A Project Report

on

Novel stock prediction for Indian stock market using Ensemble techniques

submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

19WH1A0594	Priyanka Gadela
19WH1A05B1	M. Yogitha
19WH1A05B4	K. Sai Teasmitha

under the esteemed guidance of

Ms. Shanmuga Sundari
Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD
College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090
June, 2023

DECLARATION

We hereby declare that the work presented in this project entitled **"Novel stock prediction for Indian stock market using Ensemble techniques**" submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of Ms. Shanmuga Sundari, Assistant Professor, Department of CSE.

Priyanka Gadela
(19WH1A0594)

M. Yogitha
(19WH1A05B1)

K. Sai Teasmitha
(19WH1A05B4)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**Novel stock prediction for Indian stock market using Ensemble techniques**" is a bonafide work carried out by Priyanka Gadela (19WH1A0594); M. Yogitha (19WH1A05B1); K. Sai Teasmitha (19WH1A05B4) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. E. Venkateswara Reddy
Professor and HoD,
Department of CSE

Guide
Ms. Shanmuga Sundari
Assistant Professor

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. E. Venkateswara Reddy, Professor & HOD**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Ms. Shanmuga Sundari, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for her constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. Priyanka Gadela
(19WH1A0594)

Ms. M.Yogitha
(19WH1A05B1)

Ms. K. Sai Teasmitha
(19WH1A05B4)

CONTENTS

ABSTRACT

The stock price crisis is nothing but a significant drop in stock price within a few days due to heavy selling in stocks. It can be due to financial crisis, pandemic situations etc. Stock crisis prediction helps investors to exit the market at the right time. The fair value of the stock price depends on stock financial parameters such as price to earnings, company dept etc.

In Stock Market Prediction, the aim is to predict the future value of the financial stocks of a company. The recent trend in stock market prediction technologies is the use of machine learning which makes predictions based on the values of current stock market indices by training on their previous values. Deep learning itself employs different models to make prediction easier and authentic.

Novel stock prediction for Indian stock market using algorithms Deep Neural Networks and XGBoost with measured accuracy in terms of RMSE. We would be using LSTM followed by ensemble techniques to predict the stock market price on Indian stock market dataset.

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Objectives

The objective of the proposed project is to develop a novel stock prediction model for the Indian stock market by combining Long Short-Term Memory (LSTM) with ensemble techniques. The aim is to improve the accuracy and reliability of stock market predictions, enabling investors and traders to make informed decisions and maximize their returns. The realm of interest is that the dataset of the stock prices from past years. The raw dataset must be pre- processed for data analysis. We need to compare the methods implemented in the past for the prediction of stock market value. The comparison between the results of the previous work help in determining the efficient algorithm and building the system model based on that selected algorithm.

1.2. Methodology

1.2.1. Dataset

File Size: 11.2 KB

Number of Rows: 1,610

Number of Columns: 7Description: This file contains historical stock prices of the Dow Jones Industrial Average (DJIA) index[19]. Each row represents a specific date, and the columns contain the following information:

"Date": The date of the stock price.

"Open": The opening price of the DJIA index on that date.

"High": The highest price reached by the DJIA index on that date.

"Low": The lowest price reached by the DJIA index on that date.

"Close": The closing price of the DJIA index on that date.

"Volume": The trading volume of the DJIA index on that date.

"Adj Close": The adjusted closing price of the DJIA index on that date, accounting for any corporate actions such as stock splits or dividends.

Note: The two files share the "Date" column, allowing you to merge the datasets based on this common column when combining the news headlines with the corresponding stock prices.

1.2.2. The Proposed Model

To improve the performance of the model, the proposed system forecasts the stock market price using LSTM followed by Ensemble techniques.

LSTM can capture the complex patterns and dependencies in the time series data, while the ensemble techniques can improve the overall prediction accuracy by combining the predictions of multiple models.

By combining LSTM with ensemble techniques, we can leverage the strengths of both approaches and create a more accurate and robust model for predicting stock prices.

1.3. Organization of Project

The project for novel stock prediction for the Indian market using LSTM (Long Short-Term Memory) followed by ensemble techniques can be organized into several steps. Firstly, it is crucial to define the project's objectives clearly. This includes identifying the specific Indian market(s) to focus on and determining which stocks to predict. By establishing these goals, the project can maintain a clear direction throughout its execution.

The next step involves acquiring and preprocessing the necessary data. It is important to identify reliable sources for historical stock price data as well as related financial information such as volume and market indicators. This data will be used to train and validate the models. Preprocessing techniques should be applied to handle missing values, outliers, and normalize the data to ensure accurate and consistent results.

Feature engineering plays a vital role in stock prediction. Relevant features that can potentially impact stock prices need to be selected. Additionally, domain knowledge, technical indicators, or sentiment analysis can be used to engineer additional features that provide valuable insights. These features will serve as inputs for the LSTM model and subsequent ensemble techniques.

After feature engineering, the appropriate models for stock prediction need to be selected. LSTM is a popular choice for sequential data forecasting and can effectively capture long-term dependencies. However, it's also worth considering other models such as CNN, GRU, or transformer-based models like BERT. Evaluating and comparing the performance of different models using appropriate metrics will help in identifying the most suitable model for the task.

Once the model selection is complete, the models need to be trained and validated. The dataset should be split into training, validation, and testing sets. The LSTM models are trained on the training set using historical stock data, and hyperparameters and architecture are tuned using the validation set. It is crucial to validate the models using appropriate evaluation metrics and techniques, taking into account the time-series nature of the data. Cross-validation or other time-series-specific evaluation methods can be employed.

Ensemble techniques can be explored to further enhance the predictive power of the models. Techniques such as model averaging, stacking, boosting, or bagging can be applied to combine multiple LSTM models or other models. The performance of the ensemble should be evaluated and compared with individual models to assess their effectiveness in improving prediction accuracy and robustness.

Once the models and ensembles are trained and validated, backtesting and evaluation can be performed. The models are applied to the testing set to simulate real-world scenarios, and their performance is assessed using various evaluation metrics such as accuracy, precision, recall, F1-score, and profit/loss. Backtesting is also conducted to validate the models' predictive power on historical data, allowing for a comprehensive evaluation of their effectiveness.

Documentation and reporting play a crucial role in summarizing the project. It is essential to document the methodology, data preprocessing techniques, model selection, training details, evaluation metrics used, and the results obtained. This documentation serves as a reference for future improvements and can be shared with stakeholders, colleagues, or the wider community to disseminate the findings and contribute to the field.

Lastly, the project should embrace an iterative approach for continuous improvements. As new data becomes available, feedback is received, or emerging research presents novel techniques, the models, feature engineering methods, or ensemble strategies should be refined and updated accordingly. Staying updated with the latest advancements in stock prediction and machine learning techniques ensures that the project remains relevant and effective in the ever-evolving stock market

2. LITERATURE REVIEW

Title: Ensemble Technique with Optimal Feature Selection for Saudi Stock Market Prediction: A Novel Hybrid Red Deer-Grey Algorithm
Authors: SAUD S. ALOTAIBI

Summary:
The forecast of the stock price attempts to assess the potential movement of the financial exchange's stock value. The exact estimation of the movement of share price would contribute more to investors' profit. Further, the selected features are subjected to the ensemble technique for predicting the stock movement. Saudi Arabia is a well-established country in the oil markets, and being the core member of the Organization of "Petroleum Exporting Countries, Saudi Aramco, the national oil and gas company", by producing and maintaining billions of gallons of Saudi oil, including some 260 billion tonnes in inventory. "Saudi Stock Exchange " is the largest exchange in the Middle East, identified locally by its Arabic name Tadawul[4]. The forecasting of the stock price attempts to assess the potential movement of a financial exchange's stock value.

Title: Deep Reinforcement Learning Approach for Trading Automation in the Stock Market
Authors: TAYLAN KABBANI AND EKREM DUMAN

Summary:
Results show that the addition of technical indicators and sentiment scores of the news headlines to the state representation has significantly improved the agent's performance and the superiority of using a continuous action space over a discrete one to solve the trading problem[5]. we believe that training an NLP algorithm to process the financial news content instead of only the headline may positively affect the agent performance.

Title: Manipulator Detection in Cryptocurrency Markets Based on Forecasting Anomalies
Authors: FIRAT AKBA, IHSAN TOLGA MEDENI, MEHMET SERDAR GUZEL, AND IMAN ASKERZADE

Summary:
This study addresses to apply machine learning and statistical forecasting methods to detect manipulations in Bitcoin market. Accordingly, the most commonly used price estimation methods, involving time series forecasting, machine and deep learning techniques were implemented to determine the weekly/monthly rising and falling trends of Bitcoin pricing[6]. The first priority was to make sure that price estimation methods could produce truly reliable results. To achieve this, we conducted our experiments by identifying a minimally manipulation-free zone. Afterwards, the leading sentiment analysis techniques were used in order to understand the effect of social media posts on prices. During these studies, we also enhanced the ISSFS method, which was previously suggested by us for sentiment analysis.

Title: Market Making Strategy Optimization via Deep Reinforcement Learning
Authors TIANYUAN SUN, DECHUN HUANG, AND JIE YU

Summary:
The market making strategy optimization is an attractive topic for both researchers and practitioners[7]. With the development and successful application of deep reinforcement learning models, how to use deep reinforcement learning model to market making strategy becomes an interesting research problem. In this paper, we propose an end-to-end reinforcement learning market making strategy based on recurrent deep network representation, DRLMM. It exploits LSTM network to extract temporal patterns of the market directly from the LOBs, and it learns state-action relations via a reinforcement learning approach. In order to control inventory risk and information asymmetry, a deep Q-network is introduced to adaptively select different action subsets and 9092 VOLUME 10, 2022 T. Sun et al.: Market Making Strategy Optimization via Deep Reinforcement Learning train the market making agent according to the inventory states. Experiments are conducted on a six-month Level-2 data set, including 10 stock, from Shanghai Stock Exchange in China. Our model is compared with two baseline market making strategies. Experimental results show that: 1) the DRLMM performs better than the benchmark MM strategies; 2) using

DRQN to directly extract market information and construct market features can make the state representation in DRLMM better than manually made features; 3) the adaptive action space can improve the training process of DRLMM as well as the profitability of the MM strategy.

Title: Novel Stock Crisis Prediction Technique—A Study on Indian Stock Market
Authors: NAGARAJ NAIK, AND BIJU R. MOHAN

Summary:
Stock Crisis identification is a difficult task due to more fluctuations in the stock market. This is the first approach to address stock crisis prediction based on stock financial parameters and stock price to the best of our knowledge based on literature[8]. We have proposed the Hybrid Feature Selection (HFS) algorithm to remove irrelevant stock financial parameters features. The NB classifier method is considered to find the fundamentally strong stock. Later, stock over price is identified by using the RSI method. Moving average statics are considered to identify the stock crisis points. The effectiveness of the model is quantified by using the XGBoost and DNN regression method.
Therefore in future work, different fundamentals stock and technical parameters can be applied to improve the model accuracy. We have explored a limited number of technical parameters of stock prices. In the future, the researchers can explore the other technical indicators to predict the crisis point.

Title: ML-GAT: A Multilevel Graph Attention Model for Stock Prediction
Authors: KUN HUANG, XIAOMING LI, FANGYUAN LIU, XIAOPING YANG, AND WEI YU

Summary:
In this study, we propose a multilayer graph attention neural network model for stock trend prediction[9]. We incorporate data describing financial markets, news, and corporate relations into a graph neural attention network-based model through a specific feature extraction module to compensate for the lack of prior knowledge of existing stock forecasting methods. ML-GAT aims to selectively filter different types of information to form an aggregated graph through multiple layers of attention mechanisms at different levels to learn the feature representation of nodes that are useful for prediction tasks.

Title: A Machine Learning-Based Early Warning System for the Housing and Stock Markets
Authors: Daehyeon Park and Doojin Ryu

Summary:

This study proposes an early warning system for risks of the housing markets. There may be a chance of occurrence of housing bubbles because sometimes housing market supply takes more time to construct the houses which leads to sudden increase in the prices due to which the country's economy can be collapsed. To overcome such difficulties we will be developing an early warning system using a long short-term memory (LSTM) which is used to send a signal to detect whether there are any housing market risks or not[10].

Housing market bubbles can damage the overall economy of the world. It will be more effective if we can detect the instability in housing markets and stock markets simultaneously by using long short-term memory(LSTM) neural network. We will be uniquely defining the housing bubble signals and links between the signals and stock markets.

Title: Asymmetric Risk Spillovers Between China and ASEAN Stock Markets
Authors: Jiusheng Chen and Xianning Wang

Summary:

This study finds the asymmetric spillovers between China and ASEAN stock markets by using Copula-TV-GARCH-CoVaR model and MES model which can be used to detect whether there are any risk spillovers between any countries to avoid the stock market downturns. These results are given by the international portfolio managers and policymakers.

The dependency between China and ASEAN markets have been increased over time because of Belt and Road initiatives. By using Copula-TV-GARCH-CoVaR and MES model we can evaluate the time-varying unconditional variance of stock market series. We have estimated the dependence structure between seven countries by which we have concluded that Vietnam is the only country which has less dependency structure with China stock market[11].

CoVaR finds risk spillovers between markets by providing information on the value-at-risk (VaR) of a market. The second methodology detects the spillover effect between two different markets by using the MES model.

Title: Decision Fusion for Stock Market Prediction: A Systematic Review

Authors: Cheng Zhang and Roslina B. Ibrahim

Summary:

This study represents how to provide different forecasts of a model having different structures for predicting the stock market prices[12]. We apply decision fusion to get better prediction by combining forecasts from multiple models.

A forecasting model that applies decision fusion should consists of homogenous base learners such as ANN, decision trees, SVM and LSTM. The data type of forecast is a critical factor for selecting fusion methods. Voting and Tree based methods are used for classification where as simple averages and ANN are used for regression.

We have applied techniques like Artificial Neural Networks(ANNs), fuzzy based techniques for effective predictions and deep learning methods like Long short-term memory(LSTM), convolutional neural networks(CNNs) and deep neural networks(DNNs).

Title: Stock Trend Prediction Using Candlestick Charting and Ensemble Machine Learning Techniques With a Novelty Feature Engineering Scheme

Authors: Yaohu Lin and Harris Wu

Summary:

Stock market forecasting is a knotty challenging task due to the highly noisy, nonparametric, complex and chaotic nature of the stock price time series[13]. Several machine learning techniques, including deep learning methods, are applied to stock data to predict the direction of the closing price. The forecasting of the stock market is an important objective in the financial world and remains one of the most challenging problems due to the non-linear and chaotic financial nature.

The empirical results show that the forecasting framework of this paper has predictive power, and the investment strategy based on the forecasting model can generate superior returns. We intend to incorporate more suitable machine learning methods for prediction, such as reinforcement learning methods, into the ensemble model. In addition, we can utilize additional predictive factors, to improve forecasting results in the future.

The main algorithm used is main evaluation algorithm.Recision is used to estimate the accuracy of positive samples in the prediction data and recall is used to evaluate the coverage of positive samples in the prediction data of the model. They are indicators used in statistics to measure the accuracy of a binary model. The evaluation progress tries to evaluate the results of the machine learning prediction model.

Title: One Step Ahead: A Framework for Detecting Unexpected Incidents and Predicting the Stock Markets

Authors: Ziyue Li, Shiwei Lyu and Tianpei Jiang

Summary:

Incidents such as the Fukushima nuclear disaster, and the COVID-19 out breaks severely shocked both local and global markets[14]. For investors, it is crucial to quantify the key facts that affect the incidents' impacts, and to estimate the reactions of the markets accurately and efficiently for global event-driven investment strategies.

Demonstrated the framework for detecting unexpected incidents and predicting the market directions with narrative decision logic. First, we can predict the impacts of more categories of incidents such as natural disasters and disease outbreaks in more regions. Second, we need to extract more information and use external knowledge bases to fill in missing attributes.

We choose Unexpected Incident classification model, Multiple classification model which classifies instances into one of three or more classes and also Market models to illustatrate hoe forces of supply and demand interact to determine prices and the quantity that is sold.

Title: Optimal Setting for Hurst Index Estimation and Its Application in Chinese Stock Market

Authors: Liang Ding and Yan Lin

Summary:

The results show that the settings have significant impact on the accuracy of the Hurst index estimation. The daily volatility in the Chinese stock market show long memory by the Hurst index. Our contribution is to identify the optimal model setting for the Hurst index estimation by comparing all the six optional settings in every step of estimation[15].

The H index is designed to improve upon simpler measures such as time series and estimation of H index is important for research. Therefore, H index estimation and how to deal with them are fields worthy of future research.

We use monte Carlo simulation is a model used to predict the probability of a variety of outcomes when the potential of random variables is present. It helps to explain the impact of risk and uncertainity in prediction and forecasting models.

Title: Stock Price Manipulation Detection Using Deep Unsupervised Learning: The Case of Thailand

Authors: Teema Leangarun and Poj Tangam Chit

Summary:

Manipulation is an important requirement for financial markets to maintain efficiency. The majority of the related work employed supervised learning techniques, which necessitated known manipulation patterns as examples for their models to recognize. The models were trained to recognize normal trading behaviors that were expressed in a limit order book.

Normal trading data were used for training and testing of the models, while real and synthetic manipulation data were only used for performance evaluation. For the synthesized manipulation data, the results of the study indicated that both models were effective in detecting stock price manipulation with low false positives, despite the fact that they had no prior knowledge about stock manipulations[16].

For detecting stock price manipulation, we proposed an unsurpervised deep learning approach. The performance of two deep unsupervised models: LSTM-AE and LSTM-GANs was compared in this study. Both models were trained to recognize normal trading behaviors and treat other behaviors as manipulated. Both models can identify manipulations in five out of six cases in the real manipulation data.

Title: Using Trend Ratio and GNQTS to Assess Portfolio Performance in the U.S. Stock Market

Authors: Yao Hsin Chou and Yun Ting Lai

Summary:

In order to spread the risk, investors tend to invest in a portfolio, thus, how to assess portfolio performance and identify the best portfolio with stable uptrend are important issues. However, as there are many different companies that go public in the stock market, before investors invest in the stock market, the first issue they will encounter is how to select the stocks[17]. Therefore, the important issue is how to simultaneously assess the risk and return of a portfolio.

According to the experimental results, the used method can identify the best portfolio with stable uptrend and low risk. The trend ratio is calculated by dividing return by risk, which means the return per unit risk. When a portfolio has a high trend ratio, it means the portfolio has high return and low risk. The trend ratio is determined according to the trend line to identify the portfolio trend.

Here we use the Global-best guided Quantum-inspired Tabu Search algorithm with Not-gate (GNQTS) to effectively optimize a portfolio given limited time.Also,we use metaheuristic algorithm that attempts to find the best feasible solutions of an optimization problem.It is used to solve multi-objective multiple-solution and non-liner formulations.

Title: Reducing Manual Effort to Label Stock Market Data by Applying a Metaheuristic Search: A Case Study from the Saudi Stock Market

Authors: Mohammad Alsulmi

Summary:

Stock market trading, as a part of the financial domain, has benefited from these techniques in learning models to forecast the direction of stock prices. To address this limitation, they proposed an automatic labeling approach. The results of empirical experiments demonstrate that this approach is very practical as it outperforms the current manual approaches for stock data labeling and achieves higher labeling effectiveness[18].

Various manual approaches have been proposed for the stock data labeling problem. This study defined the stock data labeling problem and formulated it as an NP-hard problem. An automatic labeling solution was then proposed that uses metaheuristic search algorithms such as hill-climbing and simulated annealing.

3.THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

3.1. Requirements Gathering

3.1.1. Software Requirements

Operating System : Windows 10

Programming Language : Python 3.8

Tool : Google Colab, Jupyter Notebook

Packages : Numpy, Pandas, Matplotlib, Scikit-learn, Tkinter

3.1.2. Hardware Requirements

Processor : Intel core i5 processor

Memory : RAM 8GB

3.2. Technological Description

1. Programming Language:
    - Python: Python is a widely used programming language with excellent libraries and frameworks for data analysis, machine learning, and deep learning. It provides the necessary tools to implement LSTM models, ensemble techniques, and data preprocessing tasks.
2. Data Processing and Analysis:
    - Pandas: Pandas is a powerful library for data manipulation and analysis. It provides data structures such as DataFrames, which can efficiently handle and process large datasets.
    - NumPy: NumPy is a fundamental library for numerical operations in Python. It offers multidimensional arrays and mathematical functions for efficient computation.
3. Deep Learning Framework:
    - TensorFlow: These are popular deep learning frameworks that provide tools for building and training neural network models. They offer extensive support for implementing LSTM models and performing various operations on tensors.
4. LSTM Model Development:
    - Keras is a high-level neural network library that runs on top of TensorFlow or PyTorch. It simplifies the process of building and training neural networks, including LSTM models, through its easy-to-use API.
5. Ensemble Techniques:
    - Scikit-learn: Scikit-learn is a widely used machine learning library that offers various

ensemble techniques, such as bagging, boosting, and stacking. It provides classes and functions for implementing these ensemble methods.

6. Data Visualization:
   - Matplotlib: Matplotlib is a popular plotting library in Python. It enables the creation of visualizations to analyze and present the data in a visually appealing manner.
   - Seaborn: Seaborn is a statistical data visualization library built on top of Matplotlib. It provides higher-level abstractions and aesthetically pleasing visualizations.
7. Other Supporting Libraries:
   - Scikit-learn: In addition to ensemble techniques, Scikit-learn provides various tools for data preprocessing, feature selection, and evaluation metrics.
   - NLTK (Natural Language Toolkit): If dealing with textual data, NLTK offers a range of tools for natural language processing tasks, such as text cleaning, tokenization, and sentiment analysis.
8. Development Environment:
   - Jupyter Notebook: Jupyter Notebook provides an interactive environment for prototyping and experimenting with code.

4. DESIGN

4.1. Introduction

The Indian stock market is known for its dynamic nature and the potential for significant returns. However, accurately predicting stock prices is a challenging task due to the complex and volatile nature of the market. To address this challenge, this project proposes a novel approach that combines Long Short-Term Memory (LSTM) with ensemble techniques to improve stock market prediction accuracy for the Indian market.

LSTM is a type of recurrent neural network (RNN) that excels in capturing long-term dependencies and patterns in sequential data.To enhance the predictive power and robustness of the LSTM model, this project incorporates ensemble techniques. Ensemble methods involve combining the predictions of multiple models to achieve a more accurate and reliable prediction. It involves acquiring historical stock price data, gathering relevant features, performing data preprocessing, and developing a LSTM model. The LSTM model will be trained on the historical data, allowing it to learn from past patterns and trends.

To enhance the predictive power and robustness of the LSTM model, this project incorporates ensemble techniques. Ensemble methods involve combining the predictions of multiple models to achieve a more accurate and reliable prediction. By aggregating the predictions of diverse models, ensemble methods can effectively mitigate individual model biases and capture a broader range of patterns and trends in the data.

The proposed approach involves acquiring historical stock price data for Indian stocks, gathering relevant features such as opening price, closing price, trading volume, and any other factors that may influence stock prices. The collected data is then preprocessed to handle missing values, outliers, and normalize the numeric features.

Once the data is preprocessed, a LSTM model will be developed and trained on the historical data. The LSTM model will learn from the temporal patterns and dependencies present in the data, enabling it to make predictions on future stock prices. The model will be optimized using techniques such as backpropagation through time (BPTT) and gradient descent optimization to minimize prediction errors.

To further enhance the accuracy of predictions, ensemble techniques will be integrated into the model. Ensemble methods such as bagging, which involves training multiple LSTM models on different subsets of the data, will be explored. The predictions of these individual LSTM models will be combined using ensemble methods such as averaging, weighted averaging, or stacking, to create an ensemble model that provides a more robust and accurate prediction.

The evaluation of the proposed approach will be conducted using appropriate metrics such as root mean squared error (RMSE), mean absolute error (MAE), or mean absolute percentage error (MAPE). The performance of the LSTM model alone will be compared against the ensemble model to assess the effectiveness of the ensemble techniques in improving prediction accuracy for the Indian stock market.

The findings and insights from this project can provide valuable information and guidance to investors, traders, and financial institutions operating in the Indian stock market. By utilizing LSTM followed by ensemble techniques, stakeholders can make more informed investment decisions, optimize trading strategies, and potentially maximize their returns in this dynamic market environment.

4.2. Architecture Diagram

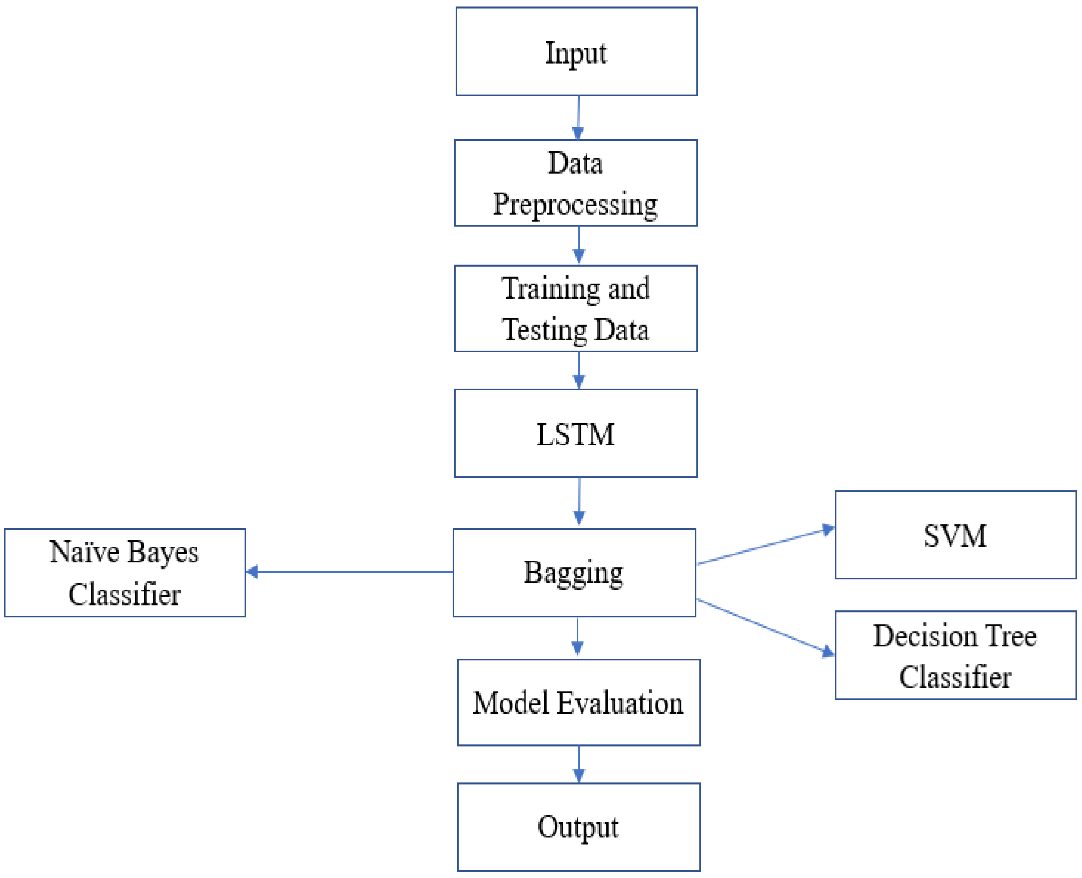

Fig. 1. Architecture of Stock market prediction

The dataset is given as input which consists of 7 attributes: Date, Open, High, Low, Close, Volume, Adj Close. We have preprocessed the data by checking for the null values and identifying and removing the outliers. We have scaled down the data between 0 and 1 by using fit transform which will generate the data into arrays. We have splitted the data into training and testing by creating separate dataframes. Training dataframe consists of close column starting from 0th index we need to go to 70% of the total value and in the testing dataframe close column starts from 70% of the total value and goes till complete length. In Fig. 1. We are implementing LSTM followed by Bagging algorithm by Using different algorithms like Naïve Bayes Classifier, SVM, Decision Tree Classifier and then we have evaluated the model by using Root Mean Squared Error.

4.3. Algorithms

4.3.1. LSTM

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture that has proven effective in modeling sequential data. Unlike traditional feed-forward neural networks, which lack memory, LSTM networks are capable of capturing long-term dependencies and handling data with time-based patterns.

The architecture of an LSTM network consists of memory cells that can store information for an extended period. These memory cells are responsible for maintaining and updating the cell state, which serves as a long-term memory. This enables LSTM to retain important information over longer sequences, making it suitable for tasks involving sequential data such as time series forecasting.

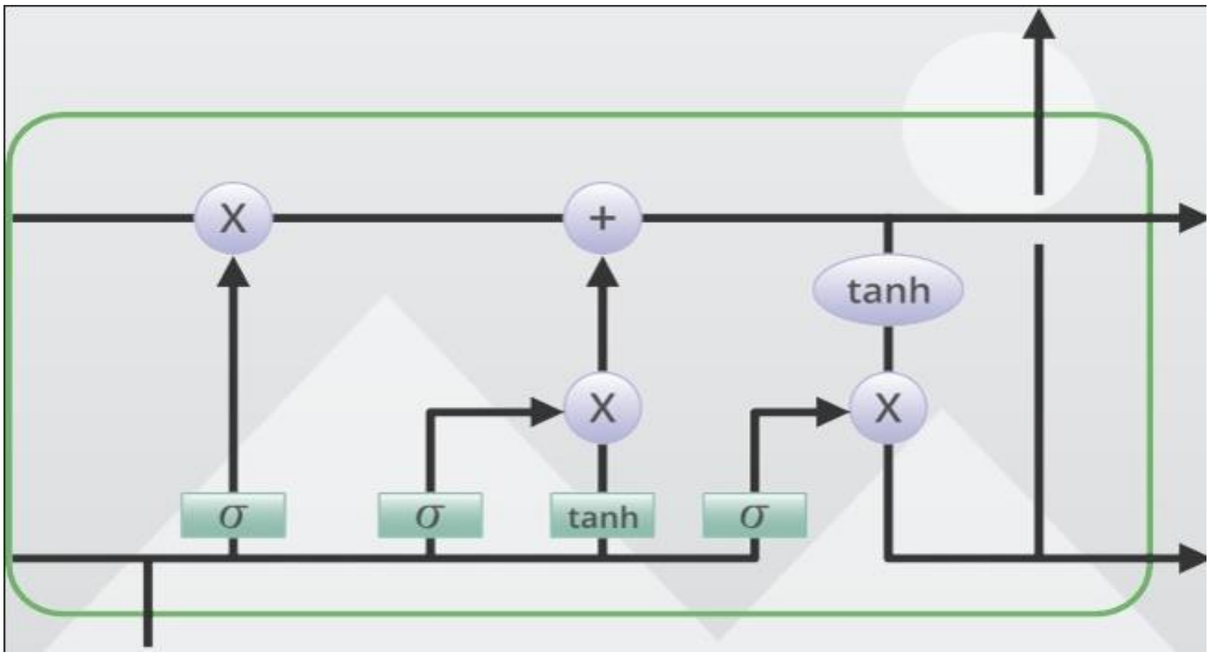

Fig. 2. Long Short - Term Memory

The key components of an LSTM unit include:

1. Cell State (Ct): The cell state carries information throughout the entire sequence and is responsible for preserving long-term dependencies. It is regulated by three types of gates: the forget gate, the input gate, and the output gate.

2. Forget Gate (ft): The forget gate determines which information from the previous cell state should be discarded or forgotten. It takes the previous hidden state (ht-1) and the current input (xt) as inputs, and its output is a sigmoid function that scales the importance of the previous cell state.
3. Input Gate (it): The input gate controls the flow of new information into the cell state. It consists of two components: the input gate itself (determines which values to update) and the candidate values (potential new values). The input gate takes the previous hidden state (ht-1) and the current input (xt) as inputs, and its output is a sigmoid function that determines the relevance of new information.
4. Candidate Values (C~t): The candidate values represent the new information to be added to the cell state. They are computed using the hyperbolic tangent (tanh) function that squashes the values between -1 and 1, allowing the LSTM to introduce new information to the cell state.
5. Updated Cell State (Ct): The updated cell state is obtained by combining the previous cell state (Ct-1) and the candidate values (C~t) using the forget gate and input gate. The forget gate decides which parts of the previous cell state should be discarded, while the input gate determines which parts of the candidate values should be added.
6. Output Gate (ot): The output gate regulates the information that will be output from the LSTM cell. It takes the previous hidden state (ht-1) and the current input (xt) as inputs, and its output is a sigmoid function that scales the updated cell state.
7. Hidden State (ht): The hidden state carries information that will be passed to the next time step and is determined by the updated cell state filtered through the output gate. The hidden state can be used for making predictions or passed as input to subsequent LSTM cells in a sequence.

During the training phase, the LSTM model learns the optimal values for its parameters through backpropagation and gradient descent optimization. The loss function is typically chosen based on the specific task, such as mean squared error (MSE) for regression problems or cross-entropy loss for classification tasks.
In the context of stock prediction, LSTM networks can analyze historical stock price data and capture patterns, trends, and relationships over time. The ability of LSTM networks to handle sequential data and capture long-term dependencies makes them well-suited for stock market prediction tasks. Overall, LSTM networks offer a powerful tool for modeling sequential data

and have been successfully applied to various domains, including natural language processing, speech recognition, and, importantly, stock market prediction.

LSTM models have shown great success in various applications, including speech recognition, natural language processing, and, importantly, time series forecasting. In the context of stock market prediction, LSTM can effectively capture the temporal dependencies and non-linear patterns in the historical stock price data, allowing for more accurate and reliable predictions.

4.3.2. Ensemble methods

Ensemble methods are techniques that involve combining the predictions of multiple models to improve overall prediction accuracy and robustness. By leveraging the diversity and collective wisdom of multiple models, ensemble methods aim to mitigate individual model biases and errors, leading to more reliable predictions. In the context of stock market prediction for the Indian market, ensemble methods can help enhance the accuracy and robustness of the predictions made by the LSTM models.

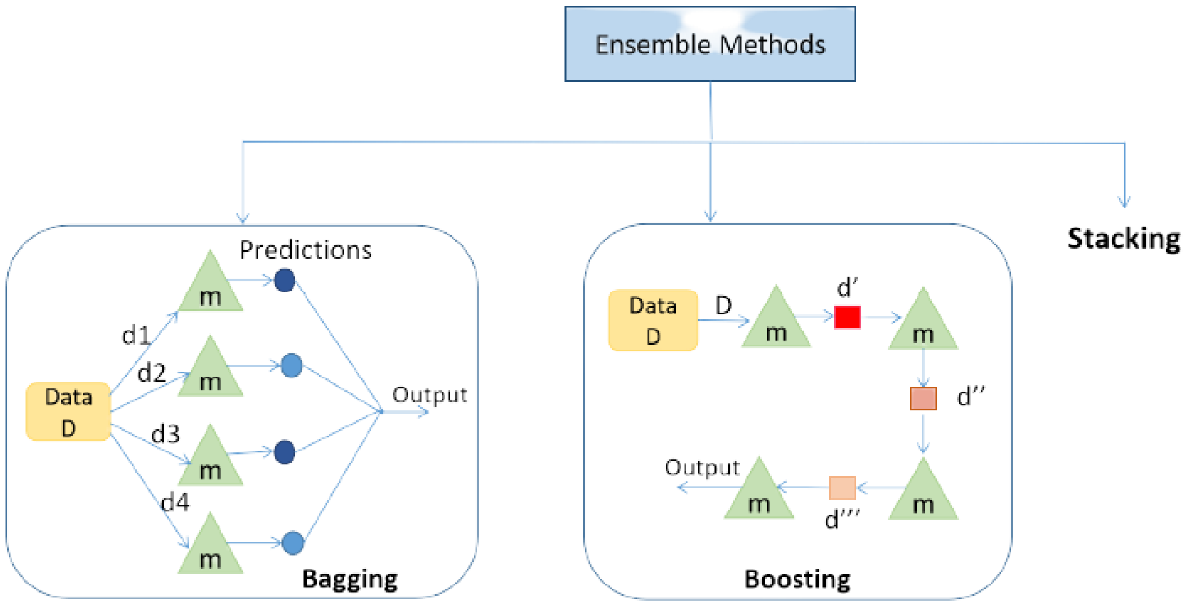

Fig. 3. Ensemble methods

There are several popular ensemble methods that can be applied to combine the predictions of multiple models. Some commonly used ensemble techniques include:

1. Bagging: Bagging (Bootstrap Aggregating) involves training multiple models independently on different subsets of the training data, sampled with replacement. Each model is trained on a different subset, and the final prediction is obtained by aggregating the predictions of all the models, typically by taking the average. Bagging helps reduce the variance of the predictions and can improve the stability and accuracy of the ensemble model.
2. Boosting: Boosting is an ensemble technique that involves training multiple models sequentially, with each model focusing on correcting the errors made by the previous models. The training process is iterative, where each model is trained to give more weight to the instances that were misclassified by the previous models. The final prediction is obtained by aggregating the predictions of all the models, typically using a weighted average. Boosting can improve the overall prediction accuracy by leveraging the strengths of each model in correcting the weaknesses of others.
3. Stacking: Stacking is a more advanced ensemble technique that combines the predictions of multiple models using a meta-model. In stacking, multiple base models are trained on the same training data, and their predictions are then used as input features for a higher-level meta-model. The meta-model learns to combine the predictions of the base models to make the final prediction. Stacking allows for more complex interactions between the base models and can potentially lead to improved prediction accuracy.

The choice of ensemble method depends on the specific problem and the characteristics of the data. Experimentation and cross-validation can help determine the most suitable ensemble technique for a given stock market prediction task. By combining the predictions of multiple LSTM models using ensemble techniques, we can leverage their individual strengths, mitigate biases, and improve the overall accuracy and robustness of stock market predictions for the Indian market.

4.3.3. Bagging

Bagging, short for Bootstrap Aggregating, is an ensemble learning technique used to improve the performance and stability of machine learning models. It involves training multiple models on different subsets of the training data and combining their predictions through voting or averaging.

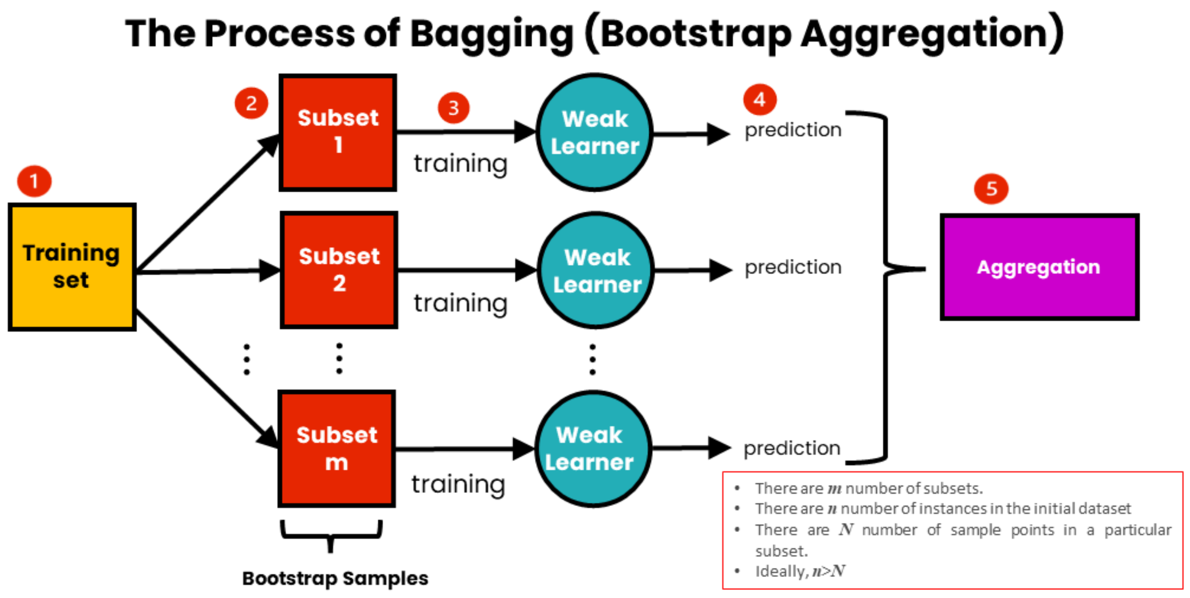
The Process of Bagging (Bootstrap Aggregation)
1
Training set
2
Subset 1
Subset 2
Subset m
3
training
training
training
Weak Learner
Weak Learner
Weak Learner
4
prediction
prediction
prediction
5
Aggregation
Bootstrap Samples
There are m number of subsets.
There are n number of instances in the initial dataset
There are N number of sample points in a particular subset.
Ideally, n>N

Fig. 4. Bagging

The process of bagging involves the following steps:

1. Data Sampling: The training dataset is randomly sampled with replacement to create multiple subsets, known as bootstrap samples. Each bootstrap sample has the same size as the original dataset but may contain duplicate instances. By using random sampling with replacement, each bootstrap sample becomes slightly different from the others.

2. Model Training: For each bootstrap sample, a base model (e.g., LSTM) is trained independently. The base model is trained on a particular bootstrap sample, which means that each model may have seen slightly different instances and patterns from the training data. This introduces diversity among the base models.

3. Prediction Aggregation: Once all the base models are trained, they are used to make predictions on new or unseen data. In the case of classification problems, the final prediction can be determined by majority voting, where the class with the most votes from the base

models is selected as the final prediction. For regression problems, the predictions of the base models can be averaged to obtain the final prediction.

The key benefits of bagging include:

1. Variance Reduction: Bagging helps reduce the variance of the predictions by aggregating the predictions of multiple models trained on different subsets of the data. By reducing variance, bagging can improve the stability and robustness of the ensemble model.
2. Mitigating Overfitting: Bagging reduces the risk of overfitting by training multiple models with different subsets of the data. Each model learns different patterns and relationships, and by combining them, the ensemble model can capture a broader range of patterns and generalizes better to unseen data.
3. Increased Prediction Accuracy: By leveraging the diversity of the base models, bagging can improve prediction accuracy compared to using a single model alone. The ensemble model benefits from the collective wisdom of the base models, which can lead to more accurate predictions.
4. Model Robustness: Bagging enhances the robustness of the ensemble model by reducing the impact of outliers or noisy data points. Since each base model is trained on different subsets of the data, outliers or noisy instances may have less influence on the final prediction.

It's important to note that while bagging can improve prediction accuracy, it may not be effective if the base models are highly correlated. To address this, different variations of bagging, such as Random Subspace Method or Random Patches, can be used to introduce additional diversity among the base models.

In the context of stock market prediction for the Indian market, applying bagging to LSTM models can help improve the accuracy and robustness of the predictions by leveraging the diversity of the models trained on different subsets of the data.

Bagging Algorithm :

```
Bagging (algorithm A, dataset S, iterations T)
1) model generation
for i = 1 to T:
        Create a bootstrap sample S(i) from S
        let O(i) be the outcome of training A on S(i)
2) prediction for the specified test instance x for i = 1 to T:
        let B(i) = output of O(i) on x return the class that is used the most B(1).. B(T)
```

Create a bootstrap sample from dataset S.
Let O(i) be the training outcome A on S(i).
For the specified test instance, B(i) is the output of O(i) on x
Which returns the class B(1)…..B(T)

4.3.4. Support Vector Machine

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features.

The main idea behind SVM is to find an optimal hyperplane that separates the data points of different classes or predicts the continuous target variable with the maximum margin. SVM achieves this by mapping the input data to a high-dimensional feature space and finding the hyperplane that maximally separates the classes or maximizes the margin in the transformed space.

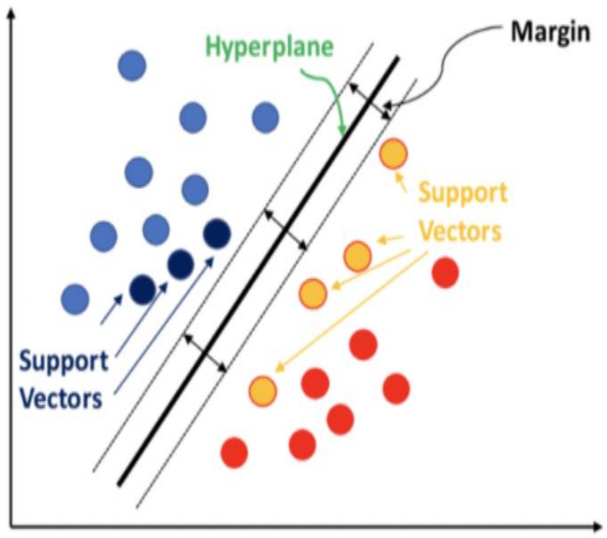

Fig. 5. Support Vector Machine

Here are the key aspects and steps involved in training an SVM model:

1. Hyperplane and Margin: In binary classification, an SVM aims to find a hyperplane that separates the data points of two classes with the maximum margin. The margin is the distance between the hyperplane and the nearest data points from each class. SVM seeks to maximize this margin, as a larger margin is associated with better generalization performance.
2. Support Vectors: Support vectors are the data points that lie closest to the decision boundary or hyperplane. These data points have the most influence on defining the hyperplane and are used to compute the margin. SVM focuses only on these support vectors during training, which makes it memory-efficient and effective even for high-dimensional data.
3. Kernel Trick: SVM can efficiently handle non-linearly separable data by using the kernel trick. The kernel function maps the input data into a higher-dimensional feature space, where it becomes linearly separable. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels. The choice of kernel depends on the problem at hand and the underlying data distribution.
4. Training and Optimization: SVM finds the optimal hyperplane by solving a constrained optimization problem. The goal is to maximize the margin while minimizing the classification error. The optimization involves solving a quadratic programming problem, which can be efficiently solved using various algorithms such as the Sequential Minimal Optimization (SMO) algorithm.

If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three. The goal of SVMs is to find the optimal hyperplane that separates the classes in the data with the largest possible margin.

4.3.5. Decision Tree Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

The components of a decision tree classifier include:

1. Root Node: The topmost node of the tree, which represents the entire dataset. It is the starting point for the decision-making process.

2. Internal Nodes: These nodes represent features or attributes in the dataset. Each internal node corresponds to a decision rule based on a specific feature. It splits the dataset into subsets based on the attribute values.

3. Branches: Branches emanate from internal nodes and represent the possible outcomes or values of the feature being tested. Each branch corresponds to a specific value of the attribute associated with the internal node.

4. Leaf Nodes: Also known as terminal nodes, these nodes represent the final outcomes or class labels. They do not have any further branches or splits. Each leaf node corresponds to a predicted class label or a decision outcome.

5. Splitting Criteria: The decision tree algorithm determines the best splitting criteria for each internal node. Common splitting criteria include Gini impurity and information gain (based on entropy). These criteria measure the homogeneity or impurity of the subsets created by the splits.

6. Pruning: Pruning is a technique used to reduce overfitting by removing unnecessary nodes or branches from the tree. It helps simplify the tree structure and improves its ability to generalize to unseen data.

7. Prediction Rules: The decision tree classifier uses the paths from the root node to the leaf nodes to make predictions. Each path corresponds to a set of decision rules based on the attribute values, leading to a final prediction at the leaf node.

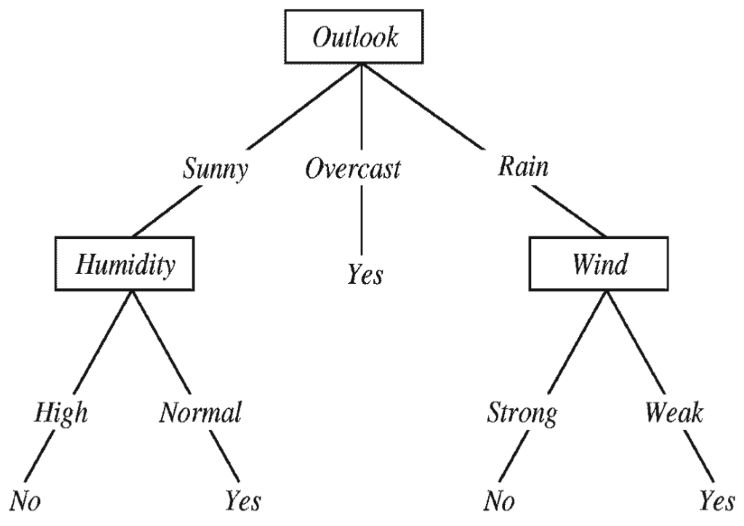

Fig. 6. Decision Tree Classifier

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

4.3.6. Naïve Bayes Classifer

A technique for creating the classifiers is called Naive Bayes. This classifier is used to supply and assign all of the class labels to the problem, which consists of feature values that are present as vectors and the class labels that are gathered from a collection of finite values. When using class variables and features that are independent of one another, there is only one way to train these classifiers.

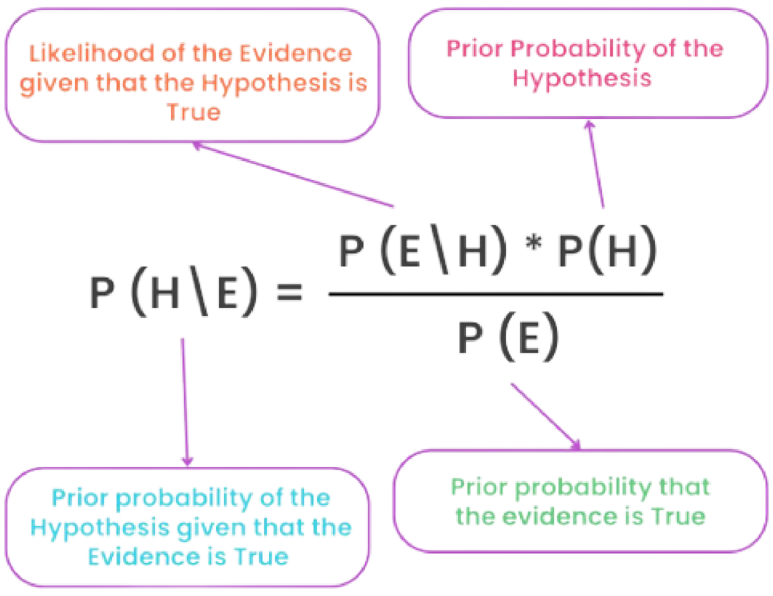
Likelihood of the Evidence given that the Hypothesis is True
Prior Probability of the Hypothesis
P (H\E) = P (E\H) * P(H) / P (E)
Prior probability of the Hypothesis given that the Evidence is True
Prior probability that the evidence is True

Fig. 7. Naïve Bayes Classifier

Here are the key components and concepts of the Naive Bayes classifier:

1. Bayes' Theorem: The Naive Bayes classifier is based on Bayes' theorem, which states that the probability of an event A given an event B can be calculated using the prior probability of A, the likelihood of B given A, and the overall likelihood of B. Mathematically, it is expressed as: P(A|B) = (P(B|A) * P(A)) / P(B).

2. Independence Assumption: The Naive Bayes classifier assumes that the features or attributes are conditionally independent given the class variable. This assumption simplifies the calculation of probabilities and makes the algorithm computationally efficient. Although this assumption may not hold in all cases, Naive Bayes often performs well in practice.

3. Training Phase: During the training phase, the Naive Bayes classifier calculates the prior probabilities and likelihoods from the training data. It estimates the prior probability of each class and the conditional probabilities of each feature given each class.

4. Prior Probabilities: The prior probability represents the probability of each class in the dataset. It is calculated as the proportion of instances belonging to each class in the training set.

5. Likelihoods: The likelihoods represent the conditional probability of each feature given each class. For categorical features, the likelihood is calculated as the proportion of instances in each class having a specific attribute value. For numerical features, the likelihood is often modeled using probability density functions such as Gaussian (Normal) distribution.

6. Posterior Probability: During the prediction phase, the Naive Bayes classifier calculates the posterior probability of each class given the input features. It uses Bayes' theorem to compute the probability of each class given the observed features.

7. Classification Rule: The Naive Bayes classifier assigns the class label that has the highest posterior probability as the predicted class for a given instance. This decision rule is known as the maximum a posteriori (MAP) rule.

Naive Bayes classifiers are easy to implement, require relatively few training examples, and can handle high-dimensional datasets. However, their performance may be impacted when the independence assumption is violated or when there are strong dependencies between features.

5. IMPLEMENTATION

5.1. Coding

Git link: https://github.com/19WH1A0594/Novel-stock-prediction-for-Indian-stock-market-using-Ensemble-techniques.git

5.2. Evaluation metrics

5.2.1. Root Mean Square Error

Root-Mean-Square-Error or RMSE is one of the most popular measures to estimate the accuracy of our forecasting model's predicted values versus the actual or observed values while training the regression models or time series models. It measures the error in our predicted values when the target or response variable is a continuous number.

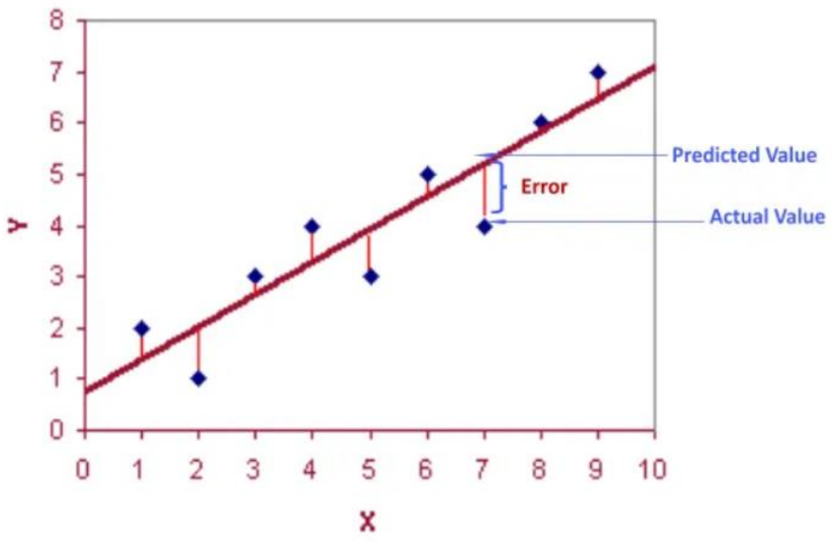

Fig. 8. Root Mean Squared Error

Thus, RMSE is a standard deviation of prediction errors or residuals. It indicates how spread out the data is around the line of best fit. It is also an essential criterion in shortlisting the best performing model among different forecasting models that you may have trained on one particular dataset. To do so, simply compare the RMSE values across all models and select the one with the lowest value on RMSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i - o_i)^2}$$

Where:

- ∑ is the summation of all values
- f is the predicted value
- is observed or actual value
- (fi — oi) 2 are the differences between predicted and observed values and squared
- N is the total sample size

5.3. Results

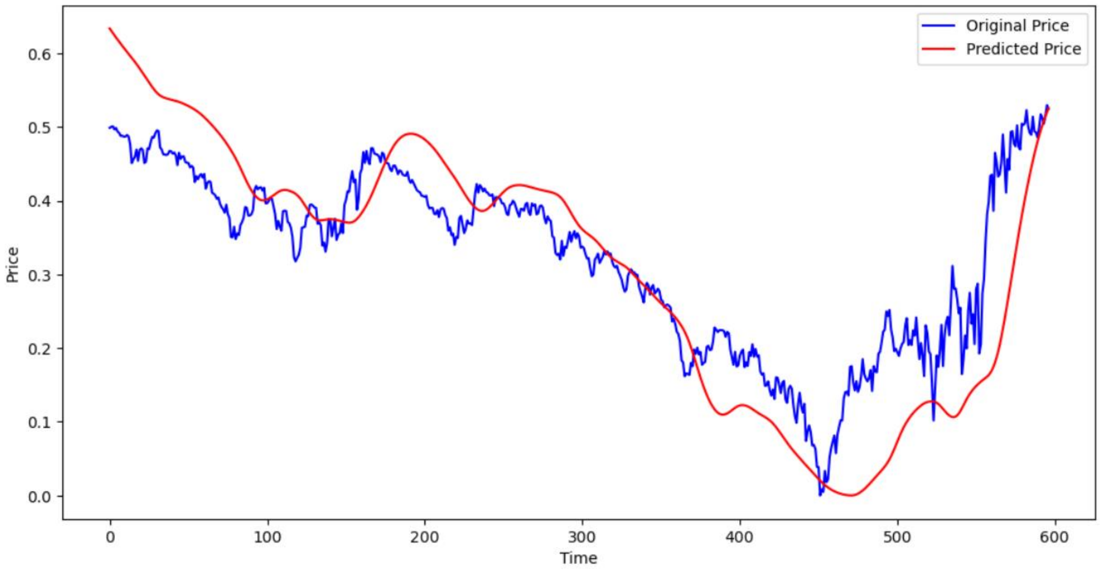

Fig. 9. Plotting the Actual and Predicted Prices

Fig. 6. Shows the graph plotting the original and predicted prices of the stock market.
We can see the offset which is the difference between the original and predicted prices since we are having less offset the prediction accuracy is good.

Table. 1. Comparison Table

Algorithms	**RMSE**
LSTM	0.33495
SVM	0.10055
Decision Tree Classifier	0.15851
Naïve Bayes Classifier	0.30899
Bagging Algorithm	0.10025

This is the comparison table which shows the root mean square error values of different algorithms used. Since the Bagging algorithm has less rmse value we can say that the model's prediction is good.

6. CONCLUSION AND FUTURE WORK

In this study, we proposed a novel stock prediction model for the Indian stock market using ensemble techniques that combine deep learning and machine learning approaches. We used an LSTM model to capture the complex patterns and dependencies in the time series data and applied ensemble techniques to improve the prediction accuracy. Transfer learning approaches have shown promising results in various deep learning applications. The proposed model could be extended to other financial markets beyond the Indian stock market.

In Future we can explore and incorporate additional features that can provide valuable information for stock prediction. This may include financial ratios, market sentiment indicators, news sentiment analysis, macroeconomic indicators, or technical indicators. Investigate the impact of different feature combinations on prediction performance.

7. REFERENCES

[1] P. Yu and X. Yan, ‘‘Stock price prediction based on deep neural networks,’’ Neural Comput. Appl., vol. 32, no. 6, pp. 1609–1628, Mar. 2020.

[2] W. Khan, M.A. Ghazanfar, M.A. Azam et al., "Stock market prediction using machine learning classifiers and social media news", J Ambient Intell Human Compute, 2020.

[3] S. Singh, K. S. Parmar, and J. Kumar, ‘‘Soft computing model coupled with statistical models to estimate future of stock market,’’ Neural Comput. Appl., pp. 1–19, 2021.

[4] Y. Trichilli, M. B. Abbes, and A. Masmoudi, ``Predicting the effect of Googling investor sentiment on Islamic stock market returns: A five-state hidden Markov model," Int. J. Islamic Middle Eastern Finance Manage, vol. 13, no. 2, pp. 165-193, Feb. 2020.

[5] Taylan Kabbani And Ekrem Duman, ” Deep Reinforcement Learning Approach for Trading Automation in the Stock Market,” Aug.2022.

[6] W. Wei, Q. Zhang, and L. Liu, ``Bitcoin transaction forecasting with deep network representation learning," IEEE Trans. Emerg. Topics Comput., early access, Jul. 20, 2020.

[7] B. Gasperov and Z. Kostanjcar, ``Market making with signals through deep reinforcement learning," IEEE Access, vol. 9, pp. 6161161622, 2021.

[8] Nagaraj Naik,Biju R. Mohan, “ Novel Stock Crisis Prediction Technique—A Study on Indian Stock Market ’’, 2021.

[9] Kun Huang, Xiaoming Li, Fangyuan Liu, Xiaoping Yang, and Wei Yu, “ML-GAT:A Multilevel Graph Attention Model for Stock Prediction”, Aug.2022.

[10] . I.-H. Cheng, S. Raina, and W. Xiong, ‘‘Wall street and the housing bubble,’’ Amer. Econ. Rev., vol. 104, no. 9, pp. 2797–2829, Sep. 2014.

[11] Jiusheng Chen and Xianning Wang, “Asymmetric Risk Spillovers Between China and ASEAN Stock Markets”, Oct.2021.

[12] Cheng Zhang, Nilam N. A. Sjarif, and Roslina B. Ibrahim, “Decision Fusion for Stock Market Prediction: A Systematic Review”, Jul.2022.

[13] Yaohu Lin, Shancun Liu, Haijun Yang, (Member, IEEE),Harris Wu,” Stock Trend Prediction Using Candlestick Charting and Ensemble Machine Learning Techniques With a Novelty Feature Engineering Scheme”, July 23, 2021.

[14] Ziyue li , Shiwei Lyu , Haipeng Zhang , (Member, IEEE), AND Tianpai Jiang, ‘‘One Step Ahead: A Framework for Detecting Unexpected Incidents and Predicting the Stock Markets’’, February 25, 2021.

[15] Liang Ding , Yi Luo, Yan Lin , Yirong Huang, ‘‘Optimal Setting for Hurst Index Estimation and Its Application in Chinese Stock Market’’, July 7, 2021.

[16] Teema Leangarun , (Graduate Student Member, IEEE), Poj Tangamchit , (Member, IEEE), Suttipong Thajchayapong , (Member, IEEE), ‘‘Stock Price Manipulation Detection Using Deep Unsupervised Learning: The Case of Thailand’’, August 5, 2021

[17] Yao-Hsin Chou , (Member, IEEE), Yun-Ting Lai , Yu-Chi Jiang 2 , Shu-Yu Kuo, “Using Trend Ratio and GNQTS to Assess Portfolio Performance in the U.S. Stock Market” , June 28, 2021.

[18] Mohammad Alsumi, “Reducing Manual Effort to Label Stock Market Data by Applying a Metaheuristic Search: A Case Study From the Saudi Stock Market “ , August 12, 2021.

[19]https://www.kaggle.com/datasets/aaron7sun/stocknews?select=upload_DJIA_table.csv

8. APPENDIX

IMPORTING DATASET

```
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.pyplot as plt

df= pd.read_csv('C:/Users/priyanka/Desktop/Major Project/Datasets/upload_DJIA_table.csv')
df.head()
df.isnull().sum()
```

DATA PREPROCESSING

```
# Calculate Z-scores
z_scores = np.abs(stats.zscore(df['Close']))
# Define threshold for outliers
threshold = 3
# Identify outliers
outliers = np.where(z_scores > threshold)[0]
# Remove outliers
df = df.drop(df.index[outliers])
# Plot the data without outliers
plt.plot(df['Close'])
df.tail()
df.size

df = df.drop(['Adj Close'], axis = 1)
plt.plot(df.Close)
```

MOVING AVERAGES

```
ma100 = df.Close.rolling(100).mean()
ma100
```

```
# moving average for this graph is finding out the average of the previous 100 day, it will get a value and is plotted in the graph.
```

TRAINING AND TESTING

```
# split the data into training and testing
# create the dataframe for training data
# dataframe consists of close column starting from 0th index we need to go to 70% of the total value
data_training = pd.DataFrame(df['Close'][0:int(len(df)*0.70)])
# create the dataframe for testing data
# dataframe consists of close column starting from 70% of the total value and goes till complete length
data_testing = pd.DataFrame(df['Close'][int(len(df)*0.70): int(len(df))])

# 70% is the training data and 30% is the testing data.
print(data_training.shape)
print(data_testing.shape)

# we will be scaling down the data between 0 and 1
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1)) # creating the object for minmaxscaler and define the feature range between 0 and 1

# fitting the training data into minmaxscaler and converted into an array, this fit_tranform will automatically generate the array
data_training_array = scaler.fit_transform(data_training)
data_training_array
data_training_array.shape

import numpy as np
x_train = []
y_train = []

for i in range(100, data_training_array.shape[0]):
    x_train.append(data_training_array[i-100: i])
    y_train.append(data_training_array[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

x_train.shape
```

LSTM MODEL

```
from keras.layers import Dense, Dropout, LSTM
from keras.models import Sequential
model = Sequential()
model.add(LSTM(units = 50, activation = 'relu', return_sequences = True, input_shape =
(x_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
model.add(Dropout(0.3))
model.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
model.add(Dropout(0.4))
model.add(LSTM(units = 120, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(units = 1))
model.summary()
model.compile(optimizer='adam', loss = 'mean_squared_error')
model.fit(x_train, y_train, epochs = 3)

past_100_days = data_training.tail(100)
final_df = past_100_days.append(data_testing, ignore_index=True)
final_df.head()
input_data = scaler.fit_transform(final_df)
input_data
input_data.shape
x_test = []
y_test = []
```

```
for i in range(100, input_data.shape[0]):
    x_test.append(input_data[i-100: i])
    y_test.append(input_data[i, 0]) # this 0 is closing price column

x_train, y_train = np.array(x_train), np.array(y_train)
x_test, y_test = np.array(x_test), np.array(y_test)
print(x_test.shape)
print(y_test.shape)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
y_predicted = model.predict(x_test)
y_predicted = scaler.fit_transform(y_predicted)
rmse = np.sqrt(np.mean(((y_predicted - y_test) ** 2)))
rmse
print("Accuracy of LSTM is :", 1.96 * rmse)
y_predicted.shape
y_test
y_predicted
scaler.scale_
scale_factor = 1/1.5785754
y_predicted = y_predicted * scale_factor
y_test = y_test * scale_factor
```

DIFFERENCE BETWEEN THE ACTUAL AND PREDICTED PRICES

```
plt.figure(figsize=(12,6))
plt.plot(y_test, 'b', label = 'Original Price')
plt.plot(y_predicted, 'r', label = 'Predicted Price')
plt.xlabel('Time')
plt.ylabel('Price')
plt.legend()
plt.show()
```

ENSEMBLE TECHNIQUES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
#from xgboost.xgbclassifier import XGBClassifier
#from xgboost import XGBClassifier
from sklearn import metrics

import warnings
warnings.filterwarnings('ignore')
# Machine learning
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
# For data manipulation
import pandas as pd
import numpy as np
# To plot
import matplotlib.pyplot as plt
plt.style.use('seaborn-darkgrid')
# To ignore warnings
import warnings
warnings.filterwarnings("ignore")

features = df[['open-close', 'low-high']]
target = df['target']

scaler = StandardScaler()
features = scaler.fit_transform(features)

x_train, x_test, y_train, y_test = train_test_split(
    features, target, test_size=0.1, random_state=5)
```

```
print(x_train.shape, x_test.shape)
```

SVM

```
from sklearn.svm import SVC
svm = SVC(random_state = 1)
svm.fit(x_train,y_train)
# y_pred = svm.predict(X_valid)
print("accuracy of svm is :",svm.score(x_test,y_test))
y_pred = svm.predict(x_test)
rmse = np.sqrt(np.mean((y_test-y_pred)**2))
print(rmse)
```

DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
#y_pred = dt.predict(X_test)
print('Accuracy of Decision Tree is:',dt.score(x_test,y_test))
y_pred = dt.predict(x_test)
rmse = np.sqrt(np.mean((y_test-y_pred)**2))
print(rmse)
```

NAÏVE BAYES CLASSIFIER

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train,y_train)
print('accuracy of Naive Bayes in test data is :', nb.score(x_test,y_test))
y_pred = nb.predict(x_test)
rmse = np.sqrt(np.mean((y_test-y_pred)**2))
print(rmse)

from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier

pipeline = make_pipeline(StandardScaler(),
                        LogisticRegression(random_state=1))
```

BAGGING CLASSIFIER

```
# Instantiate the bagging classifier
bgclassifier = BaggingClassifier(base_estimator=pipeline, n_estimators=10,
                                  max_features=0.9,
                                  max_samples=1000,
                                  random_state=20, n_jobs=10

                                 )
# Fit the bagging classifier
bgclassifier.fit(x_train, y_train)
print('Accuracy: %.3f ' %bgclassifier.score(x_test, y_test))
y_pred = bgclassifier.predict(x_test)
rmse = np.sqrt(np.mean((y_test-y_pred)**2))
print(rmse)
```