A Project Report On

Quantitative Finance Model Visualizer Using High-Frequency Trading Algorithms

Submitted in partial fulfillment of the requirement for the 8^{th} semester

Bachelor of Engineering

in

Computer Science and Engineering

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560111

Submitted By

Soumya Kumari 1DS19CS161

Xitiz Verma 1DS19CS193

Yashaswini shree 1DS19CS196

Mohammed shagil Khan 1DS20CS415

Under the guidance of

Prof. Sunanda

Assist. Professor , CSE , DSCE

Annina Simon

Co-guide - Data Engineer at ANZ Bank

2022 - 2023

Department of Computer Science and Engineering

DAYANANDA SAGAR COLLEGE OF ENGINEERING

Bangalore - 560111

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Dayananda Sagar College of Engineering

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade

Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560111

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **Quantitative Finance Model Visualizer Using High-Frequency Trading Algorithms** is a bonafide work carried out by **Xitiz Verma [1DS19CS193]**, **Soumya Kumari [1DS19CS161], Yashaswini Shree [1DS19CS196]** and **Mohammed Shagil Khan [1DS20CS415]** in partial fulfillment of 8th semester, Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during the year 2022-23.

Prof. Sunanda	**Dr. Ramesh Babu D R**	**B G Prasad**
(Guide)	Vice Principal & HOD	Principal
Asst Prof. CSE, DSCE	CSE, DSCE	DSCE
Signature:.....	Signature:.....	Signature:.....

Name of the Examiners:	Signature with date:
1.....	.....
2.....	.....

Acknowledgement

We are pleased to have successfully completed the project **Quantitative Finance Model Visualizer Using High-Frequency Trading Algorithms**. We thoroughly enjoyed the process of working on this project and gained a lot of knowledge doing so.

We would like to take this opportunity to express our gratitude to **B G Prasad**, Principal of DSCE, for permitting us to utilize all the necessary facilities of the institution.

We also thank our respected Vice Principal, HOD of Computer Science & Engineering, DSCE, Bangalore, **Dr. Ramesh Babu D R**, for his support and encouragement throughout the process.

We are immensely grateful to our respected and learned guide, **Prof. Sunanda**, Assistant Professor CSE, DSCE and our co-guide **Annina Simon** for their valuable help and guidance. We are indebted to them for their invaluable guidance throughout the process and their useful inputs at all stages of the process.

We also thank all the faculty and support staff of Department of Computer Science, DSCE. Without their support over the years, this work would not have been possible.

Lastly, we would like to express our deep appreciation towards our classmates and our family for providing us with constant moral support and encouragement. They have stood by us in the most difficult of times.

Soumya Kumari 1DS19CS161

Xitiz Verma 1DS19CS193

Yashaswini shree 1DS19CS196

Mohammed shagil Khan 1DS20CS415

Abstract

Quantitative Finance Model Visualizer plays a vital role in predicting stock prices with high accuracy. The project is about quantitative finance which refers to the methodology of applying disciplines from mathematics, statistics, and finance with computer programming to create a financial tool for future stock prediction. Stock market prediction is a very tough job as very high precision is needed. So many models have been used earlier but none of them are reliable and consistent. Thus, we propose a solution using deep learning, as it uses current output in further steps and understands the relationship between them.RNN has a vanishing gradient problem which happens due to the use of the same parameters repeatedly. This problem is avoided by the use of different parameters at each step. We balance the situation by adding variable-length sequences, generating variable-length sequences, and keeping the learnable parameters count constant. We are introducing gated cells like LSTM. Also, RNN can't retain the data for a long duration. Hence, we are planning to use the LSTM model which is a kind of RNN that has long-term memory. Lstm works on time series data and time series data can be influenced by any tiny noise. Time series data helps us to track differences over time. For this project, we are using highfrequency data which means everyday data fluctuation is included. Also, we are using the random forest for noise reduction and outlier detection. Random forest is normalizing our output and provides accuracy.

Table Of Contents

List of Figures

Chapter 1

Introduction

Shares of businesses are traded on the stock market for stockbrokers. One of the most difficult tasks is projecting stock prices since accurate forecasting is essential to success in the stock market. Due to the stock market's volatility, a number of strategies are employed to predict price, but none of them has been demonstrated to be a reliable tool for consistent prediction. As a result, we suggested the Artificial Neural Network (ANN) technique since, after learning from and analyzing the initial inputs and their relationships, ANN can generalize and forecast data. In our Quant Major Project, we set out to develop models on basis of technical indicators and Long Short Term Memory to forecast the price of a stock that varies daily as well as financial statistics that can be seen visually, like Skewness, Kurtosis, Holistic Volatility, etc., that are all compiled from day-end data for any publicly traded stock. The regression Long-Short Term Memory ANN model is trained after we first obtain a dataset of intraday trading of any publicly traded stock of the Indian Stock Exchange Market from Kaggle. A special type of artificial neural network called Long Short-Term Memory (LSTM) is employed in deep learning and artificial intelligence. One of the largest innovations to occur in the last 15 years is Algorithmic trading's High-Frequency Trading (HFT) division. The ability of a trader to take orders with extremely little lead time is referred to as HFT or nano trading. This model is run using price history together with technical analysis signals and strategies, and the results are assessed using profitability and performance measures. Based on historical stock data, the neural network model is successfully used to predict the daily highest, lowest, and closing values of business stocks in a short amount of time, however, it is ineffective in predicting the return rate of the stocks.

Various features such as stochastic indicators, moving averages, and RSI is extracted from the historical stock data to train the ANN model. The dataset is then divided into training and testing sets which are used for the accuracy of the ANN model.

1.1 Real World Application

Thus we are combining both Data mining and neural networks in our proposed system to first collect and refine stock data then analyze these data with the ANN method and provide the result of the input data in prediction using data mining and LSTM algorithm to predict the stock value more accurately.

1.2 Organisation of Project Report

The project report is organized as follows: In Chapter (2) we discuss the problem statement and the proposed solution. We also take a look at the systems that exist today and the drawbacks they face. Chapter (3) takes a more in-depth look at various hardware and software based solutions that exist, with a survey on existing literature available. Chapter (4) looks at the architecture of the proposed solution with an overview of the system design, utilizing system block diagrams and data flow diagrams. Chapter (5) dives into the Implementation of the solution, by describing the hardware and software requirements, along with dataset descriptions and implementation details. Chapter (6) describes our testing process, while Chapter (7) looks at our experimentation process and the obtained results. Chapter (8) summarizes our findings and concludes the paper.

Chapter 2

Problem Statement and Proposed Solution

2.1 Problem Statement

To develop a quantative tool for stock market prediction using deep learning algorithms and statistical parameters.

2.2 Existing Systems

There are several existing methods used for stock market prediction. Here are some commonly used approaches:

1. Fundamental Analysis: This method involves analyzing the financial health, performance, and prospects of a company to determine its intrinsic value. Factors such as revenue, earnings, industry trends, and management quality are considered to predict the future stock price.

2. Technical Analysis: Technical analysts study historical price and volume data to identify patterns, trends, and signals that can indicate future price movements. They use various tools and techniques, such as moving averages, chart patterns, and oscillators, to make predictions based on past market behavior.

3. Machine Learning and Artificial Intelligence: With the availability of vast amounts of data and computing power, machine learning and artificial intelligence algorithms are being increasingly used for stock market prediction. These models analyze historical data, market trends, news sentiment, and other relevant factors to make predictions. Common machine

learning algorithms used include regression models, decision trees, random forests, and neural networks.

4. Sentiment Analysis: Sentiment analysis involves analyzing news articles, social media feeds, and other textual data to gauge market sentiment and investor emotions. By monitoring positive or negative sentiment towards a particular stock or the overall market, predictions can be made about future price movements.

5. Expert Opinions: Financial analysts and experts provide their opinions and predictions based on their knowledge, experience, and research. These opinions can be valuable inputs for investors but should be considered alongside other methods for a comprehensive analysis.

2.3 Proposed Solution

The main objectives are:

- To Predict Result with more accuracy
- Solves the vanishing gradient issues in other RNN models
- To make model more efficient
- Consider other parameters too while making predictions like statistical indicators.

2.3.1 KNN

K-Nearest Neighbors (KNN) is a simple and popular supervised machine learning algorithm used for both classification and regression tasks. It is a non-parametric algorithm, meaning it doesn't make any assumptions about the underlying data distribution.

In KNN, the training dataset consists of feature vectors and their corresponding class labels (for classification) or target values (for regression). During the prediction phase, the algorithm calculates the distance between a test sample and all the training samples. The

K nearest neighbors are the data points in the training set that are closest to the test sample based on some distance metric, typically Euclidean distance.

For classification, KNN assigns the class label that is most frequent among its K nearest neighbors. The value of K is typically chosen beforehand by the user. In case of a tie, the algorithm may use different techniques like majority voting or distance weighting to determine the final class label.

For regression, KNN predicts the target value for the test sample by taking the average (or weighted average) of the target values of its K nearest neighbors shown in Figure 2.1.

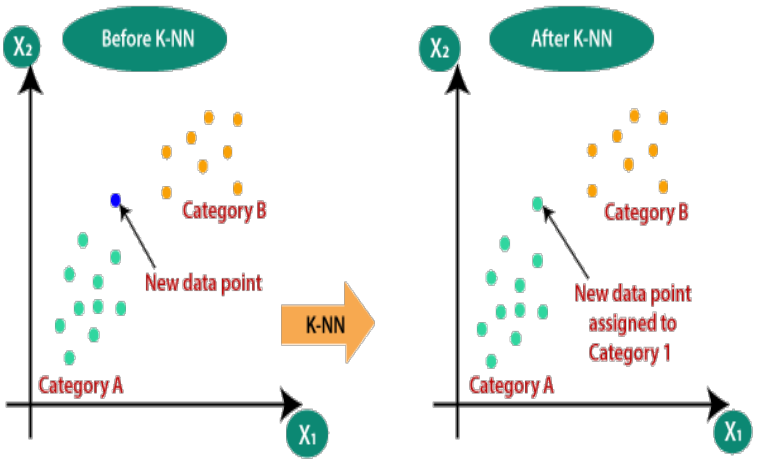

Figure 2.1: KNN Model Diagram

Some key considerations when using KNN:

1. Choice of K: The value of K affects the model's performance. A smaller K value can lead to more flexible and possibly noisy predictions, while a larger K value may smooth out the decision boundary and potentially miss local patterns.

2. Distance Metric: The choice of distance metric, such as Euclidean, Manhattan, or Minkowski, can influence the algorithm's performance. The metric should be selected based on the nature of the data and the problem at hand.

3. Feature Scaling: It is often recommended to scale the features before applying KNN to ensure that no single feature dominates the distance calculations.

4. Computational Complexity: As KNN calculates distances to all training samples during prediction, it can be computationally expensive for large datasets. Efficient data

structures like KD-trees or ball trees are used to speed up the nearest neighbor search process.

KNN is a straightforward algorithm and can be easily implemented. However, it may not perform well in high-dimensional spaces or when there is a large imbalance in the class distribution. It is important to preprocess the data, choose appropriate K and distance metrics, and consider the limitations of the algorithm when applying KNN to real-world problems.

2.3.2 SVR

Regression analysis uses a sort of machine learning method called Support Vector Regression (SVR). Finding a function that minimises the prediction error and roughly approximates the relationship between the input variables and a continuous target variable is the aim of SVR. SVR looks for a hyperplane that best matches the data points in a continuous space, unlike Support Vector Machines (SVMs) utilised for classification tasks. In order to do this, the input variables are mapped to a high-dimensional feature space, and the hyperplane that maximises the distance (margin) between it and the nearest data points while minimising the prediction error is found.

By applying a kernel function to translate the data to a higher-dimensional space, SVR is able to manage relationships that are not linear between the input variables and the target variable. As a result, it is an effective tool for regression problems where the relationships between the input variables and the target variable may be complex. Similar to SVM, Support Vector Regression (SVR) tackles regression issues.

Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems.

Here's a brief overview of how SVR works:

Data Preparation: The training data consists of input feature vectors and their corresponding target values. The data may need to be preprocessed and scaled to ensure each feature has a similar range.

Kernel Selection: A kernel function is selected, which determines the transformation of the input features into a higher-dimensional space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.

Model Training: SVR aims to find a hyperplane in the transformed feature space that best fits the training data. The hyperplane is defined by support vectors, which are a subset of training samples that lie closest to the hyperplane. The training process involves solving an optimization problem to minimize the error between the predicted values and the actual target values, while also maximizing the margin.

Model Prediction: After the SVR model is trained, it can be used to make predictions on new, unseen data points. The model uses the support vectors and their associated coefficients to predict the target values.

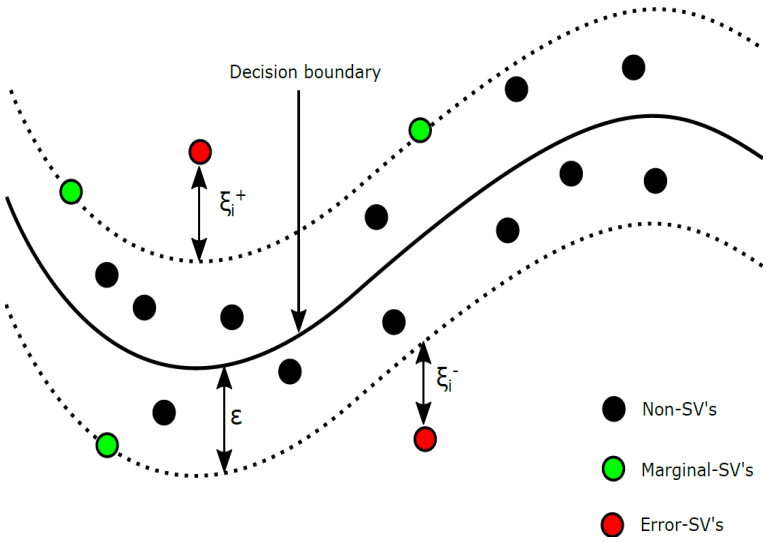

Figure 2.2: SVR Model Diagram

SVR has several advantages:

It can handle non-linear regression tasks by using different kernel functions.

It is effective in handling datasets with a high number of features.

In Figure 2.2 it is shown that it can handle outliers by using a margin of tolerance. However, there are also some considerations when using SVR:

Selection of the appropriate kernel function and tuning of hyperparameters can be crucial for achieving good performance. The algorithm's computational complexity can be high, particularly when dealing with large datasets. Interpretability of the model may be

limited compared to simpler linear regression models. Overall, SVR is a powerful algorithm for regression tasks, particularly when dealing with non-linear relationships between features and targets. It is widely used in various domains, including finance, economics, and engineering.

As the SVM for regression issues, Support Vector Regression can be viewed as its opponent.

SVR offers a strong prediction model while acknowledging the existence of non-linearity in the data.

2.3.2.1 LSTM

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that is widely used for sequence modeling and time series analysis. Lstm works on time series data and has retention capability which resolves the vanishing gradient problem in previous Models.

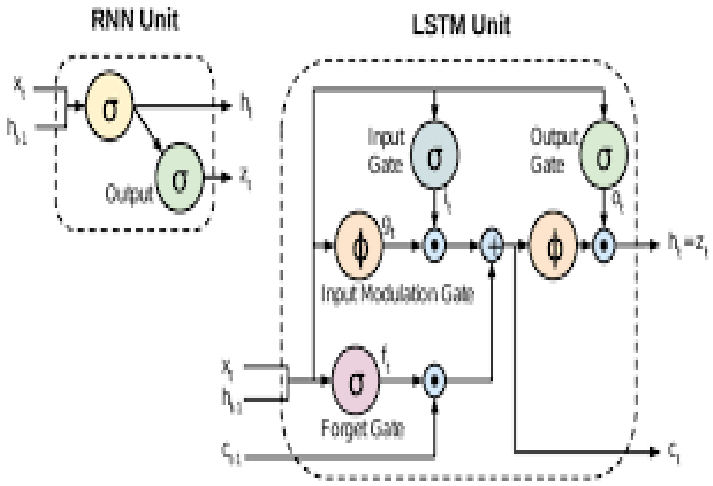

Figure 2.3: LSTM vs RNN Model

LSTM models are designed to address the vanishing gradient problem faced by traditional RNNs. They are capable of capturing long-term dependencies and handling se-

quences with time lags or temporal dependencies. The key component of an LSTM model is the LSTM cell shown in Figure 2.3, which consists of various interconnected gates that control the flow of information. The main gates in an LSTM cell are the input gate, forget gate, and output gate. These gates enable the model to selectively remember or forget information over time. LSTM models utilize memory cells to store and update information. The memory cells maintain a hidden state and a cell state, which allow the model to retain information over long sequences. The cell state can be modified through the gating mechanisms, and the hidden state is used to make predictions or propagate information to subsequent time steps. LSTM models are trained using backpropagation through time (BPTT), which is an extension of backpropagation for RNNs. The weights of the LSTM cells are updated using gradient descent optimization algorithms, such as stochastic gradient descent (SGD) or Adam, to minimize the prediction error between the model's output and the ground truth. LSTM models have proven effective in various applications, including natural language processing (NLP), speech recognition, sentiment analysis, machine translation, time series forecasting, and financial market analysis. They excel in tasks involving sequential or temporal data, where capturing long-term dependencies is crucial.

LSTM models are specifically designed to overcome the vanishing gradient problem, which is a common issue in traditional RNNs. The vanishing gradient problem occurs when gradients diminish exponentially over time, making it difficult for the network to capture long-term dependencies. LSTMs use specialized memory cells and gating mechanisms to selectively retain or forget information over longer sequences, allowing them to capture and propagate important information over time.

2.3.3 LSTM with GRU

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) are both types of recurrent neural network (RNN) architectures that have been widely used for sequential data processing tasks, such as natural language processing, speech recognition, and time series analysis. Both models aim to address the vanishing gradient problem that can occur in tra-

ditional RNNs, which hinders their ability to capture long-term dependencies in sequences.

LSTM is a more complex architecture compared to the basic RNN. It introduces three gates: the input gate, forget gate, and output gate. These gates control the flow of information through the LSTM unit, allowing it to selectively retain or discard information from the past and update the current state. The input gate determines how much of the new input to add to the memory cell, the forget gate decides which parts of the previous state to forget, and the output gate regulates the amount of information to output from the cell.

GRU, on the other hand, is a simplified variant of the LSTM architecture that combines the forget and input gates into a single ”update gate” and merges the cell state and hidden state into a single state vector. The update gate determines the extent to which the previous state influences the current state and how much of the new input should be incorporated.

In terms of performance, LSTM and GRU have shown similar capabilities in capturing long-term dependencies in sequential data. LSTM is known for its effectiveness in modeling complex sequences and has been widely adopted in various applications. GRU, being a more compact model with fewer gates, may require less computational resources and training time. It has been observed that GRU can perform comparably to LSTM on certain tasks while having a simpler architecture.

Both LSTM and GRU have their strengths and weaknesses, and the choice between them often depends on the specific problem at hand, the available computational resources, and the trade-off between model complexity and performance. Experimentation and evaluation on the target task are usually necessary to determine which model performs better for a given application.

2.3.4 Random Forest

Random Forest is a popular machine learning algorithm that belongs to the ensemble learning family. It is primarily used for classification and regression tasks and is effective in handling complex datasets. The algorithm combines multiple decision trees and generates predictions by aggregating the results from each tree.

Here's a brief overview of how Random Forest works:

Dataset: Random Forest operates on a labeled dataset consisting of input features and corresponding target variables.

Random Subset: From the original dataset, a random subset of the data is selected. This random sampling is performed with replacement, which means that each tree in the Random Forest can contain multiple instances of the same sample.

Decision Trees: A decision tree is built using the random subset of the data. At each node of the tree, a feature is selected based on a criterion such as Gini impurity or information gain. The tree splits the data based on these selected features, creating branches and leaf nodes that represent the predictions shown in Figure 2.4 .

Multiple Trees: The process of building decision trees is repeated multiple times to create an ensemble of trees. Each tree is trained on a different random subset of the data.

Voting: Once the ensemble of decision trees is created, predictions are made by aggregating the results from each individual tree. In classification tasks, the most popular class among the trees is chosen as the final prediction. In regression tasks, the predictions are averaged or aggregated using other techniques.

Regarding the kind of constructive classifiers, random forest is one of the ensemble learning techniques that falls under the homogeneous base learner group. Since all base learners are decision trees, as their name suggests, they are more straightforward to construct than comparable techniques.

The Decision Tree: The Foundation of Random Forest: The decision tree approach divides the position of features repeatedly into fictitious limb regions, each of which serves as a basis for a new approximation. It is made up of edges and nodes. While the branches indicate a spectrum of values, a node represents a specific attribute. These value ranges serve as dividing lines between the set of values for the specified attribute. The root node, which is often referred to as the first split point, is where the decision tree building process begins. The divisions of the entire dataset are determined by this split. Entropy and information gain calculations or the Gini index are used to determine the divide. Up until there is no longer a need for partitioning, the dividing process is carried out from top to bottom. The final partitions are represented by the leaves that are present at the end of decision tree. The decision tree's root node has no incoming edges, but the test nodes must have at least one

edge from the upper node, which may also have one or more outgoing edges. The choice is made by the leaves node, also referred to as the decision node, which holds the anticipated output.

The values of the characteristics of the provided data are used to group the data in the Decision Tree. The pre-classified data is used to create a Decision Tree. The features that divide the data into the most appropriate classes are chosen for classification. The division of the data items is done by the values of these feature values. Recursively, this technique is applied to every divided subset of the data items. As soon as every data item in the current subset belongs to the same class, the process is finished.

The Decision Tree algorithm shown in Figure 2.4 is used in many different domains, including statistical data comparison, text classification, and text extraction. In addition, it has some shortcomings, such as a propensity to over fit the training data, sensitivity to minute changes in the data, challenges handling continuous variables, a bias towards features with more levels, a lack of global optimisation, difficulties with missing data, and interpretability problems for large trees.

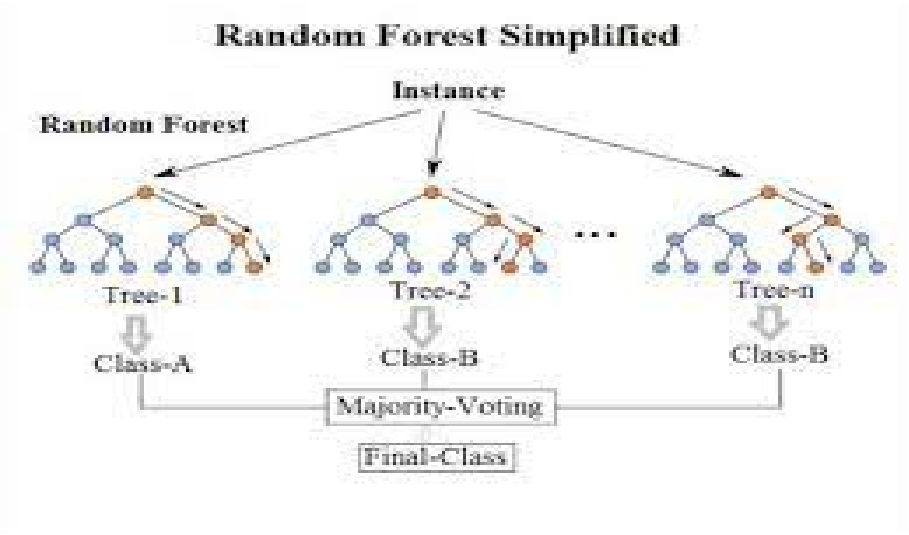

Figure 2.4: Random Forest Model Diagram

The key advantages of Random Forest are:

Robustness: Random Forest is less prone to overfitting compared to a single decision tree. The ensemble nature of Random Forest helps to reduce the impact of noisy data and outliers.

Feature Importance: Random Forest provides a measure of feature importance, indicating which features have the most significant influence on the predictions.

Handling Large Datasets: Random Forest can handle large datasets with high dimensionality and a mix of categorical and numerical features effectively.

Nonlinear Relationships: Random Forest can capture complex nonlinear relationships between features and target variables.

However, it is important to note that Random Forest also has some limitations, including:

Interpretability: The interpretability of Random Forest is lower compared to a single decision tree. It may be challenging to explain the underlying reasoning behind the model's predictions.

Training Time: Training a Random Forest can be computationally expensive, especially when dealing with a large number of trees and features.

Memory Usage: Random Forest can consume a significant amount of memory, especially when working with large datasets.

Overall, Random Forest is a versatile and powerful algorithm that is widely used in various domains due to its ability to handle complex data, reduce overfitting, and provide robust predictions.

2.4 System Requirements

•Processor : 4 Core, 8 Thread Processor

• Memory : Minimum of 16 GB ram for computation

• Video Memory : Minimum of 3 GB of VRAM for Object Detection and another 2 GB for feature extraction and the ANN.

Chapter 3

Literature survey

Bing Y et al.,[2] explores the application of artificial neural networks (ANNs) for stock market prediction. ANNs are a type of machine learning model inspired by the structure and function of the human brain. They consist of interconnected nodes (neurons) organized in layers and are capable of learning complex patterns from data.In the context of stock market prediction, ANNs can be trained on historical stock price and market-related data to learn patterns and relationships that may help forecast future stock prices or market trends. The specific approach and architecture of the neural network used in the paper are not available without further details.

The paper discuss the methodology used to collect and preprocess the data, the architecture of the neural network model, the training process, and the evaluation metrics used to assess the performance of the model in predicting stock market behavior. It also present and discuss the results obtained from the experiments conducted.

Gurjar M et al.,[3] has argued that A stock market is a marketplace where stockbrokers can trade firm shares. Stock value

Prediction is one of the most difficult problems to solve since a stock market prediction requires a high degree of precision. There are numerous ways to forecast stock market prices, but none are 100 percent accurate.

Due to its volatile nature, none of those strategies has been demonstrated to be a consistently

reliable forecast tool. Because ANN can generalise and predict data after learning from and analysing the initial inputs and their relationships, we suggested the ANN technique in this research. To forecast stock prices, we employed a feed forward network and a backward propagation method. In this article, we discussed a technique that uses the ANN back propagation algorithm to predict future stock price values for a given day based on some input.

Hiransha M. Gopalakrishnan EA et al.,[4] have applied the neural network, one of the intelligent data mining techniques, in a variety of fields. In the modern economy, stock market data analysis and prediction play a significant role. Forecasting methods can be divided into linear (AR, MA, ARIMA, ARMA) and non-linear (ARCH, GARCH, and neural network) models. In this study, we employ four deep learning architecture types to predict a company's stock price based on historical data: Multilayer Perceptron (MLP), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) shown in Figure 3.1, and Convolutional Neural Network (CNN). In this case, we are using the closing prices from two distinct stock exchanges (NYSE).
The network was trained using the stock price of a single NSE-listed business, and it made predictions for five different NSE-listed and NYSE-listed companies. CNN is reportedly outperforming the other models, according to observations. Despite having been trained on NSE data, the network was able to predict for the NYSE. This was made possible since the internal dynamics of both stock markets are similar. The findings were compared to the ARIMA model, and it was found that the neural networks outperformed the linear model at the time (ARIMA).

Utomo D. et al.,[5] focuses on predicting stock prices using a backpropagation neural network (BPNN) with specific enhancements. Backpropagation is a commonly used algorithm for training neural networks. It involves calculating the gradients of the network's weights using the chain rule and adjusting the weights iteratively to minimize the prediction

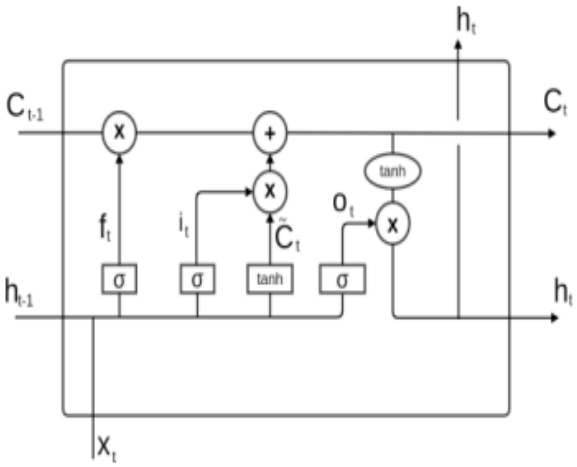

Figure 3.1: LSTM Cell

error.

The paper introduces additional techniques to enhance the performance of the BPNN model. Two such techniques mentioned in the title are gradient descent with momentum and adaptive learning rate. Gradient descent with momentum involves incorporating a fraction of the previous weight update to accelerate convergence and overcome local optima. Adaptive learning rate refers to dynamically adjusting the learning rate during the training process to improve convergence speed and accuracy.

The author describes the methodology shown in Figure 3.2 used to collect and preprocess the stock price data, as well as the architecture and parameters of the BPNN model. They might also discuss the training process, including the specific implementation details of gradient descent with momentum and adaptive learning rate.

Chandar SK et al.,[6] focuses on predicting stock market prices using a combination of wavelet transform and artificial neural network (ANN). Wavelet transform is a mathematical technique used for analyzing and processing signals. It decomposes a signal into different frequency components and provides a time-frequency representation of the data.

In the context of stock market prediction, wavelet transform can be used to extract meaningful features from the stock price data by decomposing it into different scales or frequencies.

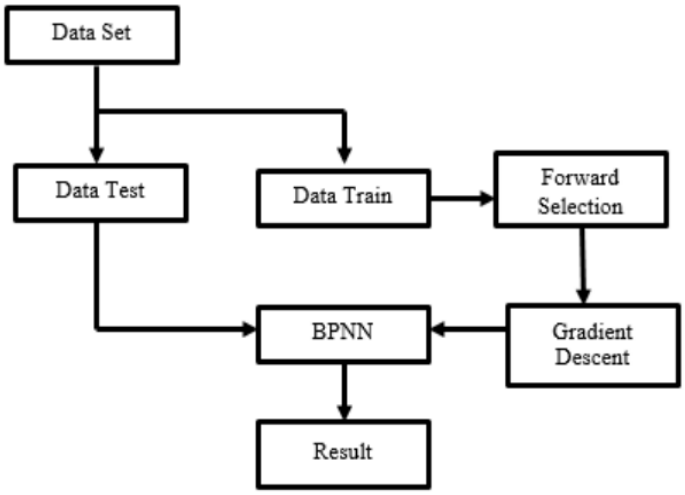

Figure 3.2: Proposed Method

These features can capture both short-term and long-term patterns in the data.

The paper describe the methodology used to preprocess the stock price data using wavelet transform and extract relevant features. The authors also discuss the architecture and parameters of the artificial neural network model used in the study.

Artificial neural networks are machine learning models inspired by the structure and function of the human brain. They consist of interconnected nodes (neurons) organized in layers and are capable of learning complex patterns from data and the two-level decomposition of a signal is shown in Figure 3.3.

The authors describe the training process of the ANN, including the optimization algorithm used (such as backpropagation), the number of layers and neurons, and the activation functions employed. They may also discuss any additional techniques or modifications used to enhance the performance of the hybrid model.

Borovkova S et al.,[7] suggested an ensemble of long-short-term memory (LSTM) neural networks that use a wide range of technical analysis indicators as network inputs. The suggested ensemble functions online, weighting the individual models in accordance with their most recent performance, which enables us to creatively address potential non-stationarities. Area under the receiver operating characteristic curve is a measurement of the models' performance. On a number of US large-cap stocks, we assess the prediction ability of our model and compare it to lasso and ridge logistic classifiers. It is discovered

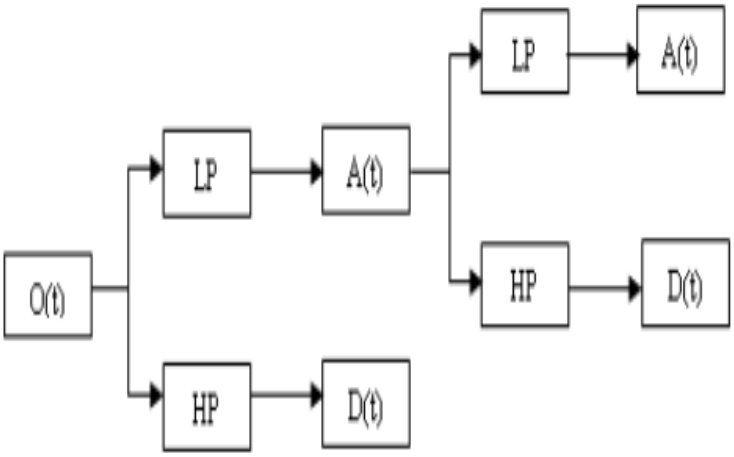

Figure 3.3: A two-level decomposition of a signal o(t)

that the suggested model outperforms benchmark models and equally weighted ensembles. The paper likely focuses on the application of Long Short-Term Memory (LSTM) neural networks for high-frequency stock market classification. LSTM is a type of recurrent neural network (RNN) that is well-suited for processing sequential data and capturing long-term dependencies. In the context of stock market classification, the paper may explore the use of LSTM networks to analyze high-frequency stock market data. High-frequency data refers to data that is collected at a very short time interval, such as tick data or data with a frequency of seconds or milliseconds. The authors describe the methodology used to collect and preprocess the high-frequency stock market data. They may discuss the architecture and parameters of the ensemble of LSTM neural networks, which refers to the combination of multiple LSTM models to improve classification accuracy. Ensemble methods typically involve training multiple models and combining their predictions to make a final decision. The paper may discuss how the ensemble is formed, such as through averaging or voting, and how the individual LSTM models are trained and weighted shown in Figure 3.4. Furthermore, the authors present experimental results, evaluating the performance of the ensemble of LSTM networks for stock market classification. They may compare the performance of the ensemble approach with individual LSTM models or other classification techniques. Evaluation metrics such as accuracy, precision, recall, or F1 score could be used to assess the effectiveness of the proposed ensemble approach.

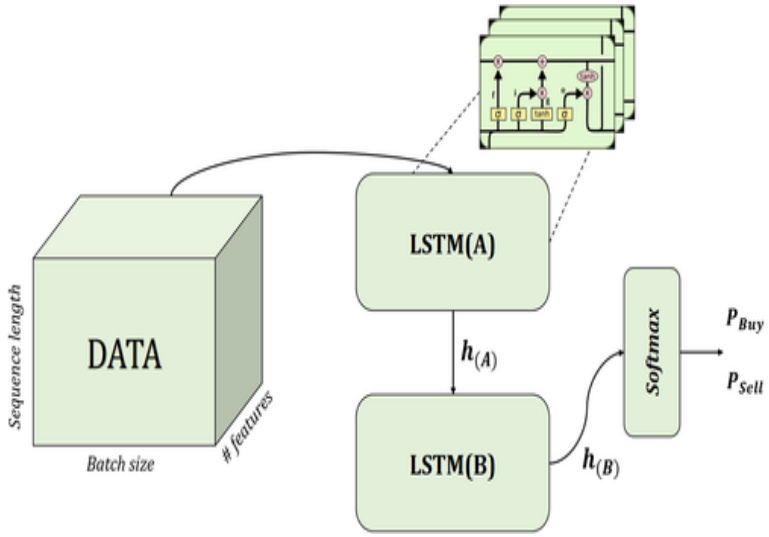

Figure 3.4: Stacked-LSTM with a softmax layer

Araujo RDA et al.,[8] address the issue of forecasting financial time series, various models have been put forth. The random walk dilemma (RWD), which arises from all of these models even with sophisticated approaches, is a conundrum. To solve this problem for financial time series with a daily frequency, the idea of time phase adjustment was put out. However, when trading platforms developed, operations on the stock market were now performed at fractions of a second intervals, necessitating the examination of high-frequency financial data series. This paper introduces a model for high-frequency stock market forecasting known as the increasing decreasing linear neuron (IDLN). For the suggested model architecture, a descending gradient-based technique with automatic time phase modification is also provided. Additionally, a collection of high-frequency financial time series from the Brazilian stock market were used in an experimental investigation, and the outcomes outperformed those of forecasting models that had been previously established in the literature. The paper likely presents a hybrid model designed for forecasting high-frequency stock market data. High-frequency data refers to data collected at a very short time interval, such as tick data or data with a frequency of seconds or milliseconds.

The author describe the methodology used to collect and preprocess the high-frequency stock market data. They may discuss the different techniques or models integrated into the

hybrid model and how they are combined to improve forecasting accuracy.
The specific components and techniques used in the hybrid model are not known without further details from the paper. However, hybrid models typically combine various forecasting methods such as statistical models, machine learning algorithms, or time series analysis techniques.
The paper provide insights into the architecture and parameters of the hybrid model, including the approach used for training and optimization. The authors may discuss the selection and weighting of individual models or techniques within the hybrid framework.

Deepak RS et al.,[9] highlighted the most important aspect of stock market forecasting is a high level of precision and accuracy. Most people employ technical, fundamental, or time series analysis in the projections of the stock brokers. However, these approaches cannot be relied upon completely, necessitating the development of a backup strategy for stock market forecasting. In this research, we present a Machine Learning (ML) approach that will be trained using stock data that is currently available, gain intelligence, and then apply the learned information for precise prediction. Artificial Neural Networks (ANN) were discovered to be the most useful consideration after extensive research on numerous algorithms and their suitability for diverse problem areas. Wide-ranging features and cross validation sets can be implemented using neural network models because they have features and adjustable parameters. The Bombay Stock Exchange (BSE) index data set was employed as the major essential strategy in this paper for testing the concept of machine learning.
The method chosen to be used is supervised classifier and machine learning to capture the most accurate output. Results are evaluated using a separate set of feature lists and an SVM classifier for binary classification as shown in Figure 3.5. Although there is an ideal methodology for some problems, the majority of the Machine Learning approach to tackling business problems has an advantage over statistical methods that do not involve AI. A comparison of certain applications was made, and the results showed that ANNs are widely used in stock price forecasting because they produce more predictable results with a higher degree of control.

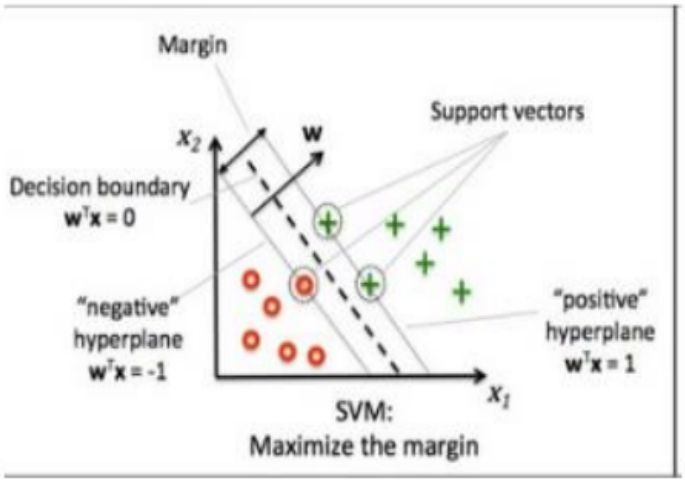

Figure 3.5: The Support Vector Machine Decision Boundary

Murat A. Basaran et al.,[10] Forecasted techniques using fuzzy time series are growing in popularity recently. The two subclasses of these techniques are univariate and multivariate procedures as shown in Figure 3.6. Real time series data can be impacted by a variety of events, as is well recognised. Using a multivariate fuzzy time series forecasting model in this situation may be more logical in order to provide more precise forecasts. The most widely used technique for obtaining fuzzy forecasts when a multivariate fuzzy time series approach is used is employing tables of fuzzy relations. However, using this strategy requires careful computational planning. In this paper, a novel method has been presented for identifying fuzzy links without employing fuzzy logic relation tables.

Hossain MZ et al.,[11] proposes an intelligent stock market forecasting system using the ability of neural network and fuzzy inference system to discover patterns in nonlinear and chaotic systems.The goal of this study, which uses a sample of the entire population as its subject, is to predict stock price using financial data from Bangladeshi company BEXIMCO Ltd., a member of the Chittagong Stock Exchange. A nonlinear fuzzy-neural network model used the stock market as an external variable to make the forecast. Terms

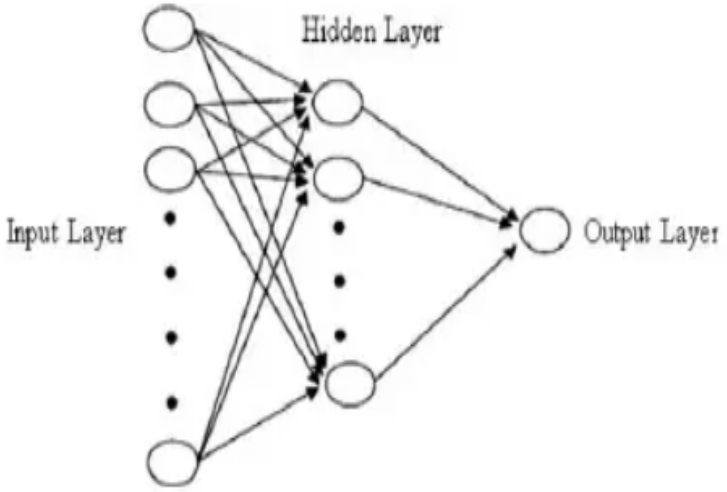

Figure 3.6: Multilayer feed forward ANN with one output neuron

used in general artificial intelligence, artificial neural networks, machine learning, back-propagation algorithms, fuzzy inference systems, and stock markets. The paper focuses on predicting stock market prices using a hybrid model of artificial intelligence (AI). A hybrid model combines multiple AI techniques or models to leverage their respective strengths and enhance prediction accuracy. The authors discuss the methodology used to collect and pre-process the stock market data. They might describe the specific AI techniques incorporated into the hybrid model, which could include machine learning algorithms, neural networks, or other AI methods. The specific AI techniques used in the paper are not known without further details. However, machine learning algorithms such as regression models, decision trees, support vector machines (SVM), or ensemble methods are commonly employed for stock market prediction. The authors also discuss the architecture and parameters of the hybrid model, as well as the approach used to train and optimize it. It has been described how the individual AI techniques are combined and how their predictions are aggregated or weighted within the hybrid framework. The goal of this study, which uses a sample of the entire population as its subject, is to predict stock price using financial data from Bangladeshi company BEXIMCO Ltd., a member of the Chittagong Stock Exchange. A nonlinear fuzzy-neural network model used the stock market as an external variable to make the forecast as shown in Figure 3.7. Terms used in general artificial intelligence, artificial neural networks, machine learning, backpropagation algorithms, fuzzy inference systems,

and stock markets.

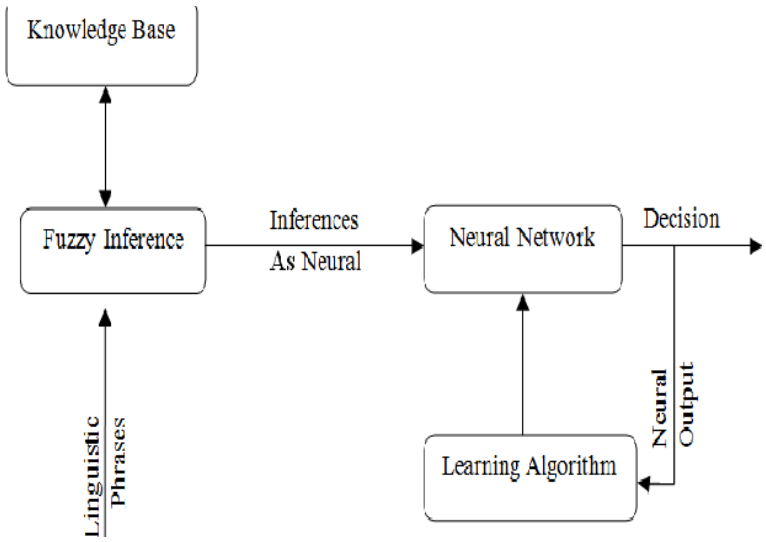

Figure 3.7: The basic structure of the Fuzzy-neural system

J. H. Wang et al.,[13] created a prediction method that can be used to predict the mid-term price trend in the Taiwan stock market (also known as the Taiwan Stock Exchange Weighted Stock Index, or TSEWSI). A recurrent neural network that was trained using features taken from ARIMA studies forms the system's foundation. The TSEWSI series can be recognised as a nonlinear variation of ARIMA(1,2,1) by comparing the raw data and then evaluating the autocorrelation and partial autocorrelation function charts. It has been demonstrated that neural networks trained using second difference data provide more accurate predictions than neural networks taught using raw data. We feedback the difference between two subsequent predictions during backpropagation training in addition to the conventional error modification term in order to modify the connection weights. According to empirical findings, networks created utilising 4-year weekly data are capable of accurately predicting market trends for up to six weeks. According to the findings of our experiments, the ARIMA-based recurrent neural network can anticipate the market trend with respectable accuracy. Additionally, we have established that the TSE:WSI series can be stationary following a second difference operation and labelled it as ARIMA(1,2,1). We think that by including other feature data, such as trading volume, interest rates, etc., the

prediction accuracy can be increased. Future work can use filtering [11 to remove noise or outliers as well as exploring time series identification because stock price series is a high noise dynamic system.

T. W. Chiang et al.,[15] talks about the problem of time-series forecasting, a novel complex neurofuzzy autoregressive integrated moving average (ARIMA) computation approach is proposed. To create the proposed computer model, which is known as the CNFS-ARIMA, the proposed approach merges a complex neurofuzzy system (CNFS) employing complex fuzzy sets (CFSs) and ARIMA models. The real and imaginary sections of the complex-valued output of CNFS-ARIMA can be used for two alternative functional mappings. The so-called dual-output property is this. The development of CNFS-ARIMA did not involve any ambiguous If-Then rules. Structure learning and parameter learning are used to help the CNFS-ARIMA self-organize and self-tune during the process of formation. Fuzzy If-Then rules' premise sections are described by a class of CFSs, while their consequent parts are described by ARIMA models. The membership degrees of CFS, an advanced fuzzy set, are complex-valued within the complex plane's unit disc. CNFS-ARIMA models have good nonlinear mapping capabilities for time-series forecasting thanks to the synergistic benefits of CNFS and ARIMA. The proposed approach is put to the test using many benchmark time series, and the outcomes are contrasted with those of competing approaches. Additionally, the proposed method for carrying out the dual-output forecasting experiments uses real-world financial time series, such as the Dow Jones Industrial Average Index (DJI), the Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX), and the National Association of Securities Dealers Automated Quotation (NASDAQ). The experimental results reveal that the suggested strategy performs very well.

C. Narendra Babu et al.,[14] talked about highly volatile, stock market time series data (TSD) present a significant research challenge for the time series community. The majority of prediction issues focus on one-step forecasting and use linear conventional models like the auto regressive integrated moving average (ARIMA) or generalised auto regressive

conditional heteroscedastic (GARCH). However, when N rises, two challenges appear if any prediction model is used for multi-step or N-step ahead prediction. The first is a decline in forecast accuracy, while the second is a failure to retain the data trend or dynamics over the whole prediction horizon. This study develops a linear hybrid model with ARIMA and GRACH shown in Figure 3.8 that preserves data trend and provides decent prediction accuracy. As a result, a simple moving average (MA) filter is used to partition the given TSD into two distinct series. One of them is accurately modelled with ARIMA, and the other with GARCH. The final model predictions are then created by merging the forecasts from the two models. In order to assess the precision of the suggested model, data from the Indian stock market is taken into account. The proposed model surpasses the others for multi-step forward prediction in terms of both prediction accuracy and retaining data trend, according to a comparison of this model's performance with traditional methods.

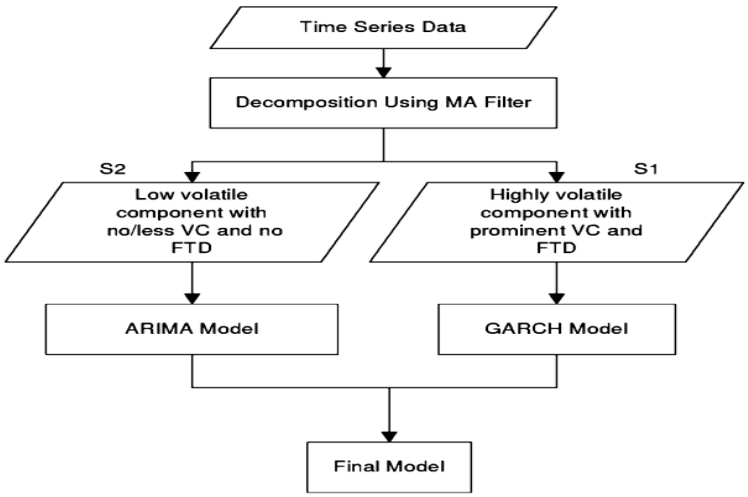

Figure 3.8: The Proposed hybrid ARIMA-GARCH Model

Ritika Singh et al.,[16] argued that stock market viewed as chaotic, intricate, volatile, and dynamic. Its prediction is undoubtedly one of the most difficult challenges in time series forecasting. Additionally, current Artificial Neural Network (ANN) techniques don't yield positive outcomes. Meanwhile, improvements in machine learning have produced positive outcomes for language processing, image classification, and speech recognition.

Since both stock data and digital signal processing involve time series, the same techniques can be used. The learning from this paper can also be used to analyse voice time series data. In this study, deep learning for stock prediction is introduced, and its effectiveness is assessed using Google stock price multimedia data (chart) from the NASDAQ. The purpose of this research is to show how deep learning may increase the precision of stock market forecasts. In order to do this, the (2D)2PCA + Deep Neural Network (DNN) method is contrasted with the most recent 2-Directional method. Principal component analysis in two dimensions (2D) and radial basis function neural network (RBFNN). With an enhanced accuracy of 4.8 percent for Hit Rate and a window size of 20, it is discovered that the suggested method outperforms the current method RBFNN. When the proposed model's outcomes are contrasted with those of the Recurrent Neural Network (RNN), it is discovered that the accuracy for Hit Rate has been increased by 15.6 percent.In comparison to RBFNN and 43.4 percent better than RNN, DNN has a correlation coefficient between the actual and anticipated return that is 17.1 percent higher.

Vaibhav Kumar and ML Garg,[17] argued that predicting stock prices is a very risky and lucrative strategy. There are numerous techniques and resources available for this. In this study, we introduce a hybrid deep neural network for stock price forecasting as shown in Figure 3.9. Based on a set of parameters, this model will forecast the closing price of any stock for the day. This hybrid model will combine a deep neural network and a fuzzy inference method. Although fuzzy inference systems can handle uncertain information, one of their limitations is that they cannot learn from example sets. Although a deep neural network with multiple hidden layers is capable of learning from experience, it is unable to handle uncertain information. If these two models are combined, the resulting model will be able to learn from examples and deal with ambiguous input. Through the use of this model, several predictions will be made in this article, and it will be put to the test on several stocks. In our research, we created a hybrid deep learning model that combines fuzzy inference with deep neural networks. In this model, we used both fuzzy logic and deep learning. This model was used to forecast a stock's day-ending price. This model performed remarkably when we used it on 7 stocks. Around 97.5 percent of the time, it accurately predicts a stock's day-ending price. Today, the application of this concept is fairly broad. In our

study, this model was used to predict the stock market. However, there are many jobs today where using predictive analytics to project future values is necessary. This idea is utilised by numerous financial institutions while making financial investments. Every forecast made inside the group or business is predicated on a set of input values. Therefore, this model will be quite helpful for those businesses involved in predictive analytics. This model can be used by businesses who want to forecast the sale value of their products depending on various variables, in addition to businesses that make financial investments. Additionally, there is still room to enhance this model's functionality and precision by including more features. Therefore, it can be claimed that this model has a very broad opportunity and open scope.

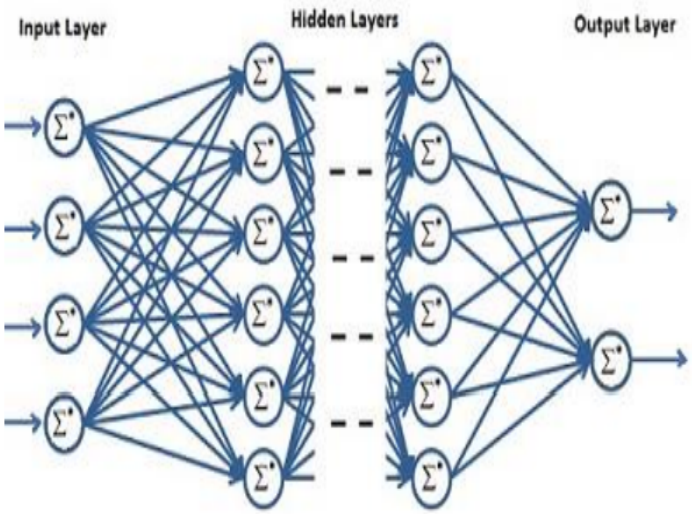

Figure 3.9: Hybrid Deep Neural Network (HDDN)

D. M. Q. Nelson et al.,[18] argued that the stock market is a very complicated, chaotic, and dynamic environment, making price predictions extremely difficult. Numerous studies from numerous fields have attempted to tackle this problem, and many of them have centred on machine learning techniques. There are numerous instances where machine learning algorithms have produced acceptable outcomes when making that kind of prediction. This article examines how LSTM networks can be used in that situation to forecast stock price patterns based on price history and technical analysis indicators. To do this, a prediction

model was created, and a number of experiments were run. The outcomes of these tests were then analysed using a number of standard metrics to determine whether this particular sort of algorithm offers any advantages over other Machine Learning techniques and investment strategies. The results are encouraging, with an average accuracy of 55.9 percent when predicting whether the price of a certain stock will group or not in the near future. We can see that, with very few exceptions, the model suggested in this article performs better than the baselines. The results are quite encouraging because they have shown to be more accurate predictors than other methods currently used in the literature. Despite the very broad input dimension, the algorithm has shown that it is capable of learning from it without the need of any dimension reduction techniques, such as feature selection, for instance. It has shown significant improvements in accuracy when compared to previous machine learning models, but on the other hand, we think the variance may be decreased, which would make the model more trustworthy.Although it didn't necessarily have the best performance when compared to the baselines, it's crucial to note that it was able to keep the financial results favourable for all stocks. Another benefit is that it had a high return ratio per operation, which indicates that it was more successful at identifying high variances, which is crucial when accounting for transaction costs and taxes. Additionally, it is possible to draw the conclusion that the LSTM-based model has fewer risks in comparison to the other techniques by looking at the maximum losses.

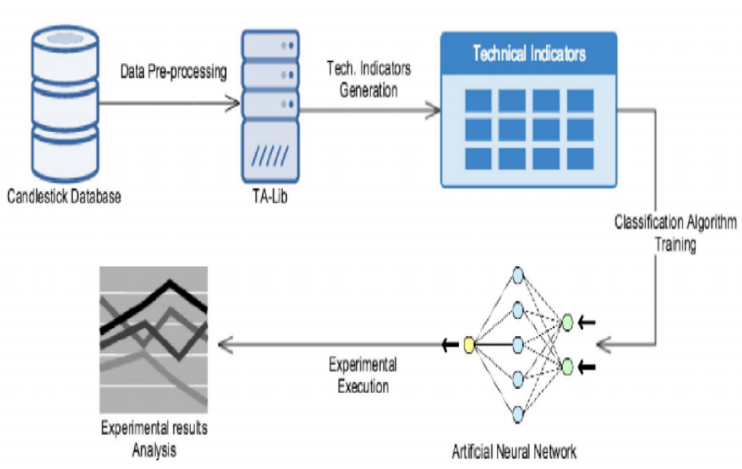

Figure 3.10: Methodology

S. Hochreiter and J. Schmidhuber,[19] argued that recurrent backpropagation requires a relatively long time to learn to store information over long time periods, primarily due to insufficient, fading error backflow. We quickly summarise Hochreiter's (1991) analysis of this issue before introducing the long short-term memory (LSTM), a brand-new, effective gradient-based approach. By enforcing constant error flow through constant error carousels within certain units, LSTM can learn to bridge minimal time gaps greater than 1000 discrete-time steps by truncating the gradient where doing so does not impair it. Multiplexing gate units acquire the ability to open and close the constant error flow.The computational complexity of the LSTM is O and it is local in space and time. We use local, distributed, real-valued, and noisy pattern representations in our studies with synthetic data. LSTM produces far more successful runs and learns considerably more quickly when compared to real-time recurrent learning, back propagation through time, recurrent cascade correlation, Elman nets, and neural sequence chunking. Additionally, LSTM resolves difficult, artificial long-time lag tasks that no previous recurrent network algorithms have ever been able to. Recurrent networks have the potential to retain representations of recent input events as activations via their feedback connections (short-term memory as opposed to long-term memory represented by slowly changing weights). For a variety of applications, such as speech processing, non-Markovian control, and music composition, this could be important (Mozer, 1992). However, the most popular algorithms for learning what to store in short term memory are either extremely time-consuming or completely ineffective, particularly when there are significant minimal time lags between inputs and corresponding teacher signals. Although theoretically intriguing, current techniques do not clearly outperform, for example, backpropagation in feedforward networks with constrained time windows as shown in Figure 3.11. This article reviews an issue analysis and offers a solution.

L. C. Cheng et al.,[20] argued that the stock market's turbulent, unpredictable, and dynamic atmosphere makes stock prediction difficult. Many research use deep learning

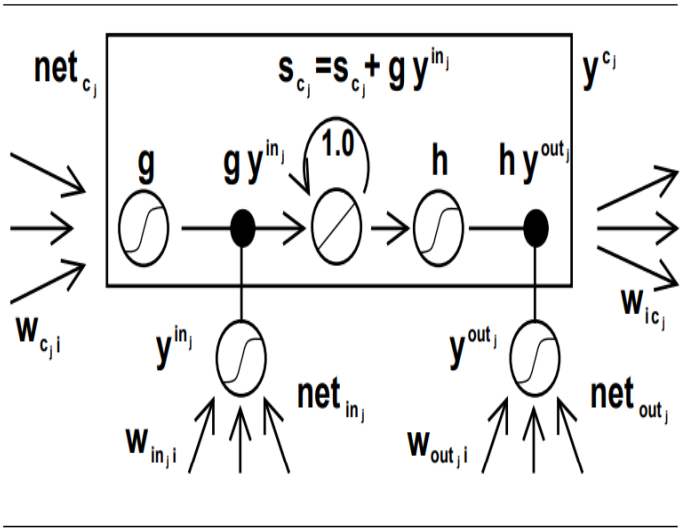

Figure 3.11: Architecture of memory cell cj (the box) and its gate units inj, outj.

models to forecast changes in stock prices. Although the attention mechanism has lately gained favour in neural machine translation, attention-based deep learning models for stock prediction have received little attention. An attention-based long short-term memory model is put out in this paper to forecast stock price movement and develop trading techniques. We employ deep learning to forecast stock price alterations in order to generate trading strategies. Using technical indicators and historical price data, we train models, then utilise the predicted outcome to choose a trading strategy as shown in Figure 3.12.

Figure 3.12: Proposed method

L. Zhang et al.,[21] argued that the formation of stock prices is based on short- and/or long-term commercial and trading activities that represent various trading frequency patterns. However, these patterns are frequently elusive because in the actual world, they

are influenced by a variety of hazy political-economic factors, including business performances, governmental policies, and even breaking news that spreads across markets. Additionally, because stock price time series are non-stationary and non-linear, forecasting future price movements is quite difficult. We suggest a unique State Frequency Memory (SFM) recurrent network to overcome these issues, capturing multi-frequency trading patterns from historical market data to generate long- and short-term forecasts over time. The SFM, which was inspired by the Discrete Fourier Transform (DFT), breaks down the hidden states of memory cells into various frequency components, each of which simulates a certain frequency of latent trading patterns that underlie stock price fluctuations. Then, using an Inverse Fourier Transform (IFT), future stock values are forecasted as a nonlinear mapping of the sum of these components. While a short-term prediction often relies on high-frequency trading patterns, a long-term prediction should place more emphasis on the low-frequency trading patterns aiming for long-term return. Modelling multi-frequency trading patterns can enable more accurate predictions for varied time ranges. Unfortunately, there is no model that clearly differentiates between different trading pattern frequencies to provide dynamic predictions in literature. The studies using actual market data also show that the SFM performs more competitively than state-of-the-art techniques. In order to anticipate the trend of stock prices, we develop a State Frequency Memory regression network in this paper that learns trading patterns with numerous frequencies. Due to the trading actions carried out at various speeds, the stock price actually reflects multi-frequency patterns. Finding pertinent frequency patterns offers helpful hints on the price trend going forward. To the best of our knowledge, no price prediction model approach now in use can discern between the various multi-frequency patterns that underlie price time-series. The SFM represents the trading patterns as various frequency components using the joint state-frequency memory states. The gating structure like the LSTM also reveals the long-term dependencies. Results from experiments using actual price data show that the SFM outperforms both the AR and traditional LSTM models in its ability to identify and control multi-frequency patterns in stock prices and their comparison is shown in Figure 3.13.

Y. Zhao et al.,[22] argued that the issue of time series forecasting is examined. A time

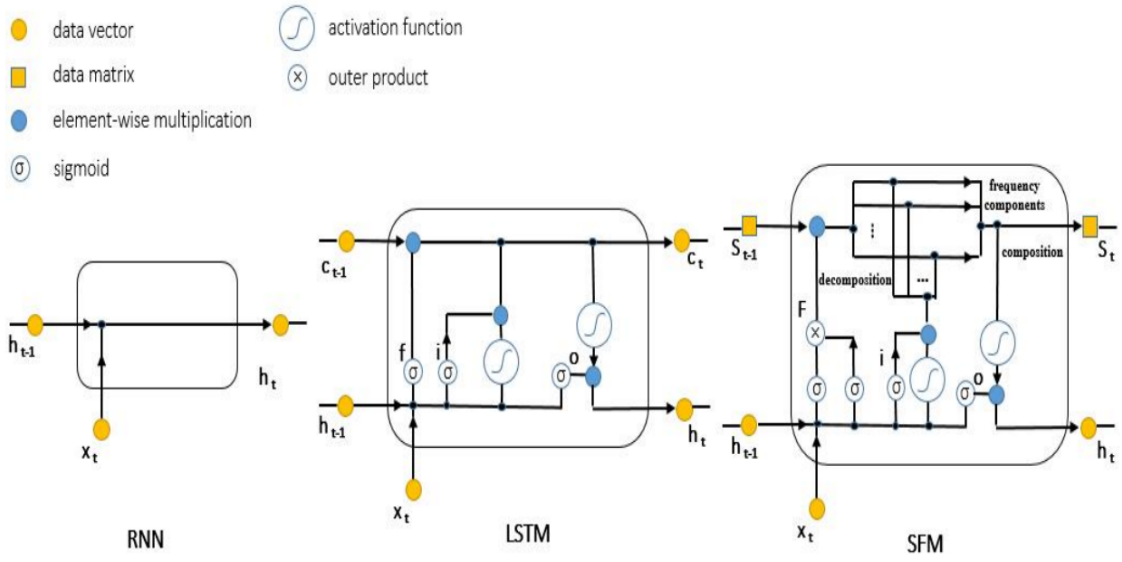

Figure 3.13: Comparison between the cell structures of the RNN (left), the LSTM (middle) and the SFM (right)

series is described as a collection of data points arranged chronologically. Many time series data from real-world events are driven by a number of latent components that happen at various frequency. These frequency-domain components are unable to be recognised and distinguished by time series forecasting systems currently in use. This paper suggests using the wavelet transform to explicitly reveal frequency-domain information from a univariate time series in order to increase forecasting accuracy. This approach is inspired by the recent development of signal processing and speech recognition techniques that decompose a time series signal into its time-frequency representation, a scalogram (or spectrogram). We use various neural networks, based on the altered data, to simultaneously capture local time-frequency properties and the overall long-term trend. We also leverage the attention mechanism as showin in Figure 3.14 to effectively combine local and global features. The experimental findings on actual time series demonstrate that our suggested strategy outperforms different baseline methods in terms of performance. In this study, we suggest applying a wavelet treatment to univariate time series in order to improve forecasting accuracy. The wavelet transform can explicitly reveal the latent components from complex time series at various frequencies. We create a brand-new attention-based neural network that uses CNN to extract regional time-frequency features and LSTM to simultaneously

detect long-term global trends. Our model includes an attention module that dynamically determines the relative weights of various local features based on how the time series trend is represented. We also use hidden layers to discover high-order feature interactions and gently combine local and global features. The experimental findings on two real-world datasets confirm the value of time-frequency data from wavelet transformed time series and the efficiency of our approach in terms of prediction accuracy.

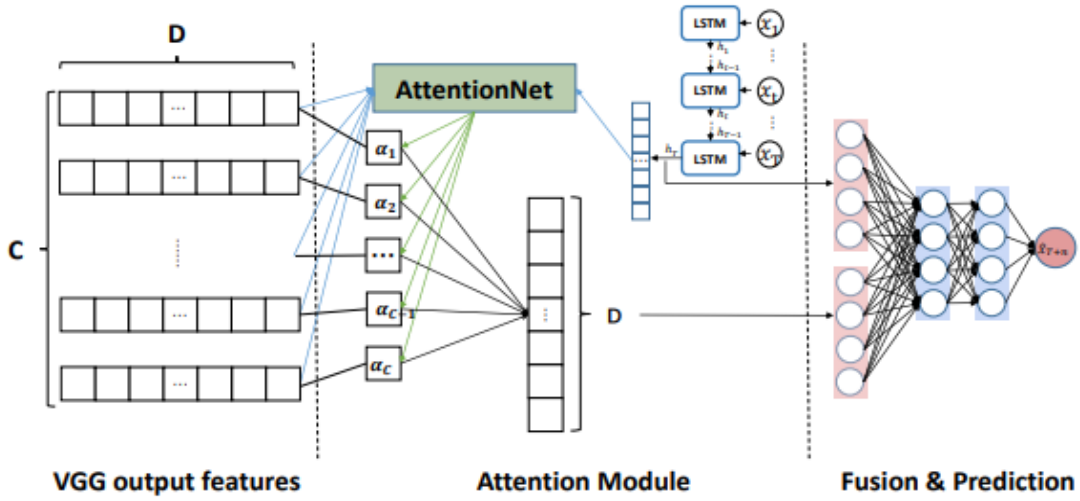

Figure 3.14: Attention, fusion and prediction

Y. Zhao et al.,[23] argued that in order to estimate a company's default probability, we suggest using a dynamic forecasting methodology called DMDP (dynamic multi-source default probability prediction). To evaluate the credit risk of companies that are listed on a stock market, the default probability is a crucial consideration. Our DMDP framework, which aims to support financial institutions in decision-making, not only examines financial data to capture a company's historical performance, but also makes use of a long short-term memory model to dynamically incorporate daily news from social media to take market participants' perceptions and public opinion into account. Two major contributions are made by the investigation of this work. First, we leverage unstructured news that has been

gleaned from social media to lessen the influence of the issue of financial fraud on the forecast of default probabilities. In order to combine structured financial parameters and unstructured social media data for default probability prediction, we also offer a neural network method. Numerous experimental findings show that, when compared to different baselines, the DMDP as shown in Figure 3.15 is effective at predicting the likelihood of default for mainland Chinese listed companies.

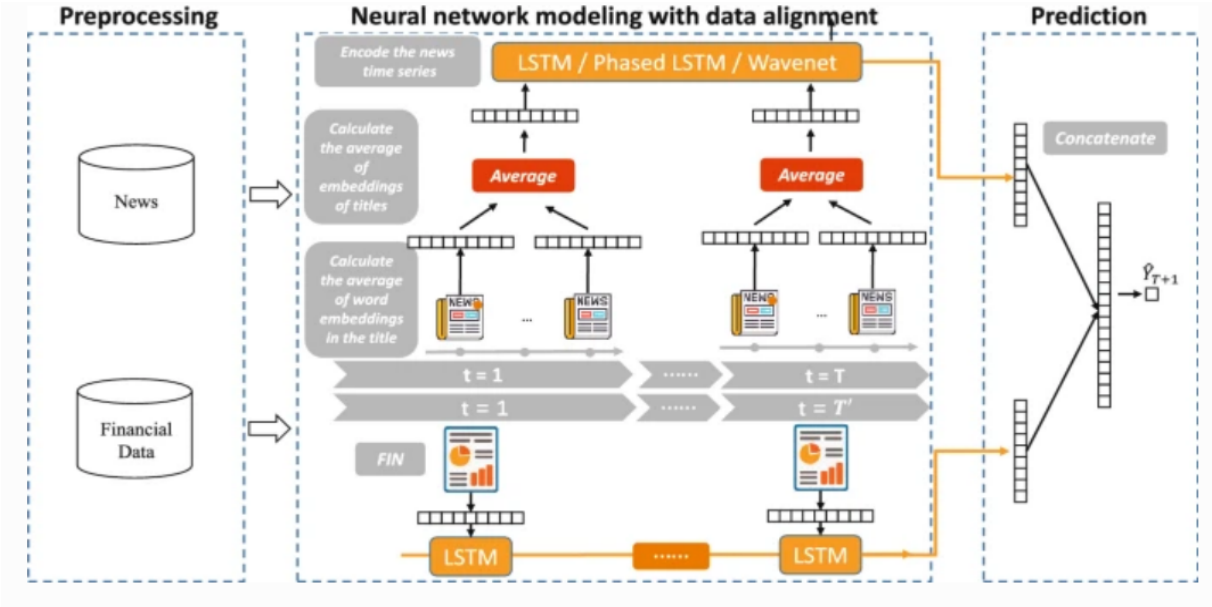

Figure 3.15: DMDP framework

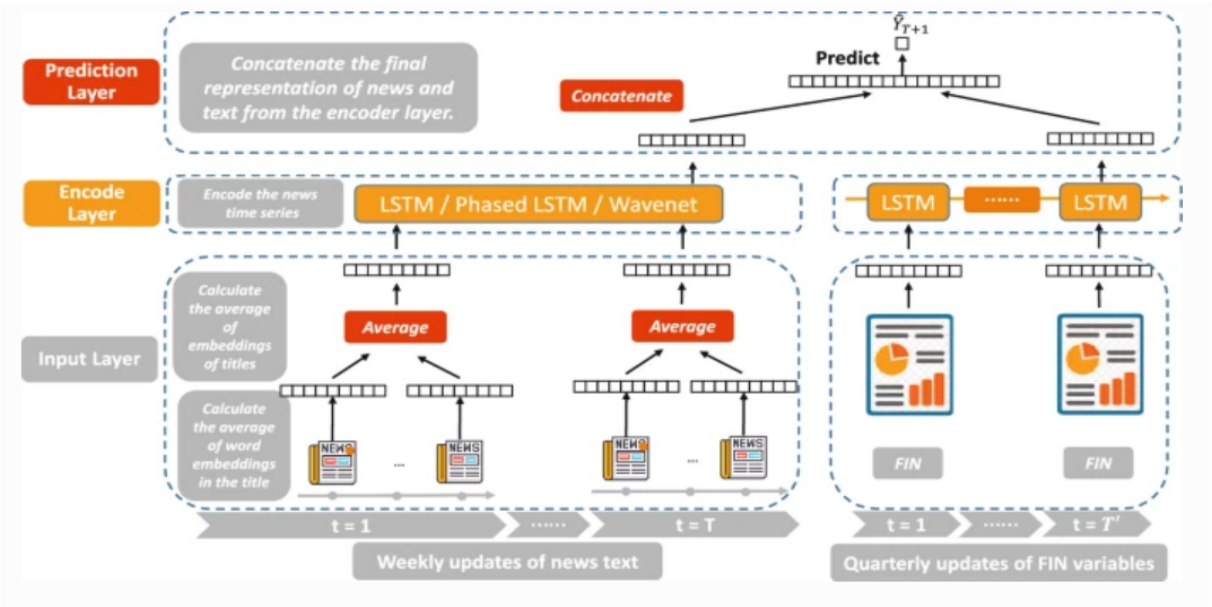

Figure 3.16: The architecture of the proposed neural prediction model

Korhan Gokmenoglu1 et al.,[24] argued that the objective of this study is to reevaluate the connection between exchange rate and stock market returns for a subset of emerging markets. The quantile-on-quantile approach is used to offer a comprehensive and in-depth picture of the relationship between the variables that are the subject of the inquiry. This method can show how the variables' relationships are heterogeneous and variable at different quantiles. The estimation result shows that, unless specific market conditions are met, exchange rate adjustments have little impact on the stock market performances of the analysed nations. According to the empirical findings, exchange rate flexibility is a key factor in deciding whether market returns are bearish or bullish.

After all the literature survey , we have proposed our solution on deep learning hybrid model which combines lstm and gru for etter acuuracy. We have implemented many models like KNN , SVR , LSTM , LSTM with GRU and random forest for comparison . And later we proceeded with hybrid model which is LSTM with GRU as it is giving very less error values and has improved our model accuracy.

Chapter 4

Architecture and System Design

4.1 Software Overview

It shows the modules involved in building the system i.e,

- Backend Server

- Web User Interface

4.1.1 System Block Diagram

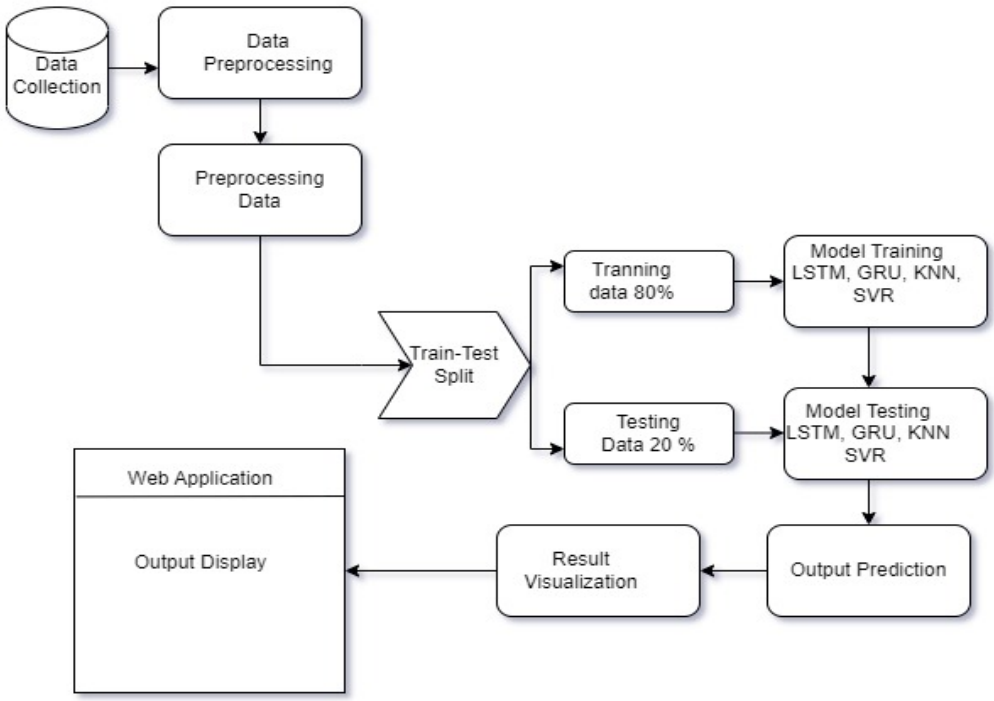

Figure 4.1: System Block Diagram

4.1.2 Data Flow Diagram

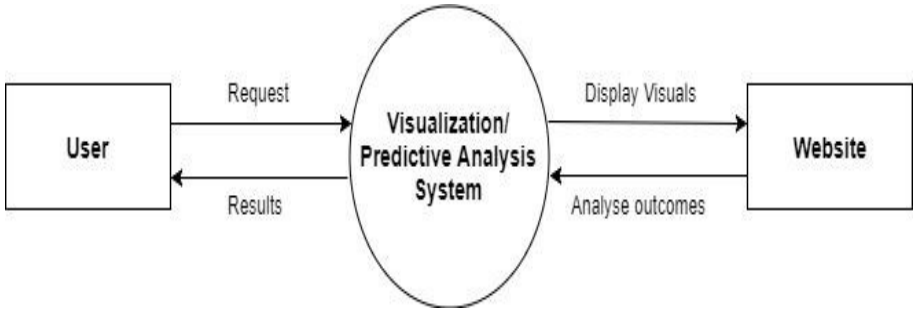

Figure 4.2: Level 0 Data Flow Diagram

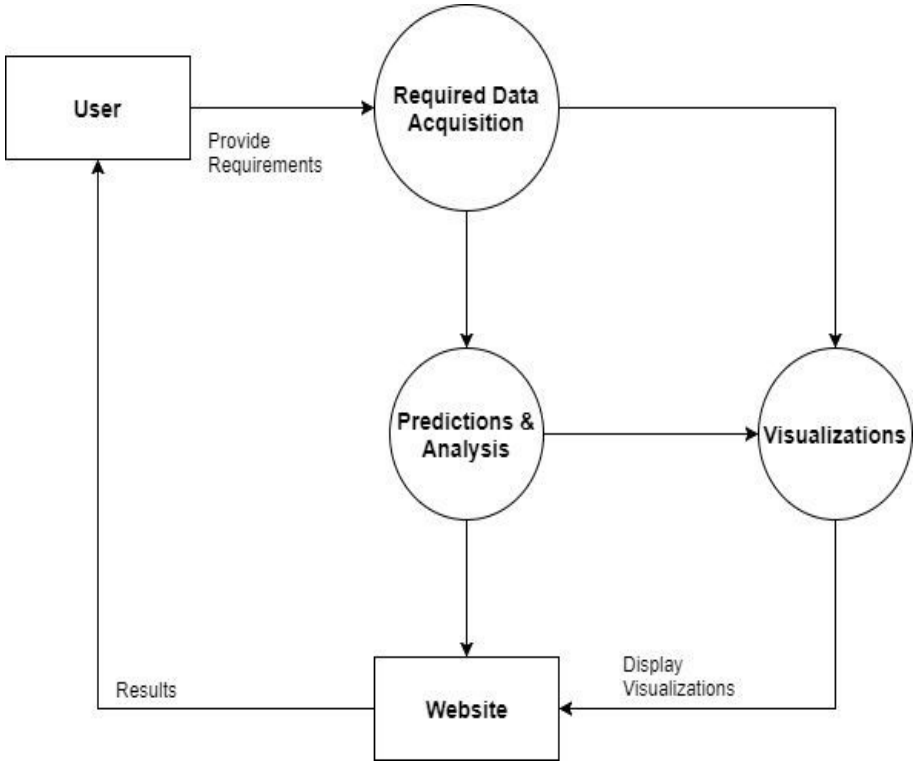

Figure 4.3: Level 1 Data Flow Diagram

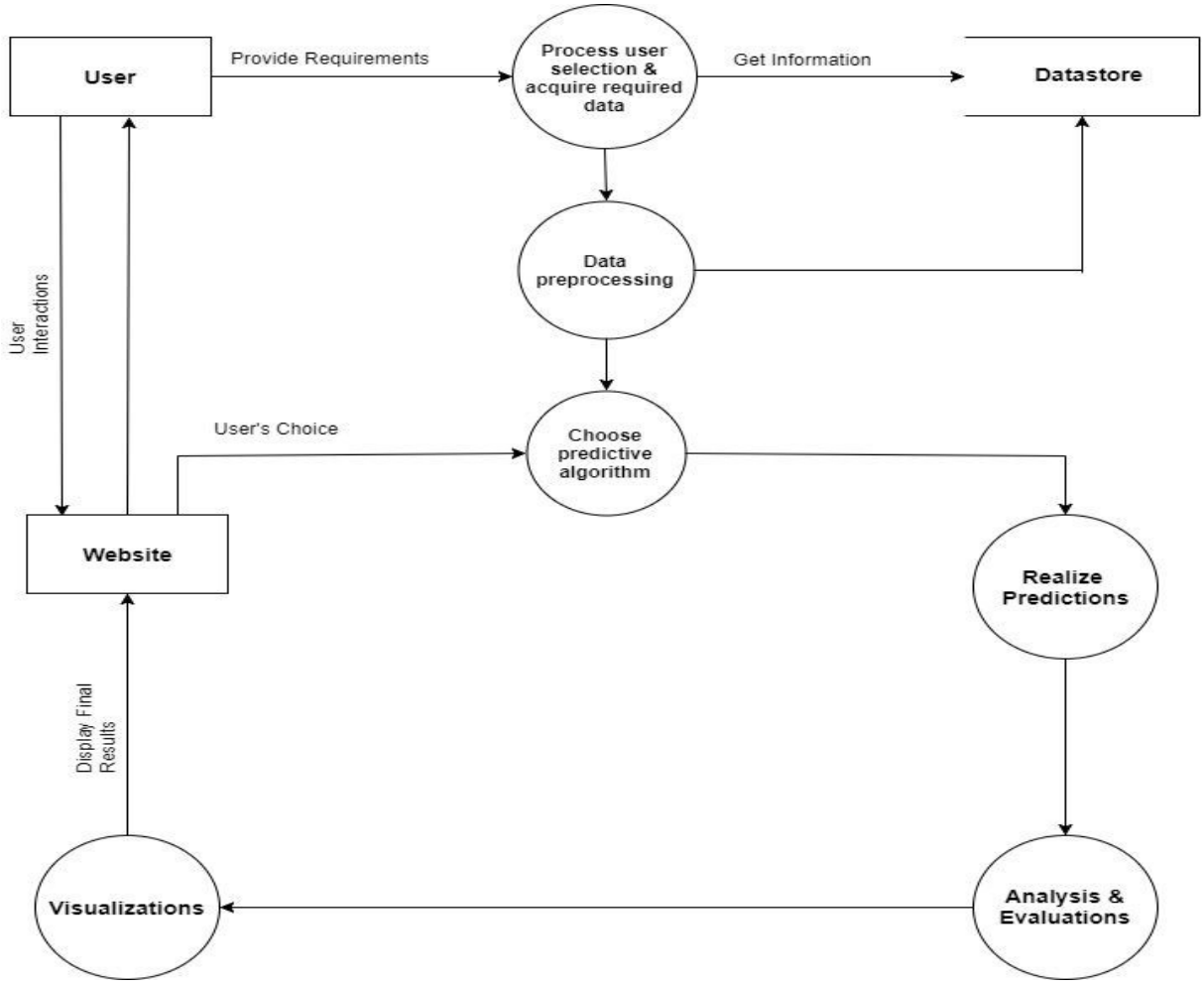

Figure 4.4: Level 2 Data Flow Diagram

Here , first we are getting data from dataset.Data preprocessing is performed to clean, transform, and normalize the data before feeding it into the model. This step may include tasks such as removing missing values, handling outliers, scaling features, encoding categorical variables, or splitting the data into training and validation sets. During model training, the prepared data is fed into the model in batches. The model processes each batch and updates its internal parameters based on the computed loss function and the chosen optimization algorithm. This step involves forward propagation, where the input data is passed through the layers of the model, and backward propagation, where the gradients are calculated and used to update the model's parameters. Once the model has been trained and evaluated satisfactorily, it can be deployed for inference or prediction on new, unseen data. After getting the predicted output user can make his decision and data get displayed

on a website as shown in Figure 4.4

4.1.3 Use Case Diagram

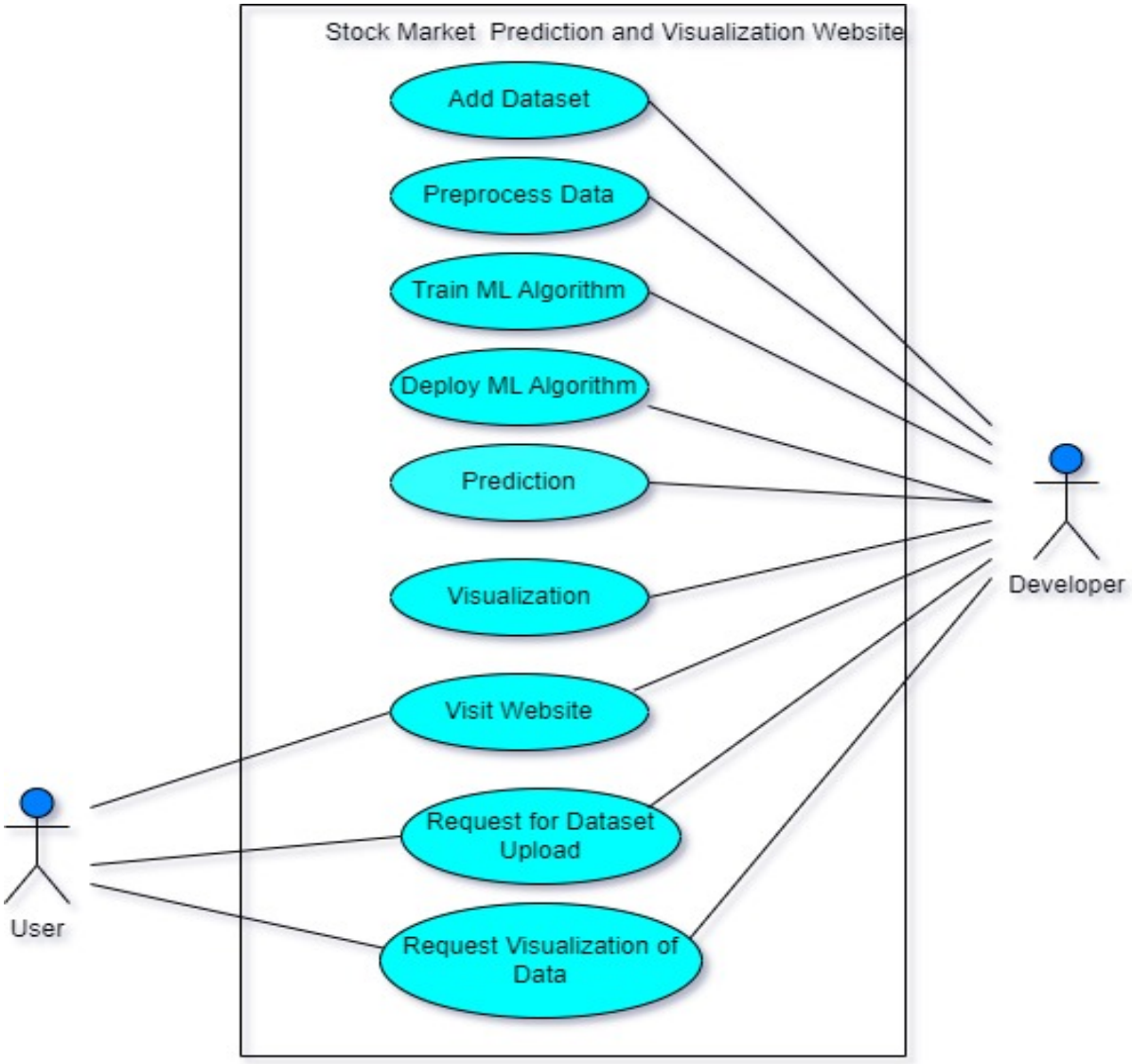

Figure 4.5: Use Case Diagram

The scope and high-level functions of a system are described in use-case diagrams. The interactions between the system and its actors are also depicted in these diagrams. Use-case diagrams explain what the system does and how the actors interact with it, but they do not show how the system functions inside, as seen in Figure 4.5

4.1.4 Class Diagram

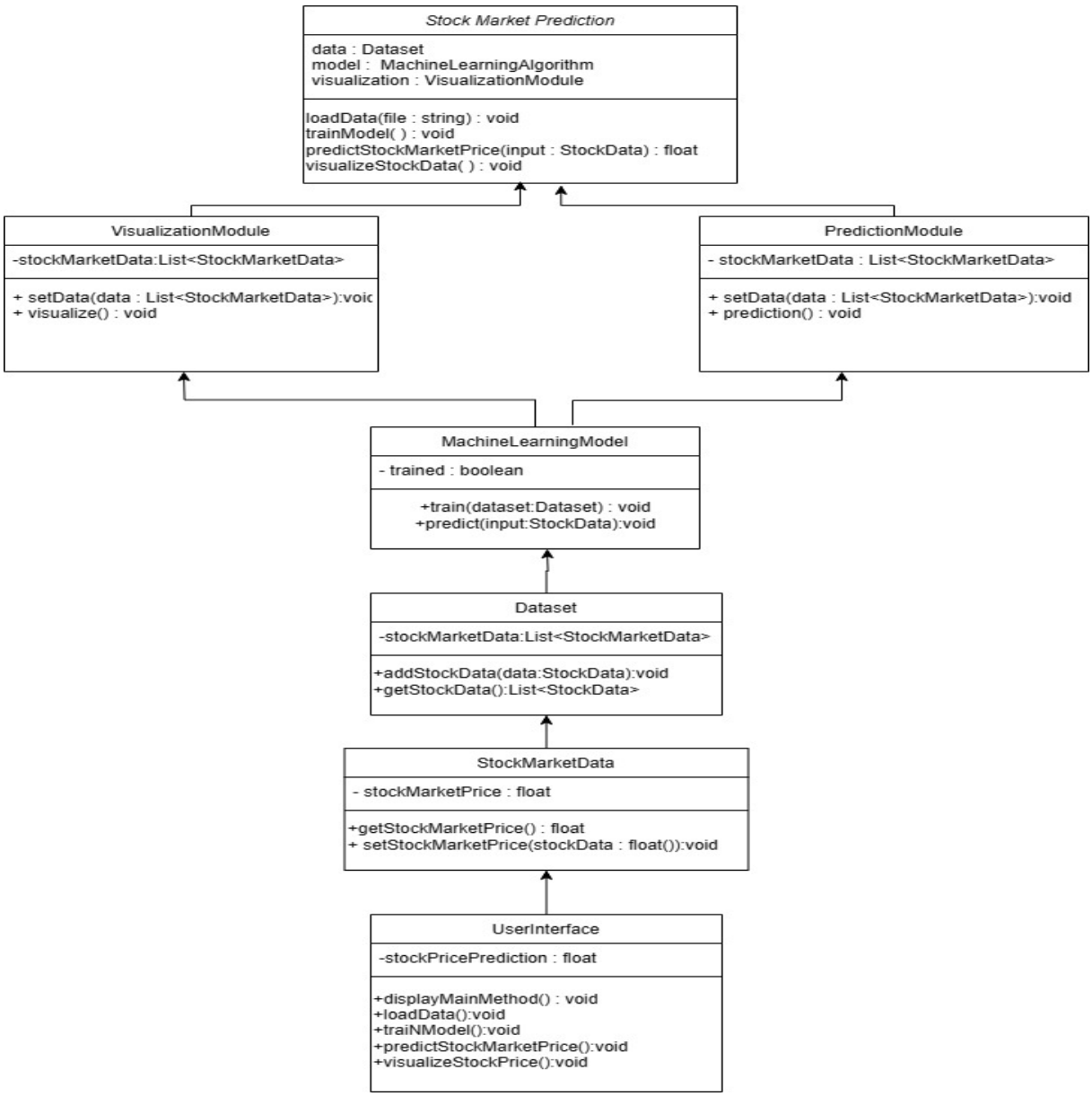

Figure 4.6: Class Diagram

The class diagrams are the system or subsystem's blueprints. Class diagrams can be used to represent the system's constituent parts, show how they are related to one another, and explain the functions and services each one performs. As seen in Figure 4.6, class diagrams are helpful at many stages of system design.

Chapter 5

Implementation

5.1 Implementation Platform

5.1.1 Hardware

i) Processor:Intel core i7

ii) RAM: 16GB

iii) GPU: Nvidia GeForce RTX 3070 8GB

iv) Storage platform: NVME SSD

5.1.2 Software

i) Operating System: Windows 11, 64bit

ii) Software Used : TensorFlow, OpenCV, Flask , Jupiter Notebook

iii) Programming Languages : Python 3, Javascript

iv) Server: Python HTTP server

5.2 Implementation Details

5.2.1 Organisation of files

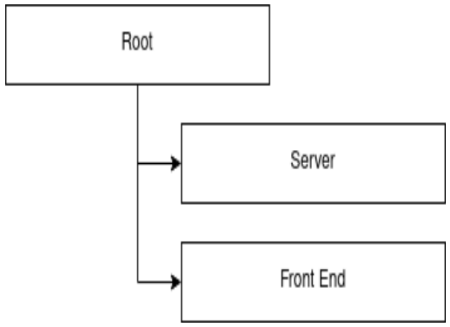

Figure 5.1: Root Directory Structure

5.2.2 Implementation Workflow

To build our LSTM-based Recurrent Neural Network (RNN) to predict any stock price step by step. It is split into 7 parts as below as shown in Figure 5.2

Problem statement

Clearly define the problem you want to solve, which is predicting stock prices using an LSTM-based RNN. Specify the time horizon you want to predict (e.g., next day, next week), as this will affect the data processing and model building steps.

Data processing

a. Collect Historical Data: Gather historical stock price data for the target stock. This can be obtained from various financial data sources or APIs. b. Data Preprocessing: Clean and preprocess the data. This may involve removing missing values, handling outliers, and normalizing the data to a consistent scale. c. Splitting Data: Divide the dataset into training and testing sets. Typically, a larger portion is allocated for training (e.g., 70-80 percent) to train the model, while the remaining portion is used for evaluation.

Model building

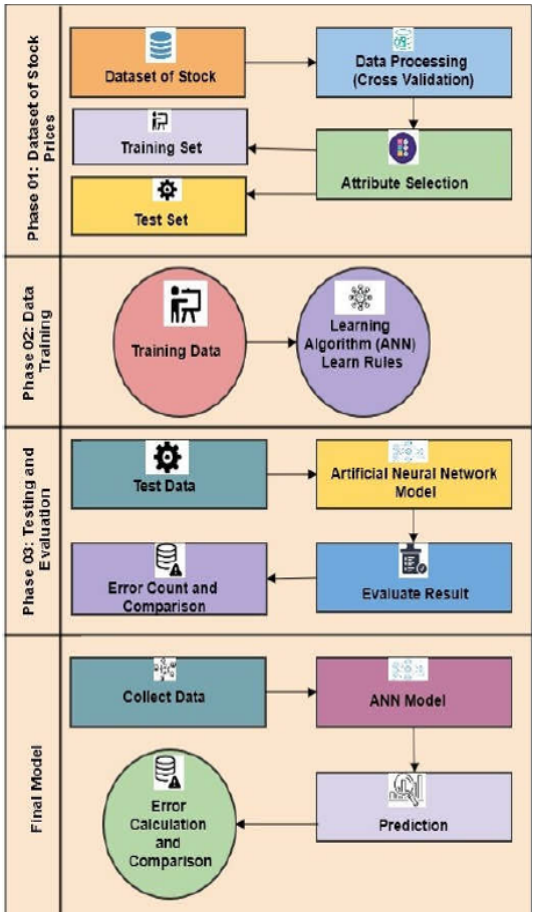

Figure 5.2: Implementation Workflow

a. Import Libraries: Use libraries such as TensorFlow or Keras to build the LSTM-based RNN model. b. Define Model Architecture: Specify the number of LSTM layers, the number of units in each layer, and any other relevant parameters. Experiment with different architectures to find the best configuration. c. Add Dropout: Consider adding dropout layers to prevent overfitting. Dropout randomly drops out a fraction of the units during training, which helps to regularize the model. d. Output Layer: Include an output layer that predicts the desired stock price based on the input sequence. e. Compile the Model: Proceed to compile the model with an appropriate loss function (e.g., mean squared error) and an optimizer (e.g., Adam optimizer).

Model compiling

Configure the model by specifying the optimizer, loss function, and evaluation metrics to be used during training.

Model fitting

a. Train the Model: Fit the model to the training data using the fit() function. Specify the number of epochs (iterations) and the batch size. Monitor the training progress by observing the loss and other relevant metrics. b. Validate the Model: Validate the model's performance using the testing dataset. Evaluate metrics such as accuracy, mean absolute error (MAE), or root mean squared error (RMSE).

Model prediction

a. Generate Predictions: Use the trained model to predict stock prices for the desired time horizon. Provide the model with input sequences and obtain the predicted values. b. Inverse Scaling: If you previously scaled the data, reverse the scaling to obtain the actual stock price predictions.

Result visualization

Visualize the predicted stock prices alongside the actual prices to assess the model's performance. Plotting line charts or candlestick charts can help you compare the predicted and actual values.

5.3 Dataset

We're using time-series data which is taken by the top 50 company NIFTY stocks from 2000 to 2021. Our dataset is taken from Kaggle which is a dependable and secure source. Our choice of using Indian Nifty 50 Stocks is motivated by two factors: lots of data and high liquidity. The abundance of data that characterizes Indian Nifty 50 Stocks, even in small time intervals, is essential for training our models since a shortage or absence of trading data can disrupt the learning process. Our dataset contains the date, symbol, series, prev close, high, low, open, VWAP(volume weighted average price), volume, turnover, trades, deliverables, and percent deliverables. Raw trade data contain the date and time that the trade took place, the price of the trade, and its size (in stocks).

Features	Meaning
Date	The stock value date
Open	Open price of the stock, at the beginning of trading day
High	Highest point of the stock price, on a trading day.
Low	Lowest point of the stock price, on a trading day.
Close	Close price of the stock, at the end of a trading day.
Adj close	Amended closing price for dividends of stock value the stock's value after distributing dividends.
Volume	Number of traded stocks in the market over a period

Figure 5.3: Dataset Columns

We are using High frequency data which refers to time-series data collected at an extremely fine scale. As a result of advanced computational power in recent decades, high frequency data can be accurately collected at an efficient rate for analysis.

Our dataset is inter day data which is also high frequency which means we are considering everyday data .

Chapter 6

Testing

The Deep Learning model's performance was tested based on our dataset. The dataset was split into train, test and evaluation data at 80 percent for training, 20 percent for testing and and the remaining 20 percent for evaluation of the models performance.

The testing process for machine learning (ML) models involves evaluating the performance, robustness, and generalization capabilities of the model.

We are analyzing the test coverage to ensure that all critical functionalities and scenarios have been adequately tested.

Steps invloded in our testing process :

1.Preparing a representative test dataset that covers a wide range of scenarios and variations. The dataset should be separate from the training data used to build the model. It should include both labeled data for supervised learning and unlabeled data for unsupervised learning or anomaly detection.

2.Defining the objectives of the testing process. Determine what aspects of the ML model you want to evaluate, such as accuracy, precision, recall, F1-score, or other relevant metrics. Consider factors like model performance, reliability, scalability, interpretability, and fairness.

3. We are providing an overview of the test case execution results, highlighting any failed or blocked test cases. Also, reporting the identified defects or issues, along with the respective fixes or workarounds implemented.

4. Presenting the performance metrics and compare them against the defined thresholds or benchmarks. Identifying any bottlenecks or performance degradation areas that require

attention.

5. Conducting a final review of the testing process and test results. Obtain stakeholders' approval and sign-off, indicating their satisfaction with the testing outcomes. This sign-off serves as a validation that the quantitative finance model visualizer using high-frequency trading algorithms meets the specified requirements and is ready for deployment.

Continuously iterate on the testing process and incorporate feedback from users and stakeholders to improve the system's reliability, accuracy, and usability. Document the testing process, including the test objectives, dataset details, evaluation metrics, performance analysis, and any relevant insights or recommendations. This documentation serves as a record of the model's performance and can help guide future improvements or model iterations.

Chapter 7

Result

We have split train and test data into 65 percent to 35 percent. We have trained different models and predicted the output. We are analyzing the results of advanced stock prediction models using Support Vector Regression (SVR), Random Forest Regression (RFR), k-Nearest Neighbors (KNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU), we are considering the following aspects:

Prediction Accuracy: Compare the accuracy of each model in terms of how closely their predicted values align with the actual stock prices or returns. Calculate metrics such as mean absolute error (MAE), mean squared error (MSE), or root mean squared error (RMSE) to quantify the prediction errors.

Performance Metrics: Evaluate additional performance metrics specific to stock prediction tasks. These may include metrics such as directional accuracy, which measures how often the model correctly predicts the direction of stock price movements (e.g., up or down).

Visualizations: Visualize the predicted stock prices or returns against the actual values to observe any patterns or discrepancies. Plots such as line graphs or candlestick charts can provide a visual representation of the model's predictions and their alignment with the true values.

Feature Importance: If applicable, analyze the feature importance provided by models such as RFR to identify the most influential factors affecting stock prices. This can help gain insights into which variables have the strongest impact on the predictions.

Model Comparison: Compare the performance of the different models used (SVR, RFR, KNN, LSTM, GRU) based on the evaluation metrics and visualizations. Identify the strengths

and weaknesses of each model and determine which one performs best for the specific stock prediction task.

Time Series Analysis: For LSTM and GRU models, analyze the ability of these models to capture temporal dependencies and patterns in the stock data. Evaluate whether these models successfully capture long-term dependencies and effectively predict stock price movements over time.

Model Robustness: Assess the robustness of the models by testing their performance on different time periods or multiple stocks. This helps determine if the models generalize well to unseen data and can be used for broader stock prediction tasks.

Limitations and Future Improvements: Discuss any limitations or challenges encountered during the analysis, such as data availability, model assumptions, or overfitting. Suggest potential improvements or further research directions to enhance the accuracy and reliability of the models.

By considering these aspects, you can gain insights into the performance, strengths, weaknesses, and potential areas of improvement for the SVR, RFR, KNN, LSTM, and GRU models in advanced stock prediction tasks.

We are calculating following values for comparision of models

Variance regression score

The variance score explains the dispersion of errors of a given dataset, and the formula is written as follows: Here, and Var(y) is the variance of prediction errors and actual values respectively. Scores close to 1.0 are highly desired, indicating better squares of standard deviations of errors.

R2 Score

R-squared (R2) is a statistical measure that represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.

Evaluation metrices RMSE, MSE and MAE

Root Mean Square Error (RMSE), Mean Square Error (MSE) and Mean absolute Error (MAE) are a standard way to measure the error of a model in predicting quantitative data.

Result of all models are shown below:

Figure 7.1: SVR Model Result

Figure 7.2: KNN Model Result

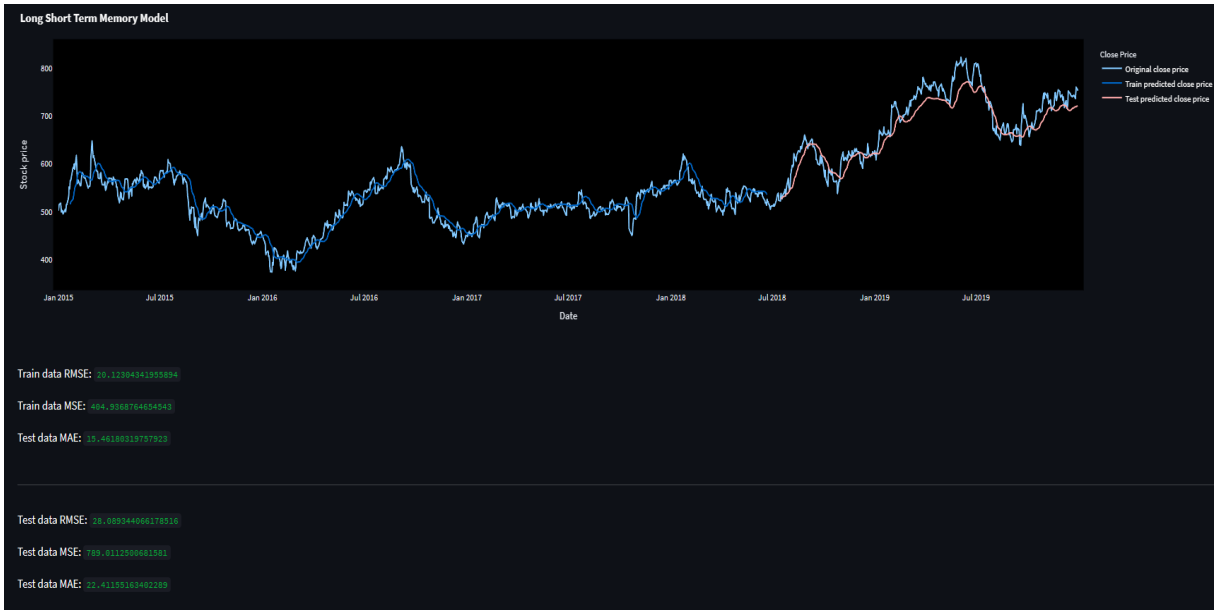

Figure 7.3: LSTM Model Result

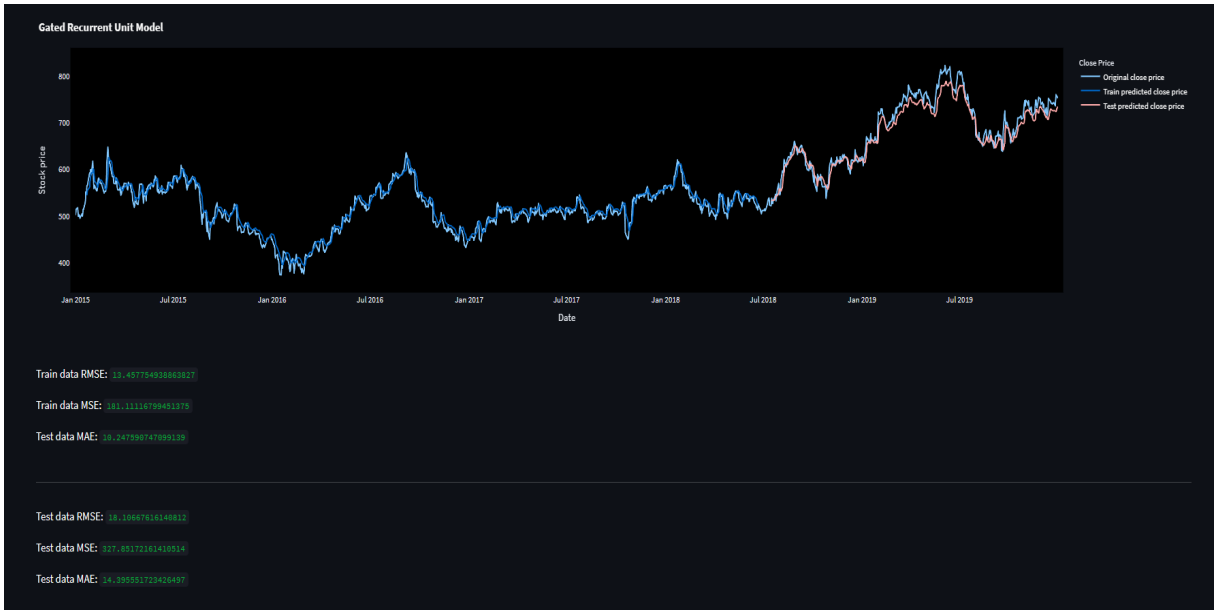

Figure 7.4: GRU Model Result

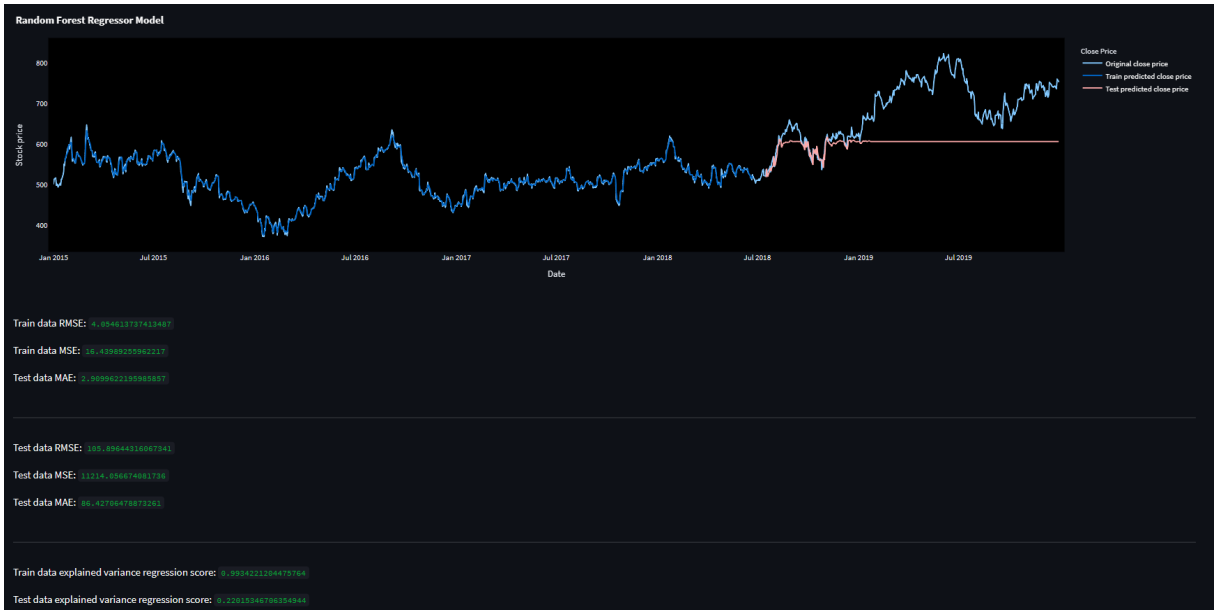

Figure 7.5: Random Forest Model Result

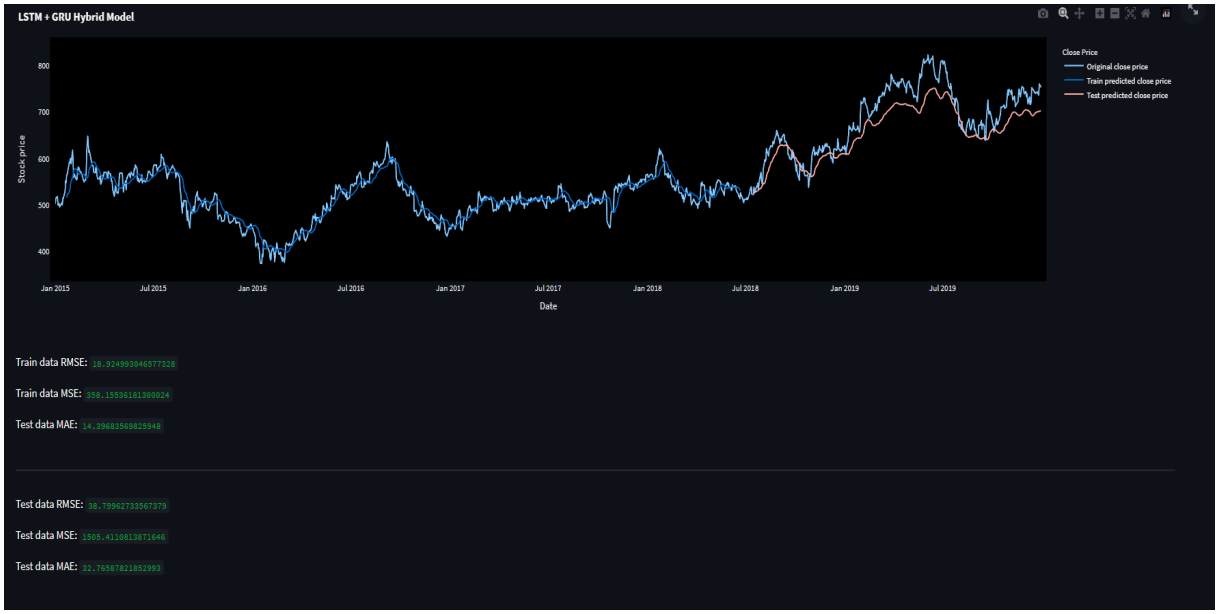

Figure 7.6: LSTM with GRU Model Result

At last , we have compared all the models and showing in a comparision table with their respective error values in figure 7.7. After seeing the result , we have come to the conclusion that Lstm with GRU hybrid model is giving the most accurate result with low error rates. Hence, GRU with LSTM hybrid models out performs all other models.

RELIANCE STOCK *CLOSE PRICE* PREDICTION

Total Rows = 249 and Columns = 7 Training data = 65% (161 rows) Testing Data = 35% (88 rows)

Algorithms	Main tuning Parameters (Hyper parameters)	RMSE		MSE		MAE		Variance Regression Score		R^2 score		Mean Gamma Deviance		Mean Poisson Deviance	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Super Vector Regressor	kernel = 'rbf' C = 0.001 gamma=0.1 degree= 3 epsilon=0.1	37.27	36.34	1389.39	1320.79	31.28	27.23	0.90	0.84	0.90	0.83	0.00032	0.00030	0.67	0.63
Random Forest	n_estimators=100 random_state=O	15.94	36.39	254.20	1323.92	11.31	28.44	0.98	0.84	0.98	0.83	6.09874	0.00030	0.12	0.63
K-nearest Neighbor	n_neighbours = 15 metric='minkowski'	48.43	57.96	2345.29	3359.48	36.20	42.47	0.83	0.61	0.83	0.57	0.00054	0.00077	1.13	1.61
LSTM (epchs=200, Batch=5)	loss='mse' optimizer='adam' Three LSTM layers with 32 nodes	33.42	34.92	1116.76	1219.28	25.71	26.02	0.92	0.85	0.92	0.85	0.00026	0.00028	0.54	0.58
GRU (epchs=200, Batch=5)	loss='mse' optimizer='adam' Four GRU layers with 32 nodes	24.43	48.30	597.04	2332.68	18.72	36.05	0.96	0.71	0.96	0.70	0.00014	0.00053	0.59	1.12
LSTM + GRU (epchs=200, Batch= 5)	loss='mse' optimizer='adam' Two GRU layers with 32 nodes and two LSTM layers with 32 nodes	29.59	37.13	875.56	1378.43	21.97	27.93	0.94	0.83	0.94	0.83	0.00021	0.00031	0.42	0.66

Figure 7.7: Result Comparison Table

Conclusion

This study used the Indian Stock Market financing dataset and a variety of methodologies.We have created an application for predicting close stock prices using the LSTM algorithm. We are doing this for the closing stock price of any particular firm. We used datasets from the Nifty50 Stocks and obtained extremely high accuracy for them. Positive outcomes have been achieved thanks to these strategies, which have improved prediction accuracy. The study and forecast of stocks using recently released machine learning techniques have produced encouraging results, paving the way for their implementation in lucrative exchange schemes.We have come to the conclusion that by applying machine learning techniques, stock market predictions can be made more effectively and accurately. By using a dataset that is far bigger than the one being utilized right now, the stock market prediction system can be significantly improved in the future. The accuracy of our prediction models would increase as a result. The stock market is renowned to be unpredictable, volatile, and nonlinear. It is highly challenging to predict stock prices with any degree of accuracy because of numerous (macro and micro) variables, including politics, global economic conditions, unforeseen events, a company's financial performance, and others. To find out what accuracy rate different machine learning models provide, additional research could be done. We can expand this program in the future to forecast bitcoin trade, and we can also use sentiment analysis for more accurate forecasts.

References

1. Selvin S, Vijayakumar R, Gopalakrishnan EA, Menon VK, Soman KP. Stock price prediction using LSTM, RNN, and CNN sliding window model. In 2017 international conference on advances in computing, communications, and informatics (icacci). 2017:1643-1647. IEEE.

2. Bing Y, Hao JK, Zhang SC. Stock market prediction using artificial neural networks. In the Advanced Engineering Forum. Trans Tech Publications Ltd. 2012;6:1055-1060.

3. Gurjar M, Naik P. Mujumdar G. Vaidya T. Stock market prediction using ANN. Int. Res. J. Eng. Technol. (IRJET). 2018;5:2758-2761.

4. Hiransha M. Gopalakrishnan EA, Menon VK, Soman KP. NSE stock market prediction using deep-learning models. Procedia computer 2018;132:1351-1362. science.

5. Utomo D. Stock price prediction using back propagation neural network based on gradient descent with momentum and adaptive learning rate. Journal of Internet Banking and Commerce(2017).

6. Chandar SK, Sumathi M, Sivanandam SN. Prediction of the stock market price using a hybrid of wavelet transform and artificial neural network. Indian Journal of Science and Technology (2016).

7. Borovkova S, Tsiamas I. An ensemble of LSTM neural networks for high-frequency stock market classification. Journal of Forecasting. 2019;38(6):600-619.

8. Araujo RDA, Oliveira AL, Meira S. A hybrid model for high-frequency stock market forecasting. Expert Systems with Applications. 2015;42(8):4081-4096.

9. Deepak RS, Uday SI, Malathi D. Machine learning approach in stock market prediction. International Journal of Pure and Applied Mathematics. 2017;115(8):71 77.

10. Sarioglu E, Aladag CH, Yolcu U, Uslu VR, Basaran MA. A new approach based on artificial neural networks for high-order multivariate fuzzy time series. 2009

11. Miah MBA, Hossain MZ, Hossain MA, Islam MM. Price prediction of the stock market using a hybrid model of artificial intelligence. International Journal of Computer Applications (2015).

12. G. Box and G. Jenkins, Time Series Analysis: Forecasting and Control, Holden-Day, San Francisco, CA, USA, 1976.

13. J. H. Wang , "Stock market trend prediction using ARIMA-based neural networks," Proceedings of International Conference on Neural Networks (ICNN). View at: Google Scholar.

14. C. N. Babu and B. E. Reddy, "Selected Indian stock predictions using a hybrid ARIMA-GARCH model," in Proceedings of the International Conference on Advances in Electronics, Venice, Italy, August 2015. View at: Google Scholar.

15. C. Li and T. W. Chiang, "Complex neuro fuzzy ARIMA forecasting—a new approach using complex fuzzy sets," IEEE Transactions on Fuzzy Systems, vol. 21, no. 3, pp. 567–584, 2013. View at: Publisher Site — Google Scholar

16. R. Singh and S. Srivastava, "Stock prediction using deep learning," Multimedia Tools and Application, vol. 76, no. 18, pp. 18569–18584, 2017. View at: Publisher Site — Google Scholar

17. K. Vaibhav and M. Garg, "Prediction of stock closing price by the hybrid deep neural network," European Journal of Advances in Engineering and Technology, vol. 5, no. 4, pp. 282–287, 2018. View at: Google Scholar

18. D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with LSTM neural networks," in Proceedings of the Interna-

tional Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, May 2017. View at: Google Scholar

19. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997. View at: Publisher Site — Google Scholar

20. L. C. Cheng, Y. H. Huang, and M. E. Wu, "Applied attention-based LSTM neural networks in stock prediction," in Proceedings of the International Conference on Big Data (Big Data), Seattle, WA, USA, December 2018. View at: Google Scholar

21. L. Zhang, C. Aggarwal, and G. J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD), Halifax, Canada, August 2017. View at: Google Scholar

22. Y. Zhao, Y. Shen, Y. Zhu, et al., "Forecasting wavelet transformed time series with attentive neural networks," in Proceedings of the International Conference on Data Mining (ICDM), Singapore, November 2018. View at: Publisher Site — Google Scholar

23. Y. Zhao, Y. Shen, and Y. Huang, "DMDP: a dynamic multi-source default probability prediction framework," Data Science and Engineering, vol. 4, no. 1, pp. 3–13, 2019. View at: Publisher Site — Google Scholar

24. Korhan Gokmenoglu1, Baris Memduh Eren1,2, and Siamand Hesami1, " Exchange rates and stock markets in emerging economies: new evidence using the Quantile-on-Quantile approach",2021. —Google Scholar

25. AlexSherstinsky, "Fundamentals of Recurrent Neural Network and Long short term memory", 2019.