highest and lowest price predictions respectively.

Prior to implementing the final ensemble, a single model was trained to determine whether the architecture is optimal. The below graph illustrates the loss curve recorded in TensorBoard; what can be noticed is that the model has overfit, which is inevitable as there are limited data samples.

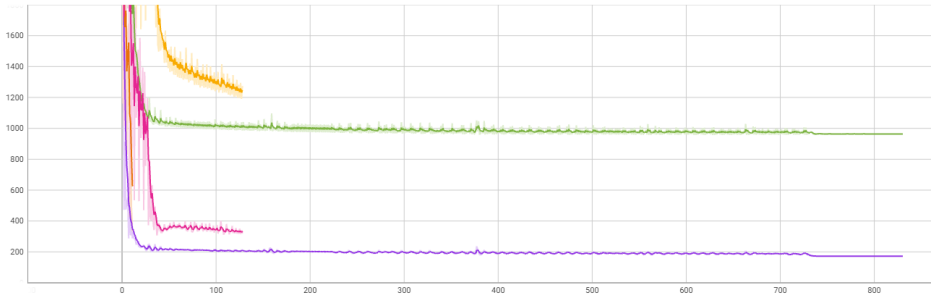


Figure 22: TensorBoard loss curve (*Self-Composed*)

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**.

The standard prediction and test set evaluation technique was applied to both models comparing the ensemble with the naïve forecast.

Table 23: Univariate model evaluation

	MAE	MSE	RMSE	MAPE	MASE
Naïve forecast	951	2021966	1421	2.56%	1.00
Ensemble	950	2013928	1419	2.56%	0.99

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

Based on the above evaluation, the multivariate model performance is better than univariate. This is likely due to it being more robust as the dataset utilized was enriched with multiple exogenous features.

It can also be observed that the Naïve forecast is better in performance in multivariate while the opposite is the case for univariate. This is likely due to the increased complexity of the ensemble architecture compared to the Naïve model, which can add more noise especially since a more diverse dataset is used.

8.5 Benchmarking

Extensive benchmarking was not possible as the algorithm is novel, and no system exists that utilizes all the features as in the implemented application. However, as demonstrated above, the Naïve model – that is notoriously difficult to beat in open systems - was benchmarked against to establish a baseline that could be used as a benchmark in future.

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 that was carried out.

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 they perform as expected.

Table 25: Module & integration testing

Module	**Input**	**Expected result**	**Actual result**	**Status**
Data fetcher	Triggered periodically	Fetch & update datasets.	Datasets scraped and stored into database.	Passed
NLP data parser	Latest scraped data.	Perform sentiment analysis on tweets.	Sentiment analysis is performed and the scores are weighed and saved in the database.	Passed
Data preprocessor	Latest obtained & parsed data.	Combine all datasets, clean, process and remove unneeded columns.	Datasets are cleaned and combined into a single dataset which is 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 requirements were tested by determining how well does the system align with the non-functional requirements mentioned in **Chapter 4** and the design goals stated in **Chapter 5**. **APPENDIX F.2** shows the breakdown of the non-functional testing that was carried out.

8.9 Limitations of the testing process

The system was tested completely, however, the algorithm itself could not be tested. Analyzing the LTSs complexities was possible and presented in **Chapter 6**; however, the mathematical expressions that are calculated behind the scenes could not be tested effectively. Nevertheless, simple debugging such as printing the calculated values on the terminal and comparing it against a calculator was performed; however, this does not qualify as a complete testing of the algorithm. It is worth noting however that the formula itself was accepted by the research community *(the formula was proposed in the author's extended review paper as a rectification to the identified limitations)*.

8.10 Chapter summary

This chapter initially presented the objectives of conducing testing and the criterion on which it was performed. Testing was carried out on the core research component by evaluating the models in use, and the system was evaluated by functional, non-functional and integration testing. Finally, any limitations of this procedure were specified.

CHAPTER 09. EVALUATION

9.1 Chapter overview

Upon implementation, evaluation is required to ensure that the system has met the requirements gathered in **Chapter 4**. This chapter discusses the project's evaluation - in detail, it presents 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 to design and implement the LTS algorithm, rigorous qualitative analysis from niche experts is required to ensure its robustness. Additionally, as the implemented system uses a model that presents a numerical output, quantitative analysis is required to determine its performance – which was presented in **Chapter 8**. In this chapter, thematic analysis conducted on the feedback obtained by the above-mentioned qualitative analysis will be presented.

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 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 sufficient 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 self-evaluation that was performed by the author according to the above-mentioned criteria.

Table 27: Self-evaluation of the author

CR ID	Author's self-evaluation
CR1	The chosen domain is novel and has hardly any experts with the technical knowledge, hence, it has limited research available. However, the domain has extremely high potential on being a stepping stone for more capable DL algorithms due to its ability to adapt its evaluation strategies and not being static.
CR2	The author considers the contributions made as highly impactful given the short time duration. 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 (LSTMs, RNNs, GRUs etc.), 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 and set higher milestones.
CR4	Compared to other DL domains where ample research and literature is available to aid the researcher with their project, the chosen domain has possibly not even ten complete papers to read and understand the domain in detail. A lack of research lead to difficulty as there was no clear path to follow.
CR5	Almost all the information available – ranging from research papers to recorded video conferences - on neural ODEs/SDEs & LTCs was gathered by the author and presented in this document to ensure that the quality is of the highest possible standard.
CR6	Development was performed in stages: the algorithm implementation followed by the system implementation. • To ensure that the LTS works and acts the same as other existing DL algorithms, it was built following recommended practices by TensorFlow, allowing it to be used the same way as other Keras layers. • The system was built following industry standards and techniques to ensure that it is as robust as possible and to align with recommended best practices. Cutting-edge tools and technologies sought after by companies and employers were utilized in implementation. Even though the system is the work of a single person, it was developed with the mindset of there being multiple contributors. Hence, it exhibits clear documentation, commit messages, workflows etc. within the GitHub repositories.
CR7	The system was developed considering that the end-users have no technical knowledge by making it as user-friendly as possible.
CR8	Upon completion of the system, the author identified a couple of enhancements and improvements that could be the subject of future work.

9.5 Selection of evaluators

The selection of the evaluators was done based on grouping required to obtain feedback for all aspects of the project. The table below shows the grouping breakdown.

Table 28: Categorization of selected evaluators

CAT ID	Category
CAT1	Experts with knowledge in neural ODEs, SDEs and LTCs or traditional DL/ML.
CAT2	Experts with knowledge on blockchain and cryptocurrencies.
CAT3	End users of the application, including crypto buyers and traders.

9.6 Evaluation results & expert opinions

Thematic analysis was conducted to analyze the opinions and feedback received by the above-mentioned experts. The table below discusses the findings that emerged.

Table 29: Thematic analysis of expert feedback

CR ID	CAT ID	Theme	Summary of opinions

9.7 Limitations of evaluation

Due to there being very limited experts in the research domain, only a few experts' opinions were obtained. During the requirement gathering phase, this was made evident as the author had reached out to many ML/DL domain experts to gather more insights on the research concept, however only a few were able to provide such information. Hence, only the experts specified in **APPENDIX**

G.1 had the necessary knowledge and expertise to provide constructive feedback. Nevertheless, traditional ML/DL experts were also selected to obtain feedback as it still would be beneficial.

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 and achievement of the design goals are presented in **APPENDIX G.3**.

9.10 Chapter summary

This chapter focused on the evaluation of the implemented system. Prior to conducting evaluations, criteria were set, to ensure that all aspects of the system are looked into. The author performed self-evaluation, and obtained necessary feedback from evaluators which were then analyzed by thematic analysis. Finally, the evaluation of the functional and non-functional requirements was presented along with the achievement of the proposed design goals.

CHAPTER 10. CONCLUSION

10.1 Chapter overview

This chapter concludes the author's research journey by marking its final remarks. In detail, it defines the author's contributions, achievement of the stated objectives, any deviations taken from the proposed scope, challenges encountered, and future research possibilities and limitations. Furthermore, new skills the author obtained during the progression of the research as well as how their existing skills were utilized, are documented.

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 a novel algorithm (LTS) inspired by the LTC so that it can build intelligent systems by developing a novel architecture for TS forecasting, which could be the stepping stone to be further expanded to other domains as well.

The aim was successfully attained by designing and developing the proposed LTS architecture in a way such that it was capable to be used as other existing layers. This algorithm was then utilized in a BTC forecasting application to evaluate its performance and to determine whether it broke limitations in TS forecasting algorithms.

10.2.2 Achievement of objectives

The final status of each objective that was stated in chapter one has been marked alongside the objective itself in **APPENDIX H.1**.

10.3 Utilization of knowledge from the degree

A tremendous amount of knowledge was required to bring the proposed research to fruition. Fortunately, a solid foundation had been laid by modules the author had completed during the course of the degree – of them, the most beneficial ones are highlighted in the table below.

Table 30: Knowledge utilized from the degree

Modules	**Knowledge utilized**

Mathematics for Computing	In-depth mathematical knowledge was required to build the algorithmic formula. This module laid the foundation to get started with learning more advanced mathematical concepts.
Object Oriented Programming	This module provided experience in creating full-stack applications that can be considered as useful in the real world. Building a backend and integrating an API with a client was an important skill that was required in the creation of the MVP.
Software Development Group Project	The motivation to undergo a challenging project was obtained through this module. From pitching an idea to presenting the final product in competitions, this module laid the groundwork for future research, design, development and evaluation.
Algorithms: Theory Design and Implementation	The knowledge obtained from this module was paramount when analyzing the performance and complexities of the developed algorithm.

10.4 Use of existing skills

- **Full-stack development** – the author has a few years of experience in full-stack development having done his internship at 99x working on web and mobile development. 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 on ML & DL applications. 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 as only 50% of the work in the deployment pipeline. The author had to learn a few techniques to create APIs that would serve the created models.
- **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. MIT OpenCourseWare was used heavily to understand and learn these techniques.

10.6 Achievement of learning outcomes

The achievement of the learning outcomes is presented in **APPENDIX H.2**.

10.7 Problems and challenges faced

Challenges are inevitable in implementation; the below table describes them and their mitigation.

Table 31: 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 in turn, only a few evaluators, the ones interviewed were pretty much best in class and gave valuable insights.
Steep mathematics learning curve.	Designing the final formula was done almost blindly by reading through centuries-old documentations where the initial mathematical and scientific concepts were proposed.
Unclear direction to follow due to the lack of documentation of SDEs.	The author read through the few available literature and had to understand it as clearly as possible. The author published a paper with the proposed formula as soon as possible to determine whether it was correct and to go forward with implementation.
Problem domain	
Twitter API rate-limitations provided access to tweets for only the past 7 days.	The author developed a dedicated scraper to fetch the data. Only 500 tweets were collected for each day to avoid rate limitations and as this was not the primary focus of the research.
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 aimed to fully automate the ML pipeline of updating the model with the latest available data; however, was only capable of partially automating it. Scraping the datasets regularly was automated, however, triggering a model deployment can only be performed locally as automating this procedure requires dedicated cloud servers which are expensive.

The features in scope proposed in the project proposal are stated in **APPENDIX A.2**. Based on the proposed scope, no deviations have been taken in the category of "in-scope". Unfortunately, due to time constraints, none of the "desirables" category had been implemented.

10.9 Limitations of the research

- The application does not forecast in real time, rather gives a forecast only for specific days.
- Only 500 tweets were scraped for each day and considered to obtain the sentiment values.
- Tweets with only a tag of "bitcoin" were scraped (other tags such as "btc" could be used).
- The model must be manually trained and deployed as only the updating of data has been automated.

10.10 Future enhancements

- Identify and determine how other features impact the price (Reddit, Facebook, Fiat coins).
- The LTS uses the Euler-Maruyama SDE solver which does handle all forms of noise. Nevertheless, other SDE solvers must be evaluated to determine if there would be a better performing option.
- The LTS can make use of a hybrid SDE solver that combines the implicit and explicit Euler-Maruyama solvers than the utilized implicit Euler-Maruyama.
- LTS with reverse mode AD must be evaluated instead of the utilized BPTT, to determine memory and time efficiency.
- Benchmark the LTS in the M4 competition to determine whether it is a SOTA TS forecasting algorithm.
- Enhance the Twitter sentiment weighing formula to consider more factors.
- Test the weighing formula to determine whether the score calculated aligns with sentiment analysis guidelines.

10.11 Achievement of the contribution to the body of knowledge

Upon completion of this project, the author managed to achieve contributions to the research domain of DL and TS forecasting and the problem domain of BTC.

10.11.1 Research domain contributions

- SDE-based liquid neural network – a novel implementation of the LTC with adaptability to instantaneous changes; TensorFlow V2 implementation of the LTC & LTS algorithms.
- Surpassed limitations of TS forecasting algorithms by implementing a flexible algorithm that adapts to changing characteristics.

10.11.2 Problem domain contributions

- A novel algorithm to weigh Twitter sentiments based on influencer metrics.
- Created a robust implementation of a BTC forecasting model that considers multiple exogenous features that affects the historical price.
- Surpassed the naïve forecast performance in open systems.

10.11.3 Additional contribution

- Data extraction and preprocessing scripts for Twitter tweets, Twitter volume, block reward size, Google Trends and BTC historical prices.

10.12 Implementation code

The code written and associated research work conducted is in GitHub for convenience, in the below links:

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

10.13 Concluding remarks

This marks the end of researching, designing and developing a novel algorithmic solution to solve a notorious problem in time series forecasting that hindered its performance compared to other DL domains for quite some time. Its application was demonstrated in a BTC forecasting application exhibiting promising performance that can be improved upon to apply it 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].

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].

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].

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].

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].

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].

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].

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 diagram below depicts the prototype feature diagram proposed in the proposal document

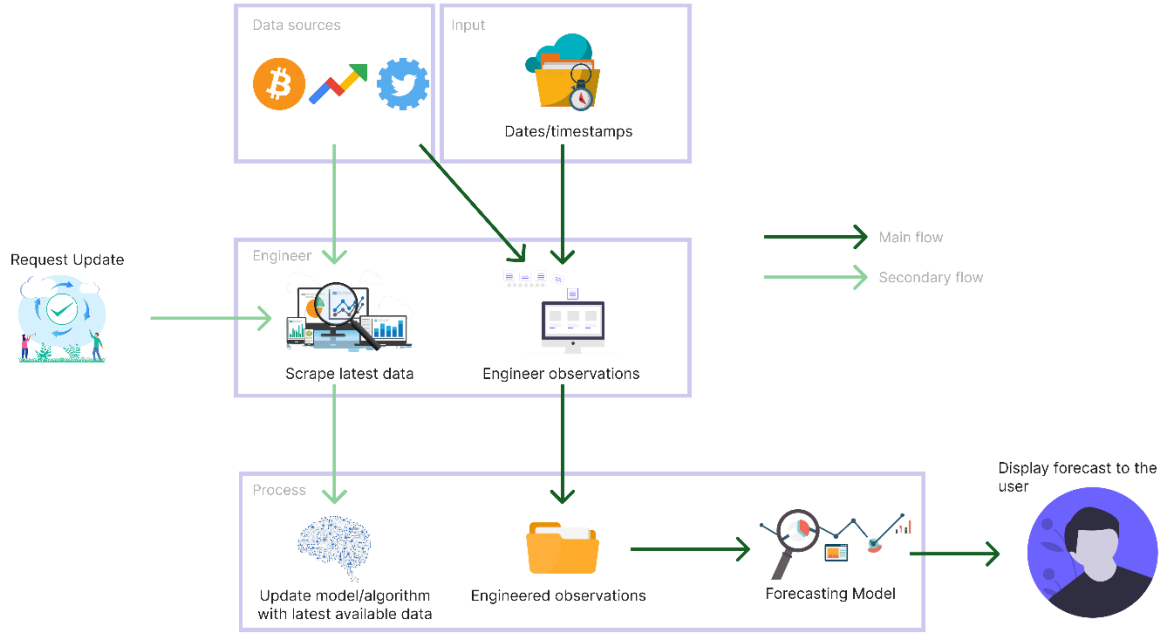


Figure 23: Prototype feature diagram (*Self-Composed*)

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).
- Explainable AI for neural SDEs and neural ODEs.

APPENDIX B – LITERATURE REVIEW

B.1. Analysis of forecasting algorithms

Table 32: Analysis of forecasting algorithms

Ref.	Brief	Improvements/Contribution	Limitations/Future work
Statistical-based forecasting algorithms			
Box et al., 2015	**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.
Engle, 1982	**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.
Taylor and Letham, 2017	**Prophet**. A modular regression model with interpretable parameters. These	Solves forecasting at scale, where scale refers to three types.	It uses simple and weak assumptions and produces much poorer

	parameters can be adjusted according to the problem by domain experts, similar to ARIMA.	1) A large number of people forecasting. 2) A large variety of problems. 3) A large number of forecasts being created.	performance than ARIMA. And it does not model relationships between the past and future.
DL-based forecasting algorithms			
Hochreiter and Schmidhuber, 1997	**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.
Cho et al., 2014	**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.
Oreshkin et al., 2020	**N-BEATS**. An architecture that solves the univariate time series point forecasting problem. It carries some benefits, some of which are being	Outperformed the M4 competition winner of the previous year and improved the statistical benchmark forecast.	Tailored specifically for univariate TS analysis, therefore, would perform poorly on multivariate analysis. Additionally, Meta-learning is

	understandable, easily applicable to multiple other fields, and being fast to train.		speculated to be a reason for the performance and must be investigated.
Lim et al., 2019	**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.
Hasani et al., 2020	**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.

B.2. Studies associated with these algorithms

Table 33: Few studies associated with these algorithms

Ref.	**Technology**	**Outperforms**	**Findings**
Statistical-based forecasting algorithms			
Zhang et al., 2022	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.

Yenidogan et al., 2018	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			
Kuan et al., 2017	GRU	LSTM, traditional GRU	Scaled exponential linear units were proposed to deal with vanishing gradients. These architectures significantly outperformed LSTM and traditional GRU models.
Ugurlu, Oksuz and Tas, 2018	GRU	MLP, LSTM	Performed much better than LSTM and trained faster as it is simply a simpler form of LSTM.
Wang, Liao and Chang, 2018	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.
Sagheer and Kotb, 2019	LSTM	ARIMA, GRU	Genetic algorithms were used to create an optimized LSTM architecture.
Bouktif et al., 2018	LSTM	MLP, Linear regression	The most optimal time lags and number of layers for an LSTM was found using genetic algorithms.
Fischer and Krauss, 2018	LSTM	MLP, Logistic regression	Peeked into the internals of the LSTM to find common stock patterns in noisy data.

Pan et al., 2019	LSTM	MLP, traditional LSTM	Utilized a novel attention-based LSTM architecture to improve the performance of traditional LSTMs.
Oreshkin et al., 2020	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.
Hasani et al., 2020	LTC	LSTM, GRU, 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.

APPENDIX C – SRS

C.1. Requirement elicitation methodologies

Table 34: Stakeholder groups

Group	Stakeholders	Reason	Instrument
G1	Domain experts (neural ODE/SDE and blockchain/crypto)	Gather any insights and knowledge specifically in the research domain to answer research questions and anything the author may have missed.	Interview
G2	End users (trader & buyer)	Gather requirements for supplementary application implementation.	Survey
G3	Competitors	Analyze any existing systems and literature in the research and problem domain.	LR/Observations
G4	Developers	Ensure completion and feasibility of the project.	Prototyping

C.2. Interview analysis

Table 35: 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 36: Interview thematic analysis themes, conclusions & evidence

Theme	Conclusion	Evidence
Research component		
Research Problem & Gap	The interviewees validated the research gap and the defined problem. They were also happy that the author had been conducting this research, as few papers were 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 were concerned that ODEs and SDEs could be expensive to compute and hence could take some time, which can be an issue given that the forecasts must be produced quickly. Therefore, the author must optimize the model as much as possible to avoid user-unfriendliness.	"They are expensive to compute." "It can be resource-intensive."
Advice	The author had initially planned on only creating an implementation of the LTC architecture proposed by Hasani et al. (2020). However, the author could further improve the architecture by using SDEs instead (the base LTC uses ODEs), which could manifest into a novel algorithm, which is the author's current aim as it carries more	"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."

	significance and a potentially more outstanding contribution.	
Other suggestions	What was concluded here was that XAI is primarily present for image classification, and there needs to be more literature on the TS domain. However, XAI integration into TS modelling could be confusing and complicated due to the temporal component. Additionally, XAI for SDEs needs to be researched, which the author could look into if time permits.	"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."
Problem domain		
Robustness	The interview was an additional validation for the data collected in the survey. Most suggestions were to use as many extra features as possible to make the model robust. Therefore, the author will ensure that they utilize the mentioned exogenous 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 37: Survey thematic analysis codes, themes & conclusions

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

Theme	Conclusion	Evidence
Robustness	Participants believed that prediction needed more than just including historical prices and that social media Trends and other factors (ex: sentiment) are required to make the system as robust and performant as possible.	"Use previous trends in the past." "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 couple of participants requested that the system provide some cryptocurrency news to make it convenient and make the inference procedure as straightforward as possible, so there is no hindrance.	"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 an ideal scenario if the system could tune the hyperparameters of the model in use, which could be an excellent enhancement to the system as the model anyways retrains periodically.	"Coming from 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	"Predict the market for any given time duration."

	believes they will not be able to cover, given the time allotted.	"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 38: Use case description UC:03; UC:04; UC:05

Use case	Manage exogenous features
Id	UC:03; UC:04, UC:05
Description	Manage and process new data without the need for manual interaction.
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. A Cron job triggered fetches the latest historical prices, tweets, Twitter volume, trends, and block reward size data. MF2. Twitter volume, Google trends, and block reward size are scaled and cleaned. MF3. Tweets undergo sentiment analysis to determine current speculation. MF4. The sentiment is further weighted based on the Tweeter's importance (ex: Elon Musk) MF5. Features are combined with historical closing prices to create an enriched dataset and retrain the model.
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	A new enriched dataset with the features is generated.

Table 39: Use case description UC:07

Use case	Update model hyperparameters
Id	UC:07
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 (as the model would ideally need to be retrained upon hyperparameter tuning).
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. Obtains the latest available data. 2. Performs sentiment analysis and self-retrains. MF4. The system retrains itself with the data and new hyperparameters.
Alternative flows	None
Exceptional flows	None
Post-conditions	The model is updated with the chosen hyperparameters.

C.5. Functional requirements

Table 40: '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

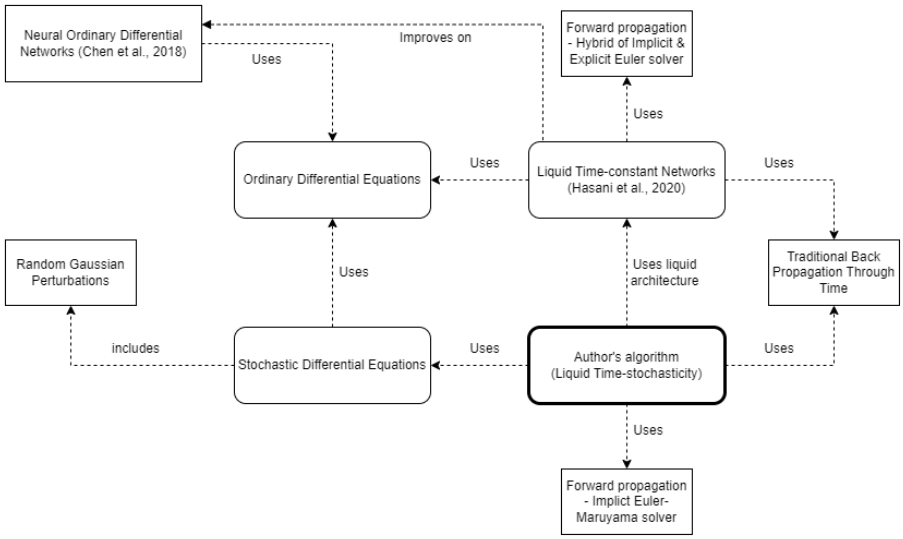


Figure 24: Algorithm intuition (*Self-Composed*)

What exactly an SDE solves compared to an ODE?

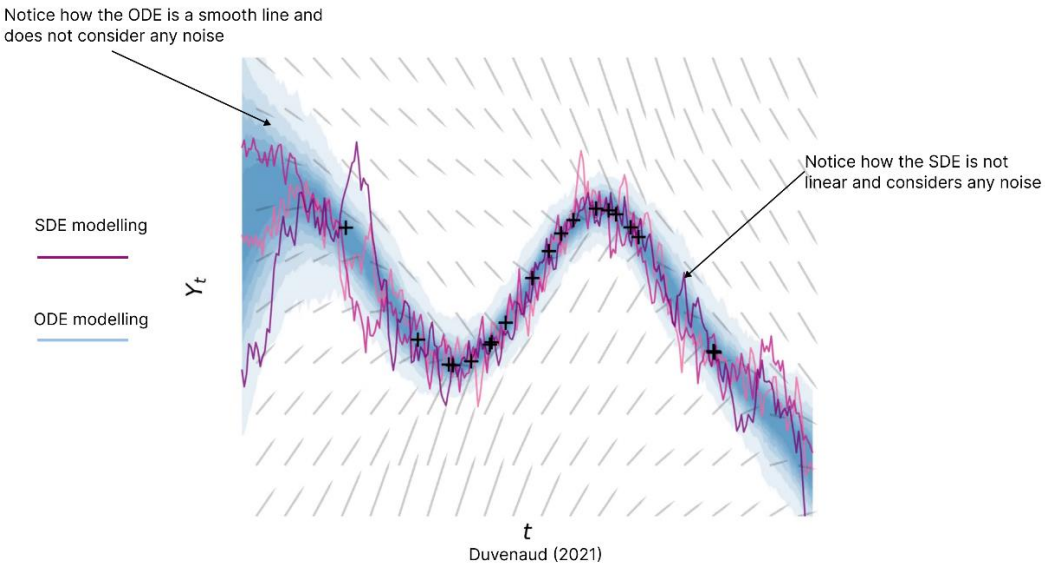


Figure 25: Understanding what an SDE solves

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

Existing LTC architecture

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

τ	*Time-constant*
$x(t)$	*Hidden state*
$I(t)$	*Input*
t	*Time*
f	*Neural network*
θ, A	*Parameters*

The above formulation was proposed by Hasani et al. (2020), where a system of linear ODEs is used to declare the flow of the hidden state; the ODEs are of the following form.

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

Where S(t) represents the following nonlinearity

$$S(t) = f(x(t), I(t), t, \theta)(A - x(t))$$

The equation manifests by plugging the above equation into the system of linear ODEs.

Formulation

Step 01 – 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.

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

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:

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

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

However, noise can be of varying intensities (some could be high, some could be low). Considering this varying intensity, the SDE can be further expressed as follows:

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

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.

Step 02 – adding neural networks into SDE dynamics

Based on the findings of Duvenaud (2021), 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.

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

A neural network can be integrated into the above equation to solve the system, resulting in the following equation:

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

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

Step 03 – 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.

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

As the above equation is a linear system of ODEs initially proposed by Lapicque (1907), the author could add the uncertainty noise to the equation to produce the following:

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

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:

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

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

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

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

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

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

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

D.2. Tweet sentiment weighing 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 different weighing factors applied to each metric. This is determined by observing how much of an impact each metric makes. Applying these weighing factors will produce the following:

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

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

To avoid a specific metric from dominating the score (if the number of followers is in the millions and number of retweets is in the thousands, the number of followers would dominate the score), a logarithm can be applied. Applying a logarithm can normalize and balance all the metrics into a similar scale to avoid any bias, as such accurately reflect each metric's true impact on the score. Furthermore, the author will add the value "1" to each metric prior to obtaining the logarithmic value. Performing this will prevent mathematical errors that would arise for the case of a metric having the value of 0. The terms can now 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 fall within any range and must be normalized to ensure that the final score will not exceed 1. Normalization can be applied as follows:

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

Finally, this influencer score can be applied to the vanilla compound score to obtain a weighted compound score.

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

D.3. LTS algorithm complexity analysis

Table 41: Complexities of BPTT and adjoint sensitivity

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. UI wireframes

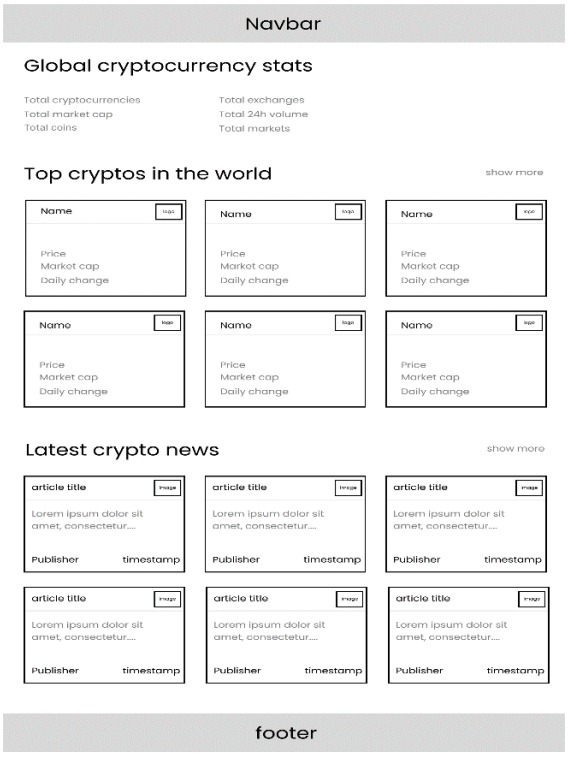


Figure 26: UI wireframes – Home (*Self-Composed*)

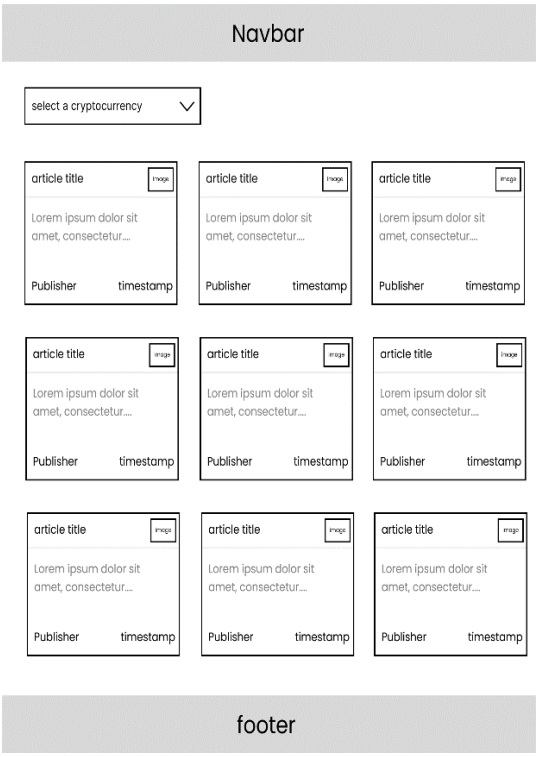


Figure 27: UI wireframes – News (*Self-Composed*)

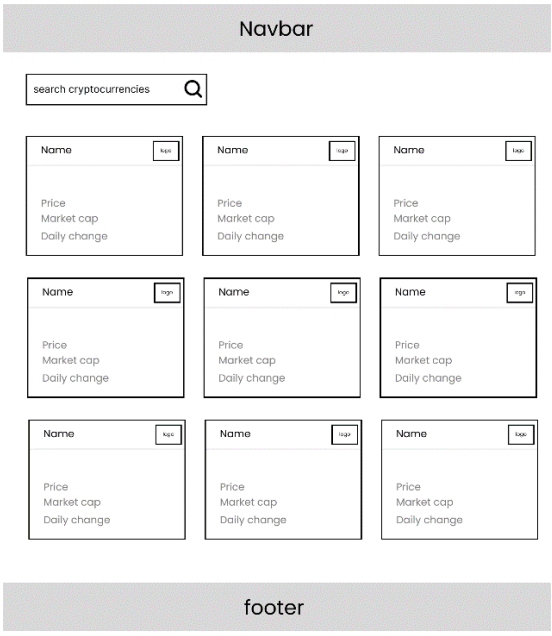


Figure 28: UI wireframes – Cryptocurrencies (*Self-Composed*)

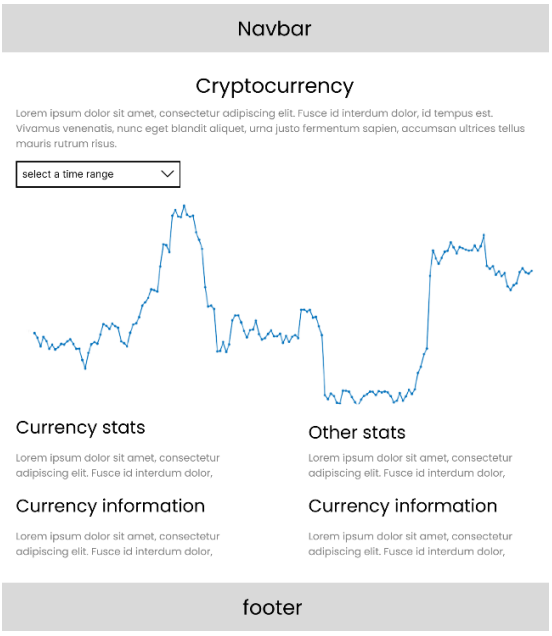


Figure 29: UI wireframes – Cryptocurrency (*Self-Composed*)

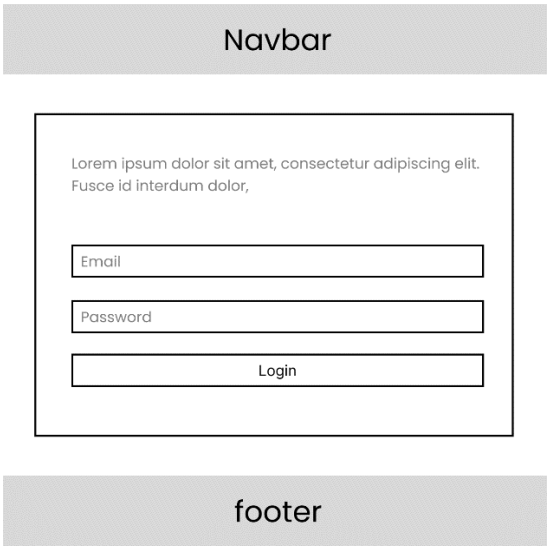


Figure 30: UI wireframes – Admin login (*Self-Composed*)

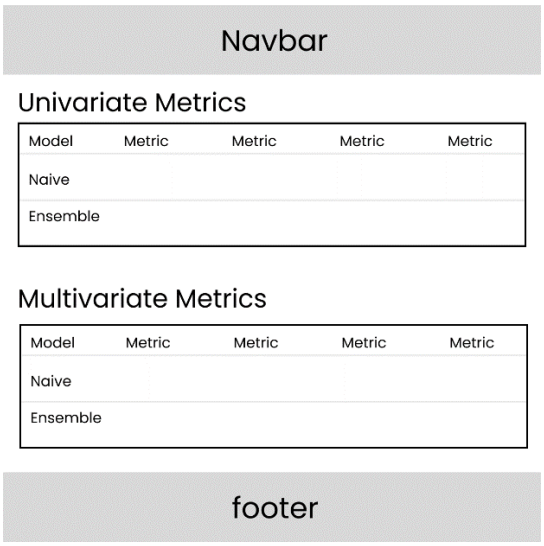


Figure 31: UI wireframes – Admin metrics (*Self-Composed*)

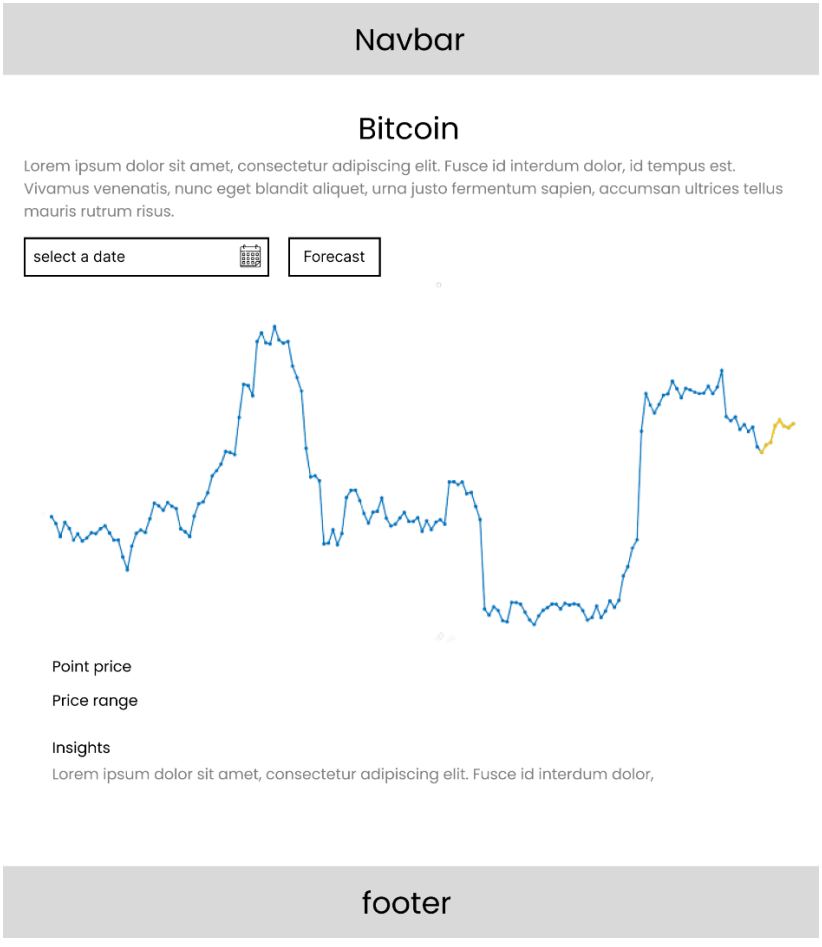


Figure 32: UI wireframes – Forecast (*Self-Composed*)

APPENDIX E – IMPLEMENTATION

E.1. Selection of programming language

The below table summarizes the analysis of the language chosen for the data science component, where each option was given a score within H – High, M – Medium, and L – Low.

Table 42: 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 an additional benefit 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 spending 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 43: Selection of DL framework

Framework	Description
TensorFlow	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**. Although it is more complicated, the higher-level API: Keras, is now officially a part of TensorFlow. Therefore, model development has become much more straightforward. Additionally, building the algorithm requires more low-level details. (Kurama, 2022)	

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

Table 44: Selection of UI framework

Framework	Description
Angular	Suitable for large-scale applications with dedicated submodules for particular functionalities. However, it can be less performant in comparison and unnecessarily heavy.
Vue	A tiny framework that takes little to no time to startup and is much more intuitive as the code is simple. Additionally, based on simulations, it has been identified to perform better than Angular and React. However, it has much fewer resources.
Svelte	The most lightweight and genuinely reactive. Much more performant than the rest; however, it has a small community of developers and is relatively new.

React	Customizable and promotes code reusability via functions as components. It carries a large community and is open-source while being SEO-friendly. Additionally, the React developer tools is very handy.
Conclusion Based on the analysis, the author chose **React** as the GUI built will be simple, and there is no requirement for large-scale applications, as it is not the primary focus. (Patadiya, 2021)	

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

Table 45: Selection of web framework

Framework	**Description**
Flask	A very lightweight framework that provides only the simplest of functionalities. However, it is the preferred choice for ML API development because it is light.
Django	Suitable for more larger scaled applications that provide a vast range of functionalities, it is stricter and less flexible. Therefore, is much more demanding and heavier.
Conclusion The author chose **Flask** as it provides only the necessities in exposing an ML model and since the luxury features provided by Django (ex: authentication) were not required. (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 above script describes a couple of functions that can be used to fetch the latest BTC historical prices data and create a new updated CSV file that can be later read from by the model. A third-party API was used to fetch the data, as existing APIs are all discontinued.

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 fetch the Twitter volume and block reward from a website that publicly exposes this data. Therefore, a simple website scraping tool can be used without authentication or authorization.

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*)

Obtaining the tweet data required a more tedious process as the Twitter API had been updated only to provide tweets for the past week. However, third-party libraries offer this functionality. Tweets fetched were limited to 500 for a single day due to time, performance, and storage constraints, and the application is not the core contribution. Initially, tweets were fetched up to a specific time point; in the future, the above script could be run to scrape tweets of particular dates that are described to be from the days currently existing in the data folder up to the day at which the script is run. There is a further limitation as only '#bitcoin' is searched.

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

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

    print('Cleaning tweets ... ')
    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)
        df_filtered.to_csv(f'{FOLDER_PATH}/{filename}.csv')

    print('Tweets cleaned')
    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.

E.6. Preprocessing

Tweet sentiment analysis

The main step of preprocessing is to perform sentiment analysis on the obtained tweet data. In this research, the VADER sentiment analyzer is used as determined in previous chapters.

```
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 is used to perform sentiment analysis on the tweets and concatenates the negative, positive, neutral, and compound scores into the existing tweet dataset. The compound score is weighed beforehand by utilizing the proposed sentiment weighing formula in **Chapter 6** (as only the compound score is used on forth, 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*)

As the other data being used directly creates a single CSV file with a row for each date, the condensation process is unnecessary. However, as the tweet data fetched consists of a separate CSV file for each date, this data must be compressed to the same format as other datasets.

The above script condenses the tweet dataset into a single CSV file by averaging the sentiment scores for each day.

Final dataset creation

```
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 above script is used to create the final dataset that the model uses. It fetches all the datasets and combines them into a single data frame. Initially, a helper function removes unneeded columns from the data files, which were decided upon conducting correlation tests. The mean of their respective columns imputes missing values of each feature of specific dates. This combined dataset can be saved so the model can finally utilize it.

E.7. User interface

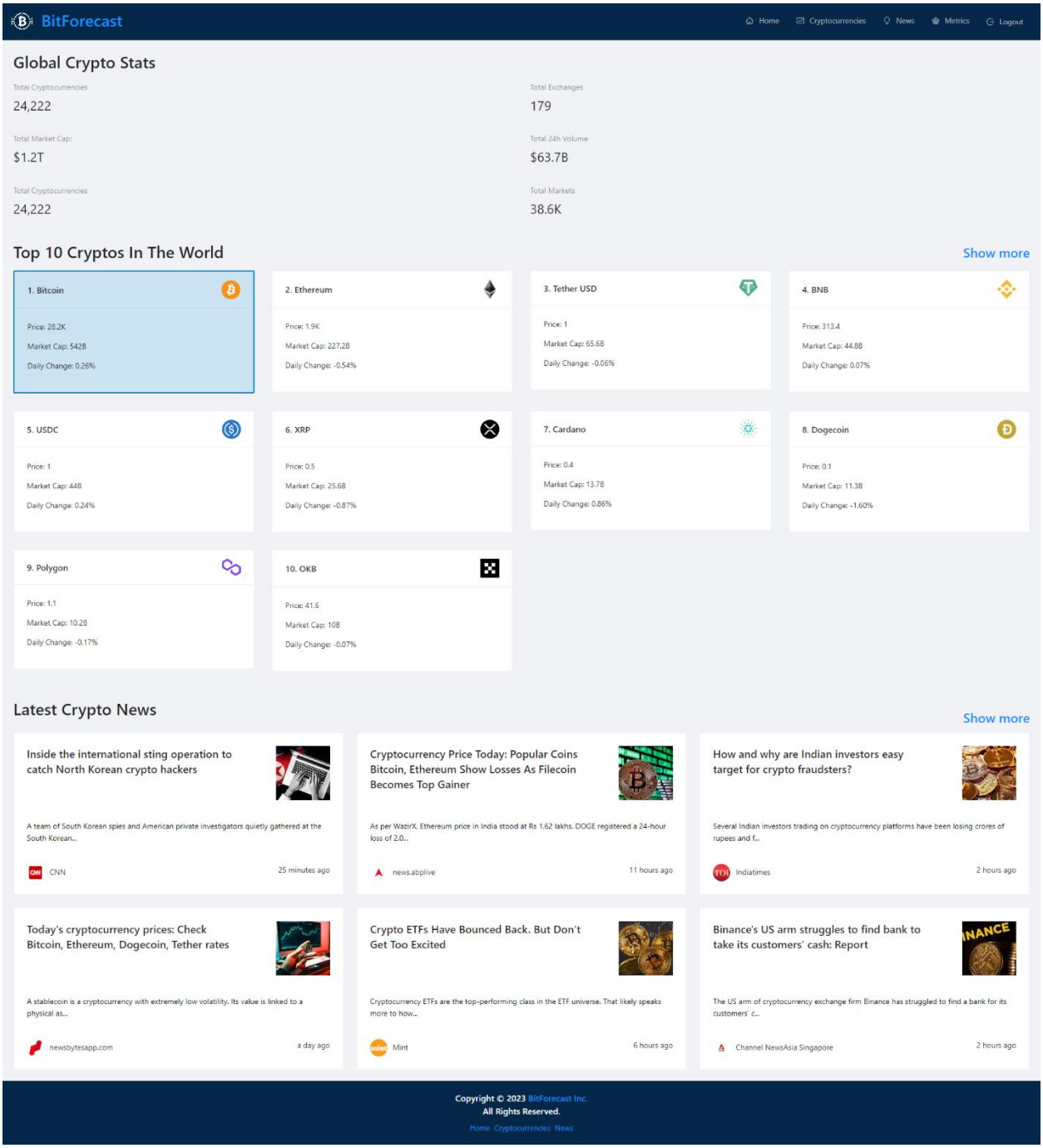


Figure 42: GUI - Home (*Self-Composed*)

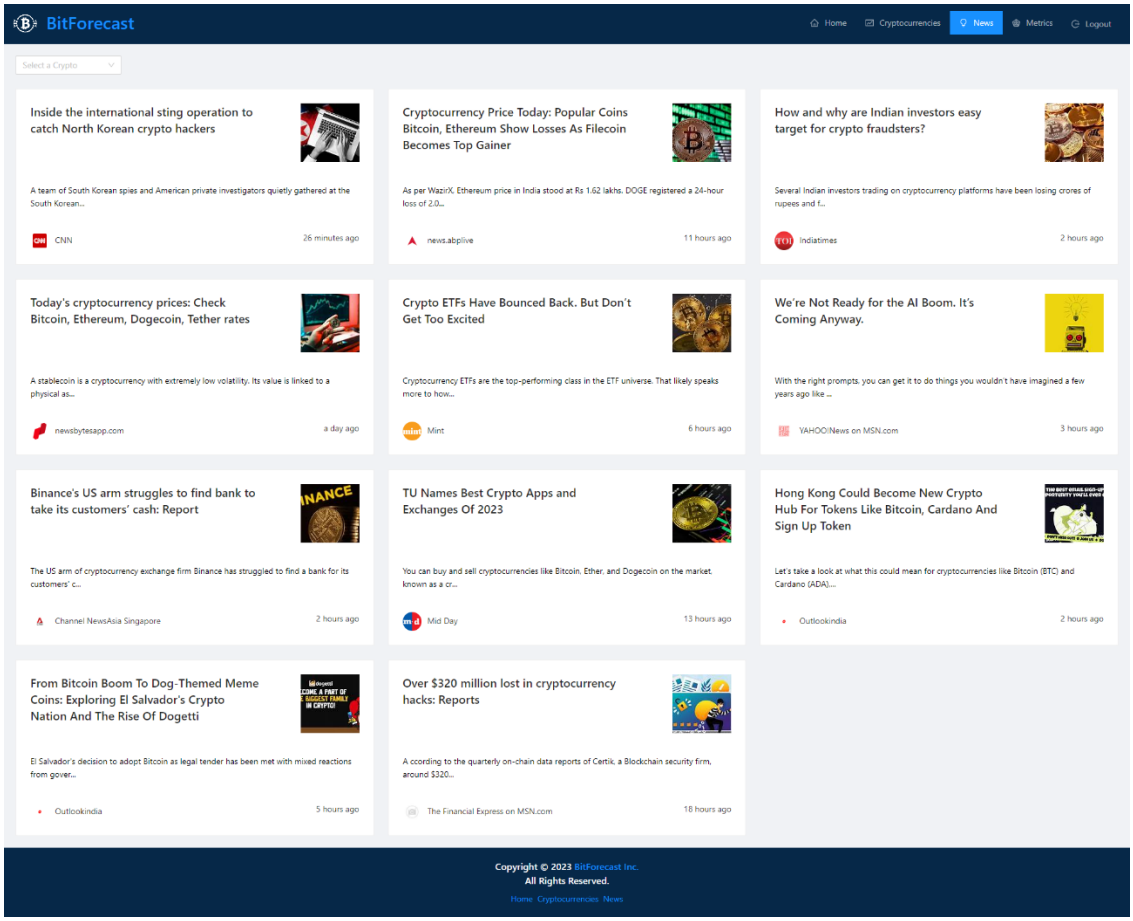


Figure 43: GUI - News (*Self-Composed*)

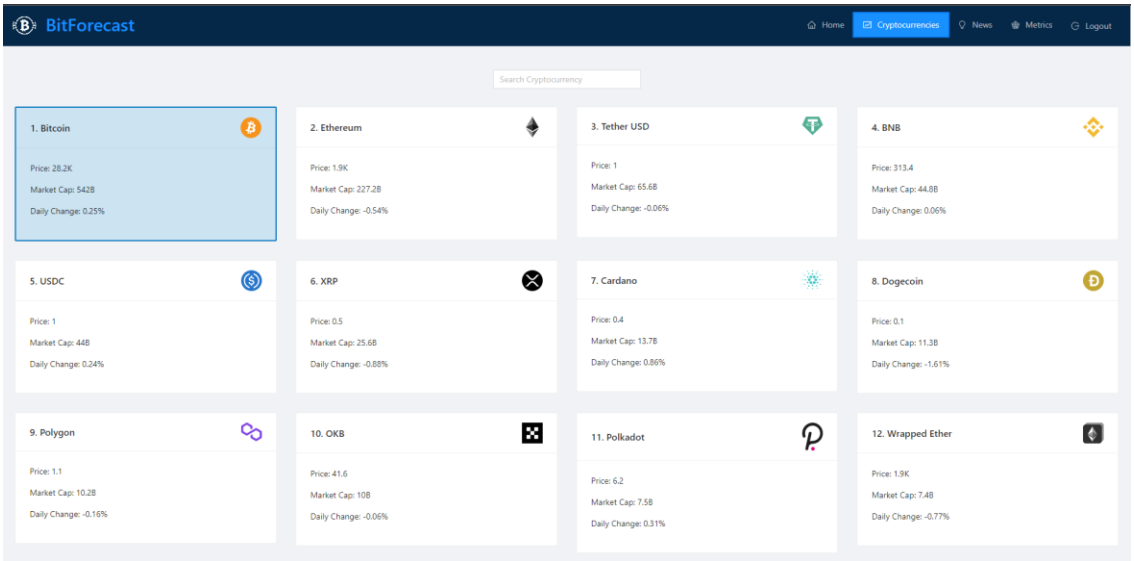


Figure 44: GUI - Cryptocurrencies (*Self-Composed*)

Figure 45: GUI - Cryptocurrency (*Self-Composed*)

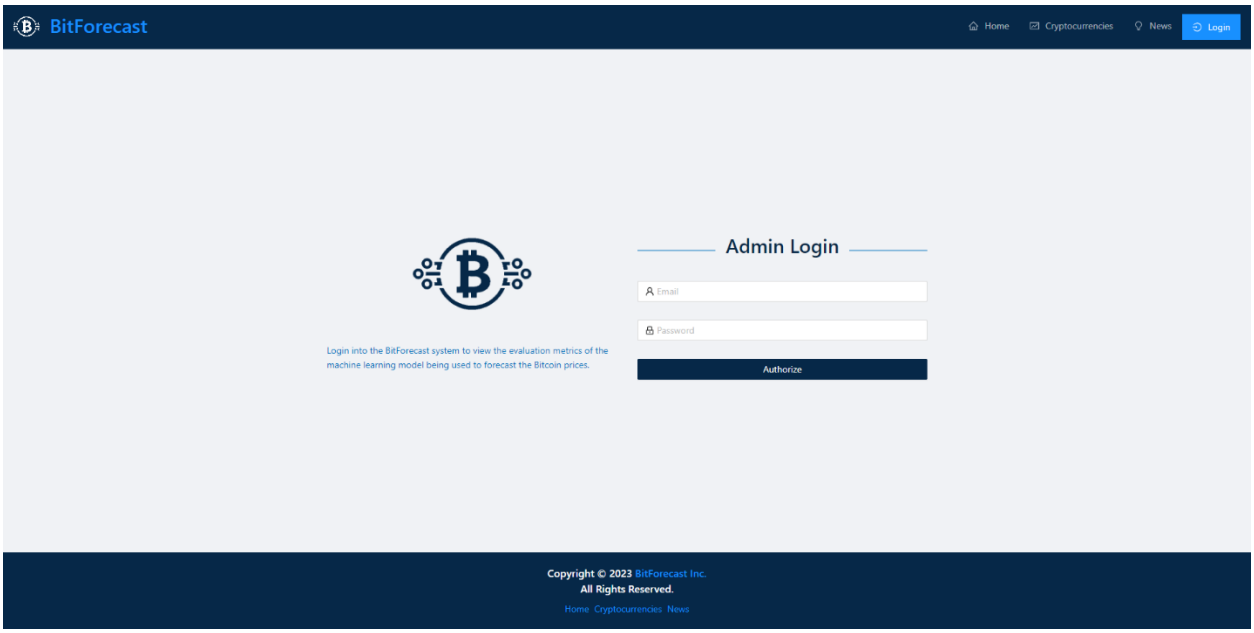


Figure 46: GUI - Admin login (*Self-Composed*)

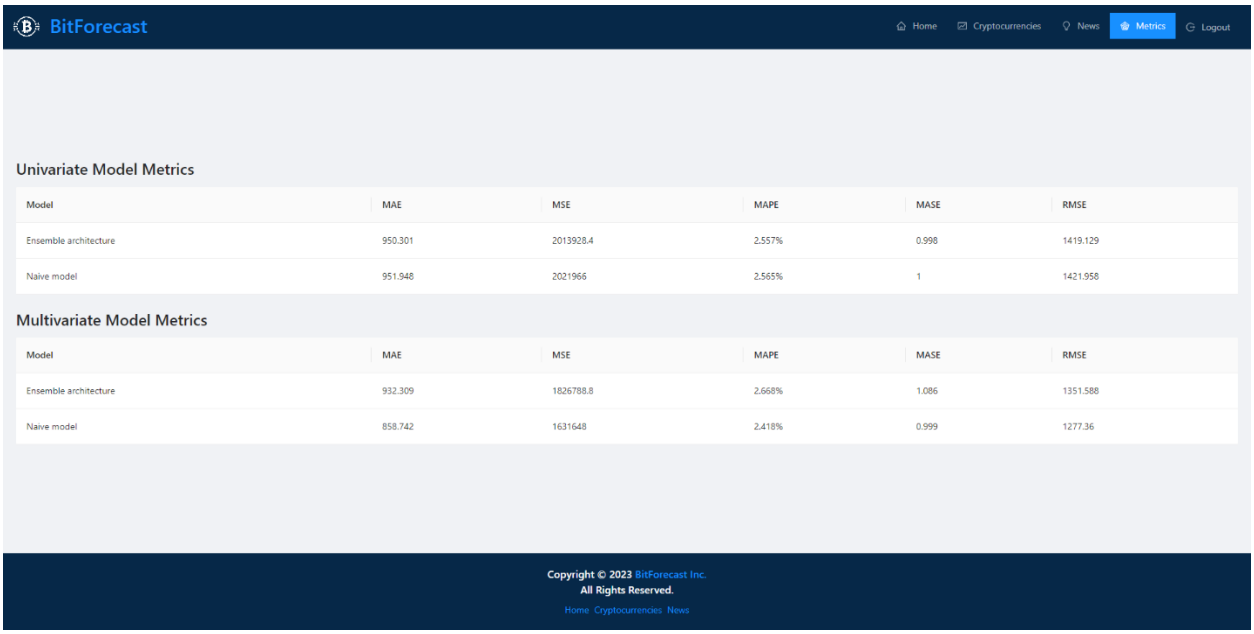


Figure 47: GUI - Admin metrics (*Self-Composed*)

Figure 48: GUI - Forecast (*Self-Composed*)

APPENDIX F – TESTING

F.1. Functional testing

Table 46: Functional testing

Test case	ID	Action	Expected result	Actual result	Status
Research level					
1	FR1	-	The LTS follows recommended standards so that it can be scalable and built upon.	The architecture was built as a Keras layer so behind-the-scenes techniques that need to happen is handled by Keras.	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 of the chosen dates are 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 model's responses of the chosen dates are returned to the user.	Passed

5	FR6, FR7	CRON script is triggered periodically.	The latest data 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 datapoints that are the future predictions.	Passed
7	FR11	Sentiments extracted from scraped data.	Sentiments are weighed based on influencer score by the proposed formula.	The sentiments undergo a weighing stage where, based on certain metrics, the score is changed.	Passed
8	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 applied performance, GUI and maintainability testing, and a few test-cases to determine if the system meets the non-functional requirements and the design goals.

Performance testing

The author had deployed the API and model; therefore, there is no requirement of having a high GPU and CPU power. Docker, GitHub Actions and Heroku with basic Dynos were utilized for deployment purposes, which is capable of serving requests for small-scale applications. However, for large-scale purposes, it is recommended that the Dynos are scaled up, as the application would

not be able to handle multiple requests concurrently. 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 determined that the need for developing a simple and effective GUI was important. The GUI was tested by Google Lighthouse to determine its performance and accessibility, the diagram below illustrates the obtained results.

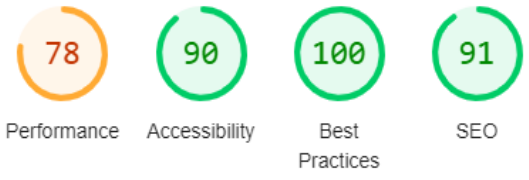


Figure 49: Lighthouse home page (*Self-Composed*)

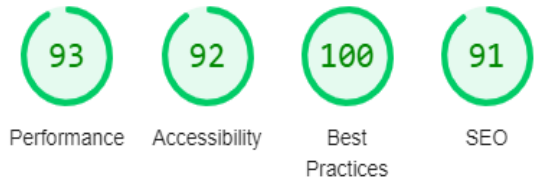


Figure 50: Lighthouse login page (*Self-Composed*)

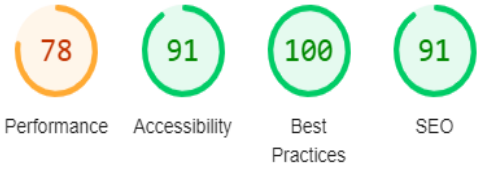


Figure 51: Lighthouse cryptocurrencies page (*Self-Composed*)

Figure 52: Lighthouse cryptocurrency page (*Self-Composed*)

Figure 53: Lighthouse news page (*Self-Composed*)

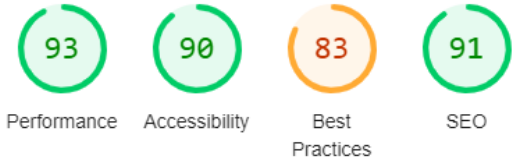


Figure 54: Lighthouse metrics page (*Self-Composed*)

The results vary from page-to-page. This is likely since most pages utilize third party APIs to render information, hence demonstrating a subpar performance value.

Maintainability testing

Maintainability is important so that future research on the system and especially the developed algorithm can be conducted seamlessly. CodeFactor and CodeQL were used to ensure that the repositories are maintained and documented well and that there are no vulnerabilities.

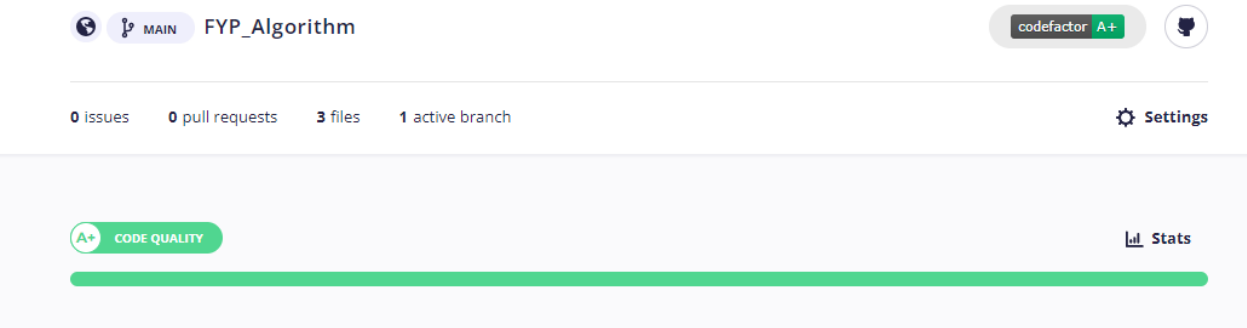


Figure 55: CodeFactor - Algorithm repository (*Self-Composed*)

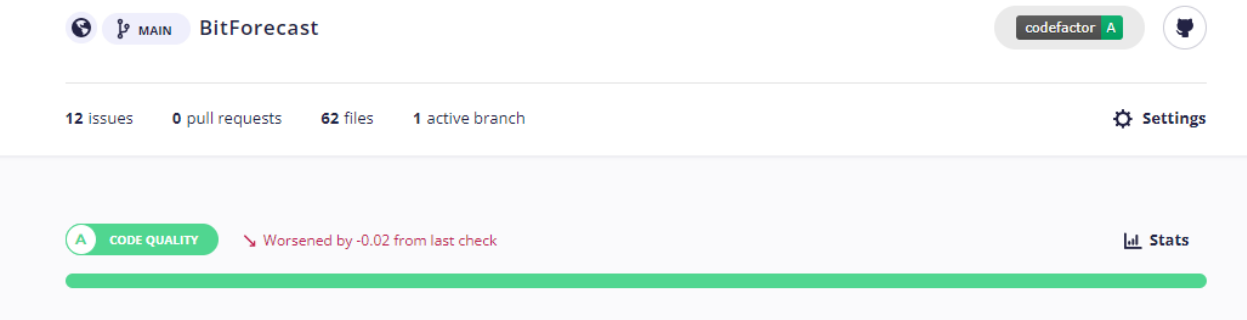


Figure 56: CodeFactor - Application repository (*Self-Composed*)

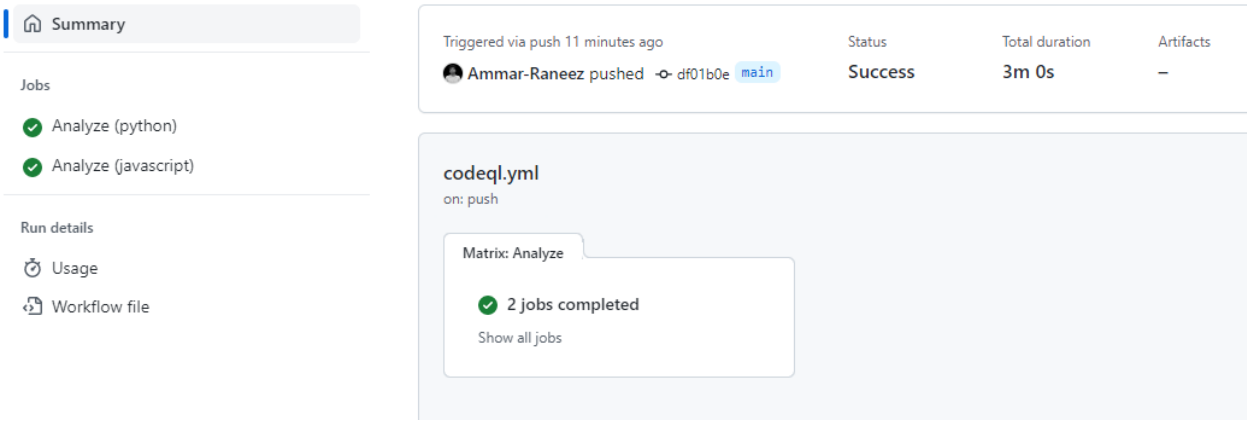


Figure 57: CodeQL - Algorithm repository (*Self-Composed*)

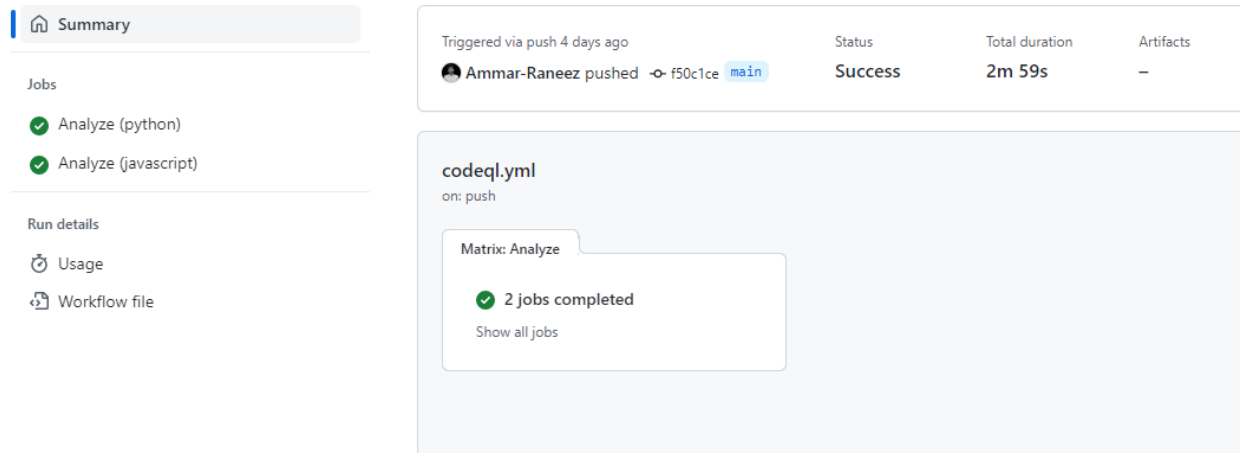


Figure 58: CodeQL - Application repository (*Self-Composed*)

Test cases

Table 47: Non-functional testing

Non-functional requirements		
Test case	**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, no technical information is displayed to the users, except for admins who log in.
4	NFR4	Docstrings were added for each method and comments wherever necessary.
5	NFR5	The responses are plotted within the graph itself 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 admin authentication is handled by Firebase. Therefore, security is integrated to some extent.

8	NFR8	The application is responsive across a wide range of screen sizes.
Design goals		
9	DG1	The data are stored in MongoDB and fetched whenever necessary. For the case where the available data is not up-to-date, the 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 shown to users on what happens behind the scenes.
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 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

Table 48: Selected expert evaluator details

ID	Affiliation	Expertise related to the research
Research domain		
EV1	Google Brain visiting researcher and Associate Professor at University of Toronto.	Neural ODEs and SDEs.
EV2	Research scientist at Deepmind.	Neural ODEs and SDEs.
EV3	Research scientist at Meta AI.	Probabilistic DL and differential equations.
EV4	PhD candidate at University of Nottingham.	ML & DL.
Problem domain		
EV5	Prefer not to say	Blockchain and cryptocurrencies.
EV6	Prefer not to say	Cryptocurrencies and crypto exchanges.

G.2. Evaluation of functional requirements

Table 49: Evaluation of the implementation of functional requirements

ID	Description	Priority	Use Case	Evaluation
Research level				
FR1	A robust and scalable implementation of the novel algorithm must follow recommended standards.	M	-	Implemented
FR2	The developed algorithm must be able to be 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:03	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:04	Implemented
FR7	The script must extract trends and sentiments from obtained data.	M	UC:05	Implemented
FR8	Users should be able to view a range of prices along with the single-point price.	S	UC:03	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:03	Implemented
FR11	The script could weigh sentiment based on any influential personnel's tweet.	C	UC:06	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:03	Not-considered
FR13	Admins could authenticate and update the model with different parameters.	C	N/A	Not-considered
FR14	Admins could get additional information about a prediction, such as the evaluation metric and accuracy.	C	N/A	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 50: Evaluation of the implementation of non-functional requirements

ID	Requirement	Description	Priority	Evaluation
NFR1	Performance	The system must take little time to generate a forecast, given that a couple of extra features are in use.	Important	Implemented
NFR2	Performance	The system must not unnecessarily keep updating its data.	Important	Implemented
NFR3	Usability	The user interface must be simple and effective and provide user-friendly errors if any occur.	Important	Implemented
NFR4	Maintainability	The author must document the codebase well in case of future reference, mainly the algorithm development repository.	Important	Implemented
NFR5	Quality	The output must be of good quality so that it provides vital insights.	Desirable	Implemented
NFR6	Scalability	The system must be deployed to a cloud with no scaling issues and good resources for efficient and optimal performance, especially as there could be multiple concurrent active user requests.	Desirable	Implemented (~50%)
NFR7	Security	The system must be resilient to attackers, specifically to prevent data manipulation.	Desirable	Implemented
NFR8	Compatibility	To ensure compatibility, the developer must test the system on most browsers and mobile phones.	Desirable	Implemented
NFR9	Availability	In critical failures, the primary operator must be available and solve issues as soon as possible.	Desirable	Not-implemented
Non-functional requirement completion percentage = $\frac{7.5}{9} * 100 = 83.3\%$				

Table 51: Evaluation of the achievement of design goals

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 52: Status of research objectives

Objective	Description	Status
Problem Identification	Understand and document the identified problem and provide reasoning on what makes it novel. **R**O1: Conduct research on a domain of interest and identify a comprehensive enough issue that requires solving. **R**O2: Delve deeper into the identified problem to obtain a general understanding on how to approach solving the problem. **R**O3: Split the problem down into manageable subsections so it is easier to digest and to solve one section at a time. **R**O4: Design a respective schedule, associated deliverables, and the Gantt chart.	Completed
Literature Review	Collate relevant information by reading, understanding, and evaluating previous work. **R**O5: Conduct preliminary studies and investigations on existing TS forecasting systems and algorithms. **R**O6: Analyze the requirement for specialized TS algorithms. **R**O7: Conduct research on neural ODEs, LTCs & SDEs. **R**O8: Obtain deep insights into the architecture behind the LTC. **R**O9: Research and obtain insights on factors affecting the price of BTC. **R**O10: Research on existing BTC forecasting and related open market systems. **R**O11: 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. **R**O13: Gather the requirements and architectures of LTCs and SDEs. **R**O14: Collate the most up-to-date details of BTC and obtain insights on the perspectives of the end users. **R**O15: Design necessary diagrams to justify the product's specification.	
Design	Design the architecture and a corresponding system capable of effectively solving the identified problems. **R**O16: Design necessary diagrams required to understand the algorithm RO17: Design diagrams required to understand the supplementary system being developed. **R**O18: Design the novel algorithm and analyze its complexities.	Completed
Implementation	Implement a system that is capable of addressing the research gaps. **R**O19: Design, evaluate and pick necessary technologies best-suited for the implementation. **R**O20: Develop an efficient LTC implementation. **R**O21: Build on the LTC to implement the LTS. **R**O22: Integrate the algorithm developed into a TS forecasting application. **R**O23: Integrate the intelligent system into a client application to display forecasts. **R**O24: Design and implement an automated flow to update the built network with the latest data. **R**O25: Design and implement a pipeline for easy deployments. **R**O26: Consider any legal, social, ethical & professional issues upon implementation.	Completed

Evaluation	Effectively test the algorithm implemented, the system, and the respective data science model using recommended techniques. **R**O27: Evaluate the developed algorithm and the respective model against the evaluation metrics researched in the literature review. **R**O28: Create a test plan & test cases and perform unit, performance, and integration testing.	Completed
Documentation	Document the progression of the research project and inform about any challenges faced. **R**O29: Create a coherent report of new skills obtained, evaluations, contributions etc., and ensure that all the above-stated objectives are met.	Completed

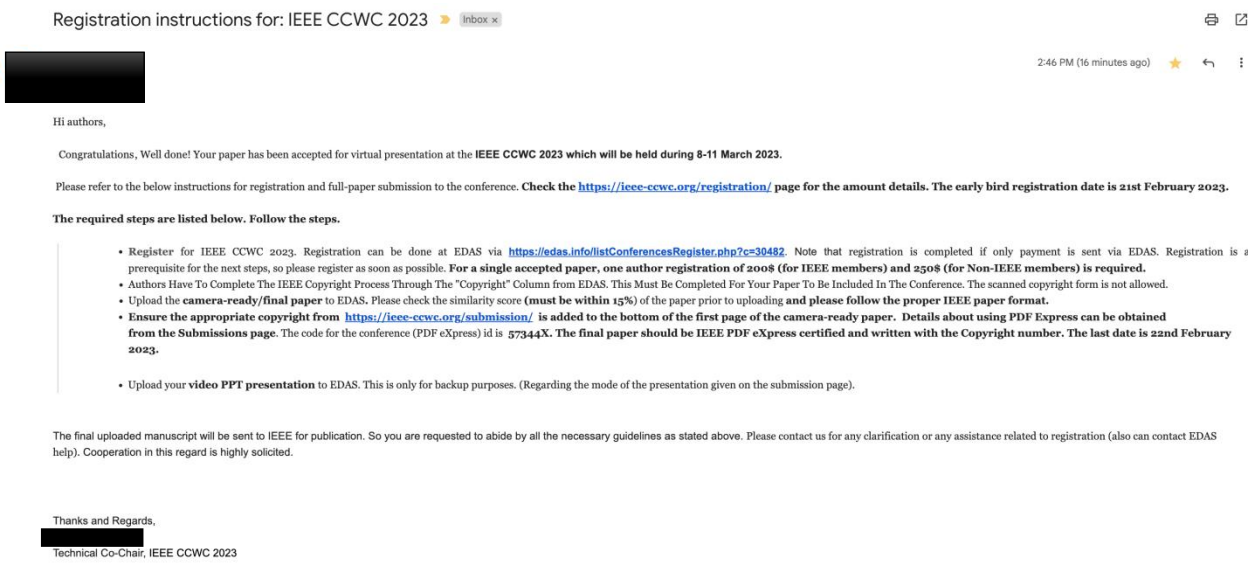
H.2. Achievement of learning outcomes

Table 53: Achievement of learning outcomes

Description	**Learning outcome(s)**
The project was broken down into two main 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 project plan as milestones to achieve within a set time frame and successfully complete the project within the given timescale.	LO2
Project requirements were collected and analyzed from two parties: academic researchers and end users to obtain insights into developing the LTS algorithm and prioritizing features that must be implemented in the BTC forecasting application.	LO3

Literature was read and critiqued upon, both recent and some over a century old, to understand in detail certain mathematical concepts.	LO4
Having obtained requirements, insights and knowledge. The author worked on implementing the two subproblems one unit at a time, whilst learning any new skills required. The author met with the supervisor regularly to make sure that they are on the right track by producing milestone deliverables. Any SLEP issues were also considered and documented.	LO5, LO6, LO7
The research was documented and each individual chapter was presented with the supervisor as milestones. They were then updated based on feedback from the supervisor and the module leader. Before completion of this dissertation, two document artefacts were submitted: the project proposal and PSDP. Additionally, papers were presented in conferences to justify the authors solution.	LO8

H.3. Extended review paper 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).

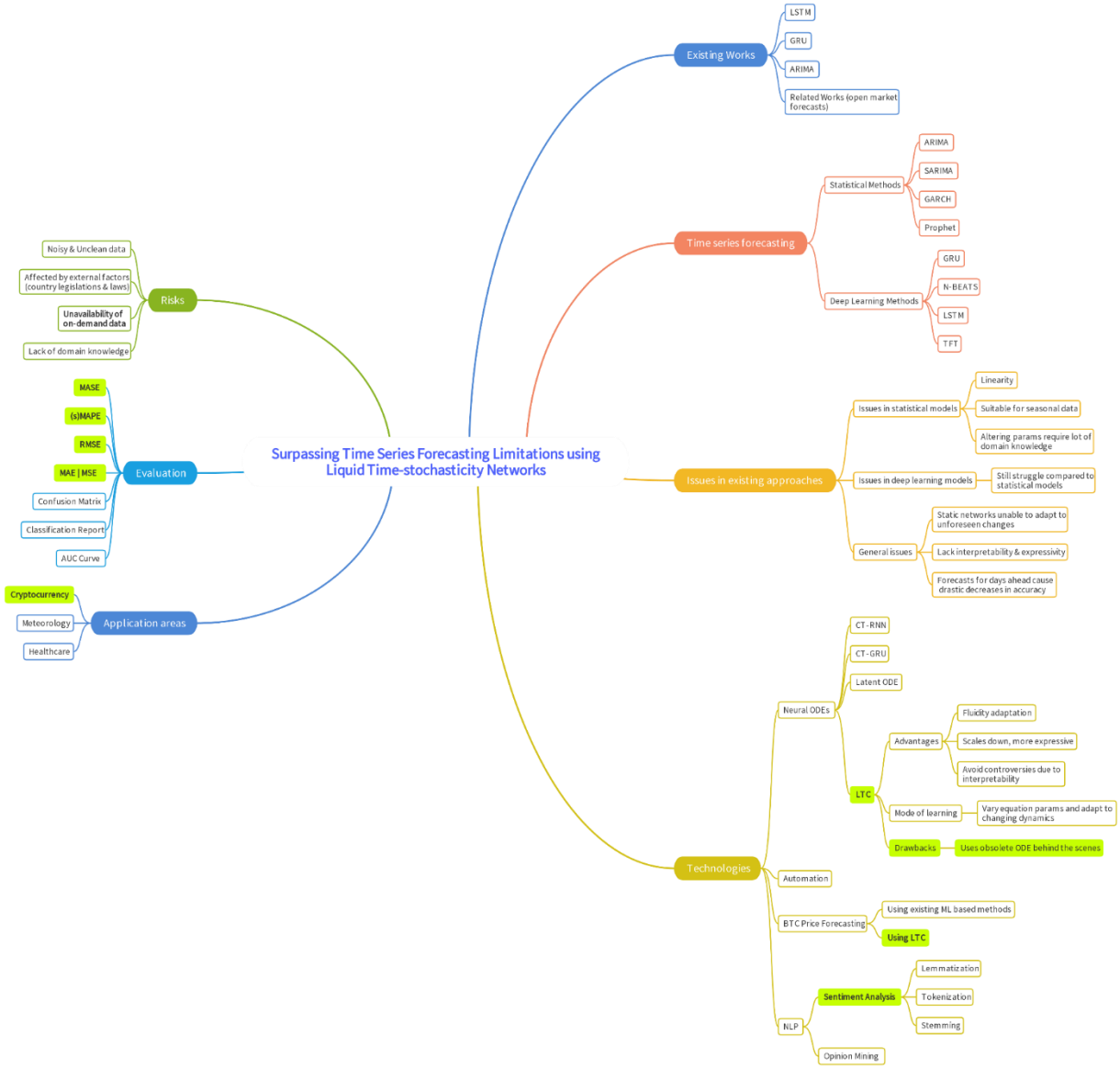
The final uploaded manuscript will be sent to IEEE for publication. So you are requested to abide by all the necessary guidelines as stated above. Please contact us for any clarification or any assistance related to registration (also can contact EDAS help). Cooperation in this regard is highly solicited.

Thanks and Regards,

[redacted]

Technical Co-Chair, IEEE CCWC 2023

APPENDIX I – CONCEPT MAP



A clearer version can be found [Here]