**Chapter I**

**INTRODUCTION**

Algorithmic trading, sometimes called high-frequency trading, is the use of automated systems to identify true signals among massive amounts of data that capture the underlying stock market dynamics also it relies on computer programs that execute algorithms to automate some, or all, elements of a trading strategy. Machine Learning has therefore been central to the process of algorithmic trading because it provides powerful tools to extract patterns from the seemingly chaotic market trends

These algorithms encode various activities of a portfolio manager who observes market transactions and analyzes relevant data to decide on placing buy or sell orders. The sequence of orders defines the portfolio holdings that, over time, aim to produce returns that are attractive to the providers of capital, taking into account their appetite for risk.

The ability to accurately predict the stock market, by minimizing misclassification of price direction (increase or decrease) or minimizing the difference between predicted and expected price values, would provide a significant advantage over other traders, with the potential to greatly increase returns on investments. Coupling such a framework with Portfolio Optimization and an intelligent trading agent would reduce risk and allow for long-term investment security.

Ultimately, the goal of active investment management consists in achieving alpha, that is, returns in excess of the benchmark used for evaluation. The fundamental law of active management applies the information ratio (IR) to express the value of active management as the ratio of portfolio returns above the returns of a benchmark, usually an index, to the volatility of those returns. It approximates the information ratio as the product of the information coefficient (IC), which measures the quality of forecast as their correlation with outcomes, and the breadth of a strategy expressed as the square root of the number of bets. Hence, the key to generating alpha is forecasting. Successful predictions, in turn, require superior information or a superior ability to process public information. Algorithms facilitate optimization throughout the investment process, from asset allocation to idea generation, trade execution, and risk management. The use of ML for algorithmic trading, in particular, aims for more efficient use of conventional and alternative data, with the goal of producing both better and more actionable forecasts, hence improving the value of active management.

**ARTIFICIAL INTELIGENCE**

“Artificial Intelligence is neither a new technology nor a machine”. Artificial intelligence is the recognition of outcome-direction which is the rapid analysis of live data to achieve the expected goal. Outcome-directed thinking splits from the confines of the rule-directed approach that is accomplished through artificial intelligence.

“Natural Language Processing (NLP) is a theory motivated range of computational techniques, for the automatic analysis and representation of human language”. Language Processing technology has made great advancements in machine learning based systems to be able to extract meaning from natural language utterances also known as sentiment analysis.

Establishing the context of the of the users message is a vital feature that allows the chatbot to deal with situations that it may not be able to carry out a specific action for. This is due to the user input being very vague or may have an alternative meaning. The context is the capacity of a chatbot to sustain its state, also referred to as the number of user supplied input (utterances) when the context is extracted and the appropriate intent is paired to conduct the desired action for the user.

**CONCLUSION**

In this project, we have formulated the portfolio management problem into a deep RL problem

**Chapter II**

**LITERATURE REVIEW**

**2.1 PROBLEM IDENTIFICATION**

It is evident that the investment industry has evolved dramatically over the last several decades and continues to do so amid increased competition, technological advances, and a challenging economic environment.  The return provided by an asset is a function of the uncertainty or risk associated with the financial investment. An equity investment implies, for example, assuming a company's business risk, and a bond investment implies assuming default risk. To the extent that specific risk characteristics predict returns, identifying and forecasting the behavior of these risk factors becomes a primary focus when designing an investment strategy. It yields valuable trading signals and is the key to superior active-management results. The industry's understanding of risk factors has evolved very substantially over time and has impacted how ML is used for algorithmic trading.

**2.2 PROPOSED SOLUTION**

The proposed solution is to Alpha factors are designed to extract signals from data to predict asset returns for a given investment universe over the trading horizon. A factor takes on a single value for each asset when evaluated, but may combine one or several input variables. The process involves multiple steps.

The Research phase of the trading strategy workflow includes the design, evaluation, and combination of alpha factors. ML plays a large role in this process because the complexity of factors has increased as investors react to both the signal decay of simpler factors and the much richer data available today.

Alpha factors emit entry and exit signals that lead to buy or sell orders, and order execution results in portfolio holdings. The risk profiles of individual positions interact to create a specific portfolio risk profile. Portfolio management involves the optimization of position weights to achieve the desired portfolio risk and return a profile that aligns with the overall investment objectives. This process is highly dynamic to incorporate continuously-evolving market data.

The incorporation of an investment idea into an algorithmic strategy requires extensive testing with a scientific approach that attempts to reject the idea based on its performance in alternative out-of-sample market scenarios. Testing may involve simulated data to capture scenarios deemed possible but not reflected in historic data.

**Chapter III**

**METHODOLOGY**

**3.1 MACHINE LEARNING FOR TRADING**

An algorithmic trading strategy is driven by a combination of alpha factors that transform one or several data sources into signals that in turn predict future asset returns and trigger buy or sell orders.. Market and Fundamental Data and Alternative Data for Finance cover the sourcing and management of data, the raw material and the single most important driver of a successful trading strategy. [Alpha Factor Researc](https://github.com/stefan-jansen/machine-learning-for-trading/blob/master/04_alpha_factor_research)h outlines a methodologically sound process to manage the risk of false discoveries that increases with the amount of data. [Strategy Evaluation](https://github.com/stefan-jansen/machine-learning-for-trading/blob/master/05_strategy_evaluation) provides the context for the execution and performance measurement of a trading strategy.

First and foremost, it illustrates how a broad range of supervised, unsupervised, and reinforcement learning algorithms can be used to extract signals from a diverse set of data sources relevant to different asset classes. It demonstrates how to develop an end-to-end trading strategy and presents ML models as building blocks that extract or combine alpha factors using a systematic workflow. The modular approach implies that not every algorithm is shown as part of a complete strategy. This allows to develop the mathematical and statistical background that in turn facilitate the tuning of an algorithm or the interpretation of the results.

Investors can extract value from third-party data more than other industries. As a consequence, we cover how to work with market and fundamental data but also how to source, evaluate, process, and model alternative data sources such as unstructured text and image data.

* 1. **DEVELOPMENT METHODOLOGIES**

Deciding upon an appropriate methodology is vital for the overall development of any software application to ensure a realistic timeframe is established for each stage of the project and requirements are clearly outlined.

Various development methodologies will be discussed and considered for the development and design of this software. This section will highlight the development methodology that is best suited to this project.

* + 1. **WATERFALL MODEL**

This is a very traditionally methodology, which is usually introduced when you initially learn about software development. The waterfall model is a very predictive approach to software development that consists of 5 stages to include; requirements gathering, analysis, design, implementation and testing. Each stage is completed subsequently of one another. A major drawback of the waterfall model is that it is very inflexible, as the project is broken up into phases.

Each phase is given a deadline in order for a deliverable to be produced at the end of each phase to adhere to the overall project schedule. The success and progression of the project is measured from the project deliverables, design documents and test plans. As each phase of the project is outlined at the beginning of the project lifecycle and targets have been set it’s difficult to integrate new requirements or a change in requirements that may be identified at a later stage as it would adversely affect the overall project schedule. The waterfall model moves a lot of risk and difficult components towards the end of the project life cycle.

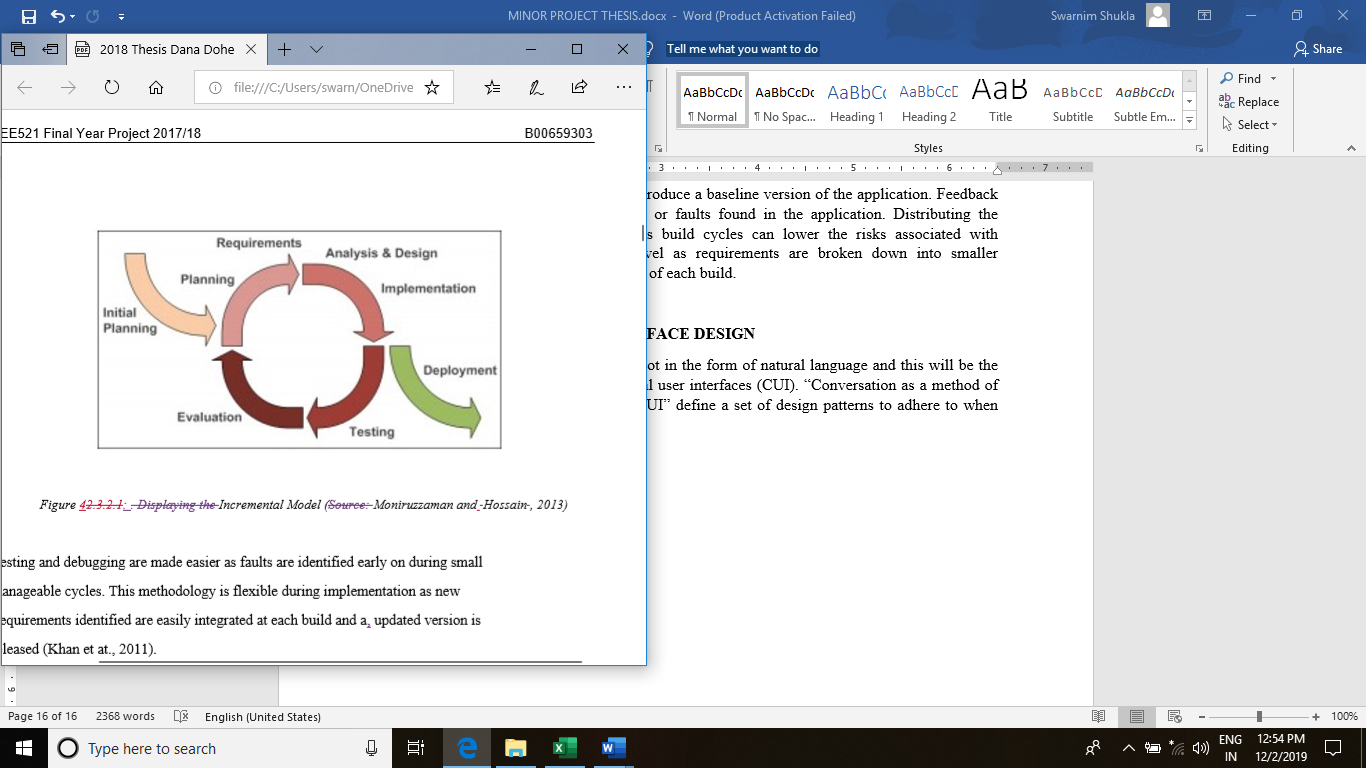
* + 1. **INCREMENTAL MODEL**

This software methodology evolved from the waterfall model. The application is designed, developed and tested using iterative incremental build stages. At the end of each build a subsystem or feature will be created.

The project will progress in complexity as new requirements are likely to be discovered and implemented in each incremental build, developing on top of the functionality from the last build leading to the overall development of the application.

It is very common for software to be released in stages, it is critical that component versions utilized within the software are managed throughout the entire lifecycle using version control tools such as GitHub.

Each build will only last a few weeks to produce a baseline version of the application. Feedback can be given on any requirement errors or faults found in the application. Distributing the development of the project over various build cycles can lower the risks associated with development to a more manageable level as requirements are broken down into smaller functionality to be implemented at the end of each build.



*Fig. 3.2 Incremental Model*

Testing and debugging are made easier as faults are identified early on during small manageable cycles. This methodology is flexible during implementation as new requirements identified are easily integrated at each build and a, updated version is released.

**3.2.3 CHOSEN METHODOLOGY**

The incremental model is the most suitable development methodology to implement for this project. The flexibility of the incremental model makes it ideal for this project as it is likely new requirements will be identified during the later stages of development and each iterative build makes it easy to implement new requirements throughout the development process.

**3.3 FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS**

Functional and Non-functional requirements are identified through the analysis of the data collected. Functional requirements are the features and functionality that the system must have or be able to perform whereas non-functional requirements define the manner or characteristic the system must have such as: performance, usability, modifiability, maintainability, security, scalability, reliability, availability, configurability and design constraints.

**3.3.1 FUNCTIONAL REQUIREMENTS**

• The chat bot must allow users to view information about branches in the college.

• The Chat bot will allow users to view their college branches, nearest airports and railway stations.

• The Chat bot will assist users with their queries and carry out appropriate actions such as scheduling appointments. with finance consultants. • Users will be able to converse with the Chat bot through voice or text commands and it will understand what the user is saying through natural language understanding

• The chat bot should be able to maintain the conversational state when the context may be unclear through previous messages and conversations.

• Provide text responses.

**3.3.2 NON-FUNCTIONAL REQUIREMENTS**

• The chatbot must be efficient with very little lag in response time for instance no longer than 3 seconds to reply to a user message.

• The chatbot must be reliable with no faults or bugs

• The chats must be able to track previous messages too.

• The chatbot must be secure as sensitive data is being used

• The use of natural language used to interact with the chatbot promotes human computer interaction.

• Provide accurate responses for given input.

* Appropriately able to handle unknown messages.

**3.4 HARDWARE AND SOFTWARE SPECIFICATION**

The software and hardware requirements necessary to implement the chatbot are stated below :

**3.4.1 SOFTWARE REQUIREMENTS**

* **Python Libraries :**

1. pandas, numpy, yfinance (Dataset Preparation)

2. zipline, sklearn, alphalens, tqdm, scipy, matplotlib. (Alpha Factors Research)

* **Scripts** :

1. project\_helper

2. risk\_model

3. factors\_pipeline

4. portfolio\_optimizer

5. no\_overlap\_classifier

**3.4.2 HARDWARE REQUIREMENTS**

* PC running Linux/Debian based OS.
* Processor – Intel Core i5

**3.6 ML4T**

**3.6.1 Introduction**

**Linear/Polynomial/Logistic Regression** is regarded to be one of the simplest methods of Machine Learning, modelling the relationship between an output variable and one or more(Multiple Regression) input variables using an unknown function. Multiple Regression is usually required in stock market prediction using Regression (such as in Gustaf Forslund and David˚ Akesson’s attempt due to the many variable inputs that can aﬀect a price’s value. Whilst Regression techniques have been used extensively in the algorithmic trading community3due toits ease of understanding for newcomers to the ﬁeld, it is rarely used by itself for predicting stockprices in research. Instead, Linear/Polynomial Regression is used for technical analysis, being useful in identifying price trends in the long-term but often rarely outperforms the prediction accuracy of other models in the short-term.

**Artiﬁcial Neural Networks** are non-linear models that simulate the processes between neurons and synapses within the brain in order to model complex problems. They have been popular for use in predicting the Stock Market due to their black-box nature, allowing users to try an arbitrary amount of features to generate usable results, without having to know how it works. There are many variations of ANN that can be applied to time-series forecasting, such as Multi layer Perceptrons (MLP) - the simplest variant or Convolutional Neural Networks(CNN) which encode price information into image formats to allow for convolutions to be performed. Memory-based networks such as Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks allow for predictions to be remembered and forgotten ; this can be useful when predicting extreme changes in a short space of time via the remembering of previous examples. Compared to other models, RNN and LSTM are suited towards sequential data such as stock prices, whereas other models must take a non-sequential, sample-independent interpretation of the data, considering only a single input and output sample at a time.

**Random Forests** are ensemble models that utilize a set of weak learners (in this case trees) and apply them to a separate subspace of the input matrix, taking the majority or mean output of all the trees to create a single strong learner. Based on previous research, this method is very popular, reaching accuracies of up to 94.533% for predicting the price direction of a stock at 3 months. Random Forests in general provide very accurate results and are very eﬃcient during runtime, compared to single trees which suﬀer from high variance and bias. They are more interpretable than models such as Neural Networks, with important deciding features being observable based on the splitting attributes at each node.

**ALPHA FACTORS**

Alpha factors are transformations of market, fundamental, and alternative data that contain predictive signals. They are designed to capture risks that drive asset returns. One set of factors describes fundamental, economy-wide variables such as growth, inflation, volatility, productivity, and demographic risk. Another set consists of tradeable investment styles such as the market portfolio, value-growth investing, and momentum investing.

There are also factors that explain price movements based on the economics or institutional setting of financial markets, or investor behavior, including known biases of this behavior. The economic theory behind factors can be rational, where the factors have high returns over the long run to compensate for their low returns during bad times, or behavioral, where factor risk premiums result from the possibly biased, or not entirely rational behavior of agents that is not arbitraged away.

## **From signals to trades: backtesting with zipline**

Zipline is a Pythonic algorithmic trading library. It is an event-driven system for backtesting and live-trading engine powering [Quantopian](https://www.quantopian.com/)  to facilitate algorithm-development and live-trading. It automates the algorithm's reaction to trade events and provides it with current and historical point-in-time data that avoids look-ahead bias.

Historically, alpha factors used a single input and simple heuristics, thresholds or quantile cutoffs to identify buy or sell signals. ML has proven quite effective in extracting signals from a more diverse and much larger set of input data, including other alpha factors based on the analysis of historical patterns. As a result, algorithmic trading strategies today leverage a large number of alpha signals, many of which may be weak individually but can yield reliable predictions when combined with other model-driven or traditional factors by an ML algorithm.

The open source [zipline](http://www.zipline.io/index.html) library is an event-driven backtesting system maintained and used in production by the crowd-sourced quantitative investment fund [Quantopian](https://www.quantopian.com/) to facilitate algorithm-development and live-trading.

It automates the algorithm's reaction to trade events and provides it with current and historical point-in-time data that avoids look-ahead bias.

### The architecture – event-driven trading simulation

* A zipline algorithm will run for a specified period after an initial setup and executes its trading logic when specific events occur.
* These events are driven by the trading frequency and can also be scheduled by the algorithm, and result in zipline calling certain methods.
* The algorithm maintains state through a context dictionary and receives actionable information through a data variable containing point-in-time (PIT) current and historical data.
* The algorithm returns a DataFrame containing portfolio performance metrics if there were any trades, as well as user-defined metrics that can be used to record, for example, the factor values.

The Pipeline API facilitates the definition and computation of alpha factors for a cross-section of securities from historical data. A pipeline defines computations that produce columns in a table with PIT values for a set of securities. It needs to be registered with the initialize() method and can then be executed on an automatic or custom schedule. The library provides numerous built-in computations such as moving averages or Bollinger Bands that can be used to quickly compute standard factors but also allows for the creation of custom factors as we will illustrate next.

Most importantly, the Pipeline API renders alpha factor research modular because it separates the alpha factor computation from the remainder of the algorithm, including the placement and execution of trade orders and the bookkeeping of portfolio holdings, values, and so on.

### A single alpha factor from market data

We are first going to illustrate the zipline alpha factor research workflow in an offline environment.

The notebook [single\_factor\_zipline](https://github.com/stefan-jansen/machine-learning-for-trading/blob/master/04_alpha_factor_research/01_factor_research_evaluation/01_single_factor_zipline.ipynb) develops and test a simple mean-reversion factor that measures how much recent performance has deviated from the historical average. Short-term reversal is a common strategy that takes advantage of the weakly predictive pattern that stock price increases are likely to mean-revert back down over horizons from less than a minute to one month.

### Combining factors from diverse data sources

The Quantopian research environment is tailored to the rapid testing of predictive alpha factors. The process is very similar because it builds on zipline, but offers much richer access to data sources.

The notebook [multiple\_factors\_quantopian\_research](https://github.com/stefan-jansen/machine-learning-for-trading/blob/master/04_alpha_factor_research/01_factor_research_evaluation/02_multiple_factors_quantopian_research.ipynb) illustrates how to compute alpha factors not only from market data as previously but also from fundamental and alternative data.

## Separating signal and noise – how to use alphalens

Quantopian has open sourced the Python library [alphalens](https://github.com/quantopian/alphalens) for the performance analysis of predictive stock factors that integrates well with the backtesting library zipline and the portfolio performance and risk analysis library pyfolio that we will explore in the next chapter.

alphalens facilitates the analysis of the predictive power of alpha factors concerning the:

* Correlation of the signals with subsequent returns
* Profitability of an equal or factor-weighted portfolio based on a (subset of) the signals
* Turnover of factors to indicate the potential trading costs
* Factor-performance during specific events
* Breakdowns of the preceding by sector

The analysis can be conducted using tearsheets or individual computations and plots. The tearsheets are illustrated in the online repo to save some space.

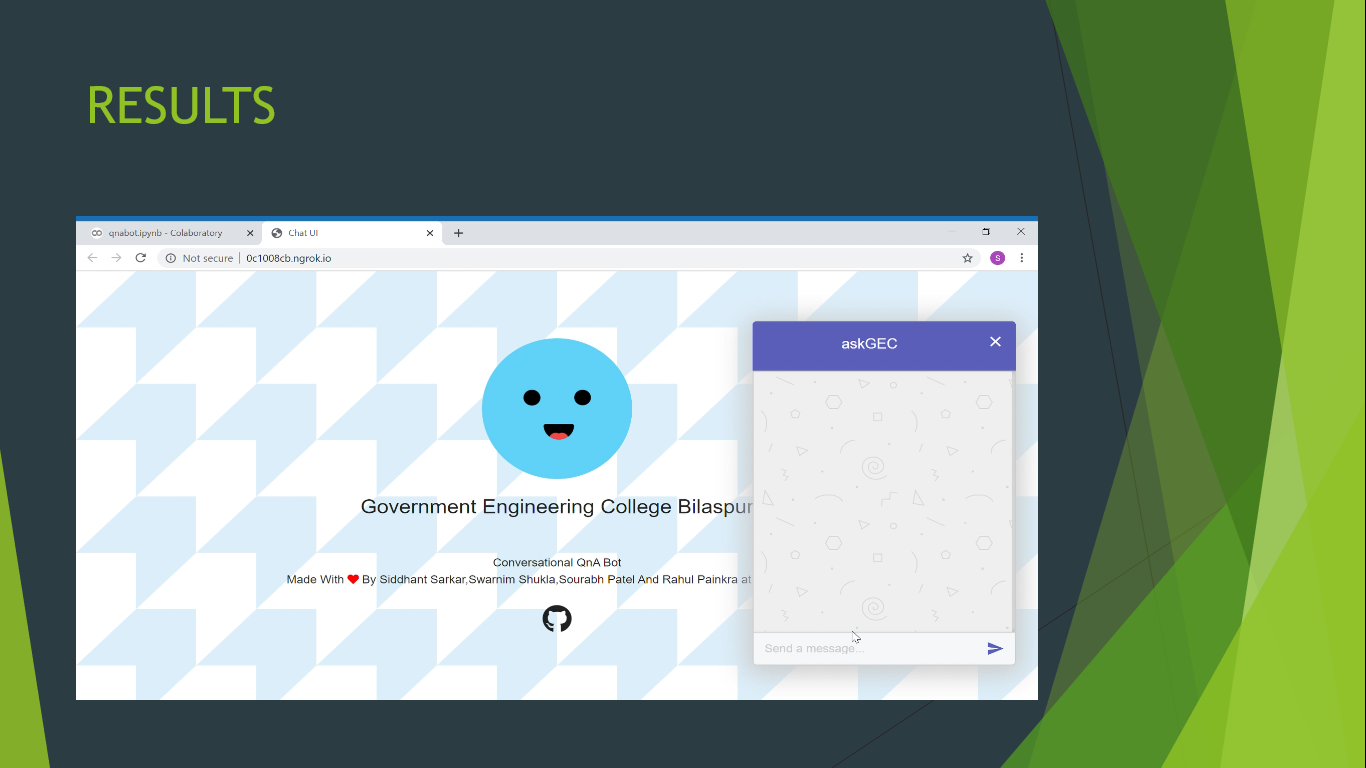
**Portfolio Optimization**

Portfolio Optimisation concerns the selection of stock market indices or stocks from a large feature space, such that the selected indices or stocks provide the least investment risk. This provides a good base-point for automatically trading stocks, by eliminating particularly unpredictable or volatile stocks and giving a more reliable outcome. The majority of the methods of Portfolio Optimisation are based on Mean-Variance Portfolio Optimisation Theory, proposed by Harry Markowitz[18] and have since improved upon his work, allowing for various constraints, such as total amount of shares held, to be considered. Modern attempts at solving this problem include the use of Artificial Intelligence and metaheuristics such as Genetic Algorithms and Simulated Annealing. Although each method has its strengths and weaknesses[15], they have shown good results, boasting returns of over 26%[4]; considering this information, the project will focus on the use of Artificial Intelligence methods when determining optimal portfolio configurations.

**Chapter IV**

**RESULT AND PERFORMANCE ANALYSIS**

The Result Responses are generated within 2-3 seconds of time.



*Fig 4.1 Chat Window*

**Chapter V**

**CONCLUSION AND FURTHER WORK**

**CONCLUSION**

[Alpha Factor Researc](https://github.com/stefan-jansen/machine-learning-for-trading/blob/master/04_alpha_factor_research)h outlines a methodologically sound process to manage the risk of false discoveries that increases with the amount of data. Such stock market trends using machine learning algorithms is a challenging task due to the trends being masked by various factors such as noise and volatility. In addition, the market operates in various local-modes that change from time to time making it necessary to capture those changes in order to be profitable while trading.

Although our algorithms and models were simplified, we were able to meet our expectation of reaching modest profitability. As per our sequential analysis it became clear that factoring in time-locality and capturing the features after smoothing, to reduce volatility improves profitability and precision substantially.

In conclusion, our experience in this project suggests that machine learning has great potential in this field and we hope to continue working on this project further to explore more nuances in improving performance via better algorithms as well as optimizations.

**FUTURE SCOPE**

* Ensemble Of Generative Model And Our Fine Tuned Model.
* Better Reply Generation For More Human Like Feel.
* Getting More Complete And Precise Knowledgebase Covering Every Aspect Of Our Institution For Better Resolving Queries.
* More Fine Tuning With Hyperparameter Optimization For Better Results.
* Integrating Data Collection Mechanism For Storing Conversation History.
* Integrating Google Text To Speech And Speech To Text.
* Integrating A Self Learning Mechanism By Learning From Past Conversation History To Make Our Bot Better Day By Day.

**Chapter VI**

**REFERENCES**

* **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (**<https://arxiv.org/pdf/1810.04805.pdf>)
* **HuggingFace's Transformers: State-of-the-art Natural Language Processing (**<https://arxiv.org/abs/1910.03771>)
* **GPT2** <https://openai.com/blog/better-language-models/>