ANALYSIS OF STOCK PREDICTION USING MACHINE LEARNING

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

V.G. JYOTHI PRIYA(37110301)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119

MARCH-2021

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar nagar, Rajiv Gandhi Salai, Chennai– 600119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of V.G.Jyothi Priya (37110301) who carried out the project entitled **"**Analysis of stock prediction using Machine Learning" under my supervision from November 2020 to April 2021.

Internal Guide

Dr. S. Jancy MCA., MBA., M.TECH., PH.D.

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.

Dr. L. Lakshmanan, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner **External Examiner**

DECLARATION

I V.G.JYOTHI PRIYA hereby declare that the Project Report entitled "ANALYSIS OF STOCK PREDICTION USING MACHINE LEARNING" done by me under the guidance of Dr. S. Jancy MCA., MBA., M.TECH., Ph.D., Department of Computer Science and Engineering and Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor ofEngineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing and Dr. L. Lakshmanan, M.E., Ph.D., and Dr. S. Vigneswari, M.E., Ph.D., Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr. S. Jancy MCA., MBA., M.TECH., Ph.D. for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

Predicting inventory marketplace prices is a compound task that typically includes in covering human-computer interplay. Due to have a mutual courting or connection nature of stock expenses, conventional batch processing methods can't be make sensible and effective use for inventory market Evaluation. We advocate an algorithm that make practical and effective use of a type of recurrent neural community (RNN) called Long Short Term Memory (LSTM), wherein the weights are adjust for single data factors the use of stochastic gradient descent. This will offer greater accurate effects when in comparison to current inventory rate prediction algorithms. The network will be drill and Assess from free for blunders with diverse sizes of facts, and the results are arranged in tabular form. A comparison with admire to accuracy is then accomplished in opposition to an Artificial Neural Network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSIONS
KPI	Key performance indicator
GDP	Gross Domestic product
CPI	Consumer Price Index
EMH	Efficient Market Hypothesis
GA	Genetic Algorithms
ANN'S	Artificial Neural Networks
LSTM	Long short term memory
RNN	Recurrent Neural Network
RMSE	Root Means Square Error
ESM	Exponential Smoothing Model
AICBIC	Akaike Information Criterion Bayesian information criterion
GRU'S	Gated Recurrent Unit's
MPS	Model Prediction Score
MDS	Model Domestic Score
SPS	Snake Prediction Scores
MAS	Model Accuracy Score

CHAPTER-1
INTRODUCTION

The stock marketplace is a Huge array of investors and sellers who purchase and promote stock, push the price up or down. The costs of shares are administered by means of the standards of hobby and deliver, and the inevitable objective of buying shares is to bring in coins with the aid of buying stocks in groups whose observe esteem (i.E., share price) is needed to rise. Financial exchanges are firmly related with the universe of economic elements — the ascent and fall of offer prices can be observed back to some Key Performance Indicators (KPI's). The 5 most usually utilized KPI's are the preliminary stock fee ('Open'), end of-day value ('Close'), intraday low cost ('Low'), intra-day top fee ('High'), and all out extent of shares exchanged for the duration of the day ('Volume').

Dissemination and inventory expenses are mainly Dependent upon suppositions the capacity to look approximately the monetary exchange. It is close to tough to assume inventory costs to the T, inferable from the unpredictability of components that assume a giant element inside the development of fees. Be that as it may, it's miles achievable to motive an informed to evaluate of costs. Stock costs never shift in separation: the improvement of one will in preferred have a torrential slide impact on a few distinct shares too. This part of inventory value improvement can be utilized as a sizable apparatus to foresee the expenses of several stocks at once. Because of the continuous quantity of money protected and wide variety of trades that happen each moment, there comes a trade something of extensive worth among the precision and the quantity of forecasts made; all things taken into consideration, maximum stock expectation frameworks are finished in a conveyed, modify style. These are a portion of the contemplations and difficulties regarded in financial alternate investigation.

Financial markets are one of the most fascinating inventions of our time. They have had a significant impact on many areas like business, education, jobs, technology and thus on the economy. Over the years, investors and researchers have been interested in developing and testing models of stock price behaviour.

However, analysing stock market movements and price behaviours is extremely challenging because of the markets dynamic, nonlinear, nonstationary, nonparametric, noisy, and chaotic nature, stock markets are affected by many highly interrelated factors that include economic, political, psychological, and company-specific variables. Technical and fundamental analysis are the two main approaches to analyse the financial markets. To invest in stocks and achieve high profits with low risks, investors have used these two major approaches to make decisions in financial markets.

fundamental analysis is mainly based on three essential aspects (i) macroeconomic analysis such as Gross Domestic Product (GDP) and Consumer Price Index (CPI) which analyses the effect of the macroeconomic environment on the future profit of a company, (ii) industry analysis which estimates the value of the company based on industry status and prospect, and (iii) company analysis which analyses the current operation and financial status of a company to evaluate its internal value. Different valuation approaches exist for fundamental Analysis. The average growth approximation technique compares Stock-A with other stocks in the same category to better understand valuations, i.e., assuming two companies have the same growth rate, the one with the lower Price-to-Earnings (P/E) ratio is considered to be better. Hence the fair price is the earnings times target P/E. The P/E method is the most commonly used valuation method in the stock brokerage industry. The constant growth approximation technique such as Gordon's growth model is one of the best-known classes of dividend discount models. It assumes that dividends of a company will increase at a constant growth rate forever but less than the discount rate. demonstrated the utility of fundamental analysis through the use of financial ratios to separate good stocks from poor stocks. The authors compared their one-year return against the benchmark—i.e., Nifty—which gives an accuracy of 74.6%. This is one of the few papers which focus on using fundamental features (i.e., company-specific ratios) to identify stocks for investments.

Furthermore, grouped the domains of technical analysis into sentiment, flow-of-funds, raw data, trend, momentum, volume, cycle, and volatility. Sentiment represents the behaviours of various market participants.

Flow-of-funds is a type of indicator used to investigate the financial status of various investors to pre-evaluate their strength in terms of buying and selling stocks, then, corresponding strategies, such as short squeeze, can be adopted. Raw data include stock price series and price patterns such as K-line diagrams and bar charts. Trend and momentum are examples of price-based indicators, trend is used for tracing the stock price trends while momentum is used to evaluate the velocity of the price change and judge whether a trend reversal in stock price is about to occur. Volume is an indicator that reflects the enthusiasm of both buyers and sellers for investing, it is also a basis for predicting stock price movements. The cycle is based on the theory that stock prices vary periodically in the form of a long cycle of more than 10 years containing short cycles of a few days or weeks. Finally, volatility is often used to investigate the fluctuation range of stock prices and to evaluate risk and identify the level of support and resistance.

Sentiments can drive short-term market fluctuations which in turn cause disconnects between the price and true value of a company's shares but over long periods of time, however, the weighing machine kicks in as a company's fundamentals ultimately cause the value and market price of its shares to converge. A prominent example comes from the Nobel Laureate Robert Shiller, who showed that stock prices are extremely volatile over the short term but somewhat predictable by their price-to-earnings over long periods. She explained what returns to expect from the stock markets considering the economic scenario and suggested that in the future, returns could be substantially lower. Shiller (2000) also suggested that stocks are overvalued, and the bubble will burst anytime. In the year 2000, rightly so, we witnessed the dotcom bubble burst.

Stock market price prediction is a tricky thing. Several theories regarding stock markets have been conceptualized over the years. They either try to explain the nature of stock markets or try to explain whether the markets can be beaten. One such popular and most debated theory given by Fama (1970) is the Efficient Market Hypothesis (EMH) which states that at any point in time, the market price of a stock incorporates all information about that stock. In other words, the stock is accurately valued until something changes. There are three variants of EMH

(i) the weak form which is consistent with the random walk hypothesis, and that stock prices move randomly while price changes are independent of each other hence, it is not possible to beat the market by earning abnormal returns on the basis of technical analysis; (ii) the semi-strong form which states that prices adjusted rapidly according to market and public information such as dividend, earnings announcements, and political or economic events, hence it is not possible to earn abnormal returns on the basis of fundamental analysis; and, finally, (iii) the strong form which states that prices reflect market, public, and private information as such no investor has monopolistic access to information.

According to EMH, price changes are unpredictable and forecasting a financial market is a hopeless effort. However, (Abu-Mostafa and Atiya 1996) argued that the existence of so many price trends in financial markets and the undiscounted serial correlations among fundamental events and economic figures affecting the markets are two of many pieces of evidence against the EMH. Researchers andInt. J. Financial Stud. 2019, 7, 26 3 of 22 investors disagree with EMH both empirically and theoretically, thereby shifting the focus of discussion from EMH to the behavioural and psychological aspects of market players (Naseer and Tariq 2015). According to Zhong and Enke (2017), financial variables, such as stock prices, stock market index values, and the prices of financial derivatives are therefore thought to be predictable. Many widely accepted empirical studies show that financial markets are to some extent predictable (Chong et al. 2017). Criticism of EMH has given rise to an increasing number of studies that question the validity of EMH and introduce new and successful approaches that combine technical analysis indicators and chart patterns with methodologies from econometrics, statistics, data mining, and artificial intelligence (Arévalo et al. 2017).

Many new technologies and methods have been proposed over the years to try and predict stock prices via many avenues, thanks to the challenging and ever-changing landscape of stock markets (Chen and Chen 2016). In this paper, we focus on two topics, namely, stock analysis and stock prediction. We look at the research in the past, but mainly focus on modern techniques, highlighting some of the main challenges they pose and recent achievements for stock analysis and prediction. Finally, we discuss potential challenges and possible future research directions.

We organize the rest of this paper as follows. Section 2 provides a background review and taxonomy of the various approaches to stock market analysis. Section 3 describes a literature study on stock markets analysis and prediction. Section 4 discusses and compares the approaches mentioned in Section 3. Section 5 provides an overview of challenges and additional areas for future research. Finally, Section 6 concludes the paper.

EXISTING SYSTEM:

Customary ways to deal with financial exchange investigation and stock value expectation incorporate principal examination, which takes a gander at a stock's previous exhibition and the overall believability of the actual organization, and factual examination, which is exclusively worried about calculating and distinguishing designs in stock value variety. The last is generally accomplished with the assistance of Genetic Algorithms (GA) or Artificial Neural Networks (ANN's), yet these neglect to catch relationship between's stock costs as long haul worldly conditions. Another significant issue with utilizing straightforward ANNs for stock expectation is the wonder of detonating/disappearing gradient[4], where the loads of an enormous organization either become excessively huge or excessively little (separately), definitely easing back their union to the ideal worth. This is regularly brought about by two variables: loads are introduced haphazardly, and the loads nearer to the furthest limit of the organization additionally will in general change significantly more than those toward the start. An elective way to deal with securities exchange examination is to lessen the dimensionality of the information [2] and apply include decision calculations to waitlist a center arrangement of highlights (like GDP, oil value, expansion rate, and so on) that greatestly affect stock costs or money trade rates across business sectors [10]. Notwithstanding, this strategy doesn't consider long haul exchanging methodologies as it neglects to consider the whole history of patterns; besides, there is no arrangement for exception identification.

PURPOSE OF PROJECT:

The purpose of this project is to accurately predict the future closing value of a given stock across a given period of time in the future. For this project I have used a Long Short Term Memory networks – usually just called “LSTMs” to predict the closing price of the S&P 500 using a dataset of past prices

- **Achievements:**
    - Built a model to accurately predict the future closing price of a given stock, using Long Short Term Memory Neural net algorithm.
    - Achieved Mean Squared Error rating of just 0.00093063.

Things i have learnt by completing this project:

- How to apply deep learning techniques: Long Short Term Memory Neural Network algorithms.
- How to use keras-tensorflow library.
- How to collect and preprocess given data.
- How to analyze model's performance.
- How to optimise Long Short Term Memory Neural Network algorithm, to ensure increase in positive results.

PROPOSED SYSTEM

We propose an internet learning calculation for foreseeing the finish of-day cost of a given stock (see Figure 2) with the assistance of Long Short Term Memory (LSTM), a sort of Recurrent Neural Network (RNN).

BATCH VERSUS ONLINE LEARNING ALGORITHMS:

On the web and group learning calculations vary in the manner by which they work.

In an online calculation, it is conceivable to stop the enhancement interaction in a learning run and still train a compelling model. This is especially helpful for huge informational indexes, (for example, stock cost datasets) when the intermingling can be estimated and learning can be stopped early.

The stochastic learning worldview is unmistakably utilized for internet learning on the grounds that the model is prepared on each information point — every boundary update just uses a solitary haphazardly picked information point, and keeping in mind that motions might be noticed, the loads at last merge to a similar ideal worth. Then again, cluster calculations keep the framework loads consistent while figuring the blunder related with each example in the information. That is, every boundary update includes an output of the whole dataset — this can be profoundly tedious for enormous assessed stock worth data. Speed and precision are correspondingly huge inpredicting patterns in stock value development, where costs can vacillate fiercely inside the range of a day. In that capacity, a web based learning approach is better for stock value expectation.

LSTM

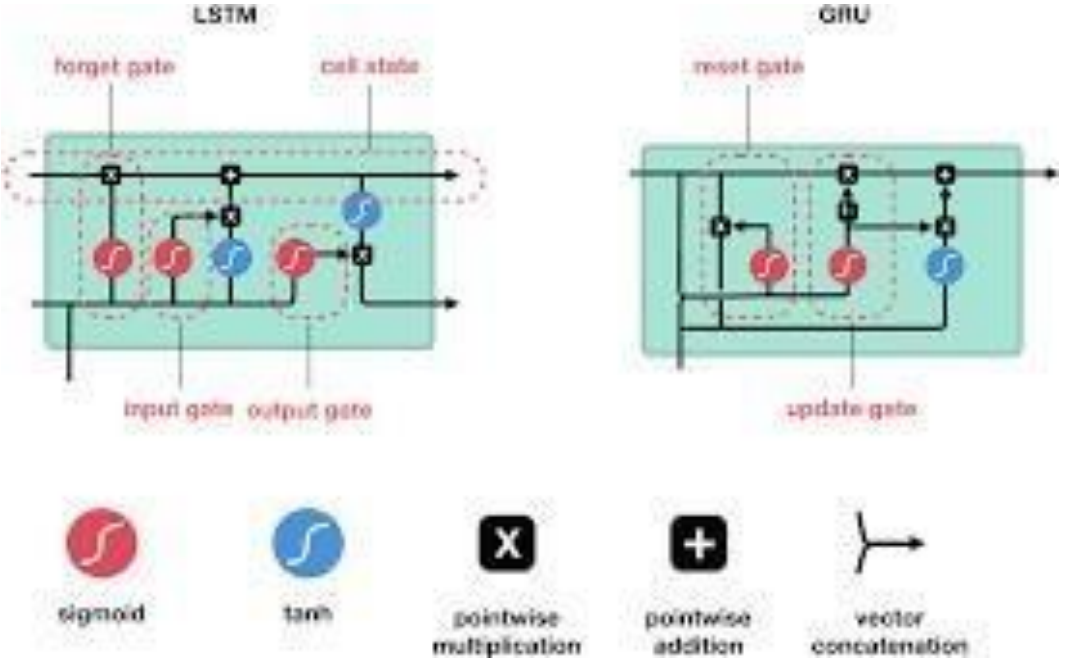


Fig - 1: An LSTM memory cell

LSTM's are an exceptional subset of RNN's that can get setting explicit otherworldly compulsion for extensive stretches of time. Each LSTM neuron is a memory cell that can store other data i.e., it keeps up its own cell state. While neurons in typical RNN's just take in their past secret state and the current contribution to yield another secret express, a LSTM neuron additionally takes in its old cell state and yields its new cell state. A LSTM memory cell, as portary in Figure 1, has the accompanying three segments, or doors:

1. FORGET GATE: The fail to remember door chooses when explicit parts of the cell state are to be supplanted with later data. It yields esteems near 1 for parts of the cell express that ought to be held, and zero for values that ought to be ignored.

2. INPUT GATE: In light of the information (i.e., past yield o(t-1), input x(t), and past cell state c(t-1)), this part of the organization learns the conditions under which any data ought to be put away (or refreshed) in the cell state.

3. OUTPUT GATE: contingent upon the information and cell express, this bit chooses what data is proliferated forward (i.e., yield o(t) and cell state c(t)) to the following hub in the organization.

Hence, LSTM networks are ideal for investigating what variety in one stock's cost can mean for the costs of a few different stocks throughout a significant stretch of time. They can likewise choose (in a powerful design) for how long data about explicit past patterns in stock value development should be held to all the more precisely foresee future patterns in the variety of stock costs.

CHAPTER 2
LITERATURE SURVEY

The underlying focal point of our writing review was to investigate explicit internet learning calculations and check whether they could be adjusted to our utilization case i.e., dealing with continuous stock value information. These included Online AUC Maximization, Online Transfer Learning, and Online Feature Selection.

In any case, as we couldn't locate any conceivable change of these for stock value forecast, we at that point chose to take a gander at the current frameworks, break down the significant downsides of the equivalent, and check whether we could refine them. We focused in on the relationship between's stock information (as unique, long haul worldly conditions between stock costs) as the central point of interest that we wished to address. A short inquiry of conventional answers for the above issue drove us to RNN's and LSTM.

In the wake of choosing to utilize a LSTM neural organization to perform stock forecast, we Asked various papers to consider the idea of slant plunge and its different kinds. We closed our writing study by seeing how incline drop can be utilized to tune the loads of a LSTM organization and how this cycle can be can be improve.

Numerous statistical techniques have been tried and tested for stock market analysis and prediction. The Exponential Smoothing Model (ESM) is a popular smoothing technique which is applied on time series data, it essentially uses the exponential window function for smoothing time series data and analyse the same (Billah et al. 2006). De Faria et al. (2009) compared the ANN and adaptive ESM model for predicting the Brazilian stock indices. Their experiment revealed the predictive power of ESM and the results for both methods show similar performances although the neural network model

i.e., the multilayer feedforward network slightly outperformed the adaptive ESM in terms of the Root Means Square Error (RMSE). Dutta et al. (2012) took an interesting path by selecting the financial ratios as independent variables for a logistic regression model and then analysed the relationship between these ratios and the stock performances.

The paper focused on a classification task for predicting companies which are good or poor based on their one-year performance. The results show that the financial ratios—like net sales, book value PE, price-to-book (P/B), EBITDA, etc.—classify the companies into good and poor classes with an accuracy of 74.6%, which is a good indication of why company health matters in stock analysis and prediction.

Devi et al. (2013) tried to address some issues not currently addressed in most of the stock analysis literature, such as the dimensionality and expectancy of a naïve investor.

The authors essentially utilize the historical data of four Indian midcap companies for training the ARIMA model. The Akaike Information Criterion Bayesian Information Criterion (AICBIC) test was applied to predict the accuracy of the model. Testing the model on individual stocks and the Nifty 50 Index showed that the Nifty Index is the way to go for naïve investors because of low error and volatility.

Ariyo et al. (2014) explore the extensive process of building ARIMA models. To identify the optimal model out of all the ARIMA models generated, the authors chose criteria like the standard error of regression, adjusted R-square, and Bayesian information . The best ARIMA model, based on the above criteria, did a satisfactory job in predicting the stock prices of Nokia and Zenith Bank.

Furthermore, Ariyo et al. (2014) made a solid case not to undermine the powers of ARIMA models in terms of stock analysis because it can compete reasonably well against the emerging forecasting techniques available today for short term prediction.

Bhuriya et al. (2017) implemented variants of regression models to predict the stock price of Tata Consultancy Services stock based on five features i.e., open, high, low, close price, and volume. The paper compares the performances of the linear, polynomial, and Radial Basis Function (RBF) regression models based on the confidence values of the predicted results.

In addition, Bhuriya et al. (2017) reported that the linear regression model outperformed the other techniques and achieved a confidence value of 0.97.

CHAPTER 3
AIM AND SCOPE OF PRESENT INVESTIGATION

AIM OF THE PROJECT:

In Stock Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Machine learning itself employs different models to make prediction easier and authentic. The paper focuses on the use of Regression and LSTM based Machine learning to predict stock values. Factors considered are open, close, low, high and volume.

SCOPE AND OBJECTIVES:

As a scope of this project, this research would like to perform a comparative analysis with deep learning classifiers and extreme learning classifiers with the help of a feature reduction algorithm based on the parameters used for stock market prediction. Along with this, research would also like to study and implement economic growth model for stock market prediction and the analysis of how economic growth model will affect in stock market prediction in comparison to the linear regression model and with specialized machine learning techniques.

the objective of this study is to predict the future stock market prices in comparison to the existing methodologies such as regression or continuous learning and by modifying them with the current methodologies efficiently by analyzing the recent trends of various researchers.

- In the current emerging competitive market, predicting the stock returns as well as the company's financial status in advance will provide more benefits for the investors in order to invest confidently.
- Stock prediction can be done by using the current and previous data available on the market. Methods: The performance metrics that need to be attained in case of stock prediction are accuracy, scalability and less time consumption.
- There are many researches done so far in order to predict the stock market to achieve the defined metrics.
- Many models have been available in the field of data mining for predicting the stock market such as if-then-else rules, Artificial Neural Network (ANN), Fuzzy systems, Bayesian algorithm and so on.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS:

FUNCTIONAL REQUIREMENTS:

Requirement Analysis After the extensive analysis of the problems in the system, we are familiarized with the requirement that the current system needs. The requirement that the system needs is categorized into the functional and non-functional requirements.

These requirements are listed below:

Functional Requirements Functional requirement are the functions or features that must be included in any system to satisfy the business needs and be acceptable to the users. Based on this, the functional requirements that the system must require are as follows:

- The system should be able to generate an approximate share price.
- The system should collect accurate data from the NEPSE website in consistent manner.

The prediction shall abide by the following functional requirements:

1. Prior to application of stock recommendations, the database is updated by the latest values.
2. The charts and comparison of the companies would be done only on the latest data stock market data.
3. The user is provided with a login, logging into which enables the user to view his past stock purchases and future recommendations.
4. The user can look previous data Information which was collected.
5. Each user has a friend list and can also be recommended on their buying patterns.
6. The user can also be recommended on the basis of the trending stocks which would require the data regarding the stocks

NON-FUNCTIONAL REQUIREMENTS: Non-functional requirement is a description of features, characteristics and attribute of the system as well as any constraints that may limit the boundaries of the proposed system. The non-functional requirements are essentially based on the performance, information, economy, control and security efficiency and services. Based on these the non-functional requirements are as follows:

- The system should provide better accuracy.
- The system should have simple interface for users to use.
- To perform efficiently in short amount of time.

1. Reliability: The reliability of the product will be dependent on the accuracy of the data, date of purchase, how much stock was purchased, high and low value range as well as opening and closing figures. Also the stock data used in the training would determine the reliability of the software.

2. Security: The user will only be able to access the website using his login details and will not be able to access the computations happening at the back end.

3. Maintainability: The maintenance of the product would require training of the software by recent data so that there commendations are up to date. The database has to be updated with recent values.

4. Portability: The website is completely portable and the recommendations completely trustworthy as the data is dynamically updated.

5. Interoperability: The interoperability of the website is very high because it synchronize all the database with the wamp server

CHAPTER 4
SYSTEM DESIGN AND METHODOLOGY

SYSTEM ARCHITECTURE

The architecture of the system follows a client-server model, where the server and the client are loosely coupled.

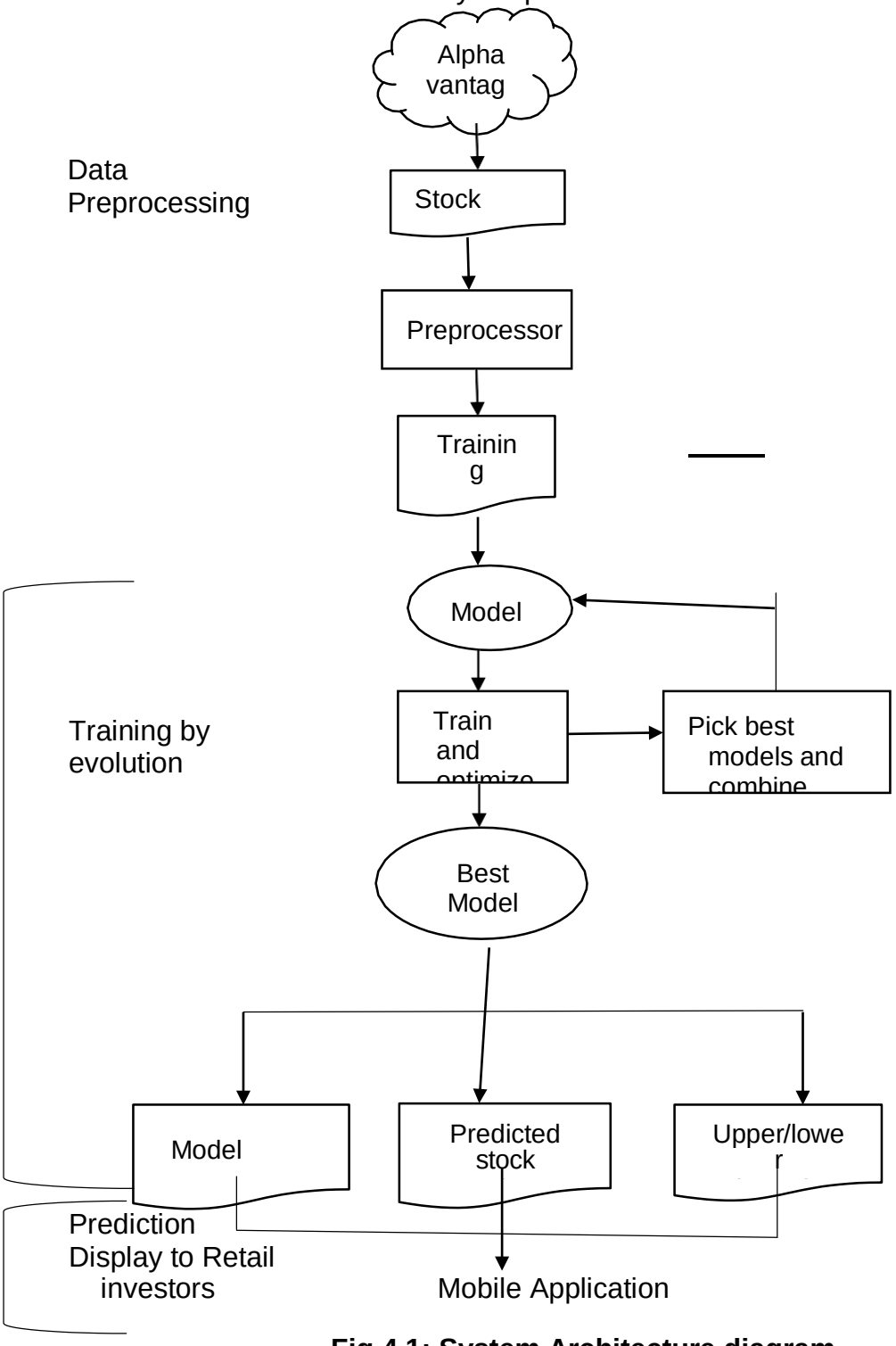


Fig 4.1: System Architecture diagram

After relevant stock data are retrieved from the third-party data provider through the cloud, the backend pre-processes the data and builds the models. After that, predictions are made and the prediction results will be stored on another cloud, which can be retrieved from the mobile application.

The advantages of the loosely coupled architecture include improved scalability and ease of collaboration. The workload for the cloud which serves the models and the one which serves the mobile application will be very different. One cloud serves the model prediction results, which are simple text files; another cloud serves the mobile application with a lot of rich user content such as images and large UI libraries. Having two clouds to adapt to two different demand patterns is more efficient, especially since cloud providers these days usually serve content on demand.

Also, the separation allows different team members in the team to focus on different parts after agreeing on a common interface. It speeds up development as team members responsible for different parts of the system do not need to take care of the underlying implementation details. Also, it is easier to swap out different components, e.g. to replace the models the team could simply make changes to the backend, while the frontend remains unaffected.

RESEARCH DESIGN

PROBLEM FRAMING

The problem of the project is set to predict the stock price for the next 10 business days. “10 days” is chosen as the timeframe as short term price movements tend to depend more on trend momentum and price pattern, while long term price movements depend on the fundamentals of a stock (e.g. company management capabilities, revenue model, market demand, macroeconomic factors, etc.).

The loss function of the training algorithm is the mean squared error of the 10 predicted stock prices. The training algorithm or optimizer is set to minimize its value, and it serves as the basic performance metric for comparing different models.

Other scores are defined to provide more in-depth insights on a model predictability performance and finance-domain-based comparisons between models for investors.

Two different prediction approaches are mainly tested, predicting the stock prices for the next 10 days directly and predicting the stock price of the next day 1 at a time. It is suspected that the two different problem framing approaches will result in different abstractions learned hence performance for different use-cases.

As different stocks have very different characteristics and the stock prices exhibit different trends, individual models will be built for separate stocks.

For the project, S&P 500 stocks from different industries are selected. Multiple factors are considered when picking the stocks, including stock price volatility, the absolute magnitude of the price, the respective industries, company size, etc., and stocks exhibiting different characteristics are picked. The stocks are listed as below:

-Alphabet Inc., GOOGL (Technology)
- Amazon.com Inc., AMZN (Technology)
- Apple Inc., APPL (Technology)
- AT&T Inc., T (Telecom Services)
- Boeing Co., BA (Industrials)
- Caterpillar Inc., CAT (Industrials)
- Facebook Inc., FB (Technology)
- General Electric Co, GE (Industrials)
- Harley-Davidson, Inc., HOG (Consumer Cyclical)
- Microsoft Inc., MSFT (Technology)
- Procter & Gamble Co, PG (Consumer Defensive)

ROBUST DESIGN

For the research side, the system is designed to be as robust as possible to facilitate model testing. Each model can be defined by a pair of model options and input options, specifying the model configurations and the inputs it takes. This accelerates the process of testing out different model and/or input configuration combinations.

DATA PREPROCESSING

Raw stock price data is pre-processed before inputting into machine learning models. Pre-processing includes transforming the raw data into a format that models can take from and operate on, most likely feature matrix. It also attempts to extract some features, financial-domain-specific especially, manually to improve results, allowing the model to learn more abstractions.

Two key features are selected as the input. First is a fixed-length list of some raw historical data like stock price and daily percentage change. The fixed length chosen specifies the length of the historical period to look back from today when predicting future stock prices. Referring to the principle of technical analysis, as the stock price reflects all relevant information, a technical analyst would focus on the trading pattern of the stock rather than the economic fundamentals and company fundamentals. Therefore, by getting a period of historical stock prices as the input for the training model, it could be a piece of useful information in finding the trading patterns and hence predicting the trend of future stock prices. Given a set lookback period, it is assumed that the price movement patterns that are predictive would occur in the specified historical period.

The second feature input is arithmetic moving averages. As mentioned in 1.1, one of the obvious approaches for retail investors to identify the trend of the market is through moving averages.

With the robust system design, different period of moving averages could be used as the input into the model for stock price prediction, for example, a set of 22, 50, 100, 200 days moving averages, which are commonly used by investors [13].

PREDICTION OUTPUT

As mentioned in 4.2.1, 2 different prediction approaches are tested, which will have different outputs.

For 10-day predictions, there will be 10 output units, resulting in a one-dimensional vector with 10 stock prices, where the i-th element represents the i-th day stock price prediction.

For 1-day prediction, there will be 1 output unit which is the stock price in the following day. The predicted stock price of will then be the input of the next prediction, to predict the stock price in the second day, the process repeats until all 10 predictions are generated.

MODEL

Different common neural network models are tested, including dense neural network, simple recurrent neural networks (RNNs), long short-term memory networks (LSTMs) and gated recurrent unit networks (GRUs).

Different model architectures are tested by changing the number of hidden layers, the number of hidden units per hidden layer, and the activation function or recurrent activation function used in each hidden layer.

All recurrent neural networks, RNNs, LSTMs, and GRUs, are set to have the same high-level architecture (Figure 2.2), a stack of recurrent layers by passing the full output sequence to the next layer, followed by a stack of dense layers.

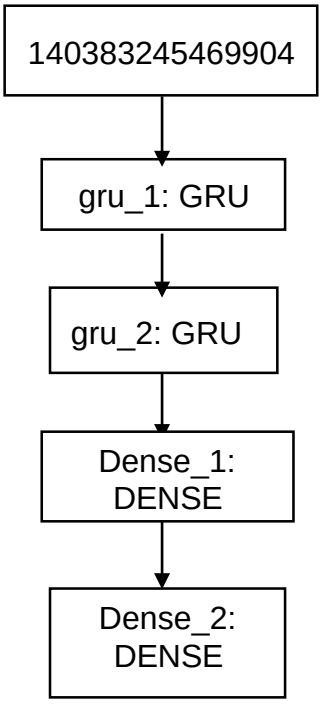


Fig 4.2: Example of the common high-level architecture of recurrent neural networks.

Linear Regression on features, as well as trend lines which interpolate the stock prices next 10 days linearly, are also tested.

MODEL ARCHITECTURE AND HYPER PARAMETER SEARCH WITH EVOLUTION ALGORITHM:

Designing neural network architecture is challenging, even for computer scientists, researchers and machine learning experts. The team does not have the expertise in designing innovative and suitable architectures that will fit the requirement. Given the huge number of architecture types and hyper-parameters for each model, the search space is basically infinite, so a brute-force approach with grid search would not be practical.

Inspired by the paper [12], this project replicates the evolution algorithm in the context of stock price prediction. The algorithm serves as a heuristic for architecture search, using reasonable and affordable computing power to search for ideal architectures.

The same evolution algorithm was used in the paper [12] to train large-scale image classifiers. The following is the evolution algorithm used, and the corresponding algorithm parameters are defined in Appendix E.

1. Create a population of size POPULATION_SIZE of random simple neural networks.

2. Train all neural networks in the population.

3. Calculate the mean squared error on the test set for each trained neural network.

4. Randomly select 2 networks. Select the one with better performance (lower error) as the parent network, and remove the one with a worse performance from the population.

5. Mutate the parent network to generate a new network and add it to the population.

6. Train the new network.

7. Calculate the mean squared error of the new network on the test set.

8. Repeat steps 3 - 7 for ITERATIONS number of iterations.

Different mutations are used at each step to slowly evolve the population, for example adding a dense layer, changing the number of units in a certain layer or changing the learning rate. For a full mutation list, see Appendix D. In theory, it is also possible to put the model inputs as a variable into the evolution algorithm, using the algorithm to find the opincrease the search space .

PERFORMANCE EVALUATION

MOTIVATION

As mentioned in 4.2.1, apart from the mean squared error that a model tries to minimize, different finance-specific scores are introduced to evaluate and compare performance of different models, namely model accuracy score, model trend score and stock buy/sell score. The scores are also designed to convey useful and meaningful messages to help investors understand a stock and make investment decisions.

MODEL ACCURACY SCORE

The first indicator of the performance is the Model Accuracy Score (MAS). It describes the accuracy of the price prediction regarding the actual price. It is a weighted sum of Model Prediction Score (MPS) and Model Direction Score (MDS), ranging in [0,1]. A variable α is declared to adjust the weighting between MPS and MDS contributing to MAS. Its formula is defined below:

$$\text{Model Accuracy Score(MAS)} = (1-\alpha)\cdot \text{MPS} + \alpha \cdot \text{MDS}$$

MPS is the average of Snake Prediction Scores (SPS). Each SPS is calculated by the prediction error in each of the 10-day disjoint segments, where the error is basically an average of the absolute relative change between the predicted prices and the actual prices over the 10 days. It is defined that SPS is 0 If the error is larger than the standard deviation of the stock, as the prediction would have no reference value under this circumstance. If otherwise, a scoring concave upward function is applied to scale the error to a range of [0,1] based on the standard deviation.

A concave upward function is applied because the marginal usefulness of the model decreases with a marginal increase in error.

Model Prediction Score(MPS)=1 ÷ |□|∑$^{|S|}$ SPS$_i$

Snake Prediction Score(SPS)

=

$$\begin{cases} (\square - \square \div \square)^4 & \square \square \square \leq \square \\ \square \text{therewise} & \end{cases}, \square h \square \square \square \square \frac{1}{10} \sum_{\square}^{10} = 1 \left| \frac{\square i}{Pi} - 1 \right|$$

Meanwhile, MDS is the average of Snake Direction Scores (SDS). Each SDS is evaluated by the alignment of the prediction direction and the actual direction of the stock trend in each of the 10-day disjoint segments. If the prediction has a different direction with the actual direction, it means the prediction is giving a false trend signal to the users. Thus, SDS is 0. Otherwise, SDS would be evaluated based on the direction of the estimation error. In other words, if the prediction is overestimated, SDS is 0.8. Otherwise, it is 1. It is because it is assumed that an underestimated prediction means the model is more reserved and is better off than an overestimating model.

APPLICATION DESIGN

USER GROUPS

Users are separated into two groups, normal users and advanced users. For users that would like to know about the historical (test set) performance of a model and more information behind the machine learning models like the architecture and inputs, they can enable advanced user mode in the settings page to view those details in each individual stock page.

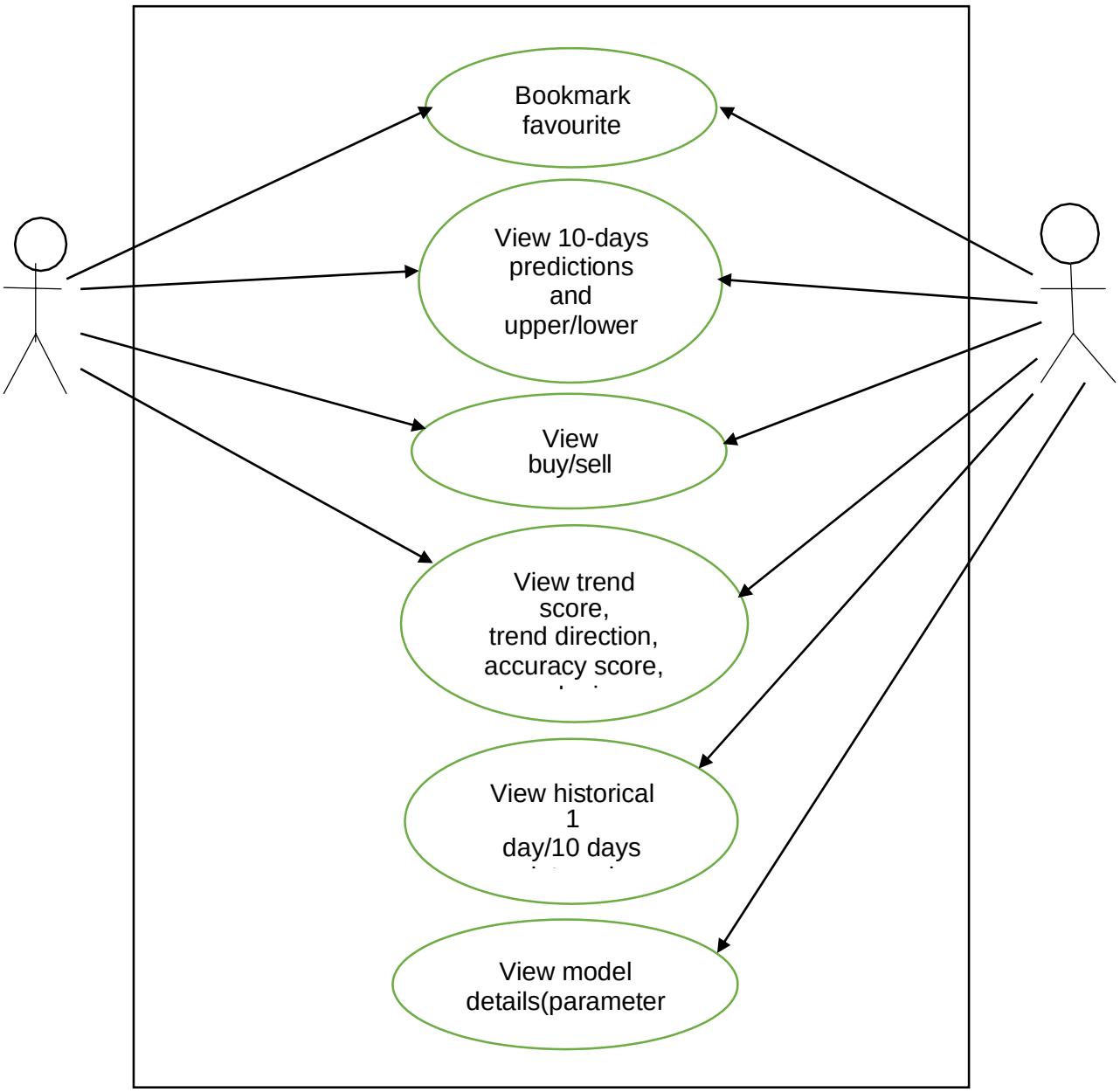


Fig 4.3: functionality accessible by normal users and advanced users

USER JOURNEY

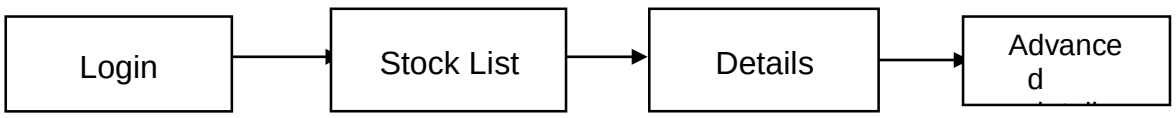


FIG 4.4 User Journey

First of all, users need to login to use the system, as there will be customization options for different users. Since users might not want to create a separate account just for our application, it will more convenient if users can log in with their existing social media accounts. In particular, Facebook login is a good option, since there are over 2 billion users worldwide. Thus, it might be possible to reach a larger market by adopting Facebook login. Only the very basic user information like the user name will be collected by the system.

For normal users (advanced user mode disabled), after logging into the system, they can view the stock list and search from it by stock name. After they find the stock that they are interested in, they can bookmark the stock as a favorite stock for easier access later. After selecting a stock, they can view 3 key information in the details page.

First, the next 10-day predictions and the corresponding upper/lower bounds of the pre-selected best model together with 3 months of historical stock prices.

Second, they can look at the buy/sell score to get an intuitive sense on whether the stock is currently on an uptrend or downtrend based on the predictions.

Third, the user can view the individual trend score, predicted trend direction, accuracy score, and predicted price movement for each individual prediction model.

For advanced users, apart from the 3 key information, they can view 2 additional pieces of information for more in-depth insights, understanding and analysis.

First, they can toggle to view the historical predictions, which helps to evaluate the trustworthiness of different models. There are 2 ways to view the historical prediction performance. For 1-day historical prediction, it could let the user understand how well the model could predict the next day stock price. For 10-days historical prediction (defined as Snakes in 2.2.7.2), it could let the user understand 10 trials of how well a model could predict the stock price in 10-day segments. Thus, this information would help the user to get more information to determine whether the model is relevant and accurate enough for their references.

The second additional information is the layers, hyperparameters and model inputs configured for each individual machine learning model. This information would be useful for them to understand the topology of the model.

All data, including stock prices, predictions, historical predictions and all scores will be updated everyday to reflect the latest information and predictions, which allows users to revisit their investment position.

UI/UX

PROGRESSIVE WEB APPLICATION MOTIVATION

The application is written as a progressive web application (PWA) [14] instead of a native mobile application. The motivation behind this is that the application could be inherently adapt to desktop and mobile usage. It would be more costly to create native desktop and native mobile application separately. The web application can also allow the system to keep only one centralized instance, where information only has to be updated once without any duplicated effort.

A progressive web application is typically implemented as a single page application [15], where pages do not reload entirely like a web page refresh. Instead, the web application only uploads the components as needed as the users interact with the application. It enables smoother, more app-like user experience.

LAYOUT MOTIVATION

The priority of information display should depend on the relevancy of such information from the user's point of view. In the stock list page, a "favorite" section is placed above the "recent" section, and "others" are placed below the two (Figure 2.5 middle). This design considered the relevance of the individual stock in the user perspective. Favorite stocks should have higher priority as these are their focused stock that they are interested in or have owned shares. The "recent" section is crucial as it provides quick access for the user to revisit their recent history and find the stocks that are closely relevant to what they have been checking on recently.

Inside the stock details page, the stock name with its stock code has a significant color and portion at the top of the display, and a gray colored industry tag is placed above the stock name. When the user clicked on the stock name, the section will expand to display a brief overview of the stock. This design can make the user quickly recognize which stock they are checking on.

As the core component of the application is the predictions with upper and lower bounds that provide insight for the retail investors to review the trend of the stock. The chart is placed just below the stock name and being centered on the screen whenever the page is loaded. This design let the user quickly review the prediction trend and recognize whether the predicted trend would interest them to continue checking out the stock or not.

The chart is also very important for the advanced users to cross-check whether the predictions have reference value according to its past performance. The 10-day interval historical prediction and 1-day interval historical prediction are plotted on the chart along with the historical stock price when the user enabled such options, this serves the purpose of letting the user know how well the model could predict the trend in the past. The legend labels are placed on the top of the graph to make it crystal clear what does each colour of the line represents. Regarding the Inter activeness of the chart, the chart changes with the time frame that the user has selected. When the user hovers on the line of the chart, actual values and its corresponding legend label are displayed, the chart interface makes it easy for the user to validate actual values and demonstrate trust for the flexibility of user able to review on the historical performance of each model.

Following the chart, a buy/sell score is represented by a red-green gradient bar indicating the trend of the stock. It summarizes all the available predictions provided by different machine learning models. This provides a quick overview of the user to evaluate the trend. As it serves as a weighted average according to the trend prediction accuracy and the trend direction of each machine learning model, the red-green gradient bar simplifies all the findings and summarizes such trend predictions into an easy to interpret figure. The middle of the red-green gradient bar is coloured as white because it means the stock does not have obvious direction according to the consensus of the predictions of the machine learning models. The colour coding makes it obvious for the user to recognize the information at a glance.

The table with checkbox layout is designed for showing detailed results of each machine learning model. The user can correlate the model with its trend predictability and accuracy. This information would be useful for advanced users to evaluate which model topology or search algorithms are useful in providing insightful predictions. Although the trend scores and accuracy scores calculated are in range [0, 1], it is scaled to a 0 to 10 scoring scale, which allows easier understanding and perception. The checkbox interface allows further interactivity with the

chart display. The user could compare and contrast the recent and historical predictions of different machine learning models, and determine which of the algorithm could be as of most useful according to their definition. The table allows sorting according to trend score, trend prediction, accuracy score, and price prediction. This allows the user to prioritize the information according to the metric they would like to investigate. The table is sorted by the trend score in descending order and the best model with highest trend score is pre-selected initially, as that is presumed to be the first model that users care about.

For advanced users that have machine learning backgrounds, the application caters to their needs to look at the layers, hyperparameters, and inputs of the machine learning models. The model details page could provide insights for those users to further investigate the prediction method on their own and let them understand the underlying hypothesis of the machine learning model the application chose to include.

To let the application be smarter and more consistent in terms of user experience, user preferences, including whether a user is an advanced user and which historical predictions to view, are saved on a cloud database, updated and retrieved whenever the user logins and interacts with the preference settings.

A simple loading bar is also included for better user experience, as all data is get from the cloud, and the loading time may vary among users depending on the internet connection.

IMPLEMENTATION:

RESEARCH IMPLEMENTATION

All machine learning-related code are written in Python. Neural networks are implemented with Keras while linear regression model is implemented with scikit-learn.

STOCK PRICE DATA

Collection Data is collected from Alpha Vantage Stock Price API [18]. It offers up to 20 years of daily stock price information on S&P500 stocks. A Python script is written to retrieve stock prices of different stocks automatically. The retrieved stock prices are stored as .csv files in a local folder during development and testing. In deployment, the downloaded stock price data will be transformed into a 2D JavaScript array and uploaded to Firebase Cloud Storage immediately. A cron job that launches the data-fetching and data-uploading script is scheduled to run every 8 p.m. (EDT) after NYSE and NASDAQ are closed.

DATA PRE-PROCESSING

3 Python scripts are written to transform the raw stock prices (.csv files) into feature vectors, for training, predicting and testing respectively. The scripts take the input options and the raw stock prices as inputs and produce the correct features by building the lookback arrays and the moving averages. It concatenates the features into the final feature vectors, which will be passed to the model for training or testing. The 3 scripts share common operations in building a dataset except the output size and the range of dates to build from, so common functions are written to centralize the logic instead of repeating the same index-calculation-intensive work across functions.

NumPy and Pandas are used to build the datasets. Numpy is a library that provides effective n-dimensional array data structures as well as functions for array manipulations. It is frequently used for machine learning tasks because it is much for performant than Python lists, as Numpy arrays are implemented as densely packed lists, instead of a dynamic array where the elements are not stored contiguously.

Pandas is a popular framework for pre-processing time series data. It has various utilities for reading raw input files such as .csv and transforming time series data to the correct format. Pandas uses NumPy as the underlying data structure, so it is very convenient to interoperate between the two.

MODEL

A model base class is used as a common interface for all machine learning models. All models then have their own model class, specifying model-specific details like methods to build the model, train the model, use the model and save the model.

To decouple model configurations from software code to provide flexibility and robustness and save engineering effort, each model is defined by a JSON object, which specifies the model's architecture and hyper parameters with model options and the model inputs with input options. A corresponding model can then be created by passing the object to the model class constructor.

The model options specify which machine learning model to use, and the hyper parameters for the model like the number of hidden layers, the number of hidden units, activation functions used, as well as optimization algorithms and loss functions. Some example model options are in Appendix A.

Apart from model configurations, the input can also vary, as there are many possible features that could be added to or removed from the feature vectors. The input options specify the features input that a model should expect, like the number of previous stock prices as features and different moving averages. The input options are related to a model in terms of the input format. All neural networks built in Keras requires the input tensor shape for layer shape inference during model building, a Python function is written to calculate the input shape for a given input option. Some example input options are in Appendix B.

TRAINING

In training, a randomized initial model is first generated from the model options definition. A training set is generated by the build training dataset script, which generates the training set features from the input options and the raw stock price data. Then, the data is fed into the model for training.

SAVING TRAINED MODEL

All trained models are saved for predicting stock prices in the future. Keras models are saved in h5 format, and scikit-learn models are saved with a Python library named pickle. A dedicated saving format is designed (Appendix C), such that same models (same hash for same model options and input options) for different stocks are saved in the same directory with no collision.

PREDICTING STOCK PRICE

When predicting stock price, the saved model will first be loaded. Then, a feature vector specified by the input options is built with the build predict dataset script, which is the same as the build training dataset except it returns a flatten 1D feature vector.

The feature vector is inputted into the model to predict stock price. For 10-day predict, the predictions are directly outputted. For 1-day predict, the predicted stock price is appended to the raw dataset as if it happened before, then a new feature vector is generated for predicting the stock price for the day after, the process is repeated to predict the stock prices for all next 10 days.

PERFORMANCE EVALUATION

Each model is evaluated on the test set. A test set can be generated by the build test dataset script, which could generate either a full test set for predicting the last 100 days stock price in 1-day or 10-day disjoint intervals.

MODEL SCORE, BUY/SELL SCORE

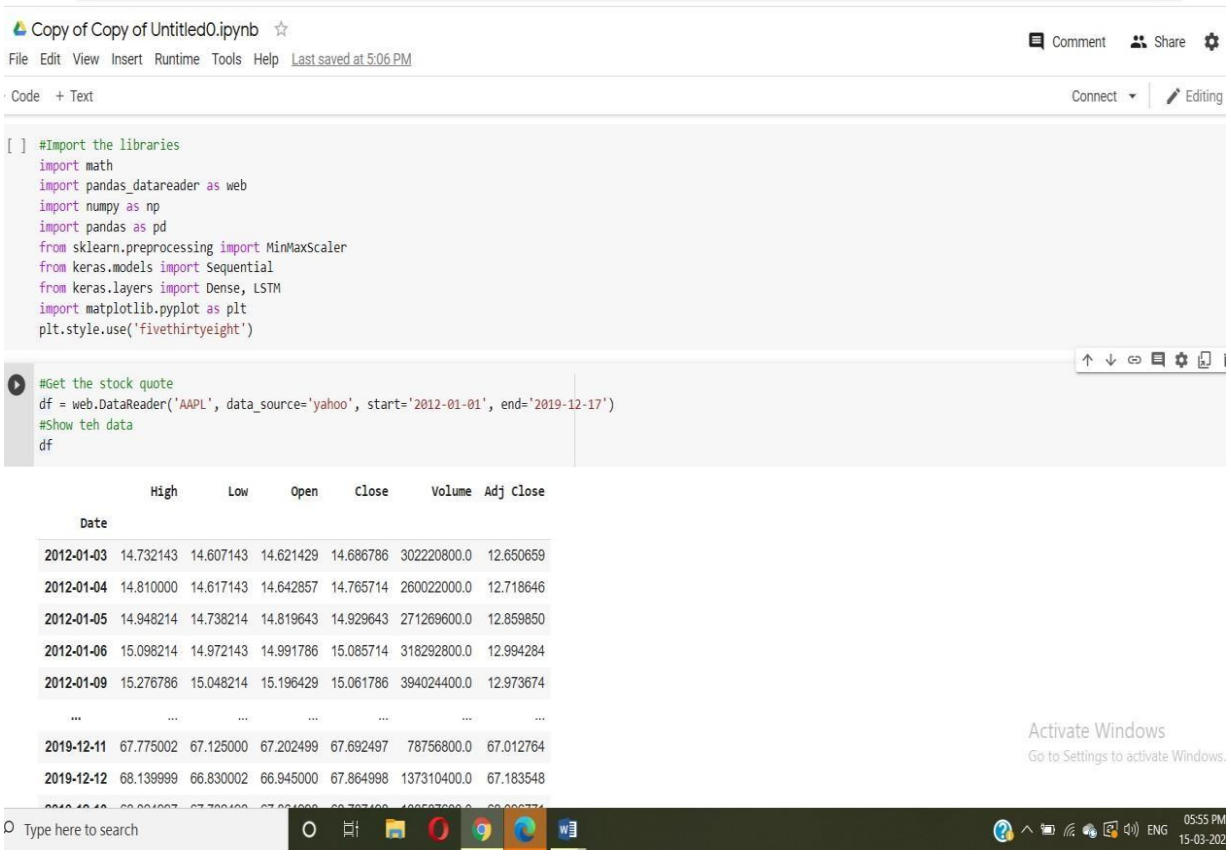
Functions are written to calculate different scores for users, 1 for calculating model trend score, 1 for model accuracy score, and 1 for buy/sell score. For parts that share the same calculation just with different offsets, helper functions are written to separate the main calculation function from the repeating steps.

SAVE PREDICTIONS

For each stock, a prediction file can be generated from the save predictions script. It includes all the data and results that the application needs to display, including all 10-day predictions from all models, both 1-day predict test set and snakes test, and the model options and input options for each model. The saved predictions file is then saved to Firebase Cloud Storage and served to the application. During development, the saved predictions file is saved in a local directory.

CHAPTER 5
RESULTS AND DISCUSSIONS

It shows the actual and the predicted closing stock price of the company Alcoa Corp, a large-sized stock. The model was trained with a batch size of 512 and 50 epochs, and the predictions made closely matched the actual stock prices, as observed in the graph.It predicted closing stock price of the company Dixon Hughes Goodman Corp, a smallsized stock. The model was trained with a batch size of 32 and 50 epochs, and while the predictions made were fairly accurate at the beginning, variations were observed after some time.We can thus infer from the results that in general, the prediction accuracy of the LSTM model improves with increase in the size of the dataset.



Copy of Copy of Untitled0.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 5:06 PM

Comment Share

Code + Text

Connect Editing

2012-01-03	14.732143	14.607143	14.621429	14.686786	302220800.0	12.650659
2012-01-04	14.810000	14.617143	14.642857	14.765714	260022000.0	12.718646
2012-01-05	14.948214	14.738214	14.819643	14.929643	271269600.0	12.859850
2012-01-06	15.098214	14.972143	14.991786	15.085714	318292800.0	12.994284
2012-01-09	15.276786	15.048214	15.196429	15.061786	394024400.0	12.973674
...	...	...	...	...	...	...
2019-12-11	67.775002	67.125000	67.202499	67.692497	78756800.0	67.012764
2019-12-12	68.139999	66.830002	66.945000	67.864998	137310400.0	67.183548
2019-12-13	68.824997	67.732498	67.864998	68.787498	133587600.0	68.096771
2019-12-16	70.197502	69.245003	69.250000	69.964996	128186000.0	69.262459
2019-12-17	70.442497	69.699997	69.892502	70.102501	114158400.0	69.398575

2003 rows × 6 columns

```
#Get the number of rows and columns in the data set
df.shape
```

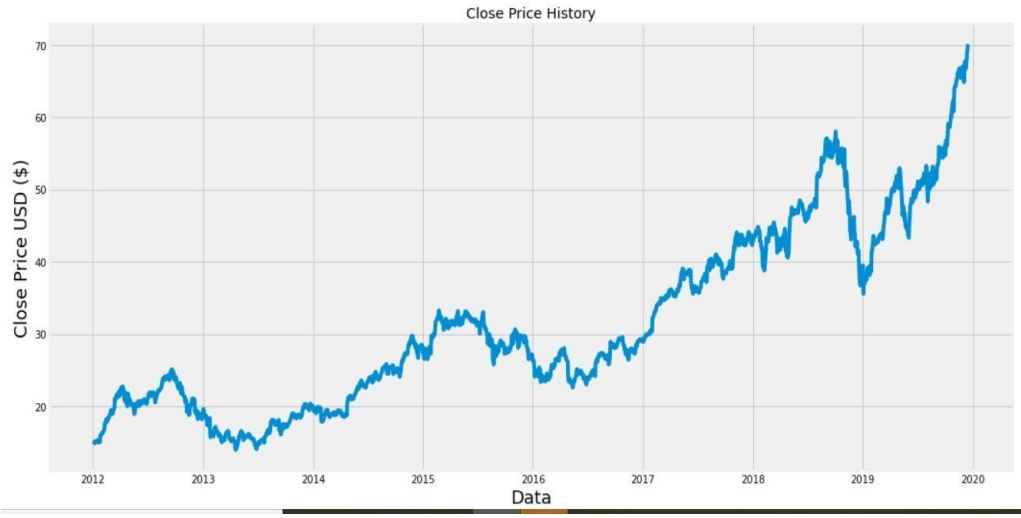
```
(2003, 6)
```

```
#Visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Data', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```


Type here to search ENG 05:55 PM 15-03-202

e + Text Cor

```
#Visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Data', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```



File Edit View Insert Runtime Tools Help Last saved at 5:06 PM

Code + Text Connect

```
#Create a new dataframe with only the Close column
data = df.filter(['Close'])
#Convert the dataframe to a numpy array
dataset = data.values
#Get the number of rows to train the model on
training_data_len = math.ceil( len(dataset) * .8 )

training_data_len
```

```
1603
```

```
#Scale the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)

scaled_data
```

```
array([[0.01316509],
       [0.01457063],
       [0.01748985],
       ...,
       [0.97658263],
       [0.99755134],
       [1.        ]])
```

```
#Create the training data set
#Create the scaled training data set
train_data = scaled_data[0:training_data_len , :]
#Split the data into x_train and y_train data sets
x_train = []
y_train = []
```

Activate Windows
Go to Settings to activate Windows.

Type here to search

Copy of Copy of Untitled0.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:06 PM

Comment Share

Code + Text Connect Editing

```
       [1.        ]])
```

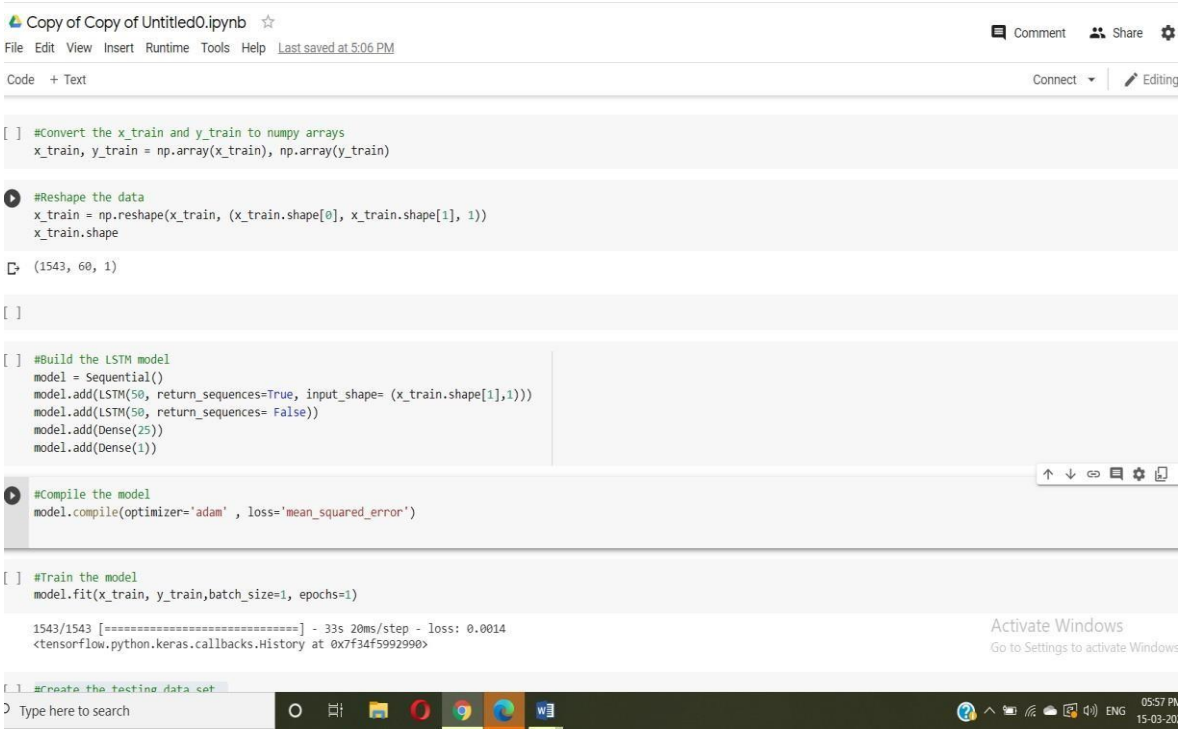
```
#Create the training data set
#Create the scaled training data set
train_data = scaled_data[0:training_data_len , :]
#Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60, len(train_data)):
  x_train.append(train_data[i-60:i, 0])
  y_train.append(train_data[i, 0])
  if i<= 60:
    print(x_train)
    print(y_train)
    print()
```

```
[array([0.01316509, 0.01457063, 0.01748985, 0.02026915, 0.01984303,
       0.02080338, 0.02036454, 0.01962679, 0.01862191, 0.02173194,
       0.02453668, 0.02367172, 0.01893355, 0.02345548, 0.01900352,
       0.03569838, 0.03440732, 0.0360927 , 0.03973694, 0.04194384,
       0.0417594 , 0.0410789 , 0.04397903, 0.04670744, 0.04979839,
       0.05479095, 0.0652785 , 0.06543749, 0.07127594, 0.07563885,
       0.06814049, 0.07102789, 0.07097066, 0.07906688, 0.07791571,
       0.08004628, 0.08387497, 0.08600558, 0.09214292, 0.09661394,
       0.09790501, 0.09835659, 0.09071194, 0.08886753, 0.08914103,
       0.09632778, 0.09835024, 0.10269409, 0.11293358, 0.12659476,
       0.12403805, 0.1240444 , 0.13392141, 0.13701237, 0.13481179,
       0.13280207, 0.13070964, 0.13766105, 0.14243103, 0.14442805])]
[0.13949272033425864]
```

Activate Windows
Go to Settings to activate Windows.

```
#Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)
```


Copy of Copy of Untitled0.ipynb
File Edit View Insert Runtime Tools Help Last saved at 5:06 PM
Comment Share
Code + Text
Connect Editing
#Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)
#Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
x_train.shape
(1543, 60, 1)
#Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1],1)))
model.add(LSTM(50, return_sequences= False))
model.add(Dense(25))
model.add(Dense(1))
#Compile the model
model.compile(optimizer='adam' , loss='mean_squared_error')
#Train the model
model.fit(x_train, y_train,batch_size=1, epochs=1)
1543/1543 [==============================] - 33s 20ms/step - loss: 0.0014
<tensorflow.python.keras.callbacks.History at 0x7f34f5992990>
Activate Windows
Go to Settings to activate Windows
Type here to search
ENG


Copy of Copy of Untitled0.ipynb
Edit View Insert Runtime Tools Help Last saved at 5:06 PM
Comment Share
+ Text
Connect
#Create the testing data set
#Create a new array containing scaled values from index 1543 to 2003
test_data = scaled_data[training_data_len - 60: , :]
#Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:]
for i in range(60, len(test_data)):
x_test.append(test_data[i-60:i, 0])
#Convert the data to a numpy array
x_test = np.array(x_test)
#Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
#Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
#Get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
rmse
2.033191595077515
Activate Windows
#Plot the data

Copy of Copy of Untitled0.ipynb ☆

Edit View Insert Runtime Tools Help Last saved at 5:06 PM

Comment Share

e + Text Connect

```
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

```
#Get the root mean squared error (RMSE)
rmse = np.sqrt( np.mean( predictions - y_test )**2 )
rmse
```

```
2.033191595077515
```

```
#Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
#Visualize the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Data', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','Val','Predictions'],loc='lower right')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

Activate Windows
Go to Settings to activate Windows

Model

Copy of Copy of Untitled0.ipynb

Edit View Insert Runtime Tools Help Last saved at 5:06 PM

Comment Share

+ Text Connect

```
#Show the valid and predicted prices
valid
```

Date	Close	Predictions
2018-05-17	46.747501	45.672462
2018-05-18	46.577499	45.579502
2018-05-21	46.907501	45.436813
2018-05-22	46.790001	45.339325
2018-05-23	47.090000	45.258682
...	...	...
2019-12-11	67.692497	62.851219
2019-12-12	67.864998	63.124947
2019-12-13	68.787498	63.400753
2019-12-16	69.964996	63.776123
2019-12-17	70.102501	64.312767

400 rows × 2 columns

```
#Get the quote
apple_quote = web.DataReader('AAPL',data_source='yahoo', start='2012-01-01', end='2019-12-17')
#Create a new dataframe
new_df = apple_quote.filter(['Close'])
#Get teh last 60 day closing price values and convert the dataframe to a array
last_60_days = new_df[-60:].values
```

Activate Windows
Go to Settings to activate Win

Type here to search

```
[ ] #Get the quote
    apple_quote = web.DataReader('AAPL',data_source='yahoo', start='2012-01-01', end='2019-12-17')
    #Create a new dataframe
    new_df = apple_quote.filter(['Close'])
    #Get teh last 60 day closing price values and convert the dataframe to a array
    last_60_days = new_df[-60:].values
    #Scale the data to be values between 0 and 1
    last_60_days_scaled = scaler.transform(last_60_days)
    #Create an empty list
    X_test = []
    #Append teh past 60 days
    X_test.append(last_60_days_scaled)
    #Convert the X_test data set to a numpy array
    X_test = np.array(X_test)
    #Reshape the data
    X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1))
    #Get the predicted scaled price
    pred_price = model.predict(X_test)
    #undo the scaling
    pred_price = scaler.inverse_transform(pred_price)
    print(pred_price)
```

```
[[64.84326]]
```

```
#Get the quote
apple_quote2 = web.DataReader('AAPL', data_source='yahoo',start='2019-12-18', end='2019-12-18')
print(apple_quote2['Close'])
```

```
Date
2019-12-18    69.934998
Name: Close, dtype: float64
```

Activat
Go to Set

Type here to search

CHAPTER 6
CONCLUSION AND FUTURE WORK

The aftereffects of examination between Long Short Term Memory (LSTM) and Artificial Neural Network (ANN) show that LSTM features a preferred forecast truth over ANN. Financial exchanges square measure troublesome to screen and need loads of setting once trying to decipher the event and foresee costs. In ANN, every secret hub is simply a hub with a solitary enactment work, whereas in LSTM, each hub could be a memory cell that may store contextual knowledge. consequently, LSTMs perform higher as they will monitor the setting express transient conditions between stock costs for a a lot of drawn out timeframe whereas performing arts forecasts. An investigation of the outcomes likewise demonstrates that the 2 models provide higher preciseness once the scale of the dataset increments. With a lot of data, a lot of examples are often full-clad by the model, and therefore the a lot of the layers are often higher modified. At its core, the securities market could be a reflection of human emotions. Pure calculation and analysis have their limitations; a attainable extension of this stock prediction system would be to enhance it with a news feed analysis from social media platforms like Twitter, wherever emotions square measure gauged from the articles. This sentiment analysis are often connected with the LSTM to raised train weights and additional improve accuracy.

Financial markets provide a unique platform for trading and investing, where trades can be executed from any device that can connect to the Internet. With the advent of stock markets, people have the opportunity to have multiple avenues to make their investment grow. Not only that, but it also gave rise to different types of funds like mutual funds, hedge funds and index funds for people and institutions to invest money according to their risk appetite. Governments of most countries invest a part of their healthcare, employment, or retirement funds into stock markets to achieve better returns for everyone.

Online trading services have already revolutionised the way people buy and sell stocks. The financial markets have evolved rapidly into a strong and interconnected global marketplace. These advancements bring forth new opportunities and the data science techniques offer many advantages, but they also pose a whole set of new challenges. In this paper, we propose a taxonomy of computational approaches to stock market analysis and prediction, present a detailed literature study of the state-of-the-art algorithms and methods that are commonly applied to stock market prediction, and discuss some of the continuing challenges in this area that require more attention and provide opportunities for future development and research. Unlike traditional systems, stock market today are built using a combination of different technologies, such as machine learning, expert systems, and big data which communicate with one another to facilitate more informed decisions. At the same time, global user connectivity on the internet has rendered the stock market susceptible to customer sentiments, less stable due to developing news, and prone to malicious attacks. This is where further research can play an important role in paving the way how stock markets will be analysed and made more robust in the future. A promising research direction is to explore various algorithms to evaluate whether they are powerful enough to predict for the longer term, because markets act like weighing machines in the long run having less noise and more predictability. Hybrid approaches that combine statistical and machine learning techniques will probably prove to be more useful for stock prediction.

FUTURE WORK:

More work on refining key phrases extraction will definitely produce better results. Enhancements in the preprocessor unit of this system will help in improving more accurate predictability in stock market.

- Twitter feeds message board, Extracting RSS feeds and news.
- Considering internal factors of the company likes Sales, Assets etc.

REFERENCES

[1] Nazar, Nasrin Banu, and Radha Senthilkumar. “An on-line approach for feature choice for classification in huge knowledge.” Turkish Journal of engineering & pc Sciences twentyfive.1 (2017): one63-171.

[2] Soulas, Eleftherios, and Dennis Shasha. “Online machine learning algorithms for currency exchange prediction.” engineering science Department in big apple University, Tech. Rep thirtyone (2013).

[3] Suresh, Harini, et al. “Clinical Intervention Prediction Understanding exploitation Deep Networks.” arXiv preprint arXiv:1705.08498 (2017).

[4] Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. “On the problem of coaching repeated neural networks.” International Conference on Machine Learning. 2013.

[5] Zhu, Maohua, et al. “Training Long remembering With Sparsified random Gradient Descent.” (2016).

[6] Ruder, Sebastian. “An summary of gradient descent optimisation algorithms.” arXivpreprintarXiv: 1609.04747.(2016).

[7] Ding, Y., Zhao, P., Hoi, S. C., Ong, Y. S. “An reconciling Gradient technique for on-line United Self-Defence Force of Colombia Maximization” In AAAI (pp. 2568-2574). (2015, January).

[8] Zhao, P., Hoi, S. C., Wang, J., Li, B. “Online transfer learning”. computer science, 216, 76-102. (2014)

[9] Hossain, Mohammad Asiful, Rezaul Karim, Ruppa K. Thulasiram, Neil D. B. Bruce, and Yang Wang. 2018. Hybrid Deep Learning Model for Stock Price Prediction. Paper presented at the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, November 18–21.

[10] Lv, Dongdong, Shuhan Yuan, Meizi Li, and Yang Xiang. 2019. An Empirical Study of Machine Learning Algorithms for Stock Daily Trading Strategy. Mathematical Problems in Engineering.

[11] Milosevic, Nikola. 2016. Equity Forecast: Predicting Long Term Stock Price Movement Using Machine Learning. arXiv.

[12] Pagolu, Venkata Sasank, Kamal Nayan Reddy, Ganapati Panda, and Babita Majhi. 2016. Sentiment Analysis of Twitter Data for Predicting Stock Market Movements. Paper presented at the 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, India, October 3–5.

[13] Peachavanish, Ratchata. 2016. Stock selection and trading based on cluster analysis of trend and momentum indicators. Paper presented at the International Multi Conference of Engineers and Computer Scientists, Hong Kong, China, March 16–18.

[14] Roondiwala, Murtaza, Harshal Patel, and Shraddha Varma. 2017. Predicting Stock Prices Using LSTM. International Journal of Science and Research (IJSR) 6: 1754–56.

[15] Seng, Jia-Lang, and Hsiao-Fang Yang. 2017. The association between stock price volatility and financial news—A sentiment analysis approach. Kybernetes 46: 1341–65.

[16] Shah, Dev, Campbell Wesley, and Zulkernine Farhana. 2018. A Comparative Study of LSTM and DNN for Stock Market Forecasting. Paper presented at the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, December 10–13.

[17] Velay, Marc, and Fabrice Daniel. 2018. Stock Chart Pattern recognition with Deep Learning. arXiv

[18] u, Yumo, and Shay B. Cohen. 2018. Stock movement prediction from tweets and historical prices. Paper Presented at the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, July 15–20.

[19] Yang, Bing, Zi-Jia Gong, and Wenqi Yang. 2017. Stock Market Index Prediction Using Deep Neural Network Ensemble. Paper Presented at the 2017 36th Chinese Control Conference (CCC), Dalian, China, July 26–28.

[20] Zhang, Jing, Shicheng Cui, Yan Xu, Qianmu Li, and Tao Li. 2018. A novel data-driven stock price trend prediction system. Expert Systems with Applications 97: 60–69.

APPENDIX:

```
#Import the libraries
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
#Get the stock quote
df = web.DataReader('AAPL', data_source='yahoo', start='2012-01-
01', end='2019-12-17')
#Show teh data
Df
#Get the number of rows and columns in the data set
df.shape
#Visualize the closing price history
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Data', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
#Create a new dataframe with only the Close column
data = df.filter(['Close'])
#Convert the dataframe to a numpy array
dataset = data.values
```

```
#Get the number of rows to train the model on
training_data_len = math.ceil(len(dataset) * .8 )
training_data_len
#Scale the data
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
#Create the training data set
#Create the scaled training data set
train_data = scaled_data[0:training_data_len , :]
#Split the data into x_train and y_train data sets
x_train = []
y_train = []
for  i  in  range(60,  len(train_data)):
  x_train.append(train_data[i-60:i,  0])
  y_train.append(train_data[i, 0])
  if i<= 60:
    print(x_train)
    print(y_train)
    print()
#Convert the x_train and y_train to numpy arrays
x_train, y_train = np.array(x_train), np.array(y_train)
#Reshape the data
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
x_train.shape
#Build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape= (x_train.shape[1],1)))
model.add(LSTM(50, return_sequences= False))
model.add(Dense(25))
model.add(Dense(1))
```

```
#Compile the model
model.compile(optimizer='adam' , loss='mean_squared_error')
#Train the model
model.fit(x_train, y_train,batch_size=1, epochs=1)
#Create the testing data set
#Create a new array containing scaled values from index 1543 to 2003
test_data = scaled_data [training_data_len - 60:]
#Create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:]
for i in range(60, len(test_data)):
x_test.append(test_data[i-60:i, 0])
#Convert the data to a numpy array
x_test = np.array(x_test)
#Reshape the data
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
#Get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
#Get the root mean squared error (RMSE)
rmse = np.sqrt( np.mean( predictions - y_test )**2 )
rmse
#Plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
#Visualize the data
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Data', fontsize=18)
```

```
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','Val','Predictions'],loc='lower right')
plt.show(
#Show the valid and predicted prices
valid
#Get the quote
apple_quote = web.DataReader('AAPL',data_source='yahoo', start='2012-01-01', end='2019-12-17')
#Create a new dataframe
new_df = apple_quote.filter(['Close'])
#Get teh last 60 day closing price values and convert the dataframe to a array
last_60_days = new_df[-60:].values
#Scale the data to be values between 0 and 1
last_60_days_scaled = scaler.transform(last_60_days)
#Create an empty list
X_test = []
#Append teh past 60 days
X_test.append(last_60_days_scaled)
#Convert the X_test data set to a numpy array
X_test = np.array(X_test)
#Reshape the data
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1))
#Get the predicted scaled price
pred_price = model.predict(X_test)
#undo the scaling
pred_price = scaler.inverse_transform(pred_price)
print(pred_price)
#Get the quote
apple_quote2 = web.DataReader('AAPL', data_source='yahoo',start='2019-12-18', end='2019-12-18')
print(apple_quote2['Close'])
```