STOCK PRICE PREDICTION USING DEEP NEURAL NETWORKS

A MAJOR PROJECT REPORT

Submitted by

P.S.S.GOPICHAND

A. MOHAN SAI

T. ANU

Under the Guidance of

Mr.K.Sambath Kumar

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS & COMMUNICATION ENGINEERING

Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

JUNE 2022

Vel Tech
Rangarajan Dr. Sagunthala
R&D Institute of Science and Technology
(Deemed to be University Estd. u/s 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this major project report entitled **"STOCK PRICE PREDICTION USING DEEP NEURAL NETWORKS "** is the bonafide work of **" P.S.S.GOPI CHAND (18UEEC0346), A.MOHANSAI (18UEEC0016) and T.ANU (18UEEC0457)"**. who carried out this work under my supervision.

SUPERVISOR

Mr.K.Sambath Kumar
Professor
Department of ECE

HEAD OF THE DEPARTMENT

Dr.P.ESTHER RANI
Professor
Department of ECE

Submitted for Evaluation of Major Project Review held on:_____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected Founder President and Chancellor **Col. Prof. Dr. R. Rangarajan**, Foundress President **Dr. R. Sagunthala Rangarajan**, Chairperson, Managing Trustee and Vice President.

We are very thankful to our beloved Vice Chancellor **Prof. Dr. S. Salivahanan** for providing us with an environment to complete the work successfully.

We are obligated to our beloved Registrar **Dr. E. Kannan** for providing immense support in all our endeavours. We are thankful to our esteemed Dean Academics **Prof. Dr. M. J. Carmel Mary Belinda** for providing a wonderful environment to complete our work successfully.

We are extremely thankful and pay gratitude to our Dean/SOEC **Dr. V. Jayasankar** for his valuable guidance and support on completion of this Major project.

It is a great pleasure for us to acknowledge the assistance and contributions of our Head of the Department **Prof. Dr. P. Esther Rani** for her useful suggestions, which helped us in completing the work in time and we thank her for being instrumental in the completion of final year (8th sem) with her encouragement and unwavering support during the entire course.

We are extremely thankful and pay gratitude to our supervisor **Prof Mr.K.Sambath Kumar** for his valuable guidance and support on completing this Major project report in pleasant form.

We thank our department faculty, supporting staffs and our parents for encouraging and supporting us throughout the study to complete this Major project report.

P S S GOPICHAND

A MOAHNSAI

T ANU

TABLE OF CONTENTS

ABSTRACT

It is a research hotspot in academic circles and the financial world to work out how financial activities are patterned and to predict their developments and changes. It is extremely challenging to predict financial data development trends due to their incomplete, complex and fuzzy nature. The stock market attracts millions of investors on a daily basis. An effective stock price prediction model can assist managers, investors, and decision makers in making appropriate and effective decisions. There are a number of constantly changing factors that affect financial data fluctuations. Forecasting and analyzing financial data are therefore nonlinear and time-dependent problems.

Deep neural networks (DNNs) is better choice for solving nonlinear problems than traditional machine learning algorithms. In this project, for predicting stock prices, a DNN-based prediction model is built on top of long- and short-term memory networks (LSTMs). Rather than predicting stock prices approximately, Deep Neural Networks can forecast them with high degree of accuracy. In addition, we propose to do a comparative analysis of the proposed model with the different methods using the simulation results obtained.

Keywords: Deep Neural Networks, Long- and Short-term Memory.

LIST OF FIGURES

LIST OF ABBREVIATIONS

NN	Neural Networks
DL	Deep Learning
DNN	Deep Neural Networks
LSTM	Long-Short-Term Memory
RNN	Recurrent Neural Networks
AI	Artificial Intelligence
ANN	Artificial Neural Networks
NLP	Natural Language Processing
CAP	Credit Assignment Path
MLP	Multi Layer Perceptron
CNN	Concolutional Neural Networks
IPO	Initial Public Offering

CHAPTER 1

INTRODUCTION

Investments in stocks have the advantage of fulfilling various investment desires and needs, expanding investment options, expanding investment channels and enabling investors to obtain income corresponding to their investment goals. Stocks also offer some flexibility and liquidity to investors. Stocks possess three major characteristics: (1) nonreturnability: once sold, a stock may not be returned to the company or refunded. Instead, it can only be sold to a third party through the secondary market. (2)Earnings are unpredictable: in order to earn a profit or lose a loss on the stock market, investors must take greater risks as the stock exchange market and the business operations of companies are uncertain and changeable. (3) Speculative: It is risky to speculate on the stock market because its price fluctuates frequently. There are high risks associated with stock markets because prices are inherently unstable and speculative. In recent years, the number of financial activities has grown rapidly with rapid economic development, and the variation trend has also become increasingly complicated. Trading in stock market can be done physically or electronically. When an investor buys a company stock, this mean that this investor becomes an owner of the company according to the ownership percentage of this company‘s shares. This give the stockholders rights on the company‘s dividends. Financial data of stock market is of complex nature, which makes it difficult to predict or forecast the stock market behavior. Research in academic and financial circles is focused on understanding financial patterns and predicting their development. A method or set of methods that provide an approximate prediction of financial data can be used to analyze the development and changes of the financial market at the macro level and allows investors to make trading decisions and plan trades at the microscopic level, they can maximize profits. Because It is extremely difficult to predict the development trend of financial data due to their incomplete, ambiguous, and complex nature. Using historical stock trading data, news, and investor sentiment, this paper implements a stock trend prediction system that allows investors to reduce or avoid risky investments by recommending stocks and predicting market trends. There are a number of machine learning algorithms that have been employed with good results in forecasting, such as logistic regression, genetic algorithms, and support vector machines. Researchers have begun to take deep neural networks into account in order to portray stock prices and predict stock movement patterns as a result of the rise of neural network

technology. Stock trend prediction has received numerous improvements over the years in terms of algorithms and optimizations, all of which have been successfully utilized in practice.

The previous stock trend prediction algorithms use the historic time series stock data. The typical scientific stock price forecasting procedures are focused on the statistical analysis of stock data. In the paper will develop a stock data predictor program that uses previous stock prices and data will be treated as training sets for the program to predict the stock prices of a particular share this program develops a procedure.

The use of Neural Networks(NNs) to predict time series has been widely used in the analysis of inaccurate and noisy data. With the newly emerging deep learning (DL) algorithms, large volumes of nonlinear data can be trained to create deep NNs (DNNs) with many hidden layers capable of capturing abstract nonlinear relationships. In this project, we propose to use Long and Short-Term Memory(LSTM) for the prediction of stock prices. Long-short-term memory (LSTM) node-based Recurrent Neural Networks (RNN) models are effective and expandable when used to solve various problems dealing with series data. The Long Short Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of addressing linear problems. LSTM is a deep learning technique. Long-term Memory (LSTM) Units are enforced to learn very long sequences. This is a more general version of the gated recurrent system. LSTM is more benign than other deep learning methods like RNN or traditional feed forward because LSTMs tackle the evanescent gradient issue.

In the areas of handwriting recognition, concise natural language translation, and audio frequency analysis, these models have achieved exceptional results and are applicable in general use.

This model considers the historical equity share price of a company price and applies RNN (Recurrent) technique called Long Short Term Memory (LSTM). The proposed approach considers available historic data of a share and it provides prediction on a particular feature. The features of shares are Opening price, day High, day Low, previous day o price, Close price, Date of trading, Volume and adjacent close. The proposed model uses the time series analysis in order to predict a share price for a required time span.

Additionally, a deep neural network stock trend prediction system is also developed, and the trained prediction model is uploaded to the stock prediction module. After development and testing, the whole stock trend prediction system is completed by analyzing the requirements of the system and developing each of its functional modules. With the help of this system, investors can make stock selections and investments.

1.1 DEEP LEARNING

Deep learning is a neural network with three or more layers, which is a subset of machine learning. It allows the network to discover patterns in large amounts of data by attempting to simulate the system's behavior - although it is far from matching the ability of the human brain. Additional hidden layers can help to optimize and refine neural networks for accuracy by improving their accuracy even with a single layer.

Applications and services driven by artificial intelligence (AI) rely on deep learnings to automate tasks and perform analytical functions without human intervention. Technology like deep learning (such as virtual assistants, voice-activated TV remote controls, and fraud detection for credit cards) is used in many everyday products and services, as well as in emerging technologies (such as self-driving cars).

In artificial neural networks, or deep learning neural networks, data inputs, weights, and bias are combined to simulate the human brain. Objects in the data are accurately recognized, categorized, and described using these elements. An integrated deep neural network consists of multiple layers of nodes, each of which refines and optimizes prediction or categorization by building upon the previous layer. This process is known as forward propagation. The visible layers of a deep neural network contain the input and output layers. The input layer holds the data that is processed by the deep learning model, while the output layer holds the prediction or classification that is made.

This type of process adjusts the weights and biases of the function as it moves backwards through the layers in order to train a model using algorithms like gradient descent. Using both forward propagation and back propagation, neural networks can accurately predict and correct any errors. As the algorithm gets more accurate over time, it becomes more precise.

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. This helps consumer electronics meet user expectations, and it is crucial for safety-critical applications like driverless cars. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

There are two main reasons it has only recently become useful: Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less.

Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks.

The term “deep” usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 150.

Deep learning models are trained by using large sets of labeled data and neural network architectures that learn features directly from the data without the need for manual feature extraction.

Deep learning applications are used in industries from automated driving to medical devices.

Automated Driving: Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

Aerospace and Defense: Deep learning is used to identify objects from satellites that locate

areas of interest, and identify safe or unsafe zones for troops.

Medical Research: Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells.

Industrial Automation: Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

Electronics: Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

1.1.1 Deep Learning VS. Machine Learning

By studying a wide variety of data and by applying different learning methods, deep learning differs from classical machine learning.

A machine learning algorithm uses structured, labeled data to make predictions, i.e., the input data is organized into tables defining specific features from the data. In other words, the resulting structured data is not necessarily unstructured; if it is, preprocessing is usually applied to make it more structured.

In contrast to machine learning, deep learning does not require a lot of data preprocessing. Feature extraction is automated through these algorithms, allowing them to ingest and process unstructured data, such as text and images, and remove some of the dependence on humans.Say, for instance, we have a collection of photos of pets that we'd like to categorize by "cat", "dog", "hamster", etc. Deep learning algorithms determine what characteristic must be present in an animal to distinguish it from others (e.g. ears). Human experts establish this hierarchy of features in machine learning. As a result of gradient descent and backpropagation, the deep learning algorithm increases its accuracy and adapts to a new photo of an animal so that it can make predictions about it with greater accuracy. There are several types of learning that machine learning and deep learning models can do, such as supervised learning, unsupervised learning, and reinforcement learning. To make predictions or categorize input data, supervised learning uses labeled datasets; for this to work, some human intervention must be incorporated. Unsupervised learning, on the other hand, ignores the labeled datasets and instead is based on discovering patterns within data and clustering it accordingly. In reinforcement learning, a model improves its accuracy based upon feedback so that it optimizes its reward when performing an action.

1.2 DEEP NEURAL NETWORKS

The human brain is the inspiration behind Deep Neural Networks. Deep Neural Network software predicts and provides solutions far beyond the "if and else" conditions. The output of Deep Neural Network AI can be obtained without producing any code or programming. Learning and experiences are the basis for conclusions (like the human brain). The use of deep neural networks in multiple

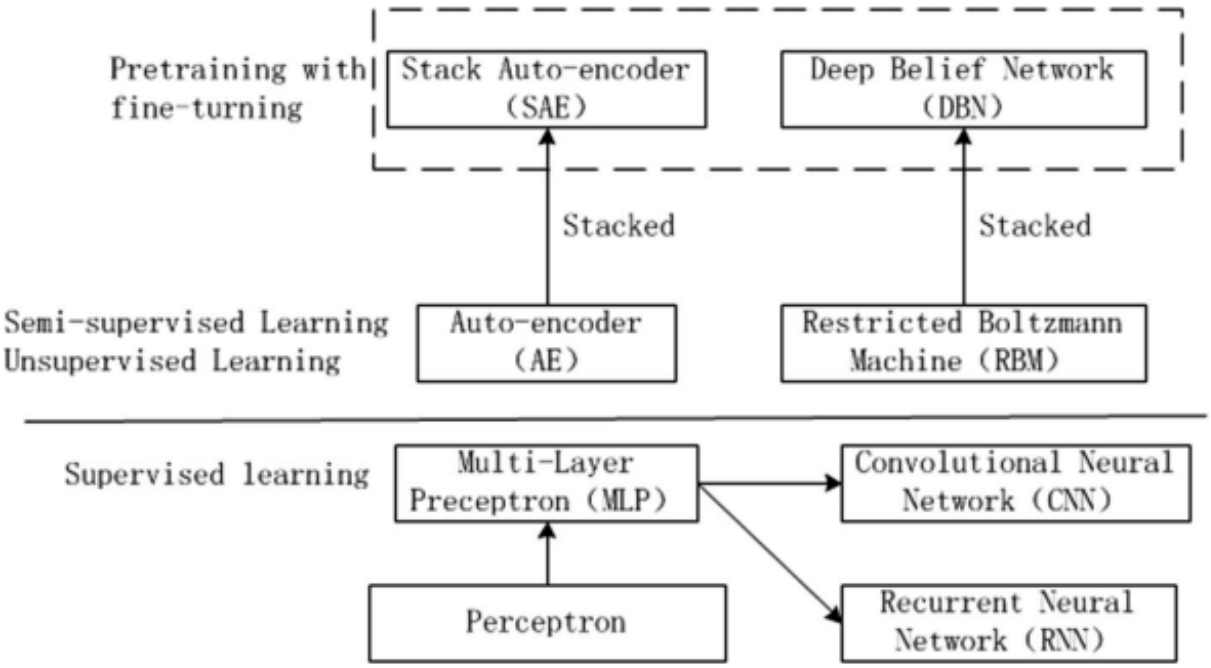

Figure 1.1: DL model categorization diagram

sectors is becoming increasingly common. The following are examples: Alexa, Siri, Google Assistant, Amazon recommendation engines, Tesla autonomous driving technology, Prisma, and FaceApp. Virtual assistants have become increasingly popular around the world. AI is revolutionizing the world with deep neural networks.

These algorithms are a type of data-driven algorithm that is important in the field of machine learning. It can be used to predict nonlinear, time-varying data satisfactorily because it is adaptive, it is capable of approximating nonlinear functions, and it is quite powerful. The majority of NNs, however, usually rely on the attribute features of the input data. Sparse input features and missing data will likely lead to underfitted predictions. It may be necessary to extend the model training process and to take a risk of overfitting if input features are highly correlated and there are more attribute attributes than samples. Machine learning researchers are currently focused on deep neural networks which are the predecessors of ANNs. There are two main differences between DNNs and conventional ANNs. (1) Generally, DNNs use multiple layers of hidden information or multiple stages of nonlinear information processing. (2) Whether supervised methods or non-supervised feature representation methods are used to construct continuous, deeper and more abstract hidden layers. A branch of machine learning that has grown into an enormous branch by integrating knowledge and advanced technologies from many fields has evolved into deep learning. In order to process nonlinear information, the basic functionality of DL is still to build multilayer models.

Nodes, the building blocks of neural networks, are connected together. As part of the neuron network, these cells function as the brain's neurons. A neuron triggers a process when it receives a

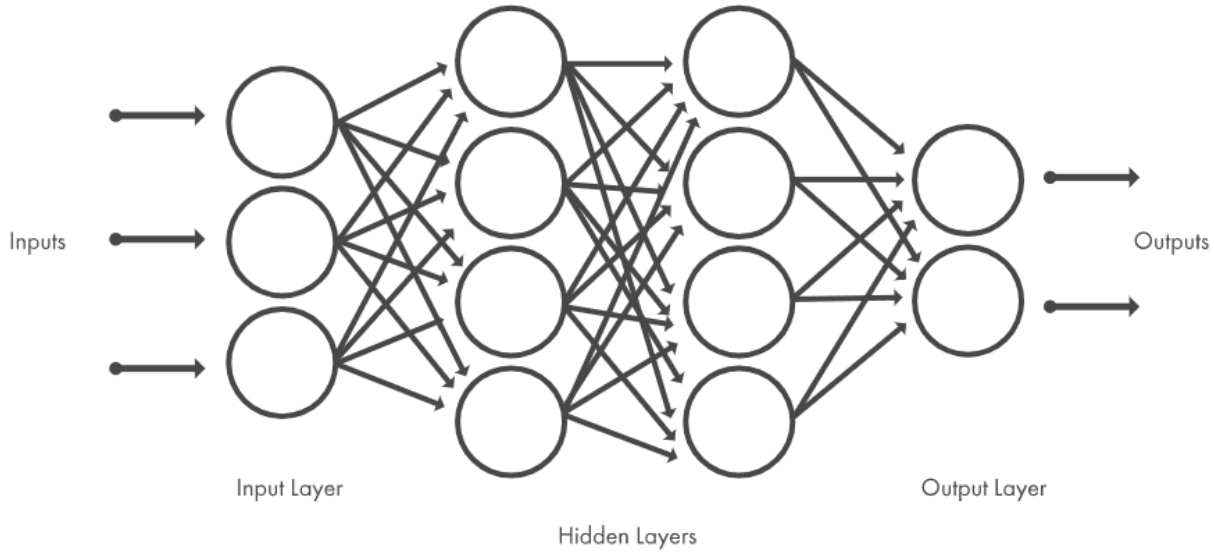

Figure 1.2: Deep Neural Networks

signal. Depending on the input received, the signal is passed between neurons. This creates a complex network that adapts to the input received through feedback.

Layers are created by grouping nodes together. It is possible to solve a problem by processing the different layers between input and output. Therefore, deep learning is a term that refers to networks of many layers to be processed.

With CAP (Credit Assignment Path), you can see how many steps you need to solve a problem. An augmented neural network is considered deep when its CAP index exceeds 2. Applied Deep Neural Network techniques are extremely efficient and useful in real-world applications. Using deep neural networks, AI-based robots such as Alpha 2 are able to send messages, speak and even obey voice commands etc.

1.3 STOCKS

Stocks are units of ownership in a company, also known as shares of stock or equities. When you buy a share of stock, you're purchasing a partial ownership stake in a company, entitling you to certain benefits. Understanding what stocks are and how they work is one of the keys to investing, since stocks play a central role in building a well-balanced investment portfolio. Stocks are an investment that means you own a share in the company that issued the stock. Simply put, stocks are a way to build wealth.

This is how ordinary people invest in some of the most successful companies in the world. For companies, stocks are a way to raise money to fund growth, products and other initiatives.

When you buy the stock of a company, you're effectively buying an ownership share in that company.

Companies raise capital to fund their operations by selling shares of stock. When companies sell stock, they're inviting investors to purchase a fractional ownership interest in the company, making them part owners. "Equity" is a way to describe ownership, and "equities" are an alternative name for stocks. Companies can also issue bonds to raise capital, although buying bonds makes you a creditor, without any ownership stake in the company.

When you buy shares of stock in a company, you gain certain privileges depending on the types of shares you own, including:

Voting rights: You may have the right to vote at the company's annual shareholder meetings.

Dividends: You may receive a share of the company's profits.

Capital appreciation: When the company's stock price goes up, your shares increase in value (and when the price of a stock declines, the value of your shares fall). While stocks give you an ownership share in a company, owning shares of stock doesn't mean you're entitled to a say in the company's day-to-day operations. Owning stock means you're trusting the company's leaders to run the business the way they see fit. If you don't like the performance of a company, you sell your shares and choose a new home for your investment dollars.

When private companies decide to sell shares of stock to the general public, they conduct an initial public offering (IPO). When you read that a company is "going public," that means they are conducting an IPO where they make shares available for purchase by investors via public stock markets.

During an initial public offering, the company and its advisors disclose how many shares of stock will be issued and set an IPO price. Funds raised from the sale of stock during an IPO go directly to the company. Once the offering is complete, the shares of stock are traded on the secondary market—otherwise known as "the stock market"—where the stock's price rises and falls depending on a wide range of factors.

There are different types of stocks:

Companies issue a variety of different types of stock. Common stock and preferred stock are among the most common varieties, and some companies have different classes of stock. These different types of stock determine voting rights, dividend payments, and your rights for recouping your investment if the company goes into bankruptcy.

Common stock vs Preffered stock

Companies frequently issue different classes of stock, often designated with a letter, such as A, B, or C. Additional share classes are typically issued with specific voting rights per class and exist to help company founders or executives retain a greater degree of control over the company.

Take Alphabet, the holding company that owns APPLE. Alphabet has three classes of stock. Class A stock (APPLE) gets one vote for each share. Class B stock is held by the company's founders and gets 10 votes per share. Class B shares are not publicly traded, and exist to help the founders retain control over the company. Class C stock (APPLE) has no voting rights, and is largely held by employees and some common shareholders.

Types of dividends

Depending on the type of stocks you own, companies may share their profits with you via dividends. Investors receive dividend payments quarterly or annually, with payments allocated based on how many shares of the company's stock you own. Holders of preferred stock have a priority claim to dividends, ahead of common stock shareholders. Regardless of the type of stock you own, the principles governing dividends are essentially the same.

For example, say a company has positive earnings for the quarter and issues a $0.42 preferred stock dividend. If you own 100 shares of the company's preferred stock, you'll receive a cash dividend of $42. Many companies also offer a dividend reinvestment plan (DRIP) that allows you to reinvest your cash dividend payments back into the stock, expanding your holdings and keeping your cash hard at work in your portfolio.

Companies sometimes issue stock dividends. If a company declares a stock dividend of 5% and you hold 100 shares of that company, you'd receive five additional shares of stock, bringing your holdings to 105 shares. However, the value of each outstanding share would decrease by 5%, making the value of your shares the same.

Companies also issue hybrid dividends that are a combination of cash and stock. Hybrid dividends are rare but have been used in the past by companies as a way of sharing profits with their shareholders.

Why stocks

Owning shares of stock gives you the potential to share in the profits of the world's most successful companies. The SP 500, one of the most common indexes that track stock performance in the U.S., delivered investors a 7% average annual rate of return, adjusted for inflation, in the period from 1959 to 2009. Compared to Barclay's U.S. Aggregate Bond Index which has returned an average of 4.67%, stocks outperform fixed-income investments over the long term.

While buying them isn't without risk, investors use stocks as one of the core tools to grow their savings and plan for long-term financial goals like retirement and educational savings. As stock prices go up, so does your savings balance. But be aware that stock prices also go down, and sometimes lose all of their value and become worthless. There's no guarantee that you'll recoup your investment.

Stocks are one of the basic ways to diversify an investment portfolio. Investors buy different stocks in companies large and small in a wide variety of industries to help mitigate risk, as different sectors of the economy thrive at different times. For example, a company selling paper products might experience record sales during an economic crisis like COVID-19 whereas an automaker might have below-average sales performance. Owning a variety of different stocks can help investors enjoy gains in thriving sectors while offsetting losses in others.

1.4 BACKGROUND

The following section discusses what machine learning is and why researchers have been using

these algorithms in the past to predict stock prices. In addition, we will describe the technologies we use in our research.

In computer science, machine learning refers to the process of giving computers the ability to learn. A machine learning algorithm can be divided into two categories. The first is supervised learning and the second is unsupervised learning. Machine learning models are trained by providing an algorithm and data, so the model can learn its parameters based on the data.

1.4.1 Supervised Learning

Machine learning algorithms are trained with input examples and associated labels in supervised learning.Models attempt to approximate the function y = f(x) as closely as possible, where x represents the input example, whereas y represents its label. We are using supervised learning here as we are using a training dataset with correct labels in order to train the algorithm. Using the output variable as the basis of grouping, supervised learning algorithms can also be classified as regression algorithms and classification algorithms. The regression task is called a regression exercise if the output is a continuous variable. Several examples of regression tasks are predicting house prices and stock prices. This type of task is called a classification task when the output variable is a categorical variable, such as color, shape type, etc.

Supervised learning algorithms are used by the majority of machine learning applications. A supervised learning algorithm, however, may be a logistic regression, a linear regression, a support vector machine or a random forest.

1.4.2 Unsupervised Learning

Machine learning is an unsupervised process in which a model seeks out patterns in unlabeled datasets with little human supervision. In contrast, supervised learning techniques, such as classification or regression, require inputs and observations to build a model, which then maps the inputs to the observations. A model that uses unsupervised learning must search for patterns in the data, as only the inputs are available. The dataset is not labeled when unsupervised learning occurs, so the model learns useful properties from the structure of the dataset if it is not labeled. The model is not told what it should learn; it must interpret the unlabeled data and find patterns on its own. Since we have little or no knowledge of the data, unsupervised learning algorithms are harder to develop. In unsupervised learning tasks, examples are typically grouped together, dimensionality is reduced and density is estimated. The algorithm attempts to discover patterns by analyzing the underlying structure of the input examples. Clustering and association functions are two further categories of unsupervised learning algorithms. Clustering is the process of discovering groups or clusters in the data by using algorithms like k-means. Apriority, for instance, is an algorithm that attempts to predict the future purchase behaviors of customers using association rules.

1.4.3 Support Vector Machine

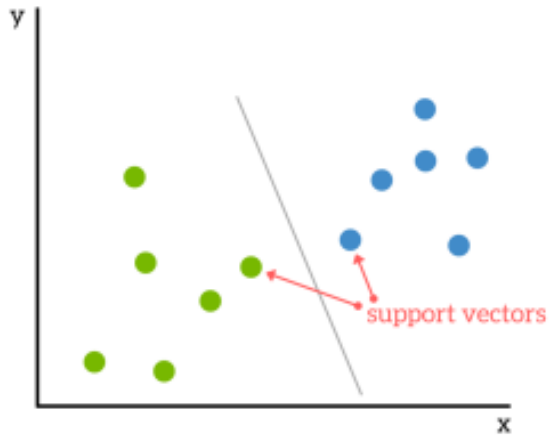

Figure 1.3: Support vector machine

SVMs, which are also known as Support Vector Machines, are popular algorithms for supervised learning, used both for classifying and analyzing data. Nevertheless, it is primarily used to solve classification problems in machine learning. SVM's goal is to determine which line or decision line will segregate n-dimensional space into classes to make it easier for us to place new data points in the correct category down the road. A hyperplane is a boundary that represents the best decision. Hyperplanes are created by selecting extreme points and vectors. Thus, the algorithm is referred to as Support Vector Machine, since these extreme cases are called support vectors.

SVM plots every point in the dataset in an n-dimensional space, with the dimensions corresponding to the features. Data points represent the location of a feature on the axis associated with their value. When training, SVM creates an optimal hyperplane for each input example based on its labeling. Plotting the unlabeled data point and checking the position of the point in relation to the hyperplane is the method for determining the class of the new data point.

Models like this one can choose a decent set of features, control their fitting, and assess their stock indicators. A significant challenge with SVM is that it can be used when the input variables may be of a high dimensionality. It is extremely difficult to train a model when the number of features is in the hundreds or thousands.

1.4.4 Multi Layer Perceptron

A multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected

as a directed graph between the input and output layers. MLP uses backpropogation for training the network. MLP is a deep learning method.

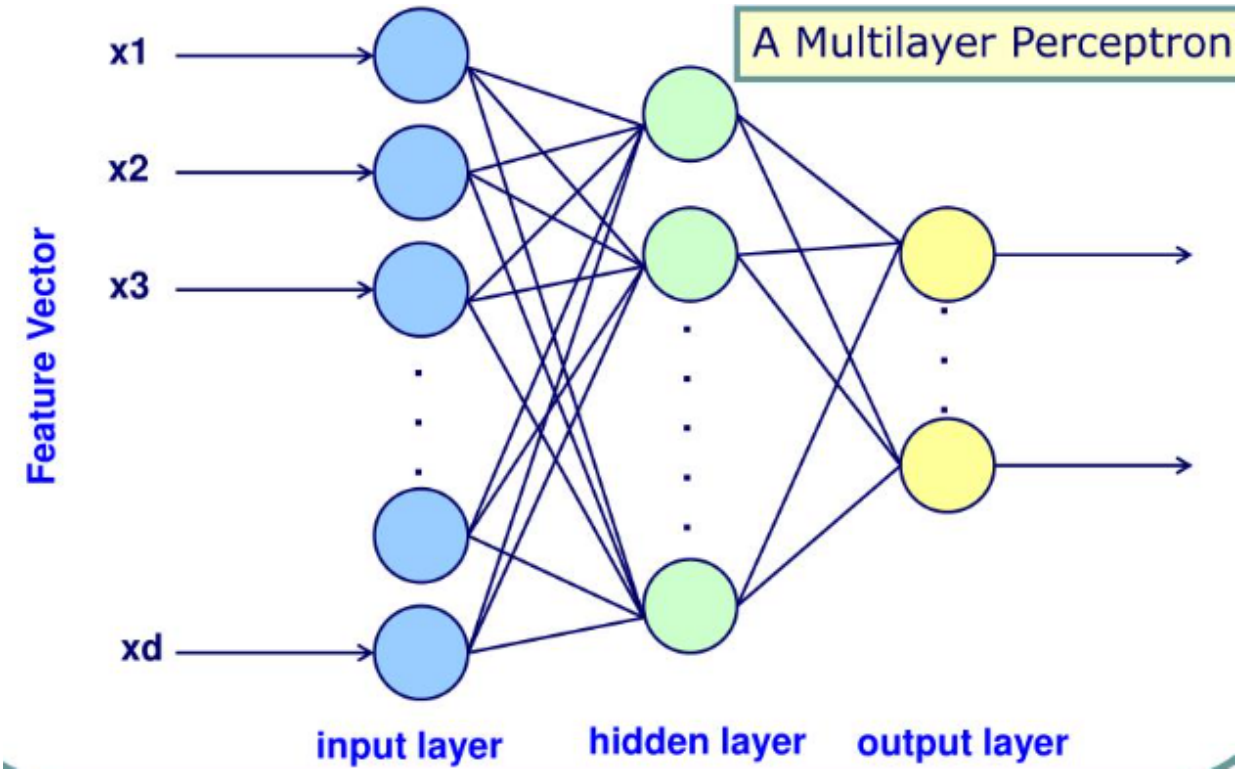

Figure 1.4: Multi Layer Perceptron

A multilayer perceptron is a neural network connecting multiple layers in a directed graph, which means that the signal path through the nodes only goes one way. Each node, apart from the input nodes, has a nonlinear activation function. An MLP uses backpropagation as a supervised learning technique. Since there are multiple layers of neurons, MLP is a deep learning technique.

MLP is widely used for solving problems that require supervised learning as well as research into computational neuroscience and parallel distributed processing. Applications include speech recognition, image recognition and machine translation.

1.4.5 Artificial Neural Networks

In contrast to task specific algorithms, deep learning is one of the broader classes of machine learning methods that rely on learning data representations. Feature extraction and the transformation of features can be performed automatically using deep learning models, which rely on multi-layered non-linear processing units called neurons. Artificial neural networks are networks of neurons of such a type. Nonparametric representations of information, such as Artificial Neural Networks(ANN), relate the output to the input variables in a way that is nonlinear. An ANN resembles the structure of neurons in the human brain because it is composed of nodes which are interconnected. The neurons in these layers are arranged in successive layers, with the output of the current layer of neurons be-

coming the input of the layer above. Feed forward neural networks are thought to be those in which the connections between neuron layers don't form a cycle. A feed forward neural network is created by applying a function to the output of the previous layer. Activations on inputs are applied by the hidden layer for final predictions and hidden inputs are transformed into something that the output layer can use.

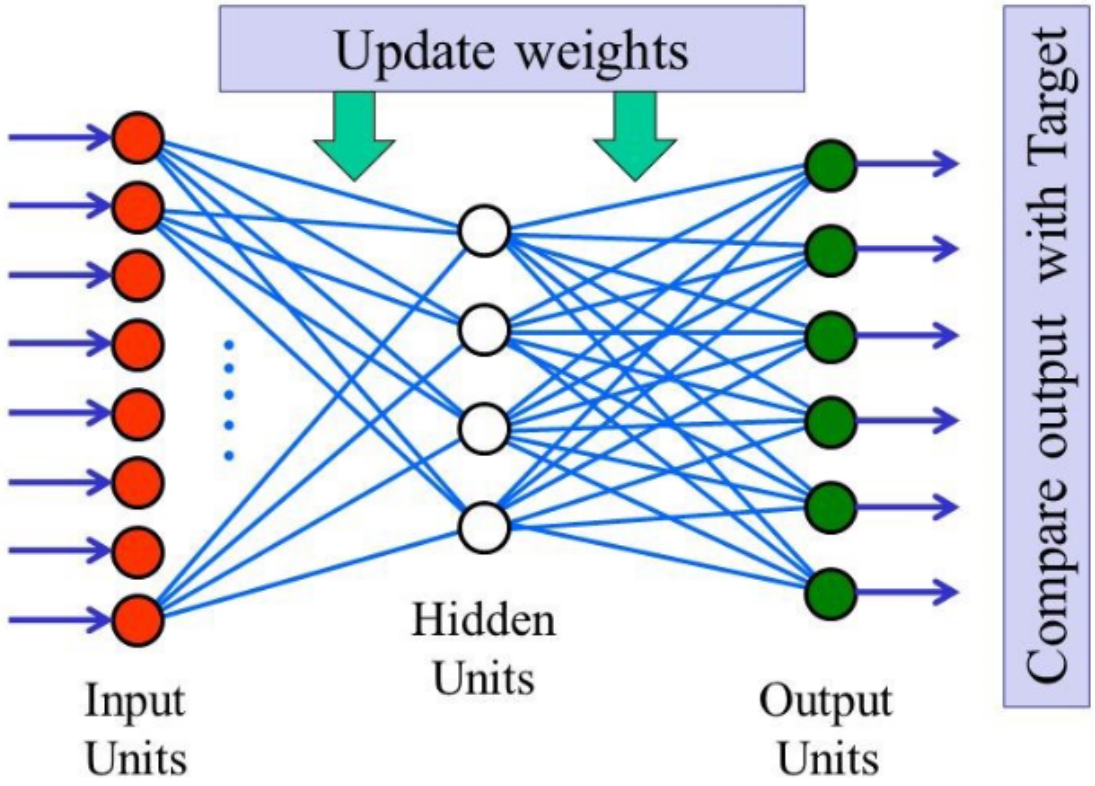

Figure 1.5: Artificial Neural Network

1.4.6 Convolutional Neural Networks

A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. Convolutional neural networks are widely used in computer vision and have become the state of the art for many visual applications such as image classification, and have also found success in natural language processing for text classification.

Convolutional neural networks are very good at picking up on patterns in the input image, such as lines, gradients, circles, or even eyes and faces. It is this property that makes convolutional neural networks so powerful for computer vision. Unlike earlier computer vision algorithms, convolutional neural networks can operate directly on a raw image and do not need any preprocessing.

A convolutional neural network is a feed-forward neural network, often with up to 20 or 30 layers. The power of a convolutional neural network comes from a special kind of layer called the

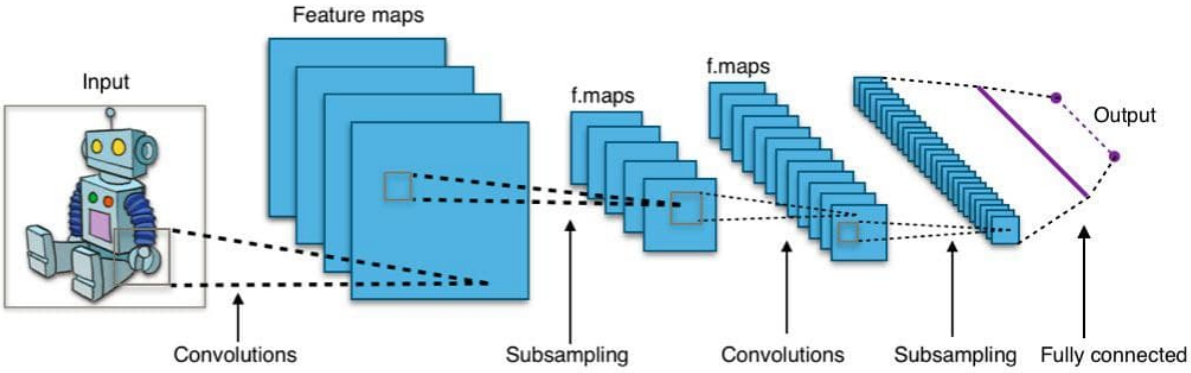

Figure 1.6: Convolutional Neural Network

convolutional layer.

Convolutional neural networks contain many convolutional layers stacked on top of each other, each one capable of recognizing more sophisticated shapes. With three or four convolutional layers it is possible to recognize handwritten digits and with 25 layers it is possible to distinguish human faces.

The usage of convolutional layers in a convolutional neural network mirrors the structure of the human visual cortex, where a series of layers process an incoming image and identify progressively more complex features.

Max pool Layer:

Generally, a pooling layer is inserted between successive convolution layers in a convolutional network architecture. Pooling layer progressively reduces the spatial size of the representation thereby significantly reducing the number of parameters, computation in the network and controlling overfitting. Pooling layer is simply a max operation applied on every slice of input with specified dimension. Since the neighboring values in the convolutions are strongly correlated, it is understandable to reduce the output size by subsampling the response from filter. It is most common to use a max pool layer of size 2, as values further apart are less correlated.

1.4.7 Recurrent Neural Networks

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate. Like feedforward and convolutional neural networks (CNNs), recurrent neural networks utilize training data to learn. They are distinguished by their "memory" as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence. While future events would also

be helpful in determining the output of a given sequence, unidirectional recurrent neural networks cannot account for these events in their predictions. As a class of ANN, recurrent neural networks (RNN) utilize directed graphs between their neurons, or in simpler terms, a self-loop in the hidden layers. Thus, RNNs can learn the current state of the hidden neurons based on the previous state of those hidden neurons. RNNs take into account the information they have learned from previous inputs as well as their current input. They learn sequential information by using internal memory or states. In addition to learning handwriting, speech, and other skills this way, they are also able to learn a variety of other things as well. An input vector unit in a traditional neural network is

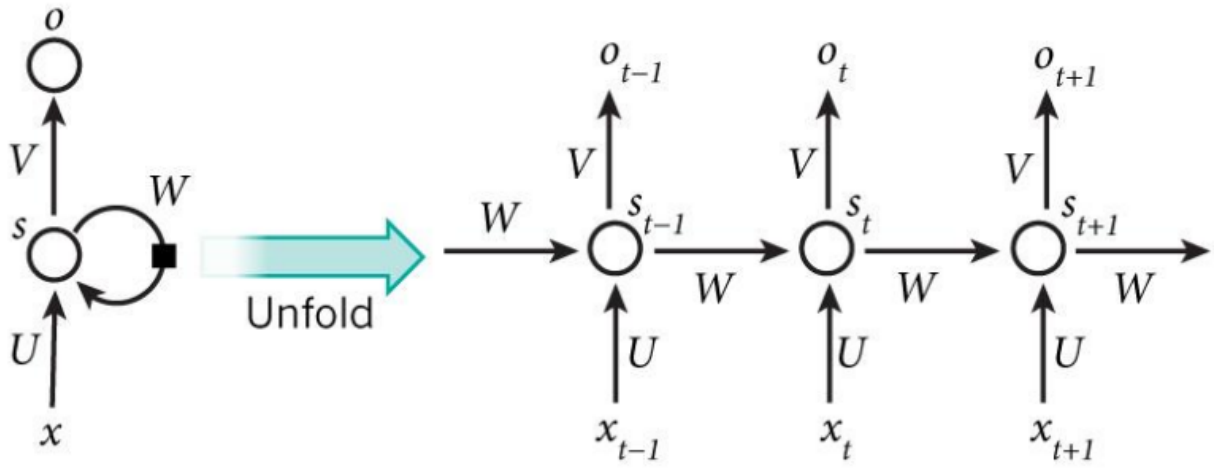

Figure 1.7: Basic Recurrent Neural Network

assumed to be independent of the other units. Due to this, conventional neural networks are unable to utilize sequential information. In contrast, in the RNN model, sequential data from the time series generate a hidden state, which is then added to the output of the network based on the hidden state. This use case is best suited to RNNs due to the fact that stock trends are a time series data. The RNN model is expanded to represent a complete network. Updates to the weights of hidden layers are propagated back to the output layer based on the outcome of the output layer. In deep neural networks, however, this change will be vanishingly small to the layers in the beginning, preventing the weights from changing their value and stopping the network from learning. In fact, this is known as the "vanishing gradient problem".

1.4.8 Long- Short Term Memory

This type of recurrent neural-network architecture solves the problem of vanishing gradients using long short-term memories (LSTMs). As long-term memory models, LSTMs work well on a wide range of problems and can learn very long-term dependencies. The architecture of modern LSTM cells was developed by a wide range of researchers, including the original authors. The architecture of modern LSTM cells was developed by a wide range of researchers, including the original authors.

The layers of recurrent neural networks usually loop back to the first layer in the chain like structure. Figure 4 illustrates how this looping module may look in standard RNNs, which have a very simple structure. Structures such as this which control flow can be created using a tanh layer.

An LSTM, on the other hand has four networks interacting in a special way. LSTMs, instead of the simple structure shown in figure below, utilize four layers that are interconnected in a very complicated manner. The gates that implement LSTMs are normally referred to as "forget" gates. It is possible to backpropagate errors through any number of virtual layers by controlling these gates. The network can do tasks that depend on events that were thousands of time steps ago with this mechanism.

CHAPTER 2

LITERATURE SURVEY

Jürgen Schmidhuber, [1] describes Deep neural networks have been making progress so well because of the availability of data and the scale at which computation is available. A number of pattern recognition and machine learning contests have been won by deep neural networks. Several previous millenniums' work is included in this historical survey. There are two kinds of learners based on the extent of their credit assignment paths, which are chains of causal connections between actions and consequences that can be learned. As well as reviewing deep supervised learning, I also review reinforcement learning evolutionary computation, as well as indirect searching for short programs encoding deep and large networks.

Luca Di Persio and O. Honchar, [2] describes An Ensemble of deep neural network ksare also applied to the problem of forecasting in stocks but they involve the training of each expert separately and then to obtain the results averaging methods are applied. In order to predict stock market price movements, artificial neural networks have been applied, namely MLP, CNN, and RNN. For the SP500 index, they compared daily training results, showing that convolutional neural networks (CNNs) are more powerful than all the other architectures they considered. Another novel approach was implemented that outperformed other neural network algorithms. A crucial part of forecasting stock prices is feature preprocessing, as shown by the study.

Sreelekshmy Selvin, [3] describes Interest rates, various ratios of prices (such as Price to earnings ratio), parameters related to economics form the basis for the Fundamental Analysis whereas past price values, historical data, volumes traded over a particular time, various moving averages form the basis of Technical Analysis. In this experiment, the values of Open, High, Low, and Close are used for the purpose of prediction and sliding-window approach is used for training the neural networks.

Ayman E. Khedr et al., [4] describes, classification techniques like the k-nearest neighbor, Naïve Bayes algorithm are used to predict the stock trend rise or fall but this paper tries to solve the problem as a regression by making use of Deep Neural Networks. It is evident that the LSTM model in and Conv1D_LSTM model in are able to capture the dynamics of time-data better than the CNN model in.

D. Ashok Kumar and S. Murugan, [5] describes The neural network approaches had also

been investigated for stock markets of India in where for performance analysis metrics like RMSE, MAE, MAPE had been used. It is evident that the LSTM model in and Conv1D_LSTM model in are able to capture the dynamics of time-data better than the CNN model in.

Jasemi et al. [6] studied the information hidden in Japanese stock candlestick charts using a multi-layer perception (MLP) model. A Japanese candlestick was utilized for the experiments and to describe how the approach was run. This approach asserts that if the concept of NN is applied both as a learner and as an analyzer of several predefined rules, good results may be obtained. In two configurations of Raw data-based and Signal based, nodes in the NN held variables influencing the opinions of Japanese Candlestick analysts. In order to build the neural networks, training and testing data sets were used. The training data set was used to build the neural networks, while the testing set was used to examine the forecasting accuracy.

In 2009, Vanstone and Finnie [7] proposed an NN-based empirical methodology for designing automated trade systems based on ANNs. Using soft computing technologies that include artificial neural networks, this paper presents a methodology for developing robust mechanical trading systems. There is a particular emphasis on designing these steps to fit the realistic constraints in which a neural network will ultimately operate in the paper. The steps involved in creating a neural network for stock market trading are described. Transparency and clarity associated with such a common methodology should make previous publications trustworthy and reusable.

Xiong et al., [8] predicted the linear pattern of a time series using an ARIMA model, estimated the nonlinear residual of the time series using an NN model and combined the results to form the final prediction for the time series. They examined the effectiveness of the hybrid prediction model through simulations based on a RMB exchange rate series.

Wu et al., [9] performed an empirical study on the relationship between volume and price in China's stock market using the exponential generalized auto-regressive conditional model and a backpropagation NN based on the theory for the relationship between volume and price.

Li [10] predicted error series using empirical mode decomposition in conjunction with a support vector machine (SVM). In this study, the most effective model used is SVM, which has the least amount of error among other models. Consequently, this method can accurately predict the future price of stocks. Although SVM helps to study sequential data, it suffers from overfitting when studying sequential data.

Shen et al., [11] constructed a deep belief network using a continuous restricted Boltzmann machine and combined it with the conjugate gradient method to predict exchange rates. Through a comparison, they found that the DL network structure was superior to conventional feedforward NNs. Using deep belief networks (DBNs) for forecasting exchange rates is presented in this paper. The DBN model is updated to model continuous data by the use of continuous restricted Boltzmann machines (CRBM). Experiments are performed to determine the optimal structure of DBN used in exchange rate forecasting. Additionally, conjugate gradients are applied to accelerate the learning process for DBN. The experimental work tests three exchange rate series and evaluates the proposed

method using six evaluation criteria. Based on comparisons with traditional forecasting methods, such as feed forward neural networks (FFNN), the proposed method is applicable and works better than traditional methods at predicting foreign exchange rate movements.

Ding et al., [12] predicted the changes in stock index prices from an event-driven perspective. They constructed a prediction model using a convolutional NN and used it to measure the short- and long-term effects of events on the changes in stock prices. Eventembedding-based document representations perform better than discrete event-based methods, whereas deep convolutional neural networks can provide a greater understanding of the long-term impact of news events than standard feedforward neural networks. The accuracy of their method was 6% higher than that of a conventional method for the stocks in the Standard and Poor's (SP) 500.

Zhao et al., [13] constructed a prediction model using a de-noising autoencoder and performed multiple sets of experiments based on datasets extended using the bagging method. They also used the prediction model to predict oil prices. For learning representations from data and generating forecasts, the SDAE deep learning model is applied. Combining SDAEs into an ensemble model, bagging is a powerful ensemble method which produces better results than individual SDAEs. Compared with benchmark models such as econometric models (e.g., RW and MRS), artificial intelligence models of shallow architectures (i.e., SVR, SVR-B, FNN and FNN-B and SDAE's base model SDAE), SDAE-B outperformed benchmark models. In addition to the statistical analysis, the proposed approach may prove useful in forecasting crude oil prices, suggesting that it can be used as a promising tool.

Krauss et al.,[14] studied the application of several integrated methods, including DNNs, gradient boosted regression trees and random forests, in statistical arbitrage. In addition, they also proposed an equal-weighted integrated model based on multiple models and, with it, achieved excess earnings in the SP 500. Survivor bias is removed from each model by training it on lagged returns of stocks in the SP 500 index. Based on a probability forecast, daily one-day-ahead trading signals have been generated since 1992 for stocks that are likely to outperform their general market counterparts

Lecun et al.,[15] described , the newly emerged deep learning (DL) algorithms can be trained with massive volumes of nonlinear data and used to construct deep NNs (DNNs) with multiple hidden layers and capture more abstract nonlinear relationships between data. In fields such as speech recognition, visual object recognition, object detection, and drug discovery and genomics, these methods have significantly advanced the state-of-the-art.The deep learning algorithm uses backpropagation to discover intricate structure within large data sets. The algorithm tells the machine how to adjust its internal parameters in order to compute the next layer's representation based on the previous layer's representation. Compared to conventional machine learning algorithms, these DL algorithms can be used to solve nonlinear problems more satisfactorily.

Box et al., [16] proposed classical time series analysis method for time series on both stationary and non-stationary condition. One of the most prominent univariate time series models is the autoregressive integrated moving average (ARIMA) model. The popularity of the ARIMA model is due to its statistical properties as well as the well-known Box-Jenkins methodology in the model

selection procedure. However, the effect of this model depends on the smoothing of non-stationary time series data, especially on nonlinear financial time series.

X. Dai and M. Bikdash, [17] use hypothesis-testing-based adaptive spline filtering (HASF) algorithm to discover the trends of fragmented time series for mHealth Apps. By using hypothesis testing and variance minimization, HASF adjusts the nodes of the spline, which increases its robustness. By filling in gaps with data from a previous trend analysis provided by HASF, an additional improvement is possible. There are three variants for filling gaps of missing data, but the most effective method seems to comprise of covering data dense regions with cubic splines, matching them for continuity and smoothness with linear splines. A cubic spline fitting is used to fill in small gaps. By simply transferring the importance of missing data to the existing neighbors, the existing measurements are weighted according to their importance.A comparison of the heart rate, blood pressure, and noisy sine data sets is carried out in order to demonstrate and evaluate the methods.

Tay and Cao [18] and Kim [19] studied various parameters(the upper bound and the kernel parameter) of SVM to select optimal values for the best prediction performance. Using a multi-layer back-propagation neural network against SVM in this paper, it is examined whether SVM can be used as a forecasting tool for financial time series. Data sets derived from the Chicago Mercantile Exchange are used to collate five real futures contracts. A comparison of SVM and BP neural networks based on the criteria of normalized mean square error (NMSE), mean absolute error (MAE), directional symmetry (DS) and weighted directional symmetry (WDS) found SVM to outperform the BP neural network. The study examines the effect of free parameters on the performance of SVMs since there is no structured way to choose them.

Hassan and Nath [20] and Gupta and Dhingra [21] shows that Hidden Markov Model(HMM) is widely used in financial time series price forecasting and is better than ANN and ARIMA method in price predicting accuracy. However, the hyper-parameter settings (such as the number of hidden state) of HMM is easily influenced by people and there is no authoritative principle to guide the selection of parameters.

Fischer and Krauss [22] applied LSTM networks to a large-scale financial market trend and price prediction task. However, both of them are leverage LSTM model by accumulate or forget information in a long period of time to modeling the temporal relationship of sequence data. Memory-free methods, such as random forests, deep neural networks, and logistic regression classifiers, are found to be inferior to LSTM networks. While there was a clear outperformance relative to the general market from 1992 to 2009, LSTM profitability has fluctuated around zero following transaction costs as of 2010. In addition, we shed light on artificial neural networks' black box by revealing their sources of profitability.

Zhang et al., [23] proposed a novel State Frequency Memory(SFM) recurrent network to capture the multi-frequency trading patterns from past market data to make long and short term predictions over time. Specifically, SFM was used to decompose the hidden states of memory cells into multiple frequency components, each of which models a particular frequency of latent trading

pattern underlying the fluctuation of stock price. However, most of these works use neural network to extract frequent pattern form Multivariate time series and combine with regression or classification model to do prediction but lack of consideration of frequent patterns with data.

M. Wen et al., [24] decribed stocks meet different investment desires and investment needs, expand the range of investment options, expand investment channels, to some extent meet the possibility of investors to obtain the corresponding income, and to some extent enhance the flexibility and liquidity of capital. By using motifs (frequent patterns) in sequence reconstruction, they developed a new method to eliminate the noise of financial time series, and then used convolutional neural networks to extract the spatial structure. Compared to the traditional signal processing approaches and the frequency trading patterns modeling approach with deep learning in stock trend prediction, the present methodology shows high efficiency in feature learning and outperformance with 4% to 7% improvement in accuracy.

Zhang et al., [25] describes,In data mining and statistics, multiple regression model is a technique for estimating the relationship between a dependent and one or more independent variables. In multiple regression, an attempt is made to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation.

Kamley et al., [26] applied the multiple regression approach to predict the stock market price from the stock market data (yahoo fnance) using three variables, namely: open, high and close. Diferent pre-processing steps are applied for the cleansing, integration, and transformation of data. They achieved a prediction accuracy of 89%. However, the major limitation of their work includes lack sufcient training of the model due to poor selection of stock indicators.

Hiba Sadia et al ., [27] It is also mentioned that stock value Prediction by methods for Multi-Source multiple instance learning unequivocally foreseeing the protections trade is a troublesome task, anyway the web has wind up being a useful gadget in making this task less difficult, due to the related course of action of the data, it is certainly not difficult to evacuate certain inclinations right now, it is less difficult to establish associations between different variables and, for the most part, a case of adventure. The way in which budgetary trade information can be adequately predicted is through the use of some different options from specific legitimate data and the use of different strategies, such as the use of a feeling analyzer, to suggest a remarkable relationship between the emotions of individuals and how they are influenced by the enthusiasm for express stocks. One of the progressively noteworthy areas of the desire strategy was to extract huge events from web news to see how they had an impact on stock costs. It is also mentioned that trade prediction protection: using historic data analysis. The stock or offer expense can be foreseen using chronicled data and its example in all actuality there is need to apply counts to anticipate the expenses. The customary frameworks are just worried about variety of an element that is selected for forecast. The latter is usually achieved with the benefit of the Genetic Algorithms (GA) or the Artificial Neural Networks (ANN's), but they neglect to establish a relationship between their stock costs as long-distance fleeting dependencies.

RautSushrut et al. [28] suggested that supervised learning classifier be used to forecast stock

price movement based on financial index data, and determine their ability. In the financial market computational analytical approaches have been portfolio modeling. A discussion about the statistical AI methodology has been addressed; the usage of SVM methodology has been shown in the paper and also shown that tactical methodologies can be applied to predict the stock prices.

Manoj S Hegde et al.[29] investigated that the Long Short Term Memory (LSTM) networks are a type of recurrent neural network (RNN) capable of solving in volute linear problems, and also there is a discussion about the usage of RNN (Recurrent Neural Networks) to predict the share prices.

M. Roondiwala et al.[30] proposed that the Long Short-Term Memory is the most popular RNN architecture. In the secret network layer, LSTM introduces a memory cell; a processing device that replaces conventional artificial neurons, using these memory cells, networks can effectively link memory and remote input in time, making it suitable to dynamically capture data structure over time with a high predictive limit. It is also shown in the paper that the stock prediction can be done on the NIFTY50 shares. The data collection is one of the major step and later the training of our model and there is a need to test the algorithm by applying different data set to the algorithm. Our procedure will be discussed in coming sections.

As Kim and H. Y. Kim et al. [31] identified that .another significant issue with basic ANNs for stock forecast is the marvel of detonating fleeting inclination, where the loads of a gigantically huge system either become excessively massively enormous or too inuscule (respectively),drastically easing back their union to the ideal worth. This is regularly brought about by two components: loads are instated self-assertively and the loads progressively proximate to the end of the system moreover slope to transmute significantly more than those at the beginning. It is also mentioned in the paper that the usage of LSTM networks can be applied in procedure of predicting share prices.

As discussed by S. Selvin et al.[32], Customary types for dealing with the financial exchange investigation and the stock-value forecast include a major review of the past stock-exhibition gander and the general credibility of the organization itself, and a measurable investigation that is solely concerned with the calculation and recognition of stock-value designs, it also mentioned in the paper that different types of analysis that can be performed in order to predict the stock value.

Loke.K.S et al.[33] suggested that the volatile nature of the stock market is an area that needs a lot of analysis based on historical data. Traditional stock trend forecast algorithms use historical time series stock data, traditional technical forecasting procedures for stock prices are based on statistical data analysis. in the paper author also talks about the change and advancements in the process of predicting stock prices using AI and Machine Learning methodologies, there are many research which are being conducted to find a accurate model to predict the stock prices and there is no universal solution which is available to apply, hence the historic data of a share will be considered for stock price prediction.

Xi Zhang1 et al. [34] suggested that the stock markets play critical roles in modern society's economic operations. It is also metioned in the paper that the analysis can be performed an the data that is retrieved from a legitimate source and proposed a methodology in which we can utilize multiple

source of information to predict the stock values.

Tao Xing and Yuan Sun et al. [35] suggested a model which considers the historical equity share price of a company price and applies RNN (Recurrent) technique called Long Short Term Memory (LSTM). The proposed approach considers available historical data of a share and it applies prediction on a particular feature. The features of shares are Opening price, day High, day Low, previous day o price, Close price, Date of trading. The proposed model uses the time series analysis in order to predict a share price for a required time span.

CHAPTER 3

METHODOLOGY

3.1 BLOCK DIAGRAM OF PROPOSED MODEL

Figure 3.1 shows the Block Diagram of stock price prediction using deep neural networks the finest method for prediction that works well for recognizing the species of flower is the main purpose of our methodology. Our deep learning models are created using the LSTM deep neural networks algorithm. A dataset containing stock price of APPLE is used in both the testing and training of model. We import dataset of stock price for data processing, then we use parameters like High,low,open,close,volume for training the dataset using LSTM. Then we predict the data for next ten days using the model. TensorFlow, a Python-based toolkit, is used to implement this algorithm. Our aim is to determine which prediction model has the greatest validity. As a means of determining model accuracy, we have divided the original sample data into training and testing sets, so that a trained model can be evaluated.

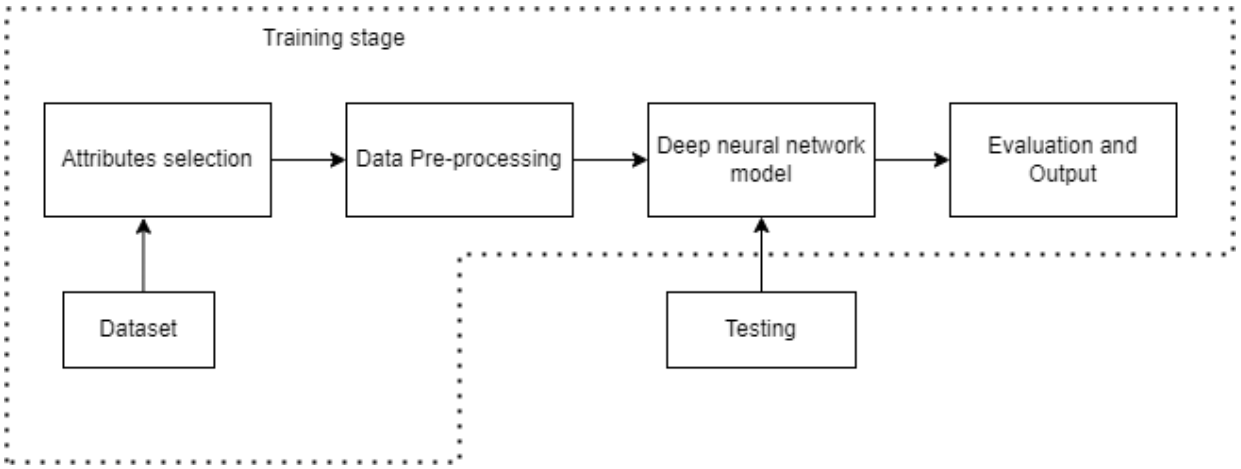

Figure 3.1: Block diagram of proposed model

3.2 DATA COLLECTION

To build a machine learning model, data collection is crucial. No matter how good a model is, without accurate data, it can't learn anything. The dataset used for machine learning is usually the collection of information that is needed to train and predict the model. Structured and unstructured datasets can be distinguished by the tabular format of the structured dataset, where a row represents a record and a column represents a feature, and unstructured datasets constitute images, text, speech, audio, etc., which are collected and processed through Data Acquisition, Data Wrangling, and Data Exploration. These datasets are used for training and calculating the accuracy of the mode by dividing them into training, validation, and test sets.

1. **Training sets**

These datasets are used to train the model, i.e. the weight of the model is updated using these datasets.

2. **Validation sets**

i. Overfitting can be reduced using these types of datasets. When we test the model with data that is not used in the training, we can check whether the accuracy of the model is increased.

ii. Resulting from high variance, such as overfitting, is the increase in accuracy over the training dataset while the decrease over the validation dataset.

3. **Test sets**

i. We can sometimes unintentionally overfit our model when we modify it based on the output of the validation set and unintentionally peek into the validation set, making it overfit on both.

ii. In order to overcome this complication, we have developed a test dataset that is only used to verify the accuracy of the final model output.

The attribute or feature definitions determine the structure and properties of the dataset. Some datasets are created manually or by using an algorithm when testing an application. There are four types of data included in the dataset: numerical,text, time series and categorical. If you're predicting the car price, for example, numerical data is required. An observation or a sample is represented by every row of the dataset.

1. Numerical Dataset

The term numerical data refers to any data points that have a numerical value. These data points can either be discrete or continuous. Continual data can have any value within a given range, whereas discrete data is supposed to have a specific value. Suppose the doors of the cars have discrete numbers, i.e. either two, four, six, etc. and the price of the car is continuous, for example 1,000 or 1,250.5 dollars. The numerical data types are float64 and int64.

2. **Categorical Dataset**

Characteristics are represented by categorical data. An example of this would be the automobile color or production date. If the numerical value indicates a class, then it may also be a numerical value. In a gas car, the number 1 can be used, while the number 0 can be used for a

diesel car. Categorical data can be used to form groups but cannot be mathematically manipulated. Categorical data is an object data type.

3. **Timeseries Dataset**

Sequences of numbers collected over time at periodic intervals are referred to as a sequence. The price of a stock must be updated after a constant interval of time as it is in the stock market. With this type of data, you can easily monitor the timestamp of the data by tracking its temporal field.

4. **Text Dataset**

There is nothing more literal than text data. As the model is mathematical and needs to inform of numbers, the first step to handling test data is converting them into numbers. We might be able to formulate words by using functions as a bag of words.

3.3 FINANCIAL DATA PREDICTION

A time series of financial data, in particular stock indexes, is often characterized by multiple variables. The factors can be categorized into macroscopic and microscopic variables. Macroscopic factors impact the market on the macro scale (e.g. economic policies or national gross domestic product), while microscopic factors impact the market on the micro scale (e.g. random events, irrational emotional fluctuations of investors, and market rumors). Analyzing macro- and micro-level factors will allow the financial market, which has constantly changing factors, to make better decisions. Nevertheless, since these factors are subject to change frequently, it can be considered that they depend on unknown systems. To collect and analyze all the macro- and micro-variables and determine their relative influence would be nearly impossible.In the literature relating to mathematical and machine learning approaches to time series, financial time series prediction is often regarded as one of the most difficult areas to study. Observing financial data in time series form is the first step of financial time series analysis. Prices (stock prices, stock indexes, exchange rates, and futures prices) are often observed, as well as returns (stock returns, stock index returns, interest rates, and futures returns), fluctuation, trade volume, and companies' financial variables (bond issues and hedging techniques). The rate of return exhibits excellent statistical properties as a stable series and is usually used as a measurement of trading experiments since it is unaffected by the scale of investment.

3.4 PROPOSED ALGORITHM

Figure 3.1 shows how to predict stock prices using the proposed methodology. Preprocessing begins the process. We need scaling because some features may have high impact than others which will affect the performance of predictions. The final training data set is derived from the output of pre-processing. When building a model, a deep learning method is applied to the training data set to build the model and test it. An input to the model-building phase is a data set that contains training examples. In general, it falls into three divisions: training, validation, and testing. The training data set contains 80% of the data, including the validation data set, and the test data set contains 20%

of the data. In experimentation, LSTM appeared as the most effective algorithm, and it was tested against a dataset consisting of stocks over 41 years. Training and testing result in a model that can predict the stock prices, and predict the next output.

3.4.1 Algorithm

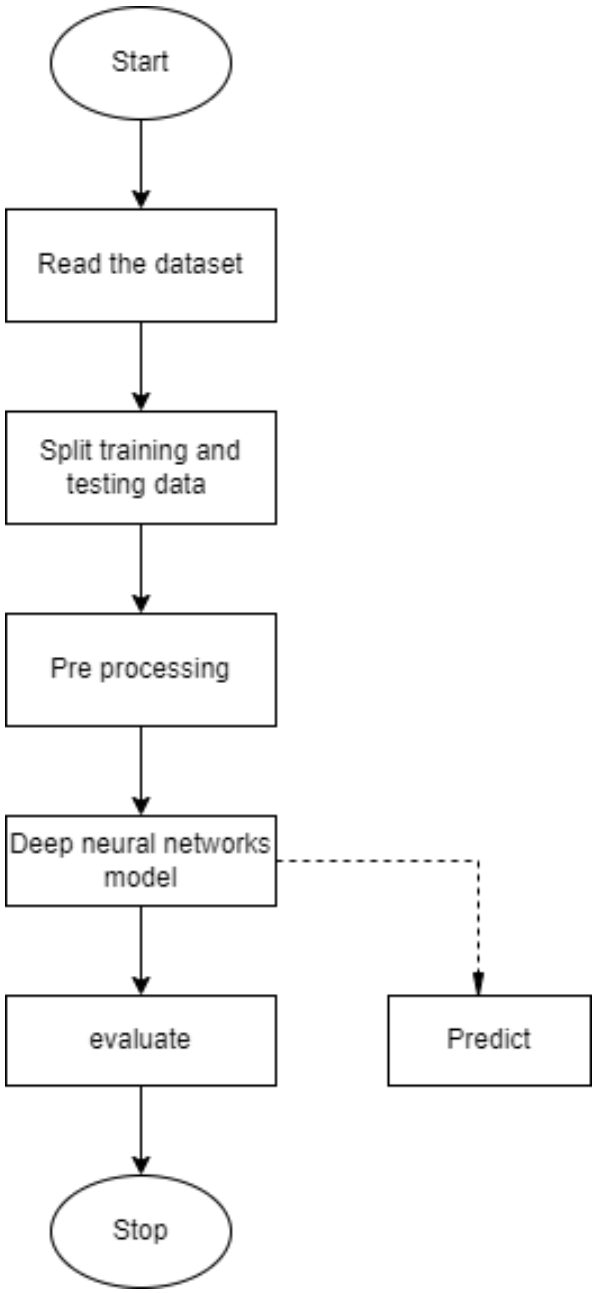

Figure 3.2: Flowchart of proposed method

3.4.2 Pre Processing

As a scaler, the MinMaxscaler sets the minimum and maximum values to 0 and 1, respectively. Feature scaling is performed using this class that is part of the sklearn.preprocessing module. To train machine learning models, you will need to understand these concepts since algorithms that require features to be scaled. The process of scaling consists in transforming numerical features so that their values fall within a similar range. In supervised learning models, the feature scaling helps to prevent biases towards a specific range of values. The impact of certain features may be greater than that of others, which will affect a prediction's accuracy or give certain variables an unfair advantage. Training models under such circumstances would place specific classes at a disadvantage. Therefore, scaling algorithms are important to standardize feature values so that you can take advantage of their benefits.

Scaling features in this manner allows all features to be scaled in the same way, thereby avoiding problems such as the following:

1. Loss in Accuracy

2. When data values are of different magnitudes, requiring more computational power.

For instance, The high, low, and volume feature values in the data set we used in this project are particularly important. For volume you can find values between 0 and millions. Those of high and low are expressed in decimals. The LSTM algorithm tries and finds optimized values to handle large value feature values when applied to such data. Thus result in suboptimal results. A feature scaling approach can be applied here. In order to enable the deeep learning algorithms to behave better, the value of features must be transformed in a similar range as all other features.

In normalization, features are rescaled from 0 to 1, which is a form of min-max scaling. In order to normalize the data, one or more feature columns can be scaled using the min-max method. Using min-max scaling, the following formula can be used to normalize data. The use of normalization is necessary when the data must be within bounded intervals.

Divide the maximum value encountered by the range between the maximum and minimum or subtract the minimum value and divide the range by the maximum and minimum.

$$y = (x–min)/(max–min) \tag{3.1}$$

Sklearn.preprocessing contains a class called MinMaxScaler, which is used for normalizing data.

1. For scaler construction, it is necessary to fit the scaler using training data. To do this, the minimum and maximum observable values will be estimated using the training data using fit() function.

2. Scale the training data: This means that your model will be trained using the normalized data using transform() function.

3. Henceforth follow the code to apply the scaling to data. In other words, you can prepare new data in the future based on which to make predictions.

With the MinMaxScaler, variables are rescaled into a range of [0,1], although a preferred scale can be specified as a tuple containing the min and max for each variable.

3.5 PREDICTION MODEL

3.5.1 Keras

The Keras library is an High-level open-source library that runs on Theano, CNTK, or TensorFlow, and is written in Python. A faster neural network experimentation is made possible by its extensibility, user-friendliness, and scalability. CNNs can be set up separately or together. CNN's are not the only types they support. The program works quickly and easily with Python. In terms of neural layers, cost functions, optimization schemes, activation features, and regularization schemes, all standalone modules that can be combined to create models were designed for people, not computers. Modules can be introduced quickly as groups and functions change. Modules can be introduced quickly as groups and functions change. Unlike separate configuration files, all models are embedded in the code. As a backend engine, Keras uses a specialized library of tensor manipulation that can handle low-level calculations such as tensioner products, convolutions, etc., rather than managing all of the low level calculations. Rather than integrating a single library of tensors and operations, Keras provides a link to many back-end engines instead of depending on a single library of tensors and operations. Those are CNTK, Tensorflow, Theano.

3.5.2 Pandas

Pandas is a free, open-source data-processing library designed mostly for intuitively and easily working with relational or labeled data. For manipulating numerical data and time series, it provides a variety of data structures and operations. NumPy is used to implement this library. Pandas is a fast, high-performance package that provides users with all of the capabilities of NumPy.

Import the pandas Library:

import pandas as pd

Pandas are called pd in this context. The alias does not require the library to be imported, it just helps write less code every time a property or method is called. For manipulating data, Pandas provide the following two data structures:

1) Series

Suitable for holding any type of data (integer, string, float, Python objects, etc.), Pandas Series are labelled arrays. Each axis label is referred to as an index. An Excel sheet contains a column named Pandas Series. It is not necessary that a label be unique, but it must have a hashable form. Datasets can be loaded into Pandas Series from any of a variety of sources, including databases, CSV

files, and Excel files. There are several ways to create Pandas Series, including lists, dictionaries, scalar values, etc.

2) Dataframe

Pandas DataFrame is a two-dimensional data structure that allows for a two-dimensional size mutability with a possibly heterogeneous layout. Data frames are two-dimensional data structures, i.e., they align data along rows and columns in a tabular manner. There are three primary components to Pandas DataFrame, namely data, rows, and columns. It is normal practice to create Pandas DataFrames by loading the data from an existing storage source, such as a SQL database, CSV file, or Excel file. There are three types of Pandas DataFrame: lists, dictionaries, and dictionaries of dictionaries.

Data scientists use Pandas. Pandas are used with other libraries for data science since they are combined with pandas. Pandas incorporates many NumPy structures, which means that many NumPy structures are used or replicated in Pandas. Pandas data are often input into Matplotlib plot functions, SciPy statistics, and Scikit-Learn Machine Learning algorithms. The Pandas program can be run from any text editor, but Jupyter Notebook is recommended because it allows for code to be executed in a particular cell rather than in the entire file. Plots and data frames in Pandas can also be viewed with Jupyter.

3.5.3 Numpy

In general, Numpy is a package used to work with arrays. Furthermore, Numpy can also be used with Python as a multidimensional container for generic data kage that can be used for scientific computing. The library implements a multidimensional array object, various derived objects (such as masked arrays and matrices) and a number of routines that perform fast operations on arrays, such as mathematical, logical, shape, sorting, selection, I/O, discrete Fourier transformations, basic linear algebra, basic statistical operations, and random simulations.

Import numpy as np

NumPy is built around ndarray, an object that is at its center. There are n-dimensional arrays of homogeneous data types in this program, and many operations are compiled for speed. NumPy arrays differ from Python sequences in several important ways:

1) As opposed to Python lists (which expand dynamically), NumberPy arrays start out with a fixed size. When you change the size of an ndarray, the original array is deleted and a new one is created.

2) NumPy arrays must all contain the same type of elements, so their memory sizes are the same. In Python, including NumPy, it is possible to have arrays of different sized objects, allowing for arrays of varied sizes.

3) In addition to supporting advanced mathematics operations, Numpy arrays can support other types of data operations as well. The Python built-in sequences aren't typically as efficient and

require less code to perform such operations.

4) Scientists and mathematicians increasingly are using Python-based packages with NumPy arrays. Although Python sequences may be input, NumPy arrays are converted beforehand, and NumPy arrays are often output. Thus, just knowing how to use Python's built-in sequences types is not sufficient to effectively use today's scientific/mathematical Python software - one also needs to know how to use NumPy arrays.

3.5.4 MatplotLib

MatplotLib library provides data visualization and graphical plotting functionality for Python and NumPy, which are both cross-platform. Because of this, it is an excellent open source alternative to MATLAB. In addition to embedding plots in GUI applications, matplotlib's APIs (Application Programming Interfaces) are available as well. Visualizing data with Matplotlib is an intuitive and easy-to-use Python library. The Matplotlib Python library provides amazing visualizations of arrays in 2D.

Matplotlib is structured in such a way that generating a visual data plot is generally as simple as typing a few lines of code. There are two APIs that the scripting layer provides:

1) Matplotlib.pyplot, which is at the top of the hierarchy, provides the Python code objects for pyplot.

2) It is an OO (Object-Oriented) API that allows objects to be combined in a more flexible manner than Pyplot. Matplotlib's backend layers can be accessed directly through this API.

A MATLAB-like stateful interface is provided by the pyplot API. MATLAB was originally designed as an open source alternative to matplotlib. This is a more powerful, customizable, and easier to use API than Pyplot, but the interface is more difficult to learn. Therefore, pyplot is used more frequently, and as such is referred to here as the default interface. In order to work with plots, you need to understand matplotlib's pyplot API:

There is a top-level container for Figure in Matplotlib.pyplot. All the visual elements embedded in a plot including one or more axes are included.

There are almost all the graph elements in matplotlib.pyplot.axes: Axis, Tick, Line2D, Text, etc., with axes also setting the coordinates. It refers to the plotted area of data. Usually, the X- and Y-axises are present, as well as perhaps a Z-axis, as well.

Visualizing huge amounts of data in easily digestible visuals is one of the greatest benefits of visualization.There are many plots available in Matplotlib:

1) Line plot

2) Bar Plot

3) Histograms

4) Scatter plot

3.5.5 LSTM

Recurrent NNs (RNNs) produce explicit time models through their self-connection between hidden layers and improve their nodes to store long-term information. In terms of natural language processing and audio frequency analysis, RNNs have shown positive results. Traditional RNNs contain hidden layers that are connected between the nodes. Network models are able to memorize because of these recurrent feedback links. Consequently, they can model time-varying information. Depending on the timeliness of data transfer, the model depth will be determined. The problem with earlier RNNs is that they cannot handle large time spans of information and can cause vanishing gradients when used for large time span models. Deep RNNs can model effectively on time scales since the nodes are optimized. This prevents the occurrence of vanishing gradients.

In the realm of series data analysis, RNN models with long- and short-term memory nodes can be used effectively and expanded to address a variety of problems. In handwriting recognition, concise translation of natural language, and analysis of audio frequency data, these models have achieved exceptional results in long-term time-dependent information capture. In Recurrent Neural Networks (RNNs), Long short-term memories (LSTM) are used to prevent the output of a neural network from either decaying or exploding as it goes through feedback loops. Recurrent neural networks are better at recognizing patterns thanks to feedback loops. Long-short-term memory networks, which alleviate a phenomenon known as vanishing gradients, provide better performance and reduce the requirement of past input memory for solving sequence learning tasks. The LSTM node consists of what are known as dynamic gate structures such as input, forget, and output gate structures, as shown in Figure 3.3.

Furthermore, LSTM nodes contain a recurrent logic structure called a nerve cell, which records information about a node. Let us examine an LSTM node in more detail.

1. $A^{(i)}$, $A^{(f)}$, $A^{(c)}$, $A^{(o)}$ are the input data weights.
2. $R^{(i)}$, $R^{(f)}$, $R^{(c)}$, $R^{(o)}$ are the recurrent data weights.
3. V belongs to R^N are peehole weight.
4. $d^{(i)}$, $d^{(f)}$, $d^{(c)}$, $d^{(o)}$ belongs to R^N are the offset weights.
5. At time t, $x^{(t)}$, $y^{(t)}$ represent the output and input of the node.
6. Formula 3.2 represents the output $f^{(t)}$ of the forget gate at time t, where t is the time of the gate.

$$f^{(t)} = sigma(A^{(f)}x^{(t)} + R^{(f)}h^{(t-1)} + d^{(f)}) \tag{3.2}$$

7. Formula 3.3 represents the output $i^{(t)}$ of the input gate at time t, where t is the time of the gate.

$$i^{(t)} = sigma(A^{(i)}x^{(t)} + R^{(i)}h^{(t-1)} + d^{(i)}) \tag{3.3}$$

8. The input and cell structure of a node are represented by $c^{(t)}$,$C^{(t)}$, respectively, at time t, expressed in formulas 3.4 and 3.5

$$C^{(t)} = (i^{(t)}. * c^{(t)} + f^{(t)}. * C^{(t-1)}) \tag{3.4}$$

9. Formula 3.6 represents the output $O^{(t)}$ of the output gate at time t, where t is the time of the gate.

$$O^{(t)} = sigma(A^{(o)}x^{(t)} + R^{(o)}h^{(t-1)} + d^{(o)}) \tag{3.5}$$

10. figure 3.7 $h^{(t)}$ is the final output of the node.

$$h^{(t)} = (O^{(t)}. * tanh(c^{(t)})) \tag{3.6}$$

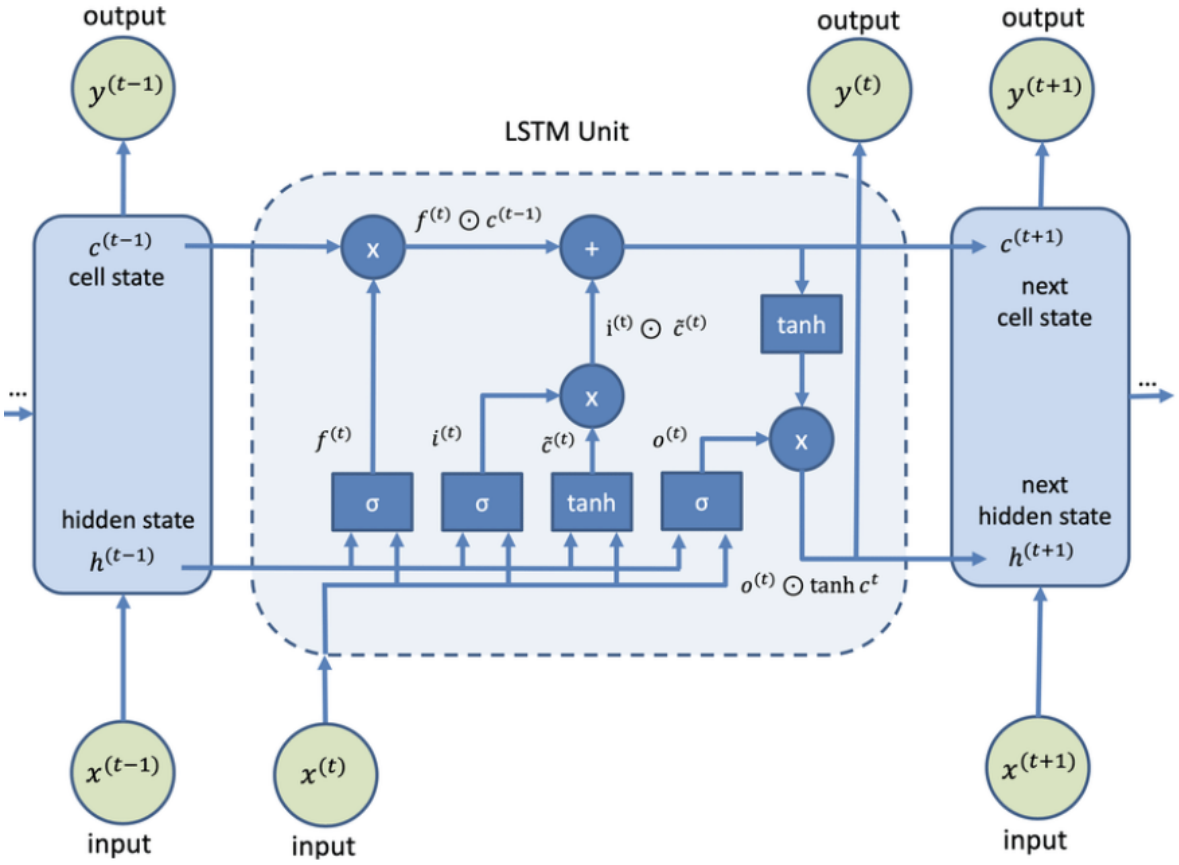

Figure 3.3: Architecture of LSTM

LSTM nerve cell models are elaborated with input gates and output gates, which respectively determine the amount of information that can be added to and output from the nerve cell after processing. Accordingly, the forget gate indicates how much of the output from the previous moment will be retained for calculations in the next moment. Due to their unique node structure, LSTM NNs can capture mid- and long-term data, whereas RNNs, in comparison, suffer from time scale gradient vanishing problems, which are caused by generalized additive models.

LSTM networks consist of multi-hidden layer nodes that are defined in this study. Multiple LSTM node layers form the hidden layers, and the layers are connected to one another the same way as the hidden layers of RNNs. The hidden layer architectures are typically created using experiences from previous studies. The model is set up so that all the hidden layers have the same number of nodes in order to reduce the number of comparison parameters. So the model consists K LSTM nodes for each of its L hidden layers.

A number of sequence learning problems can be tackled using LSTMs due to their ability to learn long-term dependencies, including language modeling and translation, acoustic modeling of speech, speech synthesis, speech recognition, audio and video data analysis, handwriting recognition, and protein secondary structure prediction.

3.5.6 Optimizer

Optimizers are algorithms or methods used to minimize an error function(loss function)or to maximize the efficiency of production. Optimizers are mathematical functions which are dependent on model's learnable parameters i.e Weights Biases. Optimizers help to know how to change weights and learning rate of neural network to reduce the losses.

Adam(Adaptive Moment Estimation)

Adam optimizer is one of the most popular and famous gradient descent optimization algorithms. It is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients , similar to momentum and also the decaying average of the past squared gradients , similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods.

$$\mathrm{w_t} = \mathrm{w_{t-1}} - \frac{\eta}{\sqrt{\mathrm{S_{dw_t}} - \varepsilon}} * \mathrm{V_{dw_t}}$$

$$b_\mathrm{t} = b_\mathrm{t-1} - \frac{\eta}{\sqrt{\mathrm{S}_{db_\mathrm{t}} - \varepsilon}} * \mathrm{V}_{db_\mathrm{t}}$$

Advantages:

1) Easy to implement

2) Computationally efficient.

3) Little memory requirements.

3.5.7 Loss

Loss is a value that represents the summation of errors in our model. It measures how well (or bad) our model is doing. If the errors are high, the loss will be high, which means that the model does not do a good job. Otherwise, the lower it is, the better our model works.

To calculate the loss, a loss or cost function is used. There are several different cost functions to use. Each penalizes errors in different ways, and the problem determines which one is better to use. Cross-Entropy and Mean Squared Error are the most commonly used for classification and regression problems, respectively.

However, whether the loss is high or low is not the most important inference we can learn from it. If we plot loss results over time, we can see whether our model is learning, and how fast.

This is because, in Deep Learning, the loss function is used by the model to learn. The goal of the model is to minimize the value of the loss. This is done by using techniques such as gradient descent, which changes the model parameters using the information of the result of the loss.

Seeing the loss over time can yield interesting findings of our models. If the loss value is not decreasing, but it just oscillates, the model might not be learning at all. However, if it's decreasing in the training set but not in the validation set (or it decreases but there's a notable difference), then the model might be overfitting. In other words, it might be overlearning from the training examples, becoming useless in new examples. If that's the case, it would be interesting to use regularization, use simpler models, or, in Deep Learning, just reduce the learning rate.

3.5.8 Mean Squared Error

When developing a model for a system, the evaluation of the model is an extremely important aspect. The mean squared error of the model's prediction, when used as a parameter to validate its quality, is an excellent parameter for determining whether it can be considered as reliable.

A regression line's mean squared error represents the distance between the line and a set of data points. An estimate of the expected loss during years with squared errors should be calculated by applying the risk function. Mean Square Error (MSE) is calculated as the mean or average of the difference between the actual and estimated values.

From Mean Squared Error we can observe the following:

1) Since forecasted values may differ from actual values in either direction, a sum of the differences may equal zero. A false conclusion can be drawn that the forecast is accurate.

2) When this square is compared to others, the difference becomes larger.Therefore, this one high value results in higher means. A large number of deviators or outliers influence MSE.

3) When we compute a square, we find that all errors are positive, and the mean is positive, indicating there is some difference between the estimates and the actuals. Lower mean indicates a closer match between forecast and actual.

In Data Science, this can be used as a measure to evaluate models, since this indicates how close the forecast is to reality.

MSE as Model Evaluation Measure

In Supervised Learning, the data set includes independent variables as well as dependent variables. The predictive model is built using independent variables and then the target is predicted. A regression model can be used to predict the dependent variable if it is numeric. Model evaluation can be done using MSE in this situation.

We find linear regression lines for given data points by analyzing the data.The MSE method can be used to determine which line best describes given data points. In the above diagram, predicted values are shown as points on the line and actual values are shown as circles. Distance between data

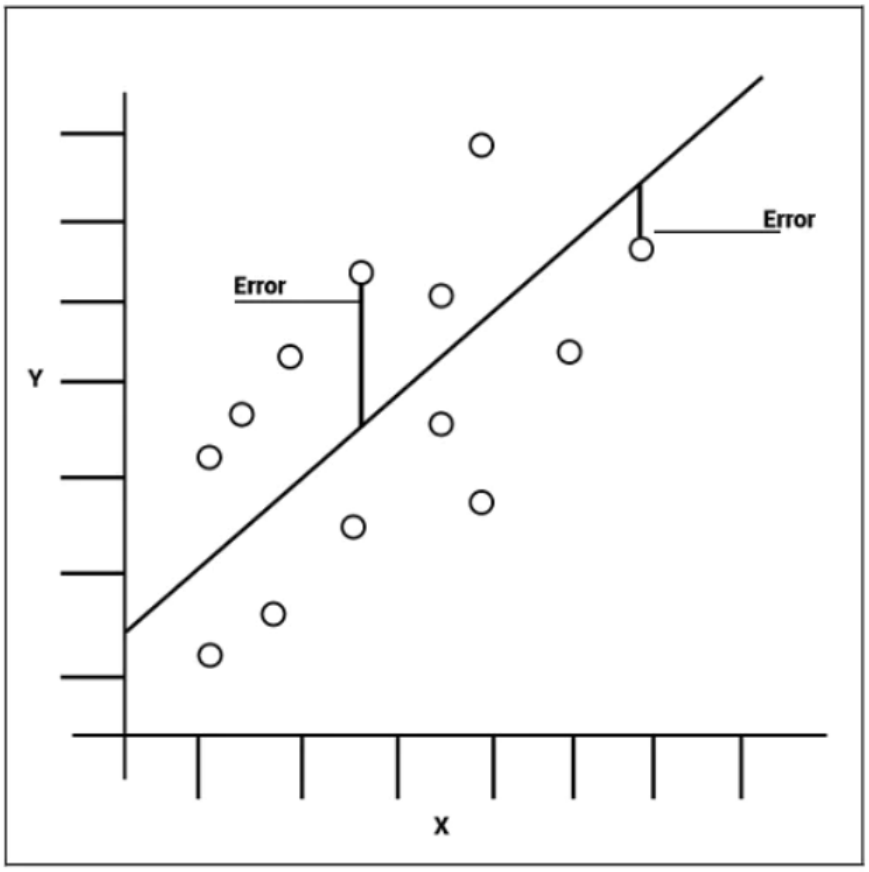

Figure 3.4: Regression Line

point and fitted line is used as the indicator of prediction error. Calculated as the average of all squares for all data points, MSE is the average of all the LSEs for that line. A line with the minimum MSE or the least MSE out of all possibilities for a given dataset is considered as the best fit.

Let N be the number of data points in a given dataset. SSE1 signifies the sum of squared errors, SSE2 signifies the sum of averaged errors, and so on. The sum of squared errors for each line are SSE1/N, SSE2/N, ... SSEn/N. The minimum sum of squared errors is also for the line having the lowest MSE.To find a regression line, many best-fit algorithms use methods based on least-squared-error methods. Due to the error being squared, the MSE unit order is higher than the error unit. Many times, the square root of the MSE is used to obtain the same unit order. Such errors are called Root Mean Square Errors (RMSEs). The root mean square error is equal to SQRT(MSE). Additionally, it is used to measure the effectiveness of models.

CHAPTER 4

IMPLEMENTATION

4.1 PROPOSED APPROACH

The proposed framework that learns online anticipating the close costs of the stock with the assistance of Long Short Term Memory (LSTM). The Long Short Term Memory (LSTM) is a counterfeit intermittent neural system (RNN) design used in the field of deep learning, Unlike standard feed forward neural systems, LSTM has input associations. Not only does the procedure not focus on single information (e.g. pictures) but also on full information arrangements, (For example, a speech or a video). For example, LSTM is material for undertakings, such as unpartitioned, associated penmanship recognition, speech recognition and recognition of peculiarities in arranged traffic or IDS (interruption location frameworks).

Before processing the data there is a important step that is to collect the information from market. Information assortment is the principle step in our proposed framework importing of the information from advertise clearing organizations like Kaggle and github. The dataset that will be utilized in the market expectation must be utilized to be separated dependent on different perspectives. Information assortment additionally supplements to upgrade the dataset by including more information that is outside. Our information for the most part comprises of the earlier year stock costs. For python available packages for retrieving the data pandas.

The next step is to preprocess the data; in this step the Information Pre-Processing is a significant advance in information mining here the change in crude information into a basic configuration is required. The information which is retrieved from source will be conflicting, fragmented and it will contain mistakes. The preprocessing step will purify the information; toward the end there is a need to perform highlights scaling which will restrict the factors. The preparation of the model incorporates cross-approval, which is a very well-founded, projected execution of the model using the preparation information. the purpose of the tuning models is to explicitly tune the calculation training is to add information to the calculation itself. The test sets are immaculate, as a model ought not to be made a decision about dependent on concealed information. Scale up the information to the genuine offer

costs. The final step is to draw the data using visualization technique that helps to show the variation of data in the outcome of our algorithm.

4.2 METHODOLOGY IMPLEMEMNTATION

To predict stock market values, we need to know the data we will use. APPLE's stock price will be examined in this section based on NIFTY dataset spanning over 41 years(1980 to 2021). Our total data set consists of 10272 rows in which we are using 80% i.e., 8217 rows of our dataset as training set and remaining 20% i.e., 2055 rows as testing set.

Using numerical analysis, a model for predicting APPLE stock prices is developed using recurrent neural networks. This type of model is good for learning and predicting time series. The best candidate for this task is RNNs, since the stock market data is a time series. This is accomplished by using a special type of RNNs, known as Long Short-Term Memory. A Long Short-Term Memory (LSTM) cell is a specially designed cell that may help the network to memorize long-term dependencies, as mentioned in Chapter 3. The aim of numerical analysis is to determine the trends and patterns of stock prices over time.

Every working day of the stock market, KAGGLE updates the value of the stocks of APPLE. Since the market is closed on weekends and holidays, there is a gap between Saturdays and Sundays. For each date, we provide the Opening Value, the Highest and Lowest values of the stock on the same day, as well as the Closing Value.

Adjusted Close Value refers to a stock's value after dividends are declared. Also, the total number of stocks traded in the market is provided. With this information, Machine Learning/Data Scientists can analyze the historical data of the APPLE stock and formulate algorithms that can detect trends.

4.2.1 Parameters used

List of parameters/Symbols used in this project is listed in Table 1.

Table 4.1: Parameters used

Classifier	**Meaning**
Date	Date of stock price
open	Opening price of share
close	closing price of share
High	Highest share value of the day
Low	Lowest share value of the day
Volume	Number of shares traded
Adj close	Price after stock dividend

4.2.2 Reading the dataset

There are 10272 rows. The dataset is loaded and read using pandas. Data.head() returns number of rows. We gave 2 so it returns first 2 rows. Then Data.drop removes the specific column or row which is not needed. Here we removed (Adj close) column which is not needed for our prediction. We converted date column to date time object which is previously a string.

```
[ ]  data = pd.read_csv('/content/drive/MyDrive/AAPL.csv')
     data.head(2)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	0.100600	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	0.095352	175884800

```
[ ]  # drop not required column
     data.drop(columns=['Adj Close'], inplace=True)
```

```
[ ]  data['Date'] = pd.to_datetime(data['Date'])
     data
```

	Date	Open	High	Low	Close	Volume
0	1980-12-12	0.128348	0.128906	0.128348	0.128348	469033600
1	1980-12-15	0.122210	0.122210	0.121652	0.121652	175884800
2	1980-12-16	0.113281	0.113281	0.112723	0.112723	105728000
3	1980-12-17	0.115513	0.116071	0.115513	0.115513	86441600
4	1980-12-18	0.118862	0.119420	0.118862	0.118862	73449600
...	...	...	...	...	...	...
10267	2021-09-01	152.830002	154.979996	152.339996	152.509995	80313700
10268	2021-09-02	153.869995	154.720001	152.399994	153.649994	71115500

Figure 4.1: Reading data using Pandas

We plotted the graph of apple data before training using matplotlib.

```
[ ]  fig = plt.figure(figsize = (25, 7))
     plt.plot(data.Date,data.High, color = 'red', label = 'APPLE Price')
     plt.xlabel('Year')
     plt.ylabel('$')
     plt.legend()
     plt.show()
```

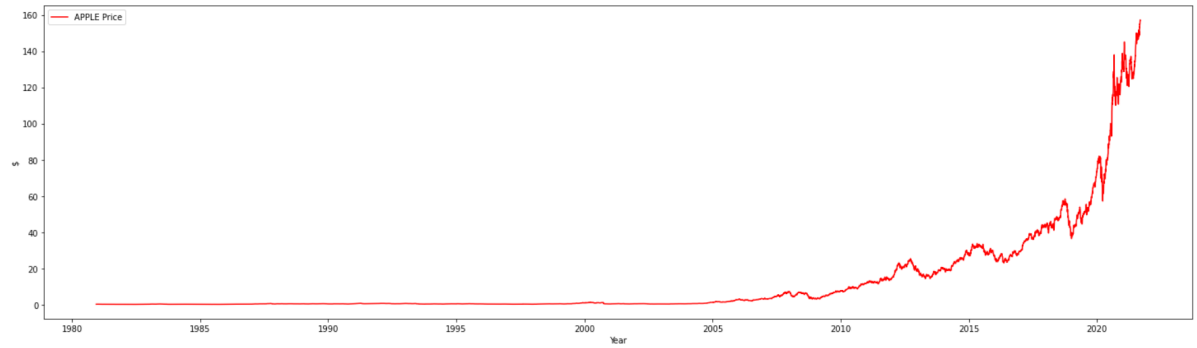

Figure 4.2: Apple stock price before prediction

4.2.3 Splitting tha data

We split the data into training and testing sets. We use 80% of our for training. Then we split our data using train_size*data.shape. So we get 80% of the data. Then we choose the factors we need, Later we splitted it from 1st row to split_index is training set and split_index to last row is testing set as you can see in figure 4.3.

```
# 80% will be used for traning, and 20% for testing

train_size = 0.8        # 80%
split_index = int(train_size * data.shape[0])

factors_column = ['Open', 'High', 'Low', 'Close', 'Volume']
y_col_index = 3 # Close

train_set = data[factors_column].values[:split_index]
test_set = data[factors_column].values[split_index:]
```

```
split_index
```

```
8217
```

Figure 4.3: Train and Test split

4.2.4 Preprocessing using min max scaler

To ease the convergence of the algorithm, it is useful to normalize the data. We see here what are the maximum and minimum values in the data and normalize them accordingly using min max scaler. We imported minmaxscaler from sklearn for preproccesing for better learning the model. We scale the data from 0 to 1 as in fugure 4.4. We need scaling because some features may have high impact than others which will affect the performance of predictions. In normalization, features are

```
from sklearn.preprocessing import MinMaxScaler

# scale our price from 0 to 1

sc = MinMaxScaler(feature_range = (0, 1))
train_set_scaled = sc.fit_transform(train_set)
test_set_scaled = sc.fit_transform(test_set)
```

```
train_set_scaled[0]
```

```
array([0.00314274, 0.00315307, 0.00317783, 0.00316636, 0.0631981 ])
```

Figure 4.4: Normalizing the data

rescaled from 0 to 1, which is a form of min-max scaling. In order to normalize the data, one or more feature columns can be scaled using the min-max method. Using min-max scaling, the following formula can be used to normalize data. The use of normalization is necessary when the data must be within bounded intervals. The LSTM algorithm tries and finds optimized values to handle large value feature values when applied to such data. Thus result in suboptimal results.

4.2.5 Time window

While doing statiscal analysis on any dataset we need to calculate various statiscal measure in various form. Have you ever tried to calculate any measure for a specific numbers of rows and then moving to another set of row by increasing the index value of every row by one. It will give us the statiscal measure for every set of data and by this we can get the idea that how the measure is changing with the rows. This can be done by time window function.

This python source code does the following : 1. Creates your own time series data.

2. Adding new columns to datagram

By using time_window parameter we combine data of 60 days. We print the shape of our training set of closing price for APPLE Dataset. We are using 60days price for predicting the next day price.

```
# this function will combine data of 60 days (we can change it using time_window parameter)
days_step = 1         # skip 7 days in between, can be set to 1 day
time_window = 60     # 8 weeks
def generate_data(series, y_col_index, time_window=60, days_step=1):
  X = []
  y = []
  for i in range(60, len(series)):
    X.append(series[i-time_window: i: days_step])
    y.append(series[i, y_col_index])
  return (np.array(X), np.array(y))
```

```
X_train, y_train = generate_data(train_set_scaled, y_col_index, time_window, days_step)
X_test, y_test = generate_data(test_set_scaled, y_col_index, time_window, days_step)
```

```
print('Training data shape (Input): ',X_train.shape)
print('Training data shape (Output): ',y_train.shape)
print('We will observe {} days price to predict next day price'.format(X_train.shape[1]))
```

```
Training data shape (Input):  (8157, 60, 5)
Training data shape (Output):  (8157,)
We will observe 60 days price to predict next day price
```

Figure 4.5: Combining data using time_window parameter

4.3 LSTM IMPLEMENTATION

Recurrent Neural Networks (RNN) are a family of neural networks that excels in learning from sequential data. A class of RNN that has found practical applications is Long Short-Term Memory (LSTM) because it is robust against the problems of long-term dependency. There is no shortage of articles and references explaining LSTM. There is also no shortage of good libraries to build machine learning applications based on LSTM. Keras has now over 50,000 stars at the time of this writing

suggesting a strong popularity among machine learning practitioners.

Here is the code for building LSTM:

model = Sequential()

layer 1 model.add(LSTM(units = 50, return_sequences = True, input_shape = X_train.shape[1:])) model.add(Dropout(0.2))

layer 2 model.add(LSTM(units = 30, return_sequences = True)) model.add(Dropout(0.2))

layer 3 model.add(LSTM(units = 10, return_sequences = True)) model.add(Dropout(0.2))

layer 4 model.add(LSTM(units = 1))

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            11200

 dropout (Dropout)           (None, 60, 50)            0

 lstm_1 (LSTM)               (None, 60, 30)            9720

 dropout_1 (Dropout)         (None, 60, 30)            0

 lstm_2 (LSTM)               (None, 60, 10)            1640

 dropout_2 (Dropout)         (None, 60, 10)            0

 lstm_3 (LSTM)               (None, 1)                 48

=================================================================
Total params: 22,608
Trainable params: 22,608
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.6: Build LSTM model

4.3.1 Epochs

Then we compile the model with 50 epochs in 64 batches.

The number of epochs is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

One epoch means that each sample in the training dataset has had an opportunity to update the internal model parameters. An epoch is comprised of one or more batches. For example, as above, an epoch that has one batch is called the batch gradient descent learning algorithm.

You can think of a for-loop over the number of epochs where each loop proceeds over the training dataset. Within this for-loop is another nested for-loop that iterates over each batch of samples, where one batch has the specified "batch size" number of samples.

The number of epochs is traditionally large, often hundreds or thousands, allowing the learning algorithm to run until the error from the model has been sufficiently minimized. You may see examples of the number of epochs in the literature and in tutorials set to 10, 100, 500, 1000, and larger.

It is common to create line plots that show epochs along the x-axis as time and the error or skill of the model on the y-axis. These plots are sometimes called learning curves. These plots can help to diagnose whether the model has over learned, under learned, or is suitably fit to the training dataset. The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters.

Think of a batch as a for-loop iterating over one or more samples and making predictions. At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient.

A training dataset can be divided into one or more batches.

When all training samples are used to create one batch, the learning algorithm is called batch gradient descent. When the batch is the size of one sample, the learning algorithm is called stochastic gradient descent. When the batch size is more than one sample and less than the size of the training dataset, the learning algorithm is called mini-batch gradient descent.

4.4 TRAINING OF OUR MODEL

Monitoring the training progress of networks used for machine learning is often important. It is possible to learn how training is going by plotting different metrics throughout the training session. Loss can be tracked in a variety of ways, such as how quickly the network improves and whether it is overfitting the data. A figure is created and training metrics are displayed every time the train Network creates a figure with 'training-progress' specified as the 'Plots' value in training Options.

```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
history = model.fit(X_train,
                    y_train,
                    epochs = 50,
                    batch_size = 64,
                    validation_data = (X_test, y_test))
```

```
Epoch 22/50
128/128 [==============================] - 14s 110ms/step - loss: 2.1003e-04 - val_loss: 4.4489e-04
Epoch 23/50
128/128 [==============================] - 14s 108ms/step - loss: 2.4897e-04 - val_loss: 3.8971e-04
Epoch 24/50
128/128 [==============================] - 14s 107ms/step - loss: 2.0630e-04 - val_loss: 4.4210e-04
Epoch 25/50
128/128 [==============================] - 14s 110ms/step - loss: 1.9784e-04 - val_loss: 4.9117e-04
Epoch 26/50
128/128 [==============================] - 14s 109ms/step - loss: 2.0697e-04 - val_loss: 3.8332e-04
Epoch 27/50
128/128 [==============================] - 14s 109ms/step - loss: 2.2274e-04 - val_loss: 6.1764e-04
Epoch 28/50
128/128 [==============================] - 14s 109ms/step - loss: 1.9997e-04 - val_loss: 4.5290e-04
Epoch 29/50
128/128 [==============================] - 14s 109ms/step - loss: 2.0864e-04 - val_loss: 5.8512e-04
Epoch 30/50
128/128 [==============================] - 14s 109ms/step - loss: 1.8053e-04 - val_loss: 5.8900e-04
Epoch 31/50
128/128 [==============================] - 14s 109ms/step - loss: 1.9898e-04 - val_loss: 5.0092e-04
Epoch 32/50
128/128 [==============================] - 14s 109ms/step - loss: 1.8808e-04 - val_loss: 4.3316e-04
Epoch 33/50
128/128 [==============================] - 14s 111ms/step - loss: 1.6927e-04 - val_loss: 3.3662e-04
Epoch 34/50
128/128 [==============================] - 14s 110ms/step - loss: 1.7932e-04 - val_loss: 3.6563e-04
Epoch 35/50
128/128 [==============================] - 14s 110ms/step - loss: 1.8100e-04 - val_loss: 3.8316e-04
Epoch 36/50
128/128 [==============================] - 14s 111ms/step - loss: 1.7213e-04 - val_loss: 3.7504e-04
Epoch 37/50
128/128 [==============================] - 14s 111ms/step - loss: 1.5647e-04 - val_loss: 3.0469e-04
Epoch 38/50
128/128 [==============================] - 14s 111ms/step - loss: 1.6930e-04 - val_loss: 2.8573e-04
Epoch 39/50
128/128 [==============================] - 14s 111ms/step - loss: 1.6598e-04 - val_loss: 4.3062e-04
```

Figure 4.7: Compiling LSTM model

Gradient estimation and updating of network parameters are performed in each iteration. Validation metrics are displayed in the figure if validation data is specified in the training options. The graph 3.10 with x-axis as Year and y-axis as loss, plots the following:
(Blue line)Training loss and (Red line)validation loss: The loss from each minibatch, its smoothed version, and the loss from the validation set constitute the training loss and validation loss, respectively.

The loss function is the Mean Squared Error if the final layer of your network is a classification layer. Train Network returns a trained network after it has completed training. When the training is complete, view the results to find out the accuracy of the final validation and the reason for the training termination. Final metrics are displayed on the plots for the final validation. It is often the case that the verification metrics used to evaluate a network's validation differ from the ones used

```
plt.plot(range(len(history.history['loss'])),history.history['loss'], label='loss')
plt.plot(range(len(history.history['val_loss'])),history.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
```

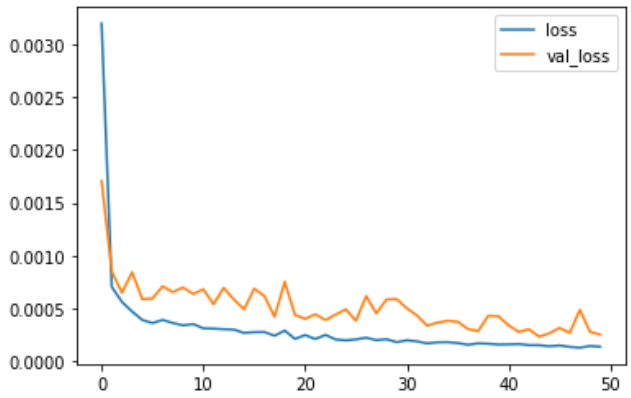

Figure 4.8: Loss Graph

during training if it contains batch normalization. During training, different operations are performed in batch normalization layers versus the final network.

Our next step is to use time window to decrease the model's loss. Figure 3.11 shows the graph with x axis as Year and y-axis as loss, after applying time window to all the columns. This involves making little changes to the input by considering all the columns. So that it will be like the algorithm is training on more data and biuld the LSTM again and plotted the graphs. After training the model, we obtained a 10% decrease in loss.

model = Sequential()

layer 1 model.add(LSTM(units = 128, return_sequences = True, input_shape = X_train.shape[1:])) model.add(Dropout(0.2))

layer 2 model.add(LSTM(units = 64, return_sequences = True)) model.add(Dropout(0.2))

layer 3 model.add(LSTM(units = 32, return_sequences = True)) model.add(Dropout(0.2))

layer 4 model.add(LSTM(units = 16, return_sequences = True)) model.add(Dropout(0.2))

layer 4 model.add(LSTM(units = 5))

We test our model by predicting values. We use 60 values after split index and comapre the actual closing price of the data and predicted price.

```
# this function will combine data of 60 days (we can change it using time_window parameter)
time_window = 7*4      # 4 weeks
days_step = 7          # skip 7 days in between, can be set to 1 day
def generate_data(series, time_window=60, days_step=1):
  X = []
  y = []
  for i in range(60, len(series)):
    X.append(series[i-time_window: i: days_step])
    y.append(series[i])                              # <---- only changed this, insetead of taking only closing price, every column value is used
  return (np.array(X), np.array(y))
```

```
X_train, y_train = generate_data(train_set_scaled, time_window, days_step)
X_test, y_test = generate_data(test_set_scaled, time_window, days_step)
```

```
print('Training data shape (Input): ',X_train.shape)
print('Training data shape (Output): ',y_train.shape)
print('We will observe {} days price to predict next day price'.format(X_train.shape[1]))
```

```
Training data shape (Input):  (8157, 4, 5)
Training data shape (Output):  (8157, 5)
We will observe 4 days price to predict next day price
```

Figure 4.9: Time window on all factors

4.5 TESTING OF TRAINED MODEL

A network is evaluated through testing. Using the network to test prediction efficiency shows how efficiently the network is categorizing the data. The images were compared with their predicted labels in Google COLAB. Testing data is shown in figure 4.10 and 4.11:

Testing of prediction and achieved 99% correct prediction as shown in Figure 4.11.

```
fig = plt.figure(figsize = (15, 5))
plt.plot(data.Date.values[ split_index+60 : ], data.Close.values[ split_index+60: ], 'r-', label = 'Actual APPLE Price')
plt.plot(data['Date'].values[split_index+60 : ], test_prediction[:,3], 'b--', label = 'Prediction')

plt.title('APPLE Price Prediction')
plt.xlabel('Date')
plt.ylabel('APPLE Price')
plt.legend()
plt.show()
```

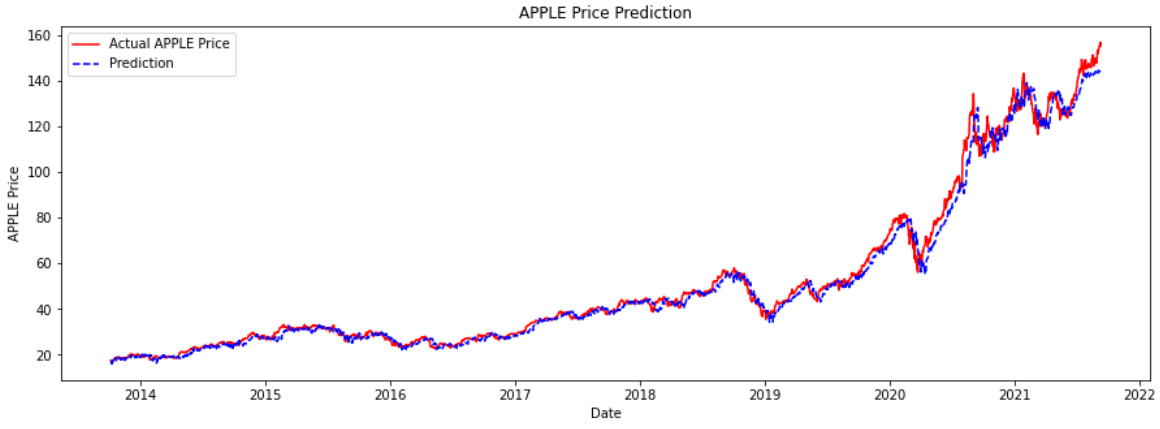

Figure 4.10: Testing of prediction

Then we create function for predicting next days price by using factors column and plotted it in the graph as shown in figure 4.12.

def predict_n_days(df, n=30, step=1):

```
fig = plt.figure(figsize = (25, 7))
plt.plot(data.Date, data.Close, 'r-', label = 'APPLE Price')
plt.plot(predicted_df.Date.values[-days: ], predicted_df.Close[-days:], 'b--', label = 'Future Prediction')

plt.title('APPLE Price Prediction')
plt.xlabel('Date')
plt.ylabel('APPLE Price')
plt.legend()
plt.show()
```

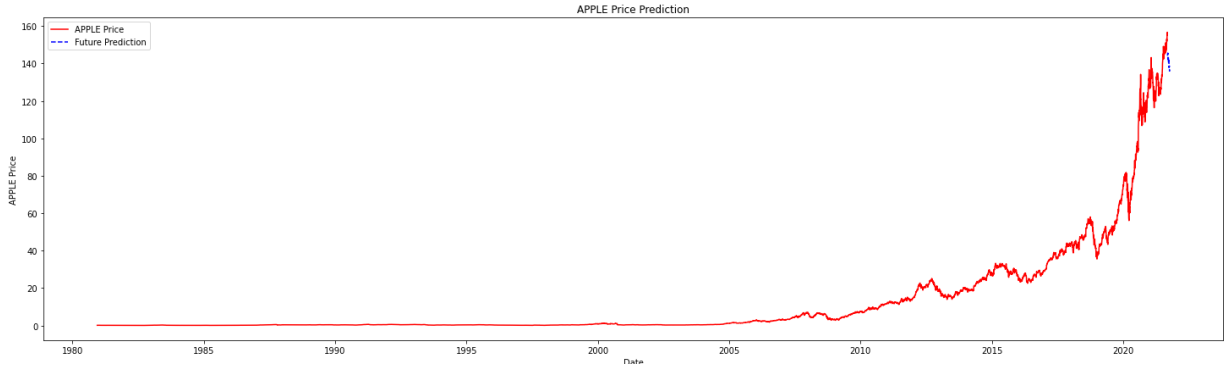

Figure 4.11: Prediction for next 30 days

for i in range(n):

X = df[factors_column].values[df.shape[0]-time_window::step]

X = sc.transform(X)

y = model.predict(np.expand_dims(X, axis=0))

y = sc.inverse_transform(y)[0]

next_day_prediction = key: value for key, value in zip(factors$_c olumn, y$)

next_day_prediction['Date'] = df.iloc[-1].Date + pd.Timedelta(days=1)

df = df.append(next_day_prediction, ignore_index=True)

return df

days= 30

predicted_df = predict_n_days(data,days,days_step)

4.6 SOFTWARE REQUIREMENT

Nowadays, machine learning and deep learning has become the hottest trend of the Computer Science industry. Many students are trying to learn and build amazing projects with it. We all know that just studying or reading or watching a tutorial is of no use if you didn't try it out on your own. But in order to do that, you need really advanced specifications, for your system to withstand such a workload. And not everyone can afford a laptop with such specifications.

Google Colab was developed by Google to provide free access to GPU's and TPU's to anyone who needs them to build a machine learning or deep learning model. Google Colab can be defined as an improved version of Jupyter Notebook. Jupyter Notebook is an application that allows editing and running Notebook documents through a web browser or an Integrated Development Environment (IDE). Instead of files, you will work with Notebooks. Programming Languages are an intermediate form between human-understandable language and machine understandable language. Every application is built using one of the many programming languages available. Maybe a person with a computer science background can understand, but not everyone can. Remember, as Software Developers, we develop applications for people with little computer science knowledge.

Consider you are creating a machine learning model to improve customer satisfaction for a local store, in that case you will have to explain how the model can do this task, and you can't just explain him with your code base. Most people facing this situation will prepare a separate presentation. Notebooks were created so that it is not necessary. Notebook documents can include executable lines of code along with text, images, figures, tables, graphs, equations, and much more graphical data. In simple words, Notebook documents are a way of creating human-readable executable documents.

Google Colab comes pre-installed with the most popular machine learning libraries. Colab comes pre-installed with Keras, PyTorch, TensorFlow, which saves you the time and hassle of setting up a local environment. Every Notebook you create in the Google Google Colab is saved on the cloud. This lets you access and work with those Notebooks from any machine. All you need is a browser and a reliable network connection, and you can work from anywhere and anytime. Collaboration is another amazing reason to choose Google Google Colab when you are working on a project with a team of developers. You can share your Notebook with your teammates and assign them roles so that they can only perform operations that fit their roles. Google Colab provides free access to GPUs and TPUs developed by Google Research. So you can work on your personal projects with powerful GPUs irrespective of your local machine.

Data science and machine learning researchers can utilize Google Colab. No installation is necessary to use this Jupyter notebook environment. Most of the libraries and packages in it can be imported directly without the need to manually install them, making it one of the best tools for data scientists. The main difference between normal IDEs and this tool is you don't need to install libraries. Python can be written and executed in your browser with Collaboratory, or "Colab" for short

1. You don't need to configure anything.
2. GPUs are available for free.
3. The sharing process is simple.

It does all this in just a few lines of code, with the ability to import an image dataset, train a machine-learning model on it, and evaluate the model. Colab notebooks run code on Google's cloud servers, so we have taken advantage of the power of Google hardware, including GPUs and TPUs. It only takes a browser to get started.

Google Colab provides tons of exciting features that any modern IDE offers, and much more. Some of the most exciting features are listed below:

Interactive tutorials to learn machine learning and neural networks.

Write and execute Python 3 code without having a local setup.

Execute terminal commands from the Notebook.

Import datasets from external sources such as Kaggle.

Save your Notebooks to Google Drive.

Import Notebooks from Google Drive.

Free cloud service, GPUs and TPUs.

Integrate with PyTorch, Tensor Flow, Open CV.

Import or publish directly from/to GitHub.

4.7 OVERLL DESIGN DEVELOPMENT

In the beginning of this project, we investigated what machine learning libraries are available, how Long-Short-Term Memory are designed, and collected datasets. This project focused on designing and optimizing neural networks, so it chose Keras from a variety of libraries. Other libraries exist, but Keras was selected because many tutorials and documentation are available for it. Keras was used to create a prediction model for this project, and the LSTM was designed based on several tutorials on how to incorporate Keras. To adapt the initial design for the intended use of this project, we began modifying the first network. As a deliverable for this project, the design of LSTM that we tested is being submitted. The model was also trained with the dataset in the genus-species level, which uses approximately 10272 rows, which should be sufficient for most situations. More ROWS will be added in the future. We are testing and training the model currently. There are several parameters that can be customized, including the training step number, output file, and data input file, however all of them default to placing all output files in the current working directory. To input images, the input directory must specify the data in the keras files folder, and the data must be in file. Each prediction result is read by the model and is shown on the screen as an output result.

CHAPTER 5

RESULTS AND DISCUSSION

Based on the deep neural networks, we were able to predict the stock price. Through the internet, a reasonable dataset has been collected and labelled accordingly to the application, then trained using a Long-Short-Term Memory. According to our requirements, we trained LSTM using Keras and implemented the network using Keras. Our network uses preprocessing, min max scaler, so we know that the network will yield high accuracy in classifying data. To make sure we get the best out of this training, we followed several trial and error methods. Python scripts used to implement the trained Keras model are very simple, since the bulk of the work on this project was learning the design of a LSTM and collecting the data. Besides the file of stock data, the classifier also takes a text file containing the labels used in the network, and trained the model itself as its input. Using the specified model, the classifier tests the data and compares the actual price according to the predictions' confidence level.

There is potential in this project for future improvements, which means that this is almost a successful project. Currently, there are huge number of rows in the datset that can be identified. A high level of prediction has been maintained throughout the process. A high prediction rate depends on how much data there is to analyze. For this reason, we made time windowing to combine data in order to improve the quality of the results. Stock price can currently be predicted 99% accurately. Our research and project have been successful in this regard. Each row was trained by using 8217 rows. Approximately 10272 rows make up the dataset. The LSTM is consistent when it comes to producing accurate classifications. They provide nearly 99% confidence.

We used Google COLAB to carry out the experiment. The dataset has been read by using pandas. The dataset is then pre-processed using min max scaler and the combined the rows using time window parameter, and the proposed LSTM algorithm embedded with the multi-label power dataset is run on it. Compared to other classifiers, the LSTM achieves higher prediction in predicting the stock price. The prediction model is able to predict the closing price of the stock based on previous data.

5.1 PERFORMANCE ANALYSIS

By measuring the loss of the prediction algorithms, it can be determined how well they perform. **Loss**: The definition of loss function is to measure the quality of model prediction. Loss functions are used to determine the error (aka "the loss") between the output of our algorithms and the given target value. It's used to measure the predicted value of your model f(x) And the real value Y The degree of inconsistency , It's a non negative real value function , Usually use L(Y, f(x)) To express , The smaller the loss function , The better the robustness of the model .

In regression problems, the mean squared error (MSE) is used to evaluate the performance of a regression model. The mean squared error (MSE) determines the distance between the set of points and the regression line by taking the distances from the set of points to the regression line and then swapping them. Distances are nothing but errors. Squaring is only done to remove negative values and to give more weight to larger differences.

If the MSE score value is smaller it means you are very close to determining the best fit line which also depends on the data you are working on, so sometimes it may not be possible to get a small MSE score value.

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

Figure 5.1 depicts the loss of training and testing sets for Long-short term memory. During testing, LSTM was able to achieve very minimum loss.

```
plt.plot(range(len(history.history['loss'])),history.history['loss'], label='loss')
plt.plot(range(len(history.history['val_loss'])),history.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
```

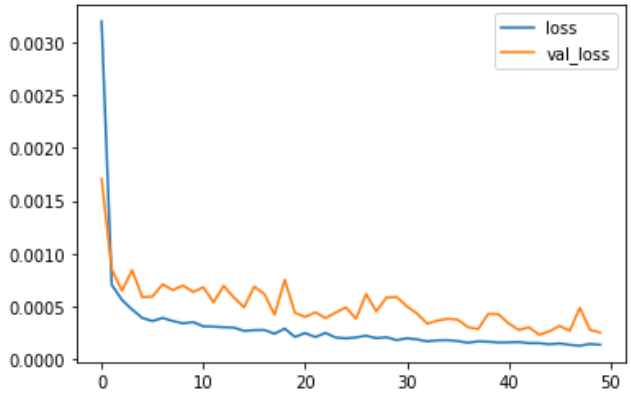

Figure 5.1: Simulation result

5.2 PERFORMANCE RESULTS

Google COLAB was used to perform the experiment. The dataset is read by using pandas. After preprocessing and time windowing the data, parameters are extracted from the dataset in preparation for applying the proposed LSTM algorithm integrated with multi-label dataset on power. In individual tests, LSTM achieves better prediction than all other methods when analyzing the prediction of stock prices. The predictions are derived from the analyzed stock price of APPLE data.

Table 5.1: Performance comparision of CNN with MLP and LSTM

Classifier	**Loss**
CNN	0.269
MLP	0.24
LSTM	0.002

The table 4.1 shows the performance of CNN, MLP and LSTM with loss. CNN and MLP didn't worked well compared to LSTM. We would like to point out that the application of CNN to selected features, turns to produce poor results. In fact, by implementing such an approach we lose the time-based structure of data. Then we trained MLP on transformed data, achieving MSE = 0.26, Hence a worse result. So we choosed LSTM as our algorithm.

5.3 OUTPUTS OF SIMULATION RESULTS

Figures 4.2, 4.3 shows the output of predicted and actual price of APPLE stock price, which was predicted correctly by our proposed deep neural netwroks model.

The implementation of proposed LSTM model using python which predicts the future price of APPLE share based on its historical data. The below visualization figure shows the visualization of APPLE prediction. In our paper the implementation of an algorithm which predicts the stock price of a share for given period of time, the below graph from our algorithm will show the predicted price of APPLE share. In the result shown in the below graph is the plotted form our algorithm outcome by applying 128 LSTM units for achieving the accuracy.

The Fig 5.2 is drawn from original dataset and also shown the result by comparing its correctness with the trained model from algorithm that is defined in the previous section. the "x" axis is Years. The "y" axis is share price. The data is slot of 10272 days is shown in the Fig 5.2

```
fig = plt.figure(figsize = (25, 7))
plt.plot(data.Date, data.Close, 'r-', label = 'APPLE Price')
plt.plot(data['Date'].values[split_index+60 : ], test_prediction[:,3], 'b--', label = 'Prediction')

plt.title('APPLE Price Prediction')
plt.xlabel('Date')
plt.ylabel('APPLE Price')
plt.legend()
plt.show()
```

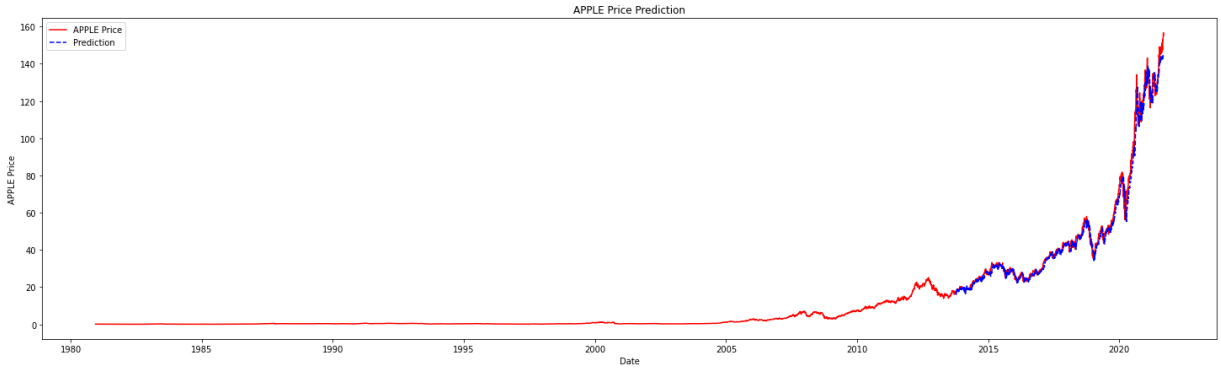

Figure 5.2: Output of Prediction

The Fig 5.3 is drawn from original dataset also shown the result by comparing its correctness with the trained model from algorithm which that is defined in the previous section. the "x" axis is Years.The "y" axis is share price. The data is slot of 2055 days is shown in the Fig 5.3.

The Fig 5.4 is drawn from original dataset also shown the result by comparing its correctness with the trained model from algorithm which that is defined in the previous section and price of next 30 days. The "x" axis is Years.The "y" axis is share price. The data is slot of 10302 days is shown in the Fig 5.4.

In the Fig 5.2, the graph has been plot for whole data set along with some part of trained data. the graph is showing the close price of APPLE share for upto 10272th day's closing price with very minimal loss. The algorithm has plotted the graph successfully along with the predicted price testing price (blue) and true price (red), there is a slight difference in predicting the price between the predicted price testing price (blue) and true price (red), which proves that our algorithm is able

```
fig = plt.figure(figsize = (15, 5))
plt.plot(data.Date.values[ split_index+60 : ], data.Close.values[ split_index+60: ], 'r-', label = 'Actual APPLE Price')
plt.plot(data['Date'].values[split_index+60 : ], test_prediction[:,3], 'b--', label = 'Prediction')

plt.title('APPLE Price Prediction')
plt.xlabel('Date')
plt.ylabel('APPLE Price')
plt.legend()
plt.show()
```

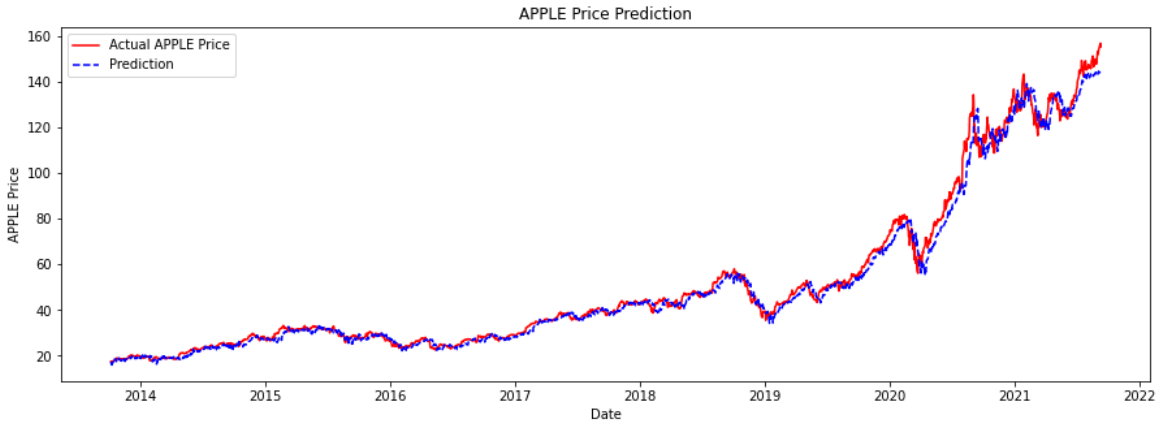

Figure 5.3: Output of Prediction

to predict the with minimum loss rate for the given complete data set of a particular share.

In the Fig 5.3, the graph is showing the close price of APPLE share for upto 2055th day's closing price with very minimal loss. The algorithm has plotted the graph successfully along with the predicted price testing price (blue) and actual testing price (red), there is a slight difference in predicting the price between the predicted price testing price (blue) and actual testing price (red), which proves that our algorithm is able to predict the with minimum loss rate of 0.002. The proposed algorithm is able to predict the share price with very low loss and error rate, if increase the epoch batch rates the training will be more efficient, in the above section we have used epoch batch size of 50 to predict the stock prices.

In the Fig 5.4, the graph is showing the close price of APPLE share for next 30 days closing price with very minimal loss. The algorithm has plotted the graph successfully along with the predicted price testing price (blue) and actual testing price (red), there may be a slight difference in predicting the price between the predicted price testing price (blue) and actual testing price (red), which proves that our algorithm is able to predict the with minimum loss rate of 0.002. The proposed algorithm is able to predict the share price with very low loss and error rate, if increase the epoch batch rates the training will be more efficient, in the above section we have used epoch batch size of 50 to predict the stock prices.

The figures shown in the previous section (fig 5.2, fig 5.3 and fig 5.4) of the proposed algorithm is able to predict the price, with loss: 0.002 in fig 5.5.

The Figure 5.6 shows the comparison between the proposed methodology and existing literatures. The Hidden markov model based model predicts the stock price with 66.034% [35], the

```
fig = plt.figure(figsize = (25, 7))
plt.plot(data.Date, data.Close, 'r-', label = 'APPLE Price')
plt.plot(predicted_df.Date.values[-days: ], predicted_df.Close[-days:], 'b--', label = 'Future Prediction')

plt.title('APPLE Price Prediction')
plt.xlabel('Date')
plt.ylabel('APPLE Price')
plt.legend()
plt.show()
```

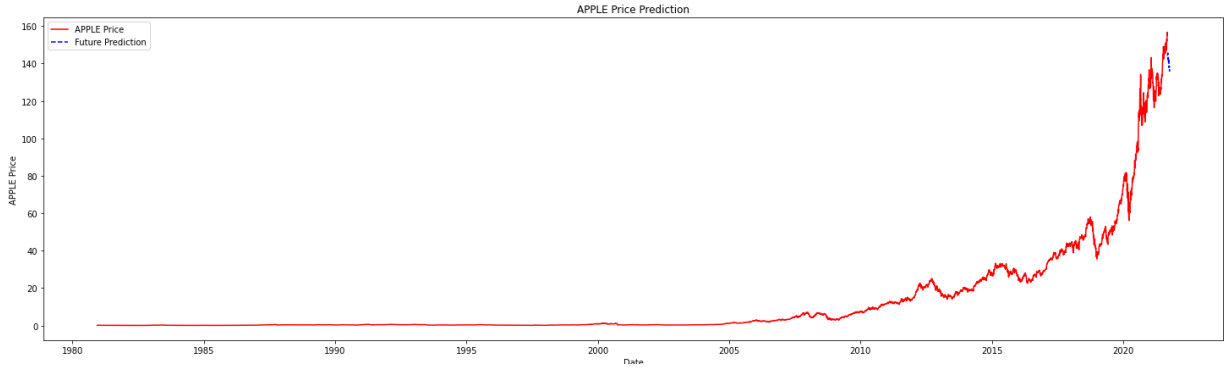

Figure 5.4: Output of Prediction for next 30 days

Multi-source Multiple Instance (M-MI) model achieved accuracy as 60.1% [34], the Random forest model gives accuracy as 62.90% [33], the LSTM-CNN achieved accuracy as 98% [18], the LSTM model achieved accuracy as 91% [30], the RNN gives accuracy as 96.1%, the LSTM model achieved accuracy as 95.83 and the CNN model achieved accuracy as 97.64% for Infosys stock [3], high-order structures, i.e., motifs achieved accuracy as 58.06% for APPLE stock [24] whereas our proposed model achieves accuracy as 99% for APPLE stock which is very accurate compared to other existing models.

```
plt.plot(range(len(history.history['loss'])),history.history['loss'], label='loss')
plt.plot(range(len(history.history['val_loss'])),history.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
```

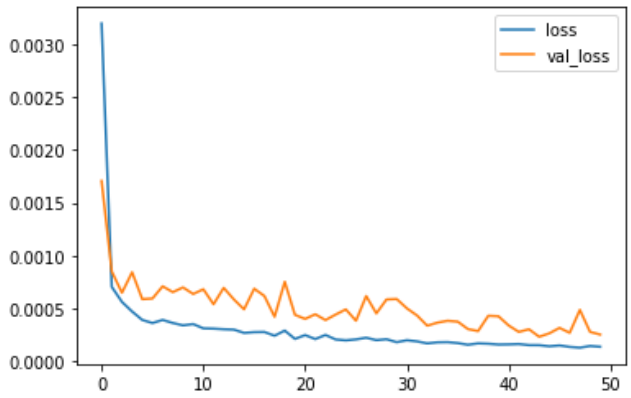

Figure 5.5: Loss in prediction

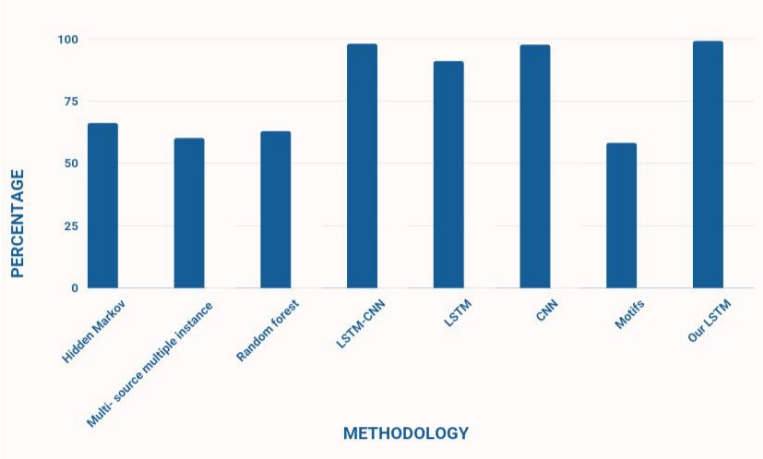

Figure 5.6: Comparison between proposed methodology and existing methods

CHAPTER 6

CONCLUSION

In this paper, We propose a deep neural networks based formalization for stock price prediction. It is seen that, deep neural network architectures are capable of capturing hidden dynamics and are able to make predictions. We trained the model using the data of APPLE and was able to predict stock price of APPLE. This shows that, the proposed system is capable of identifying some inter relation with in the data. Also, it is evident from the results that, LSTM architecture is capable of identifying the changes in trends. For the proposed methodology LSTM is identified as the best model. a prediction model is proposed for financial price data, which are non-stationary and relatively noisy time series, through three process steps, namely time series data processing, network model building and result evaluation and analysis. It uses the information given at a particular instant for prediction. Even though the other two models are used in many other time dependent data analysis, it is not out performing the LSTM architecture in this case. This is due to the sudden changes that occurs in stock markets. The changes occuring in the stock market may not always be in a regular pattern or may not always follow the same cycle. Based on the companies and the sectors, the existence of the trends and the period of their existence will differ. The analysis of these type of trends and cycles will give more profit for the investors. To analyze such information we must use networks like LSTM as they rely on the current information. The time series analysis with a RNN-based LSTM network model. Data de-noising and normalization are performed at the pre-processing stage. In addition, data structuring is also achieved by partitioning using a time window. Subsequently, a RNN-based model is designed and optimized by selecting an optimum activation function and optimization method. Ultimately, loss is used to evaluate the effectiveness of the model. Finally, various prediction methods, including the conventional CNN Deep learning, the a deep MLP model, a deep LSTM model involving no PSR process and the deep LSTM, are compared. The comparison of the prediction results for various stock indices for various periods demonstrates that the established RNN-based prediction model displays a higher prediction capacity than the other models. On the other hand, experimental results show the outperformance of our method on real financial time series datasets and validate the effectiveness on capturing trend information of stock shares. LSTM introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. In this work by

increasing the Epochs and batch size, the accuracy of prediction is more. In proposed method, we are using a test data that is used to predict which gives results that are more accurate with the test data. The proposed method is capable of tracing and prediction of stock market and the prediction will produce higher and accurate results. We are getting accurate results which will be more useful to stock analysts, Business analysts, Stock Market Investors. Our method sheds light on macroscopic pattern discovery in financial time series and provides a novel solution for price prediction. The results of the proposed model are compatible with researches that state that there is a strong relation between stock news and changes in stock prices.

6.1 FUTURE WORK

1) Our ongoing researches aim at treating more extended financial time series, and their features, in order to improve the training performance as well as related accuracy, for the particular type of neural networks considered. Techincal indicators, moving averages and stochastic oscillators, will be also taken into account to increase accuracy of prediction.

2) In future enhancement the inclusion of sentiment analysis from social media to understand what the market thinks about the price variation for a particular share and it can be implement this by adding twitter and Facebook API to our program as Facebook is a leading social media which has lots of market trend information posted by users.

REFERENCES

[1] Jürgen Schmidhuber, "Deep learning in neural networks: An overview", Neural Networks, vol. 61, pp. 85-117, 2015.

[2] Luca Di Persio and O. Honchar, "Artificial neural networks architectures for stock price prediction: Comparisons and applications", International Journal of Circuits Systems and Signal Processing, vol. 10, pp. 403-413, 2016.

[3] Sreelekshmy Selvin, "Stock price prediction using LSTM RNN and CNN-sliding window model", International Conference on Advances in Computing Communications and Informatics (ICACCI), pp. 1643-1647, 2017.

[4] Ayman E. Khedr, S.E. Salama and Nagwa Yaseen, "Predicting Stock Market Behavior using Data Mining Technique and News Sentiment Analysis", International Journal of Intelligent Systems and Applications (IJISA), vol. 9, no. 7, pp. 22-30, 2017.

[5] D. Ashok Kumar and S. Murugan, "Performance Analysis of Indian Stock Market Index using Neural Network Time Series Model", International Conference on Pattern Recognition Informatics and Mobile Engineering (PRIME), 2013.

[6] Jasemi M, Kimiagari AM, Memariani A (2011) A modern neural network model to do stock market timing on the basis of the ancient investment technique of Japanese Candlestick. Expert Syst Appl 38(4):3884–3890

[7] Vanstone B, Finnie G (2009) An empirical methodology for developing stockmarket trading systems using artificial neural networks. Expert Syst Appl 36(3):6668–6680

[8] Xiong Z (2011) Research on RMB exchange rate forecasting model based on combining ARIMA with Neural networks. J Quant Tech Econ 28(06):64–76 (in Chinese)

[9] Wu Q, Wang C, Tang Y (2013) Empirical research on volumeprice relationship based on GARCH models and BP neural network. J Sichuan Univ Nat Sci Edn 50(04):703–708 (in Chinese)

[10] Li X, Zhang Z (2014) Support vector machine method for financial time series prediction based on simultaneous error prediction. J Tianjin Univ Sci Technol 47(01):86–94 (in Chinese)

[11] Shen F, Chao J, Zhao J (2015) Forecasting exchange rate using deep belief networks and conjugate gradient method. Neurocomputing 167:243–253

[12] Ding X, Zhang Y, Liu T, et al (2015) Deep learning for eventdriven stock prediction. In Twenty-fourth international joint conference on artificial intelligence, pp 2327–2333

[13] Zhao Y, Li J, Yu L (2017) A deep learning ensemble approach for crude oil price forecasting. Energy Econ 66:9–16

[14] Krauss C, Do XA, Huck N (2017) Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the SP 500. Eur J Oper Res 259(2):689–702

[15] Lecun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521 (7553):436

[16] G. E. P. Box, G. M. Jenkins, G. C. Reinsel and G. M. Ljung, Time Series Analysis: Forecasting and Control, Hoboken, NJ, USA:Wiley, 2015.

[17] X. Dai and M. Bikdash, "Trend analysis of fragmented time series for mHealth apps: Hypothesis testing based adaptive spline filtering method with importance weighting", IEEE Access, vol. 5, pp. 27767-27776, 2017.

[18] F. E. Tay and L. Cao, "Application of support vector machines in financial time series forecasting", Omega, vol. 29, no. 4, pp. 309-317, Aug. 2001.

[19] K.-J. Kim, "Financial time series forecasting using support vector machines", Neurocomputing, vol. 55, no. 1, pp. 307-319, 2003.

[20] M. R. Hassan and B. Nath, "Stock market forecasting using hidden Markov model: A new approach", Proc. 5th Int. Conf. Intell. Syst. Design Appl. (ISDA), pp. 192-196, Sep. 2005.

[21] A. Gupta and B. Dhingra, "Stock market prediction using hidden Markov models", Proc. Students Conf. Eng. Syst. (SCES), pp. 1-4, Mar. 2012.

[22] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions", Eur. J. Oper. Res., vol. 120, no. 2, pp. 654-669, 2017.

[23] L. Zhang, C. Aggarwal and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns", Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, pp. 2141-2149, 2017.

[24] M. Wen, P. Li, L. Zhang, and Y. Chen, "Stock market trend prediction using high-order information of time series," IEEE Access, vol. 7, pp. 28299–28308, 2019.

[25] Zhang L, Zhang L, Teng W, Chen Y (2013) Based on information fusion technique with data mining in the application of finance early-warning. Procedia Comput Sci 17:695–703

[26] Kamley S, Jaloree S, Thaku RS (2013) Multiple regression: a data mining approach for predicting the stock market trends. Int J Comput Sci Eng Inf Technol Res 3(4):173–180

[27] Hiba Sadia, Aditya Sharma, Adarrsh Paul, SarmisthaPadhi, Saurav Sanyal- "Stock Market Prediction Using Machine Learning Algorithms", IJEAT, 2019.

[28] Raut Sushrut Deepak, Shinde Isha Uday, Dr. D. Malathi, "Machine Learning Approach In Stock Market Prediction", IJPAM 2017.

[29] M. S. Hegde, G. Krishna and R. Srinath, "An Ensemble Stock Predictor and Recommender System," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, 2018, pp. 1981-1985.

[30] M. Roondiwala, H. Patel and S. Varma, "Predicting stock prices using LSTM," International Journal of Science and Research (IJSR), vol. 6, no. 4, pp. 1754-1756, 2017.

[31] T. Kim and H. Y. Kim, "Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data," PloS one, vol. 14, no. 2, p. e0212320, April 2019.

[32] S. Selvin, R. Vinayakumar, E. A. Gopalkrishnan, V. K. Menon and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in International Conference on Advances in Computing, Communications and Informatics, 2017.

[33] Loke.K.S. "Impact Of Financial Ratios And Technical Analysis On Stock Price Prediction Using Random Forests", IEEE, 2017.

[34] Xi Zhang1, Siyu Qu1, Jieyun Huang1, Binxing Fang1, Philip Yu2, "Stock Market Prediction via Multi-Source Multiple Instance Learning." IEEE 2018.

[35] Tao Xing, Yuan Sun, Qian Wang, Guo Yu. "The Analysis and Prediction of Stock Price", 2013 IEEE International Conference on Granular Computing (GrC), ISBN: 978-1-4799-1282-7, DOI: 10.1109/GrC.2013.6740438.