Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK

The University of Westminster, Coat of Arms

A Novel Approach to Time Series Forecasting using Liquid Time-constant Networks

A Product Specification & Prototype Design by

Mr. Ammar Raneez

W1761196 | 2019163

Supervised by

Mr. Torin Wirasingha

February 2023

This Project Proposal 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 advancements have been stagnant lately due to limitations of the existing algorithms; therefore, systems built using these algorithms can also go only so far in performance. Although recent advancements in Deep Learning (DL) have produced significant contributions in the domain of Natural Language Processing (NLP) and Reinforcement Learning (RL), the field of TS still needs to be improved.

Multiple approaches have been taken to solve this problem, the more promising being Neural Ordinary Differential Equations (ODEs) that introduce the concept of continuous time and depth models. Although the idea seems promising, the results are unexpectedly underwhelming compared to traditional neural networks such as Long Short-Term Memory (LSTM). Having mentioned that, MIT researchers introduced Liquid Time-Constant networks (LTCs) to solve this issue; they surpassed performance expectations, however, which carries problems of its own, the primary being the obsolete architecture used behind the scenes. This research project proposes a novel implementation of the Liquid Time-Constant algorithm with a more modern architecture that uses Stochastic Differential Equations (SDEs). Furthermore, its application is demonstrated on a Time Series (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. The solution utilized traditional Backpropagation Through Time (BPTT) to produce more accurate results with the trade-off of consuming more memory, which further created a promising result in its evaluation and application in the chosen domain. However, utilizing adjoint sensitivities instead must be researched in the future as it is recommended.

Keywords: Time Series (TS) forecasting, Neural Ordinary Differential Equations (ODEs), Liquid Time-Constant networks (LTCs), 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

Acronyms

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.
NLP	Natural Language Processing.
ODE	Ordinary Differential Equations.
POC	Proof-Of-Concept.
REST	Representational State Transfer.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
SOTA	State Of the Art.
SDE	Stochastic Differential Equations.
SGD	Stochastic Gradient Descent.
TS	Time Series.
UI	User Interface.
XAI	Explainable Artificial Intelligence.

CHAPTER 01. INTRODUCTION

1.1. Chapter overview

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-constant 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 justifiable research gap, respective research objectives, and challenges that would arise. The novelty within the chosen problem and the proposed solution are also stated.

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), while regular ML methods were nowhere near.

Therefore, this competition's winner was a hybrid model of LSTM and classical statistics (Smyl, 2020). 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 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? Additionally, 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).

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

As of writing this report, there is no application of LTC networks in any domain since this novel neural ODE has only recently been announced (Hasani et al., 2020). Existing intelligent systems utilize more traditional approaches of neural nets developed some time ago.

Having mentioned that, 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; consequently, they could perform relatively poorly when used in areas of high volatility (in this case: the forecasting of BTC).

Building an algorithm inspired by the LTC architecture and its application on an ML domain that still can struggle could be the stepping stone for future intelligent systems by aiding further research. Additionally, as a supplement, it could provide hope to crypto investors for more straightforward predictions.

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; implementing an algorithm capable of having the 'liquid' adaptability mentioned could be an advancement for more capable, accurate, and interpretable TS forecasting systems.

1.4. 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?

1.5. Research aim & objectives

1.5.1 Research aim

The aim of this research is to design, develop & evaluate a novel algorithm 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 said algorithm, focused on the forecasting of BTC.

1.5.2 Research objectives

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

Table 1: Research Objectives (*Self-Composed*)

Objective	Description	Learning Outcomes	Research Questions
Literature Review	Collate relevant information by reading, understanding, and evaluating previous work. **R**O1: Conduct preliminary studies and investigations on existing TS forecasting systems. **R**O2: Analyze the requirement for specialized TS algorithms. **R**O3: Conduct a preliminary study on neural ODEs, LTCs & SDEs. **R**O4: Obtain deep insights into the architecture behind the LTC.	LO1, LO2, LO4, LO5	**R**Q1
Requirement Elicitation	Collect and analyze project requirements using appropriate tools and techniques. **R**O5: Gather the requirements and architectures of LTCs and SDEs. **R**O6: Collate the most up-to-date details of BTC. **R**O7: Design a schedule, associated deliverables, and the Gantt chart.	LO1, LO2, LO3	**R**Q1

Design	Design the architecture and a corresponding system capable of effectively solving the identified problems. **R**O8: Design an efficient and novel LTC implementation. **R**O9: Design an automated flow to update the built network with the latest data. **R**O10: Design an ML pipeline for easy deployments.	LO1	**R**Q2
Implementation	Implement a system that is capable of addressing the research gaps. **R**O11: Implement a novel LTC architecture. **R**O12: Integrate the algorithm developed into a TS forecasting application. **R**O13: Integrate the intelligent system into the prototype to display forecasts. **R**O14: Consider any legal, social, ethical & professional issues upon implementation.	LO1, LO5, LO6, LO7	**R**Q2
Evaluation	Effectively test the algorithm implemented, the system, and the respective data science model using recommended techniques. **R**O15: Create a test plan & test cases and perform unit, performance, and integration testing. **R**O16: Evaluate the developed algorithm and the respective model against the mentioned evaluation metrics.	LO4	**R**Q2, **R**Q3
Documentation	Document the progression of the research project and inform about any challenges faced. **R**O17: Document any legal, social, ethical & professional issues faced and how they were solved.	LO6, LO8	-

	RO18: Create a coherent report of new skills obtained throughout the project plan and critical evaluations.		

1.6. Novelty of the research

1.6.1 Problem novelty

The core novelty of this research can be defined as the lack of adaptability to changes in existing TS algorithms and respective systems built utilizing them (Hasani et al., 2021). In other words, they are static.

1.6.2 Solution novelty

A solution for this problem is a dynamic algorithmic architecture that can adapt and change its underlying mathematical expressions and evaluation strategies based on changes in the characteristics of the incoming data streams. Further enhancements are required to avoid sudden and tiny changes common in TS data.

1.7. Research gap

The literature defines only a single paper for the proposed algorithmic solution (Hasani et al., 2020). Where every other work is not directly related to the algorithm 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. Furthermore, no algorithmic solution exists for the proposed LTC architecture for model implementation.

Gap in existing forecasting algorithms

It is also worth noting that, because of this, 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 chosen algorithm

The proposed LTC architecture uses a sequence of linear ODEs, which are now considered obsolete and lack instantaneous adaptability. Recent advancements in this field suggest the usage of SDEs instead, as they are more flexible (Duvenaud, 2021). An additional issue is that ODEs model 'deterministic dynamics' – uncertainty, or any unobserved interactions cannot be modelled, which is inevitable in TS data.

1.8. Contribution to the body of knowledge

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

- Would a novel architecture built by a novel algorithm utilizing an LTC architecture with SDEs instead of ODEs be an advancement for TS forecasting?

1.8.1 Contribution to the research domain

An implementation of the LTC algorithm with the abovementioned change 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 it can be applied elsewhere to evaluate its performance and identify whether the architecture would also be an advancement to other domains.

1.8.2 Contribution to the problem domain

Having understood the issues in the current literature, a solution capable of solving the mentioned issues could advance for future research. Adapting to unforeseen changes and being highly expressive could be the stepping stone within the TS forecasting community.

Moreover, 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.

1.9. 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 would make 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 (A naive forecast is not necessarily bad, 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.

1.10. 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. SOFTWARE REQUIREMENTS SPECIFICATION

2.1. Chapter overview

In this chapter, the author focuses on identifying the requirements and the steps followed to gather these requirements. In detail, possible stakeholders, alongside 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 to analyze and produce functional and non-functional requirements, use case diagrams, and prototype descriptions are defined.

2.2. Rich picture

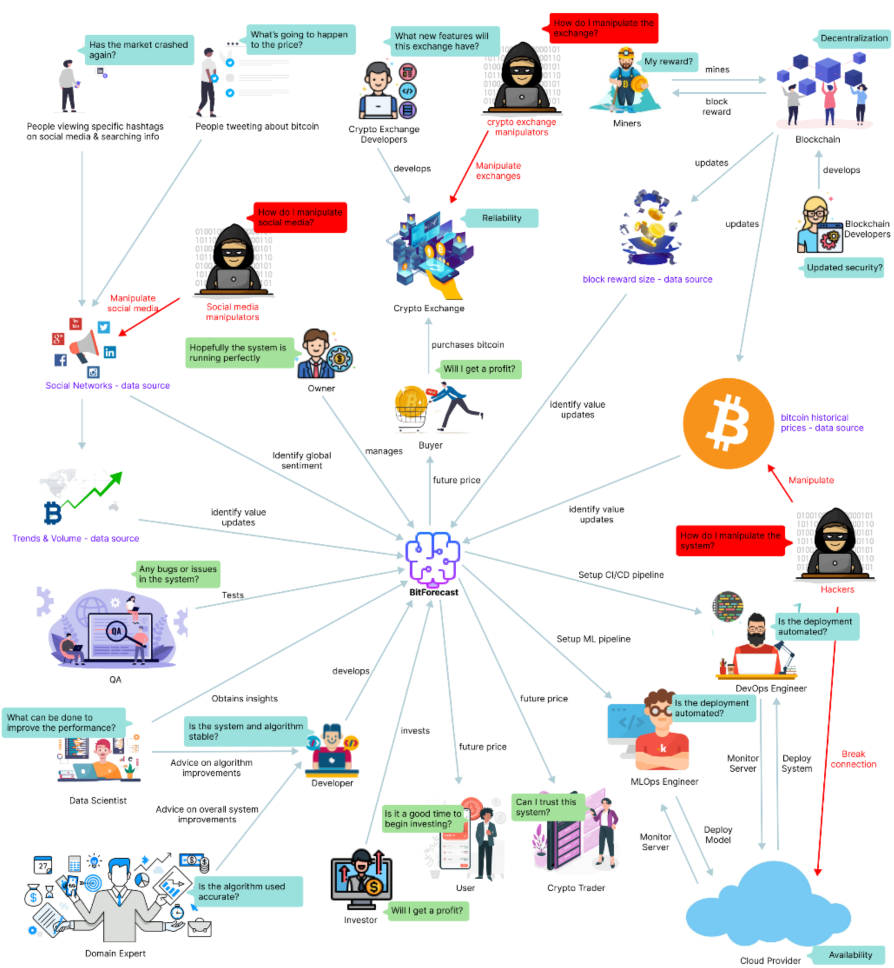


Figure 1: Rich picture diagram (*Self-Composed*)

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 possibilities of negative impact on the design and possible critical analysis are identified, alongside the knowledge the researcher could receive to improve the system.

2.3. 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.

2.3.1 Stakeholder onion model

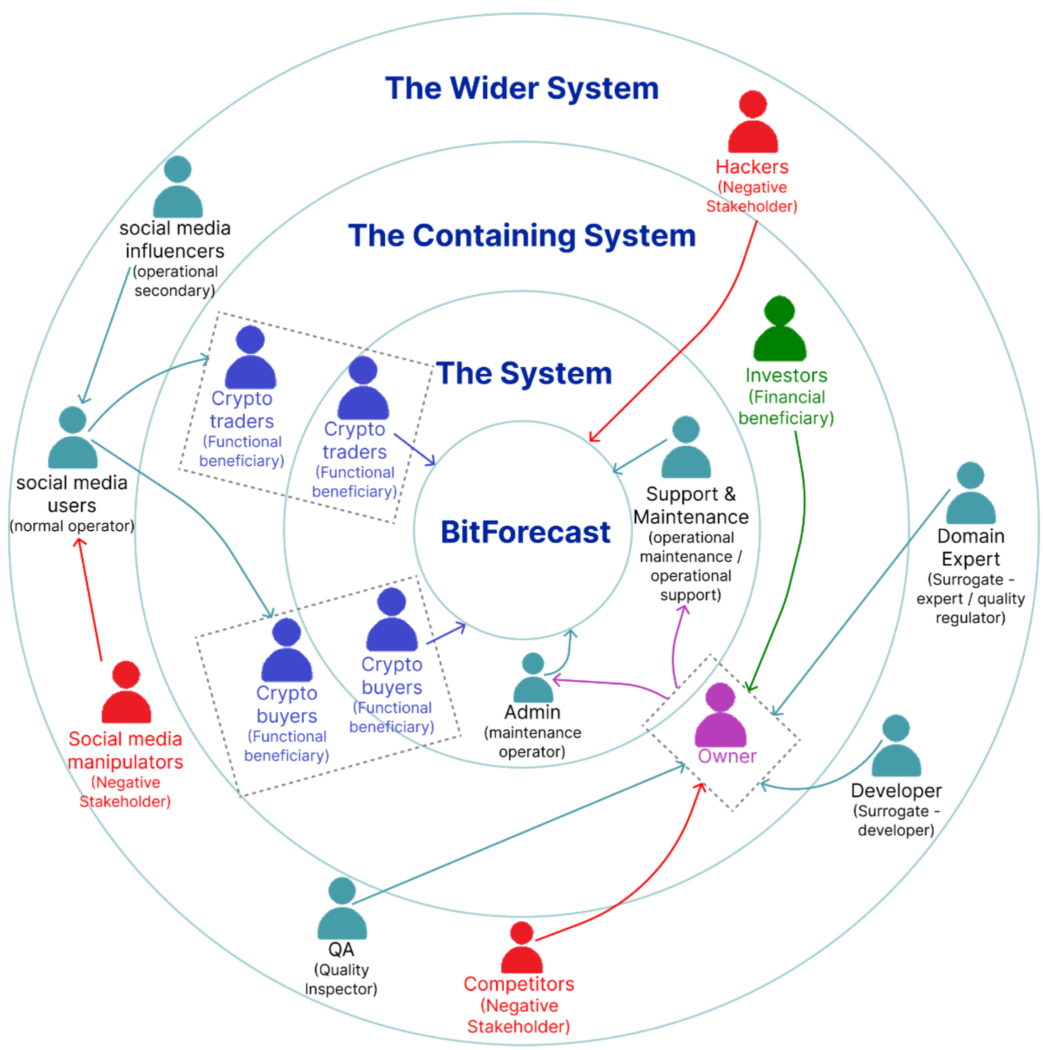


Figure 2: Stakeholder onion model (*self-Composed*)

2.3.2 Stakeholder viewpoints

Table 2: Stakeholder viewpoints (*self-Composed*)

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.
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	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.
QA	Quality inspector	Tests the system's quality to ensure stability.
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.

2.4. Selection of requirement elicitation methodologies

Requirement elicitation methodologies can be carried out to gather requirements. The following table discusses the selected ones and their purpose.

Table 3: Requirement elicitation methodologies (*Self-Composed*)

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 the mentioned hypothesis 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: Survey
Obtaining insights and expectations from end users can be gathered by conducting a survey, specifically, the questionnaire. Upon receiving this prominent information, they 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 pretty large in sample size, the survey is a powerful choice for data collection.
Method 04: 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 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.

2.5. Discussion of findings

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

2.5.1 Literature review

Table 4: Literature review findings (*Self-Composed*)

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 that the author aims to include. Therefore, using a non-linear model with multiple features would produce the optimal solution.

2.5.2 Observations

Table 5: Observations findings (*Self-Composed*)

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 supplementary BTC forecasting system utilizing as many exogenous features as possible that can be of effect. Therefore, insights into implementing the supplementary system and effective evaluation techniques were acquired

2.5.3 Survey

A survey was conducted to gather requirements from the target audience to infer functionalities to implement for the supplementary product developed. As this is not the primary focus of the research and respectively not the main form of data collection, it is placed in **APPENDIX A.2**.

2.5.4 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 their respective expertise area are also available in **APPENDIX A.3**.

Table 6: Interview thematic analysis codes, themes & conclusions (*Self-Composed*)

Code	**Theme**
Research component	
Algorithm architecture	Research Problem & Gap
Resource intensive	Requirements
Obsolete, Inflexible	Advice

Visualizations, Explainability	Other suggestions
Problem domain	
External features and trends	Robustness

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	"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."

	a novel algorithm, which is the author's current aim as it carries more 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."

2.5.5 Prototyping

Table 7: Prototyping findings (*Self-Composed*)

Criteria
To explore the feasibility of creating the primary research component.
Discussion of findings
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. Building

the algorithm is intimidating, as no proper reference exists. They realized that, alongside traditional DL theories, implementing the algorithm required more profound knowledge and understanding of SDEs and differential solvers. Furthermore, they had depended on the Twitter API to get tweet sentiment of specific days; however, this was impossible as Twitter had updated the 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.

2.5.6 Summary of findings

ID	Finding	Literature Review	Observations	Survey	Interview	Prototyping
Research component						
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.			✓		

2.6. Context diagram

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

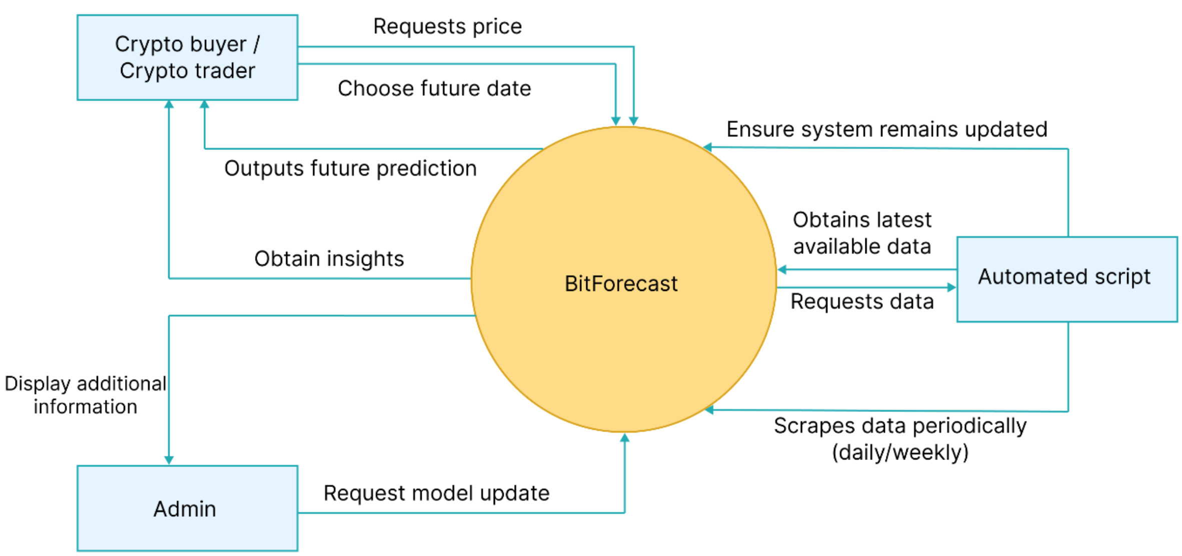


Figure 3: Context diagram (*Self-Composed*)

2.7. Use case diagram

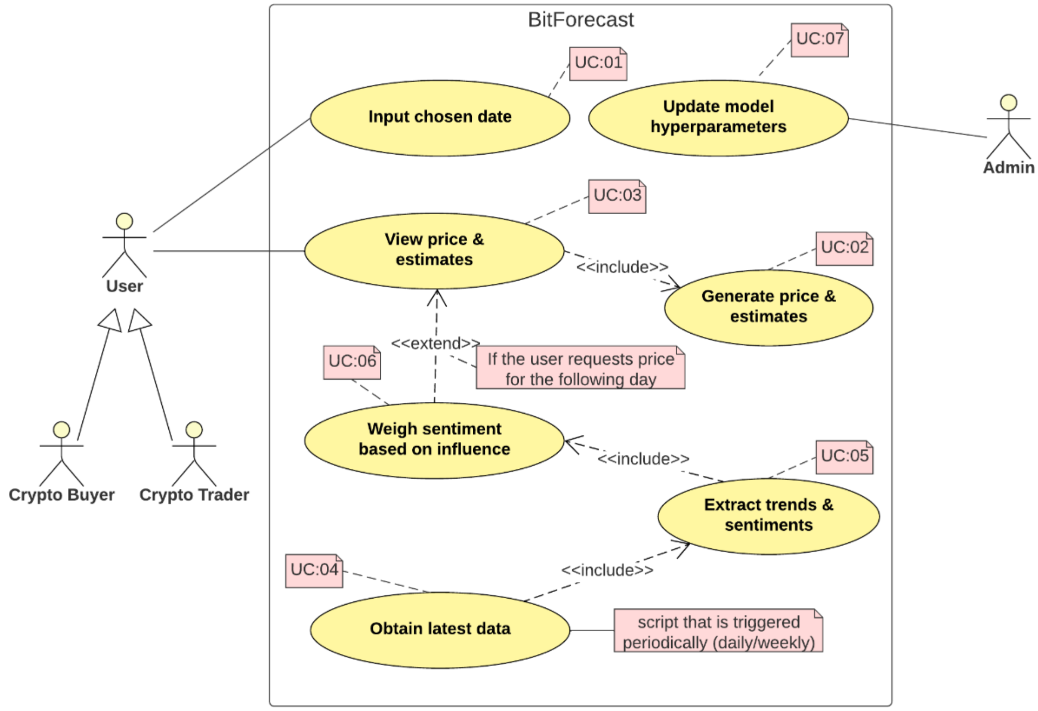


Figure 4: Use case diagram (*Self-Composed*)

2.8. Use case descriptions

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

Table 8: Use case description UC:03; UC:04 (*Self-Composed*)

Use case	Display price & estimates
Id	UC:03; UC:04
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 (if any)	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.

2.9. Requirements

2.9.1 Functional requirements

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

Table 9: Functional requirements (*Self-Composed*)

ID	Description	Priority	Use Case
Research level			
FR1	A robust and scalable implementation of the novel algorithm must be implemented that follows 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 data.	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 should 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

2.9.2 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 10: Non-functional requirements (*Self-Composed*)

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

2.10. 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, they specified the use cases, associated descriptions, and system requirements.

CHAPTER 03. DESIGN

3.1. Chapter overview

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 and the reasoning behind each choice. Moreover, the novel algorithm architecture is also proposed.

3.2. Design goals

Table 11: Design goals of the proposed system (*Self-Composed*)

Goal	**Justification**
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. The model can automatically be retrained beginning each day (which would deem the retraining step on each inference unnecessary) as the solution proposed.
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.
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. Additionally, providing insights into how the model made the inference is an added benefit if time permits.

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.

3.3. High level design

3.3.1. 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.

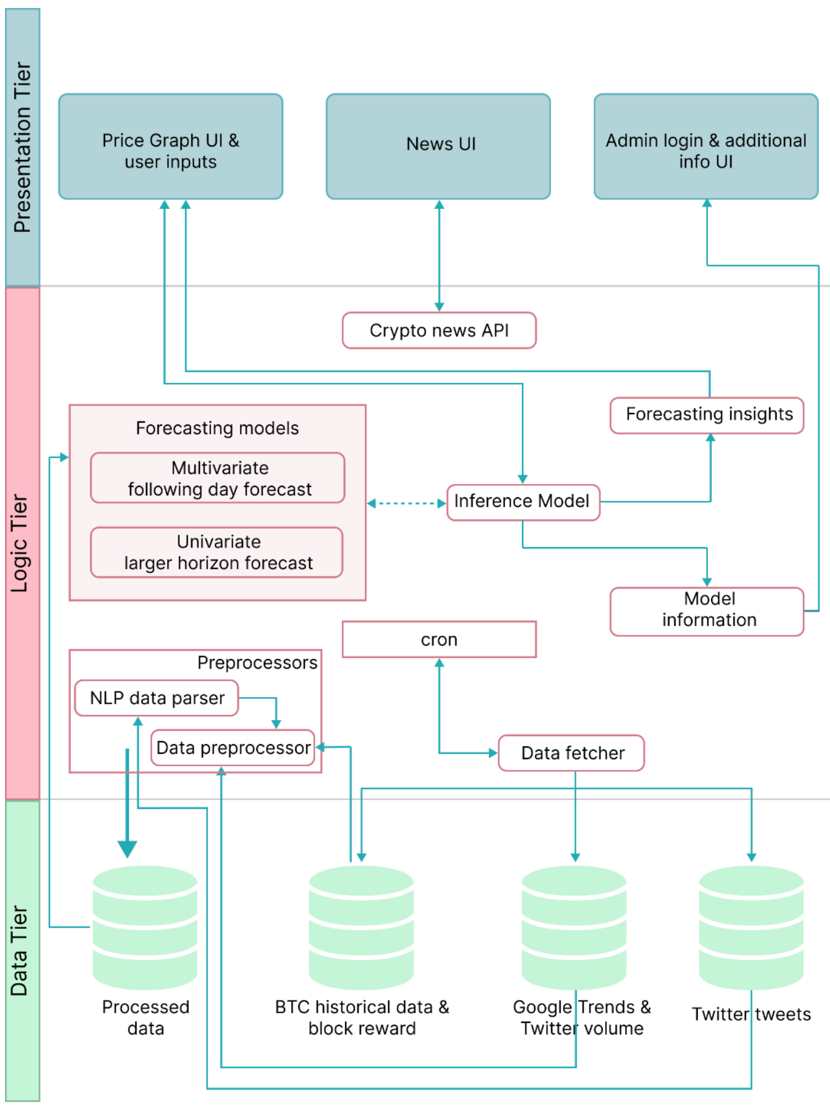


Figure 5: Three-tiered architecture (*Self-Composed*)

3.3.2. Discussion of tiers of the architecture

Data Tier

All data in this layer are fetched from an API and stored in individual documents 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 of data.
    - NLP data parser – required to perform sentiment analysis on the tweet data and named entity recognition to give more weightage to 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 for 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 news about cryptocurrency.

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.

3.4. System design

3.4.1. 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.

3.5. Design diagrams

3.5.1. 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.

3.5.1.1. Level 01 data flow diagram

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

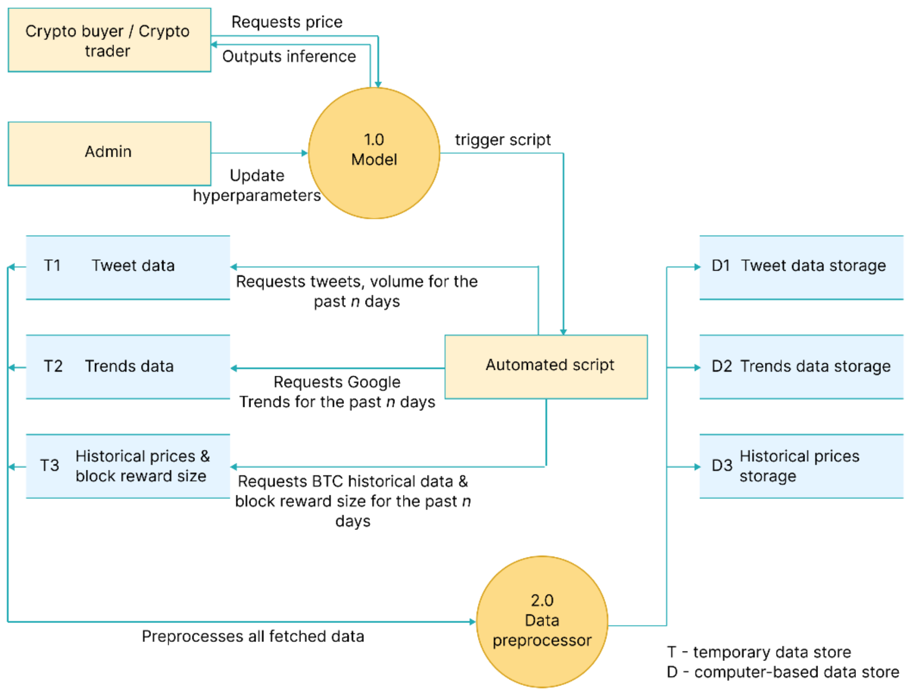


Figure 6: Data flow diagram - level 01 (*Self-Composed*)

3.5.1.2. Level 02 data flow diagram

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

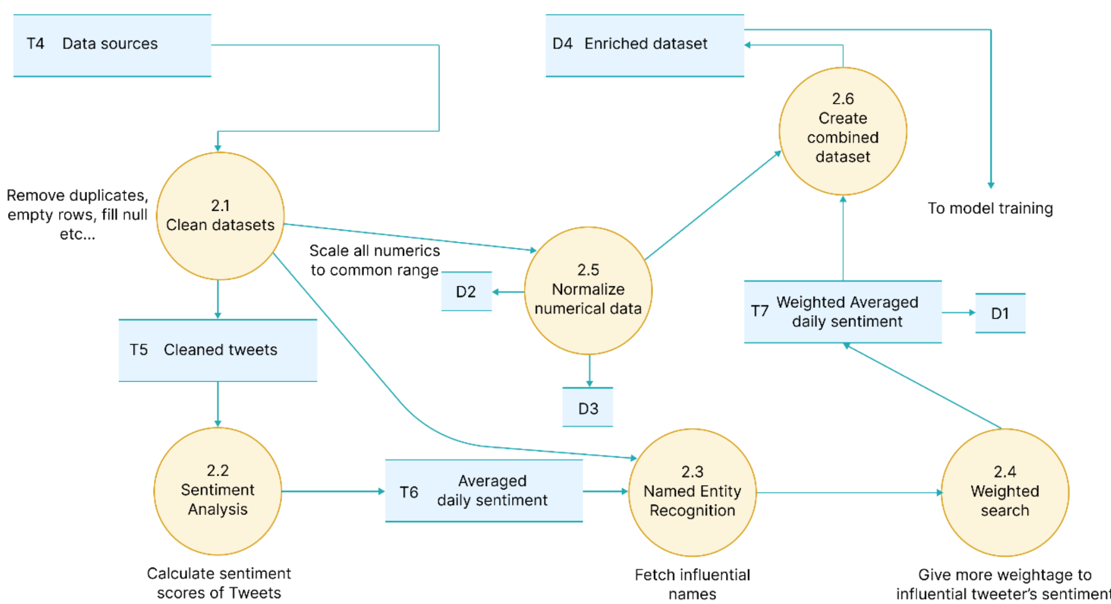


Figure 7: Data flow diagram - level 02 (*Self-Composed*)

3.5.2. Algorithmic design

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 as their primary contribution to the body of knowledge. A simple illustration is available in **APPENDIX B.1** to gather intuition.

3.5.2.1. 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.

3.5.2.2. Algorithm proposed by the author

Upon studying the abovementioned architecture, the author could utilize a linear system of SDEs to declare the flow to manifest a potentially 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.

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.

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

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:

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

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

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:

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

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

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

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

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

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

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

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).

3.5.3. Algorithmic analysis

The notable difference between the proposed architecture and traditional neural ODEs proposed by Chen et al. (2019) is the usage of the traditional BPTT approach instead of the recommended adjoint sensitivity. The below table demonstrates the difference in the complexities of these approaches.

Table 12: 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

What can be noticed from the above table is that the traditional BPTT approach yields more accurate results, with the trade-off of consuming more memory. Therefore, to obtain the best result possible, the author chose the approach of the traditional BPTT.

3.5.4. System process activity diagram

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

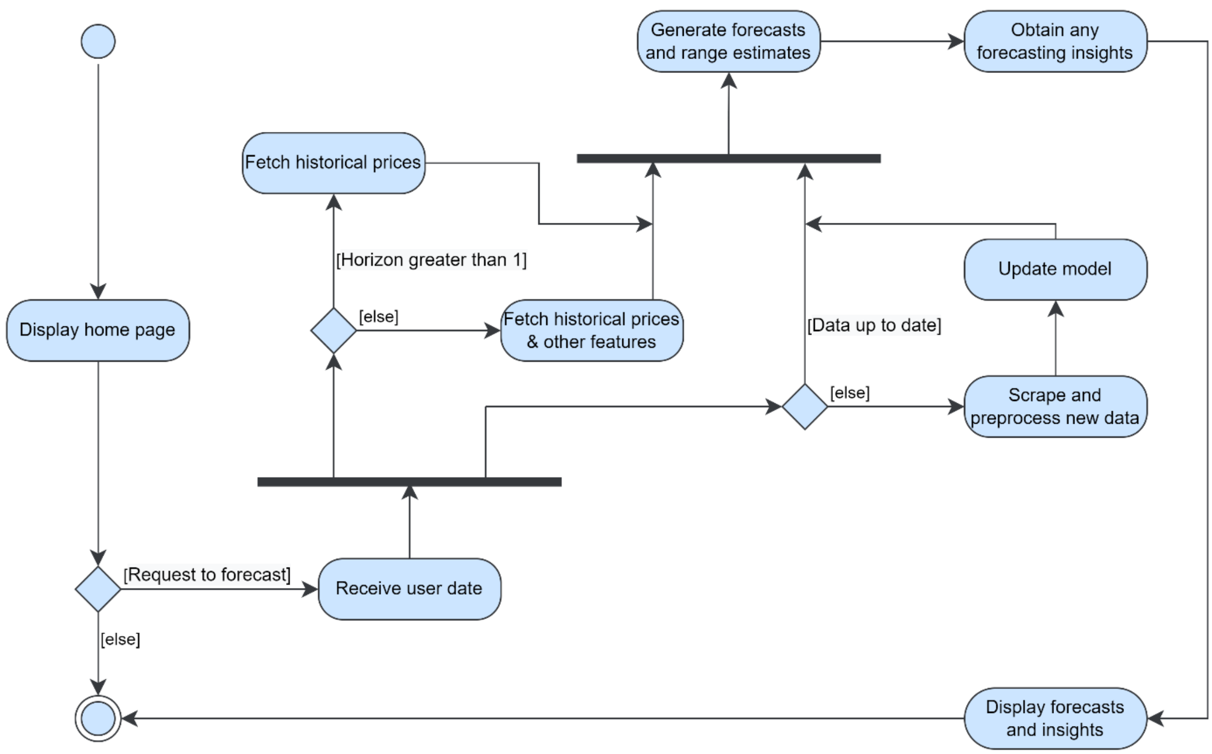


Figure 8: System process activity diagram (*Self-Composed*)

3.5.5. 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 B.2**.

3.6. Chapter summary

This chapter presented the design of the core novel algorithmic architecture, the necessary intuition behind it, and the reasons for taking specific directions over others. Additionally, it illustrated the system's design, architecture, and data and system flow alongside the wireframes that would demonstrate them in the end application.

CHAPTER 04. INITIAL IMPLEMENTATION

4.1. Chapter overview

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 alongside their reasoning.

4.2. Technology selection

4,2.1. Technology stack

The chosen technologies are depicted in the diagram below.

Figure 9: Tech stack (*Self-Composed*)

4.2.2. Selection of data

As this is a data science project, the highest quality of data is a necessity. The author utilized multiple sources of data that are potential contributions to the target inference; the following were required:

- BTC historical data
- BTC block reward size
- BTC tweets
- BTC Twitter volume
- BTC Google Trends

The univariate single horizon forecasting model utilized the above data in a combination, while the multivariate multi-horizon forecasting model utilized solely the historical data. The below table describes the sources of each respective dataset.

Table 13: Dataset sources (*Self-Composed*)

Dataset	**Source**
BTC historical data	From a third-party investing.com API.
BTC block reward size, BTC Twitter volume	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 till date were extracted from a Twitter tweet scraper.
BTC Google Trends	From the PyTrends library that provides Google Trends data.

Gathering the data was a long and arduous process as it was not as simple as downloading available datasets, and certain APIs being rate-limited. Dedicated python scripts were written to extract the data and to streamline updating available data. The author will publicize these scripts and the data to facilitate future research.

4.2.3. Selection of programming language

Programming languages were analyzed prior to 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 C.1**, the author decided to use **Python**.

To develop the user interface not much competition is present to analyze. **JavaScript** is the stand-alone leader and is the choice of the author 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.

To setup the communication between the model and the user interface APIs are required. 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.

4.2.4. Selection of development framework

4.2.4.1 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 choose the most relevant framework. The two most popular frameworks, TensorFlow and PyTorch, were analyzed in **APPENDIX C.2**, where the author opted to use **TensorFlow**.

4.2.4.2. 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 chosen for evaluation; the four being Angular, Vue, Svelte, and React. The author decided to use **React**; the evaluation of them can be found in **APPENDIX C.3**.

4.2.4.3. API web framework

As python was chosen for the API development, respective Python web frameworks must be analyzed to choose the more relevant one. Analysis was conducted between Django and Flask as they are the two most popular frameworks. **APPENDIX C.4** demonstrates the comparison where the author decided to use **Flask**.

4.2.5. Other libraries & tools

Table 14: Chosen libraries (*Self-Composed*)

Library	**Justification**
NumPy	Facilitates mathematical functions and calculations that is 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 potentially 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.
TensorBoard	Visualize and obtain insights of the model training process associated evaluation metrics and additional dashboarding.
Redux	For API requests from the client.
Ant design	Makes creating appealing user interfaces hassle-free.

4.2.6. Integrated Development Environment (IDE)

Table 15: Chosen IDEs (*Self-Composed*)

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, provides easy integration with existing Kaggle datasets and user-uploaded datasets.
Jupyter	For local trials and testing.
VSCode	Lightweight and extremely powerful. Consists of multiple shortcuts, extensions and snippets that can significantly boost development productivity.

4.2.7. Summary of chosen tools & technologies

Table 16: Chosen tools & technologies (*Self-Composed*)

Component	**Tools**
Programming languages	Python, JavaScript
Development framework	Flask, TensorFlow
UI development framework	Ant design
Libraries	React, NumPy, Pandas, Scikit-learn, Beautiful Soup, Lingua, Matplotlib, Seaborn, VADER sentiment analyzer.
IDEs	Kaggle and Jupyter notebooks; VSCode.
Version control	Git + GitHub

4.3. Implementation of core functionalities

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

4.3.1. 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.

```python
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 10: 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).

```python
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 11: 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 12: 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 13: 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 14: Algorithm – define weights and biases (*Self-Composed*)

As determined in previous chapters, 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 15: Algorithm – Euler-Maruyama SDE solver (*Self-Composed*)

4.3.2. Data fetchers

The data fetchers are scripts that are used to extract the data to be used by the model. The scripts are placed under **APPENDIX C.5**.

4.3.3. 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 C.6**.

4.4. 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 demonstrated with accompanying code snippets.

CHAPTER 05. CONCLUSION

5.1. Chapter overview

This chapter provides an initial conclusion to the research project focused on implementing the core components required to consider it a functional prototype. In detail, any deviations taken from the proposed scope and the schedule in the project proposal are mentioned. Moreover, any additional improvements required to produce an MVP alongside the current evaluation results are specified.

5.2. Deviations

5.2.1. Scope related deviations

The features in scope proposed in the project proposal are stated below.

In scope

- Implementing a novel LTC architecture capable of being used as currently existing solutions and the corresponding creation of a system.
- Periodical 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.

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.

Based on the proposed scope, no deviations have been taken in the category of “in-scope”. Unfortunately, none of the “desirables” category had been implemented due to time constraints.

5.2.2. Schedule related deviations

The schedule proposed by the author is available in **APPENDIX D.1**. Based on the proposed Gantt chart, the author's journey so far has not had any major deviations. However, a single task (no. 45) that mentions "implementing supplementary components" scheduled to be completed by January 23rd is still in progress. The progress of the Gantt chart with the updated dates provided is available in **APPENDIX D.2**.

5.3. Initial test results

Two different models were created for the prototype implementation. A univariate and multivariate model, for forecasts greater than a horizon of 1 and forecasts for a horizon of 1 respectively.

In open market forecasting it is a notorious difficulty to surpass the "naïve" forecast, which fortunately was the case for the models implemented. The below two snippets demonstrate the comparison of these models against the naïve forecast via evaluation metrics that are specified in **APPENDIX D.3**, for the test dataset.

5.3.1. Univariate model

	mae	mse	rmse	mape	mase
naive_forecast_results	951.947937	2021966.0	1421.958496	2.5654945373535156%	0.999748
ensemble_results	950.300842	2013928.375	1419.129395	2.557239532470703%	0.99827

Figure 16: Univariate model evaluation *(Self-Composed)*

5.3.2. Multivariate model

	mae	mse	rmse	mape	mase
naive_forecast_results	858.99939	1631689.25	1277.375977	2.419470787048340%	0.998859
ensemble_results_2	918.780212	1788472.375	1337.337769	2.618018627166748%	1.069594

Figure 17: Multivariate model evaluation (*Self-Composed*)

5.4. Required improvements

To consider this research successful, a couple of improvements are required.

- Enhance the performance of the system to the best possible accuracy – attempt more optimization procedures.
- Integrate the model in use to a GUI – GUI has been prepared; a simple Flask API should be created to establish a communication.
- Perform testing for each section of the application – conduct unit, performance, and integration testing.
- Compare the system's performance with existing solutions.

5.6. Demo of the prototype

A demo of the prototype was recorded and uploaded as an unlisted video on YouTube, the video can be found in `https://youtu.be/HMUa9JOKcJQ`

5.8. Implementation code

The code written for the prototype are stored in GitHub for convenience. The code can be found in the below two 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`

5.7. Chapter summary

This chapter provided the reader with an overview of the current status of the ongoing research project, including, but not limited to - deviations taken from the proposed features and schedule, the evaluation results, and any further improvements required.

REFERENCES

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

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].

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].

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].

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].

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

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].

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].

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].

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].

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].

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].

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].

A naive forecast is not necessarily bad. (2014). *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].

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].

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].

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].

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].

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].

Pontryagin, L.S., Boltyanskii, V.G., et al. (1962) The Mathematical Theory of Optimal Processes. Interscience Publishers John Wiley & Sons, Inc., New York-London, Translated from the Russian by K.N., Trirogoff.

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].

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

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.

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

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

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

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

APPENDIX A – SRS

A.1. Requirement elicitation methodologies

Table 17: Stakeholder groups (*Self-Composed*)

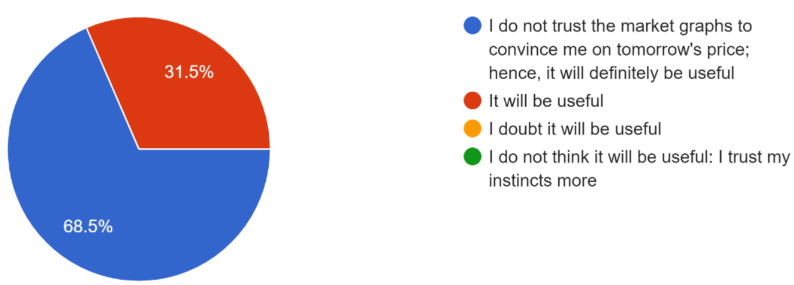
Group	Stakeholders	Reason	Instrument
G1	Domain experts (neural ODE/SDE and blockchain/crypto)	Gather any insights and knowledge specifically in the research domain to answer research questions and anything the author may have missed.	Interview
G2	End users (trader & buyer)	Gather requirements for supplementary 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

A.2. Survey analysis

Table 18: Survey analysis (*Self-Composed*)

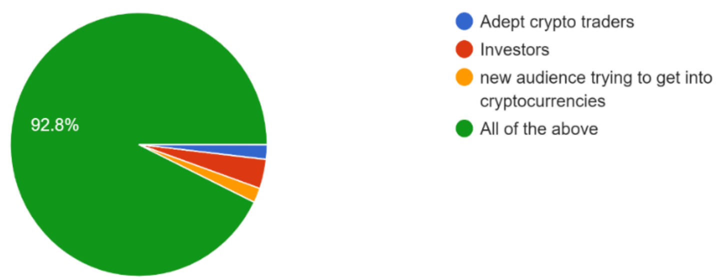
Question	How much would a system capable of assuming tomorrow's price benefit you?
Aim of question	To identify whether the system is beneficial in the first place

Findings & conclusions

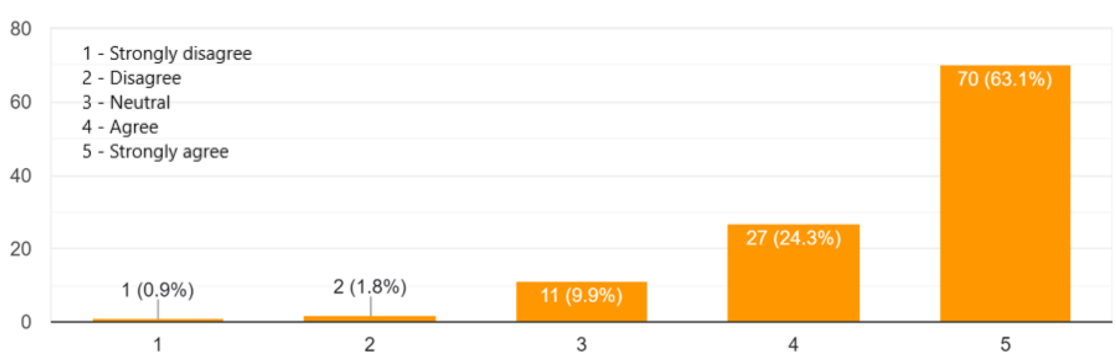


All the participants believed that the proposed system would be beneficial – where the majority had a greater belief than others. Having obtained this information, it is evident that the

<table>
<tr><td colspan="2">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.</td></tr>
<tr><td>Question</td><td>Who do you think would benefit from this system?</td></tr>
<tr><td>Aim of question</td><td>To identify beneficiaries and target audience</td></tr>
<tr><td colspan="2">Findings & conclusions

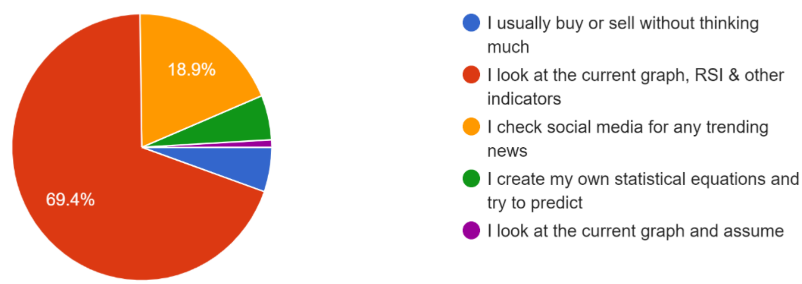


The majority of the participants believed that the system would be beneficial for expert traders, investors as well as a new audience. However, what can be identified, is that a minute portion of participants assumed that the system would be helpful primarily for people who are already involved in the 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.</td></tr>
<tr><td>Question</td><td>This system will also benefit people who are not experts in cryptocurrencies</td></tr>
<tr><td>Aim of question</td><td>To identify whether non-technical crypto traders would benefit</td></tr>
<tr><td colspan="2">Findings & conclusions

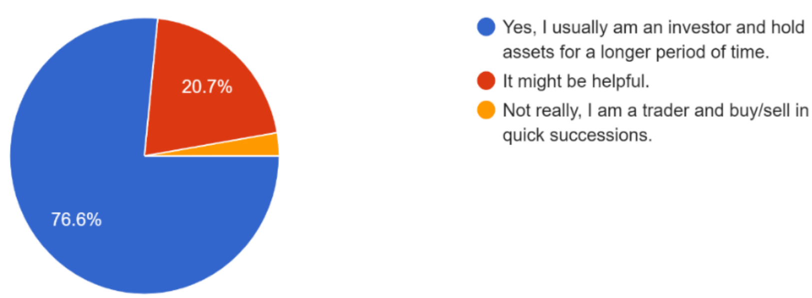


The responses to the above question show that the system will also apply to audiences who are not cryptocurrency experts. This question goes hand in hand with the previous question to</td></tr>
</table>

<table>
<tr><td colspan="2">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.</td></tr>
<tr><td>Question</td><td>How do you decide whether to buy or sell assets?</td></tr>
<tr><td>Aim of question</td><td>To understand how a buyer/seller proceeds with their decision</td></tr>
<tr><td colspan="2">Findings & conclusions

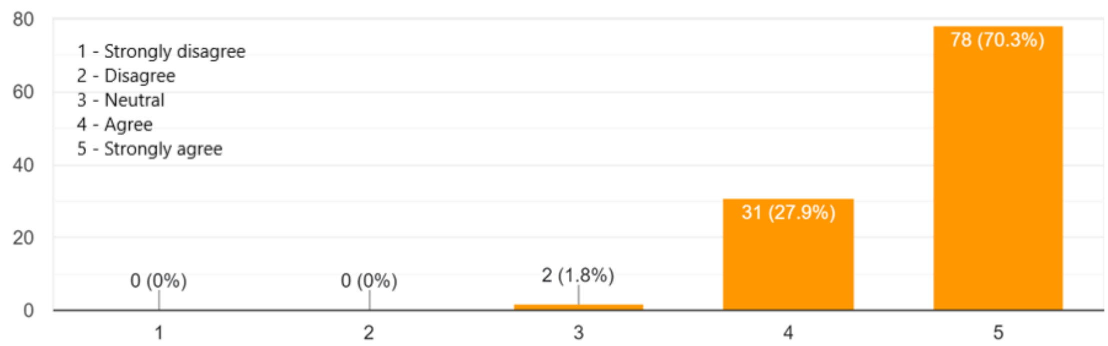


The responses to the above question are more of a ‘Know Your Customer’ question with no specific project-related purpose. Nevertheless, what can be identified is that most of the respondents have some knowledge of cryptocurrencies, where almost 70% are experienced in trading/investing cryptocurrencies – 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.</td></tr>
<tr><td>Question</td><td>Do you think predicting a more future date (ex: a week from now) is as important as tomorrow's price?</td></tr>
<tr><td>Aim of question</td><td>To identify whether a greater future date prediction is also necessary</td></tr>
<tr><td colspan="2">Findings & conclusions



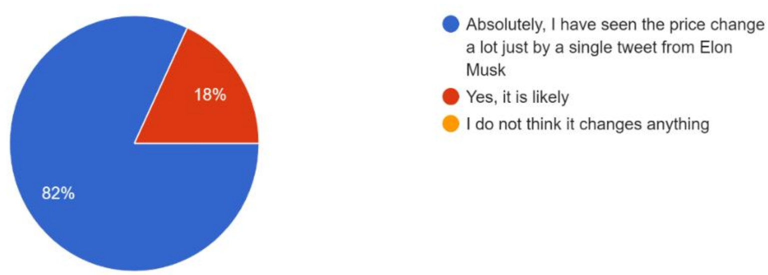
The author initially considered only having a single horizon forecast, considering the limited time. However, based on the above responses, it is evident that the audience would also expect forecasts for multi horizons. Therefore, the author will additionally aim to implement the ability of multi-horizon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

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	

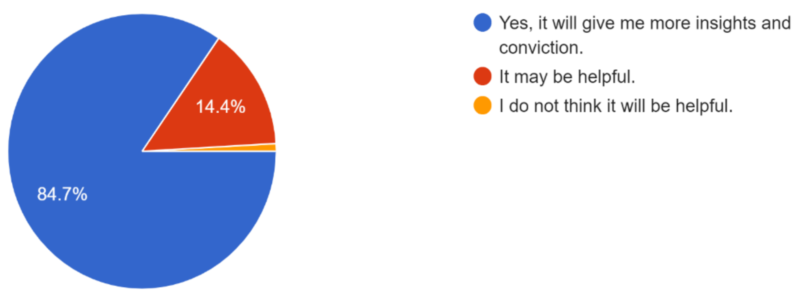


The majority of the respondents believe that social trends impact the price. Therefore, it is necessary to consider as many trends as possible. Considering the project's limited time and scope, the author has decided to use Twitter volume and Google Trends; however, Reddit, Facebook, and others would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

<table>
<tr><td colspan="2">Additionally, based on the responses, the requirement for NER and weighted search is more apparent to give more weightage to specific tweeter's sentiments.</td></tr>
<tr><td>Question</td><td>Would it be helpful to obtain a range of prices rather than a point price? (Ex: 10,000 - 15,000 instead of 12,500)</td></tr>
<tr><td>Aim of question</td><td>To identify whether including uncertainty estimates is beneficial</td></tr>
<tr><td colspan="2">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.</td></tr>
<tr><td>Question</td><td>What functionalities would you expect to have in a bitcoin forecasting system?</td></tr>
<tr><td>Aim of question</td><td>To identify any additional requirements</td></tr>
<tr><td colspan="2">Findings & conclusions
To analyze opened ended questions, the author can perform thematic analysis. The analysis, including the theme and related codes, is available below.
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.</td></tr>
<tr><td>Question</td><td>Any extra feedback you would like to provide?</td></tr>
<tr><td>Aim of question</td><td>No specific reason – is mainly used to obtain any additional feedback</td></tr>
</table>

Findings & conclusions
A few motivational sentences were submitted to inspire and motivate the author to perform to their best ability.

Table 19: Survey thematic analysis codes, themes & conclusions (*Self-Composed*)

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	"Show some news about the current cryptocurrency world in the platform, so it's convenient for the users."

	convenient and make the inference procedure as straightforward as possible so there is no hindrance.	"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 that the author believes they will not be able to cover, given the time allotted.	"Predict the market for any given time duration." "Ability to identify a pump and dump scenario compared to an actual increase in the price of stock/crypto."

A.3. Interview analysis

Table 20: Interview participant details (*Self-Composed*)

Participant 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.

A.4. Use case descriptions

Table 21: Use case description UC:05; UC:06 (*Self-Composed*)

Use case	Manage exogenous features
Id	UC:05; UC:06
Description	Manage and process new data without the need for manual interaction.
Actor	Script
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 22: Use case description UC:07 (*Self-Composed*)

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.

A.5. Functional requirements

Table 23: 'MoSCoW' technique of requirement prioritization (*Self-Composed*)

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 much 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 B – DESIGN

B.1. Algorithm intuition

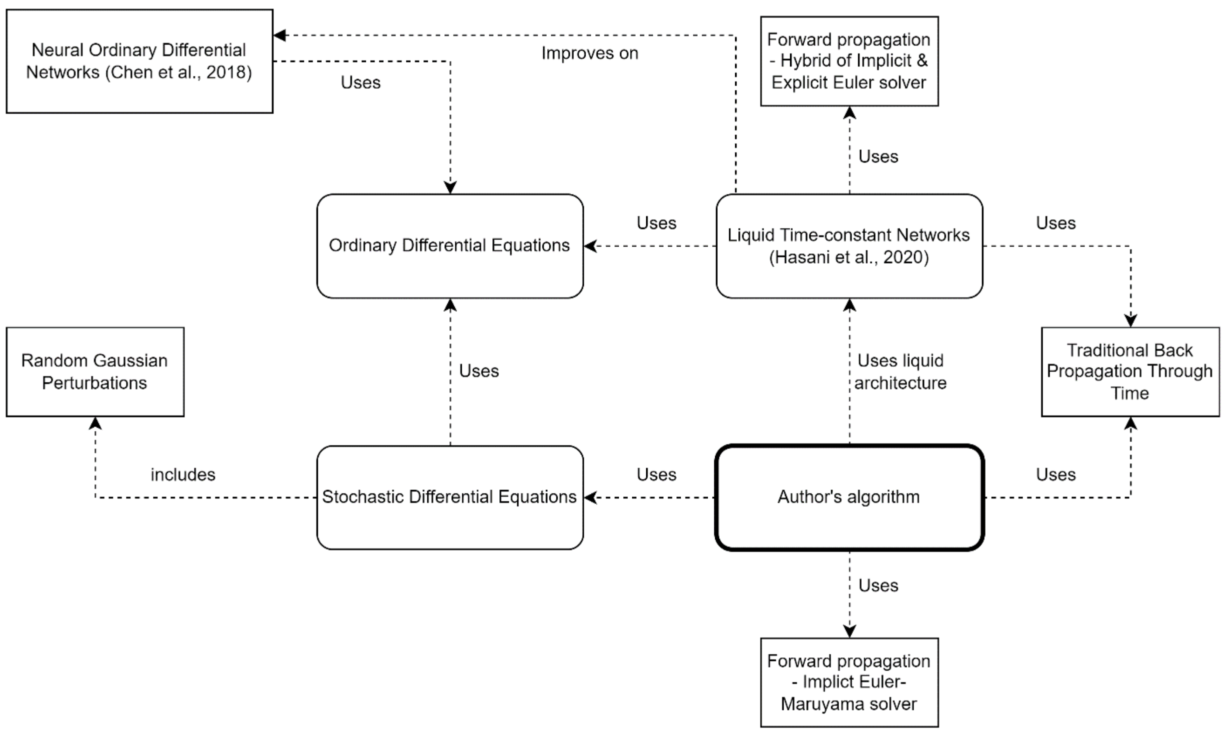


Figure 18: Algorithm intuition (*Self-Composed*)

B.2. UI wireframes

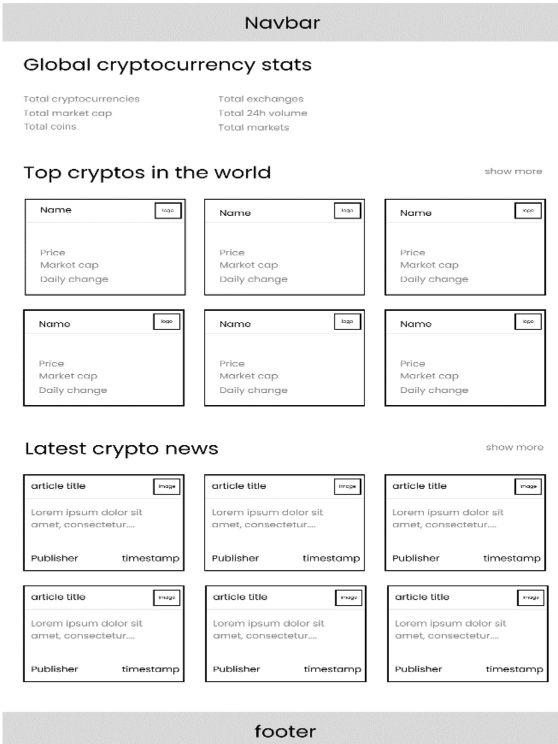


Figure 19: UI wireframes – Home (*Self-Composed*)

Figure 20: UI wireframes – News (*Self-Composed*)

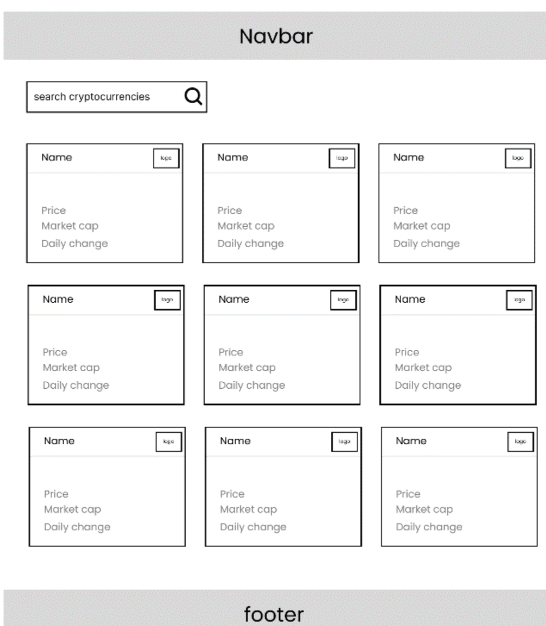


Figure 21: UI wireframes – Cryptocurrencies (*Self-Composed*)

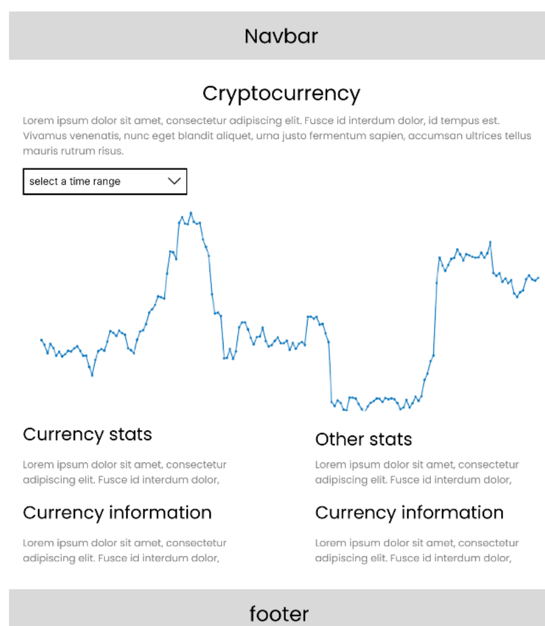


Figure 22: UI wireframes – Cryptocurrency (*Self-Composed*)

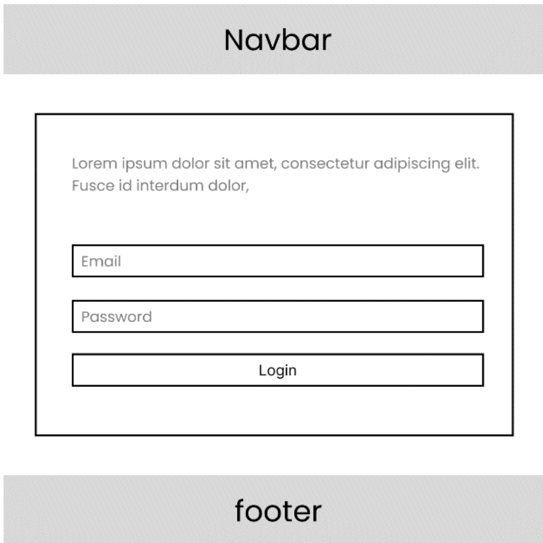


Figure 23: UI wireframes – Admin login (*Self-Composed*)

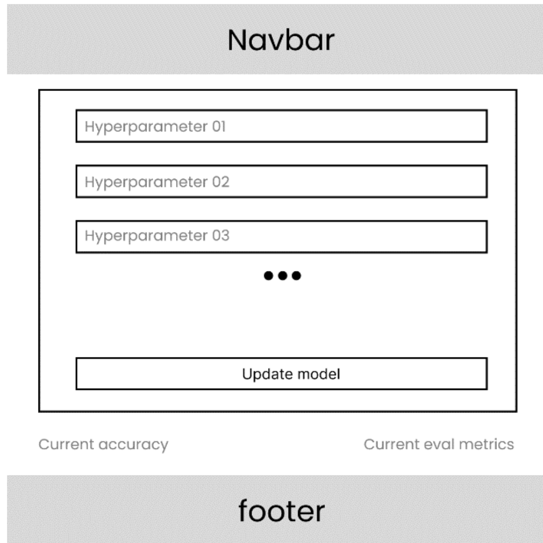


Figure 24: UI wireframes – Admin model configuration (*Self-Composed*)

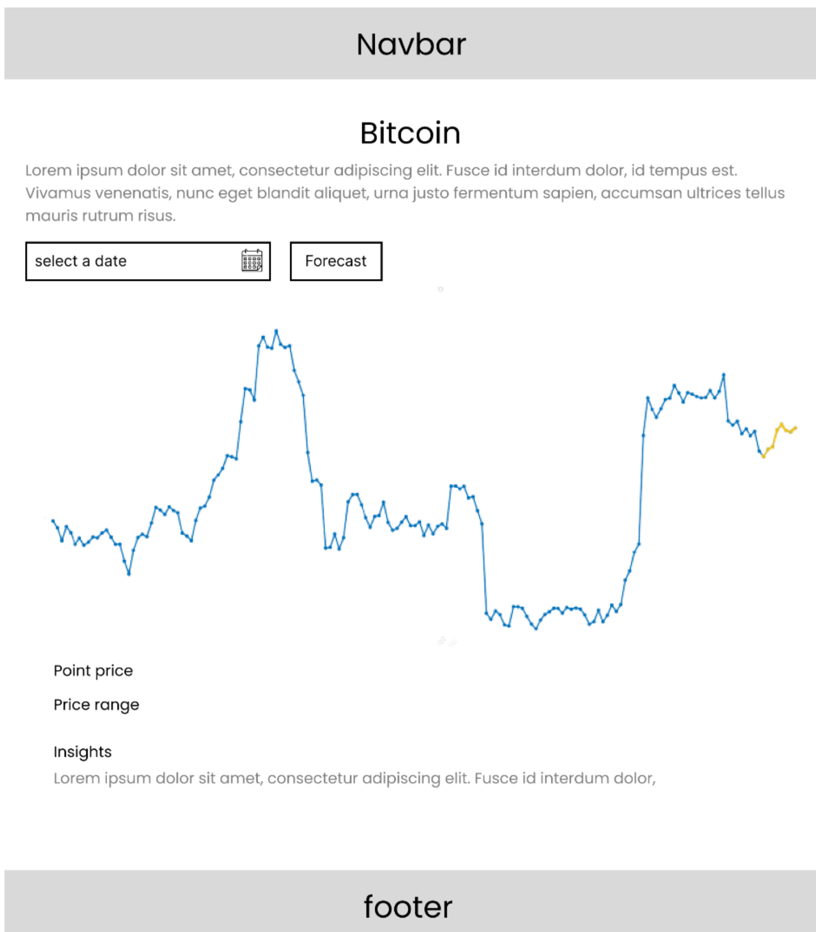


Figure 25: UI wireframes – Forecast (*Self-Composed*)

APPENDIX C – IMPLEMENTATION

C.1. Selection of programming language

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

Table 24: Selection of data science language (*Self-Composed*)

Data science To implement the core data science components two of the most popular languages that are used widely for data science were analyzed.			
Aspect	**Relevance**	**Python**	**R**
Availability of libraries.	A language that supports multiple libraries is paramount as the author would require multiple different techniques to gather the required 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 is paramount 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.			

C.2. Selection of DL framework

Table 25: Selection of DL framework (*Self-Composed*)

Framework	Description
TensorFlow	Used for production level applications, has detailed documentation, community support and handles large datasets. It also provides better visualization options which makes it easy to debug and monitor training, which is important as a novel algorithm is being built and no comparison is present.
PyTorch	Is more lightweight and developer-friendly, as it provides a more higher-level development. Therefore, 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 simpler. Additionally, building the algorithm requires more low-level details. (PyTorch vs. TensorFlow: 2022 Deep Learning Comparison | Built In, 2022)	

C.3. Selection of UI framework

Table 26: Selection of UI framework (*Self-Composed*)

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

React	Customizable and promotes code reusability via functions as components. Carries a large community and is open-source while being SEO friendly. Additionally, the React developer tools is a very handy tool.
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. (Angular vs React | Angular vs Vue | React vs Vue – Know the Difference, 2021)	

C.4. Selection of API framework

Table 27: Selection of web framework (*Self-Composed*)

Framework	**Description**
Flask	A very lightweight framework that provides only the simplest of functionalities. However, is the preferred choice for ML API development due to it being lightweight.
Django	Suitable for more larger scaled applications that provides a vast range of functionalities and 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. (Flask Vs Django: Which Python Framework to Choose?, 2021)	

C.5. Fetch data

Fetch historical prices

```python
import requests
import pandas as pd

# API reference: http://api.scraperlink.com/investpy/
BASE_URL = 'http://api.scraperlink.com/investpy/?email=your@email.com&type=historical_data&product=cryptos&symbol=BTC'
FILE_PATH = '../../ml/data/BTC_Prices.csv'

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

    df.to_csv(FILE_PATH)
```

Figure 26: 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 & block reward size

```python
URL = 'https://bitinfocharts.com/comparison/bitcoin-tweets.html#alltime'
FILE_PATH = '../../ml/data/Tweets/BTC_Tweet_Volume.csv'

response = requests.get(URL)
soup = BeautifulSoup(response.text, 'html.parser')
scripts = soup.find_all('script')

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:
            tweets.append(each)

    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
    '''

    df.to_csv(FILE_PATH)
```

Figure 27: Fetch Twitter volume (*Self-Composed*)

```python
URL = 'https://bitinfocharts.com/comparison/size-btc.html#alltime'
FILE_PATH = '../../ml/data/BTC_Block_Reward.csv'

response = requests.get(URL)
soup = BeautifulSoup(response.text, 'html.parser')
scripts = soup.find_all('script')

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 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:
            sizes.append(each)

    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
    '''

    df.to_csv(FILE_PATH)
```

Figure 28: Fetch block reward size (*Self-Composed*)

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

Fetch tweet data

```python
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.username, tweet.content])
        except Exception as e:
            print(f'Error: {e}')

    return tweets_list
```

Figure 29: Scrape tweets (*Self-Composed*)

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

    tweets_list = scrape_tweets(dates)
    df_days = process_tweets(tweets_list)

    for df in df_days:
        df.dropna(subset=['user', 'timestamp', 'text'], inplace=True)

        L = []
        for row in df['text']:
            # Use WTL 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[df['lang'] == Language.ENGLISH]
        df_filtered.drop(['lang'], axis=1, inplace=True)
        filename = str(df.iloc[0]['date'])
        df_filtered.to_csv(f'{FOLDER_PATH}/{filename}.csv')

    return df_days
```

Figure 30: Clean tweets (*Self-Composed*)

As this research is currently limited to only English, the tweets are filtered and non-English tweets are removed.

Fetch Google Trends

```python
def get_new_trends_data():
    '''
    Fetch latest Trends data
    '''

    pytrends = TrendReq()
    kw_list = ['bitcoin']
    pytrends.build_payload(
        kw_list,
        cat=0,
        timeframe='now 7-d',
        geo='',
        gprop=''
    )

    curr_data = pytrends.interest_over_time()
    return curr_data

def format_new_data(df):
    '''
    Converts the obtained data into the format of the available data
    '''

    df.rename(columns={ 'bitcoin': 'bitcoin_unscaled' }, inplace=True)

    # Create stringified dates
    df.index = [str(i) for i in pd.to_datetime(df.index).date]
    df.index.rename('date', inplace=True)

    # As the data obtained is for every hour, get an average of it all for each day
    grouped_df = df.groupby(level=0)
    avg_df = grouped_df.agg({ 'bitcoin_unscaled': 'mean' })
    return avg_df
```

Figure 31: Fetch Google Trends (*Self-Composed*)

Fetching Google Trends data was also a relatively straightforward procedure, as Python exposes a library specifically for this purpose. However, rate-limitations had to be overcome by running the script multiple times for specific data ranges at a time rather than the entire history.

C.6. Preprocessing

Tweet sentiment analysis

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

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

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

def get_sentiments(dfs):
    '''
    dfs → comes from the tweet_scraper script (only the new fetched dates)
    Updates all dfs with respective sentiment columns
    '''

    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(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)
            compound_scores.append(compound)

        df['negative_score'] = negative_scores
        df['positive_score'] = positive_scores
        df['neutral_score'] = neutral_scores
        df['compound_score'] = compound_scores
        export_data(df, df_filename)
```

Figure 32: 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, which can then be condensed down to create an average score for a single day.

Tweet dataset condensation

```python
def read_csvs():
    '''
    Load all tweet csvs in folder into a list of dfs which can then be condensed
    '''

    dfs = [pd.read_csv(f'{FOLDER_PATH}/{i}', engine='python') for i in ALL_FILES]
    return dfs

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

    condensed_df = None

    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)

    return condensed_df

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

    df.to_csv(OUTPUT_PATH)
```

Figure 33: Combine and condense tweets (*Self-Composed*)

As the other data being used directly create a single CSV file with a row for each date, the condensation process is not required. 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():
    '''
    Create and clean the final combined dataset
    '''

    exogenous_features = get_exogenous_datasets()
    filtered_prices = get_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)
    return combined_df

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

    df.to_csv(OUTPUT_PATH)

def create_final_dataset():
    '''
    Main runner
    '''

    print('\nRunning final dataset creation...', end='\n')
    combined_df = create_combined_dataset()
    export_data(combined_df)
    print('\nFinal dataset created', end='\n')
```

Figure 34: Combine all datasets (*Self-Composed*)

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

APPENDIX D – CONCLUSION

D.1. Project schedule

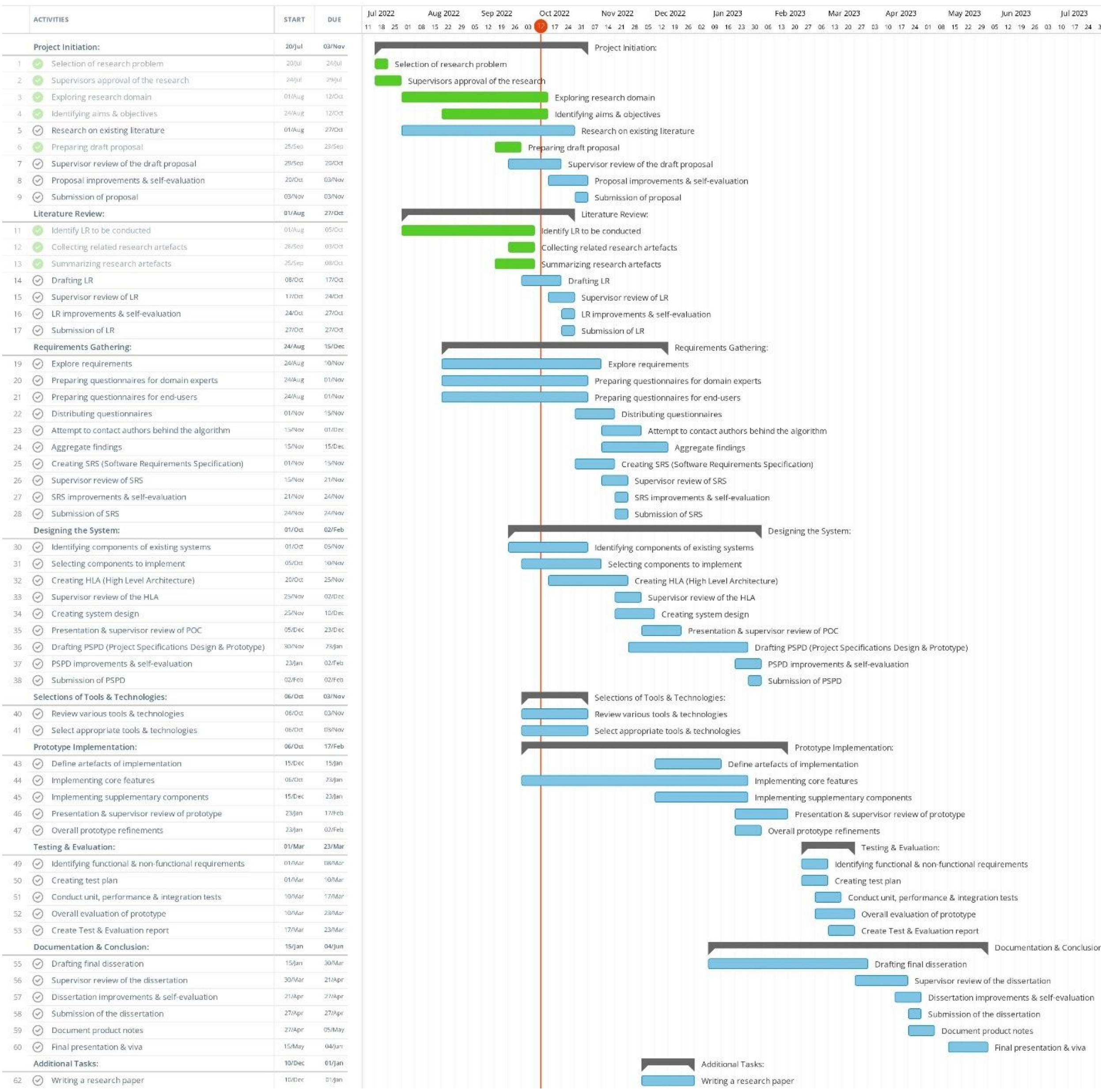


Figure 35: Initial Gantt chart (*Self-Composed*)

A clearer version can be found [Here]()

D.2. Project progress

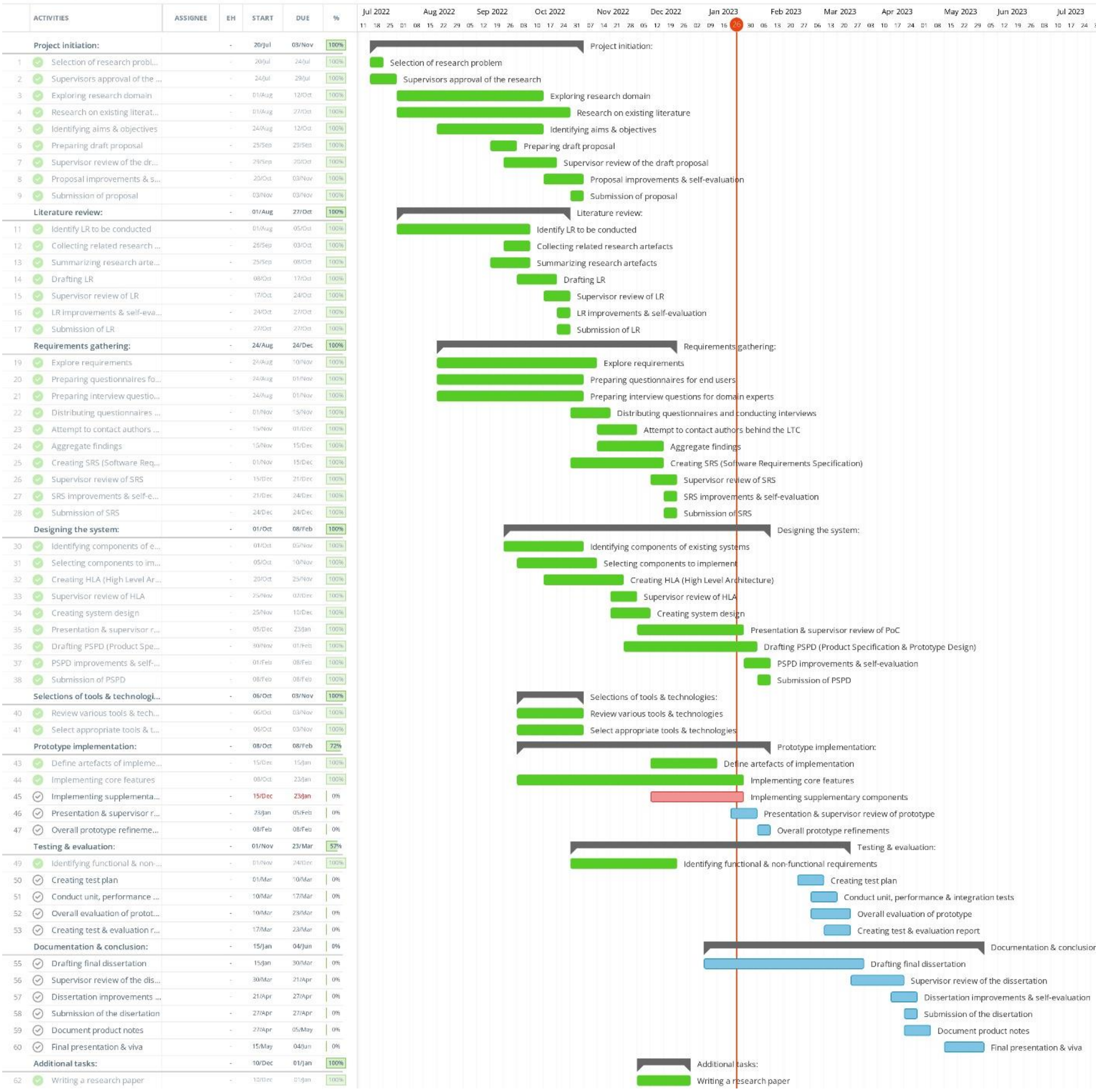


Figure 36: Current Gantt chart (*Self-Composed*)

A clearer version can be found Here

D.3. Evaluation metrics

Table 28: 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.	$\frac{100}{n} \sum_{i=1}^{n} \lvert \frac{y_i - \hat{y_i}}{y_i} \rvert$