

STOCK TIME SERIES DATA PREDICTION USING MACHINE LEARNING TECHNIQUES

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF
BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY



by

Batch- 04

G.R.K Sindhu (19JG1A1210)

P.L Prasanna (19JG1A1234)

M.Amrutha varshini (19JG1A1229)

A.Rishitha (20JG5A1202)

Under the esteemed Guidance of

Dr Dwiti Krishna Bebartta

Associate Professor
IT Department

Department of Information Technology

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR
WOMEN**

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation (NBA) for B.Tech. CSE, ECE & IT – Valid from
2019-22 to 2022-25] [Accredited by National Assessment and Accreditation Council [NAAC] with A
Grade – valid from 2022-2027] Kommadi, Madhurawada, Visakhapatnam – 530048

2019 – 2023

**GAYATRI VIDYA PARISHAD COLLEGE OF
ENGINEERING FOR WOMEN**

DEPARTMENT OF INFORMATION TECHNOLOGY



CERTIFICATE

This is to certify that the project report entitled “**STOCK TIME SERIES DATA PREDICTION USING MACHINE LEARNING TECHNIQUES**” is a Bonafede work of the following IV B.Tech. students in the Department of Information Technology of Jawaharlal Nehru Technological University, Kakinada during the academic year 2021-2022, in partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology of this university.

G.R.K Sindhu(19JG1A1210)

P.LPrasanna(19JG1A1234)

M.Amrutha varshini(19JG1A1229)

A Rishitha(20JG5A1202)

(Internal Guide)

Dr. Dwiti Krishna Bebart
Associate Professor
Department of IT

(Head of the Department)

Dr. M. Bhanu Sridhar
Head of Department
Department of IT

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose constant guidance and encouragement crown all the efforts with success.

We feel elated to extend our sincere gratitude to **Dr.Dwiti Krishna Bebarta, Associate Professor,IT** for encouragement all the way during analysis of the project. Her annotations, insinuations and criticisms are the key behind the successful completion of the thesis and for providing us all the required facilities.

We express our deep sense of gratitude and thanks to **Dr. M. Bhanu Sridhar**, Professor and Head of the Department of Information Technology for his guidance and for expressing valuable and grateful opinions in the project for its development and for providing lab sessions and extra hours to complete the project.

We would like to take this opportunity to express our profound sense of gratitude to Vice Principal, **Dr. G. Sudheer** for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We would like to take the opportunity to express our profound sense of gratitude to the revered Principal, **Dr. R. K. Goswami** for allowing us to utilize the college resources thereby facilitating the successful completion of our thesis.

We are also thankful to other teaching faculty and non-teaching staff of the Department of Information Technology for giving valuable suggestions for our project.

VISION & MISSION

Vision of the Institute

To envisage the center of learning that provides value based technical education for the holistic development of women engineers to face the social challenges.

Mission of the Institute

- ☐ To facilitate the students with value-based knowledge in Science, Engineering &
- ☐ Technology. To provide opportunities for learning through Industry, Institute, Interaction on the state-of-the-art technologies.
- ☐ To create a collaborative environment in research, innovation and entrepreneurship to flourish.
- ☐ To promote non-scholastic activities among engineers to cater society needs.

Vision of the Department

The Department of IT strives to produce competent professionals who are technically sound and ethically strong for the IT industry.

Mission of the Department

- ☐ Provide quality training that prepares students to be technically competent for the Industrial and Societal needs.
- ☐ Facilitate an environment that promotes continuous learning to face the challenges in the IT sector.
- ☐ Provide opportunities for learning, leadership and communication skill

TABLE OF CONTENT

| TOPICS | PAGE NO |
|--|---------|
| Abstract | 8 |
| List of Figures | 9 |
| List of Tables | 10 |
| List of Screens | 11 |
| | |
| 1. INTRODUCTION | |
| 1.1 Motivation of the project | 12 |
| 1.2 Problem Definition | 13 |
| 1.3 Objective of Project | 13 |
| 1.4 Limitations of Project | 13 |
| 1.5 Organization of Project | 13 |
| | |
| 2. LITERATURE SURVEY | |
| 2.1 Introduction | 15 |
| 2.2 Libraries | 20 |
| 2.3 Data set | 22 |
| 2.4 Existing system | 24 |
| 2.5 Proposed system | 25 |
| 2.5.1 Dataset collection | 25 |
| 2.5.2 Manual exploration | 25 |
| 2.5.3 Data preprocessing | 25 |
| 2.5.4 Modelling | 26 |
| 2.5.5 Advantages of proposed system | 26 |
| 2.5.6 Proposed system architecture | 27 |
| | |
| 3. REQUIREMENT ANALYSIS | |
| 3.1 Introduction | 28 |
| 3.2 Software Requirement Specification | |
| 3.2.1 Introduction | 28 |
| 3.2.2 Functional Requirements | 29 |
| 3.3 Non - Functional Requirements | 29 |

| | |
|---|----|
| 3.3.1 User Requirements | 30 |
| 3.3.2 Software Requirements | 30 |
| 3.4 Hardware Specifications | 30 |
| 3.5 Content of diagram | 31 |
| 3.6 Algorithms | |
| 3.6.1 long short term memory(LSTM) | 32 |
| 3.6.2 Support Vector Machine (SVM) | 33 |
| 3.6.3 Gated Recurrent Unit (GRU) | 36 |
| 4. DESIGN | |
| 4.1 Introduction | 37 |
| 4.2 UML Diagrams | 37 |
| 4.2.1 Basic building blocks of UML Diagrams | 38 |
| 4.2.2 Class Diagram | 39 |
| 4.2.3 Use Case Diagrams | 42 |
| 4.2.4 Sequence Diagram | 46 |
| 4.2.5 Activity Diagram | 48 |
| 4.2.6 State Chart Diagram | 50 |
| 5. IMPLEMENTATION & RESULTS | |
| 5.1 Project Implementation Steps | 52 |
| 5.2 Implementation | 52 |
| 6. TESTING & VALIDATION | |
| 6.1 Introduction | 64 |
| 6.1.1 Scope | 64 |
| 6.1.2 Defects and Failures | 64 |
| 6.1.3 Compatibility | 65 |
| 6.1.4 Input Combinations and Preconditions | 65 |
| 6.1.5 Static vs Dynamic Testing | 65 |
| 6.1.6 Software Verification and Validation | 66 |
| 6.2 Design of test cases and Scenarios | 66 |
| 6.3 Validation | 71 |
| 6.3.1 Unit Testing | 71 |

| | |
|---------------------------|-----------|
| 6.3.1.1 Black Box Testing | 71 |
| 6.3.1.2 White Box Testing | 72 |
| 6.3.2 Integration Testing | 72 |
| 6.3.3 System Testing | 72 |
| 6.4 Conclusion | 72 |
| | |
| 7. CONCLUSION | 73 |
| | |
| 8. REFERENCES | 74 |

ABSTRACT

Over the years the stock market has been considered a very risky investment by people around the globe. This project aims to understand the historical data of the stock market and derive analysis from it to reduce the gap of knowledge between the market behaviour and the investor. A stock data comprises a lot of statistical terms which are difficult to understand by a normal person who wants to step into stock market investments; this project aims at reducing the gap of knowledge. This study aims to tell the market scenario of the future by supporting it with statistical answers. Stock market volatility, Daily returns, cumulative returns, Correlations between different stocks, Sharpe Ratio of the stocks, CAGR value, Simple Moving Average are some important statistical terms to understand the risk of the investment in the stocks. For the prediction of the future behaviour of stocks we tend to create a hybrid model using LSTM and GRU algorithm which has stronger capability to predict time series stock data more accurately.

LIST OF FIGURES

| S.No | Figure No | Figure Name | Page no |
|-------------|------------------|--|----------------|
| 1 | 2.1 | Stocks | 28 |
| 2 | 2.3.1 | TCS Dataset | 32 |
| 3 | 2.3.2 | TCS-NS Dataset | 33 |
| 4 | 2.5.6 | Proposed System Architecture | 43 |
| 5 | 3.5 | Content Of The Diagram | 41 |
| 6 | 3.6.1.1 | LSTM Model | 27 |
| 7 | 3.6.1.2 | LSTM Architecture | 34 |
| 8 | 3.6.2.1 | SVM model | 8 |
| 9 | 3.6.2.2 | Datasets | 47 |
| 10 | 3.6.2.3 | 2D-space | 23 |
| 11 | 3.6.2.3 | SVM model with hyperplane | 11 |
| 12 | 3.6.3 | GRU Model | 49 |
| 13 | 4.2.2 | Class Diagram | 33 |
| 14 | 4.2.3.1 | Use Case Diagram | 48 |
| 15 | 4.2.4 | Sequence Diagram | 78 |
| 16 | 4.2.5 | Activity Diagram | 15 |
| 17 | 4.2.6 | State Chart Diagram | 43 |
| 18 | 5.1 | Google collab | 43 |
| 19 | 5.1.2 | Data Preprocessing | 42 |
| 20 | 5.2.3.1 | Plot for Training Dataset | 89 |
| 21 | 5.2.3.2 | Plot for Testing Dataset | 56 |
| 22 | 5.2.3.3 | R2 Score Error | 24 |
| 23 | 5.2.3.4 | Error for testing dataset | 75 |
| 24 | 5.2.3.5 | Error for Training and Testing Dataset | 49 |
| 25 | 5.2.3.6 | Prediction for Training Data | 65 |
| 26 | 5.2.3.7 | Prediction for Testing Dataset | 72 |

LIST OF TABLES

| S.No | Figure No | Table Name | Page No |
|-------------|------------------|-------------------|----------------|
| 1 | 6.2 | Validation Table | 66 |

List of Screens

| S.NO | Figure no | Screen Name | Page No |
|-------------|------------------|-----------------------------------|----------------|
| 1 | 5.2.3 | Dataset Output | 60 |
| 2 | 5.1.2 | Data Preprocessing Output | 57 |
| 3 | 5.2.3.5 | Error in Training and Testing Set | 63 |
| 4 | 5.2.3.1 | Plot of Training and Testing Set | 64 |
| 5 | 6.2.1 | Test Case 1 output | 69 |
| 6 | 6.2.2 | Test Case 2 output | 70 |
| 7 | 6.2.3 | Test Case 3 output | 72 |
| 8 | 6.2.5 | Test Case 4 output | 73 |

1. INTRODUCTION

1.1. MOTIVATION OF THE PROJECT

The charm of getting profit by suitably investing the stocks in the stock market attracts thousands of investors. Since every investor wants profit with less risk, they need realistic models to predict the stock price. As investors are investing more and more money in the market, they get anxious to know the future trends of the various stocks available in the market. The major part of the trends in the market is to know when to buy, hold or sell the stocks. Stock market prediction is observed as a challenging task because of high fluctuation and irregularity. Thus, numerous models have been depicted to provide the investors with more precise predictions. Stock market forecasting has attracted a lot of research interests in previous literature. With a successful model for stock prediction, we can gain insight about market behaviour over time, spotting trends that would otherwise not have been noticed. With the increasingly computational power of the computer, machine learning will be an efficient method to solve this problem. However, the public stock dataset is too limited for many machine learning algorithms to work with. We want to introduce a framework in which we integrate user predictions into the current machine learning algorithm using public historical data to improve our results. The motivated idea is that, if we know all information about today's stock trading (of all specific traders), the price is predictable. Thus, if we can obtain just partial information, we can expect to improve the current prediction a lot. With the growth of the Internet, social networks, and online social interactions, getting daily user predictions is a feasible job. Thus, our motivation is to design a hybrid model or a stronger model that will benefit everyone.

To predict the stock we have various techniques in data mining and neural networks have been employed to find out which model gives the accurate prediction. The prediction of stock is classified based on various methods like LSTM (Long short term memory), SVM (Support vector Machine), GRU (Gated Recurrent Network). The prediction of time series data is complex as the difference between data in time series data is very minute, Hence the data must be handled carefully. We have also seen recent developments in machine learning. We want to implement all models. We need to test the accuracy rate and the data need to be processed and the data need to be visualized. Then we need to check the visualisation chart and compare all models and select any two better models and combine them to get the hybrid model. Then the hybrid model data is converted to time-series data and from that data pattern is generated. At last Closing price and entry/exit points are visualised and displayed.

The main objective of this research is to improve the performance accuracy of stock prediction. Many studies have been conducted that result in restrictions of feature selection for algorithmic use. In contrast, the hybrid method uses all features without any restrictions of feature selection. Here we conduct experiments used to identify the features of a machine learning algorithm with a hybrid method. The experiment results show that our proposed

hybrid method has stronger capability to predict time series stock data compared to existing methods.

1.2 PROBLEM DEFINITION

The rate of investment and business opportunities in the Stock market can increase if an efficient algorithm could be devised to predict the short term price of an individual stock. The predicted results can be used to prevent the previous methods of stock predictions which has an error loss at an average of 20%. The overall objective of my work will be to predict accurately the closing price of the stock. Attributes considered form the primary basis for tests and give accurate results more or less. Many more input attributes can be taken but our goal is to predict with few attributes and faster efficiency the closing price of stock. Decisions are often made based on the knowledge rich data hidden in the data set and databases. We want to create a model that has stronger capability to predict the closing price of stock.

1.3 OBJECTIVE OF THE PROJECT

The objective of the project is to improve the accuracy of the time series stock data prediction based on the hybrid model.

1.4 LIMITATIONS OF THE PROJECT

Stock market prediction is a challenging task to predict the upcoming stock values. It is very difficult to predict because of its unstable nature.

With the resurgence of machine learning and artificial intelligence, never has it been easier to implement predictive algorithms both new and old. With just a few lines of code, state of the art models can be readily accessible at the fingertips of the budding data enthusiast, ready to conquer whatever insurmountable digital task may lie at hand. But a little bit of knowledge can be a dangerous thing. While much of machine learning can be attributed to statistics and programming, what is equally important, but often skipped over in favour of instant gratification, is domain knowledge.

.It is very difficult to predict because of its unstable nature. The hybrid model has the potential to generate a knowledge-rich environment which can help to significantly improve the accuracy of predicted data.

1.5 ORGANISATION OF THE PROJECT

The documentation of our project has been divided into the following sections:

Chapter 1: Introduction describes the motivation of this project and objective of the developed project.

Chapter 2: Literature survey describes the primary terms involved in the development of this project and overview of the papers that were referred before starting the project.

Chapter 3: The Analysis chapter deals with detailed analysis of the project. Software Requirement, Specification further contains user requirements analysis, software requirement analysis and hardware requirement analysis.

Chapter 4: Design includes UML diagrams along with explanation of the system design and the organisation.

Chapter 5: Contains step by step implementation of the project and screenshots of the outputs.

Chapter 6: Gives the testing and validation details with design of test cases and scenarios along with the screenshots of the validations.

Chapter 7: In this section we conclude the project by citing all the important aspects of the project along with the different ways in which the project can be developed further are given in the future enhancement section of the conclusion.

Chapter 8: This chapter contains the references to our project.

2. LITERATURE SURVEY

2.1 INTRODUCTION

The world's stock markets encompass enormous wealth. In 2019, the value of global equities surpassed \$85 trillion (Pound, 2019). As long as markets have existed, investors have searched for ways to acquire knowledge about the companies listed in the market to improve their investment returns. In the past, investors relied upon their personal experience to identify market patterns, but this is not feasible today due to the size of the markets and the speed at which trades are executed. Simple statistical analysis of financial data provides some insights but, in recent years, investment companies have increasingly used various forms of artificial intelligence (AI) systems to look for patterns in massive amounts of real time equity and economic data. These systems support human investment decision making and they have now been used for a sufficiently long period that their features and performance can be reviewed and analysed to identify which systems improve predictive performance when compared with other techniques. The objective for this study is to identify directions for future machine learning (ML) stock market prediction research based upon a review of current literature.

A systematic literature review methodology is used to identify relevant peer-reviewed journal articles from the past twenty years, evaluate and categorise studies that have similar methods and contexts, and then compare the studies in each category to identify common findings, unique findings, limitations, and areas that need further investigation. This will provide artificial intelligence and finance researchers with directions for future research into the use of ML techniques to predict stock market index values and trends. A systematic literature review methodology has provided insights into ML applications across a wide range of information technology (IT) and scientific domains.

The following are four highly cited articles that used this methodology. Wen et al. (2012) employed a systematic literature review to evaluate empirical studies of ML models for software development effort estimation (SDEE). SDEE is the process for predicting the effort required for a software development project. The selected articles were published between 2001 and 2021. The authors looked at four dimensions for each system: machine learning technique, model estimation accuracy, model components, and estimation context. This literature review methodology provided insights into the current state of SDEE research that was the basis for researcher recommendations and guidelines for practitioners.

In another IT-related context, Malhotra (2015) conducted a review of literature where ML models were used for software fault prediction. Software fault prediction is a process used in the early phases of the software development lifecycle for detecting faulty software modules or classes. The articles that were reviewed were published between 1991 and 2013. Each

article was evaluated based on their predictive performance and a comparison with other statistical and machine learning techniques. The articles were ultimately grouped into seven categories and the overall conclusion was that more work was needed to produce generalizable results for ML-based software fault prediction. The study concluded with a set of practitioner guidelines and researcher recommendations. The systematic literature review methodology has also been used in highly complex scientific prediction domains. One study evaluated machine learning flood prediction systems (Mosavi, Ozturk and Chau, 2018). Machine learning has been used to model complex physical flood processes. These models were intended to aid hydrologists in predicting floods in the short-term and long-term, and identifying cost-effective solutions that minimise risk, loss of human life and property damage. The ML methods described in each article were evaluated based on their robustness, accuracy, effectiveness and speed. Based on the findings from the review, the authors were able to provide guidelines for hydrologists and climate scientists when choosing the best machine learning technique for a prediction task.

This review methodology has also been used for analysis like, for example, ML-based orthopaedics systems (Cabitza, Locoro and Banfi, 2018). Orthopaedics is the area of medicine focused on prevention, diagnosis, and treatment of bone and muscle disorders. The authors identified studies of machine learning orthopaedics applications that were published over a twenty-year period. They appraised each article based upon its ML technique, orthopaedic application, model data, and predictive performance. This present study utilizes a methodology that is similar to the ones described above because it is also evaluating articles describing ML use for predictions in a highly complex problem domain. In this methodology, relevant articles are selected through a systematic search process and studies using similar ML methods are grouped together. A research framework (taxonomy) is then provided to encompass each of the study categories and provide a description for how the categories differ. The studies in each taxonomy category are then assessed to identify common and unique findings within the set of studies.

This provides the basis for making researcher recommendations. The remainder of the paper is organised in the following sections. First, the process used to identify relevant studies is described. Next, based on an assessment of the studies identified, a research framework (taxonomy) is presented that groups the studies based on the ML technique used to predict stock market index values and trends. The studies in each category are then individually summarized and discussed to identify common findings, unique findings, limitations, and areas where more study is needed. The final section provides some answers related to the overall study objective that focuses on identifying directions for future research and recommendations for improving study generalizability.

Money related trade judgement making is a strengthening and difficult errand of fiscal data guess. Figure about securities trade with high exactness improvement return advantage for examiners of the stocks. In perspective on the snare of budgetary trade financial data, expansion of productive models for forecast conclusion is very difficult, and it must be precise. This considered attempting to make models for guessing the securities trade and

to pick whether to buy/hold the stock using data mining and AI techniques. The AI frameworks like Naive Bayes, LSTM(Long-short term memory),SVM(Support Vector Machine),GRU(Gated recurrent network). Particular pointers are resolved from the stock prices set up on timetable data and it is used as commitments of the proposed guess models. Ten years of securities trade data has been used for the sign gauge of stock. Based on the instructive accumulation, these models can make buy/hold signals for monetary trade as a yield. The rule target of this errand is to deliver yield signal(buy/hold) as per customers essential like mean be contributed, time term of endeavour, least advantage, most prominent hardship, using data mining and AI frameworks. Forecasting the way of stock prices is a widely deliberate subject in many fields including trading, finance, statistics and computer science. Depositors in the stock market can maximise their yield by export or selling their investment if they can determine when to enter and exit a position. Specialised traders typically use essential and/or technical analysis to inspect stocks in making venture decisions. The objective function is to maximise medium to longer term profits based on S&P 500 stock market index. The inputs are the technical pointers data and the economic indicators data. Three models (k nearest neighbour, logistic regression, decision tree) are then used to predict the buy/sell decisions.

A. TIME SERIES DATA

Time series data is a collection of data that is stored periodically over a period. It is usually represented by a line chart. Time series data may be in the form of annual, quarterly, or monthly data

B. STOCKS

A stock is a general term used to describe the ownership certificates of any company. A share, on the other hand, refers to the stock certificate of a particular company. Holding a particular company's share makes you a shareholder. Stocks are of two types common and preferred. The difference is while the holder of the former has voting rights that can be exercised in corporate decisions, the later doesn't. However, preferred shareholders are legally entitled to receive a certain level of dividend payments before any dividends can be issued to other shareholders. There is also something called 'convertible preferred stock. This is basically a preferred stock with an option of converting into a fixed number of common shares, usually any time after a predetermined date.

Investing in stocks can be a key part of your personal finance strategy. The primary reason most people buy stocks is to generate a long-term return on their investment (ROI) that exceeds that of other prominent asset classes, such as bonds, real estate and commodities. Generally, this is achieved in two ways.

While many investors benefit from both high dividend yield and price appreciation, some do not. Not all stocks pay dividends, and many suffer from price depreciation rather than appreciation. As a result, prudent investors avoid establishing highly concentrated positions in a few stocks. Rather, they build diversified portfolios that include a variety of companies spanning different industries and geographic regions.

Beyond the potential financial benefits, most stocks also offer investors voting rights on key governance matters. Given their relatively small and uninfluential ownership positions, this is rarely a focal point for individual investors. However, institutional investors with significant ownership stakes tend to highly value voting rights.



Fig 2.1 : Stocks

Types of stock:

- Public stock
- Private stock
- Common stock
- Preferred stock

Public stock: There are publicly traded stocks and privately held stocks. The former is what most people think of when they hear the phrase “stock market.” Publicly traded stocks consist of fairly well-known companies whose shares are traded on highly regulated exchanges, such as the New York Stock Exchange and the Nasdaq stock market.

Private stock: Private markets involve much less regulation than public markets, and they are comparatively illiquid and volatile. To protect unsophisticated U.S. investors from these pitfalls, the Securities and Exchange Commission (SEC) largely limits investment in this space, allowing only relatively wealthy and/or highly knowledgeable, accredited investors to buy privately placed securities.

Common vs. Preferred Stocks:

Most equity investors own publicly traded common stock. These offer voting rights and the possibility for dividends and price appreciation, but there is another type of stock favoured by some investors - preferred stock. Preferred shareholders rarely have the right to vote on company matters, but they are entitled to receive dividend payments before common shareholders. Often, they receive these payments at a higher dividend yield. Preferred shareholders also have a priority claim on assets in the event of a bankruptcy proceeding or liquidation.

This priority positioning manifests itself via the risk-return trade-off, the investment principle that shows that a higher level of return is only achievable by assuming a higher level of risk. While common shareholders may have greater return potential than preferred shareholders, they also face an increased risk of losing their money because they sit at the bottom of the capital stack.

Different class of stocks:

Some companies issue different classes of stock. Generally, this is done when the company wishes to differentiate shareholder voting rights and/or dividend offerings across classes. For example, the Class A common shares of a certain company may provide greater voting power per share than the Class B common shares of the same company. Alternatively, the Class A1 preferred shares of a certain company may provide higher dividend yields than the Class B1 preferred shares of the same company.

Stock Market Prediction:

Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behaviour, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy.

C. Machine learning

Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behaviour, and so on. All these factors combine

to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy.

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

Machine learning has been applied to several types of tasks, such as time-series forecasting using machine learning techniques. Recurrent Neural Network (RNN) applies to sequential data. There are two major parts, input data and the hidden state, which collect the results from the previous node. The two parts are combined and the results are input into the next node.

RNN's algorithm includes Long Short-Term Memory (LSTM) which finds the solution for RNN problems with long sequence time series data. It works by having a cell state that holds the state of each node so that it can reverse the values obtained from the previous state and have the gate as the flow control. The Gated Recurrent Unit (GRU) operates the same as the LSTM algorithm. Each node has a process by combining the forget gate and the input gate into the update gate. Also, it includes the cell state and the hidden state together. The performance is equivalent to the LSTM algorithm. However, GRU has simpler architecture and fewer parameters to generate sequential data

2.2 Libraries:

In today's world, users can use the Python language, which is the most popular and productive language for machine learning. Python has replaced many languages as it is a vast collection of libraries, and it makes work easier and simpler.

These are the libraries that we used in our project:

- **NumPy**
- **Sklearn or Scikit-learn**
- **TensorFlow**
- **Keras**
- **Pandas**
- **Matplotlib**

Numpy:

NumPy is the most popular library in Python. This library is used for processing large multi-dimensional array and matrix formation by using a large collection of high-level mathematical functions and formulas. It is mainly used for the computation of fundamental science in machine learning. It is widely used for linear algebra, Fourier transformation, and random number capabilities. There are other High-end libraries such as TensorFlow, which use NumPy as internal functions for manipulation of tensors.

Scikit-learn:

Scikit-learn is a Python library which is used for classical machine learning algorithms. It is built on the top of two basic libraries of Python, that is NumPy and SciPy. Scikit-learn is popular in Machine learning developers as it supports supervised and unsupervised learning algorithms. This library can also be used for data-analysis, and data-mining process.

TensorFlow :

Tensorflow is an open-source library of Python which is used for high performance of numerical computation. It is a popular library, which was developed by the Brain team in Google. TensorFlow is a framework that involves defining and running computations involving tensors. TensorFlow can be used for training and running deep neural networks, which can be used for developing several Artificial Intelligence applications.

Keras:

Keras is a high-level neural networking API, which is capable of running on top of TensorFlow, CNTK and Theano libraries. It is a very famous library of Python among Machine learning developers. It can run without a glitch on both CPU and GPU. IT makes it

really easy and simple for Machine Learning beginners and for designing a Neural Network. It is also used for fast prototyping.

Pandas:

Pandas is a Python library that is mainly used for data analysis. The users have to prepare the dataset before using it for training the machine learning. Pandas make it easy for the developers as it is developed specifically for data extraction. It has a wide variety of tools for analysing data in detail, providing high-level data structures.

Matplotlib:

Matplotlib is a Python library that is used for data visualization. It is used by developers when they want to visualize the data and its patterns. It is a 2-D plotting library that is used to create 2-D graphs and plots.

It has a module pyplot which is used for plotting graphs, and it provides different features for control line styles, font properties, formatting axes and many more. Matplotlib provides different types of graphs and plots such as histograms, error charts, bar charts and many more.

2.3 Dataset:

This dataset consists of the last 5 years of stock data. It consists of technical indicators like:

On-balance volume (OBV)

Accumulation/distribution line

Average directional index

Aroon oscillator

Moving average convergence divergence (MACD)

Relative strength index (RSI)

Stochastic oscillator

It also contains attributes like open, close, high, low, volume etc.

TCS(Tata consultancy service):

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|----|------------|--------|--------|------------|---------|--------|---------|---------|---------|---------|----------|-----------|--------|--------------|------------|---|
| 1 | Date | Symbol | Series | Prev Close | Open | High | Low | Last | Close | VWAP | Volume | Turnover | Trades | Deliverable% | Deliverble | |
| 2 | 25-08-2004 | TCS | EQ | 850 | 1198.7 | 1198.7 | 979 | 985 | 987.95 | 1008.32 | 17116372 | 1.726E+15 | | 5206360 | 0.3042 | |
| 3 | 26-08-2004 | TCS | EQ | 987.95 | 992 | 997 | 975.3 | 976.85 | 979 | 985.65 | 5055400 | 4.983E+14 | | 1294899 | 0.2561 | |
| 4 | 27-08-2004 | TCS | EQ | 979 | 982.4 | 982.4 | 958.55 | 961.2 | 962.65 | 969.94 | 3830750 | 3.716E+14 | | 976527 | 0.2549 | |
| 5 | 30-08-2004 | TCS | EQ | 962.65 | 969.9 | 990 | 965 | 986.4 | 986.75 | 982.65 | 3058151 | 3.005E+14 | | 701664 | 0.2294 | |
| 6 | 31-08-2004 | TCS | EQ | 986.75 | 986.5 | 990 | 976 | 987.8 | 988.1 | 982.18 | 2649332 | 2.602E+14 | | 695234 | 0.2624 | |
| 7 | 01-09-2004 | TCS | EQ | 988.1 | 990 | 995 | 983.6 | 986 | 987.9 | 989.68 | 2491943 | 2.466E+14 | | 790586 | 0.3173 | |
| 8 | 02-09-2004 | TCS | EQ | 987.9 | 989.9 | 1004.6 | 986 | 994 | 993.65 | 996.96 | 2669544 | 2.661E+14 | | 501792 | 0.188 | |
| 9 | 03-09-2004 | TCS | EQ | 993.65 | 1006 | 1100 | 990.35 | 998.7 | 997.85 | 996.91 | 1233732 | 1.23E+14 | | 235508 | 0.1909 | |
| 10 | 06-09-2004 | TCS | EQ | 997.85 | 1039.9 | 1039.9 | 992.9 | 996.8 | 994.85 | 998.87 | 1129834 | 1.129E+14 | | 430184 | 0.3807 | |
| 11 | 07-09-2004 | TCS | EQ | 994.85 | 1035 | 1035 | 995 | 995 | 995.6 | 997.34 | 721529 | 7.196E+13 | | 198212 | 0.2747 | |
| 12 | 08-09-2004 | TCS | EQ | 995.6 | 996 | 1000.8 | 991.1 | 992.25 | 993.7 | 995.29 | 824248 | 8.204E+13 | | 220195 | 0.2671 | |
| 13 | 09-09-2004 | TCS | EQ | 993.7 | 997 | 997.9 | 978.45 | 979.65 | 979.95 | 985.16 | 993398 | 9.787E+13 | | 364189 | 0.3666 | |
| 14 | 10-09-2004 | TCS | EQ | 979.95 | 990 | 990 | 976 | 989.95 | 988.8 | 985.12 | 801884 | 7.9E+13 | | 203388 | 0.2536 | |
| 15 | 13-09-2004 | TCS | EQ | 988.8 | 991 | 1015.5 | 991 | 1002.75 | 1003.5 | 1007.31 | 2613114 | 2.632E+14 | | 735865 | 0.2816 | |
| 16 | 14-09-2004 | TCS | EQ | 1003.5 | 1005 | 1018.8 | 1002.95 | 1015.95 | 1015.65 | 1012.95 | 1916934 | 1.942E+14 | | 493998 | 0.2577 | |
| 17 | 15-09-2004 | TCS | EQ | 1015.65 | 1018 | 1020 | 1001.2 | 1005 | 1006.1 | 1009.12 | 1498536 | 1.512E+14 | | 483466 | 0.3226 | |
| 18 | 16-09-2004 | TCS | EQ | 1006.1 | 1007 | 1015 | 1002.5 | 1011 | 1008.45 | 1009.82 | 919778 | 9.288E+13 | | 240651 | 0.2616 | |
| 19 | 17-09-2004 | TCS | EQ | 1008.45 | 1012 | 1029.9 | 1010.3 | 1029.9 | 1024.5 | 1021.09 | 1828487 | 1.867E+14 | | 538239 | 0.2944 | |
| 20 | 20-09-2004 | TCS | EQ | 1024.5 | 1032.4 | 1036.9 | 1020.15 | 1022.5 | 1022.9 | 1028.83 | 1069028 | 1.1E+14 | | 305251 | 0.2855 | |
| 21 | 21-09-2004 | TCS | EQ | 1022.9 | 1024.85 | 1048.5 | 1022 | 1047.4 | 1045.8 | 1039.54 | 1737135 | 1.806E+14 | | 413801 | 0.2382 | |
| 22 | 22-09-2004 | TCS | EQ | 1045.8 | 1048.4 | 1056 | 1037.1 | 1049.9 | 1052.05 | 1047.57 | 1921448 | 2.013E+14 | | 443693 | 0.2309 | |

Fig 2.3.1 :TCS Dataset

TCS-NS(National stock):

| 1 | Date | Open | High | Low | Close | Adj Close | Volume | |
|----|------------|-----------|-----------|-----------|-----------|-----------|---------|--|
| 2 | 05-01-2022 | 3865 | 3870 | 3812.3999 | 3860.95 | 3808.4539 | 1733031 | |
| 3 | 06-01-2022 | 3812 | 3835 | 3772 | 3807.45 | 3755.6814 | 1810293 | |
| 4 | 07-01-2022 | 3820 | 3864.8999 | 3796.3999 | 3853.5 | 3801.1052 | 2460591 | |
| 5 | 10-01-2022 | 3978 | 3978 | 3861 | 3879.8501 | 3827.0972 | 3937092 | |
| 6 | 11-01-2022 | 3856 | 3925 | 3856 | 3915.8999 | 3862.6567 | 1906106 | |
| 7 | 12-01-2022 | 3925 | 3929 | 3836.55 | 3859.8999 | 3807.4177 | 3203744 | |
| 8 | 13-01-2022 | 3918 | 3923 | 3857 | 3897.8999 | 3844.9016 | 6684507 | |
| 9 | 14-01-2022 | 3877.8501 | 3977 | 3860.05 | 3968.1499 | 3914.1963 | 3348123 | |
| 10 | 17-01-2022 | 3992.7 | 4043 | 3962.3 | 4019.1499 | 3964.5032 | 3442604 | |
| 11 | 18-01-2022 | 4033.95 | 4041.7 | 3980 | 3990.6001 | 3936.3411 | 2389041 | |
| 12 | 19-01-2022 | 4012 | 4012 | 3910.5 | 3914.6499 | 3868.209 | 3102539 | |
| 13 | 20-01-2022 | 3910 | 3920 | 3811 | 3826.55 | 3781.1543 | 6176776 | |
| 14 | 21-01-2022 | 3807 | 3851.55 | 3771.1001 | 3833.5 | 3788.0215 | 3112358 | |
| 15 | 24-01-2022 | 3840 | 3849.6499 | 3740.1001 | 3771.3501 | 3726.6094 | 3258414 | |
| 16 | 25-01-2022 | 3769.5 | 3809.3999 | 3722.2 | 3769.8999 | 3725.176 | 3330501 | |
| 17 | 27-01-2022 | 3731 | 3733.3999 | 3625.1001 | 3649.25 | 3605.9578 | 5718297 | |
| 18 | 28-01-2022 | 3646 | 3729.8 | 3646 | 3690.05 | 3646.2734 | 3143862 | |
| 19 | 31-01-2022 | 3749 | 3758 | 3721.3999 | 3736.25 | 3691.9255 | 2739393 | |
| 20 | 01-02-2022 | 3770 | 3808 | 3736.3999 | 3800.6499 | 3755.5615 | 2105169 | |
| 21 | 02-02-2022 | 3827.8999 | 3864 | 3800.6499 | 3856.2 | 3810.4521 | 1984212 | |
| 22 | 03-02-2022 | 3851 | 3882.5 | 3816.05 | 3824.6001 | 3779.2275 | 1960538 | |

Fig 2.3.2 : TCS-NS Dataset

2.4 EXISTING SYSTEM

Traditionally, two main approaches have been proposed for predicting the stock price of an organisation. Technical analysis method uses historical prices of stocks like closing and opening price etc. It is the study of the price moment and patterns of the security. By scrutinising a security's fast price action, primarily through charts and indicators, traders can forecast future price direction. The basic concept behind the technical analysis strategy is first to spot a strong market trend followed by a pullback in price. The retracement should only last a short period of time.

Once the market retracement pauses, the trend will resume and continue moving in the direction of the dominant trend. The second type of analysis is Qualitative, which is performed on the basis of external factors like company profile, market situation, political and economic factors, social media etc. Qualitative analysis is used to assess a company's investment possibilities. This data includes things like management quality, stakeholder satisfaction, ethics, brand value, and so on. Qualitative analysis deals with intangible and

inexact information that can be difficult to collect and measure. Understanding people and company cultures are central to qualitative analysis.

For example, Caginalp and Laurent performed a statistical test including eight kinds of three-day patterns and noted that the candlestick patterns have predictive power. Machine Learning techniques in the area have proved to improve efficiencies by 60-83% as compared to the past methods.

Disadvantages of existing systems:

- The accuracy would decrease when setting more levels of stock market movement.
- The average of prediction accuracies using Decision Tree as the classifier are 43.44%, 31.92%, 12.06% for “two levels”, “three levels”, and “five levels” respectively.
- These results indicate that the stock price is unpredictable when a traditional classifier is used.
- Time series data are difficult to manipulate and predict.
- These results indicate that the stock price is unpredictable when a traditional classifier is used.
- Time series data are difficult to manipulate and predict.

2.5 PROPOSED SYSTEM:

After evaluating the results from the existing methodologies, we are not able to predict the price of time series stock data for the data obtained from the datasets. Inorder to achieve this, we are using various algorithms such as LSTM(Long short Term Memory),SVM(Support Vector Machine),GRU(Gated Recurrent Unit) and create a hybrid model. ML process starts from a pre-processing data phase followed by feature selection based on data cleaning, classification of modelling performance evaluation. Hybrid Model is created by combining LSTM and GRU. We are comparing the various performance parameters of the algorithms and finally choosing the best one to improve the accuracy of the result. We are using the LSTM model to predict the future closing price of the stocks in time series pattern.

Our study is based on two major parts: The pre-processing phase, when we choose the relevant attributes, and the second one applies machine learning algorithm inorder to inorder to select the algorithm that gives better accuracy. Hybrid Model is the proposed approach

used to improve the accuracy of the result. Our proposed system is divided into several phases.

2.5.1 DATASET COLLECTION:

In our case, we use a dataset of stocks of historical data for about the last five years from different companies/organisations. These datasets have the values for different attributes. The attributes of the dataset represent the attributes such as open, close prices, high and low prices and volume of the stocks having the different values for each stock.

2.5.2 MANUAL EXPLORATION:

This step is very important in the development of Machine Learning algorithms. Because we analyse the data set, where we rank or label each Stock if the price is high or low. To give the algorithms the training dataset, we form the data set.

2.5.3 DATA PRE-PROCESSING:

Data preprocessing is an important step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model.

2.5.4 MODELLING:

This is the phase where we apply and test the chosen algorithms to find the best between them. Algorithms application in this step we find that the most efficient and used in this step, we find that the most efficient and used algorithm is LSTM(Long short Term Memory). Our approach is based on the application of the two algorithms on data sets of several sizes to validate the accuracy of each one and above all find the one that is more stable in training and surely gives high accuracy.

2.5.5 Advantages of proposed systems:

- The successful prediction will maximise the benefit of the customer.

- The time series data issue can be solved by focusing on segments instead of each data point, interesting patterns can be discovered and it becomes an easy task to query, understand and mine them.
- We will use stock data of five companies from the Huge Stock market dataset consisting of data ranging from 2017 to 2022 to train different machine learning algorithms.
- Hence we will compare the accuracy of different machine learning algorithm based models to validate the results on the prediction of closing price stock data.

2.5.6 Proposed system Architecture

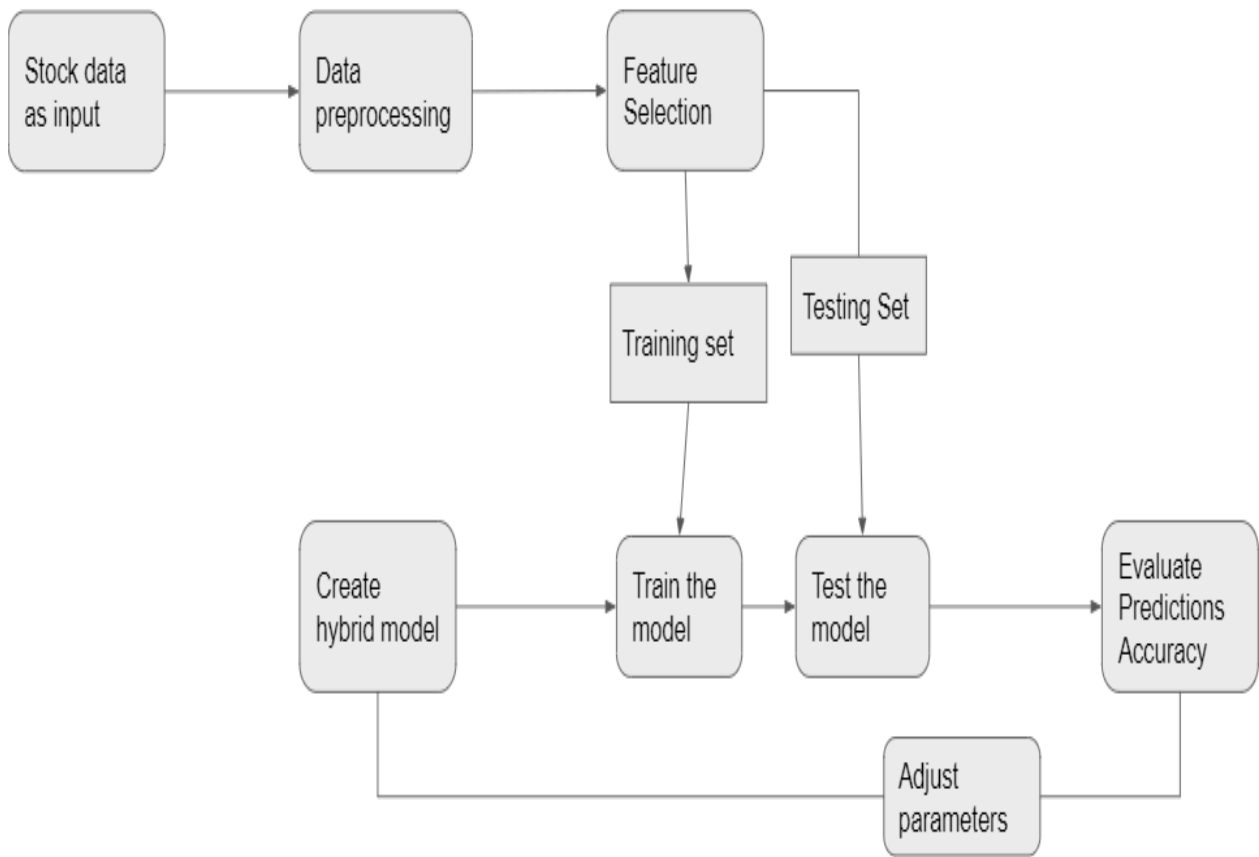


Fig 2.5.6 : Proposed System Architecture

3. REQUIREMENT ANALYSIS

3.1 INTRODUCTION

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. With the increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades in existing computer systems than technological advancements. The three primary activities involved in the analysis phase are as follows:

- Software Requirement specification
- Creating content diagram
- Performing a detailed analysis by drawing a flowchart

Requirement Analysis will identify and consider the risks related to how the technology will be integrated into the standard operating procedures. Requirements Analysis will also collect the functional and system requirements of the business process, the user requirements and the operational requirements. Hence this section deals with the software and hardware requirements with respect to our project.

3.2 SOFTWARE REQUIREMENT SPECIFICATION

3.2.1 Introduction

Requirement Determination is the process by which an analyst gains the knowledge of the organisation and applies it in selecting the right technology for an application. A Software requirement specification (SRS) is a complete description of the behaviour of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements.

In addition to use cases, the SRS also contains non-functional or supplementary requirements (Non-functional requirements are requirements which impose constraints on the design and implementation such as performance engineering requirements, quality standards, or design constraints).

3.2.2 Functional Requirements

A Functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

- Functional Requirements concerns the specific functions delivered by the system. So, functional requirements are statements of the services that the system must provide.
- The functional requirements of the system should be both complete and consistent.
- Completeness means that all the services required by the user should be defined.
- Consistency means that requirements should not have any contradictory definitions.
- The requirements are usually described in a fairly abstract way. However, functional system requirements describe the system function in details, its inputs and outputs, exceptions and so on.
- Take user id and password match it with corresponding file entries. If a match is found then continue else raise an error message.

3.3 Non-Functional Requirements

In systems engineering and requirements engineering a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. In general, functional requirements define what a system is supposed to do whereas non-functional requirements are often called qualities of a system. Non-functional Requirements may relate to emergent system properties such as reliability, response time and store occupancy or the selection of language, platform, implementation techniques and tools. They can be built on the basis of needs of the user budget constraints, organisation policies, etc.

3.3.1 USER REQUIREMENTS

In the user requirements we get the description of the users who use the system along with their characteristics and it specifies the product. It also describes briefly about the functional and data requirements of the project.

User requirements for this project are:

- Anaconda 3.7 (or)
- Jupyter

3.3.2 SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project regarding what the areas of strength and deficit are and how to tackle them.

- Operating System : windows
- Software used : Anaconda (or) jupyter

3.4 HARDWARE SPECIFICATIONS

The most common set of requirements defined by any operating system or software application is the physical computer resources also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. The following subsections discuss the various aspects of hardware requirements. Minimum hardware requirements are very dependent on the software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster Processor.

| | | |
|------------------|---|-------------------------|
| Operating system | : | Windows |
| RAM | : | 8GB |
| Hard disk | : | Greater than 16 GB |
| System Type | : | 64-bit Operating system |

3.5 CONTENT OF THE DIAGRAM

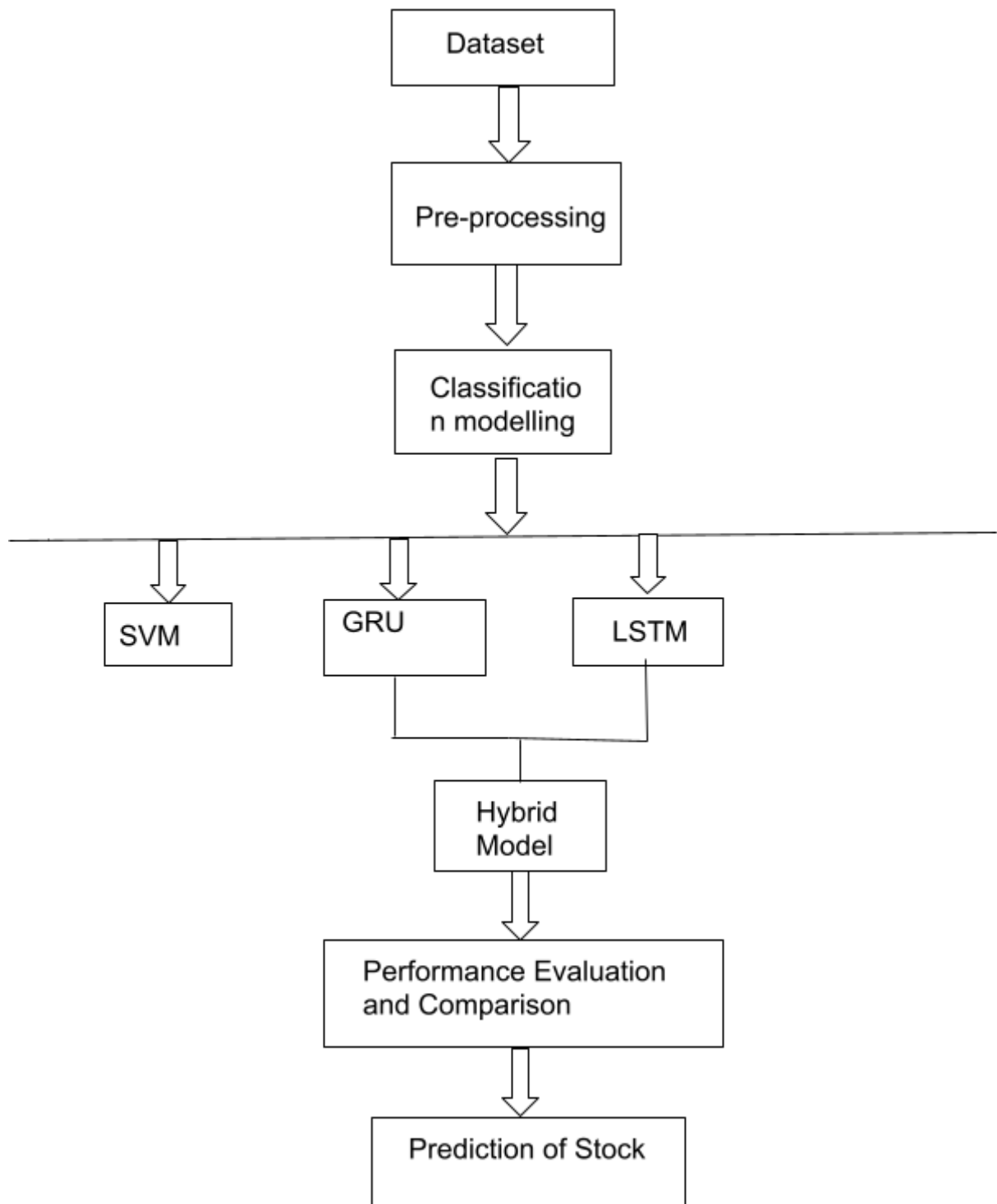


Fig 3.5: Content Of The Diagram

3.6 ALGORITHMS

3.6.1 LSTM

LST Memory is a sophisticated version of the recurrent neural networks (RNN) design that was created to represent chronological sequences and their long-range dependencies more precisely than traditional RNNs. Its main features are the internal design of an LSTM cell and the various modifications introduced into the LSTM structure, and a few applications of LSTMs that are in high demand. The article also provides an examination of LSTMs as well as GRUs. The tutorial ends with a list that outlines the drawbacks associated with the LSTM network and a description of the new models based on attention, which are swiftly replacing LSTMs in the real world.

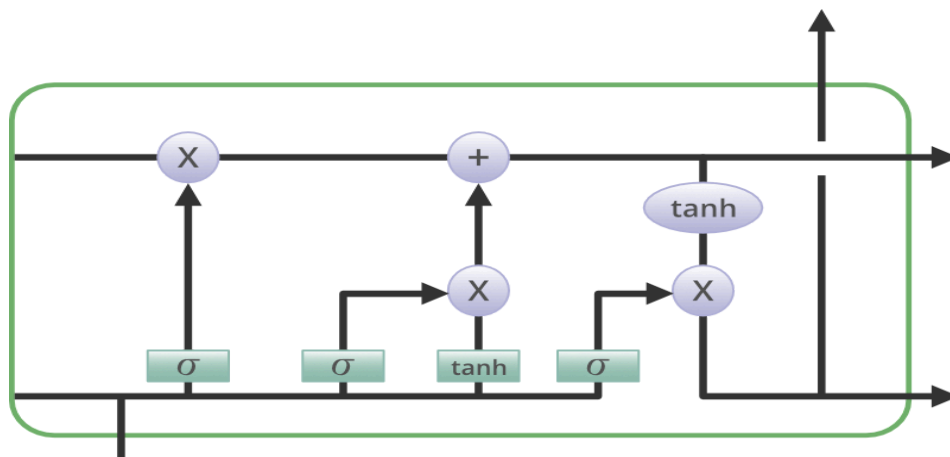


Fig 3.6.1.1 LSTM Model

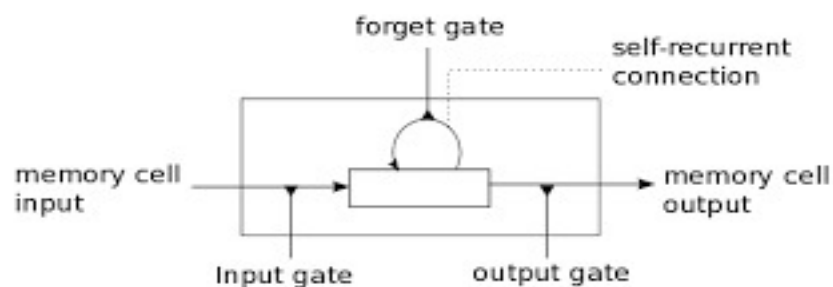


Fig 3.6.1.2 LSTM Architecture

LSTMs work in a 3-step process.

- Step 1: Decide How Much Past Data It Should Remember. ...
- Step 2: Decide How Much This Unit Adds to the Current State. ...
- Step 3: Decide What Part of the Current Cell State Makes It to the Output.

3.6.2 Support vector machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

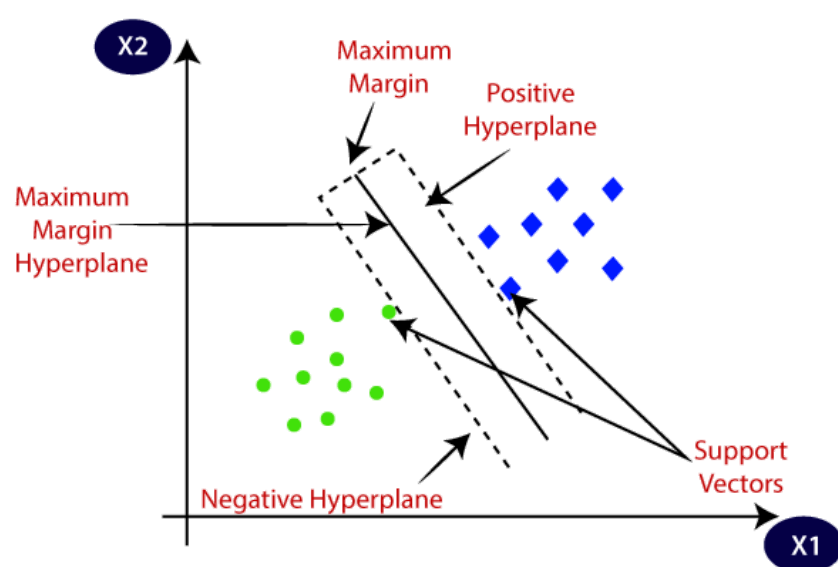


Fig 3.6.2.1 :SVM Model

How does SVM works?

The working of the SVM algorithm can be understood by using an example Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x_1 and x_2 . We want a classifier that can classify the pair (x_1, x_2) of coordinates in either green or blue. Consider the below image.

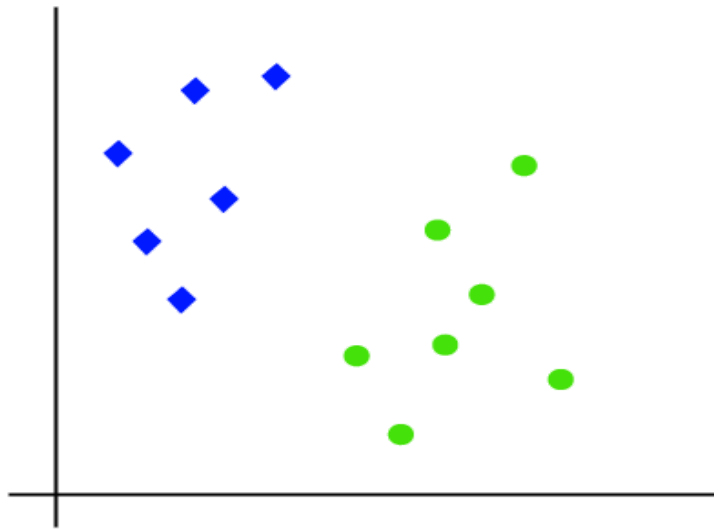


Fig 3.6.2.2 Datasets

So, as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes Consider the below image:

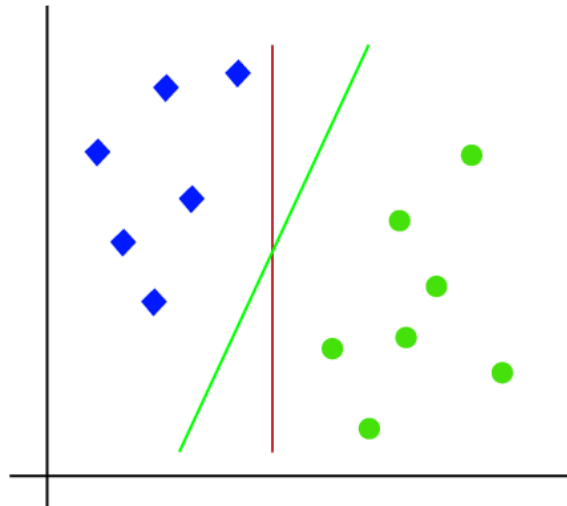


Fig 3.6.2.3 2D-space

Hence, the SVM algorithm helps to find the best line or decision boundary, this best boundary or region called a hyperplane, SVM algorithm finds the closest post of the lines from both the classes. These posts are called support vectors. The distance between the vectors and the hyperplane is called margin. And the goal of SVM is to maximise this margin. The hyperplane with maximum margin is called the optimal hyperplane.

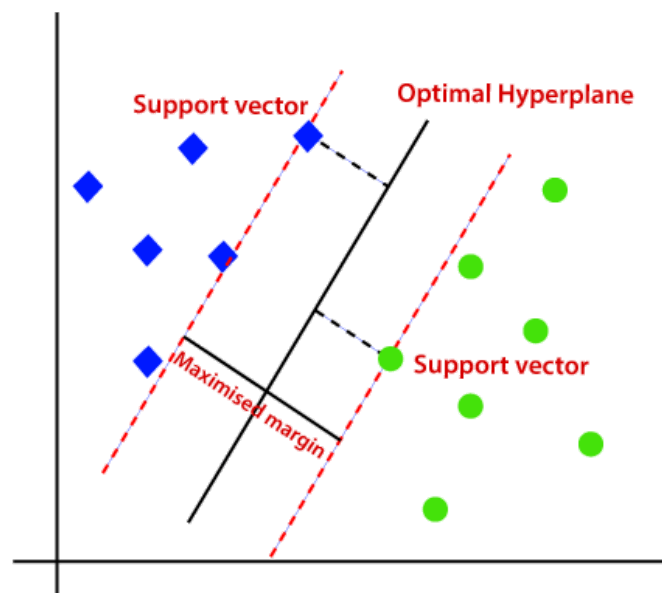


Fig 3.6.2.3 SVM model with hyperplane

3.6.3 GRU (Gated Recurrent Unit):

Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that was introduced by Cho et al. in 2014 as a simpler alternative to Long Short-Term Memory (LSTM) networks. Like LSTM, GRU can process sequential data such as text, speech, and time-series data.

The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state. The output of the GRU is calculated based on the updated hidden state.

It is calculated as follows:

- Multiply the input x_t with a weight W and $h_{(t-1)}$ with a weight U .
- Calculate the Hadamard (element-wise) product between the reset gate r_t and $U h_{(t-1)}$
- Sum up the results of step 1 and 2.
- Apply the nonlinear activation function \tanh .

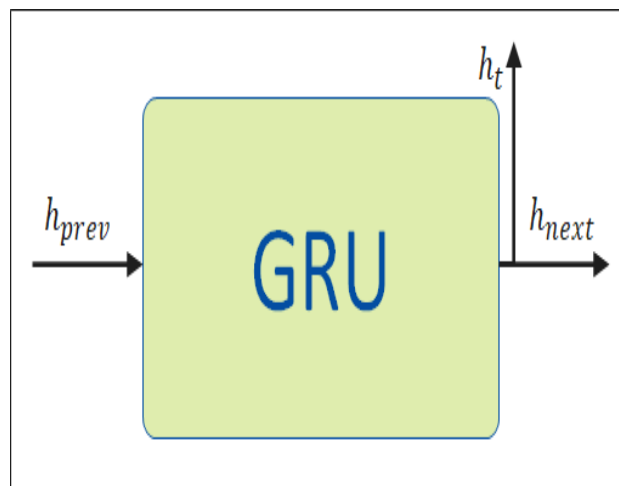


Fig 3.6.3 GRU Model

4. DESIGN

4.1 INTRODUCTION

It is a process of planning the new or modified system. Analysis specifies what a new or modified system does. Design specifies how to accomplish the same. Design is essentially a bridge between requirement specification and the final solution satisfying the requirements. Design of a system is essentially a blueprint or a solution for the system. The design process for a software system has two levels. At first level the focus is on depending on which modules are needed for the system, the specification of these modules and how the modules should be interconnected. This is what is called system designing or top level design. In the second level, the internal design of the modules, or how the specification of the module can be satisfied is described upon. This design level is often called detailed design or logic design. The first level produces system design, which defines the components needed for the system, and how the components interact with each other. Its focus is on depending on which modules are needed for the system, the specification of these modules and how the module should be interconnected.

4.2 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standard language for specifying, visualising, constructing, and documenting the artifacts of software systems. UML is different from the other common programming languages like C++, Java, and COBOL etc. It is designed to enable users to develop an expressive and ready to use visual modelling language. In addition, it supports high level development concepts such as frameworks, patterns and collaborations. UML can be described as a general-purpose visual modelling language to visualise, specify, construct and document software systems. But it is not limited within this boundary. To be more precise, UML is a pictorial language used to make software blueprints. UML has a direct relation with object-oriented analysis and design. After some standardisation UML has become an OMG (Object Management Group) standard.

The UML is a language for

- Visualising
- Specifying
- Constructing
- Documenting

Goals of UML:

1. Provide users with a ready-to-use, expressive visual modelling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialisation mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of the tools market.
6. Support higher-level development concepts such as collaborations, frameworks and components.
7. The system can be software or non-software. So, it must be clear that UML is not just a development method.

4.2.1 Basic Building Blocks of UML:

The basic building blocks in UML are **things** and **relationships**. These are combined in different ways following different rules to create different types of diagrams. In UML there are eight types of diagrams, below is a list and brief description of them. The more in-depth descriptions in the document will focus on the first five diagrams in the list, which can be seen as the most general, sometimes also referred to as the UML core diagrams.

Components of the UML:

The UML consists of a number of graphical elements that combine to form diagrams. Because it's a language, the UML has rules for combining these elements. The purpose of the diagrams is to present multiple views of the system, and this set of multiple views is called a Model.

A UML Model of a system is something like a scale model of a building. The UML model describes what a system is supposed to do. It doesn't tell how to implement the system.

These are the artefacts of a software-intensive system. The abbreviation for UML is Unified Modelling Language and is being brought of a designed to make sure that the existing ER Diagrams which do not serve the purpose will be replaced by UML Diagrams in this language as its own set of Diagrams.

Some of the Diagrams that help for the Diagrammatic Approach for the Object-Oriented Software Engineering are:

- Class Diagrams
- Use Case Diagrams
- Sequence Diagrams
- State chart Diagrams
- Activity Diagrams

Using the above mentioned diagrams we can show the entire system regarding the working of the system or the flow of control and sequence of flow, the state of the system and the activities involved in the system.

4.2.2.CLASS DIAGRAM

A Class is a category or group of things that has similar attributes and common behaviour. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle area contains the attributes and the lowest areas show the operations. Class diagrams provide the representation that developers work from.

The classes in a Class diagram represent both the main objects, interactions in the application and the classes to be programmed.

In the diagram, classes are represented with boxes which contain three parts:

1. The top part contains the name of the class. It is printed in bold and cantered, and the first letter is capitalised.
2. The middle part contains the attributes of the class. They are left-aligned and the first letter is lowercase.
3. The bottom part contains the methods the class can execute. They are also left-aligned and the first letter is lowercase. In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those described objects. With detailed

modelling, the classes of the conceptual design are often split into a number of subclasses.

Visibility:

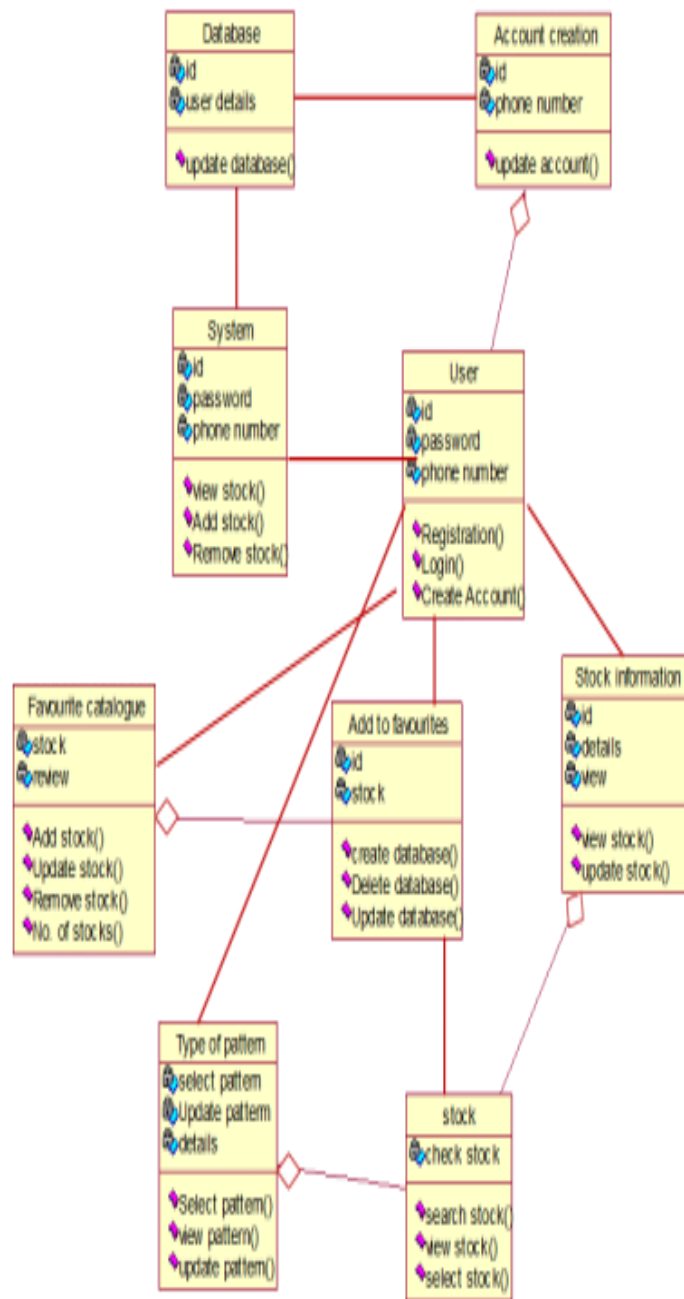


Fig 4.2.2 Class Diagram

To specify the visibility of a class member (i.e., any attribute or method), the sanitation's must be placed before the member's name:

+ Public

/ Derived

Protected

- Private

~ Package

Relationships:

A relationship is a general term covering the specific types of logical connections found on class and object diagrams. UML shows the following relationships:

1. Links: A Link is the basic relationship among objects.

2. Association: An association represents a family of links. An association can be named and the ends of an association can be adorned with role names, ownership indicators, Multiplicity, visibility, and other properties.

3. Aggregation: Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association can. In UML, it is graphically represented as a hollow diamond shape on the containing class with a single line that connects it to the contained class. The aggregate is semantically an extended object that is treated as a unit in many operations, although physically it is made of several lesser objects.

4. Composition: Composition is a stronger variant of the "has a" association relationship; composition is more specific than aggregation. Composition usually has a strong lifecycle dependency between instances of the container class and instances of the contained classes. The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained class(es) to the containing class.

5. Generalisation: The Generalization relationship ("is a") indicates that one of the two related classes (the subclass) is considered to be a specialised form of the other (the super type) and the super class is considered a 'Generalisation' of the subclass. The UML graphical representation of a Generalisation is a hollow triangle shape on the superclass end of the line

(or tree of lines) that connects it to one or more subtypes. The generalisation relationship is also known as the inheritance or "is a" relationship. Generalisation can only be shown on class diagrams and on Use case diagrams.

6. Realisation: In UML modelling, a realisation relationship is a relationship between those two model elements, in which one model element realises (implements or executes) the behaviour that the other model element specifies. The UML graphical representation of a Realisation is a hollow triangle shape on the interface end of the *dashed* line (or tree of lines) that connects it to one or more implementers. A plain arrowhead is used on the interface end of the dashed line that connects it to its users. Realisations can only be shown in class or component.

7. Dependency: Dependency is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class.

8. Multiplicity: This association relationship indicates that (at least) one of the two related classes makes reference to the other. The UML representation of an association is a line with an optional arrowhead indicating the *role* of the object(s) in the relationship and an optional notation at each end indicating the *multiplicity* of instances of that entity (the number of objects that participate in the association).

4.2.3. USE-CASE DIAGRAM

A Use-Case is a description of systems behaviour from an understanding point. For system developers this is a valuable tool: it's a tried-and-true technique for gathering system requirements from a user's point of view. That is important if the goal is to build a system that real people can use. A little stick figure is used to identify an actor; the ellipse represents use-case.

Here, we have two actors namely developer and user. The developer is involved in all the steps whereas the user is only interested in visualising the result, testing the model and displaying the result.

Use cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse

Actors: An actor is a person, Organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

Associations: Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modelled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead

is often used to indicate the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.

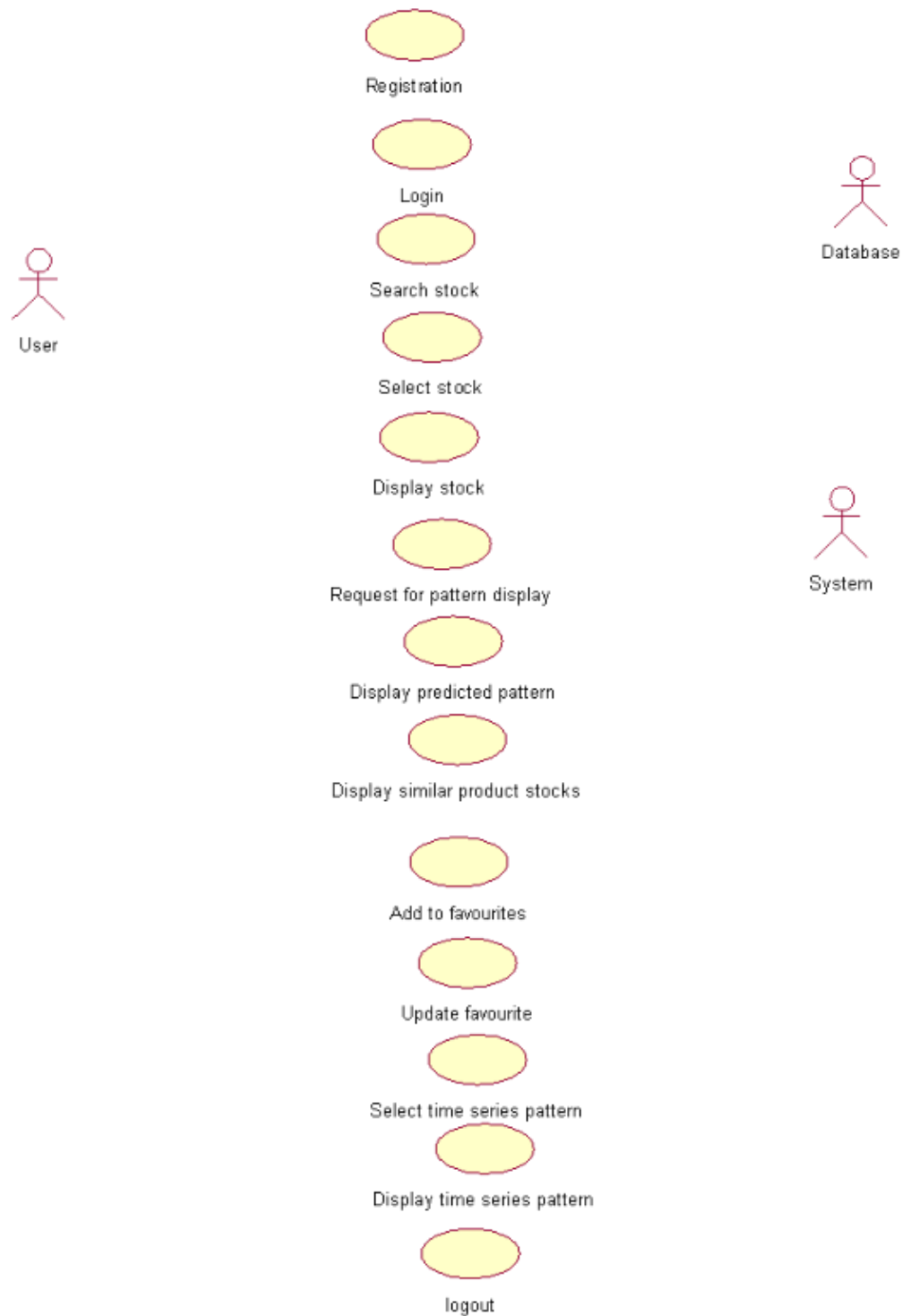


Fig 4.2.3.1: Use Case Diagram

Use Case Relationships:

Three relationships among use cases are used often in practice.

1.Include:

In one form of interaction, a given use case may include another. Include is a directed relationship between two use cases, implying that the behaviour of the included use case is inserted into the behaviour of the including use case.

2. Extend:

In another form of interaction, a given use case (the extension) may extend another. This relationship indicates that the behaviour of the extension use case may be inserted in the extended use case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label `||<<extend>>||`.

3.Generalization :

In the third form of relationship among use cases, generalisation/specialisation relationship exists. A given use case may have common behaviours, constraints and assumptions to the general use case. The notation is a solid line ending in a hollow triangle drawn from the specialised to the more general use case.

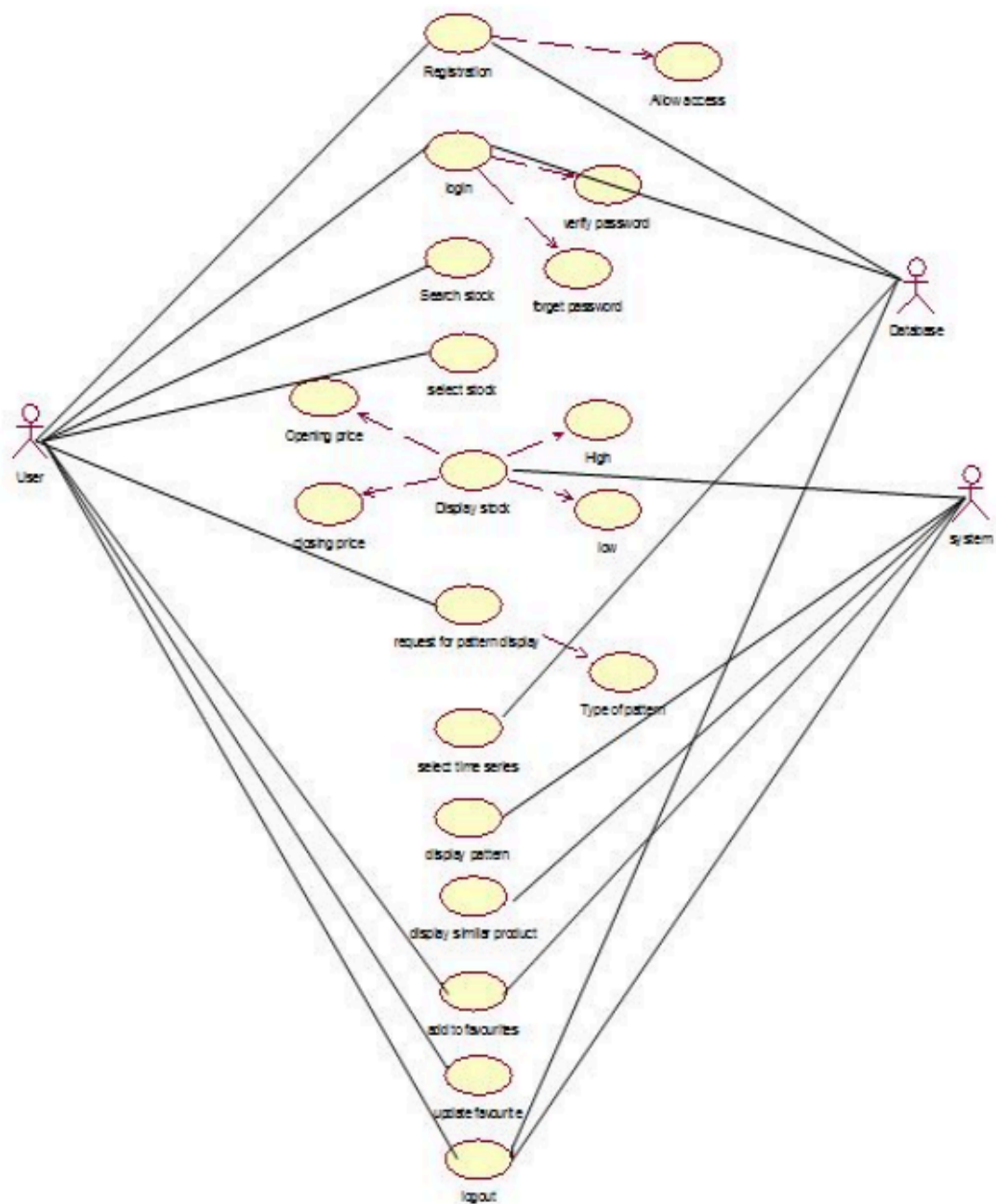


Fig 4.2.3.2 Use Case With Relationships

4.2.4.SEQUENCE DIAGRAMS

In a functioning system an object interacts with one another and these interactions occur over time. The UML Sequence Diagram shows the time-based dynamics of the interaction. The sequence diagrams consist of objects represented in the use of always named rectangles (If the name underlined), messages represented as solid line arrows and time represented as a vertical progression.

Sequence diagrams describe interactions among classes in terms of an exchange of messages overtime. It is an interaction diagram that emphasises the time ordering of messages. A narrow rectangle on an objects lifetime represents activation- an exchange of one of that object's operation. Messages are arrows that connect one.

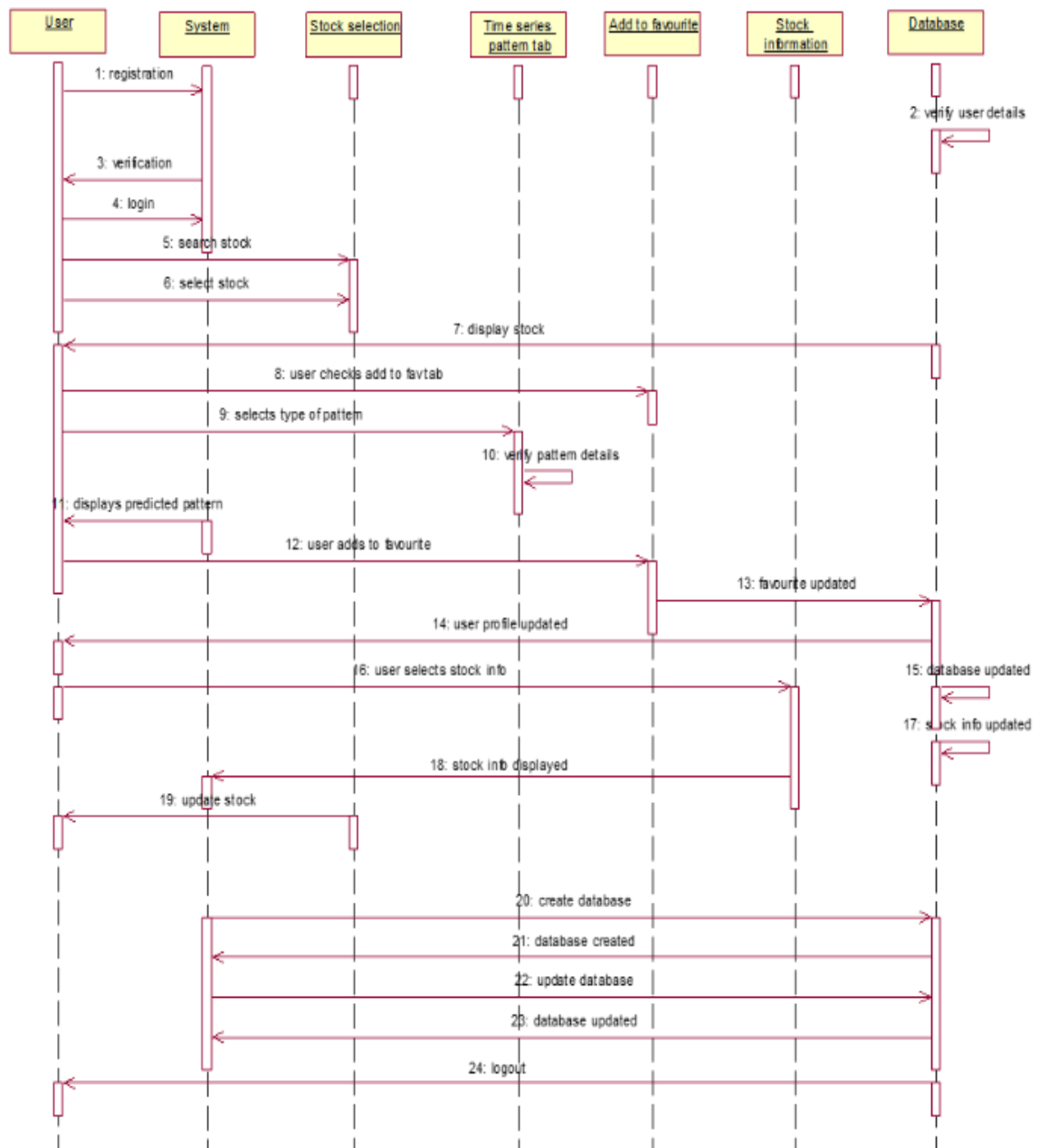


Fig 4.2.4 Sequence Diagram

4.2.5.ACTIVITY DIAGRAM:

Activity Diagrams to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on the condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. UML models basically three types of diagrams, namely, structure diagrams, interaction diagrams, and behaviour diagrams.

An activity diagram is a behavioural diagram i.e. it depicts the behaviour of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system. An activity diagram is very similar to a flowchart.

Activity is a particular operation of the system. Activity diagrams are not only used for visualising the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as –

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

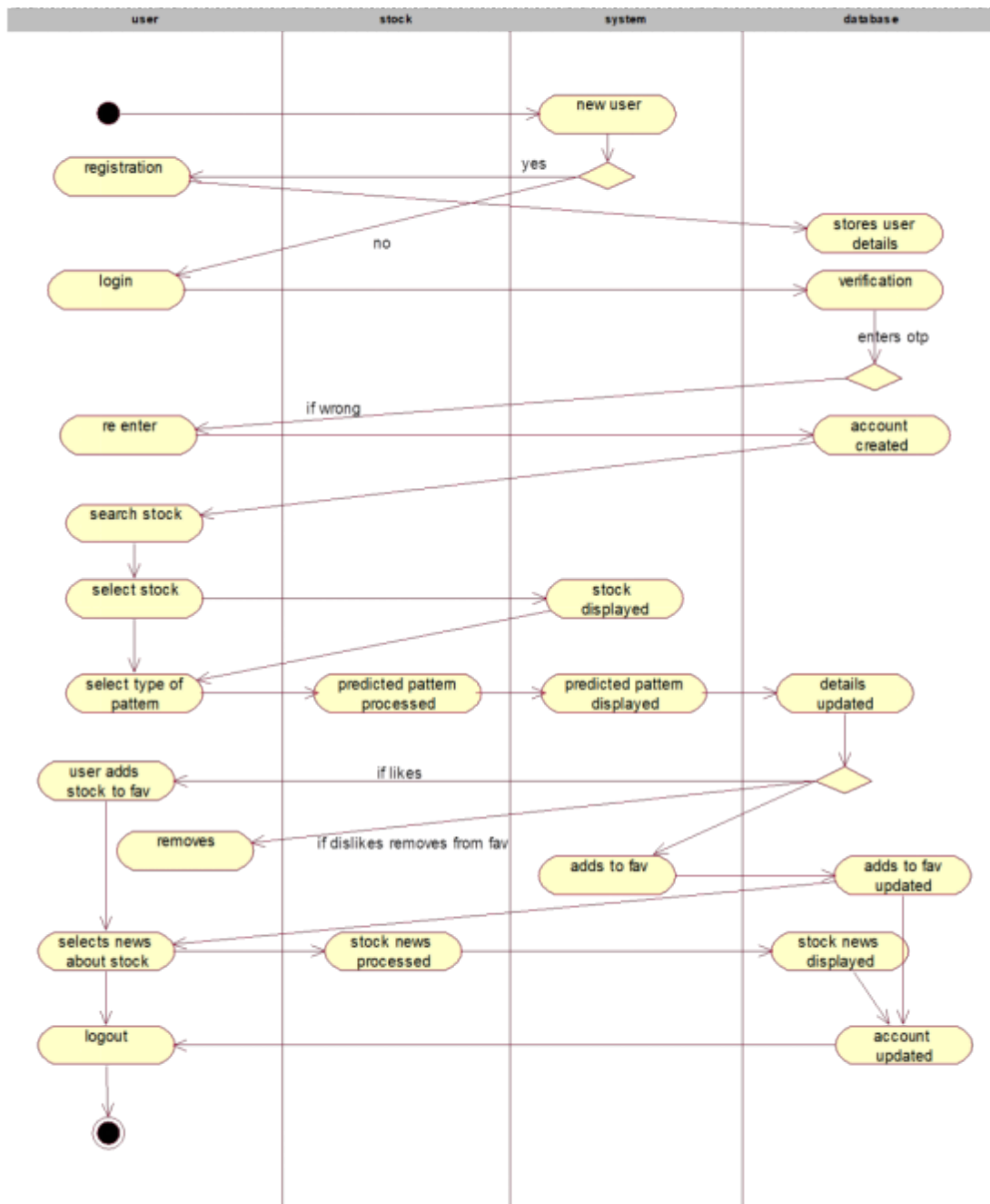


Fig 4.2.5 :Activity Diagram

4.2.6. STATE CHART DIAGRAM:

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system. A State Chart diagram describes a state machine.

Now to clarify it, a state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. As State Chart diagram defines states it is used to model lifetime of an object

Main Usage of State Chart Diagram:

So the main usages can be described as: To model object states of a system. To model a reactive system. Reactive system consists of reactive objects. To identify events responsible for state changes. Forward and reverse engineering.

Purpose:

Following are the main purposes of using State Chart diagrams:

- To model dynamic aspect of a system.
- To model life time of a reactive system
- To describe different states of an object during its life time. Define a state machine to model states of an object.
-

Before drawing a State Chart diagram we must have clarified the following points:

- Identify important objects to be analysed.
- Identify the states.
- Identify the events.

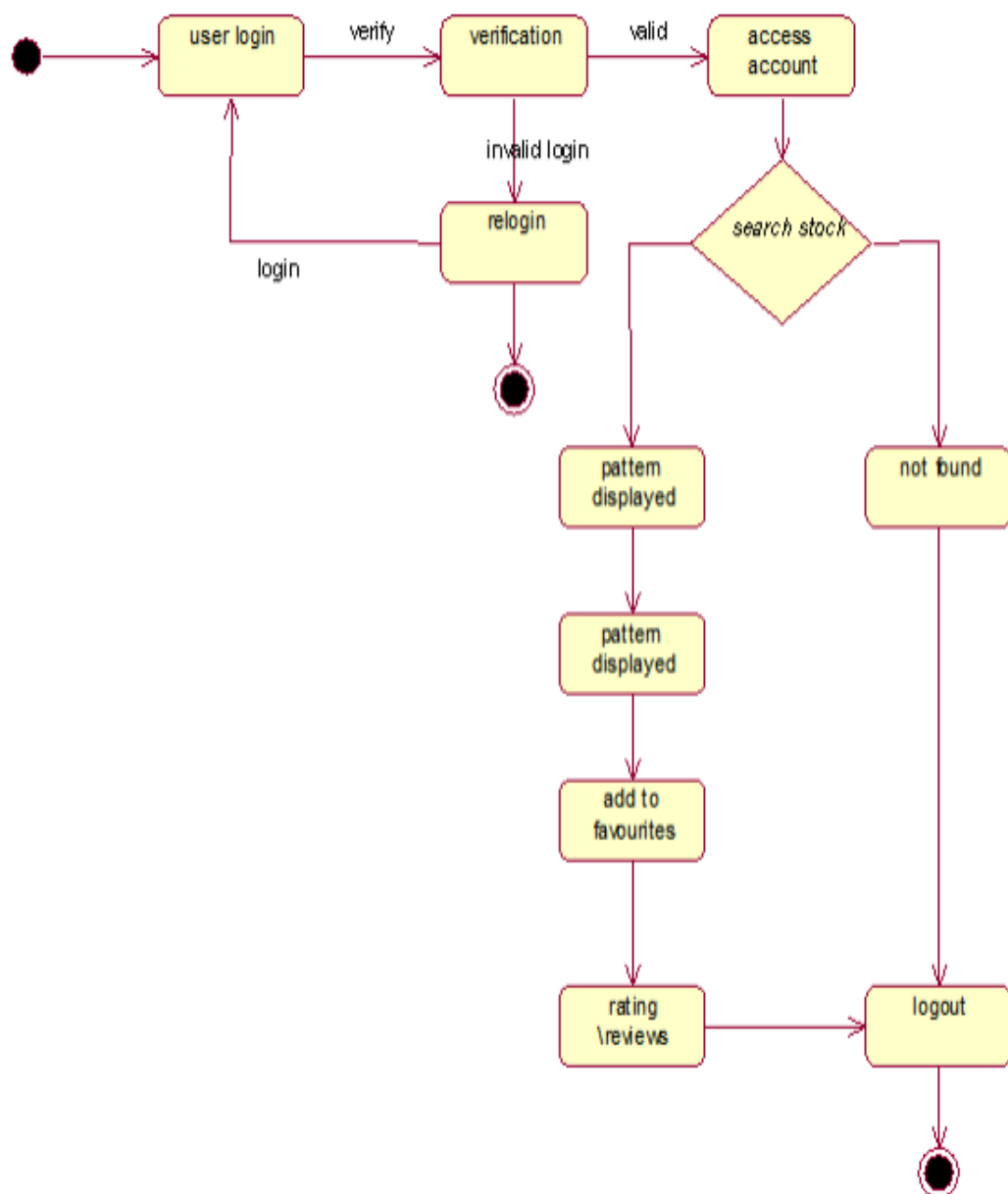


Fig 4.2.6 State Chart Diagram

5. IMPLEMENTATION AND RESULTS

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.1 PROJECT IMPLEMENTATION STEPS

- Collecting the data and performing preprocessing of data.
- Dividing the dataset into training set and testing set
- Applying the hybrid model using LSTM and GRU on dataset
- Making Predictions

5.2 IMPLEMENTATION

STEP-1: Data Preparation

Setting up Google Colab: Google Colab is an online managed Jupyter Notebook environment where you can train machine learning models on GPU. The free plan of Google Colab allows you to train the machine learning model for up to 12 hrs before the runtime disconnects. You have to select runtime as GPU before launching the Jupyter notebook as shown below

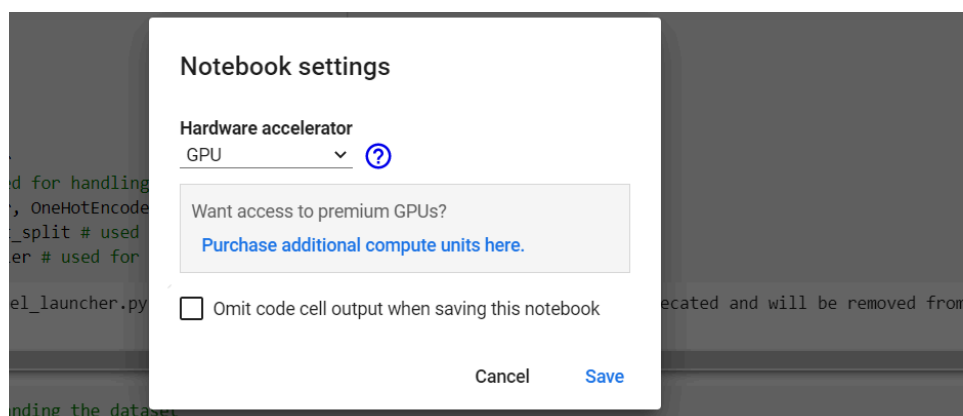


Fig 5.1 Goolge collab

We have uploaded the dataset on our google drive but before we can use it in Colab we have to mount our google drive directory onto our runtime environment as shown below. This command will generate a URL on which you need to click, authenticate your Google drive account and copy the authorization key over here and press enter.\

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

5.2.1 Data Preprocessing

Preprocessing is the step to clean the dataset

```
[ ] df.isnull().sum()

Date                0
Symbol              0
Series              0
Prev Close          0
Open                0
High                0
Low                 0
Last                0
Close               0
VWAP                0
Volume              0
Turnover            0
Trades              1683
Deliverable Volume  0
%Deliverble         0
dtype: int64
```

```
[4] df.isna().any()
```

| | |
|--------------------|-------|
| Date | False |
| Symbol | False |
| Series | False |
| Prev Close | False |
| Open | False |
| High | False |
| Low | False |
| Last | False |
| Close | False |
| VWAP | False |
| Volume | False |
| Turnover | False |
| Trades | True |
| Deliverable Volume | False |
| %Deliverble | False |
| dtype: | bool |

```
▶ df.dropna(inplace=True)  
df.isna().any()
```

| | | |
|---|--------------------|-------|
| ▶ | Date | False |
| | Symbol | False |
| | Series | False |
| | Prev Close | False |
| | Open | False |
| | High | False |
| | Low | False |
| | Last | False |
| | Close | False |
| | VWAP | False |
| | Volume | False |
| | Turnover | False |
| | Trades | False |
| | Deliverable Volume | False |
| | %Deliverble | False |
| | dtype: | bool |

Fig 5.1.2 Data Preprocessing

5.2.3 Classification Modelling:

LSTM

Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) that is specifically designed to handle sequential data, such as time series, speech, and text.

LSTM networks are capable of learning long-term dependencies in sequential data, which makes them well suited for tasks such as language translation, speech recognition, and time series forecasting.

LSTM has a chain structure that contains four neural networks and different memory blocks called **cells**.

GRU:

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014.

The GRU operates using a reset gate and an update gate. The reset gate sits between the previous activation and the next candidate activation to forget previous state, and the update gate decides how much of the candidate activation to use in updating the cell state.

GRUs have been shown to exhibit better performance on smaller datasets.

Hybrid Model using LSTM and GRU:

#installing modules

```
import pandas as pd
```

```
import numpy as np
```

```
import math
```

```
import datetime as dt
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score, r2_score
```

```
from sklearn.metrics import mean_poisson_deviance, mean_gamma_deviance, accuracy_score
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM, GRU

from itertools import cycle
```

```
# ! pip install plotly

import plotly.graph_objects as go

import plotly.express as px
```

#reading the dataset

```
df = pd.read_csv("TCS (1).csv")

df.head(20)
```

#Applying scaling method using MinMaxscaler

```
close_stock = closedf.copy()

scaler=MinMaxScaler(feature_range=(0,1))

closedf=scaler.fit_transform(np.array(closedf).reshape(-1,1))

print(closedf.shape)
```

#splitting the dataset into training and testing set

```
training_size=int(len(closedf)*0.75)

test_size=len(closedf)-training_size

train_data,test_data=closedf[0:training_size:],closedf[training_size:len(closedf),:1]

print("train_data: ", train_data.shape)

print("test_data: ", test_data.shape)

X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)
```

```
X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)
```

```
print("X_train: ", X_train.shape)
```

```
print("X_test: ", X_test.shape)
```

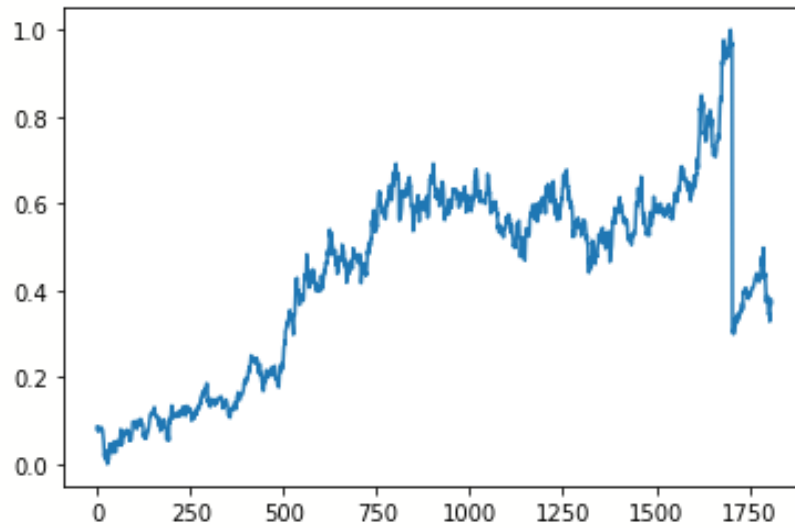


Fig 5.2.3.1 Plot for Training Dataset

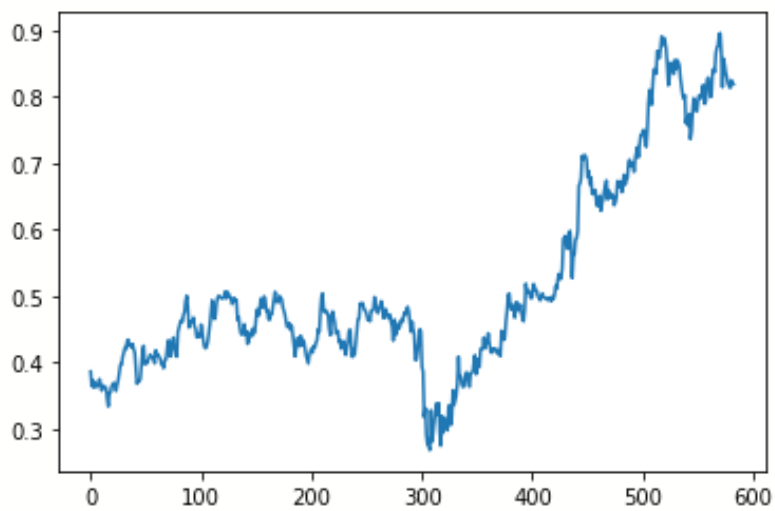


Fig 5.2.3.2 Plot for Testing Dataset

#applying the SVM model

```
from sklearn import svm

clf=svm.SVR()

clf.fit(X_train,y_train)

accuracy=clf.score(X_test,y_test)

print (accuracy)

from sklearn.svm import SVR
```

#making predictions

```
train_predict=svr_rbf.predict(X_train)

test_predict=svr_rbf.predict(X_test)

train_predict = train_predict.reshape(-1,1)

test_predict = test_predict.reshape(-1,1)

print("Train data prediction:", train_predict.shape)

print("Test data prediction:", test_predict.shape)

train_predict = scaler.inverse_transform(train_predict)

test_predict = scaler.inverse_transform(test_predict)

original_ytrain = scaler.inverse_transform(y_train.reshape(-1,1))

original_ytest = scaler.inverse_transform(y_test.reshape(-1,1))

import math

from sklearn.metrics import mean_squared_error

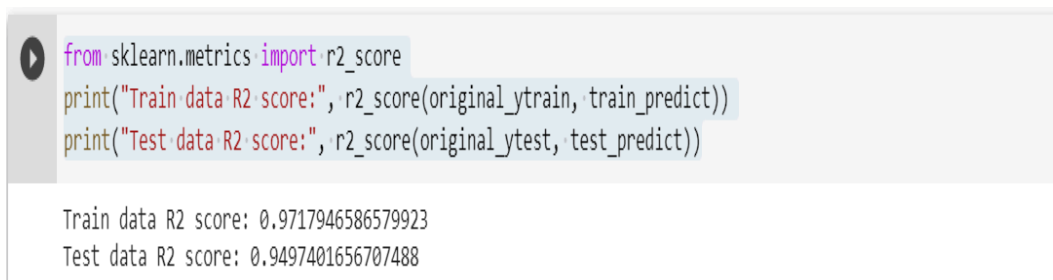
from sklearn.metrics import mean_absolute_error
```

#errors of testing and training set

```
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))  
  
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))  
  
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))  
  
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))  
  
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))  
  
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))
```

#R2 score

```
from sklearn.metrics import r2_score  
  
print("Train data R2 score:", r2_score(original_ytrain, train_predict))  
  
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```



```
from sklearn.metrics import r2_score  
print("Train data R2 score:", r2_score(original_ytrain, train_predict))  
print("Test data R2 score:", r2_score(original_ytest, test_predict))
```

Train data R2 score: 0.9717946586579923
Test data R2 score: 0.9497401656707488

Fig 5.2.3.3:R2 Score Error

#applying the LSTM model

```
from tensorflow.keras.models import Sequential  
  
from tensorflow.keras.layers import Dense  
  
from tensorflow.keras.layers import LSTM  
  
model=Sequential()  
  
model.add(LSTM(50,return_sequences=True,input_shape=(30,1)))
```

```

model.add(LSTM(50,return_sequences=True))

model.add(LSTM(50))

model.add(Dense(1))

model.compile(loss='mean_squared_error',optimizer='adam',metrics=['accuracy'])

model.summary()

import tensorflow as tf

tf.keras.layers.Flatten()

model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=5,batch_size=64,verbose=1)

model.save("keras.h5")

```

#making predictions

```

train_predict=model.predict(X_train)

test_predict=model.predict(X_test)

plt.plot(train_predict)

plt.plot(test_predict)

```

#errors of testing and training set

```

import math

from sklearn.metrics import mean_squared_error

c=math.sqrt(mean_squared_error(y_train,train_predict))

from sklearn.metrics import mean_absolute_percentage_error

from sklearn.metrics import mean_absolute_error

mean_absolute_percentage_error(y_train, train_predict)

## Test Data RMSE

d=math.sqrt(mean_squared_error(ytest,test_predict))

```

```

print(d)

print("Test data RMSE: ", math.sqrt(mean_squared_error(ytest,test_predict)))

print("Test data MAE: ", mean_absolute_error(ytest,test_predict))

```

```

[ ] ### Test Data RMSE
    d=math.sqrt(mean_squared_error(ytest,test_predict))
    print(d)
    print("Test data RMSE: ", math.sqrt(mean_squared_error(ytest,test_predict)))
    print("Test data MAE: ", mean_absolute_error(ytest,test_predict))

```

```

0.06334675894263943
Test data RMSE:  0.06334675894263943
Test data MAE:  0.03729096848281378

```

Fig 5.2.3.4 Error for testing dataset

#applying the hybrid model

```

tf.keras.backend.clear_session()

model=Sequential()

model.add(LSTM(32,return_sequences=True,input_shape=(time_step,1)))

model.add(LSTM(32,return_sequences=True))

model.add(GRU(32,return_sequences=True))

model.add(GRU(32))

model.add(Dense(1))

model.compile(loss='mean_squared_error',optimizer='adam')

model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=100,batch_size=5,verbose=
1)

model.summary()

```

#making predictions

```
train_predict=model.predict(X_train)
```

```
test_predict=model.predict(X_test)
```

```
train_predict.shape, test_predict.shape
```

#errors of testing and training set

```
✓ 0s # Evaluation metrics RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-----")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))

Train data RMSE:  55.26909835350747
Train data MSE:   3054.6732328096814
Test data MAE:    30.694323157526917
-----
Test data RMSE:   49.70938990174806
Test data MSE:    2471.023444404012
Test data MAE:    37.981083833619216
```

```
✓ 0s [26] print("Train data R2 score:", r2_score(original_ytrain, train_predict))
      print("Test data R2 score:", r2_score(original_ytest, test_predict))

Train data R2 score: 0.9911759268398272
Test data R2 score: 0.9850663383351148
```

Fig 5.2.3.5 Error for Training and Testing Dataset

```
df6 = pd.DataFrame({'Actual': original_ytrain.flatten(), 'Predicted' : train_predict.flatten()})
print(df6)
```


Plots of the hybrid model:

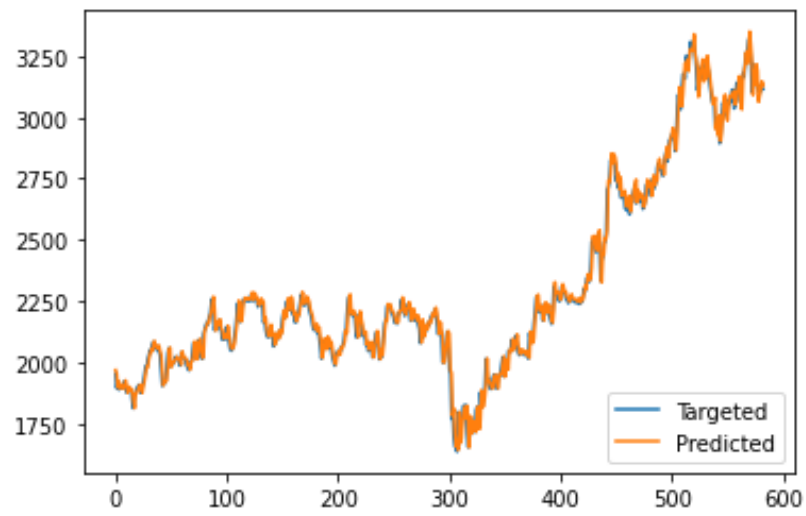


Fig 5.2.3.6: Prediction for Training Data

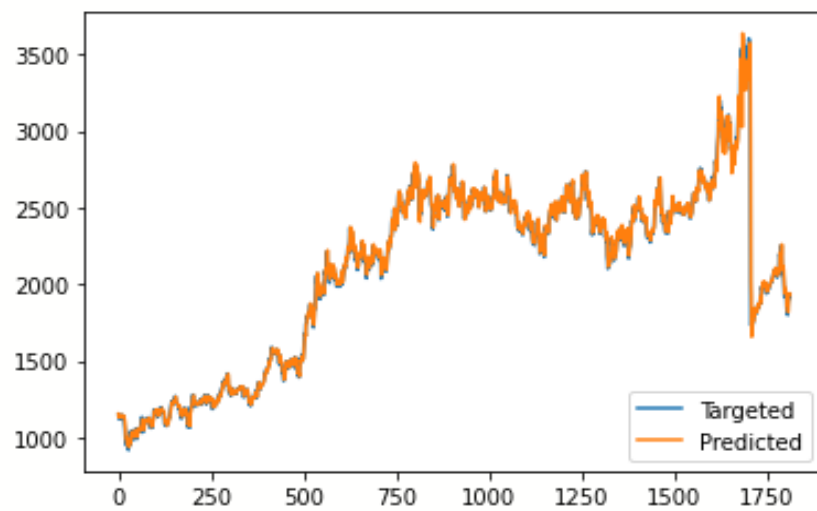


Fig 5.2.3.7 Prediction for Testing Dataset

6. TESTING AND VALIDATION

6.1 INTRODUCTION

Software testing is critical element of software quality assurance and represents the ultimate review of specification design and coding. Software testing is one of the broader topics and often referred to as verification to all the activities to all the activities that endure the software built is traceable to use requirements. Software testing can be taken as one of the errors in the developed system. Testing is the process of executing a program with the intend of finding the errors.

6.1.1 Scope

A primary purpose for testing is to detect software failures so that defects may be uncovered and corrected. This is a non-trivial pursuit. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

6.1.2 Defects and Failures

Not all software defects are caused by coding errors. One common source of expensive defects is caused by requirements gaps, eg, unrecognized requirements that result in errors of omission by the program designer. A common source of requirements gaps is non-functional requirements such as testability, scalability, maintainability, usability, performance, and security. Software faults occur through the following processes. A programmer makes an error (mistake), which results in a defect (fault, bug) in the software source code. If this defect is executed, in certain situations the system will produce wrong results, causing a

failure. Not all defects will necessarily result in failures. For example, defects in dead code will never result in failures. A defect can turn into a failure when the environment is changed. Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or interacting with different software. A single defect may result in a wide range of failure symptoms.

6.1.3 Compatibility

A frequent cause of software failure is compatibility with another application, a new operating system, or, increasingly, web browser version. In the case of lack of backward compatibility, this can occur because the programmers have only considered coding their programs for, or testing the software upon, "the latest version of this-or-that operating system. The unintended consequence of this fact is that: their latest work might not be fully compatible with earlier mixtures of software/hardware, or it might not be fully compatible with another important operating system. In any case, these differences, whatever they might be, may have resulted in software failures, as witnessed by some significant population of computer users.

6.1.4 Input Combinations and Preconditions

A very fundamental problem with software testing is that testing under all combinations of inputs and preconditions is not feasible, even with a simple product. This means that the number of defects in a software product can be very large and defects that occur infrequently are difficult to find in testing. More significantly, nonfunctional dimensions of quality (how it is supposed to be versus what it is supposed to do) can be highly subjective; something that constitutes sufficient value to one person may be intolerable to another.

6.1.5 Static Vs. Dynamic Testing

There are many approaches to software testing. Reviews, walkthroughs or inspections are considered as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. The former can be, omitted, whereas the latter takes place when programs begin to be used for the first time - which is normally, considered

the beginning of the testing stage. This may begin before the program is 100% complete in order to test particular sections of code. For example, Spread sheet programs are, by their very nature, tested to a large extent "on the fly" during the build process as the result of some calculation or text manipulation is shown interactively immediately after each formula is entered.

6.1.6. Software Verification and Validation

Software testing is used in association with verification and validation

Verification: Have we built the software right (ie, does it match the specification?) It is process based

Validation: Have we built the right software (ie, is this what the customer wants?) It is product based.

6.2. DESIGN OF TEST CASES AND SCENARIOS:

| Test No. | Test Case | Expected Output | Actual output | Result |
|----------|----------------------------|----------------------|-----------------------|--------|
| 1 | Input dataset | Display the dataset | Dataset is displayed | pass |
| 2 | pre processing the dataset | Remove the Noise | Noise is removed | pass |
| 3 | Algorithm Output | Display the accuracy | Accuracy is displayed | pass |

| | | | | |
|---|---|--------------------|-----------------|------|
| 4 | Different types of Errors Display | Displays the Error | Error displayed | pass |
|---|---|--------------------|-----------------|------|

Test case1:

| df.head(20) | | | | | | | | | | | | | | | |
|-------------|------------|--------|--------|------------|---------|--------|---------|---------|---------|---------|----------|--------------|--------|--------------------|-------------|
| | Date | Symbol | Series | Prev Close | Open | High | Low | Last | Close | VWAP | Volume | Turnover | Trades | Deliverable Volume | %Deliverble |
| 0 | 2004-08-25 | TCS | EQ | 850.00 | 1198.70 | 1198.7 | 979.00 | 985.00 | 987.95 | 1008.32 | 17116372 | 1.725876e+15 | NaN | 5206360 | 0.3042 |
| 1 | 2004-08-26 | TCS | EQ | 987.95 | 992.00 | 997.0 | 975.30 | 976.85 | 979.00 | 985.65 | 5055400 | 4.982865e+14 | NaN | 1294899 | 0.2561 |
| 2 | 2004-08-27 | TCS | EQ | 979.00 | 982.40 | 982.4 | 958.55 | 961.20 | 962.65 | 969.94 | 3830750 | 3.715586e+14 | NaN | 976527 | 0.2549 |
| 3 | 2004-08-30 | TCS | EQ | 962.65 | 969.90 | 990.0 | 965.00 | 986.40 | 986.75 | 982.65 | 3058151 | 3.005106e+14 | NaN | 701664 | 0.2294 |
| 4 | 2004-08-31 | TCS | EQ | 986.75 | 986.50 | 990.0 | 976.00 | 987.80 | 988.10 | 982.18 | 2649332 | 2.602133e+14 | NaN | 695234 | 0.2624 |
| 5 | 2004-09-01 | TCS | EQ | 988.10 | 990.00 | 995.0 | 983.60 | 986.00 | 987.90 | 989.68 | 2491943 | 2.466236e+14 | NaN | 790586 | 0.3173 |
| 6 | 2004-09-02 | TCS | EQ | 987.90 | 989.90 | 1004.6 | 986.00 | 994.00 | 993.65 | 996.96 | 2669544 | 2.661426e+14 | NaN | 501792 | 0.1880 |
| 7 | 2004-09-03 | TCS | EQ | 993.65 | 1006.00 | 1100.0 | 990.35 | 998.70 | 997.85 | 996.91 | 1233732 | 1.229917e+14 | NaN | 235508 | 0.1909 |
| 8 | 2004-09-06 | TCS | EQ | 997.85 | 1039.90 | 1039.9 | 992.90 | 996.80 | 994.85 | 998.87 | 1129834 | 1.128554e+14 | NaN | 430184 | 0.3807 |
| 9 | 2004-09-07 | TCS | EQ | 994.85 | 1035.00 | 1035.0 | 995.00 | 995.00 | 995.60 | 997.34 | 721529 | 7.196109e+13 | NaN | 198212 | 0.2747 |
| 10 | 2004-09-08 | TCS | EQ | 995.60 | 996.00 | 1000.8 | 991.10 | 992.25 | 993.70 | 995.29 | 824248 | 8.203617e+13 | NaN | 220195 | 0.2671 |
| 11 | 2004-09-09 | TCS | EQ | 993.70 | 997.00 | 997.9 | 978.45 | 979.65 | 979.95 | 985.16 | 993398 | 9.786574e+13 | NaN | 364189 | 0.3666 |
| 12 | 2004-09-10 | TCS | EQ | 979.95 | 990.00 | 990.0 | 976.00 | 989.95 | 988.80 | 985.12 | 801884 | 7.899507e+13 | NaN | 203388 | 0.2536 |
| 13 | 2004-09-13 | TCS | EQ | 988.80 | 991.00 | 1015.5 | 991.00 | 1002.75 | 1003.50 | 1007.31 | 2613114 | 2.632221e+14 | NaN | 735865 | 0.2816 |
| 14 | 2004-09-14 | TCS | EQ | 1003.50 | 1005.00 | 1018.8 | 1002.95 | 1015.95 | 1015.65 | 1012.95 | 1916934 | 1.941763e+14 | NaN | 493998 | 0.2577 |
| 15 | 2004-09-15 | TCS | EQ | 1015.65 | 1018.00 | 1020.0 | 1001.20 | 1005.00 | 1006.10 | 1009.12 | 1498536 | 1.512208e+14 | NaN | 483466 | 0.3226 |
| 16 | 2004-09-16 | TCS | EQ | 1006.10 | 1007.00 | 1015.0 | 1002.50 | 1011.00 | 1008.45 | 1009.82 | 919778 | 9.288063e+13 | NaN | 240651 | 0.2616 |
| 17 | 2004-09-17 | TCS | EQ | 1008.45 | 1012.00 | 1029.9 | 1010.30 | 1029.90 | 1024.50 | 1021.09 | 1828487 | 1.867047e+14 | NaN | 538239 | 0.2944 |

completed at 9:50 PM

Fig 6.2.1 Test Case 1 Output

Test case2:

```
[4] df.isna().any()
```

| | |
|--------------------|-------|
| Date | False |
| Symbol | False |
| Series | False |
| Prev Close | False |
| Open | False |
| High | False |
| Low | False |
| Last | False |
| Close | False |
| VWAP | False |
| Volume | False |
| Turnover | False |
| Trades | True |
| Deliverable Volume | False |
| %Deliverble | False |

dtype: bool

```
df.dropna(inplace=True)  
df.isna().any()
```

| | |
|--------------------|-------|
| Date | False |
| Symbol | False |
| Series | False |
| Prev Close | False |
| Open | False |
| High | False |
| Low | False |
| Last | False |
| Close | False |
| VWAP | False |
| Volume | False |
| Turnover | False |
| Trades | False |
| Deliverable Volume | False |
| %Deliverble | False |

dtype: bool

Fig 6.2.2 Test Case 2 Output

Test case3:

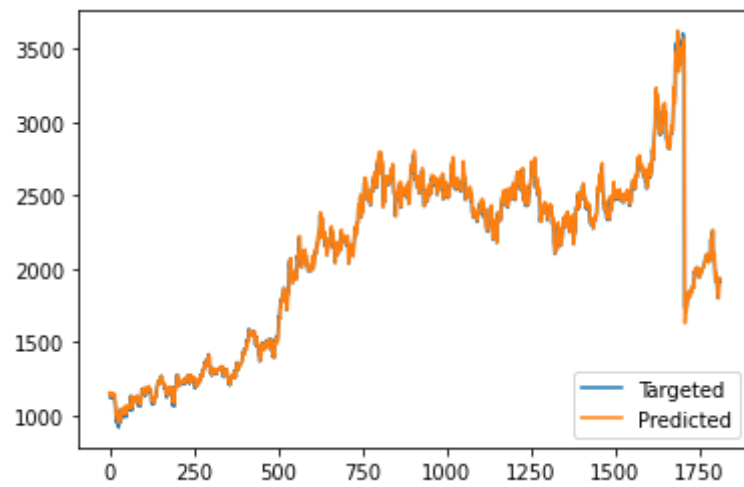


Fig 6.2.3 Prediction Plot for Training Data

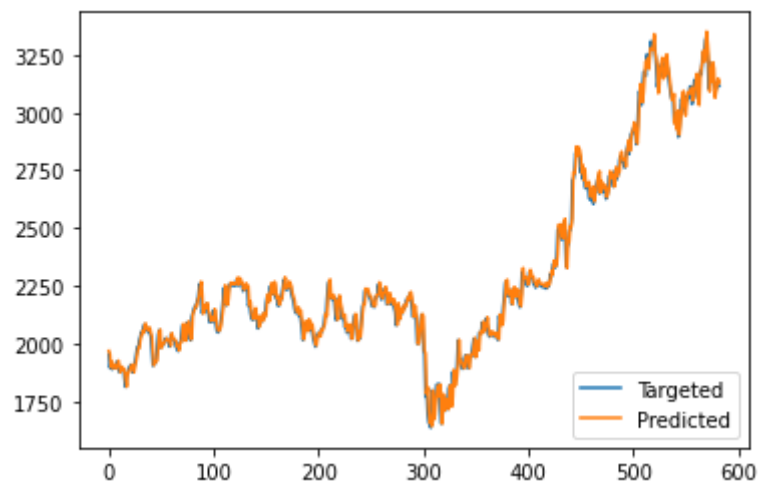


Fig 6.2.4 Prediction Plot for Testing Data

Test case4:

```
✓ 0s # Evaluation metrices RMSE and MAE
print("Train data RMSE: ", math.sqrt(mean_squared_error(original_ytrain,train_predict)))
print("Train data MSE: ", mean_squared_error(original_ytrain,train_predict))
print("Test data MAE: ", mean_absolute_error(original_ytrain,train_predict))
print("-----")
print("Test data RMSE: ", math.sqrt(mean_squared_error(original_ytest,test_predict)))
print("Test data MSE: ", mean_squared_error(original_ytest,test_predict))
print("Test data MAE: ", mean_absolute_error(original_ytest,test_predict))

Train data RMSE:  55.26909835350747
Train data MSE:  3054.6732328096814
Test data MAE:  30.694323157526917
-----
Test data RMSE:  49.70938990174806
Test data MSE:  2471.023444404012
Test data MAE:  37.981083833619216
```

```
✓ 0s [26] print("Train data R2 score:", r2_score(original_ytrain, train_predict))
print("Test data R2 score:", r2_score(original_ytest, test_predict))

Train data R2 score: 0.9911759268398272
Test data R2 score: 0.9850663383351148
```

Fig 6.2.5 :Test Case 4 Output

6.3 VALIDATION TESTING

At the culmination of the integration testing, the software was completely assembled as a package, interfacing errors were uncovered and the final series of software validation testing began. Here we test the system function in a manner that can be reasonably expected by the customer, the system was tested against system requirement specification. Different unusual inputs that the user may use where assumed and the outputs were verified for such unprecedented inputs. This test is performed to validate the software. In this the entire software will be created and will test all the components of the software together. Validation testing ensures that the product meets the client's needs.

TESTING OBJECTIVES:

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has a high probability of finding an as-yet-undiscovered error.
3. A successful test uncovers an as-yet-undiscovered error.

TEST LEVELS:

The test strategy describes the test level to be performed. There are primarily three levels of testing.

1. Unit Testing
2. Integration Testing
3. System Testing

In most software development organisations, the developers are responsible for unit testing. Individual testers or test teams are responsible for integration and system testing

6.3.1 Unit Testing:

Unit testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested using the following two strategies

6.3.1.1 Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. In this testing only the output is checked for correctness. The logical flow of the data is not checked. This testing has been used to find errors in the following categories

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors

6.3.1.2 White Box Testing

In this test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases.

6.3.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.3.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasising pre-driven process links and integration points.

6.4 CONCLUSION:

Hence, this section consists of the positive and negative test cases based on white box testing as all of them are designed based on the input given by the user.

7. CONCLUSION

The project titled Stock Time Series Data prediction using machine learning Techniques is implemented using the machine learning models LSTM and GRU which are versions of Recurrent neural networks. The LSTM and GRU models are trained by feeding past datasets and statistics upon which it has learned and adapted to the pattern and predicted the future value of stock, which is approximate and close to the original value. The LSTM model has given an accuracy of and SVM model has given better accuracy so we have combined LSTM and GRU to create a hybrid model which gave an accuracy of 93% which is the highest accuracy so far we have reached to predict the closing price of stock.

8.REFERENCES:

- [1] D Brownstone, "Using the percentage accuracy to measure neural network predictions in stock market movements", *Neurocomputing*, vol. 10, pp. 237-250, 1996.
- [2] A. S. Chen, M. T. Leung, and H. Daouk, "Application of Neural Networks to an emerging financial market: Forecasting and trading the Taiwan Stock Index", *Comput. Operations Res.*, Vol-30, pp. 901-923, 2003.
- [3] J. C. Patra, G Panda, R Baliarsingh, "Artificial neural network based nonlinearity estimation of pressure sensors", *IEEE transactions on instrument and measurement*, vol 43, no 6, pp.874-881, Dec.1994.
- [4] Pei-Chann Chang, Chin-Yuan Fan, "A Hybrid System Integrating a Wavelet and TSK Fuzzy Rules for Stock Price Forecasting", *IEEE transactions on Systems, MAN, and Cybernetics-Part-C: Applications and Reviews*, Vol. 38, No. 6, Nov-2008
- [5] Hua-Ning Hao, "Short-term Forecasting of Stock Price Based on Genetic-Neural Network", 2010 Sixth International Conference on Natural Computation (ICNC 2010), IEEE Conference Publications.
- [6] Y-H. Pao, "Adaptive Pattern Recognition & Neural Networks", Reading, MA;Addison-Wesley, 1989.
- [7] Y.H.Pao and Y.Takefji, "Functional-Link Net Computing", *IEEE Computer Journal*, pp.76-79, 1992.
- [8] Ritanjali Majhi, G Panda, G Sahoo "Development and performance evaluation of FLANN based model for forecasting of stock markets" *Experts Systems with Applications An International Journal*, ELSVIER, 2008.
- [9] Jagdish Chandra Patra, "Chebysheb Neural Network-Based Model for Dual-Junction Sollar Cells", in *IEEE Transactions on Energy Conversion*, Vol. 26 No. 1, March 2011.
- [10] Satchidananda Dehuri, "A Novel Learning Scheme for Chebysheb Functional Link Neural Networks", *Hindwai Publishing Corporation, Advances in artificial neural systems*, Vol.2011, Article ID 107489, doi:1155/2011/107489, 2011.
- [11] J. C. Patra, C Bornand, P K Meher, "Laguerre Neural Network-based Smart Sensors for Wireless Sensor Networks", *I2MTC-2009, IEEE*, 2009.

- [12] J. Biomed, “Laguerre-based method for analysis of time-resolved fluorescence data: application to in-vivo characterization and diagnosis of atherosclerotic lesions”, Opt. 11, 021004 (Mar 27, 2006); doi:10.1117/1.2186045, 2006
- [13] J. C. Patra, W C Chin, P K Meher, G Chakraborty, “Legendre-FLANNbased nonlinear channel equalization in wireless communication system”, IEEE IC on Man and Cybernetics (SMC-2008), 2008
- [14] S. K. Nanda¹, Debi P. Tripathy, S.S. Mahapatra, “Application of Legendre Neural Network for Air Quality Prediction”, The 5th PSUUNS International Conference on Engineering and Technology (ICET2011), Phuket, May 2-3, 2011.
- [15] Naadun Sirimevan; I.G. U. H. Mamalgaha; Chandira Jayasekara; Y. S. Mayuran; Chandimal Jayawardena (2020). Stock Market Prediction Using Machine Learning Techniques in Faculty of Computing, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka.
- [16]Mojtaba Nabipour , Pooyan Nayyeri , Hamed Jabani , Shahab S., (Senior Member, Ieee), and Amir Mosavi (2020). Predicting Stock Market Trends Using Machine Learning and Deep Learning Algorithms Via Continuous and Binary Data; a Comparative Analysis.
- [17]Ferdiansyah Ferdiansyah; Siti Hajar Othman; Raja Zahilah Raja Md Radzi; Deris Stiawan; Yoppy Sazaki; Usman Ependi(2020). A LSTM-Method for Bitcoin Price Prediction: A Case Study Yahoo Finance Stock Market in School of Computing, Universiti Teknologi Malaysia, Johor Bahru, Johor, Malaysia.
- [18]Sumeet Sarode; Harsha G. Tolani; Prateek Kak; C S Lifna (2019). Stock Price Prediction Using Machine Learning Techniques in Vivekanand Education Society's Institute of Technology, Mumbai, India.

