Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK



*The University of Westminster, Coat of Arms*

**GenSum**

A Product Specification & Prototype Design by

Mr. Nazhim Kalam

W1761265 | 2019281

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**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst. Integer euismod, nisi a vulputate malesuada, velit velit accumsan nibh, a blandit justo lacus vel enim. Sed ullamcorper commodo eros, non scelerisque justo dictum id. Aenean faucibus congue elit, id commodo ligula convallis a. Nullam facilisis libero id libero semper, eu pellentesque ante dapibus. Proin bibendum dictum mauris, a commodo quam dictum id. Nam placerat, nisl non congue viverra, elit quam congue dui, non congue sem nisi id metus.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst. Integer euismod, nisi a vulputate malesuada, velit velit accumsan nibh, a blandit justo lacus vel enim. Sed ullamcorper commodo eros, non scelerisque justo dictum id. Aenean faucibus congue elit, id commodo ligula convallis a. Nullam facilisis libero id libero semper, eu pellentesque ante dapibus. Proin bibendum dictum mauris, a commodo quam dictum id. Nam placerat, nisl non congue viverra, elit quam congue dui, non congue sem nisi id metus.

**Keywords:** Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin euismod metus eu est tincidunt, id venenatis mi tempor. In hac habitasse platea dictumst

**Subject Descriptors:**

* Theory of computation 🡪 Theory and algorithms for application domains 🡪 Machine learning theory 🡪 Inductive inference.
* Theory of computation 🡪 Design and analysis of algorithms 🡪 Approximation algorithms analysis 🡪 Stochastic approximation.
* Mathematics of computing 🡪 Probability and statistics 🡪 Stochastic processes.
* Information systems 🡪 Information systems applications 🡪 Data mining.
* Computation methodologies 🡪 Machine learning 🡪 Machine learning algorithms 🡪 Ensemble methods

Contents

[ABSTRACT i](#_Toc125663188)

[List of Tables vi](#_Toc125663189)

[List of Figures vii](#_Toc125663190)

[CHAPTER 01. INTRODUCTION 1](#_Toc125663191)

[1.1. Chapter overview 1](#_Toc125663192)

[1.2. Problem domain 1](#_Toc125663193)

[1.2.1 Time series forecasting 1](#_Toc125663194)

[1.2.2 Liquid Time-Constant (LTC) networks 1](#_Toc125663195)

[1.2.3 Cryptocurrencies 2](#_Toc125663196)

[1.3. Problem definition 3](#_Toc125663197)

[1.3.1 Problem statement 3](#_Toc125663198)

[1.4. Research questions 3](#_Toc125663199)

[1.5. Research aim & objectives 3](#_Toc125663200)

[1.5.1 Research aim 3](#_Toc125663201)

[1.5.2 Research objectives 4](#_Toc125663202)

[1.6. Novelty of the research 6](#_Toc125663203)

[1.6.1 Problem novelty 6](#_Toc125663204)

[1.6.2 Solution novelty 6](#_Toc125663205)

[1.7. Research gap 6](#_Toc125663206)

[1.8. Contribution to the body of knowledge 7](#_Toc125663207)

[1.8.1 Contribution to the research domain 7](#_Toc125663208)

[1.8.2 Contribution to the problem domain 7](#_Toc125663209)

[1.9. Research challenge 7](#_Toc125663210)

[1.10. Chapter summary 8](#_Toc125663211)

[CHAPTER 02. SOFTWARE REQUIREMENTS SPECIFICATION 9](#_Toc125663212)

[2.1. Chapter overview 9](#_Toc125663213)

[2.2. Rich picture 9](#_Toc125663214)

[2.3. Stakeholder analysis 10](#_Toc125663215)

[2.3.1 Stakeholder onion model 10](#_Toc125663216)

[2.3.2 Stakeholder viewpoints 11](#_Toc125663217)

[2.4. Selection of requirement elicitation methodologies 11](#_Toc125663218)

[2.5. Discussion of findings 13](#_Toc125663219)

[2.5.1 Literature review 13](#_Toc125663220)

[2.5.2 Observations 13](#_Toc125663221)

[2.5.3 Survey 14](#_Toc125663222)

[2.5.4 Interviews 14](#_Toc125663223)

[2.5.5 Prototyping 14](#_Toc125663224)

[2.5.6 Summary of findings 15](#_Toc125663225)

[2.6. Context diagram 16](#_Toc125663226)

[2.7. Use case diagram 17](#_Toc125663227)

[2.8. Use case descriptions 17](#_Toc125663228)

[2.9. Requirements 19](#_Toc125663229)

[2.9.1 Functional requirements 19](#_Toc125663230)

[2.9.2 Non-functional requirements 20](#_Toc125663231)

[2.10. Chapter summary 21](#_Toc125663232)

[CHAPTER 03. DESIGN 22](#_Toc125663233)

[3.1. Chapter overview 22](#_Toc125663234)

[3.2. Design goals 22](#_Toc125663235)

[3.3. High level design 23](#_Toc125663236)

[3.3.1. Architecture diagram 23](#_Toc125663237)

[3.3.2. Discussion of tiers of the architecture 24](#_Toc125663238)

[3.4. System design 25](#_Toc125663239)

[3.4.1. Choice of design paradigm 25](#_Toc125663240)

[3.5. Design diagrams 25](#_Toc125663241)

[3.5.1. Data flow diagrams 25](#_Toc125663242)

[3.5.1.1. Level 01 data flow diagram 25](#_Toc125663243)

[3.5.1.2. Level 02 data flow diagram 26](#_Toc125663244)

[3.5.2. Algorithmic design 27](#_Toc125663245)

[3.5.2.1. Existing LTC architecture 27](#_Toc125663246)

[3.5.2.2. Algorithm proposed by the author 27](#_Toc125663247)

[3.5.3. Algorithmic analysis 31](#_Toc125663248)

[3.5.4. System process activity diagram 31](#_Toc125663249)

[3.5.5. UI design 32](#_Toc125663250)

[3.6. Chapter summary 32](#_Toc125663251)

[CHAPTER 04. INITIAL IMPLEMENTATION 33](#_Toc125663252)

[4.1. Chapter overview 33](#_Toc125663253)

[4.2. Technology selection 33](#_Toc125663254)

[4,2.1. Technology stack 33](#_Toc125663255)

[4.2.2. Selection of data 34](#_Toc125663256)

[4.2.3. Selection of programming language 34](#_Toc125663257)

[4.2.4. Selection of development framework 36](#_Toc125663258)

[4.2.4.1 DL framework 36](#_Toc125663259)

[4.2.4.2. UI framework 36](#_Toc125663260)

[4.2.4.3. API web framework 37](#_Toc125663261)

[4.2.5. Other libraries & tools 38](#_Toc125663262)

[4.2.6. Integrated Development Environment (IDE) 39](#_Toc125663263)

[4.2.7. Summary of chosen tools & technologies 39](#_Toc125663264)

[4.3. Implementation of core functionalities 39](#_Toc125663265)

[4.3.1. Algorithm implementation 39](#_Toc125663266)

[4.3.2. Data fetchers 43](#_Toc125663267)

[4.3.3. Preprocessing 43](#_Toc125663268)

[4.4. Chapter summary 43](#_Toc125663269)

[CHAPTER 05. CONCLUSION 44](#_Toc125663270)

[5.1. Chapter overview 44](#_Toc125663271)

[5.2. Deviations 44](#_Toc125663272)

[5.2.1. Scope related deviations 44](#_Toc125663273)

[5.2.2. Schedule related deviations 44](#_Toc125663274)

[5.3. Initial test results 44](#_Toc125663275)

[5.4. Required improvements 44](#_Toc125663276)

[5.5. Demo of the prototype 45](#_Toc125663277)

[5.6. Chapter summary 45](#_Toc125663278)

[REFERENCES I](#_Toc125663279)

[APPENDIX A – INTRODUCTION IV](#_Toc125663280)

[A.1. Research questions IV](#_Toc125663281)

[APPENDIX B – SRS IV](#_Toc125663282)

[B.1. Requirement elicitation methodologies IV](#_Toc125663283)

[B.2. Survey analysis V](#_Toc125663284)

[B.3. Interview analysis XI](#_Toc125663285)

[B.4. Use case descriptions XIII](#_Toc125663286)

[B.5. Functional requirements XIV](#_Toc125663287)

[APPENDIX C – DESIGN XV](#_Toc125663288)

[C.1. Algorithm intuition XV](#_Toc125663289)

[C.2. UI wireframes XVI](#_Toc125663290)

[APPENDIX D – IMPLEMENTATION XVIII](#_Toc125663291)

[D.1. Fetch data XVIII](#_Toc125663292)

[D.2. Preprocessing XXII](#_Toc125663293)

[APPENDIX E – CONCLUSION XXV](#_Toc125663294)

[E.1. Project scope XXV](#_Toc125663295)

[E.2. Project schedule XXVI](#_Toc125663296)

[E.3. Project progress XXVII](#_Toc125663297)

# List of Tables

[Table 1: Research Objectives (*Self-Composed*) 4](#_Toc125649728)

[Table 2: Stakeholder viewpoints (*self-Composed*) 11](#_Toc125649729)

[Table 3: Requirement elicitation methodologies (*Self-Composed*) 12](#_Toc125649730)

[Table 4: Observations findings (*Self-Composed*) 13](#_Toc125649731)

[Table 5: Prototyping findings (*Self-Composed*) 14](#_Toc125649732)

[Table 6: Use case description UC:03; UC:04 (*Self-Composed*) 17](#_Toc125649733)

[Table 7: Use case description UC:05; UC:06 (*Self-Composed*) 18](#_Toc125649734)

[Table 8: Functional requirements (*Self-Composed*) 19](#_Toc125649735)

[Table 9: Non-functional requirements (*Self-Composed*) 21](#_Toc125649736)

[Table 10: Design goals of the proposed system (*Self-Composed*) 22](#_Toc125649737)

[Table 11: Complexities of BPTT and adjoint sensitivity 31](#_Toc125649738)

[Table 12: Dataset sources (*Self-Composed*) 34](#_Toc125649739)

[Table 13: Selection of data science language (*Self-Composed*) 35](#_Toc125649740)

[Table 14: Selection of DL framework (*Self-Composed*) 36](#_Toc125649741)

[Table 15: Selection of UI framework (*Self-Composed*) 37](#_Toc125649742)

[Table 16: Selection of web framework (*Self-Composed*) 37](#_Toc125649743)

[Table 17: Chosen libraries (*Self-Composed*) 38](#_Toc125649744)

[Table 18: Chosen IDEs (*Self-Composed*) 39](#_Toc125649745)

[Table 19: Chosen tools & technologies (*Self-Composed*) 39](#_Toc125649746)

[Table 20: Stakeholder groups (*Self-Composed*) IV](#_Toc125649747)

[Table 21: Survey analysis (*Self-Composed*) V](#_Toc125649748)

[Table 22: Survey thematic analysis codes, themes & conclusions (*Self-Composed*) IX](#_Toc125649749)

[Table 23: Interview thematic analysis codes, themes & conclusions (*Self-Composed*) XI](#_Toc125649750)

[Table 24: Interview participant details (*Self-Composed*) XIII](#_Toc125649751)

[Table 25: Use case description UC:07 (*Self-Composed*) XIII](#_Toc125649752)

[Table 26: ‘MoSCoW’ technique of requirement prioritization (*Self-Composed*) XIV](#_Toc125649753)

# List of Figures

[Figure 1: Rich picture diagram (*Self-Composed*) 9](#_Toc125649754)

[Figure 2: Stakeholder onion model (*self-Composed*) 10](#_Toc125649755)

[Figure 3: Context diagram (*Self-Composed*) 16](#_Toc125649756)

[Figure 4: Use case diagram (*Self-Composed*) 17](#_Toc125649757)

[Figure 5: Three-tiered architecture (*Self-Composed*) 23](#_Toc125649758)

[Figure 6: Data flow diagram - level 01 (*Self-Composed*) 26](#_Toc125649759)

[Figure 7: Data flow diagram - level 02 (*Self-Composed*) 26](#_Toc125649760)

[Figure 8: System process activity diagram (*Self-Composed*) 32](#_Toc125649761)

[Figure 9: Tech stack (*Self-Composed*) 33](#_Toc125649762)

[Figure 10: Initialize algorithm (*Self-Composed*) 40](#_Toc125649763)

[Figure 11: Build algorithm (*Self-Composed*) 40](#_Toc125649764)

[Figure 12: Algorithm – sensory, stochastic and leakage variables (*Self-Composed*) 41](#_Toc125649765)

[Figure 13: Algorithm – forward propagation (*Self-Composed*) 42](#_Toc125649766)

[Figure 14: Algorithm – define weights and biases (*Self-Composed*) 42](#_Toc125649767)

[Figure 15: Algorithm – Euler-Maruyama SDE solver (*Self-Composed*) 43](#_Toc125649768)

[Figure 16: Algorithm intuition (*Self-Composed*) XV](#_Toc125649769)

[Figure 17: UI wireframes – Home (*Self-Composed*) XVI](#_Toc125649770)

[Figure 18: UI wireframes – News (*Self-Composed*) XVI](#_Toc125649771)

[Figure 19: UI wireframes – Cryptocurrencies (*Self-Composed*) XVI](#_Toc125649772)

[Figure 20: UI wireframes – Cryptocurrency (*Self-Composed*) XVI](#_Toc125649773)

[Figure 21: UI wireframes – Admin login (*Self-Composed*) XVII](#_Toc125649774)

[Figure 22: UI wireframes – Admin model configuration (*Self-Composed*) XVII](#_Toc125649775)

[Figure 23: UI wireframes – Forecast (*Self-Composed*) XVII](#_Toc125649776)

[Figure 24: Fetch historical prices (*Self-Composed*) XVIII](#_Toc125649777)

[Figure 25: Fetch Twitter volume (*Self-Composed*) XIX](#_Toc125649778)

[Figure 26: Fetch block reward size (*Self-Composed*) XIX](#_Toc125649779)

[Figure 27: Scrape tweets (*Self-Composed*) XIX](#_Toc125649780)

[Figure 28: Clean tweets (*Self-Composed*) XX](#_Toc125649781)

[Figure 29: Fetch Google Trends (*Self-Composed*) XXI](#_Toc125649782)

[Figure 30: Analyze sentiments (*Self-Composed*) XXII](#_Toc125649783)

[Figure 31: Combine and condense tweets (*Self-Composed*) XXIII](#_Toc125649784)

[Figure 32: Combine all datasets (*Self-Composed*) XXIV](#_Toc125649785)

[Figure 33: Initial Gantt chart (*Self-Composed*) XXVI](#_Toc125649786)

[Figure 34: Current Gantt chart (*Self-Composed*) XXVII](#_Toc125649787)

**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 is attempting to enhance the performance of abstractive text summarization for movie reviews while creating a generalized solution that can be applied to other domains by optimizing a set of top-tier pretrained transformer architectures through hyperparameter optimization. The research problem, gap, challenge, and approach will be addressed, along with a review of prior research and a presentation of the expected project schedule in the work plan.

# **1.2. Problem domain**

## **1.2.1 Movie User Reviews**

As the popularity of Web 2.0, where user engagement is emphasized, increases, an increasing number of websites, such as Amazon and IMDb (a movie review website), are allowing users to post reviews on topics they are interested in (Khan, Gul, Zareei, et al., 2020).

Movie reviews found online have become a crucial source of information for users, as the amount of data on the web continues to grow (M and Mehla, 2019). Despite this, the large number of movie reviews posted online daily makes it hard for users to manually condense the information and determine if a film is of interest to them. Summarizing and extracting meaningful information from movie reviews is a challenging problem in natural language processing (Khan, Gul, Uddin, et al., 2020).

Text summarization helps users and business decision-makers by consolidating and examining a large number of online reviews (Alsaqer and Sasi, 2017). These days, before choosing to watch a film on platforms like Netflix or Amazon Prime, most people research its reviews. However, conflicting reviews can be both positive and negative, leading to a problem where users need a lot of time to decide. Summarizing the reviews makes it easier and faster for users to decide, and also allows streaming services like Netflix to quickly understand their users' viewing habits and preferences (Dashtipour et al., 2021).

## **1.2.2 Text Summarization**

In today's world, a vast amount of text material is accessible, including news articles and reviews. Text summarization enables us to quickly identify the main points of the full text by reducing its size (Mahajan et al., 2021).

The two common methods of text summarization are extractive summarization and abstractive summarization. In extractive summarization, the most crucial sentences from the context or article are selected without any modification. On the other hand, abstractive summarization generates new sentences to create the summary and is considered superior as it generates new phrases within context rather than just selecting existing sentences without alteration (Etemad, Abidi, and Chhabra, 2021).

## **1.2.3 Transformers**

Transformers in NLP is a new architecture designed to efficiently solve sequence-to-sequence tasks while managing long-range dependencies. It has outperformed other neural models such as CNN (Convolutional Neural Nets) and RNN (Recurrent Neural Nets), becoming the dominant architecture for natural language processing (Wolf et al., 2020).

Transformers utilize a self-attention mechanism that focuses on specific parts of the input sentence, followed by an encoder-decoder architecture (Etemad, Abidi, and Chhabra, 2021).

# **1.3. Problem definition**

Currently, there is a lack of research in movie review summarization that utilizes the latest deep learning approaches such as Transformers. While traditional machine learning and deep learning algorithms like Naive Bayes and RNN have been used in the past, the use of advanced deep learning techniques has the potential to improve the quality and accuracy of text summarization in this domain.

Compared to machine learning algorithms, deep learning models take more time to train, but they offer improved accuracy as they are capable of automatically extracting features and performing classification at the same time. On the other hand, machine learning algorithms require manual feature selection before training. Therefore, the implementation of deep learning techniques can enhance the quality of text summarization and facilitate better decision making for the user (Etemad, Abidi, and Chhabra, 2021).

## **1.3.1 Problem statement**

No previous research has explored using advanced deep learning techniques like Transformers to generate abstractive summaries from movie reviews, which could enhance the quality of text summarization. This solution aims to be generalized and applicable to any domain. (Khan, Gul, Zareei, et al., 2020).

# **1.4. Research questions**

The research questions proposed are available in [**APPENDIX A.1**](#_A.1._Research_questions).

# **1.5. Research aim & objectives**

## **1.5.1 Research aim**

***The aim of this research is to design, develop and evaluate an optimal generalized transformer architecture from a range of popularly used architectures by fine-tuning via hyperparameter optimization, therefore obtaining the recommended architecture's optimum performance***

This research project aims to create a functional system for abstractive text summarization based on user input reviews from various domains such as movies, hotels, ecommerce, etc. The focus will be on improving the quality of the generated text summary and optimizing performance. To achieve the desired outcome, the study will cover aspects like data preparation, analysis, hyperparameter tuning, and model evaluation.

This project aims to verify or disprove the chosen hypothesis by gathering relevant information, developing components, and assessing performance. The system will be accessible both through a hosted server and a local browser for private or public use. The data science models and their source code will be made publicly available for further research and usage in a repository. The findings from the literature review will also be documented in a review paper.

## **1.5.2 Research objectives**

The research must meet its objectives to be considered successful.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Objective | Description | LO | RQ |
| Literature Review | Complete a thorough critical review of earlier related work.  **RO1:** Make a preliminary investigation on existing abstractive text summarization using deep learning approaches.  **RO2:** Make a preliminary investigation on why transformers architecture was the chosen deep learning choice for this research.  **RO3:** Analyze the top tier transformer architectures widely used.  **RO4:** Analyzing how the models can be fine-tuned via hyperparameter optimization.  **RO5:** Analyzing the different approaches used for model evaluation.  **RO6:** Analyze how the model can be generalized for every other domain. | LO1, LO4,  LO8 | [**R**Q1](#researchq1),[**R**Q2](#researchq1),[**R**Q3](#researchq1),[**R**Q4](#researchq1) |
| Methodology Selection and SLEP Framework | This defines the outline structure for the requirement analysis and the design process followed by the social legal ethical and professional issues.  **RO1**: Analyzing the Research Methodology approaches.  **RO2**: Analyzing the Development Methodology approaches.  **RO3**: Analyzing the Project Management Methodology approaches.  **RO4**: Analyzing the Solution Methodology approaches.  **RO5**: Analyzing the Social, Legal Ethical and Professional Issues which could develop during the phase of the project. | LO2,  LO6 | [**R**Q4](#researchq1),[**R**Q2](#researchq1),[**R**Q1](#researchq1) |
| Requirement Elicitation | Defining the project's needs utilizing relevant approaches and tools in order to solve the projected research gaps and obstacles based on prior related research.  **RO1:** Gathering information related to the expected metadata required for the dataset to contain for the model training.  **RO2:** Gathering the requirements of transformer architectures for fine-tuning and understand the end to end user expectations.  **RO3:** Getting insights from domain experts to build a suitable system.  **RO4:** Gathering the requirements for handling generalization. | LO1, LO3, LO5 | [**R**Q4](#researchq1),[**R**Q2](#researchq1),[**R**Q1](#researchq1) |
| Design | Considering the following when developing the suggested system:  **RO1:** Design a component to preprocess the dataset for the respective model inputs.  **RO2:** Design a component to store the top tier transformer models with their respective metadata, to use throughout.  **RO3:** Design a hyperparameter tuning component that can improve accuracy of the transformer model.  **RO4**: Design high-level architecture for the system. | LO1, LO5 | [**R**Q2](#researchq2) |
| Implementation | Setting up a mechanism capable of addressing the gaps that were intended to be covered.  **RO1**: To develop data preprocessing component.  **RO2**: To develop a component that handles and stores the top tier transformer architectures for fine-tuning.  **RO3**: To develop the automated hyperparameter search component that handles all the top tier architectures assigned.  **RO4**: To develop a component for the model evaluations for the measured hyperparameters | LO1, LO5, LO7 | [**R**Q2](#researchq2),[**R**Q3](#researchq2) |
| Evaluation | Testing and evaluating the developed system (including the data science models with the suitable metrices)  **RO1**: Performing unit test, integration and performance testing along with a test plan created.  **RO2**: Evaluating all the transformer architectures used for fine-tune experimentations, using recommended scores such as (ROUGE, BERT SCORE). | LO1,  LO5 | [**R**Q3](#researchq3) |
| Documentation | Keeping track of and documenting the study project's ongoing progress and any challenges encountered. | LO6, LO8 | - |
| Publication | Ensure that the documentation, reports, and papers are well-structured and include a critical analysis of the research.  **RO1**: To publish a research paper on the related work done.  **RO2**: To publish the testing & evaluation results of the work done.  **RO3**: To publish the code implementation repository as public to be access by future research investigations, along with the models and datasets | LO4,  LO8 | - |

# **1.6. Novelty of the research**

## **1.6.1 Problem novelty**

The problem novelty of this research lies in the fact that there has been limited effort to enhance the performance of transformers in order to achieve improved results in textual summarization.

## **1.6.2 Solution novelty**

The solution novelty for this problem has several approaches few of which performing automated hyperparameter tuning, creating a retraining mechanism with newly exposed data and exploring any changes in the model architecture to enhance its performance further.

# **1.7. Research gap (using quilit bot from here for parapharsing)**

Based on previous work done (Khan, Gul, Zareei, et al., 2020) related to abstractive text summarization on movie reviews, the literature identify for the need of using advanced deep learning approaches to improve the performance of text summarization for this movie domain over traditional machine learning approach.

This project focuses on Empirical gap in the Movie Domain, as well as Theoretical and Performance gaps in the area of transformer optimization. Transformers plays a major role in the field of deep learning especially at problems related to Natural Language Processing, by performing hyperparameter optimization on several transformer architectures we can contribute to the enhanced quality of abstractive text summarization and create a generalized model which can be adapted with the respective domains usage and improve the performance.

# **1.8. Contribution to the body of knowledge**

Improving the performance of an existing solution is very common in the field of data science, as we can explore new algorithms or fine-tuning existing algorithms to get better results. The contributions for this project can be classified as theoretical contributions and domain contributions.

The following is a summarization of the authors contribution:

* ***Abstractive Text Summarization:*** *Automated Hyperparameter optimization + Model Retraining + Transformers + Deep Learning*
* ***Movie User Review & Generalization:*** *Research domain target is for Movie reviews, in addition the author makes the system generalized to adapt to any domain area*.

## **1.8.1 Contribution to the research domain**

There are various deep learning techniques that can be used to handle abstractive text summarization, however with respect to previous researches done, (Zhang, Xu and Wang, 2019) it is found that ***transformers*** outperform most of the other deep learning approaches as of today but there was no much research on optimizing them for a much better performance.

This research will be focused on creating a ***generalized******solution***by achieving the optimized transformer architecture from a couple of the top tier existing architectures, via fine-tuning and performing hyperparameter optimization along with handling abstractive text summarization (Liu and Wang, 2021), therefore we are able to maximize the performance of the recommended architecture. The author plans out to make use of generalization where any domain when used the model will be optimizing and adapting towards their respective domain.

## **1.8.2 Contribution to the problem domain**

Neural Networks makes up the backbone of deep learning algorithms which enables them to process complex unstructured data over normal means of machine learning algorithms (Mahajan et al., 2021). It is found that, the need for using advanced deep learning approaches has not been explored in the domain of movie review summarization.

Given that transformers perform well in this field, the proposed solution for this domain will be finding the recommended architecture along with hyper-parameter optimization, to reach its best performance. An additional contribution will be that, the proposed solution will be generalized to any other domain linked with the field of NLP text summarization.

# **1.9. Research challenge**

The main objective of this research is to achieve the generalized optimal transformer architecture for the field of NLP abstractive text summarization. Transformers were introduced in 2017 by a team at Google Brain and are the most used choice for NLP problems replacing RNN models, given that this architecture was introduced not much longer back brings to a point where there is a lack of research done in the area of transformer optimization for the purpose of abstractive text summarization. (Wolf et al., 2020).

Therefore, creating and finding the recommended transformer architecture along with the optimal parameters which also handles generalization becomes a challenge with very fewer resources to look up to.

Additionally, identifying suitable datasets for this domain (Movie Reviews Summarization) and Generalization is challenging and necessitates a substantial amount of effort in data preprocessing where it is important since we are dealing with NLP and performance optimization related domain.

# **1.10. Chapter summary**

In this chapter, the author gave an outline of the research project that was carried, the reasons why the research and problem were innovative, and the difficulties that could arise while trying to solve them. In addition, the essential objectives that must be pursued for the study to be considered effective were put out and linked to the degree's required learning outcomes.

# **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**

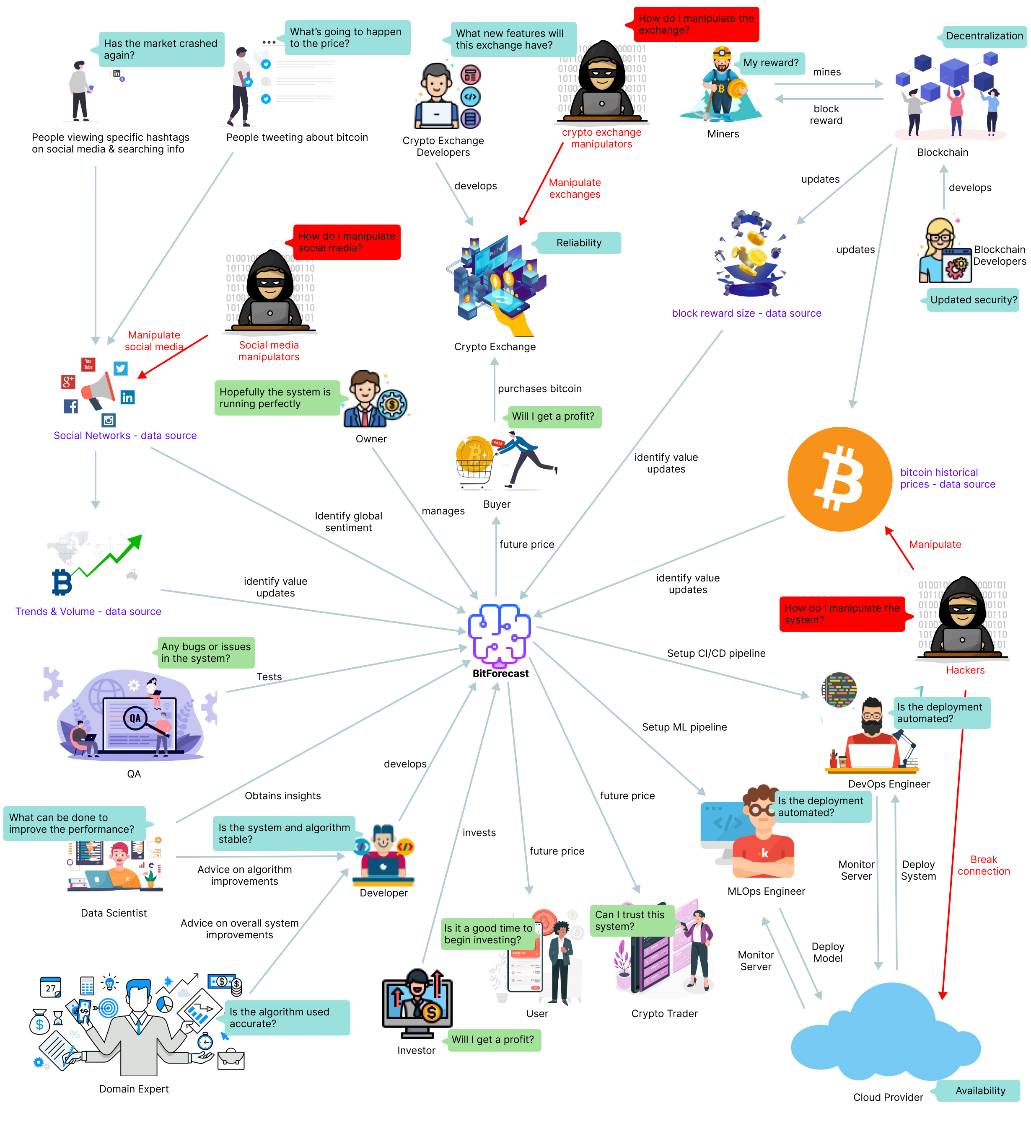


Figure 1: Rich picture diagram (*Self-Composed*)

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

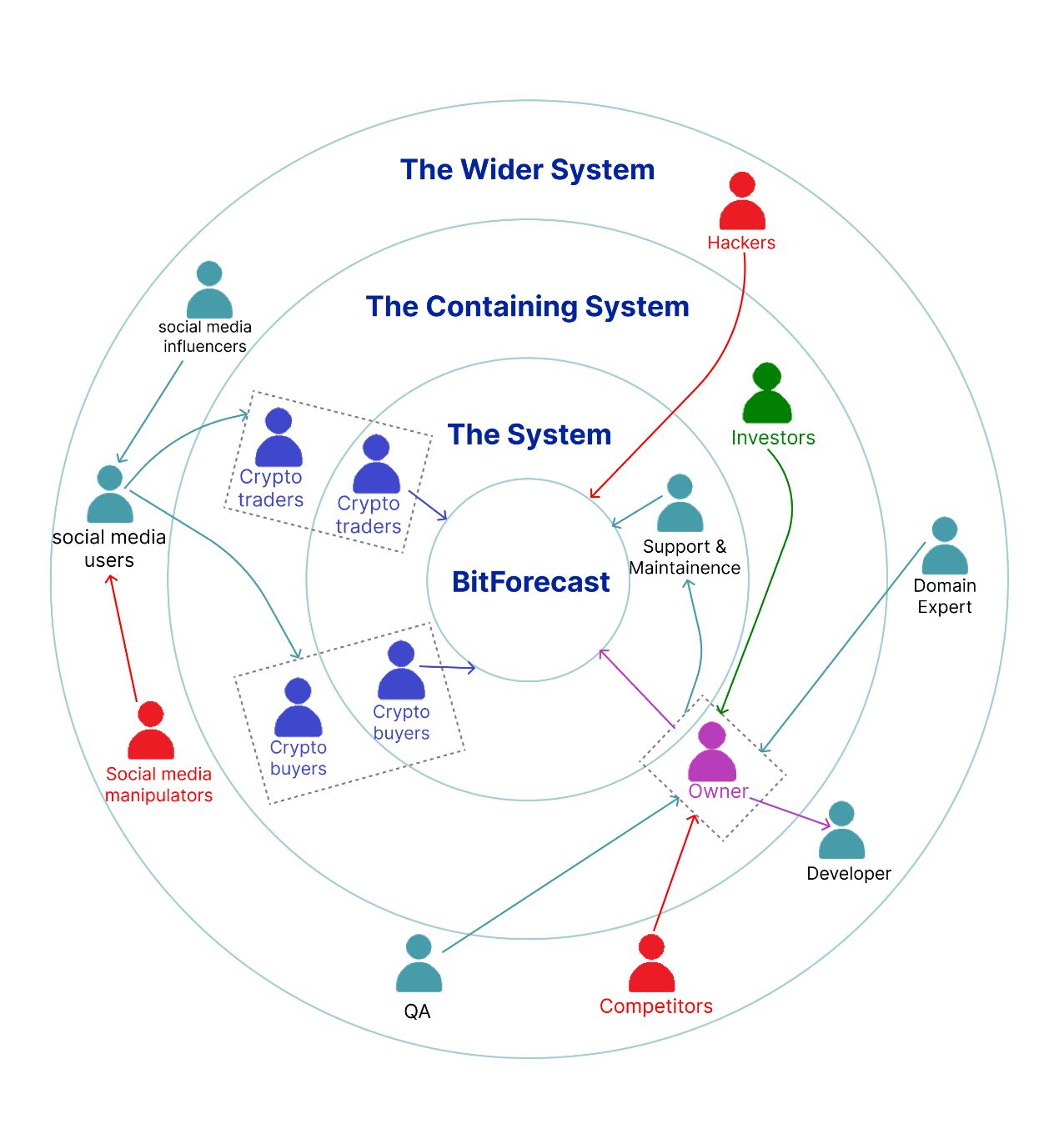


Figure 2: Stakeholder onion model (*self-Composed*)

## 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. |
| 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 | Functional beneficiary | 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 B.1**](#_B.1._Requirement_elicitation).

## 2.5.1 Literature review

|  |  |
| --- | --- |
| **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 4: Observations findings (*Self-Composed*)

|  |  |
| --- | --- |
| **Criteria** | **Discussion of findings** |
| To find approaches to creating a neural SDE to implement the core research component | 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. 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. The analysis is available in [**APPENDIX B.2**](#_B.2._Survey_analysis).

## 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 are presented in [**APPENDIX B.3**](#_B.3._Interview_analysis). The participants affiliations and their respective expertise area are also available.

## 2.5.5 Prototyping

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

|  |
| --- |
| **Criteria** |
| Prototyping was solely carried out 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.

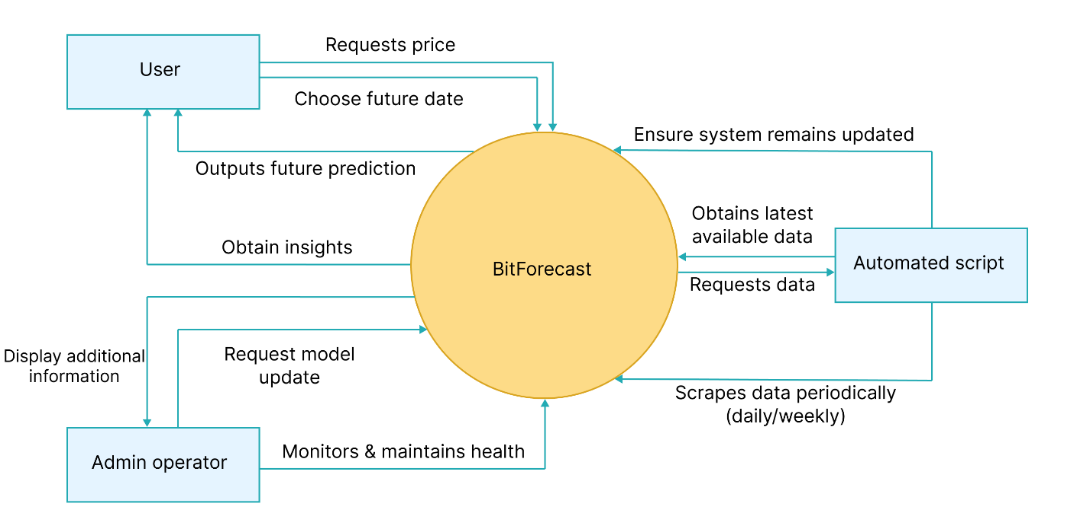


Figure 3: Context diagram (*Self-Composed*)

# **2.7. Use case diagram**

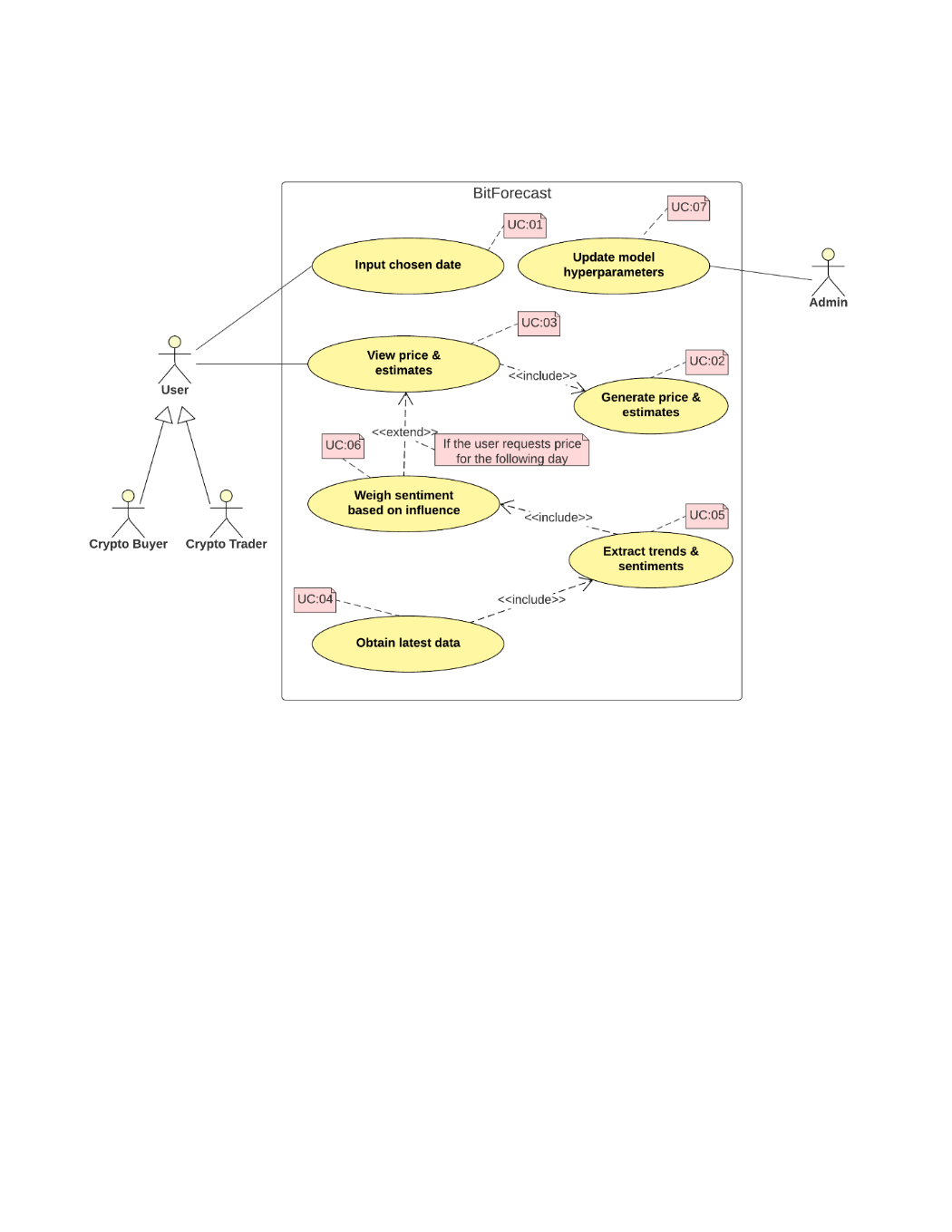


Figure 4: Use case diagram (*Self-Composed*)

# **2.8. Use case descriptions**

The core use case descriptions are presented below, any sub-descriptions are available in [**APPENDIX B.4**](#_B.4._Use_case).

Table 6: 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 | 1. User requests tomorrow’s price. 2. The system recognizes the need to utilize available exogenous features. 3. 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. 4. The system generates price and upper and lower estimations. 5. Display output to the user along with any insights. |
| Alternative flows | 1. The user requests the price for a date ahead of tomorrow. 2. The system recognizes the inability to utilize other features. 3. The system generates price and upper and lower estimations. 4. Display output to the user along with any insights. |
| Exceptional flows | 1. 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. |

Table 7: 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 | 1. A Cron job triggered fetches the latest historical prices, tweets, Twitter volume, trends, and block reward size data. 2. Twitter volume, Google trends, and block reward size are scaled and cleaned. 3. Tweets undergo sentiment analysis to determine current speculation. 4. The sentiment is further weighted based on the Tweeter’s importance (ex: Elon Musk) 5. Features are combined with historical closing prices to create an enriched dataset and retrain the model. |
| Alternative flows | None |
| Exceptional flows | 1. 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. |

# **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 B.5**](#_B.5._Functional_requirements).

Table 8: 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 | The script should weigh sentiment based on any influential personnel’s tweet. | S | UC:06 |
| FR9 | Users should be able to view a range of prices along with the single-point price. | S | UC:03 |
| FR10 | The system should generate higher and lower bound uncertainty estimations. | S | UC:02 |
| FR11 | The GUI should plot the forecast with the current prices in a single graph to show the growth/decline. | S | UC:03 |
| 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 9: 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 10: 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.

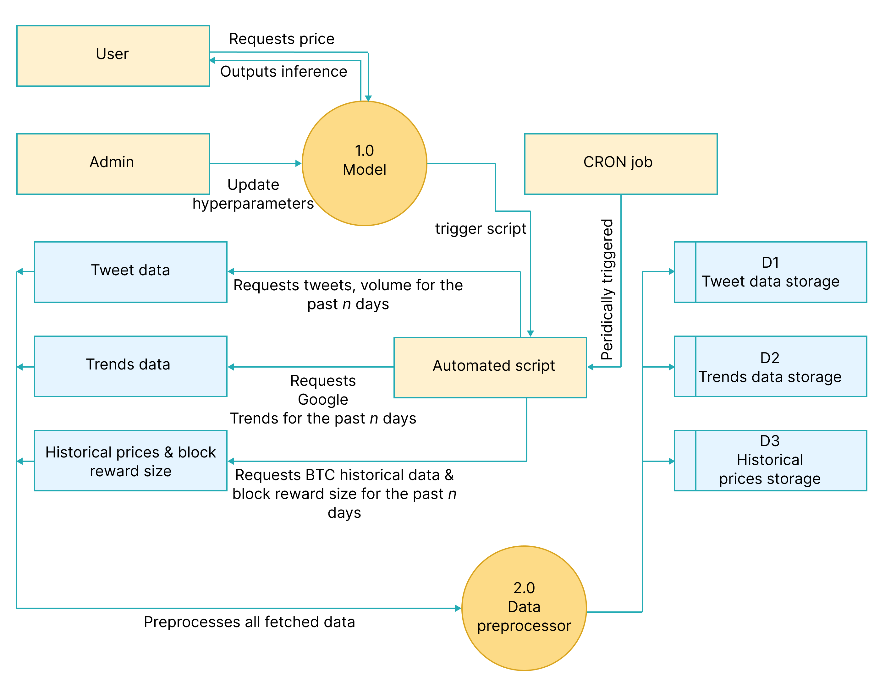


Figure 6: Data flow diagram - level 01 (*Self-Composed*)

### **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.

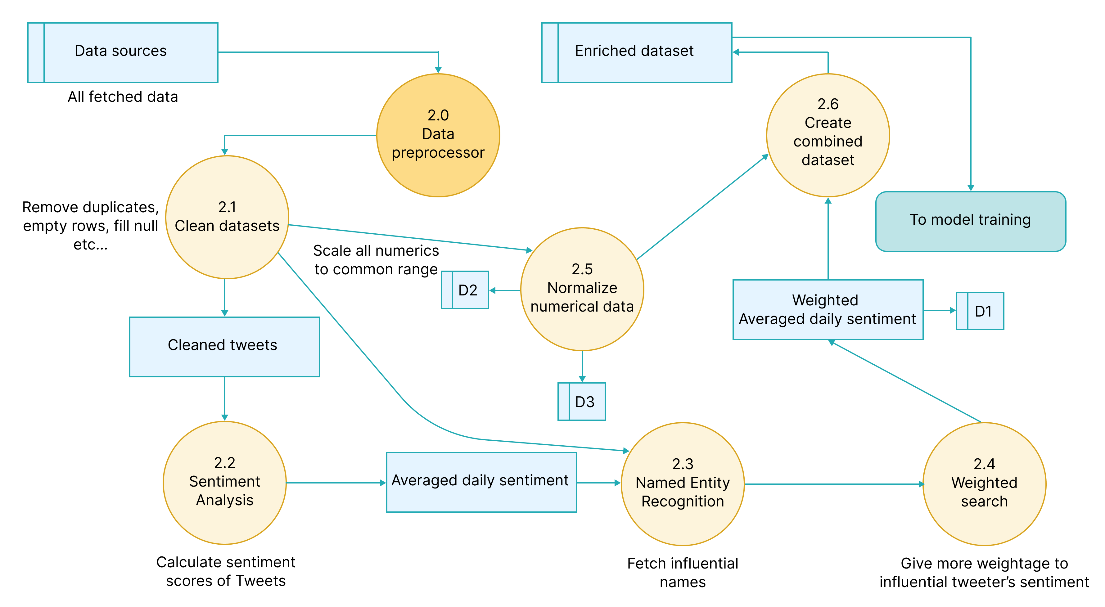


Figure 7: Data flow diagram - level 02 (*Self-Composed*)

## **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](#hasani2020ref)), which can be considered as their primary contribution to the body of knowledge. A simple illustration is available in [**APPENDIX C.1**](#_C.1._Algorithm_intuition) to gather intuition.

### **3.5.2.1. Existing LTC architecture**

|  |  |
| --- | --- |
|  | *Time-constant* |
|  | *Hidden state* |
|  | *Input* |
| *t* | *Time* |
|  | *Neural network* |
|  | *Parameters* |

The above formulation was proposed byHasani et al. ([2020](#hasani2020ref)), where a system of linear ODEs is used to declare the flow of the hidden state; the ODEs are of the following form.

*Where S(t) represents the following nonlinearity*

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.

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:

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:

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.

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

***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.

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:

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:

**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 11: 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.

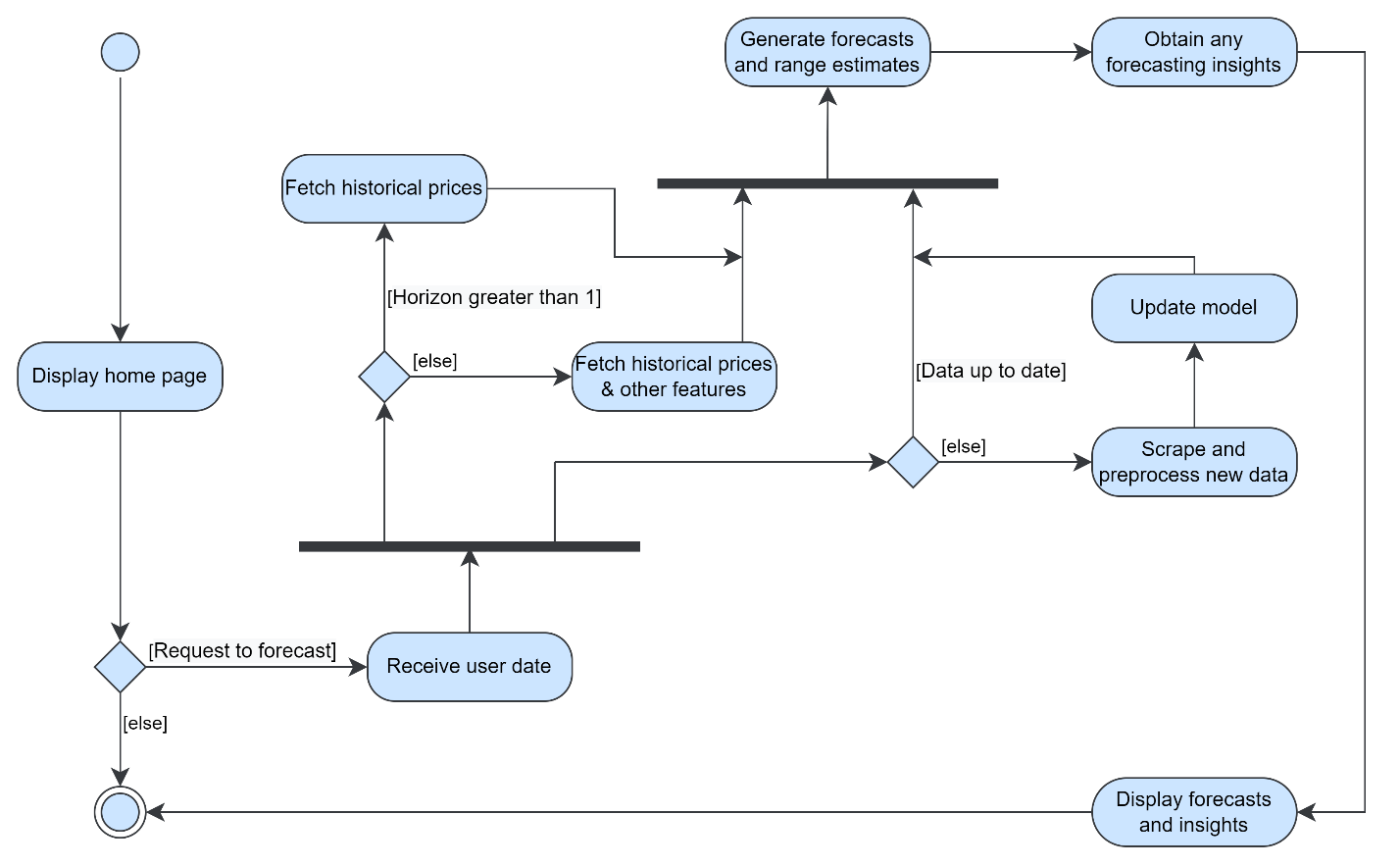


Figure 8: System process activity diagram (*Self-Composed*)

## **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 C.2**](#_C.2._UI_wireframes).

# **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 12: 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.

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 13: 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. | | | |

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.

Table 14: 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) | |

### **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.

Table 15: 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) | |

### **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.

Table 16: 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) | |

## **4.2.5. Other libraries & tools**

Table 17: 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 18: 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 19: 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.

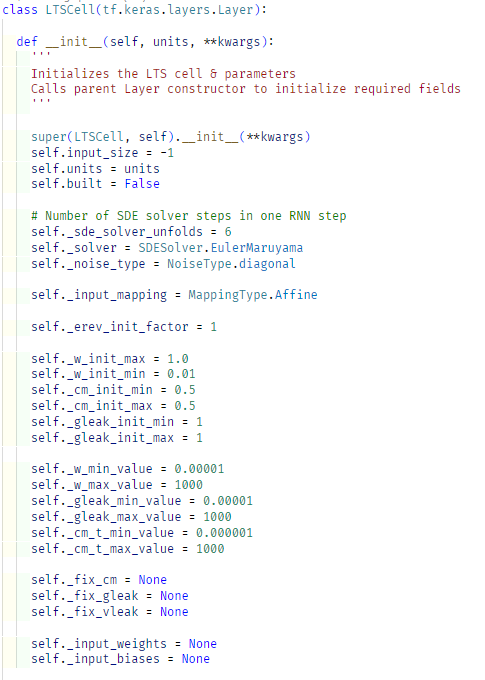


Figure 10: Initialize algorithm (*Self-Composed*)

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).

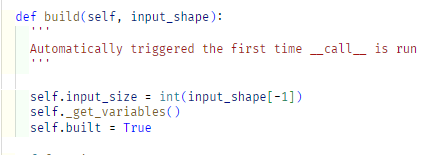
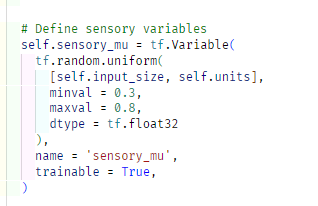
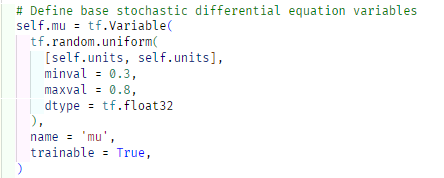


Figure 11: Build algorithm (*Self-Composed*)

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.

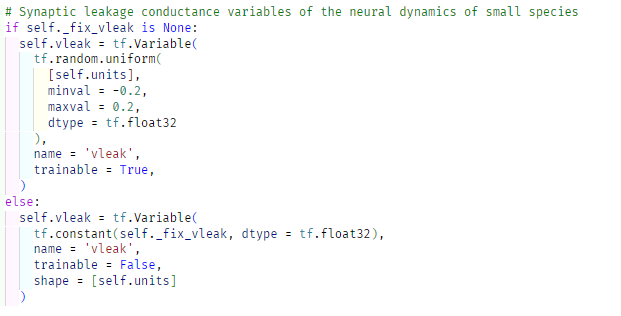


Figure 12: Algorithm – sensory, stochastic and leakage variables (*Self-Composed*)

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

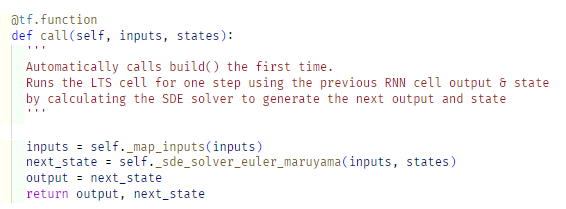


Figure 13: Algorithm – forward propagation (*Self-Composed*)

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

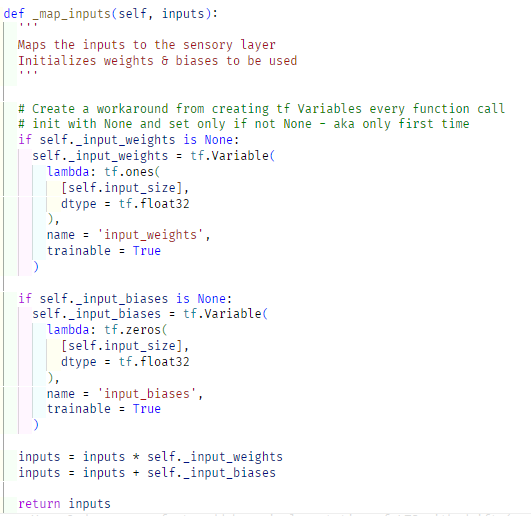


Figure 14: Algorithm – define weights and biases (*Self-Composed*)

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).

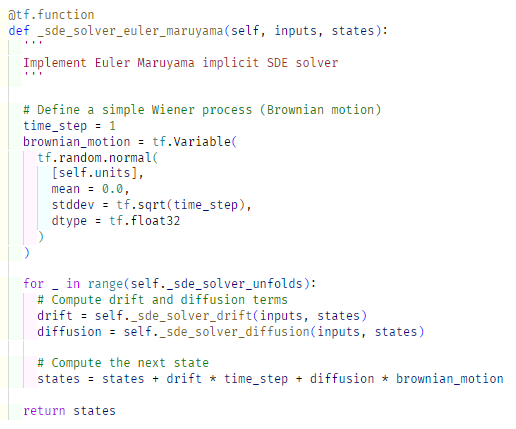


Figure 15: Algorithm – Euler-Maruyama SDE solver (*Self-Composed*)

## **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 D.1**](#_D.1._Fetch_data).

## **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 D.2**](#_D.2._Preprocessing).

# **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 available in [**APPENDIX E.1**](#_E.1._Project_scope). Based on the proposed scope, no deviations have been taken.

## **5.2.2. Schedule related deviations**

The schedule proposed by the author is available in **APPENDIX E.2**. 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 E.3**](#_E.3._Project_roadmap).

# **5.3. Initial test results**

# **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.

# **5.5. Demo of the prototype**

A demo of the prototype was recorded and uploaded as an unlisted video on YouTube, the link can be found here.

# **5.6. 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](Available%20from%20https:/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 – INTRODUCTION**

# **A.1. Research questions**

**R**Q1: What are the top tier transformer architectures widely used and know for NLP problems related to text summarization?

**R**Q2: How can a pretrained transformer architecture be fine-tuned to get the optimal hyper parameters?

**R**Q3: What kind of evaluations should we perform after fine-tuning to filter out the best transformer architecture?

**R**Q4: How can domain generalization be integrated for system?

# **APPENDIX B – SRS**

# **B.1. Requirement elicitation methodologies**

Table 20: 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 |

# **B.2. Survey analysis**

Table 21: 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 supplementary proposed system will be helpful. As identified, not a single participant thought that the system would not be beneficial. Notably, this validates the problem domain and gives the author the ‘green light’ to go ahead. | |
| **Question** | Who do you think would benefit from this system? |
| **Aim of question** | To identify beneficiaries and target audience |
| **Findings & conclusions**    The majority of the participants believed that the system would be beneficial for expert traders, investors as well as a new audience. However, what can be identified, is that a minute portion of participants assumed that the system would be helpful primarily for people who are already involved in the market – this is some evidence that the system must be made as simple as possible to attract a newer audience. It is also identified to help only a new audience – this is evidence that the system must not be immature. | |
| **Question** | This system will also benefit people who are not experts in cryptocurrencies |
| **Aim of question** | To identify whether non-technical crypto traders would benefit |
| **Findings & conclusions**    The responses to the above question show that the system will also apply to audiences who are not cryptocurrency experts. This question goes hand in hand with the previous question to confirm whether the system can target a newer audience of people to get into cryptocurrencies rather than just focusing on a niche audience who are experts or current investors/traders. | |
| **Question** | How do you decide whether to buy or sell assets? |
| **Aim of question** | To understand how a buyer/seller proceeds with their decision |
| **Findings & conclusions**    The responses to the above question are more of a ‘Know Your Customer’ question with no specific project-related purpose. Nevertheless, what can be identified is that most of the respondents have some knowledge of cryptocurrencies, where almost 70% are experienced in trading/investing cryptocurrencies – a great insight as nearly all the respondents have specific knowledge. Therefore, the author could use this to reach out to the respondents (whom they gathered requirements from) during the evaluation phase. | |
| **Question** | Do you think predicting a more future date (ex: a week from now) is as important as tomorrow's price? |
| **Aim of question** | To identify whether a greater future date prediction is also necessary |
| **Findings & conclusions**    The author initially considered only having a single horizon forecast, considering the limited time. However, based on the above responses, it is evident that the audience would also expect forecasts for multi horizons. Therefore, the author will additionally aim to implement the ability of multi-horizon forecasting. | |
| **Question** | Social media trends can impact the price |
| **Aim of question** | To identify whether the community believes that social media trends impact the price |
| **Findings & conclusions**    The majority of the respondents believe that social trends impact the price. Therefore, it is necessary to consider as many trends as possible. Considering the project's limited time and scope, the author has decided to use Twitter volume and Google Trends; however, Reddit, Facebook, and others would also provide insights and could be considered 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. Additionally, based on the responses, the requirement for NER and weighted search is more apparent to give more weightage to specific tweeter's sentiments. | |
| **Question** | Would it be helpful to obtain a range of prices rather than a point price? (Ex: 10,000 - 15,000 instead of 12,500) |
| **Aim of question** | To identify whether including uncertainty estimates is beneficial |
| **Findings & conclusions**    The author initially decided on only providing a point forecast for the system, as this research aims to develop a novel architecture for TS forecasting. However, based on the responses and while conducting prototyping, it became evident that a single-point prediction is likely to be less valuable than a range of prices. A point prediction is implausible to be accurate, which makes the requirement of uncertainty estimates more vital. | |
| **Question** | What functionalities would you expect to have in a bitcoin forecasting system? |
| **Aim of question** | To identify any additional requirements |
| **Findings & conclusions**  To analyze opened ended questions, the author can perform thematic analysis. The analysis, including the theme and related codes, is available in the below table.  Based on the analysis conducted, it is evident that the participants would appreciate some Explainability. Including XAI is an addition that the author could look into if time permits. The participants also mentioned that the system would be better performant and robust if it utilized as many exogenous factors while making it as simple as possible. Based on these findings, the author will aim to include as much Explainability as possible and make it mandatory to use the mentioned exogenous features. | |
| **Question** | Any extra feedback you would like to provide? |
| **Aim of question** | No specific reason – is mainly used to obtain any additional feedback |
| **Findings & conclusions**  A few motivational sentences were submitted to inspire and motivate the author to perform to their best ability. | |

Table 22: 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 convenient and make the inference procedure as straightforward as possible so there is no hindrance. | “Show some news about the current cryptocurrency world in the platform, so it’s convenient for the users.”  “Make the steps from choosing a date to forecasting as simple as possible.” |
| Editability | An ML-knowledgeable participant mentioned that it would be an ideal scenario if the system could tune the hyperparameters of the model in use, which could be an excellent enhancement to the system as the model anyways retrains periodically. | “Coming from machine learning point of view, I think it’ll be a good idea if there’s a functionality to change the hyperparameters used.” |
| Future considerations | A couple of participants mentioned some additional features 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.” |

# **B.3. Interview analysis**

Table 23: 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](#hasani2020ref)). However, the author could further improve the architecture by using SDEs instead (the base LTC uses ODEs), which could manifest into a novel algorithm, which is the author’s current aim as it carries more significance and a potentially more outstanding contribution. | “I think latent ODEs are obsolete.”  “You should look into latent SDEs instead.”  “Latent SDEs are more flexible, you could try applying LTC architectures to those more flexible models instead.” |
| Other suggestions | 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.” |

Table 24: 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. |

# **B.4. Use case descriptions**

Table 25: 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 | 1. Admin authorizes themselves. 2. Admin can change the hyperparameters in use to a set of predefined values. 3. 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. 4. 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. |

# **B.5. Functional requirements**

Table 26: ‘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 C – DESIGN**

# **C.1. Algorithm intuition**



Figure 16: Algorithm intuition (*Self-Composed*)

# **C.2. UI wireframes**

|  |  |
| --- | --- |
| Figure 17: UI wireframes – Home (*Self-Composed*) | Figure 18: UI wireframes – News (*Self-Composed*) |
| Figure 19: UI wireframes – Cryptocurrencies (*Self-Composed*) | Figure 20: UI wireframes – Cryptocurrency (*Self-Composed*) |

|  |  |
| --- | --- |
| Figure 21: UI wireframes – Admin login (*Self-Composed*) | Figure 22: UI wireframes – Admin model configuration (*Self-Composed*) |
| Figure 23: UI wireframes – Forecast (*Self-Composed*) | |

# **APPENDIX D – IMPLEMENTATION**

# **D.1. Fetch data**

**Fetch historical prices**



Figure 24: 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**

|  |  |
| --- | --- |
| Figure 25: Fetch Twitter volume (*Self-Composed*) | Figure 26: 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**

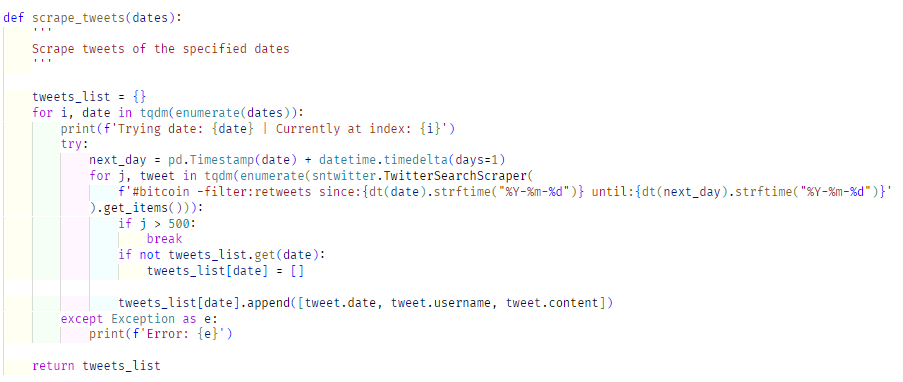


Figure 27: Scrape tweets (*Self-Composed*)

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.



Figure 28: 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**

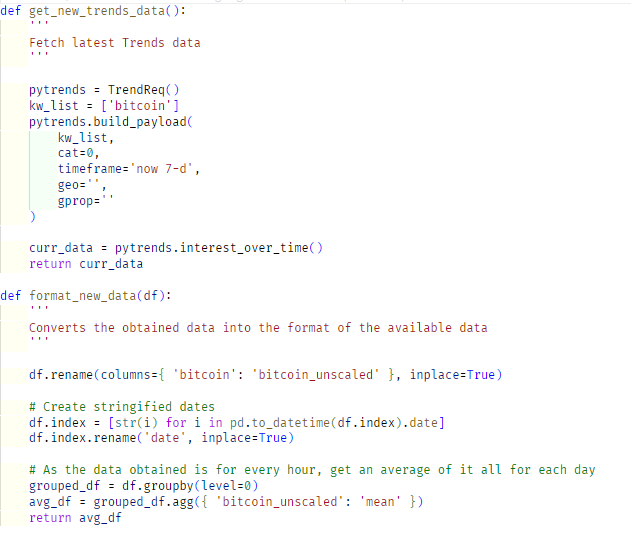


Figure 29: Fetch Google Trends (*Self-Composed*)

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.

# **D.2. 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.

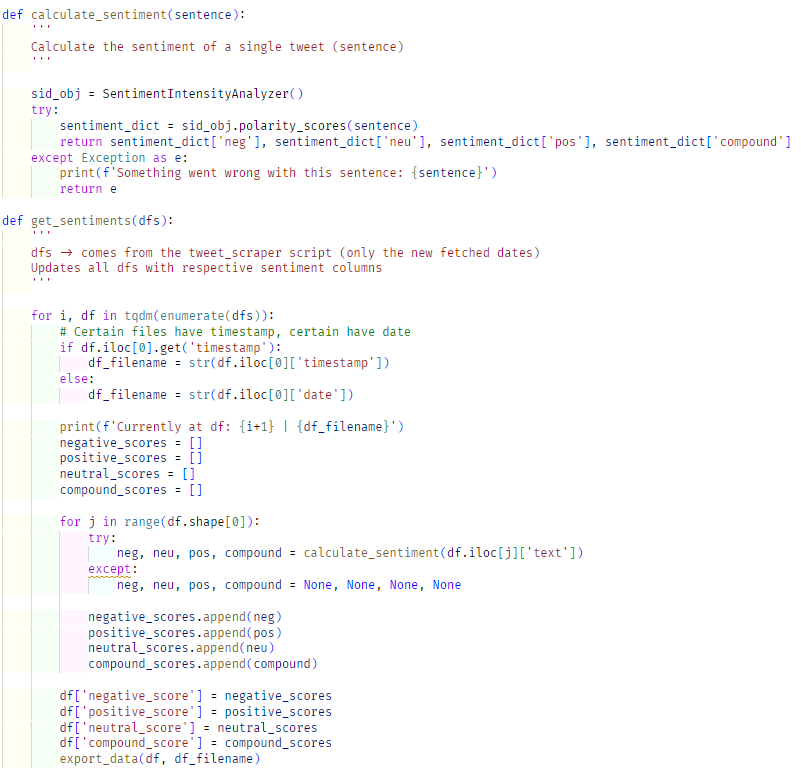


Figure 30: Analyze sentiments (*Self-Composed*)

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**

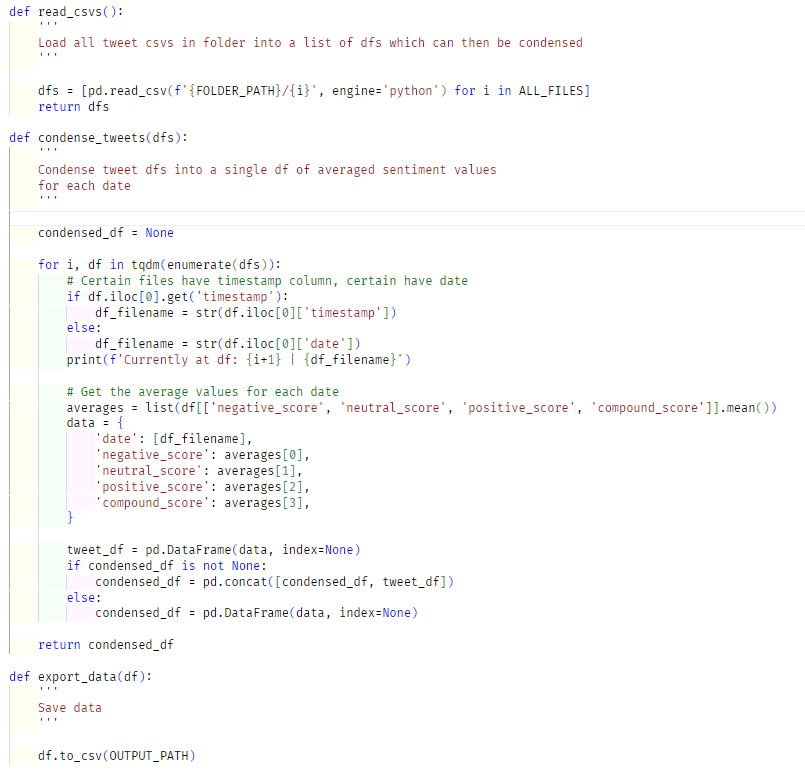


Figure 31: Combine and condense tweets (*Self-Composed*)

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**

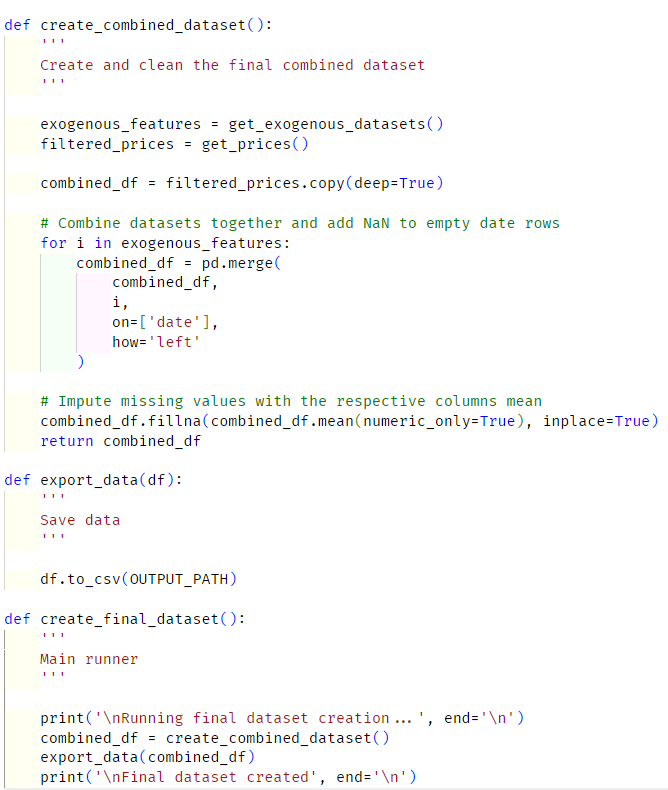


Figure 32: Combine all datasets (*Self-Composed*)

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 E – CONCLUSION**

# **E.1. Project scope**

**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 to validate or invalidate hypothesis [**H**01](#myhypothesis).
* 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.

# **E.2. Project schedule**

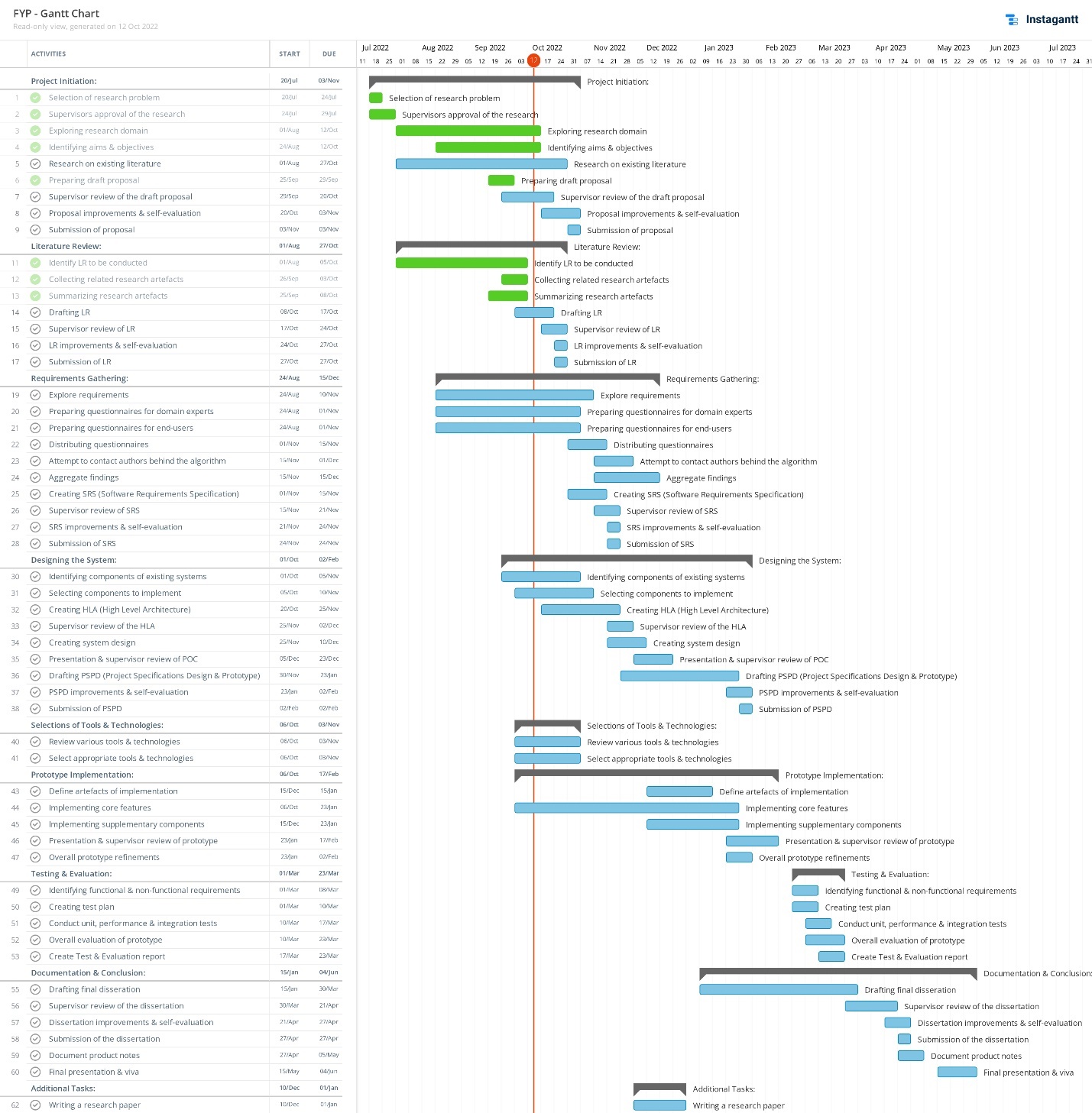
****

Figure 33: Initial Gantt chart (*Self-Composed*)

*A clearer version can be found* [*Here*](https://drive.google.com/file/d/1fpAf_W51Hc4CMBcM5A6T_b77BcfpDzYM/view?usp=sharing)

# **E.3. Project progress**

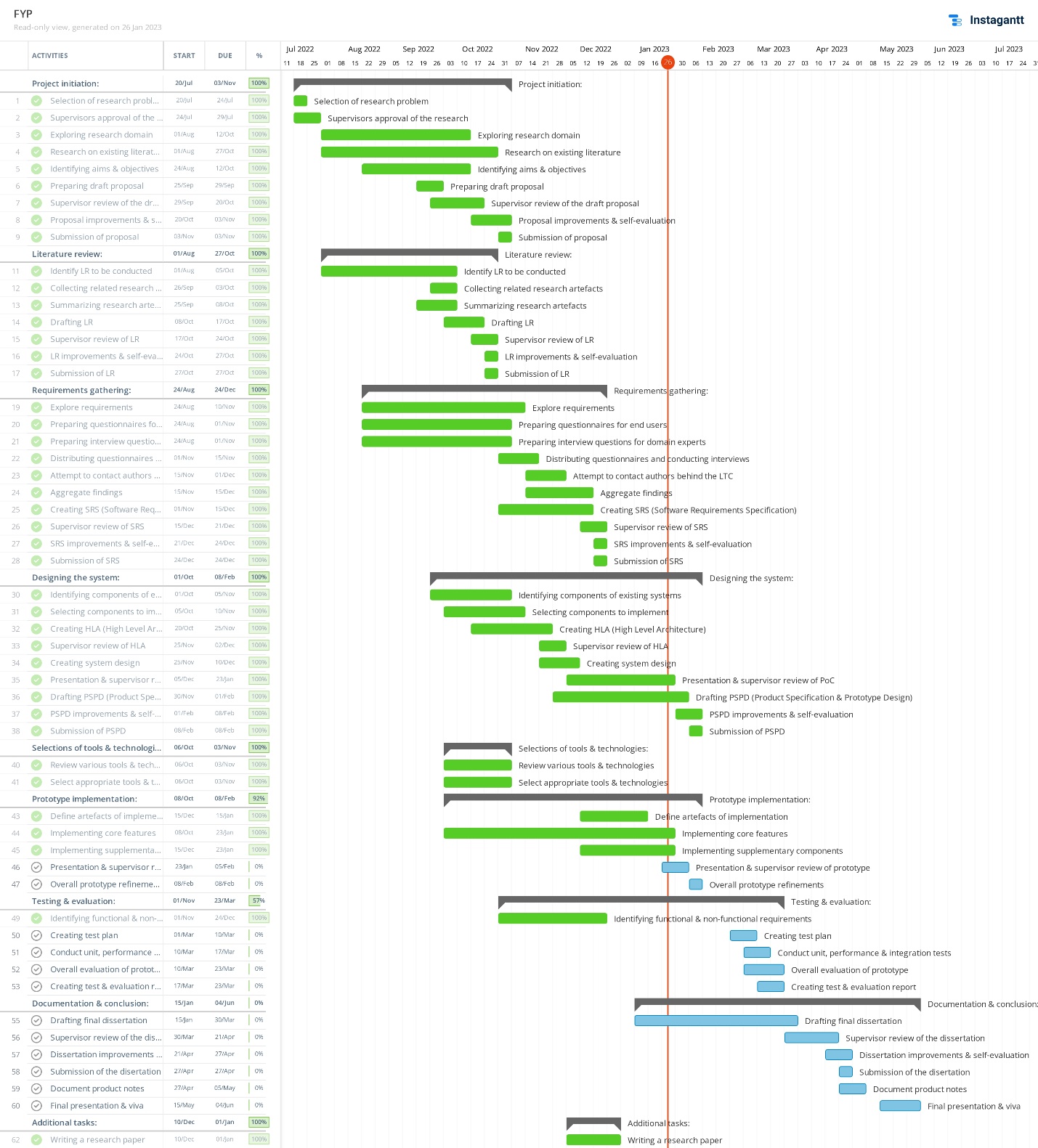


Figure 34: Current Gantt chart (*Self-Composed*)

*A clearer version can be found* [*Here*](https://drive.google.com/file/d/1hxyFfM2JPT-MGs1n7RSaJu0vGc9jKXvD/view?usp=sharing)