Stock Prediction and Forecasting using Stacked LSTM-Recurrent Neural Network Model

A dissertation submitted in partial fulfilment of the requirements for the award of degree of

Master of Science
in
Applied Mathematics

by

Urmita Sharma
(2K19/MSCMAT/24)

Under the supervision of

Ms. Sumedha Seniaray
Assistant Professor

Department of Applied Mathematics
Delhi Technological University
Delhi-110042

DECLARATION

This dissertation entitled **Stock Prediction and Forecasting using Stacked LSTM-Recurrent Neural Network Model** submitted by me to the Delhi Technological University for the prize of degree of M.Sc. in Mathematics contains a survey of the work done by various authors in this area. The work presented in this dissertation has been carried out under the supervision of **Sumedha Seniaray**, Department of Applied Mathematics, Delhi Technological University, Delhi.

I hereby state that, to the best of my understanding, the work contained in this dissertation has not been submitted earlier, either in part or in full, to this or any other university/institution for the price of any degree or diploma.

Urmita Sharma
(2K19/MSCMAT/24)

ACKNOWLEDGEMENTS

I'd like to take this opportunity to express my heartfelt gratitude to my supervisor, *Ms.Sumedha Seniaray*, for his constant encouragement, co-operation and invaluable guidance throughout this dissertation. I am thankful to *Ms.Sumedha Seniaray*, Assistant Professor, DTU, during my M.Sc. Programme, for giving the facilities to fulfil my work smoothly.

I thank the teachers of the deaprtment for imparting in me the knowledge and understanding of mathematics. Without their kind efforts I would not have reached this stage. I would like to thank the staff of Central Science Library and Department of Mathematics for being cooperative.

I am thankful to my parents for being patient, supportive, understanding and for having endless belief in me. I'd like to express my gratitude to my family and friends for assisting me in every way they could and persuading me to achieve a long-held dream. I would like to thank my friends, fellow research scholars for their support during the course of my study. Above all, I thank, *The Almighty*, for all his blessings.

Urmita Sharma
(2K19/MSCMAT/24)

ABSTRACT

With its high risk and high return, the stock market is drawing more and more people's attention these days. A stock exchange market portrays savings and investments that are beneficial to the national economy's effectiveness. Future equity returns can be predicted using publicly accessible information from the current and historical stock markets.

Stock market trend prediction has also piqued the interest of statisticians and computer scientists, owing to the fact that it poses complex modelling challenges. There are methods or algorithms that can be used to forecast stock valuation with a high degree of accuracy. Because of the noise and uncertainties involved, observing and forecasting movements in the stock market price is difficult. A variety of factors, such as a country's economic shift, commodity value, investor emotions, weather, political events, and so on, can affect the market value in a single day. Artificial intelligence and increased computing power have ushered in a new era in which programmable methods of predicting market prices have proven to be more accurate. A good forecast of a stock's future price would yield a sizable profit. In this study, RNN and LSTM are combined to forecast the stock market's movement. We have proposed a deep learning- based model to make prediction more reliable and simpler. The project focuses on the use of Long Short Term Memory algorithm (LSTM), which is an advanced form of Recurrent Neural Network. We checked the accuracy of our model using stacked LSTM and forecasted the future close prices of stock data through a backtesting method by using multi-layer LSTM networks. After performing the experiment, we were able to forecast the forthcoming 10 days closing price of the given stock.

Contents

Chapter 1

Introduction

The act of attempting to forecast the future value of a company's stock or another financial instrument traded on an exchange is known as stock market prediction. A good forecast of a stock's future price could result in a large profit. A suitable stock predictions will result in huge profits for both the seller and the broker. It is often pointed out that prediction is chaotic rather than random, which is true and means it can be expected by closely examining the stock's past performance. For forecasting the future, there have traditionally been two major methods proposed. For predicting the future price of a stock, the technical analysis approach uses historical stock prices such as closing and opening prices, volume exchanged, adjacent close values, and so on. The second type of research is qualitative, which is based on external factors such as company profile, market environment, political, and economic factors. For forecasting stock prices, sophisticated intelligent techniques based on either technological or fundamental research are now used. Machine learning is a good way to describe these types of processes. It forecasts a market value that is similar to the tangible value, improving accuracy. Because of its effective and precise measurements, the application of machine learning to the field of stock prediction has piqued the interest of many researchers. The dataset used in machine learning is crucial. Since even minor changes in the data can result in massive changes in the outcome, the dataset should be as precise as possible. On a dataset obtained from Yahoo Finance, machine learning algorithms using a recurrent neural network (RNN) based long short term memory (LSTM) model, which is a deep learning approach, are used in this project. Our project focuses on the use of a deep learning-based model, which is a machine learning approach to analysis, since the historical data set collected is difficult to analyse without the use of data mining techniques. We used Yahoo Finance's Nifty 50 monthly data for the last ten years (2011-2020). In this study, we combined RNN and LSTM to test and predict the movement of stock data, and proposed an RNN-based stacked-LSTM model based on multi-layer LSTM networks.

1.1 Deep Learning

Deep learning (also known as deep structured learning) is a form of machine learning system that uses artificial neural networks to learn representations. There are three types of learning: supervised, semi-supervised, and unsupervised. In deep learning, the term "deep" refers to the use of several layers in the network. A linear perceptron cannot be a universal classifier, but a network with a nonpolynomial activation function and one hidden layer of unbounded width can, according to early research. Deep learning is a more recent variant that involves an unbounded number of layers of bounded size, allowing for functional application and optimization while maintaining theoretical universality under mild conditions.For the sake of performance, trainability, and understandability, deep learning layers are also allowed to be heterogeneous and deviate widely from biologically informed connectionist models, hence the "structured" portion.

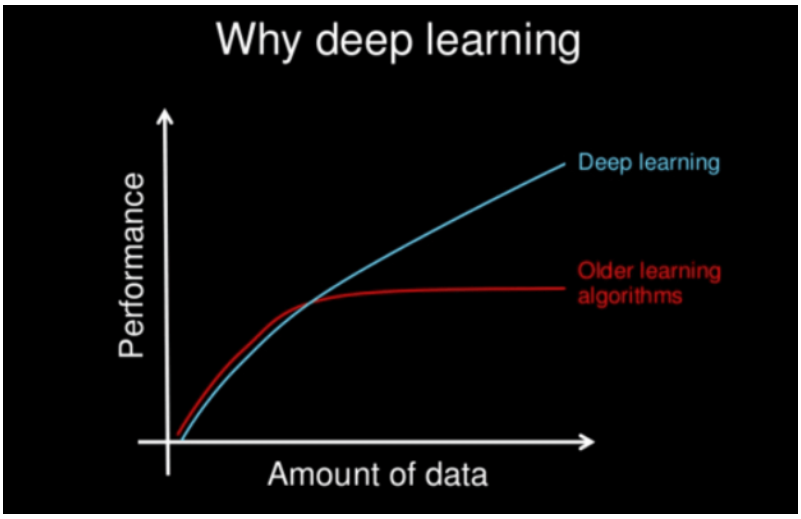


Fig.1 Performance of Deep Learning

Now we will discuss the types of deep learning neural networks learnt so far:-

1.1.1 Feedforward Neural Network

This is the most fundamental form of neural network, in which flow control starts at the input layer and moves to the output layer. These networks only have a single layer or a single hidden layer. There is no backpropagation technique in this network because data only travels in one direction. The input layer of this network receives the sum of the weights present in the input. These networks are used in the computer vision-based facial recognition algorithm.

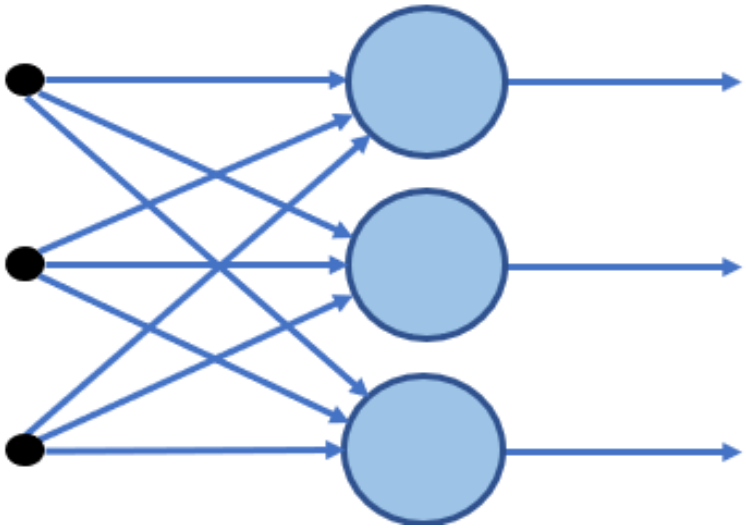


Figure 1.1: Structure of Feedforward Neural Network

1.1.2 Radial Basis function Neural Networks

This type of neural network usually has more than one layer, preferably two. In these networks, the relative distance between any two points and the middle is measured and passed on to the next layer. In order to prevent blackouts, radial base networks are commonly used in power restoration systems to restore power in the shortest time possible.

1.1.3 Multi-layer Perceptron

This form of network consists of more than three layers and is used to classify non-linear data. Every node in these networks is fully linked. Speech recognition and other machine learning technologies rely heavily on these networks.

1.1.4 Convolution Neural Network (CNN)

CNN is one of the multilayer perceptron's variants. It may have more than one convolution layer, and since it does, the network is very large and has less parameters. It is very good at recognising and recognising various picture patterns. Multilayer perceptrons are regularised variants of CNNs. Multilayer perceptrons are usually completely connected networks, meaning that each neuron in one layer is linked to all neurons in the next layer. These networks' "absolute access" makes them vulnerable to data overfitting. Regularization, or avoiding overfitting, can be accomplished in a variety of ways, including penalising parameters during preparation (such as weight loss) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach to regularisation: they take advantage of the hierarchical pattern in data and use smaller and simpler patterns embossed in their filters to assemble patterns of increasing complexity.

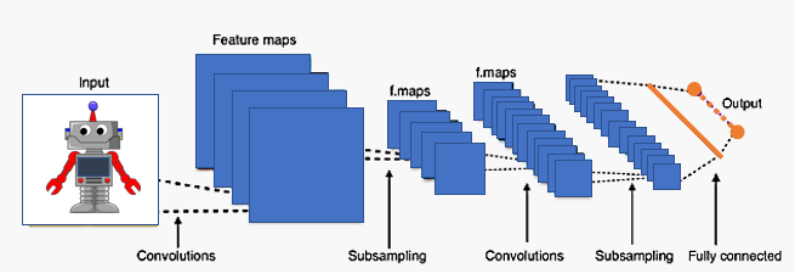


Figure 1.2: Structure of CNN

1.1.5 Modular Neural Network

A modular neural network is made up of a collection of separate neural networks that are moderated by a third party. Each independent neural network acts as a module, operating on its own set of inputs to complete a subtask of the task the network is attempting to complete. The intermediary takes each module's output and processes it to generate the network's overall output. The intermediary only accepts the outputs of the modules; it does not respond to or signal the modules in any other way. Furthermore, the modules do not communicate with one another. This network is made up of several small neural networks, rather than being a single network. The sub-networks combine to form a larger neural network, which works independently to accomplish a shared goal. These networks are extremely useful for breaking down a large-small problem into smaller chunks and then solving it.

1.1.6 Sequence to Sequence Models

In most cases, this network is made up of two RNN networks. The network is based on encoding and decoding, which means it has an encoder that processes the input and a decoder that processes the output. This type of network is commonly used for text processing where the length of the inputted text differs from the length of the outputted text.

1.1.7 Recurrent Neural Network

A RNN is a form of neural network in which a neuron's output is fed back as an input to the same node. These methods aid in the network's performance prediction. This type of network is useful for maintaining a limited state of memory, which is essential for chatbot growth. Chatbot production and text-to-speech technologies both use this type of network.A recurrent neural network (RNN) is a type of artificial neural network that uses deep learning to construct a guided graph along a temporal sequence of connections between nodes. This enables it to behave in a temporally complex manner. RNNs, which are derived from feedforward neural networks, can process variable length sequences of inputs by using their internal state (memory). As a result, tasks like unsegmented, linked handwriting recognition or speech recognition can be performed with them.

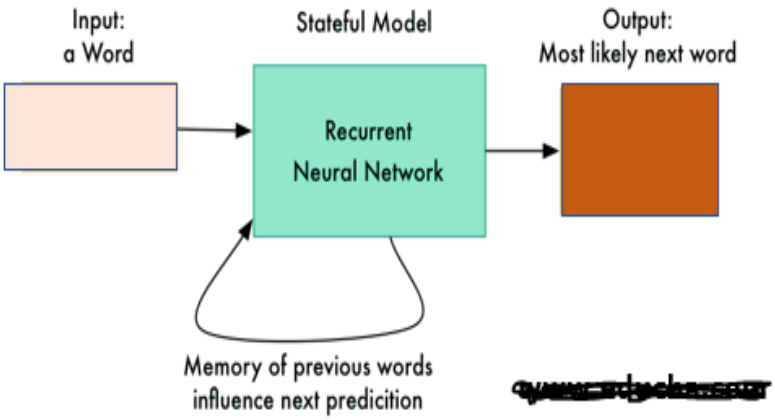


Figure 1.3: Structure of RNN

1.2 Understanding Recurrent Neural Network

Recurrent neural networks (RNNs) are a form of artificial neural network that can model sequence data. It has nodes that are linked in a graph that is powered by a time sequence. All RNNs have feedback loops in their recurrent layer. This helps them to remember information for a long time. Standard RNNs, on the other hand, can be difficult to train to solve problems involving long-term temporal dependencies. This is because the gradient of the loss function decays exponentially over time. It has the ability to remember, making them more like how humans process information and providing an effective way to solve a variety of scientific problems.

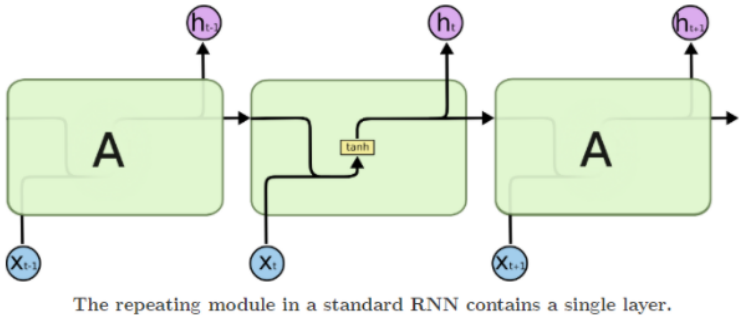


Figure 1.4: Single Layered RNN as it contains a single tanh layer. which makes it difficult to store long time memory.

1.2.1 Long-term Dependencies

Long-term dependence is a problem in RNN that arises when the network has to make a prediction that requires context. The need to understand the context can be done in a standard RNN, but this is based on how far back the memory wants to save the context information. It is possible for the RNN to predict a meaning in a simple sentence where the RNN must look back a sequence of terms. It is much more complicated in a situation where the algorithm is needed to remember a paragraph. In principle, if the parameters are set correctly by hand, this can still be done with an RNN. Fortunately, scientists do not need to change the parameters of a traditional RNN for their time data and can instead use an LSTM.

1.2.2 Stacked Long Short Term Memory (LSTM)

LSTM is a form of RNN that can take into account long-term dependability. In 1997, two physicists, Schmidhuber and Hochreiter, developed the LSTM. LSTMs are distinguished from other RNN methods by their ability to recall details. For a longer period of time to prevent long-term dependencies LSTMs have a chain structure and, like other RNNs, use gates and layers of neural networks on the inside. The LSTM's structure is built in the form of a cell state that runs through the entire device; the value is modified by gates that enable or disallow data to be applied to the cell state. There are also components known as gated cells that store information from previous LSTM outputs or layer outputs; this is where the memory element of LSTMs comes into play in 1997.
Stacked LSTM architecture is basically an LSTM Model comprised of multiple LSTM layers. Long short term memorie (LSTM) networks are a advance form of RNN. A 'memory cell' is used in LSTM modules that can store data for long periods of time. As information is stored in the memory, it is output, and it is forgotten,it is regulated by a collection of gates. New gates, such as input and forget gates, are implemented in LSTMs to solve the problem of loss function gradient, allowing for better gradient control and protection of "long term dependencies." Increase the number of repeated layers in the LSTM, which we call a stacked LSTM, to solve the RNN's long term dependence. The figure 1.5 describe the composition of LSTM nodes. While LSTMs have a chain-like structure, the repeating module is different. There are four neural network layers instead of one, each interacting in a specific way. The storage of passed data streams is the responsibility of any LSTM node, which is made up of a series of cells. Each cell's upper line serves as a transport line, carrying information from the past to the present. A tanh layer is then used to build a vector of new candidate values that could be applied to the state.Cell independence aids the model's dispose filter, which adds values from one cell to the next. Finally, the sigmoidal neural network layer containing the gates moves the cell to an optimum value by disposing or allowing data to pass through. The ability to memorise data sequences distinguishes the LSTM from other RNNs. Every LSTM node must have a collection of cells responsible for storing passed data streams; the upper line in each cell connects the models as a transport line, passing data from the past to the present; and the cells' independence aids the

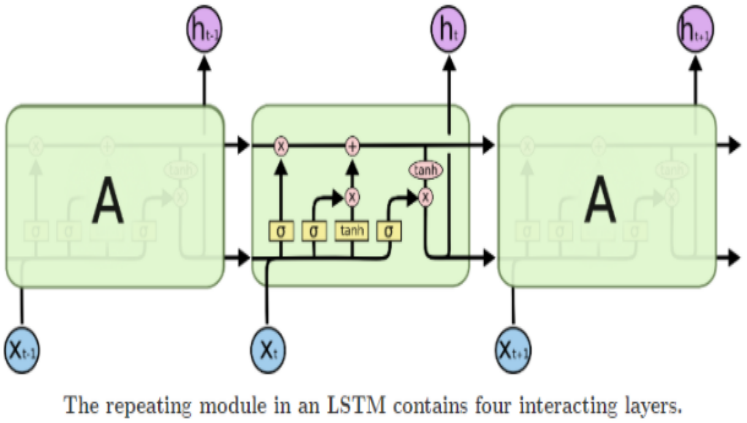


Figure 1.5: Structure of Stacked LSTM

model's disposal. Add values from one cell to another cell using a filter. Finally, the gates are powered by the sigmoidal neural network layer that makes up the gates, by discarding or allowing data to move through to an optimal value A binary value (0 or 1) is assigned to each sigmoid sheet. 0 means "allow nothing to pass through" and 1 means "allow anything to pass through." The aim is to control the state of each cell, and the gates are regulated in the following manner: - The Forget Gate generates a number between 0 and 1, with 1 denoting "fully hold this" and 0 denoting "completely miss this." - Memory Gate selects the new data to be stored in the cell. A sigmoid layer called the "input door layer" selects which values to be modified first. A tanh layer is then used to build a vector of new candidate values that could be applied to the state. - The Output Gate determines what each cell's output will be. The output value will be determined by the state of the cells, with the most recent and filtered data applied.

1.3 Machine Learning Approach

In predictive analytics, machine learning takes a different approach to a query. Predictive analytics and machine learning are expected to converge at some point in the future. It's about how thirsty people and thirsty people both come to the same glass of water. Machine learning can afford to be more versatile in its solution to a problem because it is more adaptive, younger, and has more degrees of freedom. Predictive analytics has a longer history and is more procedural in its application. There are three categories of machine learning approaches: supervised learning techniques, unsupervised learning techniques, and reinforcement approaches.

1.3.1 Supervised Learning

Supervised learning is a form of machine learning in which machines are trained using well-labeled training data and then predict the performance based on that data. The marked data indicates that some of the input data has already been

tagged with the appropriate output. In supervised learning, the training data given to the machines acts as a supervisor, instructing the machines on how to correctly predict the performance. It uses the same principle as when a student studies under the guidance of an instructor. Supervised learning can be used in the real world for things like prediction, risk assessment, image recognition, fraud detection, spam filtering, and so on. Models are trained using a labelled dataset in supervised learning, where the model learns about each type of data. The model is evaluated on test data (a subset of the training set) after the training phase is completed, and it then predicts the performance. There are numerous ways to execute supervised learning in Python, such as scikit-learn, which incorporates scipy libraries. Other libraries, such as Tensorflow and Keras, are also included. Examples of machine learning classifiers include Naive Bayes, Support vector machine, KNN, RNN and others.

1.3.2 Unsupervised Learning

Unsupervised learning is a machine learning methodology that does not use a training dataset to supervise models. Models, on the other hand, use the data to uncover secret trends and observations. It is comparable to the learning that occurs in the human brain when learning new information.Since, unlike supervised learning, we have the input data but no corresponding output data, unsupervised learning cannot be directly applied to a regression or classification issue. Unsupervised learning aims to uncover a dataset's underlying structure, group data based on similarities, and display the dataset in a compressed format. Unsupervised learning is divided into two categories: clustering and association.

1.3.3 Reinforcement Learning

The training of machine learning models to make a series of decisions is known as reinforcement learning. In an unpredictable, potentially complex world, the agent learns to achieve an objective. An artificial intelligence faces a game-like scenario in reinforcement learning. To find a solution to the problem, the machine uses trial and error. Artificial intelligence is given either incentives or punishments for the actions it takes in order to get it to do what the programmer wants. Its aim is to increase the overall reward as much as possible. Reinforcement learning differs from supervised learning in that it does not require the presentation of labelled input/output pairs or the explicit correction of sub-optimal behaviour. Instead, seeking a balance between discovery (of uncharted territory) and exploitation is the priority (of current knowledge).

1.4 Data Mining

Companies use data mining to turn raw data into usable information. Businesses can learn more about their customers by using software to search for trends in large batches of data. This allows them to create more successful marketing campaigns, boost revenue, and cut costs. Effective data collection, warehousing, and computer processing are all needed for data mining.

1.4.1 How Data Mining Works

Exploring and analysing vast blocks of data to find relevant patterns and trends is what data mining is all about. It can be used for a range of purposes, including database marketing, credit risk management, fraud detection, spam email filtering, and even determining user sentiment. There are five phases in the data mining process. Organizations begin by gathering data and loading it into data warehouses. The data is then stored and managed, either on-premises or in the cloud. Data is accessed by business managers, management teams, and information technology experts, who then decide how to arrange it.The data is then sorted by application software based on the user's responses, and the data is eventually presented in an easy-to-share format, such as a graph or table, by the end-user.

1.5 Statistics

1.5.1 Mean Squared Error (MSE)

The sum of the squares of the errors—that is, the average squared difference between the expected values and the real value—is calculated by the mean squared error (MSE) of an estimator (of a method for estimating an unobserved quantity) in statistics. MSE is a risk function that represents the squared error loss's expected value. Another way for calculating the accuracy and error in predictive models is to use MSE.

$$MSE = \frac{1}{n}\sum_{n=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

$\hat{Y}_i$ is the i^{th} predicted value and Y_i is the i^{th} actual/observed value.

Figure 1.6: Formula of MSE

1.5.2 Mean Absolute Error (MAE)

The mean absolute error (MAE) is a statistic that measures the difference in errors between paired observations describing the same phenomenon. Comparisons of expected versus observed, subsequent time versus initial time, and one measurement technique versus another measurement technique are examples of Y versus X.

$$\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}.$$

Figure 1.7: Formula of MAE

1.6 Predictive Analytics

Predictive analytics is a statistical methodology that uses statistics (both historical and current) to forecast, or 'predict,' future outcomes. It includes a number of statistical techniques (including machine learning, predictive modelling, and data mining). These results may include, for example, consumer behaviours or market shifts. Through analysing the past, predictive analytics will help us understand potential future events. Machine learning, on the other hand, is a branch of computer science that gives computers the ability to learn without being specifically programmed, according to Arthur Samuel's 1959 description. Machine learning is a branch of pattern recognition that investigates the idea that algorithms can learn from and predict data. These algorithms can also transcend software instructions to make highly accurate, data-driven decisions as they become more "intelligent."

1.6.1 How does Predictive Analytics work?

Predictive modelling is at the heart of predictive analytics. It's more of a strategy than a process. Since predictive models generally have a machine learning algorithm, predictive analytics and machine learning go hand in hand. These models can be trained to react to new data or values over time, producing the results that the company requires. Machine learning and predictive modelling are closely related fields. Predictive models are divided into two categories. There are two types of models: classification models, which predict class membership, and regression models, which predict a numerical value. Algorithms are then used to build these models. Data mining and statistical analysis are carried out by the algorithms, which identify trends and patterns in the data. Built-in algorithms in predictive analytics software solutions can be used to build predictive models. The algorithms are referred to as "classifiers" because they determine one of many categories data belongs to.

1.7 Applications of Predictive Analytics

Predictive analytics and machine learning will help organisations that have a lot of data but are having trouble turning it into valuable insights. No matter how much data an organisation has, if it can't use it to improve internal and external processes and achieve goals, it's a waste of time. The most popular applications of predictive analytics are security, marketing, operations, risk, and fraud detection. Here are a few examples of how machine learning and predictive analytics are used in various industries:

1.7.1 Banking and Financial Services

Predictive analytics and machine learning are used together in the banking and financial services industry to detect and mitigate fraud, quantify market risk, identify opportunities, and much more.

1.7.2 Security

With cybersecurity at the top of every company's agenda in 2017, it's no wonder that predictive analytics and machine learning are important components of protection. Predictive analytics is commonly used by security organisations to optimise services and efficiency, as well as to detect anomalies, fraud, better understand customer behaviour, and strengthen data security.

1.7.3 Retail

Predictive analytics and machine learning are being used by retailers to better understand customer behaviour: who buys what and where? With the right predictive models and data sets, these questions can be easily answered, allowing retailers to prepare ahead and stock products based on seasonality and customer preferences, dramatically improving ROI.

1.7.4 Financial Market

Based on current data, prediction analysis techniques can forecast future possibilities. The estimation of the stock market is a difficult problem in prediction research. The stock prices fluctuate at a very consistent pace, which adds to the data set's complexity.

Chapter 2

Literature Review

In this chapter, we will go over various literature surveys by various authors. Many researchers have made significant contributions in this field. Researchers in prediction analysis has conducted several studies on stock prices prediction using various methodologies. Here, we will go over some studies that will help us learn more about sentiment analysis and how it relates to the core subject of our dissertation.

Kunal Pahwa et.al. [1] proposed the use of regression and supervised learning techniques for stock prediction. They took 14 years GOOGL historical data from WIKI and used linear regression classifier(a supervised learning method). They trained the classifier to learn the pattern, find accuracy and predicted the adj. close price.

V.Kranthi Sai Reddy [2] proposed a machine learning technique called support vector machine (SVM)-based train stock data and forecasted stock prices. The prediction of close prices is based on IBM Inc.'s historical data. They proposed the method for predicting the regular trend of stocks since SVM does not have the problem of overfitting

Adil Moghar and Mhamed Hamiche [3] proposed an RNN and LSTM model to forecast future stock market prices. They used yahoo finance to extract regular open price data for GOOGL and NKE on the New York Stock Exchange (NYSE). They trained the model with 80% of the data and then tested it with 20% of the data. Later, they demonstrated how much the epochs enhanced their model by demonstrating accuracy on 12;25;50;100 epochs, respectively on both stocks separately.

Kai Chn, Yi Zhou,et.al. [4] proposed an LSTM approach for predicting stock returns and conducted a case study of the Chinese stock market. Their model was fitted using 900000 training sequences and evaluated using the remaining 311361

sequences. They later demonstrated that the accuracy of the LSTM model had improved.

Hossein Abbasimehra,Mostafa Shabani et.al. [5] proposed a demand forecasting approach based on a multi-layer LSTM network. The method uses the grid search method to consider different combinations of LSTM hyperparameters in order to choose the best forecasting model for a given time series. They then used demand data from a furniture manufacturer to equate their model to existing time-series forecasting techniques such as Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA).

Adrian Costea [6] discussed how to evaluate the financial performance of Romanian non-banking financial institutions (NFIs) by using fuzzy logic techniques such as Fuzzy C-Means Clustering and Artificial Neural Network (ANN) techniques. The researcher used an ANN technique and genetic algorithms to find a function that converts the input performance space into a new performance class variable

Mehar Vijh, Deeksha chandola et.al. [7] propsed the Artificial Neural Network (ANN) and Random Forest (RF) techniques to forecast the next day's closing price for five firms. They collected data of Nike, Goldman Sachs, Johnson Johnson, Pfizer, and JP Morgan Chase Co. over a 10-year period. They demonstrated how ANN outperforms RF in terms of prediction accuracy by using the metrics RMSE and MAPE and displaying their low values.

Chapter 3

Problem Definition

We gathered a 10 years monthly historical data of Nifty 50 from 2011 to 2020. We took 10 years Nifty 50 monthly data (2011-2020) from Yahoo finance. For testing and predicting the movement of the stock data, we combined RNN and LSTM and proposed an RNN-based stacked-LSTM model based on multi-layer LSTM networks in this analysis. The LSTM algorithm, which is a more advanced variant of RNN, looks for the best hyperparameters for LSTM networks. After checking the accuracy of our stacked LSTM model by showing low values of Mean Squared Error (MSE) Mean Absolute Error (MAE) we forecasted the future 10 days closing price of our data via a backtesting method. Since we tested our LSTM model on the current dataset and predicted the future value, the forecast as a result of our proposed algorithm may help people decide whether to invest in a particular company while taking into account the chaos and volatility of the stock.

3.1 Motivation

The stock market is one of the earliest methods for a regular citizen to exchange stocks, make investments, and profit from businesses that sell a piece of themselves on this site. As a result, techniques for forecasting stock prices beforehand by analysing the trend over the some previous years have been established which may prove to be extremely useful for making stock price movements in order to maximize profit and reduce losses. In the past, two key methods for forecasting an organization's stock price were suggested. Fundamental and Technical analysis are the two main components of stock market analysis:- The method of assessing a company's future profitability based on its current business environment and financial performance is known as fundamental analysis. Technical analysis, on the other hand, involves analysing statistical data and reading charts to assess stock market trends. The emphasis of this paper is technical analysis. We all know that, stock market is a vital trading medium that has an effect on everyone on a personal and national level. The basic theory is straightforward: businesses would list their stock as commodities in minimal quantities known as stocks. They do it to help the

organisation to raise money. The IPO, or initial public offering, is when a company sells its shares at a set price. This is the price at which a company sells its stock to raise money. The stock then becomes the owner's property, and he can sell it to anyone at any time on a stock exchange like the BSE or Bombay Stock Exchange. Traders and sellers are continuing to sell these shares for a profit at their own dime. However, the company only retains when the IPO proceeds. The endless jumping of shares from one party to the next in order to make more money causes the price of a specific share to increase with each successful sale. The exchange's share price falls if the company issues more stock at a lower IPO price, and buyers and sellers lose money. In a nutshell, this occurrence is the cause of people's distrust of investing in capital markets, as well as the rise and fall in stock prices. Stock market trend prediction has also piqued the interest of statisticians and computer scientists, owing to the fact that it poses complex modelling challenges. There are methods or algorithms that can be used to forecast stock valuation with a high degree of accuracy. However, one question remains: what are the chances that an individual buying shares from a specific company would turn out to be a profitable venture or a complete failure? It could be fine to invest in a specific stock if an educated guess is made on a larger scale, taking into account the organization's current production, sales, and demand. However, expecting this to function in dynamic situations when ignoring such nuanced business concepts and variables is unrealistic.

3.2 Objective

The main objective of this thesis is to do prediction analysis of 10 years stock market data and on the basis of accuracy of model we forecast the next 10 days future close price value. Since our target value in the prediction is close price.
As a result, in order to achieve this aim, we will develop a model to forecast the market price of the data we have chosen. And this model will help in forecasting the further value by using backtesting method.

3.3 Stock Market Dataset

We collected our data from Yahoo Finance. Since 2017, Verizon Media has owned Yahoo Finance as a media property. It is part of the Yahoo! network. In addition to financial news, information, and analysis, it provides stock quotes, press releases, financial reports, and original material. It also has several personal finance management software available online. It publishes original articles from its team of staff journalists in addition to partnering material from other websites. On the ranking of the largest news and media websites, SimilarWeb ranks it 15th.

Chapter 4

Methodology and Implementation

In this thesis, A model was implemented using the RNN-based Keras system and trained using data about the index's price to measure the accuracy of the stacked LSTM model proposed for predicting the Nifty 50 close price. This chapter also explains how the LSTM model's different hyperparameters were selected. For testing and predicting the movement of the stock data, we combined RNN and LSTM and proposed an RNN-based stacked-LSTM model based on multi-layer LSTM networks in this analysis. The LSTM algorithm, which is a more advanced variant of RNN, looks for the best hyperparameters for LSTM networks. After checking the accuracy of our stacked LSTM model by showing low values of Mean Squared Error (MSE) Mean Absolute Error (MAE) we forecasted the future 10 days closing price of our data via a backtesting method. We used ADAM to train and refine our model. Since we tested our LSTM model on the current dataset and predicted the future value, the forecast as a result of our proposed algorithm may help people decide whether to invest in a particular company while taking into account the chaos and volatility of the stock. We trained with 80% of the data and tested with 20% of the data.

4.0.1 Proposed Architecture (Overview)

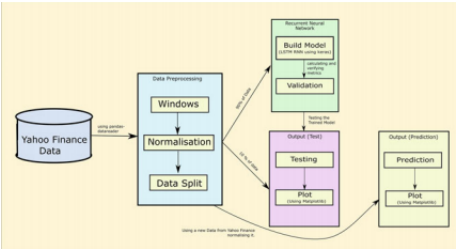


Figure 4.1: Architeture of the proposed model

4.1 Tools

4.1.1 Python

In this project, Python will be used. Python is a programming language for advanced users. It's a flexible and strong programming language. Since an interpreter is used, there is no need to compile the code in Python, making testing and debugging far faster. Python has a range of open-source libraries. It's a really common programming language. As a consequence, it can be used in web creation, software development, and system programming, among other things. It can run on R, Raspberry Pi, and Windows, among other platforms. Python's syntax is close to that of English, allowing programmers to write fewer lines of code than for other programming languages.

4.1.2 Anaconda and Jupyter Notebook

Anaconda is nothing more than a series of well-known Python packages. Conda, a packet manager, is also included (similar to pip). This Python library is well-known in the data science community. Numpy, scipy, jupyter, nltk, scikit-learn, and other common packages include numpy, scipy, jupyter, nltk, scikit-learn, and others. A variety of Python libraries are included in Anaconda. The anaconda mini is a lighter weight version of the anaconda. Anaconda also replaces their own kit, known as conda. It's a lot more efficient than PIP. Jupyter is a web-based interpreter for Python and R that allows you to communicate with them. Anaconda is a collection of Jupyter notebook libraries. Jupyter can be thought of as a digital notebook where you can run commands, draw maps, and take notes. The data scientist used this as a starting point. If you're learning Python and R, this is a fantastic resource. Shell is much inferior to jupyter.
Jupyter is an excellent tool for analytical work because it allows you to display your code in "modules," add common formatting options between modules, and integrate formatted output from modules into other modules' code. Jupyter means that other people's work can be replicated. As a consequence, if anyone returns after a few months, looking at the code, he or she can quickly understand what was attempted. And can figure out which code is in control of which conclusion or visualisation.

4.2 Nifty 50

The NIFTY 50 is an Indian stock market index that represents a weighted average of 50 of India's largest companies that are listed on the National Stock Exchange. It is one of India's two main stock indices, with the BSE SENSEX being the other. The Nifty 50 is owned and managed by NSE Indices (previously known as India Index Services Products Limited), a wholly owned subsidiary of the NSE Strategic Investment Corporation Limited. Until 2013, NSE Indices and Standard Poor's had a co-branding equity indices marketing and licencing agreement. On April 22, 1996, the Nifty 50 index, one of Nifty's many stock indexes, was unveiled. The NIFTY

50 index has evolved to become India's largest single financial commodity, with an ecosystem that includes exchange-traded funds (onshore and offshore), exchange-traded options at NSE, and futures and options abroad at the SGX. The NIFTY 50 is the world's most commonly traded deal. WFE, IOMA, and FIA polls have all supported NSE's leadership position. The NIFTY 50 index is a free float market capitalization-weighted index. At first, the index was generated using a full market capitalization method. On June 26, 2009, the computation was changed to a free-float process. The NIFTY 50 index has a base date of 3 November 1995, which is the end of the first year of activity of the National Stock Exchange Equity Market Segment. With a base capital of 2.06 trillion dollars, the index's base value is 1000.

4.2.1 Trading Strategy

A trading strategy can be built using a variety of tools, including technical indicators and fundamental indicators. There are two types of fundamental indicators: bottoms up and tops down. The basic company metrics are the bottom-up parameters, and the top down parameters are how fundamental parameters are interpreted, such as the amount of money the corporation earns and the amount of debt it owes. The top-down perspective is concerned with how the economy as a whole is changing; for example, if the BNP per capita in Sweden increases, we can expect that people in Sweden have more disposable income and that businesses selling consumer goods would increase profits. Our trading strategy is based on a technical approach that may or may not yield the best returns, as backtesting has shown that a hybrid strategy combining fundamental and technological methods yields the best outcomes. Our aim, however, is not to achieve the best trading outcomes, but to learn more about the technology that underpins them. We use indicators like the close price and historical price to find buy and sell signals in the trends, which is why our approach is strictly technical.

4.3 Data Description

The project is based on 10 years monthly historical data of Nifty 50 from 2011 to 2020 which was downloaded from Yahoo Finance.
Following are the terms that are important:-
Open Price- The price of a business day's first purchase.
Close Price- After the trading hours of the exchange where it trades, the last price paid for a share of that stock.
High- For a given time period, it is the highest price.
Low- The cheapest price for a given period of time.
Adj. Close- After any market decisions have been taken into account, the adjusted closing price adjusts a stock's closing price to reflect its worth.
Volume- In the sense of a single share of stock that is traded on a stock exchange, the number of shares exchanged in a security.

Before fitting our model, we begin by analysis by calculating descriptive of the data.

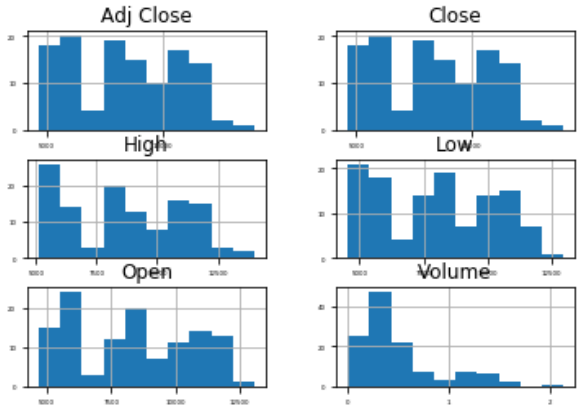


Figure 4.2: Histogram of all the columns of dataframe showing the frequency of values

	Open	High	Low	Close	Adj Close	Volume
count	120.000000	120.000000	120.000000	120.000000	120.000000	1.200000e+02
mean	8318.447901	8605.974602	8007.274183	8369.182907	8369.182907	4.643092e+06
std	2298.343767	2373.322006	2258.921885	2343.851499	2343.851499	4.227346e+06
min	4675.799805	5099.250000	4531.149902	4624.299805	4624.299805	0.000000e+00
25%	5964.800049	6134.824951	5699.025146	5923.925171	5923.925171	2.718475e+06
50%	8300.875000	8619.274903	7935.474853	8304.975097	8304.975097	3.673600e+06
75%	10450.662602	10900.550050	10098.324952	10502.312257	10502.312257	5.044475e+06
max	13062.200200	14024.849610	12962.799810	13981.750000	13981.750000	2.130300e+07

Figure 4.3: Table summarizing the results of descriptive analysis of Nifty 50 Data

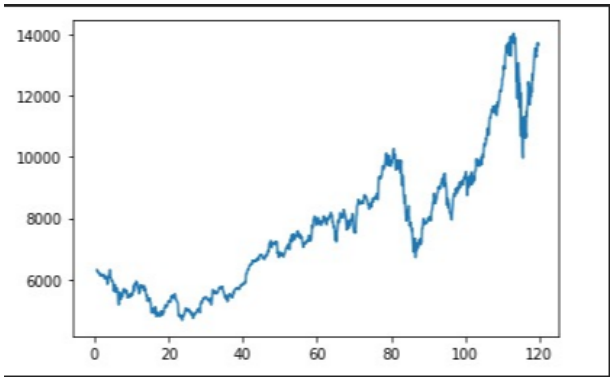


Figure 4.4: Close Price of Dataframe

The project's main aim is to forecast potential data near prices using our proposed model, so our target value is close price. The last price at which the stock exchanged during a normal trading day is the closing price. The standard benchmark by which investors calculate a stock's success over time is its closing price. Cash dividends, stock dividends, and stock splits would not be reflected in the closing price.

4.4 Data Preprocessing

Data discretization, data transformation, and data cleansing are all part of the preprocessing level. Data cleansing and data integration are two of the most important aspects of data management. To analyse, The dataset is split into two sections: training and test after it has been transformed into a clean dataset. We search for non-applicable possibilities by cleaning our files, and then we switch to feature scaling, for which we imported min-max scalar from scikit-learn, a python machine learning library. Min-Max is a technique for transforming features by scaling each one to a specific range between 0 and 1. So, before we can do anything with data, we must first perform data preprocessing, also known as data cleaning.

- We search for non-applicable possibilities by cleaning our files, and then we move on to feature scaling, for which we imported min-max scalar from scikit-learn, a python machine learning library.
- Min-Max is a technique for transforming features by scaling each one to a specific range between 0 and 1.
- Creating a data structure consisting of 10 time steps and one output. We used data from day 1 to day 10 to make predictions on the 11th day, and data from day 2 to day 11 to make predictions on the 12th day.

4.4.1 Splitting dataset into train and test split

We have split our dataset in 80:20 ratio in which we have trained 80% data and tested our model prediction on 20% data of the dataset.A huge amount of training data leads to a more powerful and reliable classifier, which improves overall precision. Testing is often a very simple procedure. Ascertain that your evaluation data that is at least 20% larger than or equal to our training data. Testing is a measure of the classifier's accuracy, and it's been found that testing is inversely proportional to a classifier's score on occasion.

4.5 Feature Extraction

The dimensionality reduction method divides and reduces a wide set of raw data into smaller groups, and feature extraction is a step in that process. As a consequence, retrieval will be more straightforward. The most significant characteristic of these big data sets is the huge number of variables. A large amount of computational power is needed to process these variables. As a consequence, by selecting and merging variables into functions, feature extraction helps in the extraction of the best feature from large data sets, effectively reducing the amount of data. These features are easy to use while also detailing the data collection process correctly and uniquely.

4.5.1 Building RNN

For building our RNN, we have imported the Keras library and packages. We imported sequential, dense, LSTM, and dropout libraries from Keras, which is a tensor flow API for creating and draining deep learning models at a high level. The dense layer in the input simply represents a The dense layer was used to perform matrix vector multiplication, and the matrix vector multiplication matrix vector multiplication matrix vector multiplication matrix vector multiplication matrix reshape our output's proportions The term "sequential" refers to a layer, a stack that allows you to construct a sequential model from scratch. It can be used to move a list through it.

4.5.2 Initialising RNN

To use a Recurrent Neural Network (RNN) for time series modelling, the network must be properly initialised, that is, the hidden neuron outputs must be set correctly at the start. An RNN is usually started with zero state values or at steady state. Such initializations mean the system to be modelled is in steady state in the sense of dynamic system recognition, i.e., capturing transient behaviour of the system is difficult if the network states are not properly initialised. If the network initial states cannot be calculated from the training data, a method for inferring them is needed, both during the training and validation phases. Now, the first step in creating a deep learning model is to read the data and then allocate it to the model.

4.5.3 Building Stacked LSTM

Three LSTM layers follow a sequential input layer in this LSTM: a dense layer with activation, a dense output layer with linear activation, and a dense output layer with linear activation. Dropouts are used to strengthen neurons, allowing them to function better. Without depending on a single neuron, you can predict the pattern.

4.5.4 Layers

4.5.4.1 Hidden Layer

When building the LSTM model, we must consider the number of hidden layers the model would include, the number of LSTM cells that should be included in each layer, and the dropout. There is no right or wrong way to choose the number of hidden layers or the number of cells inside each layer; we've seen models with 4 layers and 1000 cells and 3 layers and 2 cells that were both successful. The number of cells and layers depends on the application for which the LSTM model would be used; however, the layers are usually 1 to 5, and the cells in each layer should contain the same number of cells for finding an optimal structure.

4.5.4.2 Dense Layer

A dense layer is a densely connected NN layer (Keras), in which each cell in the next layer is connected by a dense layer. We've seen effective models that use dense layers by first creating a model of hidden layers, then adding several dense layers.

4.5.4.3 Number of Layers

We've agreed on four layers for our LSTM model: two hidden layers and two thick layers. The first hidden layer's output is connected to another hidden layer, which is then connected to a dense layer, which is then connected to another dense layer. To avoid the possibility of overfitting, dropouts are used after each hidden layer.

4.5.5 Optimal Hyperparameters

There are hyperparameters that must be properly setup and modified when constructing the LSTM model so that we can get an accurate prediction when backtesting our model. This enables us to understand how we can conduct empirical tests to identify the best hyperparameters that will increase accuracy while reducing the possibility of overfitting the data, as opposed to not conducting empirical tests on our model and data. When conducting the empirical test, we will construct our LSTM model using default hyperparameters that best suit our case, as determined by various papers that we find to be extremely useful. After that, we'll go through each hyperparameter one by one and try to find an optimal value for each one. The best value for a hyperparameter is discovered by evaluating the LSTM model using the test data and backtesting it.

4.5.5.1 Dropout

Dropout is a useful technique for reducing overfitting by selecting cells in a layer at random based on the likelihood chosen and setting their output to 0. The optimum amount of dropout was determined through an empirical test, which was then extended to all of the hidden layers. We developed and trained our LSTM model, then varied the dropout values so that the difference between consecutive dropout values remained constant. Dropouts are used to strengthen the neurons, enabling them to predict the pattern without relying on any one neuron.

4.5.5.2 Number Of Epochs

An epoch occurs when the entire training data set has been transferred through the network; hence, one epoch corresponds to one iteration of the entire training data set. When the training data is propagated through the network, we divide it into a batch size, which we set to 64. The epoch will continue until all samples have been propagated across the network, at which point one epoch will have been passed.

4.5.6 Compiling the RNN

Now, we will compile our RNN to using ADAM optimizer and loss as mean squared error which kept on decreasing when we fit our LSTM model with epochs=100, batch size=64 and verbose=1. Here,the number of epochs means how many times we go through the training set,batch size which is the hyperparameter defines the number of samples to work through before updating the internal parameters and verbose which helps to detect overfitting which happens when our model's accuracy keeps on improving.

Parameter Overview:-

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           10400
_________________________________________________________________
dropout (Dropout)            (None, 100, 50)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 50)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 100, 50)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 100)               60400
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 111,301
Trainable params: 111,301
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.5: Table shows the summary of stacked LSTM model used.

We used 4 LSTM layers with input shape-100, hidden size-50, dropout rate-0.2 and output layer-1. For training data, we used 100 epochs, batch size 64 (a gradient descent hyperparameter that specifies how many training samples must be processed before the model's internal parameters are changed), and verbose-1 (which includes both a progress bar and one line per epoch). The number of epochs is a gradient descent hyperparameter that establishes how many complete passes through the training dataset are made. The results of our research revealed that the number of epochs as well as the length of the data have a significant impact on the testing outcome after our RNN has been trained. We saw a steady decrease in validation loss as well as the values of MSE and MAE after each epoch. The accuracy of our LSTM model was then assessed using the mean squared error (MSE) and mean absolute error (MAE) as metrics.

Table 4.1: Values of MSE & MAE

Epoch Size	**MSE**	**MAE**
Epoch 1	0.1588	0.3429
Epoch 100	0.0057	0.0567

Table 4.1 shows the reduction of values of MSE & MAE from epoch 1 to epoch 100. The total epoch size was 100. The value of these two metrics kept on decreasing from running epoch 1 to epoch 100. Hence, it shows that the accuracy of our model is improving.

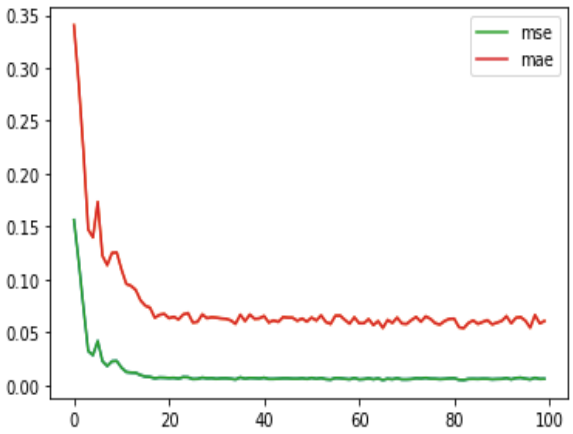


Figure 4.6: Decrease in the values of MSE MAE

Graph in the figure 4.6 clearly shows the decrease in the values of MSE and MAE after training our LSTM model at each epoch. x-axis and y-axis in the graph depicts the number of epoches and value of mse mae, respectively. The low values shows the accuracy and improvement of our proposed model.

4.5.7 Optimisation

Optimization is the process of selecting the best element from a set of alternatives based on some criteria. All quantitative disciplines, from computer science and engineering to operations analysis and economics, have optimization problems, and the development of solution methods has been of interest in mathematics for centuries.Since it's nearly not possible to build a flexible classifier in a one pass, we should constantly optimise. Optimisation,in context of deep learning, is used to train the neural networks.
Here, we used the optimizer Adam to create the LSTM model because it has a high performance and quick convergence compared to other optimizers. Adam is a deep learning deep learning model training algorithm that replaces stochastic gradient descent. Adam incorporates the best features of the AdaGrad and RMSProp algorithms to build an optimization algorithm for noisy problems with sparse gradients. We have used ADAM(Adaptive Movement Estimation) optimiser with a learning rate of 0.0005. Adam is a deep learning model training algorithm that uses stochastic gradient descent instead of stochastic gradient descent. It combines the best features of the AdaGrad and RMSProp algorithms to create an optimization algorithm for problems with sparse gradients and noisy data.

4.5.8 Regularization

Another important aspect of training the model is to keep the weights from being too high. As a result, there are overfits. Regularization is a set of techniques for preventing overfitting in neural networks and, as a result, improving the accuracy of a Deep Learning model when confronted with entirely new data from the problem

domain.
We have chosen Tikhonov Regularization for this reason which regularised all the parameters equally. Tikhonov regularisation is a basic technique for converting a linear discrete ill-posed problem into a least squares problem.

Chapter 5

Result And Analysis

We will make a prediction based on past data to make a prediction of future history data to see how reliable our LSTM model is. This will be accomplished by feeding the LSTM model the most recent prediction, which will be the training set's most recent entry results. Our findings have led to some intriguing conclusions. Initially, we wanted to see how accurate a stacked LSTM model was at predicting the stock market. We used a variety of statistical methods to assess this precision, but most importantly, we focused on the backtesting method, which is considered an industry norm in the scientific community when it comes to developing predictive models. When the model is done, we use it to produce the desired results. In our case, we'll make a graph of our findings based on our criteria and requirements that we've already covered in this paper.We trained 80% data and tested on 20% data which is clearly shown in the graph below.

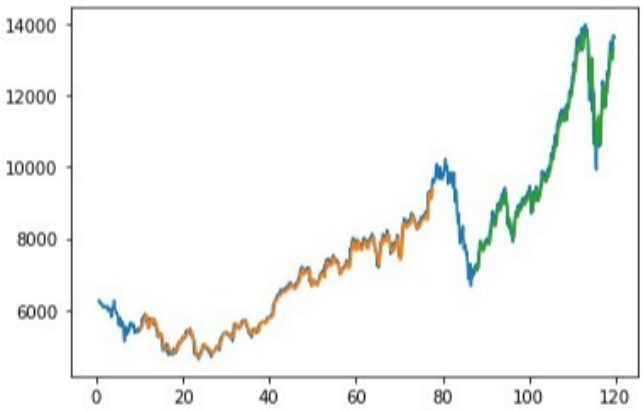


Figure 5.1: Test predict of Stock price of Nifty 50 from 2011-2020 using Stacked LSTM.

In figure 5.1, graph shows the prediction of our model on available dataset. Blue Line represents our actual given close price data. Orange Line represents our train

data of close price. Green Line represents the predicted output of test data of close price. We have divided our data in such a way that test data will be after some specific date.The graph clearly shows that our stacked LSTM model has worked well and has performed accurately on the available dataset. Every result's most important feature is its precision. Accuracy is a component that any machine learning developers always strives to improve. Following the development of the model, an endless amount of work is expended in order to improve the model's accuracy.Our graph exactly shows the accuracy of our model.
Now, we will predict next future 10 days output. We used backtesting data to run our LSTM algorithm to forecast.We have seen our model is giving accurate results we can now rely on the model for forecasting.We have taken past 10 days data as an input from our dataset and forecasted the next future 10 days close price.

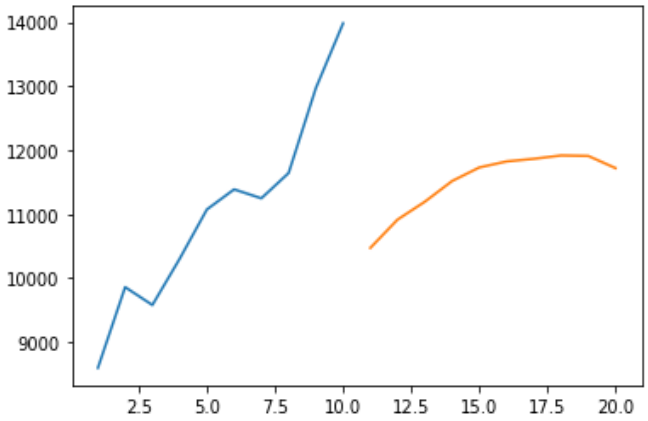


Figure 5.2: The forecast of next 10 days.

In figure 5.2, graph shows the future prediction. Blue Line represents previous 10 days data. Orange Line represents next 10 days output.Here, x-axis depicts the input values of past data and y-axis depicts the output values we want to be foreseen.
Now, we combine the past graph with the future forecasted graph and we can see that we are getting a smoothen graph.

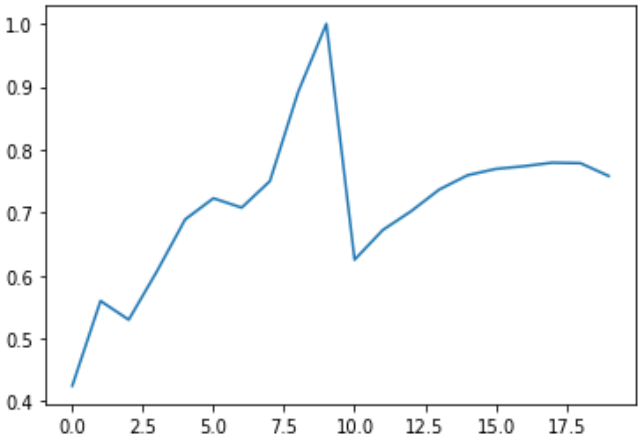


Figure 5.3: Combined graph with future 10 days output prediction.

In figure 5.3, graph shows that the stacked LSTM model has done an incredible work of forecasting the performance for next 10 days. The implementation of this model is the easiest of them all, and it took the least amount of time, saving us time that could be spent on other important tasks.
From the above predictions till now we can see that how promising results our proposed RNN based stacked LSTM model has given. Given time lags of uncertain length, stacked LSTM is well-suited to identify, process, and forecast the stock data. LSTM has an advantage over alternative models, secret Markov models, and other sequence learning methods due to its relative insensitivity to gap length. RNN has a structure that is very similar to that of a hidden Markov model.

Chapter 6

Conclusion

This project proposes an RNN-based stacked LSTM model for predicting future values for Nifty 50 stock data downloaded from Yahoo Finance over a 10-year period. We checked our model's precision on existing 80% data and tested it on 20%data, then displayed the accuracy using MSE and MAE values. Since our target value is near price and our model is delivering promising results, we have optimised it using ADAM optimiser and forecasted the market data's next 10 day future closing price via backtesting process. The main aim here is to find the most skilled algorithm for predicting potential values that is as accurate as possible. The results of the experiments show that our model has generated some promising outcomes. The results of the research revealed that our model can use the backtesting process to monitor the evolution of nearby prices. This method has resulted in improved prediction accuracy and positive outcomes. We demonstrated that our proposed model performed well on available data, allowing us to forecast the next 10 days' near price of our current dataset.To determine this accuracy, we used a deep learning approach, but most importantly, we concentrated on the backtesting process for potential near price prediction. Other methods for improving the accuracy of this model in the future, such as using bi-directional LSTM and optimising a hyperparameter to combat overfitting when backtesting the results, may improve the model's accuracy even further. Other machine learning models may also be examined to see what level of accuracy they achieve. There are many components that we think other researchers can incorporate into their theoretical frameworks for potential work in this area. The first step will be to optimise a hyperparameter in order to avoid overfitting when backtesting the results. The second part of this is to always make a forecast and compare it to real-time results. The third factor is to construct a hybrid model in which we use sentiment analysis, for example, to forecast the volatility of the index price because we found that the prediction error was higher when the index price was volatile. We would have taken into account more variables than just historical data on the index's price.

References

[1] Kunal Pahwa, Neha Agarwal,"Stock Market Analysis using Supervised Machine Learning", 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (Com-IT-Con), India, Feb 2019.

[2] V Kranthi Sai Reddy,"Stock Market Prediction Using Machine Learning",International Research Journal of Engineering and Technology (IRJET), Oct 2018 .

[3] Adil MOGHAR, Mhamed HAMICHE,"Stock Market Prediction Using LSTM Recurrent Neural Network",International Workshop on Statistical Methods and Artificial Intelligence (IWSMAI 2020) April 6-9,2020,Warsaw,Poland.

[4] Kai Chen,Yi Zhou,Fangyan Dai,"A LSTM-based method for stock returns prediction : A case study of China stock market",2015 IEEE International Conference on Big Data (Big Data).

[5] Hossein Abbasimehra,Mostafa Shabanib,Mohsen Yousefic, "An optimized model using LSTM network for demand forecasting",Computers Industrial Engineering 143 (2020) 106435.

[6] Adrian Costea,"Applying fuzzy logic and machine learning techniques in financial performance predictions",7th International Conference on Applied Statistics.

[7] Mehar Vijha, Deeksha Chandola, Vinay Anand Tikkiwal, Arun Kumar,"Stock Closing Price Prediction using Machine Learning Techniques", International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

[8] Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan, "Stock Market Prediction Using Machine Learning", December 2018.

[9] Xinyi Li,Yinchuan Li,Hongyang Yang,Liuqing Yang,Xiao-Yang Liu,"DP-LSTM: Differential Privacy-inspired LSTM for Stock Prediction Using Financial News".33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

[10] Shubharthi Dey,Yash Kumar,Snehanshu Saha,Suryoday Basak,"Forecasting to Classification: Predicting the direction of stock market price using Xtreme Gradient Boosting",October 2016.

[11] Ha Daewoo,Kim Youngmin,Jaejoon Ahn,"A Study on the Prediction of the Rise and Fall of the KOSPI 200 Stock Index Using the XGBoost Model",Journal of the Korean Data Information Science Society, vol.30(3), pp. 655-669,2019.

[12] Gers, Felix A., Nicol N. Schraudolph, Jurgen Schmidhuber,"Learning precise timing with LSTM recurrent networks", Journal of Machine Learning Research, 2002.

[13] Abadi, Martin,"Deep learning with differential privacy", Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016

[14] Ashish Pathak, Nisha P Shetty,"Indian Stock Market Prediction using Machine Learning and Sentiment Analysis", December 2017.

Appendix A

Appendix

A.1 Python Code

```
In [2]: import pandas as pd
        import matplotlib.pyplot as plt
        import pandas as pd
        import datetime
        import math
```

```
In [3]: dataset = pd.read_csv('C:/Users/HP/Desktop/Dissertation/Nifty50.csv',index_col="Date",parse_dates=True)
```

```
In [3]: dataset.head()
```

Out[3]:

Date	Open	High	Low	Close	Adj Close	Volume
2011-01-01	6177.450195	6181.049805	5416.649902	5505.899902	5505.899902	0
2011-01-02	5537.299805	5599.250000	5177.700195	5333.250000	5333.250000	0
2011-01-03	5382.000000	5872.000000	5348.200195	5833.750000	5833.750000	0
2011-01-04	5835.000000	5944.450195	5693.250000	5749.500000	5749.500000	0
2011-01-05	5766.899902	5775.250000	5328.700195	5473.100098	5473.100098	0

```
dataset.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	120.000000	120.000000	120.000000	120.000000	120.000000	1.200000e+02
mean	8318.447901	8605.974602	8007.274183	8369.182907	8369.182907	4.643092e+06
std	2298.343767	2373.322006	2258.921885	2343.851499	2343.851499	4.227346e+06
min	4675.799805	5099.250000	4531.149902	4624.299805	4624.299805	0.000000e+00
25%	5964.800049	6134.824951	5699.025146	5923.925171	5923.925171	2.718475e+06
50%	8300.875000	8619.274903	7935.474853	8304.975097	8304.975097	3.673600e+06
75%	10450.662602	10900.550050	10098.324952	10502.312257	10502.312257	5.044475e+06
max	13062.200200	14024.849610	12962.799810	13981.750000	13981.750000	2.130300e+07

```
dataset.isna().any()
```

```
Open          False
High          False
Low           False
Close         False
Adj Close     False
Volume        False
dtype: bool
```

```
dataset.shape
```

```
(120, 7)
```

```
dataset=dataset.reset_index()['Close']
```

```
dataset
```

```
0         5505.899902
1         5333.250000
2         5833.750000
3         5749.500000
4         5473.100098
             ...
115      11387.500000
116      11247.549810
117      11642.400390
118      12968.950200
119      13981.750000
Name: Close, Length: 120, dtype: float64
```

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
dataset=scaler.fit_transform(np.array(dataset).reshape(-1,1))
```

```
print(dataset)
```

```
[[0.09421371]
 [0.07576318]
 [0.12924997]
 [0.12024645]
 [0.0907085 ]
 [0.10933535]
 [0.09165961]
 [0.04025671]
 [0.03408516]
 [0.07505253]
 [0.02220156]
 [0.        ]
 [0.06144304]
 [0.08131493]
 [0.07173429]
 [0.06666881]
 [0.03205469]
 [0.06995496]
 [0.06462233]
```

.pdf” .png” .jpg” .mps” .jpeg” .jbig2” .jb2” .PDF” .PNG” .JPG” .JPEG” .JBIG2” .JB2” .eps”

```
import numpy
# convert an array of values into a dataset matrix
def create_dataset(dataset, time_step=1):
    dataX, dataY = [], []
    for i in range(len(dataset)-time_step-1):
        a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100
        dataX.append(a)
        dataY.append(dataset[i + time_step, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

```
# reshape into X=t,t+1,t+2,t+3 and Y=t+4
time_step = 10
X_train, y_train = create_dataset(train_data, time_step)
X_test, ytest = create_dataset(test_data, time_step)
```

```
print(X_train.shape), print(y_train.shape)
```

```
(85, 10)
(85,)
(None, None)
```

```
print(X_test.shape), print(ytest.shape)
```

```
(13, 10)
```

```
# reshape input to be [samples, time steps, features] which is required for LSTM
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)
```

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from keras.layers import Dropout
```

```
model=Sequential()
model.add(LSTM(50,return_sequences=True,input_shape=(100,1)))
model.add(Dropout(0.2))
model.add(LSTM(50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=100))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error',optimizer='adam', metrics=['mse', 'mae'])
```

```
model.summary()
```

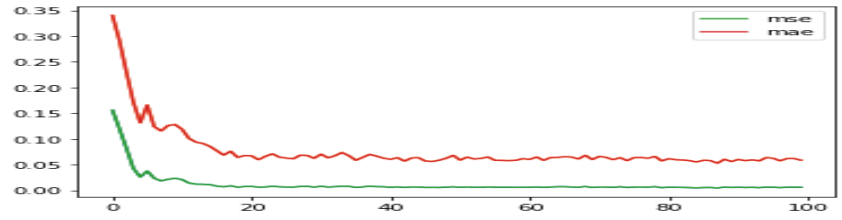
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           10400
_________________________________________________________________
dropout (Dropout)            (None, 100, 50)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 50)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
dropout_2 (Dropout)          (None, 100, 50)           0
_________________________________________________________________
lstm_3 (LSTM)                (None, 100)               60400
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 111,301
Trainable params: 111,301
Non-trainable params: 0
_________________________________________________________________
```

```
history=model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)
```

```
# plot metrics
import matplotlib.pyplot as plt
plt.plot(history.history['mse'])
plt.plot(history.history['mae'])
plt.plot(history.history['mse'],label='mse')
plt.plot(history.history['mae'],label='mae')

plt.legend()

plt.show()
```




```
model.input_shape
```

```
(None, 100, 1)
```

```
model.output_shape
```

```
(None, 1)
```

```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.4.1'
```

```
### Lets Do the prediction and check performance metrics
train_predict=model.predict(X_train)
test_predict=model.predict(X_test)
```

```
WARNING:tensorflow:Model was constructed with shape (None, 100, 1) for input KerasTensor(type_spec=TensorSpec(shape=(None, 100, 1), dtype=tf.float32, name='lstm_input'), name='lstm_input', description="created by layer 'lstm_input'"), but it was called on an input with incompatible shape (None, 10, 1).
```

```
##Transformback to original form
train_predict=scaler.inverse_transform(train_predict)
test_predict=scaler.inverse_transform(test_predict)
```

```
### Plotting
# shift train predictions for plotting
look_back=10
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = np.nan
trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(train_predict)+(look_back*2)+1:len(dataset)-1, :] = test_predict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.plot(trainPredictPlot,label='trainPredictPlot')
plt.plot(testPredictPlot,label='testPredictPlot')
plt.legend()
plt.show()
```

```
len(test_data)
```

```
24
```

```
x_input=test_data[14:].reshape(1,-1)
x_input.shape
```

```
(1, 10)
```

```
temp_input=list(x_input)
temp_input=temp_input[0].tolist()
```

```
temp_input
```

```
[0.42462958521789923,
 0.5595114563150501,
 0.5296314590750542,
 0.6067678359681612,
 0.6891995426752504,
 0.7227610143855006,
 0.7078049967649331,
 0.7500013827217598,
 0.8917654084291922,
 1.0]
```

```
# demonstrate prediction for next 10 days
from numpy import array

lst_output=[]
n_steps=10
i=1
while(i<=10):

    if(len(temp_input)>10):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1
```

```
11
2 day input [0.55951146 0.52963146 0.60676784 0.68919954 0.72276101 0.707805
 0.75000138 0.89176541 1.         0.64395994]
2 day output [[0.6894193]]
3 day input [0.52963146 0.60676784 0.68919954 0.72276101 0.707805   0.75000138
 0.89176541 1.         0.64395994 0.68941933]
3 day output [[0.71869093]]
4 day input [0.60676784 0.68919954 0.72276101 0.707805   0.75000138 0.89176541
 1.         0.64395994 0.68941933 0.71869093]
4 day output [[0.75105125]]
5 day input [0.68919954 0.72276101 0.707805   0.75000138 0.89176541 1.
 0.64395994 0.68941933 0.71869093 0.75105125]
5 day output [[0.7721143]]
6 day input [0.72276101 0.707805   0.75000138 0.89176541 1.         0.64395994
 0.68941933 0.71869093 0.75105125 0.77211428]
6 day output [[0.7816567]]
7 day input [0.707805   0.75000138 0.89176541 1.         0.64395994 0.68941933
 0.71869093 0.75105125 0.77211428 0.78165668]
7 day output [[0.7861241]]
8 day input [0.75000138 0.89176541 1.         0.64395994 0.68941933 0.71869093
 0.75105125 0.77211428 0.78165668 0.78612411]
8 day output [[0.79162514]]
9 day input [0.89176541 1.         0.64395994 0.68941933 0.71869093 0.75105125
 0.77211428 0.78165668 0.78612411 0.79162514]
9 day output [[0.79181117]]
10 day input [1.         0.64395994 0.68941933 0.71869093 0.75105125 0.77211428
 0.78165668 0.78612411 0.79162514 0.79181117]
10 day output [[0.77582085]]
[[0.6439599394798279], [0.6894193291664124], [0.7186909317970276], [0.7510512471199036], [0.7721142768859863], [0.7816566824913
025], [0.7861241102218628], [0.7916251420974731], [0.7918111681938171], [0.7758208513259888]]
```

```
day_new=np.arange(1,11)
day_pred=np.arange(11,21)
```

```
import matplotlib.pyplot as plt
```

```
len(dataset)
```

```
120
```

```
df3=dataset.tolist()
df3.extend(lst_output)
```

```
plt.plot(day_new,scaler.inverse_transform(dataset[110:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))

plt.xlabel('x-axis')
plt.ylabel('y-axis')

plt.show()
```