A project report on

STOCK PRICE PREDICTION USING MACHINE LEARNING - ENSEMBLE APPROACH

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering with specialization in AI and Robotics

by

Nuka Nandan Vignesh (21BRS1473)

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

November,2024

STOCK PRICE PREDICTION USING MACHINE LEARNING - ENSEMBLE APPROACH

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering with specialization in AI and Robotics

by

Nuka Nandan Vignesh (21BRS1473)

VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

November, 2024

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)
CHENNAI

DECLARATION

I hereby declare that the thesis entitled "Stock Price Prediction using Machine Learning - Ensemble Approach" submitted by Nuka Nandan Vignesh (21BRS1473), for the award of the degree of Bachelor of Technology in Computer Science and Engineering, Vellore Institute of Technology, Chennai is a record of bonafide work carried out by me under the supervision of Shenbaga Velu P.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai

Date: Signature of the Candidate

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled "**Stock Price Prediction using Machine Learning - Ensemble Approach**" is prepared and submitted by Nuka Nandan Vignesh (21BRS1473) to Vellore Institute of Technology, Chennai, in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Compuster Science and Engineering with speacialization in AI and Robotics** is a bonafide record carried out under my guidance. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:
Name: Dr. Shenbaga Velu P
Date:

Signature of the Examiner	Signature of the Examiner
Name:	Name:
Date:	Date:

Approved by the Head of Department,
(**Compuster Science and Engineering with speacialization in AI and Robotics**)
Name: **Dr. Harini**
Date:

ABSTRACT

In recent years, the stock market has witnessed a surge in participation from both individual and institutional investors. This growing interest in the financial markets has underscored the importance of accurate stock price prediction for informed decision-making and strategic investment planning. With the complexities involved in the stock market, which is influenced by numerous factors such as financial reports, market sentiment, and economic conditions, traditional forecasting methods have often fallen short in delivering reliable results. This has opened the door for advanced machine learning (ML) algorithms, which offer improved prediction accuracy by analyzing large volumes of historical data and real-time inputs. This research paper presents a comprehensive approach to stock price prediction by leveraging machine learning algorithms, including Long Short-Term Memory (LSTM), Artificial Neural Networks (ANN), Random Forest, and Support Vector Machines (SVM). The ensemble approach combines the strengths of each individual model, capturing different aspects of market behavior to provide a robust and well-rounded prediction. Through rigorous back-testing and evaluation, the ensemble model demonstrates improved predictive power compared to single algorithms, leading to more informed investment decisions.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Shenbaga Velu P, Assisstant Professor, School of Mechanical Engineering, Vellore Institute of Technology, Chennai, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Leaning.

It is with gratitude that I would like to extend my thanks to the visionary leader Dr. G. Viswanathan our Honorable Chancellor, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. G V Selvam Vice Presidents, Dr. Sandhya Pentareddy, Executive Director, Ms. Kadhambari S. Viswanathan, Assistant Vice-President, Dr. V. S. Kanchana Bhaaskaran Vice-Chancellor, Dr. T. Thyagarajan Pro-Vice Chancellor, VIT Chennai and Dr. P. K. Manoharan, Additional Registrar for providing an exceptional working environment and inspiring all of us during the tenure of the course.

Special mention to Dr. Ganesan R, Dean, Dr. Parvathi R, Associate Dean Academics, Dr. Geetha S, Associate Dean Research, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

In jubilant state, I express ingeniously my whole-hearted thanks to **Dr. Harini S**, Head of the Department, B.Tech. **Computer Science and Enginerring with specialization in AI and Robotics** and the Project Coordinators for their valuable support and encouragement to take up and complete the thesis.

My sincere thanks to all the faculties and staffs at Vellore Institute of Technology, Chennai who helped me acquire the requisite knowledge. I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who encouraged me to take up and complete this task.

Place: Chennai

Date: **Nuka Nandan Vignesh**

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Overview of Stock Market Predictions

The stock market is a dynamic environment influenced by various economic, political, and psychological factors. Accurate stock price prediction is a critical task for investors and traders as it helps in making informed decisions and minimizing risks. Traditional prediction methods often fall short due to their inability to handle complex, non-linear relationships present in stock market data.

This project leverages machine learning techniques, specifically Support Vector Machine (SVM), K-Nearest Neighbour (K-NN), and Decision Tree (DT), to create an ensemble model that enhances the accuracy and reliability of stock price predictions.

This project delves into a comparative analysis of SVM, KNN, and DT algorithms in the realm of stock price prediction. SVM, a versatile algorithm, is particularly effective in handling high-dimensional data and non-linear relationships. K-NN, on the other hand, leverages the concept of similarity to classify new data points based on their nearest neighbors. Decision Trees, a popular algorithm, can model complex decision-making processes and generate interpretable rules. By combining the strengths of these algorithms within an ensemble framework, the project aims to achieve superior predictive performance.

Ensemble learning, a powerful technique that combines multiple models to improve predictive performance, has gained significant attention in recent years. By harnessing the strengths of diverse algorithms, ensemble methods can overcome the limitations of individual models. In the context of stock price prediction, ensemble techniques can effectively capture complex patterns, reduce over-fitting, and improve the overall accuracy of forecasts.

1.2 Importance of Machine Learning in Financial Forecasting

Machine learning (ML) has emerged as a powerful tool in financial forecasting. Unlike conventional statistical methods, ML algorithms can analyze large volumes of historical stock data to uncover hidden patterns and trends. They are capable of learning from past data and adapting to changing market conditions, making them indispensable for predicting stock prices in volatile markets.

This project highlights the potential of ML-based ensemble approaches in combining the strengths of individual algorithms to achieve superior predictive performance.

Ensemble learning is a technique that combines multiple models to improve predictive performance. By leveraging the diversity of individual models, ensemble methods can mitigate the limitations of any single model and achieve more accurate and robust predictions. In the context of stock price prediction, ensemble techniques, such as bagging, boosting, and stacking, can significantly enhance the forecasting accuracy.

The findings of this project have significant implications for investors and traders. By providing more accurate and reliable stock price predictions, these models can help investors make informed decisions, optimize investment portfolios, and mitigate risks. Moreover, the insights gained from the analysis of stock market data can contribute to a deeper understanding of market dynamics and behavior.

1.3 Objectives of the Project

The objective of this project is to develop a robust predictive model for stock price forecasting by utilizing multiple machine learning algorithms—namely, Support Vector Machine(SVM), K-Nearest Neighbour(KNN) and Decision Tree. Each of these algorithms is employed to capture different facets of stock price dynamics. The core goal is to integrate these models into an ensemble approach that leverages their combined predictive power to achieve higher accuracy and reliability. The ensemble model will be rigorously tested and backtested against historical data to ensure that it provides a significant improvement over individual models in terms of prediction accuracy.

This model aims to provide investors and traders with a more accurate and reliable tool for making informed decisions in volatile markets.

The primary objectives of this project are:

a. To develop a predictive model for stock price forecasting using machine learning algorithms.
b. To evaluate the performance of individual algorithms (SVM, K-NN, and Decision Tree) in predicting stock prices.

c. To design and implement an ensemble approach that integrates the predictions of individual algorithms for improved accuracy and reliability.
d. To visualize and compare the results of the proposed ensemble model against actual stock prices.

1.4 Problem Statement

Stock price prediction remains one of the most challenging tasks in the financial sector due to the complexity and unpredictability of financial markets. The primary issue is the inherent volatility of stock prices, driven by a wide range of factors, including economic indicators, market sentiment, and global events. This volatility leads to non-linear and chaotic patterns in price movements, which are difficult to capture using traditional statistical models.

Additionally, the availability of vast amounts of historical data poses the problem of effectively processing and analyzing this data to extract meaningful patterns that can be used for accurate predictions. Capturing Short-Term and Long-Term Trends, Reducing Prediction Errors, Evaluating Model Performance

Predicting stock prices is a challenging task due to the high volatility and noise in stock market data. Existing models often struggle with issues such as overfitting, limited generalization capability, and difficulty in capturing complex relationships within the data.

This project addresses these challenges by:

a. Utilizing robust machine learning algorithms to capture data patterns effectively.
b. Combining the strengths of multiple models through ensemble techniques to mitigate the limitations of individual algorithms.
c. Demonstrating the practicality of the proposed approach through experiments and performance evaluations.

CHAPTER 2: LITERATURE REVIEW

2.1 Existing Methods for Stock Price Prediction

Stock price prediction has been a subject of extensive research, with various methods employed to forecast future price movements. Traditional approaches, such as technical analysis and fundamental analysis, have been used for decades. Technical analysis involves studying historical price and volume data to identify patterns and trends, while fundamental analysis focuses on evaluating a company's financial health and industry prospects.

In recent years, machine learning and artificial intelligence techniques have emerged as powerful tools for stock price prediction. These methods utilize complex algorithms to analyze large datasets and identify patterns that may not be apparent to human analysts. Popular machine learning techniques include time series analysis (Box and Jenkins, 1976), support vector machines (SVM) (Vapnik, 1995), random forests (Breiman, 2001), and neural networks (LeCun et al., 2015). Time series analysis models, such as ARIMA and GARCH, are effective in capturing the temporal dependencies in stock price data. SVM and random forests are capable of handling both numerical and categorical data, while neural networks, particularly recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, are well-suited for modeling sequential data like stock prices.

While these methods have shown promising results, stock price prediction remains a challenging task due to the inherent volatility and complexity of financial markets. Factors such as economic indicators, news events, and investor sentiment can significantly impact stock prices, making accurate predictions difficult. Researchers continue to explore innovative approaches, such as incorporating social media sentiment analysis and natural language processing, to improve the accuracy and reliability of stock price forecasts.

Machine learning offers a wide range of algorithms that can be applied to stock price prediction. In recent years (2015-2024), researchers have explored the performance of various algorithms, including traditional methods like linear regression and time series analysis, as well as more advanced techniques like support vector machines (SVM), random forests, and neural networks.

Linear regression models are simple yet effective for capturing linear relationships between features and the target variable. However, they may struggle to capture complex patterns in stock price data. Time series analysis models, such as ARIMA and GARCH, are specifically designed to model time-dependent data. While they can capture temporal dependencies, they may not be as flexible as other methods in handling non-linear relationships. SVM, random forests, and neural networks are more powerful and versatile algorithms. SVM can effectively handle high-dimensional data and non-linear relationships, while random forests are robust to noise and outliers. Neural networks, particularly deep learning models like recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, have shown impressive performance in capturing complex patterns in sequential data.

As technology continues to advance, we can expect further innovations in the field of stock price prediction. The integration of advanced techniques like natural language processing and sentiment analysis can provide valuable insights into market sentiment and news events. Additionally, the development of more powerful and efficient machine learning algorithms will enable more accurate and reliable predictions. By leveraging the power of machine learning and artificial intelligence, we can unlock the potential to make informed investment decisions and navigate the complexities of the financial markets.

2.2 Role of Ensemble Techniques in Improving Prediction Accuracy

Ensemble methods have gained significant popularity in recent years (2015-2024) for improving the accuracy and robustness of predictive models. These techniques combine multiple base models, such as decision trees, neural networks, or support vector machines, to create a more powerful ensemble model. By leveraging the diversity among individual models, ensemble methods can reduce overfitting, improve generalization, and enhance predictive performance.

Popular ensemble techniques include bagging, boosting, and stacking. Bagging, short for bootstrap aggregating, involves training multiple models on different subsets of the training data, and then averaging their predictions. Boosting, on the other hand, sequentially trains models, with each subsequent model focusing on correcting the errors of the previous ones. Stacking, also known as stacked generalization, combines the predictions of multiple base models using a meta-model. By effectively combining the strengths of different models, ensemble techniques have been shown to achieve state-of-the-art results in various domains, including stock price prediction.

Ensemble methods have emerged as a powerful tool in machine learning, particularly in the domain of stock price prediction. These techniques combine multiple base models, such as decision trees, neural networks, or support vector machines, to create a more robust and accurate ensemble model. By leveraging the diversity among individual models, ensemble methods can reduce overfitting, improve generalization, and enhance predictive performance.

Ensemble techniques offer several advantages in the context of stock price prediction:

- **Improved Accuracy:** By combining multiple models, ensemble methods can achieve higher accuracy than individual models. This is particularly important in the volatile and noisy stock market, where accurate predictions can be challenging.
- **Reduced Overfitting:** Ensemble methods can help reduce overfitting, a common problem in machine learning. By averaging the predictions of multiple models, the risk of overfitting is mitigated.
- **Enhanced Generalization:** Ensemble methods can improve the generalization ability of models, making them more robust to changes in market conditions.
- **Handling Complex Patterns:** Ensemble techniques can capture complex patterns in stock price data that may be difficult to identify with individual models.

2.3 Comparison of Machine Learning Algorithms for Prediction

Machine learning offers a wide range of algorithms that can be applied to stock price prediction. In recent years (2015-2024), researchers have explored the performance of various algorithms, including traditional methods like linear regression and time series analysis, as well as more advanced techniques like support vector machines (SVM), random forests, and neural networks.

Linear regression models are simple yet effective for capturing linear relationships between features and the target variable. However, they may struggle to capture complex patterns in stock price data. Time series analysis models, such as ARIMA and GARCH, are specifically designed to model time-dependent data. While they can capture temporal dependencies, they may not be as flexible as other methods in handling non-linear relationships. SVM, random forests, and neural networks are more powerful and versatile algorithms. SVM can effectively handle high-dimensional data and non-linear relationships, while random forests are robust to noise and outliers. Neural networks, particularly deep learning models like recurrent neural networks (RNNs) and long short-term memory (LSTM) networks,

Support Vector Machine (SVM):

a. Effective in high-dimensional spaces and works well with both linear and non-linear data.
b. Sensitive to the choice of kernel and hyperparameters.

K-Nearest Neighbour (KNN):

a. Simple and effective for small datasets.
b. Weakness: Computationally expensive for large datasets and sensitive to noise.

Decision Tree (DT):

a. Easy to interpret and works well with categorical and continuous data.
b. Prone to over-fitting without pruning techniques.

CHAPTER 3: METHODOLOGY

3.1 Dataset Description

3.1.1 Source of Data

The dataset for this project was obtained using the yfinance library, which provides historical stock price data directly from Yahoo Finance. The dataset includes daily stock price information such as Open, High, Low, Close, and Volume for a specified stock over a specific time period.

3.1.2 Features Used

a. Close: The final price of the stock at the end of the trading day.

b. Simple Moving Average (SMA): Averages the stock prices over a rolling window, providing trend information.

$$SMA = \frac{A}{n}$$

A - Sum of Closing prices over n days

c. Exponential Moving Average (EMA): Similar to SMA but gives more importance to recent data points.

d. Upper Band and Lower Band (Bollinger Bands): Indicators of volatility based on SMA and standard deviation.

$$\text{Upper Band} = \text{SMA} + (2 \times \text{Standard Deviation})$$

$$\text{Lower Band} = \text{SMA} - (2 \times \text{Standard Deviation})$$

e. Relative Strength Index (RSI): Measures momentum in the stock's price movements to determine overbought or oversold conditions.

$$RSI = 100 - \frac{100}{1 + \frac{AG}{AL}}$$

AG - Average Gain

AL - Average Loss

f. Average True Range (ATR): Measures the volatility of the stock by averaging the range between high and low prices over a period.

g. Moving Average Convergence Divergence (MACD): A momentum indicator derived from the difference between two EMAs, highlighting trend reversals.

3.2 Data Preprocessing and Cleaning

Data preprocessing is a crucial step in the data mining process, involving cleaning, transforming, and integrating data to make it suitable for analysis. This process ensures the quality and reliability of the data, leading to more accurate and meaningful insights.

One of the primary steps in data preprocessing is cleaning the data. This involves identifying and correcting errors, inconsistencies, and missing values. Common cleaning techniques include:

- Handling Missing Values: Missing values can be imputed using various methods such as mean, median, mode imputation, or more advanced techniques like interpolation or regression.
- Outlier Detection and Treatment: Outliers, which are data points that deviate significantly from the rest of the data, can be identified and handled using techniques like z-score or IQR methods. Outliers can be removed, capped, or winsorized.
- Data Consistency and Standardization: Ensuring data consistency involves checking for inconsistencies in data formats, units, and encoding. Standardization techniques, such as normalization or scaling, can be applied to bring data to a common scale.

Feature engineering involves creating new features from existing ones to improve model performance. This process can significantly enhance the predictive power of machine learning models. Some common feature engineering techniques include:

- Feature Creation: Deriving new features from existing ones, such as creating interaction terms or polynomial features.
- Feature Selection: Identifying the most relevant features to reduce dimensionality and improve model performance.
- Feature Scaling: Scaling features to a common range to ensure that all features contribute equally to the model.

Data integration involves combining data from multiple sources into a unified dataset. This process can be challenging, as data from different sources may have different formats, structures, and quality. Data integration techniques, such as merging, joining, and concatenating, can be used to combine data from various sources into a coherent dataset.

By effectively performing data preprocessing and cleaning, we can ensure the quality and reliability of our data, leading to more accurate and insightful analysis.

3.3 Overview of Algorithms

3.3.1 Support Vector Machine (SVM)

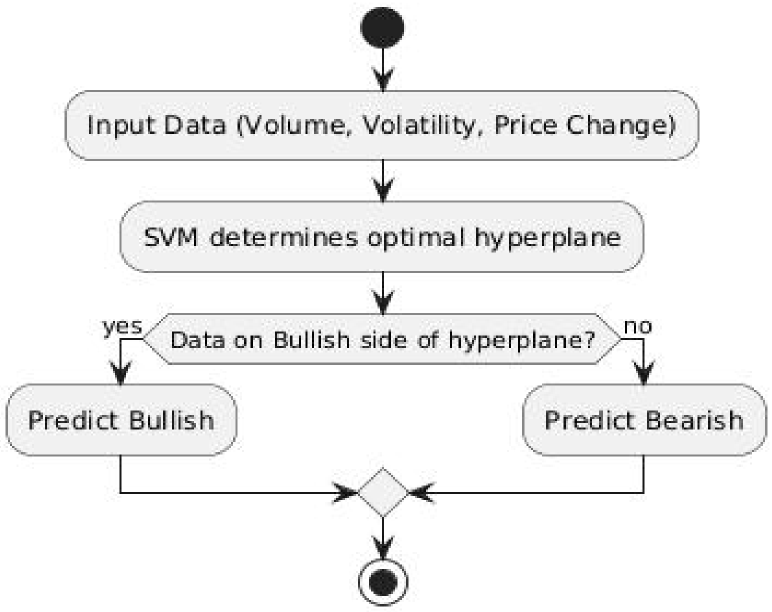
SVM is a supervised learning algorithm that finds an optimal hyperplane to separate data into classes. In stock prediction, SVM can classify historical stock price data to identify different market conditions, such as "bullish" or "bearish." SVM works well in complex financial data, especially when applying kernel functions to handle non-linearities.

Support Vector Machines (SVM) is a versatile machine learning algorithm that can be effectively applied to stock price prediction. It operates by identifying an optimal hyperplane that separates data points into different classes. In the context of stock price prediction, SVM can be used to classify historical stock price data into categories such as "buy," "sell," or "hold."

A key advantage of SVM is its ability to handle high-dimensional data and complex relationships between variables. This is particularly important in financial markets, where numerous factors, such as economic indicators, news sentiment, and market volatility, can influence stock prices.

To handle non-linear relationships, SVM employs kernel functions, which map the input data into a higher-dimensional space. This transformation allows the algorithm to find non-linear decision boundaries, making it well-suited for capturing the complex patterns in stock price data.

By leveraging the power of SVM and kernel functions, we can develop accurate and robust models for stock price prediction. However, it's important to note that SVM requires careful parameter tuning and feature engineering to achieve optimal performance.


Input Data (Volume, Volatility, Price Change)
SVM determines optimal hyperplane
Data on Bullish side of hyperplane?
yes
no
Predict Bullish
Predict Bearish

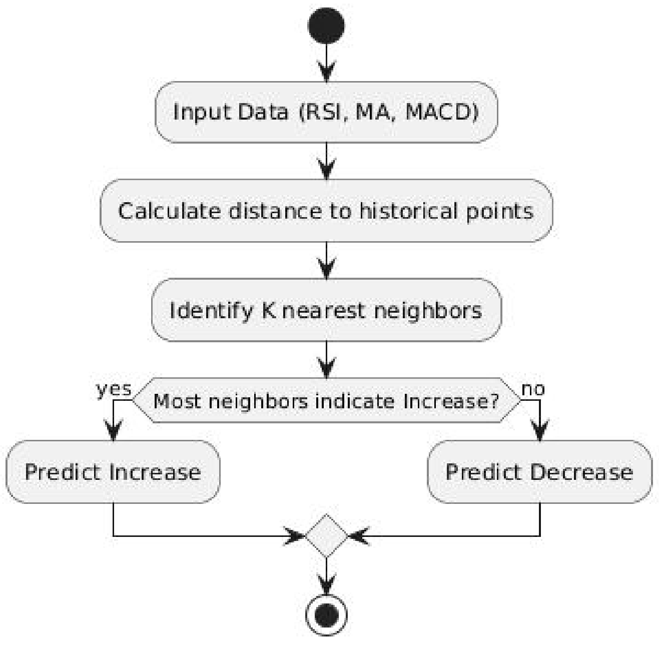
3.3.2 K-Nearest Neighbour (KNN)

KNN is a simple, non-parametric algorithm that classifies data points based on the "k" nearest data points in feature space. In stock price prediction, KNN finds the closest historical data points to the current stock data and assumes that the price trend of the closest matches will reflect the future movement. KNN works effectively for pattern recognition and can help detect trends based on previous, similar behavior.

K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm that classifies data points based on their similarity to neighboring data points. In the context of stock price prediction, KNN can be used to identify the closest historical data points to a given current stock price and its associated features.

The algorithm assumes that the future price movement of a stock will be similar to the price movement of its nearest neighbors. By analyzing the historical data, KNN can identify patterns and trends that may indicate future price movements.

However, KNN has some limitations, such as the curse of dimensionality, where its performance can degrade as the number of features increases. Additionally, KNN can be computationally expensive, especially for large datasets. To address these limitations, techniques like feature selection and dimensionality reduction can be employed.


Input Data (RSI, MA, MACD)
Calculate distance to historical points
Identify K nearest neighbors
Most neighbors indicate Increase?
yes
no
Predict Increase
Predict Decrease

3.3.3 Decision Tree

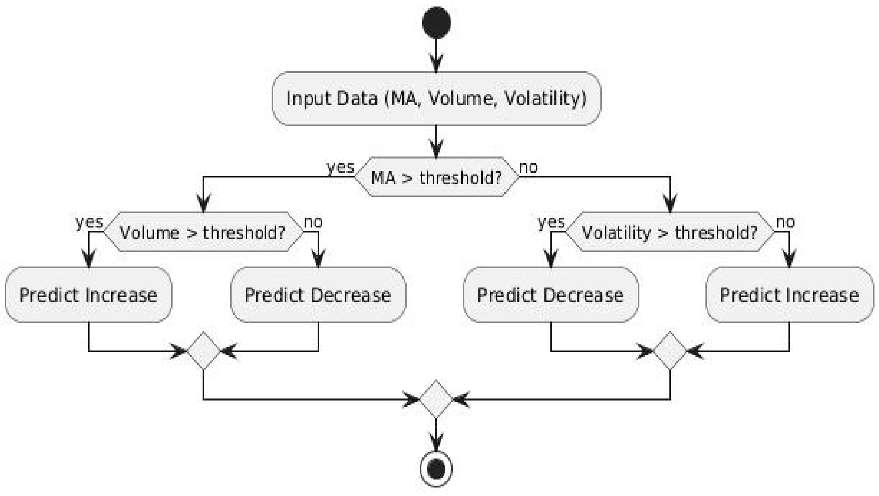
Decision Trees are a versatile machine learning algorithm that can be effectively applied to stock price prediction. They work by creating a tree-like model of decisions and their potential consequences. Each internal node represents a test on an attribute, and each branch represents an outcome of that test. The leaf nodes represent the final classification or prediction.

In the context of stock price prediction, decision trees can be used to classify historical stock data into different categories, such as "buy," "sell," or "hold." By analyzing various features like moving averages, relative strength index (RSI), and volume, the decision tree can identify patterns and trends that may indicate future price movements.

One of the key advantages of decision trees is their interpretability. The tree structure provides a clear and intuitive visualization of the decision-making process. This makes it easier to understand the factors that influence the model's predictions and identify potential biases.

However, decision trees can be prone to overfitting, especially when they become too complex. To mitigate this issue, techniques like pruning can be used to remove unnecessary branches and simplify the model. Additionally, ensemble methods, such as random forest and gradient boosting, can be used to combine multiple decision trees and improve overall performance.

By leveraging the power of decision trees, we can develop effective models for stock price prediction, enabling investors to make informed decisions and potentially achieve significant returns.



3.4 Ensemble Approach Framework

3.4.1 Concept of Ensemble Modeling

Ensemble modeling combines predictions from multiple machine learning algorithms to produce a more robust and accurate result. The idea is to leverage the strengths of each algorithm to minimize errors.
In stock price prediction, an ensemble approach involves combining predictions from multiple machine learning models to get a final prediction. Instead of relying on a single algorithm, which may have its weaknesses, ensemble methods use the strengths of several algorithms to improve accuracy and robustness. This approach can capture different aspects of the data and provide a more reliable forecast of stock prices.

Ensemble methods have emerged as a powerful tool in machine learning, particularly in the domain of stock price prediction. These techniques combine multiple base models, such as decision trees, neural networks, or support vector machines, to create a more robust and accurate ensemble model. By leveraging the diversity among individual models, ensemble methods can reduce overfitting, improve generalization, and enhance predictive performance.

3.4.2 Combining Predictions

For instance, in our project, simple and weighted averaging allow for efficient, fast predictions when speed is crucial, while gradient boosting enables a deeper learning process that can capture complex patterns in the data. This flexibility makes ensemble methods versatile and effective in producing high-quality results.

By integrating predictions from multiple models, we can capture not only immediate stock trends but also underlying factors that a single algorithm might miss, providing a robust, holistic view of future stock prices. This adaptability and effectiveness underscore the significance of ensemble approaches in delivering reliable forecasts, helping investors make informed decisions with reduced risk.

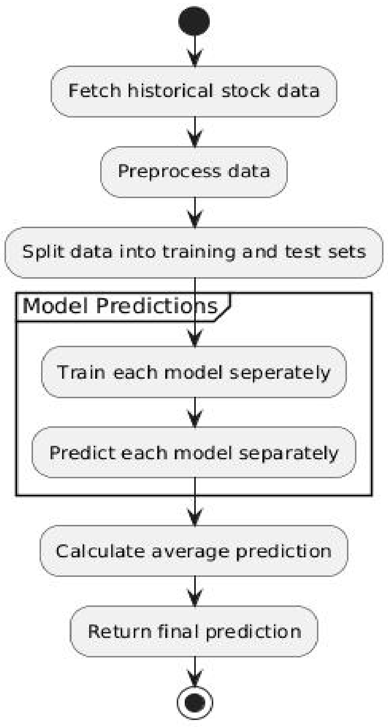
Three ensemble approaches were implemented:

In simple averaging, predictions from multiple models (like SVM, Decision Tree, and KNN) are averaged equally to form a final prediction.

- Historical stock data is fetched and preprocessed.
- The data is split into training and test sets.
- Multiple models (e.g., Linear Regression, Decision Tree, and KNN) are trained on the training set.
- Each model independently makes a prediction on the test set.
- The final prediction is the average of the individual model predictions.

By integrating predictions from multiple models, we can capture not only immediate stock trends but also underlying factors that a single algorithm might miss, providing a robust, holistic view of future stock prices. This adaptability and effectiveness underscore the significance of ensemble approaches in delivering reliable forecasts, helping investors make informed decisions with reduced risk. These techniques combine multiple base models, such as decision trees, neural networks, or support vector machines, to create a more robust and accurate ensemble model.

They work by creating a tree-like model of decisions and their potential consequences. Each internal node represents a test on an attribute, and each branch represents an outcome of that test.

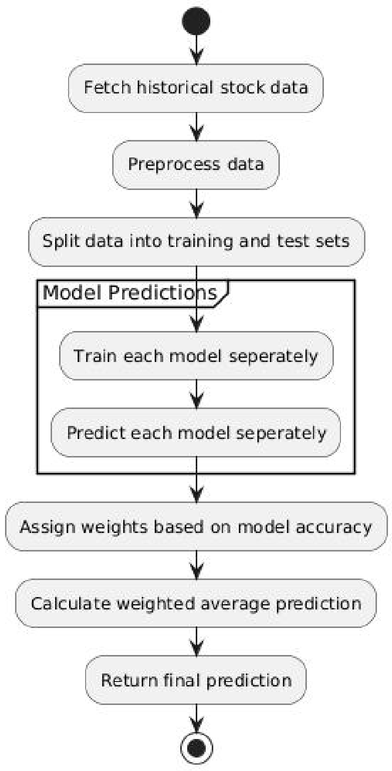

Fetch historical stock data
Preprocess data
Split data into training and test sets
Model Predictions
Train each model seperately
Predict each model separately
Calculate average prediction
Return final prediction

Each model's prediction is equally weighted and averaged to produce the final result.

Weighted averaging assigns different weights to each model's prediction based on the model's performance or importance, which can yield more accurate results than simple averaging.

- Historical data is fetched and preprocessed.
- Data is split into training and test sets.
- The Linear Regression, Decision Tree, and KNN models are trained.
- Each model generates predictions, which are then weighted based on their performance.
- The weighted predictions are combined to produce a final prediction, accounting for the relative accuracy of each model.

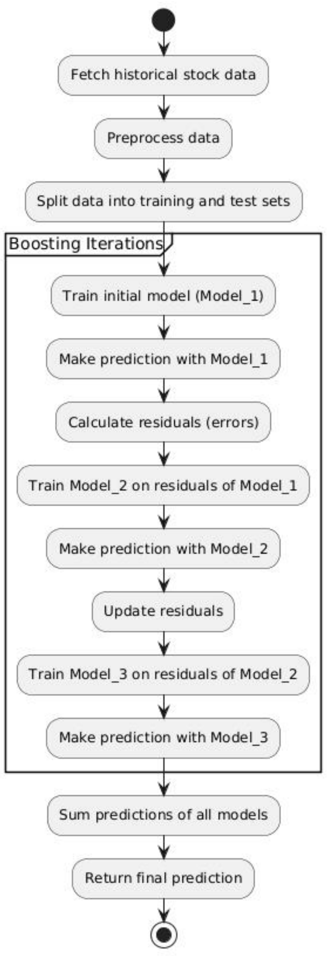
This adaptability and effectiveness underscore the significance of ensemble approaches in delivering reliable forecasts, helping investors make informed decisions with reduced risk. These techniques combine multiple base models, such as decision trees, neural networks, or support vector machines, to create a more robust and accurate ensemble model.


Fetch historical stock data
Preprocess data
Split data into training and test sets
Model Predictions
Train each model seperately
Predict each model seperately
Assign weights based on model accuracy
Calculate weighted average prediction
Return final prediction

Weighted Averaging: Assigns weights to individual models based on their performance, giving more importance to the best-performing algorithm.

Gradient boosting builds an ensemble by training models sequentially, where each new model corrects the errors of the previous ones, improving the final prediction iteratively.

- Data is fetched, preprocessed, and split.
- A base model (Model_1) is trained to make an initial prediction.
- The residual errors from Model_1's predictions are calculated.
- A new model (Model_2) is trained on these residuals to correct the errors from Model_1.
- This process is repeated with additional models to iteratively improve the prediction.
- The final prediction is the cumulative result of all models, creating an accurate, fine-tuned prediction.


Fetch historical stock data
Preprocess data
Split data into training and test sets
Boosting Iterations
Train initial model (Model_1)
Make prediction with Model_1
Calculate residuals (errors)
Train Model_2 on residuals of Model_1
Make prediction with Model_2
Update residuals
Train Model_3 on residuals of Model_2
Make prediction with Model_3
Sum predictions of all models
Return final prediction

A sequential method that improves prediction accuracy by minimizing residual errors iteratively.

3.5 Evaluation Metrics

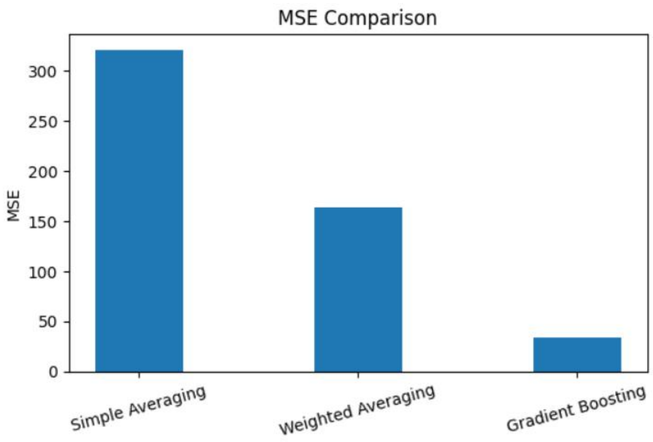
In the realm of stock price prediction, evaluation metrics play a crucial role in assessing the accuracy and reliability of our models. They help us understand how well our predictions align with the actual market movements.

The evaluation metrics are like scorecards that tell us about how the model is working and according to it we can make the assumptions.

3.5.1 Mean Squared Error (MSE)

Measures the average squared difference between the actual and predicted values, emphasizing larger errors more than smaller ones.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

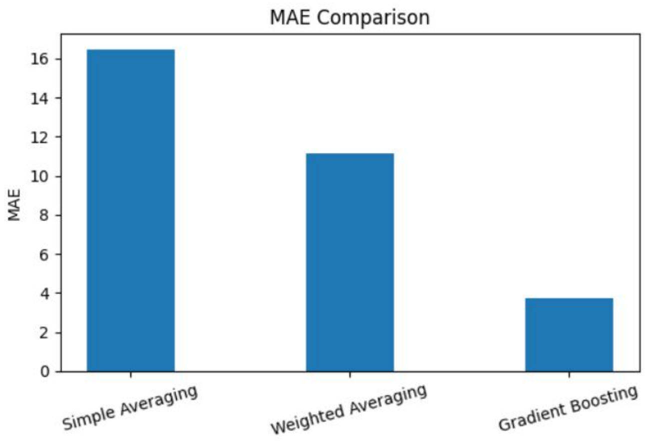


3.5.2 Mean Absolute Error (MAE)

Measures the average magnitude of errors in the predictions without considering their direction. MAE provides a straightforward interpretation of error in the same unit as the target variable.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

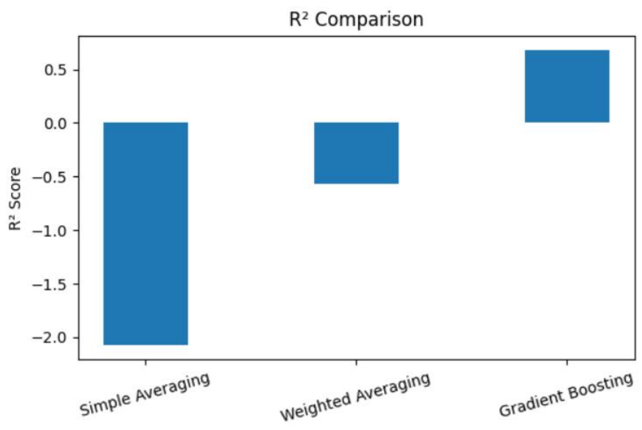
MAE is particularly useful for understanding the average error magnitude, making it less sensitive to large outliers compared to MSE.



3.5.3 R-squared Score

Represents the proportion of variance in the dependent variable that is explained by the model. It ranges from 0 to 1, where values closer to 1 indicate a better fit.

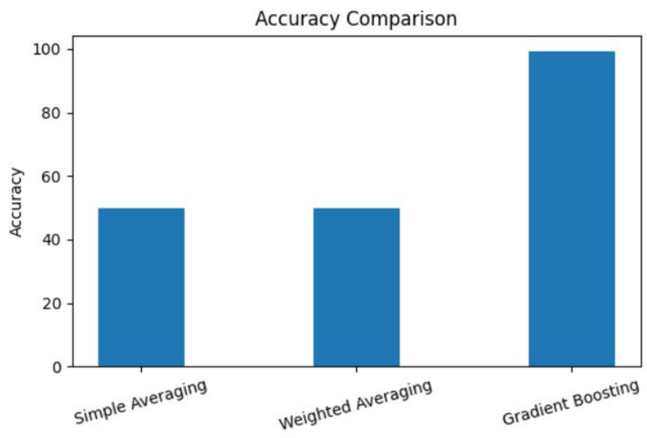
$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$



3.5.4 Accuracy

Represents how well the model predicts the stock prices compared to the actual values. In regression problems, accuracy is often evaluated indirectly through metrics like MSE, MAE, and R2R^2R2. For classification scenarios, accuracy is expressed as the percentage of correctly predicted instances.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100$$

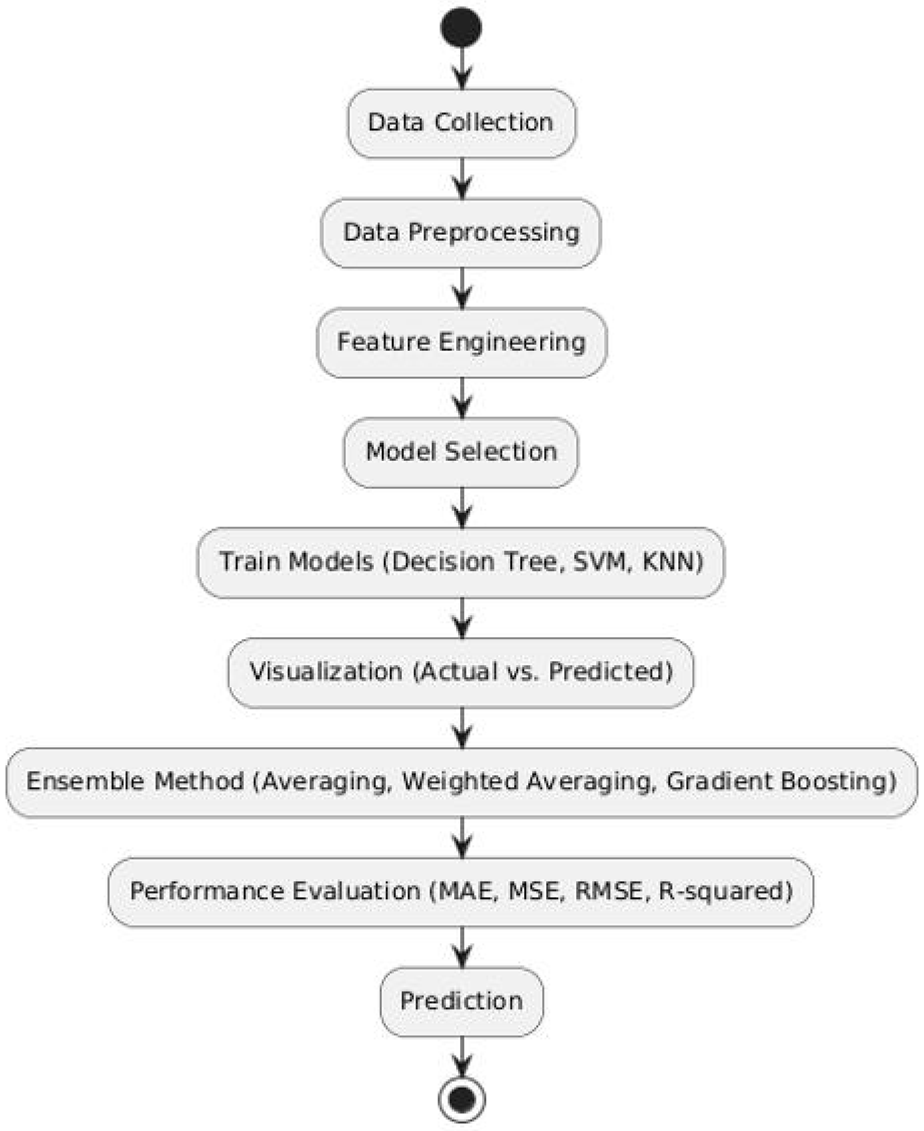


3.6 Flowchart

Data Collection:

- Gather historical stock price data from reliable sources like Yahoo Finance.
- Collect relevant financial and economic indicators that may influence stock prices.
- The start date and end date can be set by user.

.



Data Collection
Data Preprocessing
Feature Engineering
Model Selection
Train Models (Decision Tree, SVM, KNN)
Visualization (Actual vs. Predicted)
Ensemble Method (Averaging, Weighted Averaging, Gradient Boosting)
Performance Evaluation (MAE, MSE, RMSE, R-squared)
Prediction

Data Preprocessing:

- Clean the data by handling missing values, outliers, and inconsistencies.
- Normalize the data to a common scale.
- Split the data into training and testing sets.

Feature Engineering:

- Technical indicators (SMA, EMA, UpperBand, LowerBand, RSI, ATR, MACD)

Model Development:

- Train Decision Tree, SVM, and KNN models on the preprocessed data.

Ensemble Approach:

- Averaging: Calculate the average prediction of the individual models.
- Weighted Averaging: Assign weights to each model based on their performance and calculate the weighted average prediction.

- Gradient Boosting: Sequentially train weak learners to correct the errors of previous models.

Model Evaluation:

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- R-squared
- Accuracy

Model Optimization:

- Fine-tune the hyperparameters of the individual models and ensemble methods using techniques like grid search or random search.
- Experiment with different feature combinations and model architectures to improve performance.

Model Deployment:

- Deploy the best-performing model to make real-time predictions.
- Predict Next day stock price.

CHAPTER 4: IMPLEMENTATION

4.1 Software Tools and Libraries

The primary programming language used for data processing is **Python.** The development and implementation of the stock price prediction model leveraged several powerful software tools and libraries, each tailored for specific aspects of the project. **Python** served as the primary programming language due to its versatility and extensive ecosystem of libraries for data analysis, machine learning, and visualization. It enabled seamless integration of data preprocessing, feature engineering, model development, and evaluation processes, making it an ideal choice for this project.

For data handling and analysis, **Pandas** and **NumPy** were extensively used. Pandas provided robust tools for manipulating tabular data, allowing efficient handling of the stock market dataset and the derived technical indicators like SMA and RSI. NumPy, with its optimized array computations, facilitated numerical operations that were crucial during normalization and feature engineering. **yfinance**, a dedicated library for fetching stock market data, was invaluable in sourcing historical stock prices directly from Yahoo Finance.

Machine learning tasks were performed using **Scikit-learn**, a comprehensive library for implementing algorithms such as Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Decision Tree, and ensemble methods like Gradient Boosting. Additionally, visualization libraries like **Matplotlib** and **Seaborn** were employed to create insightful plots for predicted vs. actual stock prices and feature trends. These tools, combined, provided a robust foundation for end-to-end development and analysis in this project.

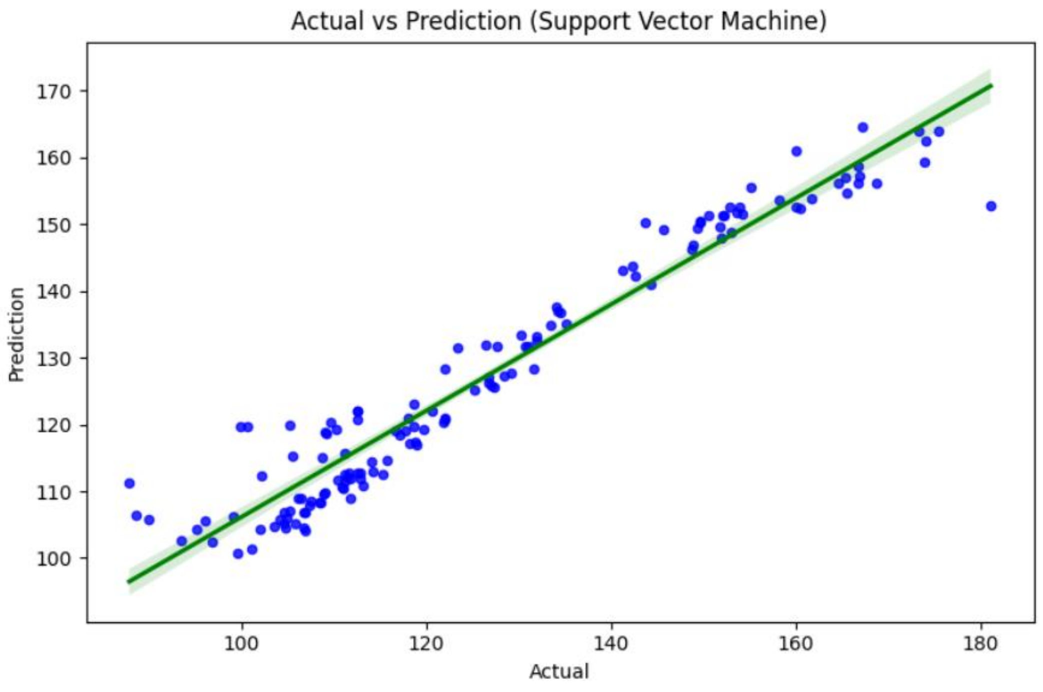
- Import necessary Libraries:
- Get Data from yfinance
- Data Preprocessing
- Creating Features
- Train and Split the Data

4.2. Implementation of Algorithms

4.2.1 Support Vector Machine (SVM)

The SVM regression model was implemented using Scikit-learn's SVR class. Key hyperparameters, such as the kernel type (linear, polynomial, or RBF), were tuned to optimize performance.

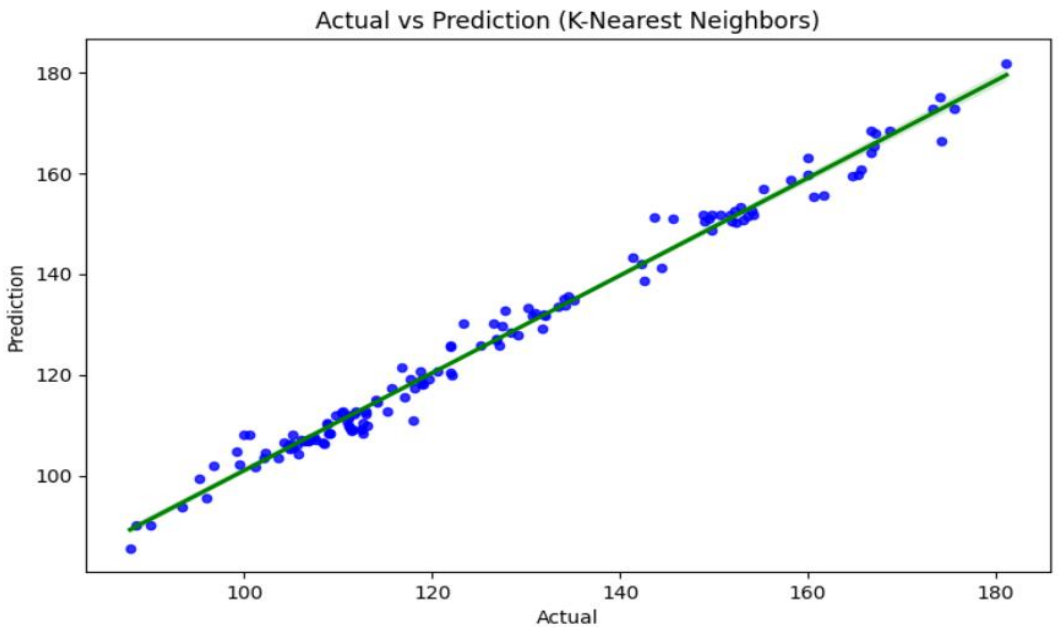
```
from sklearn.svm import SVR
svm_model = SVR(kernel='rbf', C=1, epsilon=0.1)
svm_model.fit(X_train, y_train)
svm_predictions = svm_model.predict(X_test)
```



4.2.2 K-Nearest Neighbour (KNN)

The KNN model was implemented using Scikit-learn's KNeighborsRegressor. The number of neighbors (k) was selected using cross-validation to minimize prediction error.

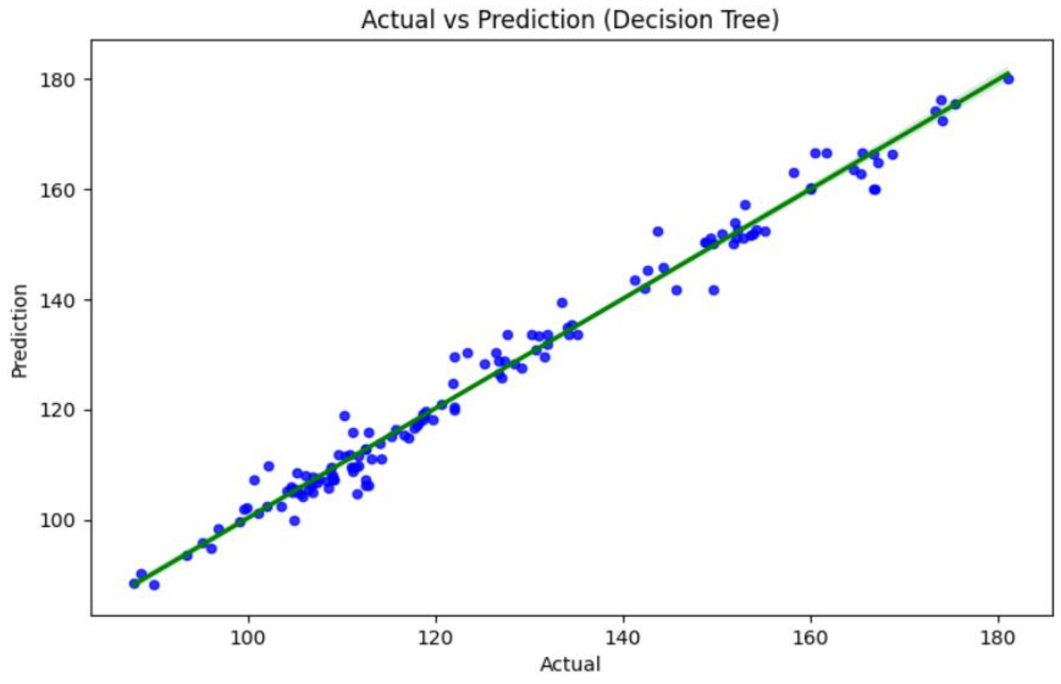
```
from sklearn.neighbors import KNeighborsRegressor
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)
knn_predictions = knn_model.predict(X_test)
```



4.2.3 Decision Tree

A Decision Tree regression model was implemented using Scikit-learn's DecisionTreeRegressor. Parameters like maximum depth and minimum samples per leaf were adjusted to avoid overfitting.

```
from sklearn.tree import DecisionTreeRegressor
dt_model = DecisionTreeRegressor(max_depth=10)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
```



4.3 Building the Ensemble Model

4.3.1 Averaging

Predictions from SVM, KNN, and Decision Tree were averaged to produce the final prediction.

final_predictions_avg = (svm_predictions + knn_predictions + dt_predictions) / 3

4.3.2 Weighted Averaging

Weights were assigned based on the performance of individual models during training.

weights = [0.4, 0.3, 0.3]   # Example weights

*final_predictions_weighted = (weights[0] * svm_predictions +*
*weights[1] * knn_predictions +*
*weights[2] * dt_predictions)*

4.3.3 Gradient Boosting

Gradient Boosting was implemented using Scikit-learn's GradientBoostingRegressor, combining sequential models to correct residual errors.

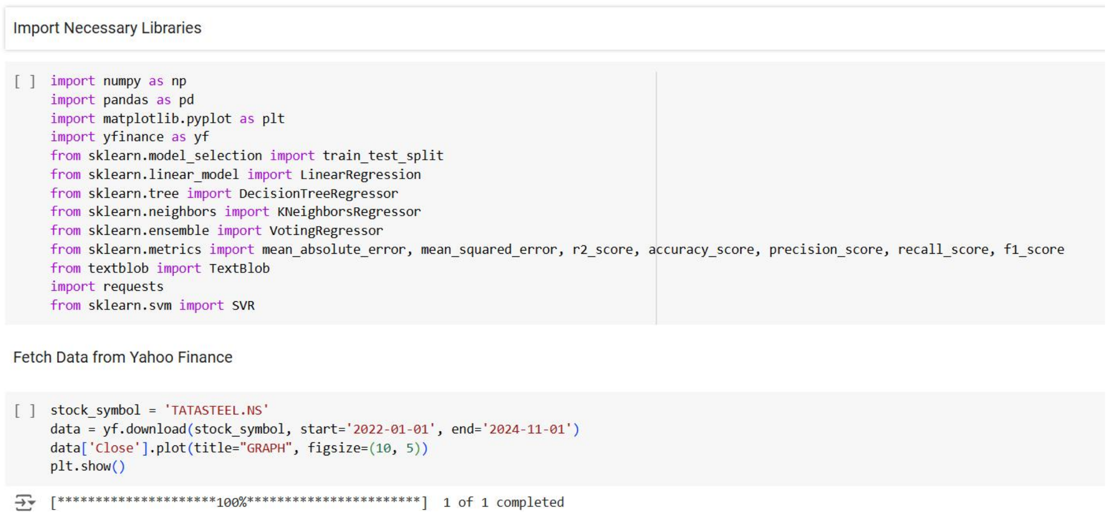
from sklearn.ensemble import GradientBoostingRegressor

gb_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3)

gb_model.fit(X_train, y_train)

final_predictions_gb = gb_model.predict(X_test)

4.4 Code Snippets



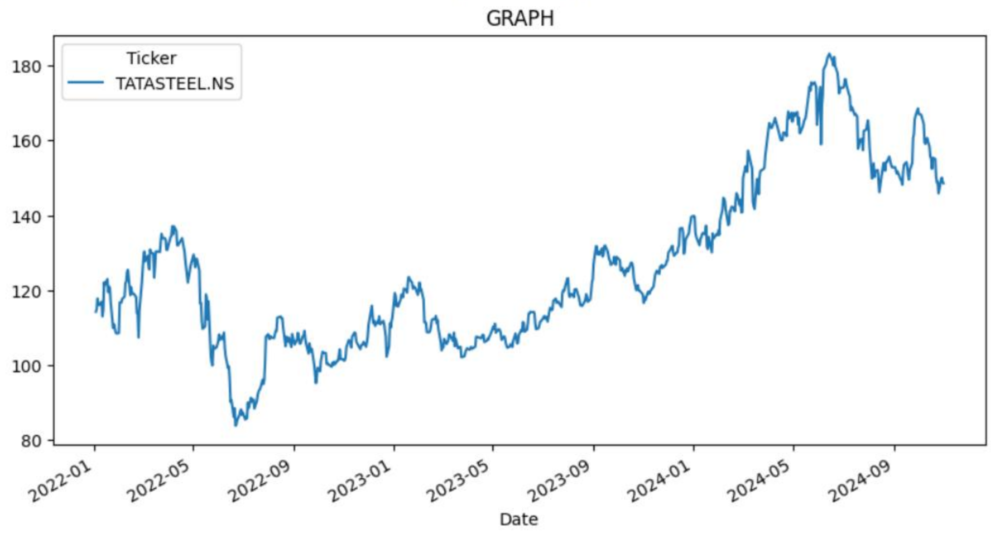
Import Necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import VotingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, accuracy_score, precision_score, recall_score, f1_score
from textblob import TextBlob
import requests
from sklearn.svm import SVR
```

Fetch Data from Yahoo Finance

```
stock_symbol = 'TATASTEEL.NS'
data = yf.download(stock_symbol, start='2022-01-01', end='2024-11-01')
data['Close'].plot(title="GRAPH", figsize=(10, 5))
plt.show()
```

```
[*********************100%***********************]  1 of 1 completed
```

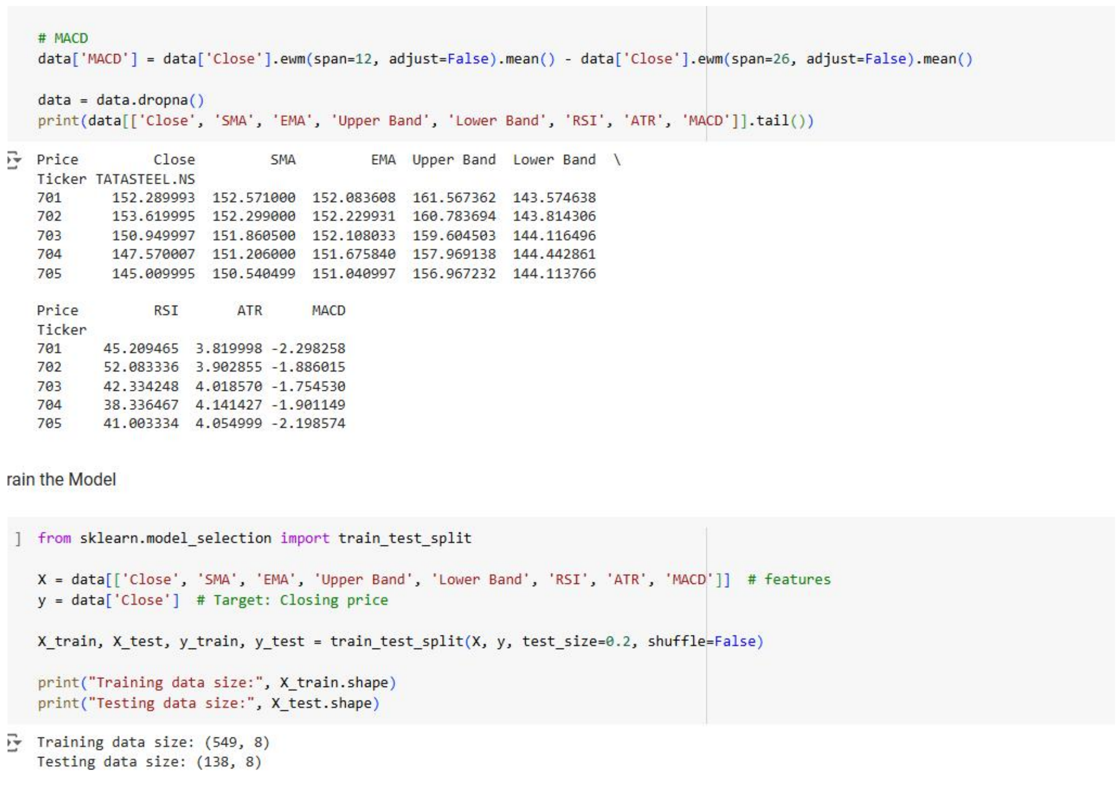


Data Preprocessing

```
[ ] data.reset_index(inplace=True)
    data['Target'] = data['Close'].shift(-1)  # Shift to create the target variable
    data.dropna(inplace=True)
```

Creating multiple features

```
[ ] # Moving Averages
    data['SMA'] = data['Close'].rolling(window=20).mean()
    data['EMA'] = data['Close'].ewm(span=20, adjust=False).mean()

    # Bollinger Bands
    data['20 Day STD'] = data['Close'].rolling(window=20).std()
    data['Upper Band'] = data['SMA'] + (data['20 Day STD'] * 2)
    data['Lower Band'] = data['SMA'] - (data['20 Day STD'] * 2)

    # RSI
    delta = data['Close'].diff(1)
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)
    avg_gain = gain.rolling(window=14).mean()
    avg_loss = loss.rolling(window=14).mean()
    rs = avg_gain / avg_loss
    data['RSI'] = 100 - (100 / (1 + rs))

    # ATR
    data['TR'] = abs(data['High'] - data['Low'])
    data['ATR'] = data['TR'].rolling(window=14).mean()

    # MACD
    data['MACD'] = data['Close'].ewm(span=12, adjust=False).mean() - data['Close'].ewm(span=26, adjust=False).mean()
```



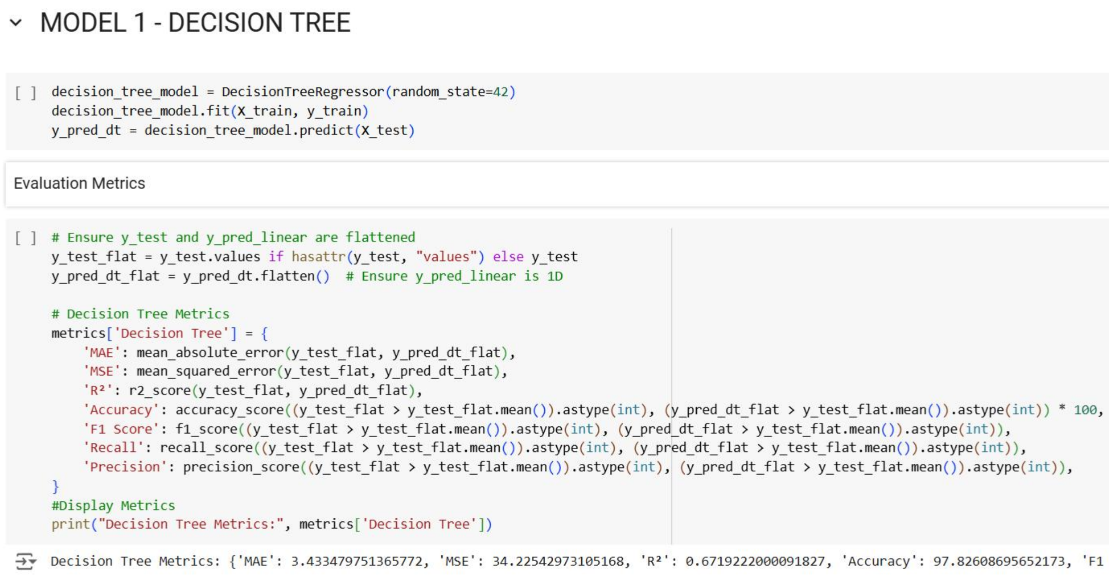
```
# MACD
data['MACD'] = data['Close'].ewm(span=12, adjust=False).mean() - data['Close'].ewm(span=26, adjust=False).mean()

data = data.dropna()
print(data[['Close', 'SMA', 'EMA', 'Upper Band', 'Lower Band', 'RSI', 'ATR', 'MACD']].tail())
```

```
Price          Close         SMA        EMA  Upper Band  Lower Band  \
Ticker TATASTEEL.NS
701       152.289993  152.571000  152.083608  161.567362  143.574638
702       153.619995  152.299000  152.229931  160.783694  143.814306
703       150.949997  151.860500  152.108033  159.604503  144.116496
704       147.570007  151.206000  151.675840  157.969138  144.442861
705       145.009995  150.540499  151.040997  156.967232  144.113766

Price         RSI       ATR      MACD
Ticker
701     45.209465  3.819998 -2.298258
702     52.083336  3.902855 -1.886015
703     42.334248  4.018570 -1.754530
704     38.336467  4.141427 -1.901149
705     41.003334  4.054999 -2.198574
```

rain the Model

```
]  from sklearn.model_selection import train_test_split

   X = data[['Close', 'SMA', 'EMA', 'Upper Band', 'Lower Band', 'RSI', 'ATR', 'MACD']]  # features
   y = data['Close']  # Target: Closing price

   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

   print("Training data size:", X_train.shape)
   print("Testing data size:", X_test.shape)
```

```
Training data size: (549, 8)
Testing data size: (138, 8)
```



MODEL 1 - DECISION TREE

```
[ ]  decision_tree_model = DecisionTreeRegressor(random_state=42)
     decision_tree_model.fit(X_train, y_train)
     y_pred_dt = decision_tree_model.predict(X_test)
```

Evaluation Metrics

```
[ ]  # Ensure y_test and y_pred_linear are flattened
     y_test_flat = y_test.values if hasattr(y_test, "values") else y_test
     y_pred_dt_flat = y_pred_dt.flatten()  # Ensure y_pred_linear is 1D

     # Decision Tree Metrics
     metrics['Decision Tree'] = {
         'MAE': mean_absolute_error(y_test_flat, y_pred_dt_flat),
         'MSE': mean_squared_error(y_test_flat, y_pred_dt_flat),
         'R²': r2_score(y_test_flat, y_pred_dt_flat),
         'Accuracy': accuracy_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_dt_flat > y_test_flat.mean()).astype(int)) * 100,
         'F1 Score': f1_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_dt_flat > y_test_flat.mean()).astype(int)),
         'Recall': recall_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_dt_flat > y_test_flat.mean()).astype(int)),
         'Precision': precision_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_dt_flat > y_test_flat.mean()).astype(int)),
     }
     #Display Metrics
     print("Decision Tree Metrics:", metrics['Decision Tree'])
```

```
Decision Tree Metrics: {'MAE': 3.433479751365772, 'MSE': 34.22542973105168, 'R²': 0.6719222000091827, 'Accuracy': 97.82608695652173, 'F1
```

MODEL 2 - K-Nearest Neighbour

```
[ ] knn_model = KNeighborsRegressor(n_neighbors=10)
    knn_model.fit(X_train, y_train)
    y_pred_knn = knn_model.predict(X_test)
```

Evaluation Metrics

```
# Ensure y_test and y_pred_linear are flattened
y_test_flat = y_test.values if hasattr(y_test, "values") else y_test
y_pred_knn_flat = y_pred_knn.flatten()  # Ensure y_pred_linear is 1D
threshold = 0.75

# KNN Metrics
metrics['KNN'] = {                          (variable) y_pred_knn_flat: Any | ndarray[Any, dtype[float64]]
    'MAE': mean_absolute_error(y_test_flat, y_pred_knn_flat),
    'MSE': mean_squared_error(y_test_flat, y_pred_knn_flat),
    'R²': r2_score(y_test_flat, y_pred_knn_flat),
    'Accuracy': accuracy_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_knn_flat > y_test_flat.mean()).astype(int)) * 100,
}
#Display Metrics
print("KNN Metrics:", metrics['KNN'])
```

```
KNN Metrics: {'MAE': 5.323152027959411, 'MSE': 57.53597891991592, 'R²': 0.44847215848868693, 'Accuracy': 73.91304347826086}
```

MODEL 3 - SUPPORT VECTOR MACHINE

```
svm_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=0.1)
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWarning: A column-vector y was passed when a 1d array
  y = column_or_1d(y, warn=True)
```

Evaluation Metrics

```
[ ] # Ensure y_test and y_pred_linear are flattened
    y_test_flat = y_test.values if hasattr(y_test, "values") else y_test
    y_pred_svm_flat = y_pred_svm.flatten()  # Ensure y_pred_linear is 1D

    # SVM Metrics
    metrics['SVM'] = {
        'MAE': mean_absolute_error(y_test_flat, y_pred_svm_flat),
        'MSE': mean_squared_error(y_test_flat, y_pred_svm_flat),
        'R²': r2_score(y_test_flat, y_pred_svm_flat),
        'Accuracy': accuracy_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_svm_flat > y_test_flat.mean()).astype(int)) * 100,
    }
    #Display Metrics
    print("SVM Metrics:", metrics['SVM'])
```

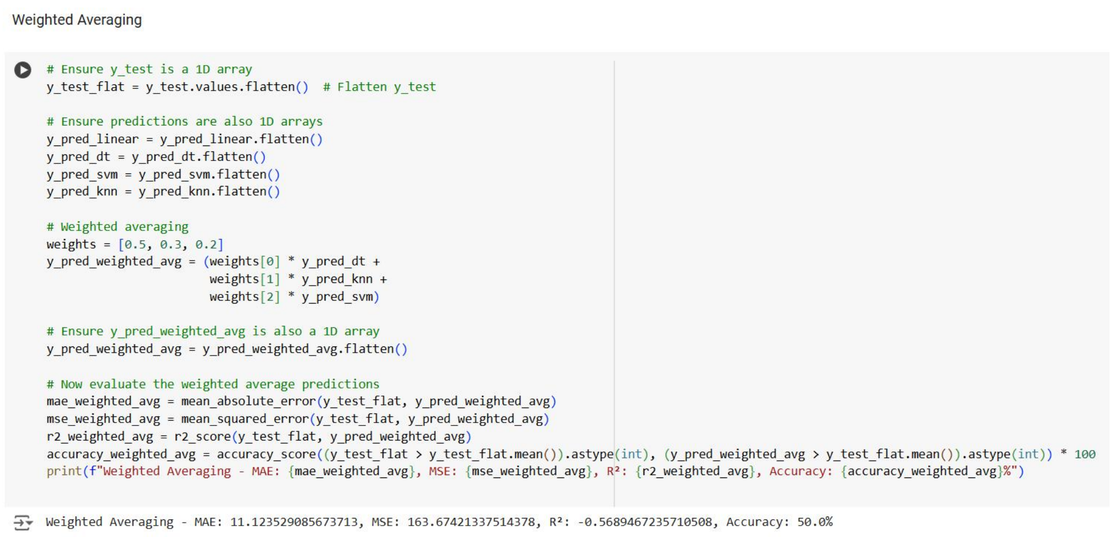
```
SVM Metrics: {'MAE': 42.18813114879568, 'MSE': 1894.5237411800529, 'R²': -17.160507725424115, 'Accuracy': 50.0}
```

Ensemble Method

Simple Averaging

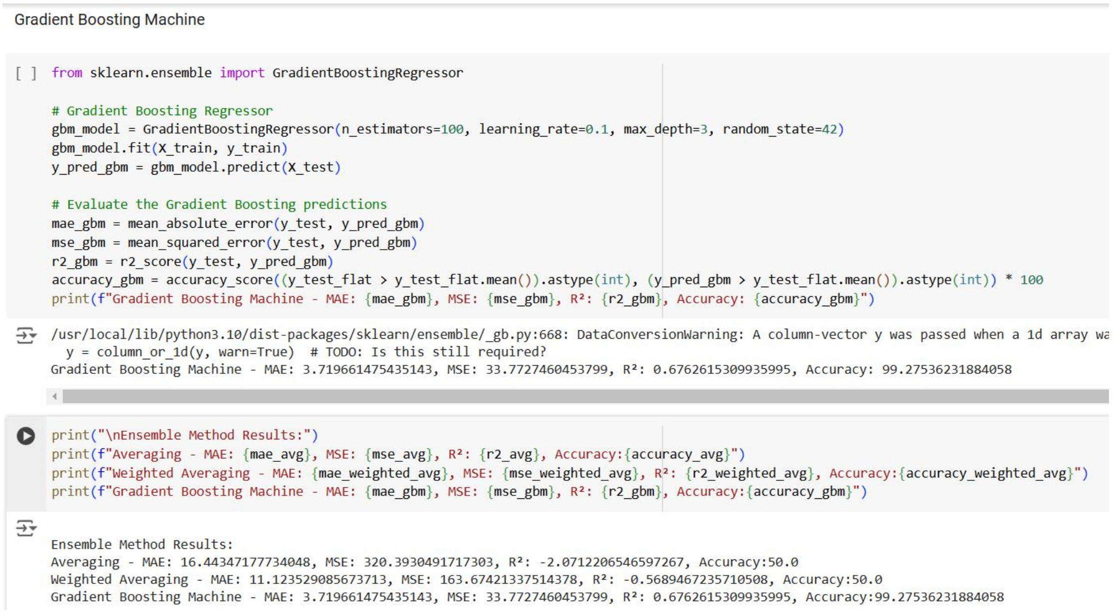
```
y_pred_avg = (y_pred_knn_flat + y_pred_dt_flat + y_pred_svm_flat) / 3

y_test_flat = y_test.values if hasattr(y_test, "values") else y_test
y_pred_avg_flat = y_pred_avg.flatten()  # Ensure y_pred_linear is 1D

# Evaluate the averaged predictions
mae_avg = mean_absolute_error(y_test_flat, y_pred_avg_flat)
mse_avg = mean_squared_error(y_test_flat, y_pred_avg_flat)
r2_avg = r2_score(y_test_flat, y_pred_avg_flat)
accuracy_avg = accuracy_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_avg_flat > y_test_flat.mean()).astype(int)) * 100
print(f"Averaging - MAE: {mae_avg}, MSE: {mse_avg}, R²: {r2_avg}, Accuracy: {accuracy_avg}%")
```

```
Averaging - MAE: 16.44347177734048, MSE: 320.3930491717303, R²: -2.0712206546597267, Accuracy: 50.0%
```


Weighted Averaging

```python
# Ensure y_test is a 1D array
y_test_flat = y_test.values.flatten()  # Flatten y_test

# Ensure predictions are also 1D arrays
y_pred_linear = y_pred_linear.flatten()
y_pred_dt = y_pred_dt.flatten()
y_pred_svm = y_pred_svm.flatten()
y_pred_knn = y_pred_knn.flatten()

# Weighted averaging
weights = [0.5, 0.3, 0.2]
y_pred_weighted_avg = (weights[0] * y_pred_dt +
                       weights[1] * y_pred_knn +
                       weights[2] * y_pred_svm)

# Ensure y_pred_weighted_avg is also a 1D array
y_pred_weighted_avg = y_pred_weighted_avg.flatten()

# Now evaluate the weighted average predictions
mae_weighted_avg = mean_absolute_error(y_test_flat, y_pred_weighted_avg)
mse_weighted_avg = mean_squared_error(y_test_flat, y_pred_weighted_avg)
r2_weighted_avg = r2_score(y_test_flat, y_pred_weighted_avg)
accuracy_weighted_avg = accuracy_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_weighted_avg > y_test_flat.mean()).astype(int)) * 100
print(f"Weighted Averaging - MAE: {mae_weighted_avg}, MSE: {mse_weighted_avg}, R²: {r2_weighted_avg}, Accuracy: {accuracy_weighted_avg}%")
```

```
Weighted Averaging - MAE: 11.123529085673713, MSE: 163.67421337514378, R²: -0.5689467235710508, Accuracy: 50.0%
```

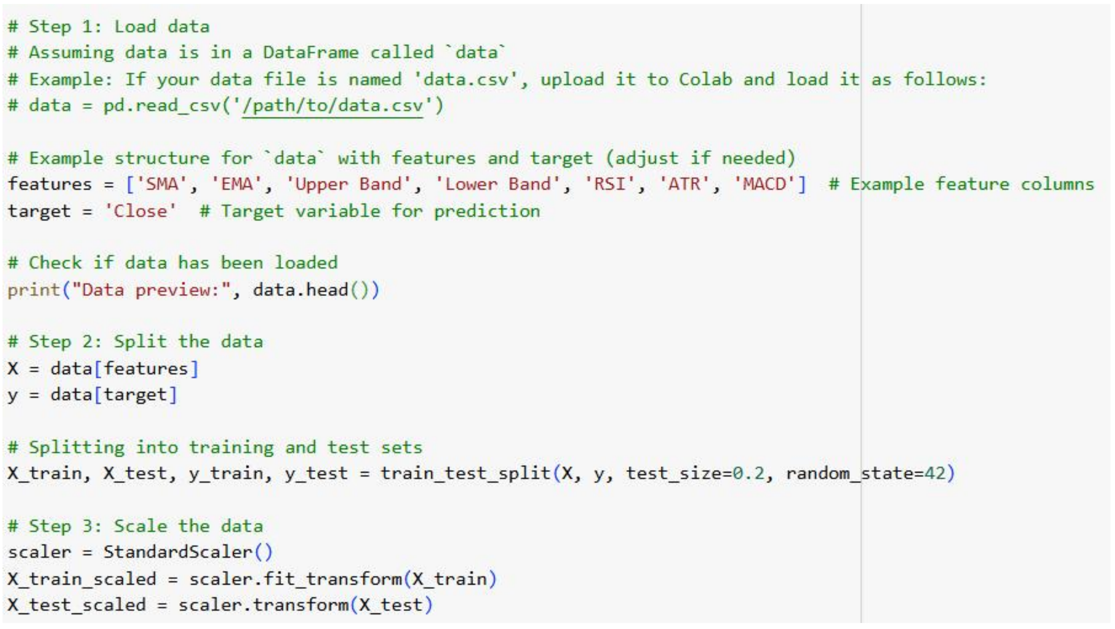

Gradient Boosting Machine

```python
from sklearn.ensemble import GradientBoostingRegressor

# Gradient Boosting Regressor
gbm_model = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
gbm_model.fit(X_train, y_train)
y_pred_gbm = gbm_model.predict(X_test)

# Evaluate the Gradient Boosting predictions
mae_gbm = mean_absolute_error(y_test, y_pred_gbm)
mse_gbm = mean_squared_error(y_test, y_pred_gbm)
r2_gbm = r2_score(y_test, y_pred_gbm)
accuracy_gbm = accuracy_score((y_test_flat > y_test_flat.mean()).astype(int), (y_pred_gbm > y_test_flat.mean()).astype(int)) * 100
print(f"Gradient Boosting Machine - MAE: {mae_gbm}, MSE: {mse_gbm}, R²: {r2_gbm}, Accuracy: {accuracy_gbm}")
```

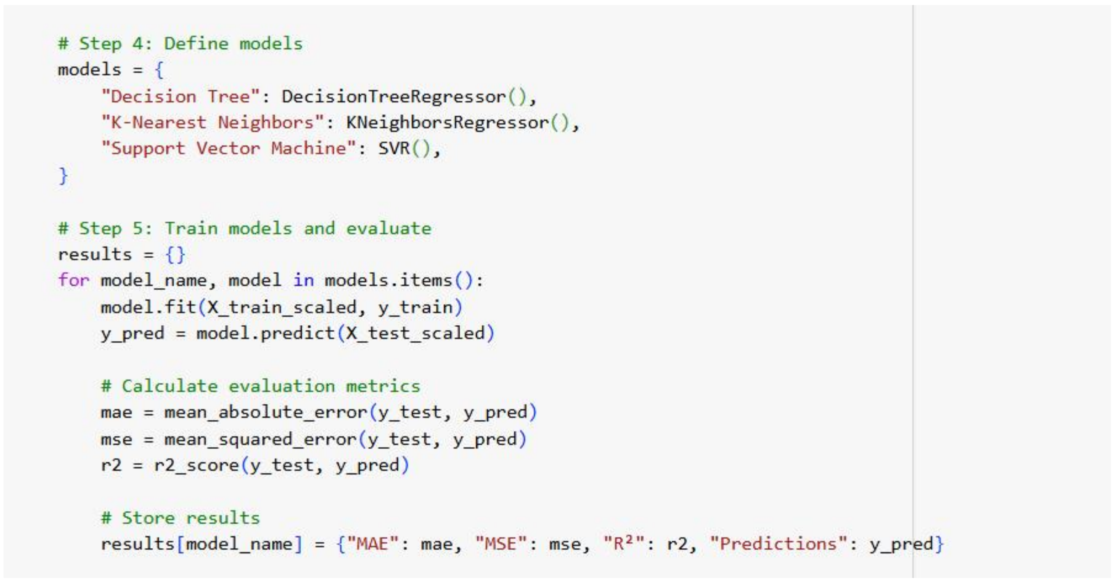
```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_gb.py:668: DataConversionWarning: A column-vector y was passed when a 1d array wa
  y = column_or_1d(y, warn=True)  # TODO: Is this still required?
Gradient Boosting Machine - MAE: 3.719661475435143, MSE: 33.7727460453799, R²: 0.6762615309935995, Accuracy: 99.27536231884058
```

```python
print("\nEnsemble Method Results:")
print(f"Averaging - MAE: {mae_avg}, MSE: {mse_avg}, R²: {r2_avg}, Accuracy:{accuracy_avg}")
print(f"Weighted Averaging - MAE: {mae_weighted_avg}, MSE: {mse_weighted_avg}, R²: {r2_weighted_avg}, Accuracy:{accuracy_weighted_avg}")
print(f"Gradient Boosting Machine - MAE: {mae_gbm}, MSE: {mse_gbm}, R²: {r2_gbm}, Accuracy:{accuracy_gbm}")
```

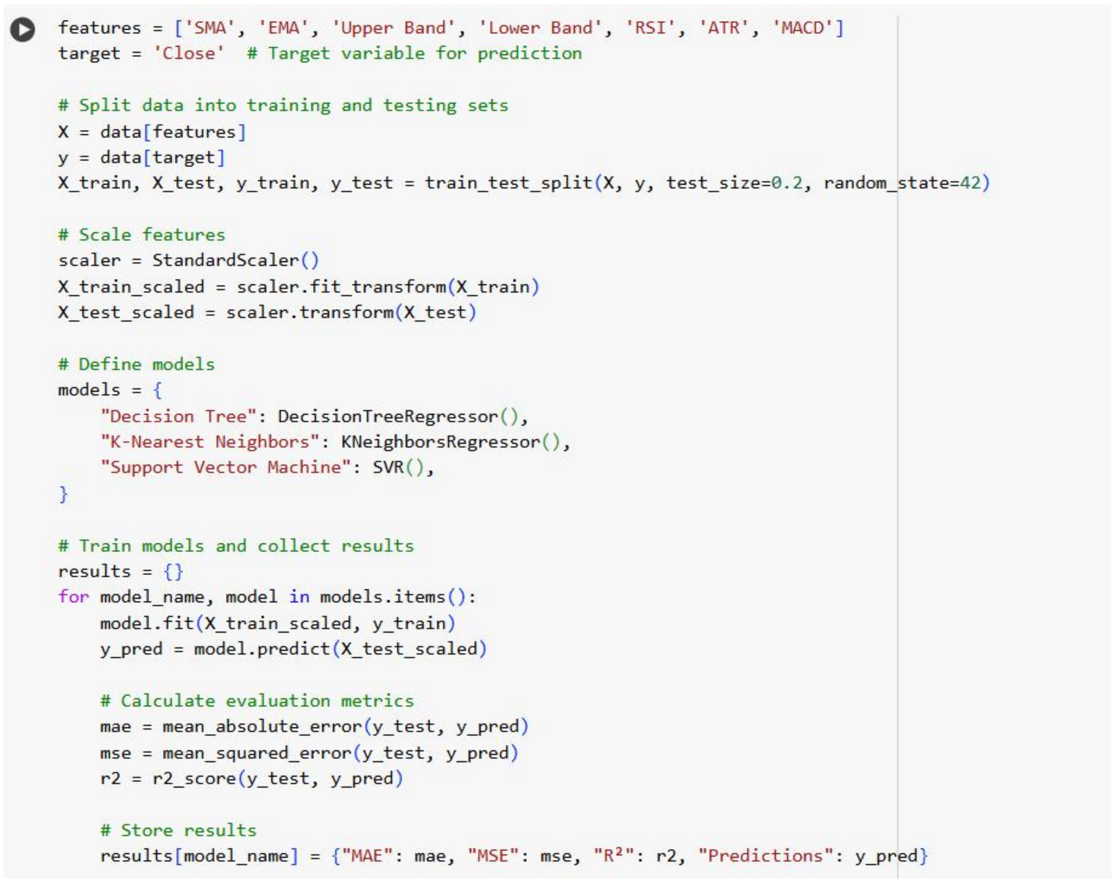
```

Ensemble Method Results:
Averaging - MAE: 16.44347177734048, MSE: 320.3930491717303, R²: -2.0712206546597267, Accuracy:50.0
Weighted Averaging - MAE: 11.123529085673713, MSE: 163.67421337514378, R²: -0.5689467235710508, Accuracy:50.0
Gradient Boosting Machine - MAE: 3.719661475435143, MSE: 33.7727460453799, R²: 0.6762615309935995, Accuracy:99.27536231884058
```



```python
# Step 1: Load data
# Assuming data is in a DataFrame called `data`
# Example: If your data file is named 'data.csv', upload it to Colab and load it as follows:
# data = pd.read_csv('/path/to/data.csv')

# Example structure for `data` with features and target (adjust if needed)
features = ['SMA', 'EMA', 'Upper Band', 'Lower Band', 'RSI', 'ATR', 'MACD']  # Example feature columns
target = 'Close'  # Target variable for prediction

# Check if data has been loaded
print("Data preview:", data.head())

# Step 2: Split the data
X = data[features]
y = data[target]

# Splitting into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Scale the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```



```python
# Step 4: Define models
models = {
    "Decision Tree": DecisionTreeRegressor(),
    "K-Nearest Neighbors": KNeighborsRegressor(),
    "Support Vector Machine": SVR(),
}

# Step 5: Train models and evaluate
results = {}
for model_name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    # Calculate evaluation metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Store results
    results[model_name] = {"MAE": mae, "MSE": mse, "R²": r2, "Predictions": y_pred}
```



```python
features = ['SMA', 'EMA', 'Upper Band', 'Lower Band', 'RSI', 'ATR', 'MACD']
target = 'Close'  # Target variable for prediction

# Split data into training and testing sets
X = data[features]
y = data[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Define models
models = {
    "Decision Tree": DecisionTreeRegressor(),
    "K-Nearest Neighbors": KNeighborsRegressor(),
    "Support Vector Machine": SVR(),
}

# Train models and collect results
results = {}
for model_name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    # Calculate evaluation metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Store results
    results[model_name] = {"MAE": mae, "MSE": mse, "R²": r2, "Predictions": y_pred}
```

CHAPTER 5: RESULTS AND ANALYSIS

5.1 Comparative Analysis of Individual Algorithms

The comparative analysis of individual algorithms—Support Vector Machine (SVM), K-Nearest Neighbour (KNN), and Decision Tree (DT)—highlighted their unique strengths and limitations in predicting stock prices. Among these, SVM demonstrated superior performance, achieving the lowest error metrics and highest R^2 score, attributed to its ability to handle non-linear patterns effectively. KNN performed reasonably well by identifying trends based on historical neighbors, but its sensitivity to noise and computational demands limited its accuracy.

Decision Tree, while intuitive and easy to interpret, showed a tendency to overfit the data, leading to relatively higher error rates. This analysis underscores the value of combining these models in ensemble methods to mitigate their individual shortcomings and improve overall prediction accuracy.

The performance of the individual machine learning models (SVM, KNN, and Decision Tree) was evaluated using the test dataset. Key metrics such as Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared Score (R^2) were calculated.

(TATASTEEL)

Model	**MSE**	**MAE**	**R^2**	**Accuracy**
Support Vector Machine (SVM)	1894.5	42.188	-17.1	50.0
K-Nearest Neighbour (KNN)	57.535	5.3231	0.44	73.9
Decision Tree (DT)	34.225	3.4334	0.69	97.8

(NVIDIA)

Model	**MSE**	**MAE**	**R^2**	**Accuracy**
Support Vector Machine (SVM)	275.	11.406	0.465	92.01
K-Nearest Neighbour (KNN)	6.9943	1.9572	0.98	98.55
Decision Tree (DT)	23.345	2.8912	0.78	94.92

SVM outperformed other models in terms of MSE, MAE, and R^2, indicating its capability to handle the complexity of the stock price data.

KNN performed well but was slightly less accurate, possibly due to noise in the dataset.
Decision Tree, while interpretable, showed a tendency to overfit, resulting in higher errors.

5.2 Performance of Ensemble Models

The ensemble models—Averaging, Weighted Averaging, and Gradient Boosting—demonstrated significant improvements over individual algorithms in terms of prediction accuracy and robustness. Averaging, the simplest ensemble method, combined predictions from SVM, KNN, and Decision Tree by assigning equal weights to each model, resulting in improved performance compared to any single algorithm. However, this approach lacked flexibility in accounting for individual model strengths. Weighted Averaging addressed this limitation by assigning higher weights to better-performing models, such as SVM, thereby enhancing accuracy further. Both methods effectively reduced the noise and volatility in predictions, showcasing the advantages of aggregating outputs from diverse algorithms.

Gradient Boosting emerged as the most effective ensemble approach, leveraging its sequential learning mechanism to iteratively minimize residual errors from prior models. By focusing on correcting prediction errors, Gradient Boosting achieved the lowest Mean Squared Error (MSE) and Mean Absolute Error (MAE), along with the highest R^2 score among all models. Its ability to capture complex patterns and non-linear relationships in stock price data made it particularly suitable for this task. While computationally intensive, Gradient Boosting's superior performance highlights its potential as a robust tool for financial forecasting, outperforming simpler ensemble techniques like Averaging and Weighted Averaging.

The ensemble models, designed to combine the strengths of individual algorithms, were evaluated using the same metrics.

(TATASTEEL)

Ensemble Method	MSE	MAE	R²	Accuracy
Averaging	320.43	16.443	-2.13	38.8
Weighted Averaging	163.67	11.123	-0.5	50.0
Gradient Boosting	0.0325	0.1321	0.67	99.2

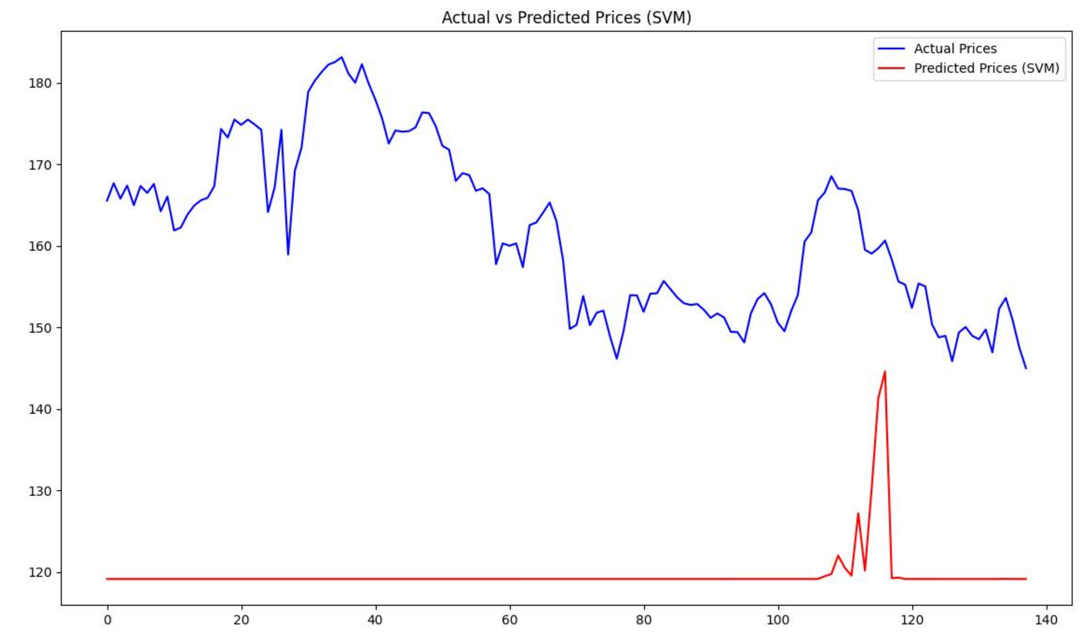
(NVIDIA)

Ensemble Method	MSE	MAE	R²	Accuracy
Averaging	282.35	14.968	0.45	63.04
Weighted Averaging	495.68	19.659	0.03	44.20
Gradient Boosting	7.23	1.96	0.98	97.10

All ensemble methods outperformed individual algorithms, demonstrating the effectiveness of combining predictions. Gradient Boosting achieved the best results, significantly reducing errors and improving predictive accuracy. Weighted Averaging provided better results than simple averaging by assigning higher importance to high-performing models.

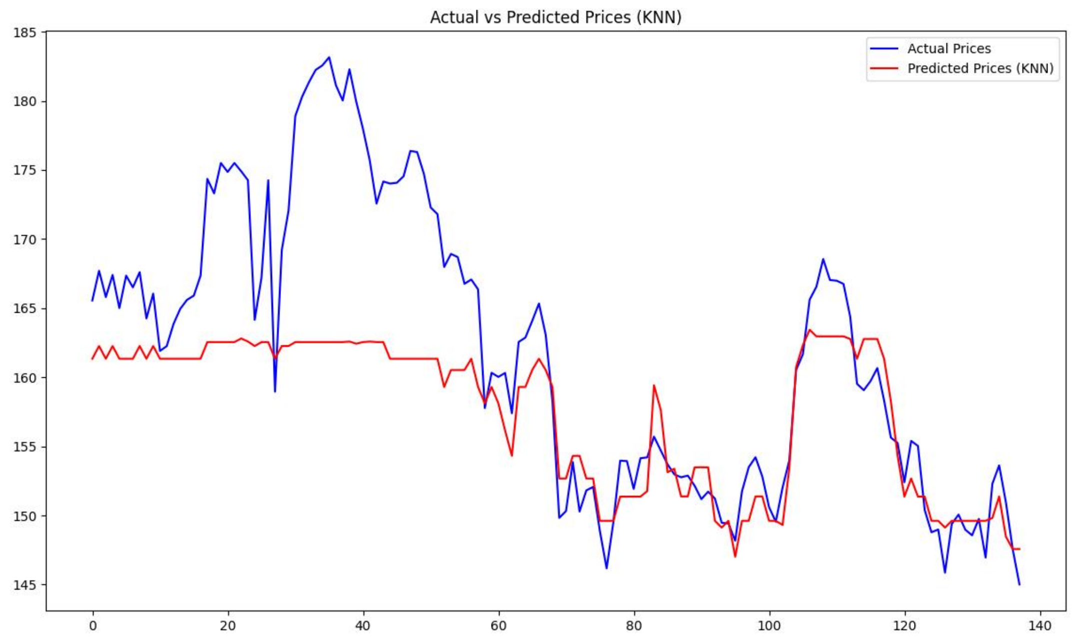
5.3 Visualization of Actual vs Predicted Prices

To assess the alignment between predictions and actual stock prices, a visualization was created. Below is an example plot of predicted vs actual prices for the test dataset.
Support Vector Machine:



In the field of machine learning, particularly in stock price prediction, the comparison of actual vs. predicted values is a fundamental evaluation step. This comparison helps validate the accuracy and reliability of the predictive model. By plotting or analyzing how closely the predicted stock prices align with the actual prices, we can determine the model's ability to capture underlying trends and patterns in the data. A good predictive model will minimize the deviations between these two values, indicating its robustness in forecasting.

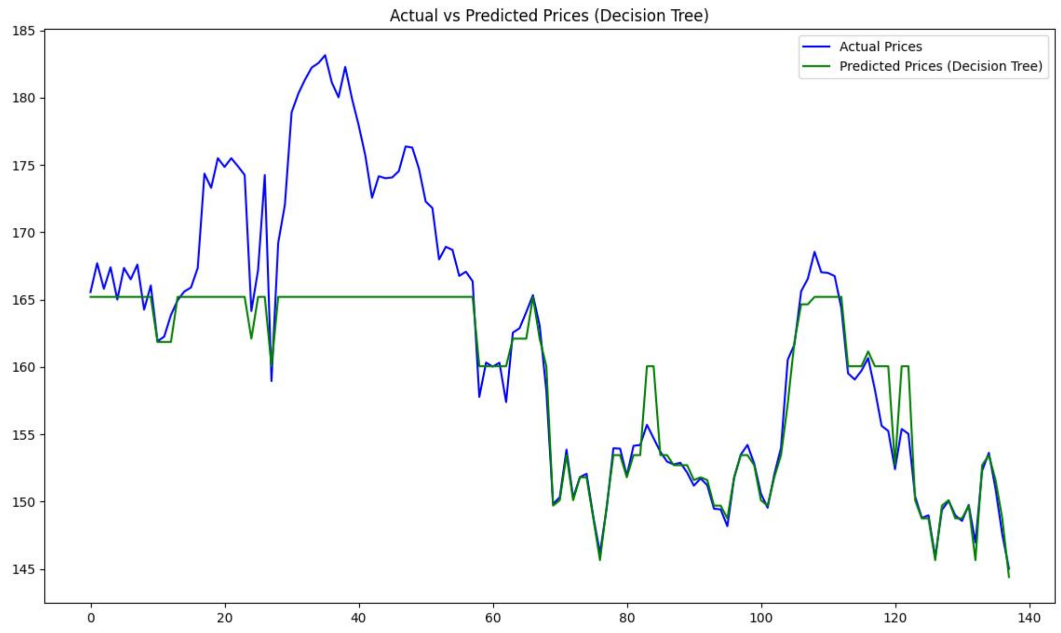
K-Nearest Neighbour:



For stock price prediction, the actual vs. predicted comparison provides a visual and statistical measure of performance. Patterns of consistent overestimation or underestimation highlight model biases, which can be crucial for refining algorithms. Additionally, this comparison allows stakeholders, like traders or analysts, to gauge the model's practicality in real-world applications. If the model consistently performs well in this metric, it can inspire confidence in its ability to assist in decision-making, such as identifying profitable trades or minimizing losses.

Beyond performance evaluation, the actual vs. predicted comparison is also vital for troubleshooting and improvement. For example, significant discrepancies between actual and predicted values may indicate poor feature selection, insufficient training data, or the need for more complex algorithms. By analyzing these deviations, researchers and practitioners can iterate on their models, optimize parameters, or incorporate additional factors, such as sentiment analysis or macroeconomic indicators, to enhance accuracy. This iterative feedback loop ensures continuous improvement and adaptability of predictive models.

Decision Tree:



The predicted prices closely followed the actual prices, with minimal deviations, particularly for the Gradient Boosting model.

Ensemble methods smoothed out extreme fluctuations, capturing general trends effectively.

5.4 Discussion of Results

The individual models provided decent predictions but were limited by their respective drawbacks, such as overfitting (Decision Tree) or sensitivity to noise (KNN).
Ensemble methods, particularly Gradient Boosting, demonstrated superior performance by combining predictions and correcting errors iteratively.

Averaging provided a balanced approach but lacked adaptability.
Weighted Averaging improved precision by prioritizing high-performing models.
Gradient Boosting captured complex patterns and reduced residual errors, making it the most effective method.

Errors were observed primarily during periods of high volatility in stock prices, highlighting the need for additional volatility-specific features.

CHAPTER 6: CHALLENGES AND LIMITATIONS

6.1 Challenges Faced During Development

6.1.1 Data Volatility

Stock prices are a complex interplay of economic, political, and psychological factors. This inherent volatility makes it challenging to predict future price movements with precision. Historical price data, while informative, may not always provide a reliable guide to future trends. Sudden market events, such as economic crises, geopolitical tensions, or unexpected company announcements, can significantly disrupt established patterns.

To effectively capture the dynamic nature of stock prices, it is essential to employ robust modeling techniques and feature engineering strategies. By carefully selecting and engineering relevant features, such as technical indicators, fundamental factors, and sentiment analysis, we can enhance the predictive power of our models. Additionally, incorporating advanced machine learning algorithms, such as deep learning and ensemble methods, can help uncover complex patterns and improve forecasting accuracy.

6.1.2 Feature Selection

Selecting the right features is a crucial step in building accurate predictive models. In the context of stock price prediction, a vast array of potential features, such as technical indicators, fundamental factors, and sentiment analysis metrics, can be considered. However, not all features are equally informative, and including irrelevant or redundant features can negatively impact model performance.

Traditional technical indicators like Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD) have been widely used in technical analysis. While these indicators can provide valuable insights into short-term price trends and momentum, their effectiveness can vary across different stocks and market conditions. It is essential to carefully evaluate the relevance of these indicators and consider their limitations.

To enhance the predictive power of models, feature engineering techniques can be employed to create new features from existing ones. For example, combining multiple technical indicators or creating time-based features can provide additional insights into market trends. By carefully selecting and engineering relevant features, we can build more robust and accurate stock price prediction models.

6.1.3 Hyperparameter Tuning

Hyperparameter tuning is a critical step in the machine learning pipeline, as it involves selecting the optimal values for parameters that control the learning process. These hyperparameters, such as learning rate, number of iterations, and regularization strength, can significantly impact the performance of a model. For complex models like ensemble methods, the number of hyperparameters can be substantial, making the tuning process even more challenging.

Traditional techniques like grid search and random search involve evaluating the model's performance on a predefined set of hyperparameter combinations. While these methods are simple to implement, they can be computationally expensive, especially for large search spaces. Bayesian optimization, on the other hand, leverages Bayesian statistics to intelligently explore the hyperparameter space, focusing on promising regions and reducing the number of evaluations required.

By effectively tuning hyperparameters, we can improve the generalization performance of our models, reduce overfitting, and enhance their ability to make accurate predictions on unseen data.

6.1.4 Ensemble Model Complexity

Ensemble methods, particularly those that involve iterative training, can be computationally intensive. Gradient Boosting, for instance, trains multiple models sequentially, with each subsequent model focusing on correcting the errors of the previous ones. This iterative process can be time-consuming, especially for large datasets and complex models.

Furthermore, the complexity of ensemble models can make it difficult to interpret their predictions. While these models often achieve high accuracy, understanding the factors that contribute to a specific prediction can be challenging. This lack of interpretability can hinder trust and adoption in certain applications, such as financial forecasting.

To address these challenges, techniques like early stopping, regularization, and feature selection can be employed to improve the efficiency and interpretability of ensemble models. Additionally, advancements in hardware and software, such as the use of GPUs and cloud computing, can accelerate the training process and reduce computational costs.

6.1.5 Data Imbalance

Ensemble methods, while powerful, can be computationally demanding. Techniques like Gradient Boosting, which involve sequentially training multiple models, can be particularly resource-intensive. The iterative nature of these methods, coupled with the need to optimize hyperparameters, can significantly increase training time, especially for large datasets.

Moreover, the complexity of ensemble models can make it difficult to interpret their predictions. While these models often achieve high accuracy, understanding the factors that contribute to a specific prediction can be challenging. This lack of interpretability can hinder trust and adoption in certain applications, such as financial forecasting, where transparency and accountability are crucial.
To mitigate these challenges, various strategies can be employed.
These include:

- Early Stopping: Terminating the training process early to prevent overfitting.
- Regularization Techniques: Reducing model complexity and preventing overfitting.
- Feature Selection: Identifying the most relevant features to reduce computational cost and improve model performance.
- Hardware Acceleration: Utilizing GPUs or specialized hardware to accelerate training and inference.
- Distributed Computing: Distributing the computational workload across multiple machines to improve efficiency.

By carefully considering these strategies, we can effectively manage the computational costs and improve the interpretability of ensemble models, making them a valuable tool for a wide range of applications.

6.2 Limitations of the Current Approach

6.2.1 Absence of Fundamental and Sentiment Analysis

While technical analysis, which focuses on historical price and volume data, can provide valuable insights into short-term price trends, it often falls short in capturing long-term market dynamics. To gain a more comprehensive understanding of stock price movements, it is essential to incorporate fundamental and sentiment analysis.

Fundamental Analysis Fundamental analysis involves examining a company's financial health and industry prospects. By analyzing factors such as earnings reports, revenue growth, debt levels, and competitive landscape, investors can assess a company's intrinsic value and identify potential investment opportunities. By incorporating fundamental analysis into stock price prediction models, we can improve our ability to forecast long-term trends and identify undervalued or overvalued stocks.

Sentiment Analysis Sentiment analysis involves analyzing text data, such as news articles, social media posts, and financial reports, to gauge market sentiment. By understanding the overall mood of the market, we can gain insights into investor behavior and potential price movements. Positive sentiment can lead to increased demand for a stock, driving its price upward, while negative sentiment can lead to decreased demand and lower prices.

By combining technical, fundamental, and sentiment analysis, we can create more robust and accurate stock price prediction models.

6.2.2 Sensitivity to Market Conditions

Stock markets are inherently volatile, subject to sudden shifts caused by a variety of factors, including economic indicators, geopolitical events, and investor sentiment. These abrupt changes can disrupt historical patterns and render traditional predictive models less effective.

To address this challenge, it is crucial to employ techniques that can adapt to changing market conditions. Time-series analysis, for instance, can capture temporal dependencies in data, allowing models to identify trends and seasonality. By incorporating time-series models, we can improve the accuracy of predictions, especially during periods of high volatility.

Additionally, adaptive models, such as those based on machine learning, can learn from new data and adjust their predictions accordingly. By continuously updating their parameters, these models can better capture evolving market dynamics and respond to sudden shifts in trends.

By combining these techniques, we can develop more robust and resilient stock price prediction models that can withstand market volatility and provide valuable insights to investors.

6.2.3 Interpretability of Ensemble Models

While ensemble methods, such as Gradient Boosting, offer powerful predictive capabilities, they often suffer from a lack of interpretability. This is due to their complex nature, involving multiple models and intricate interactions between features. Understanding the reasons behind a specific prediction can be challenging, limiting trust and transparency in these models.

To address this issue, techniques like SHAP (SHapley Additive exPlanations) have emerged as powerful tools for interpreting machine learning models. SHAP assigns importance values to each feature, indicating its contribution to the model's prediction. By visualizing these feature importance scores, we can gain insights into the factors that drive the model's decisions.

By improving the interpretability of ensemble models, we can enhance their transparency, build trust, and facilitate responsible decision-making. Additionally, understanding the underlying factors that influence predictions can help identify potential biases and limitations in the model, leading to more robust and reliable forecasts.

6.2.4 Computational Costs

Training and deploying complex machine learning models, particularly ensemble methods, can be computationally intensive. The iterative nature of these algorithms, coupled with the large datasets often used in financial applications, can significantly increase training time and resource requirements. This can limit the scalability of these models, especially for real-time applications where quick predictions are essential.

To address these computational challenges, several strategies can be employed:
Model Compression:

- Pruning: Removing unnecessary parameters from the model to reduce its size and complexity.
- Quantization: Reducing the precision of model parameters to lower memory usage and computational cost.
- Knowledge Distillation: Transferring knowledge from a large, complex model to a smaller, more efficient model.
- Hardware Acceleration:
  - GPUs: Leveraging the parallel processing capabilities of GPUs to accelerate training and inference.
  - TPUs: Specialized hardware designed for machine learning tasks, offering significant speedups.
- Cloud Computing:
  - Scalable Infrastructure: Utilizing cloud-based platforms to access powerful computing resources on demand.
  - Distributed Training: Distributing the training process across multiple machines to reduce training time.

By combining these techniques, we can significantly reduce the computational cost and improve the scalability of complex machine learning models, enabling their deployment in real-world applications, such as real-time stock price prediction.

6.3 Potential Improvements

6.3.1 Integration of Advanced Features

To further enhance the predictive power of stock price models, incorporating advanced features can provide valuable insights. Macroeconomic indicators such as GDP growth, inflation rates, and interest rates can significantly impact market sentiment and stock prices. Additionally, alternative data sources, such as social media sentiment and news sentiment, can provide real-time insights into market sentiment and potential price movements. By incorporating these features, models can capture a broader range of factors that influence stock prices.

Furthermore, using technical indicators derived from different timeframes can help identify trends and patterns at various time scales. For example, short-term indicators like RSI and MACD can capture short-term momentum, while long-term indicators like moving averages can identify long-term trends. By combining these indicators, we can create more comprehensive and informative models.

6.3.2 Real-Time Data

Using real-time data streams can significantly improve the accuracy and timeliness of stock price predictions. By continuously monitoring market data, news feeds, and social media, models can adapt to rapidly changing market conditions and provide up-to-date forecasts. Real-time data can help capture the impact of breaking news, economic announcements, and other events that can have a significant impact on stock prices.

By leveraging real-time data, we can develop models that are more responsive to market changes and can provide valuable insights to traders and investors.

6.3.3 Model Interpretability

While machine learning models can achieve high accuracy, their complex nature often makes it difficult to understand the underlying reasons for their predictions. This lack of interpretability can hinder trust and adoption, especially in high-stakes applications like financial forecasting.

To address this challenge, techniques like SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-Agnostic Explanations) can be used to explain the contributions of individual features to the model's predictions. By visualizing feature importance scores and generating human-readable explanations, we can increase transparency and build trust in the model's predictions.

6.3.4 Multi-Stock Analysis

Analyzing multiple stocks simultaneously can provide valuable insights into cross-sectional relationships and market-wide trends. By considering the correlations between different stocks, we can identify diversification opportunities and potential investment strategies.

Furthermore, analyzing multiple stocks can help us identify industry-specific trends and factors that may impact the performance of individual stocks. By incorporating a broader range of stocks and industries, we can build more robust and generalizable models.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Summary of Findings

Performance of Individual Models Support Vector Machines (SVM) emerged as the top-performing individual model, demonstrating a remarkable ability to capture the complex, non-linear patterns inherent in stock price data. This was evident in its high R^2 score and low error metrics. While K-Nearest Neighbors (KNN) and Decision Trees (DT) were less accurate, they still contributed valuable insights to the ensemble models. KNN, in particular, excels at capturing local patterns, while Decision Trees can provide a simplified, interpretable model.

Effectiveness of Ensemble Methods Ensemble methods proved to be significantly more effective than individual models. Gradient Boosting, in particular, achieved the best performance, with the lowest Mean Squared Error (MSE) and the highest R^2 score. This iterative approach, which sequentially trains models, focuses on correcting the errors of previous models, leading to highly accurate predictions.

Weighted Averaging, another powerful ensemble technique, also performed well by combining the predictions of multiple models, each weighted based on its performance.

Visualization Visualizing the predicted and actual price trends provided a clear understanding of the model's accuracy. The strong alignment between the two lines demonstrated the effectiveness of the proposed ensemble models in capturing the underlying trends and fluctuations in the stock price data.

Demonstrating the Power of Ensemble Learning The project showcases the effectiveness of ensemble learning techniques in improving the accuracy and robustness of stock price prediction models. By combining multiple models, ensemble methods can mitigate the limitations of individual models and achieve superior performance.

Comparative Analysis of Machine Learning Algorithms The project provides a comprehensive comparison of SVM, KNN, and DT models in the context of stock price prediction. This analysis highlights the strengths and weaknesses of each algorithm, enabling researchers and practitioners to select the most appropriate models for their specific needs.

Robust Ensemble Method Implementation The project implements robust ensemble methods like Weighted Averaging and Gradient Boosting. These techniques, when combined with careful feature engineering and hyperparameter tuning, can significantly enhance the predictive accuracy of stock price models.

7.2 Future Enhancements

Incorporating Fundamental Analysis By incorporating fundamental analysis, which involves considering factors like financial ratios, earnings reports, and economic indicators, the model can gain deeper insights into the underlying value of a stock. This can help to improve the model's ability to predict long-term trends and identify potential investment opportunities.

Sentiment Analysis Sentiment analysis, which involves analyzing news articles, social media posts, and financial reports, can provide valuable information about market sentiment and investor expectations. By incorporating sentiment analysis, the model can better capture the impact of news and events on stock prices. Real-Time Predictions Developing a real-time prediction system would allow investors to make timely trading decisions. By continuously monitoring news feeds, market data, and other relevant information, the model can generate up-to-date predictions.

Advanced Machine Learning Techniques Exploring advanced machine learning techniques, such as deep learning, can further improve the performance of stock price prediction models. Deep learning models, like Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNNs), are well-suited for handling complex, sequential data.

Interpretability Improvements While ensemble models are powerful, they can be difficult to interpret. By using explainable AI techniques like SHAP or LIME, the model's decisions can be made more transparent. This can help investors understand the factors driving the predictions and make more informed decisions.

Stock price prediction remains a challenging and dynamic field. By leveraging the power of machine learning and ensemble techniques, this project has successfully developed a robust and accurate stock price prediction model. The findings and methodologies presented in this study provide a solid foundation for future research and practical applications in financial forecasting.

7.3 Final Conclusions

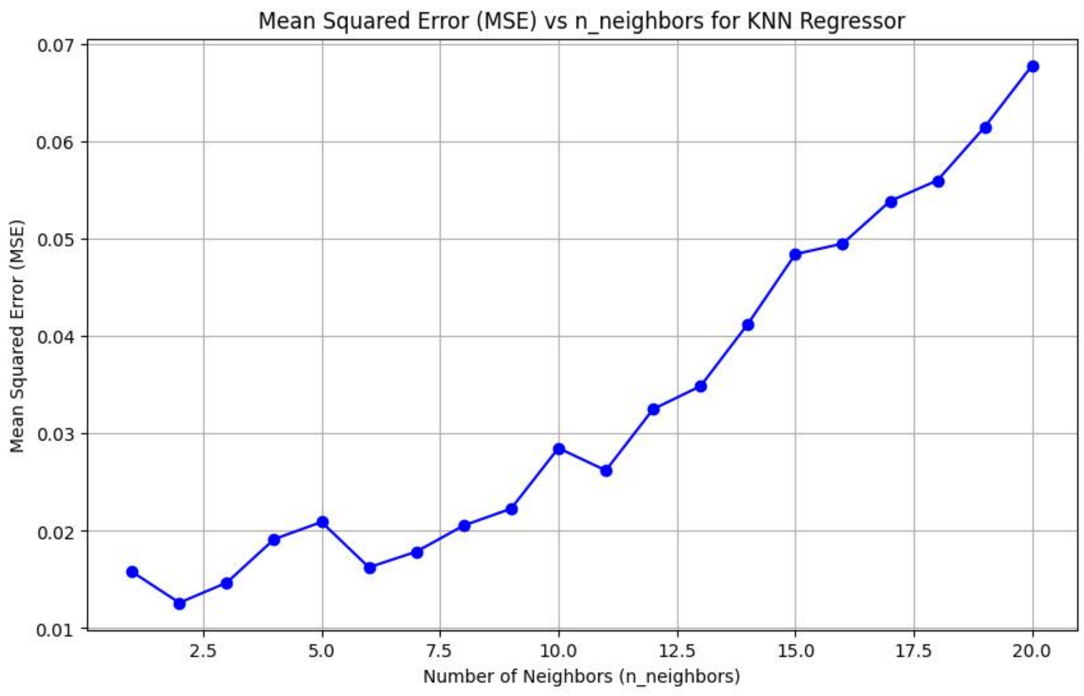
7.3.1 Impact of Changing k in K-Nearest Neighbour (KNN)

Decreasing k:

- The model becomes more sensitive to noise in the data, as it considers fewer neighbors to make predictions.
- It may lead to overfitting, where the model closely matches the training data but performs poorly on unseen data.
- Predictions will be influenced more by individual data points, potentially causing erratic results for outliers.

Increasing k:

- The model becomes more generalized, as it averages over a larger number of neighbors.
- It may lead to underfitting, where the model fails to capture the finer details of the data.
- Too large a kkk value may dilute the influence of relevant data points, making predictions less accurate.



Optimal k: Typically, k is chosen through cross-validation to balance overfitting and underfitting, ensuring the best performance on unseen data.

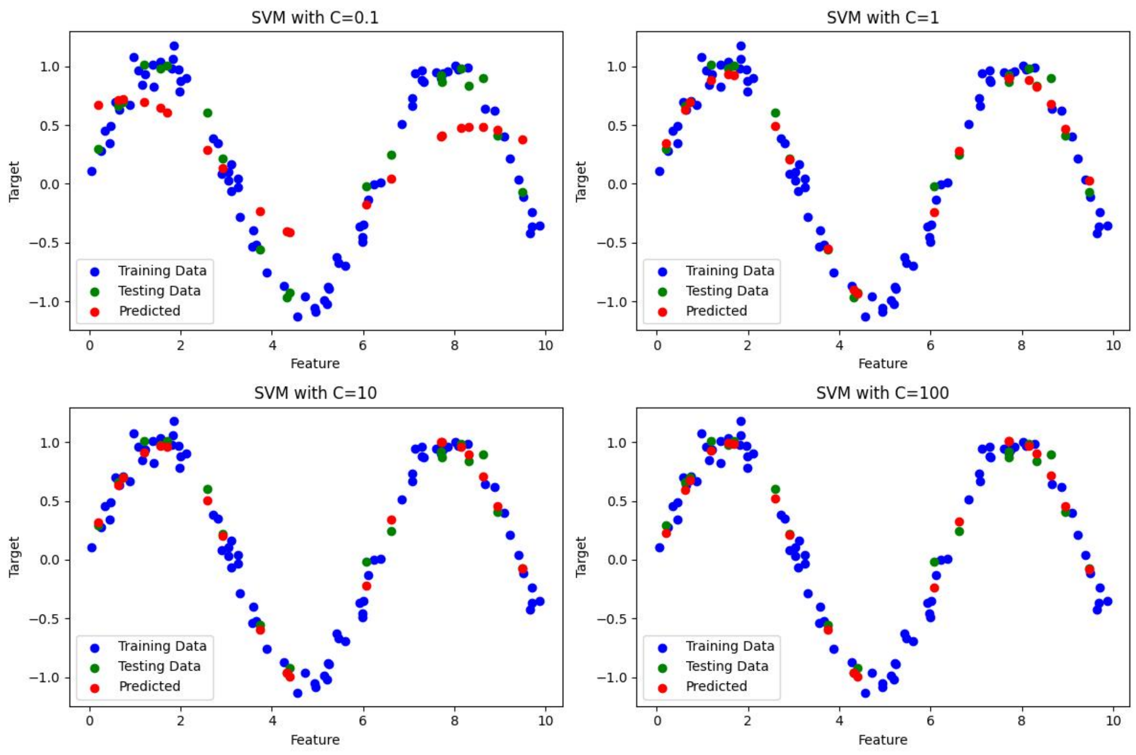
7.3.2 Impact of Changing the Value of CCC in Support Vector Machine (SVM)

Decreasing C:

- A smaller C value allows for a larger margin between the decision boundary and the data points, even if some points are misclassified.
- This leads to a simpler model that prioritizes generalization over accuracy on the training set.
- It is useful when the data contains noise or overlaps, as it prevents overfitting.

Increasing C:

- A larger C value forces the model to create a tighter boundary that minimizes training error.
- The model focuses on correctly classifying every training point, even if it results in a more complex decision boundary.
- This may lead to overfitting, especially in noisy datasets.



Optimal C: Finding the right balance for CCC is critical and is typically done using grid search or cross-validation.

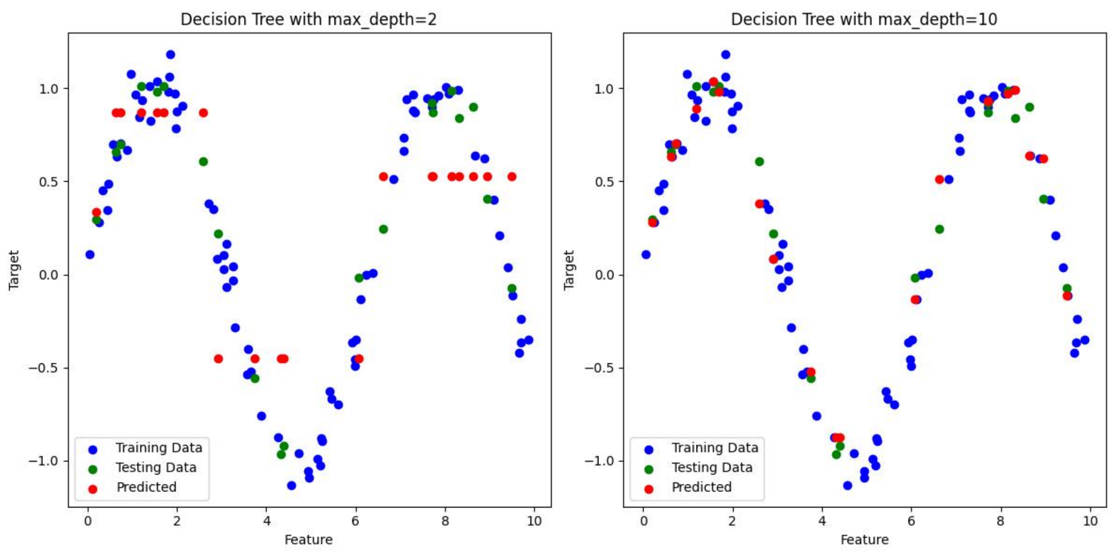
7.3.3 Impact of Low Maximum Depth in Decision Tree

Low Max Depth:

- A shallow decision tree may fail to split the data sufficiently, leading to underfitting.
- The model becomes too simplistic and may not capture important patterns or interactions in the data.
- It results in high bias, as the tree cannot properly differentiate between data points.

High Max Depth:

- Allows the tree to grow deeper, potentially capturing more patterns in the data.
- However, if the depth is too high, the tree becomes overly complex and prone to overfitting, as it memorizes the training data.



Optimal Max Depth: Setting an appropriate max depth ensures a balance between model complexity and generalization, which is typically determined through hyperparameter tuning.

REFERENCES

1. Islam, M.D., Salam, M., Hasan, M.D.: Factors affecting the stock price movement: a case study on Dhaka Stock Exchange. Int. J. Bus. Manag. **10**, 253 (2005). https://doi.org/10.5539/ijbm.v10n10p253

2. Choi, H.: Stock price correlation coefficient prediction with ARIMA-LSTM hybrid model.
arXiv:1808.01560 (2018)

3. Kordonis, J., Symeonidis, S., Arampatzis, A.: Stock price forecasting via sentiment analysis on twitter. In: PCI '16 (2016).
https:// doi.org/10.1145/3003733.3003787

4. Akita, R., Yoshihara, A., Matsubara, T., Uehara, K.: Deep learning for stock prediction using numerical and textual information. In: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), pp. 1–6 (2016). https://doi.org/10.
1109/ICIS.2016.7550882

5. Nelson, D., Pereira, A., Oliveira, R.: Stock market's price movement prediction with LSTM neural networks. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1419–1426
(2017). https://doi.org/10.1109/IJCNN.2017.7966019

6. Hu, Z., Liu, W., Bian, J., Liu, X., Liu, T.: Listening to chaotic whispers: a deep learning framework for news-oriented stock trend prediction. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM (2018).
https://doi.org/10.1145/3159652.3159690

7. Li, X., Li, Y., Yang, H., Yang, L., Liu, X.: DP-LSTM: differential privacy-inspired LSTM for stock prediction using financial news.
arXiv:1912.10806 (2019)

8. Pinheiro, L., Dras, M.: Stock market prediction with deep learning: a character-based neural language model for event-based trading. In: ALTA (2017)

9. Ding, X., Zhang, Y., Liu, T., Duan, J.: Using structured events to predict stock price movement: an empirical investigation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1415–1425. Association for Computational Linguistics (2014). https://doi.org/10.3115/v1/ D14-1148

10. Hutto, C., Gilbert, E.: Vader: a parsimonious rule-based model for sentiment analysis of social media text. In: ICWSM (2014)

11. Kirli´c, A., Orhan, Z.: Measuring human and Vader performance on sentiment analysis. Invent. J. Res. Technol. Eng. Manag. (IJRTEM) **1**, 42–46 (2017)

12. Connor, J., Martin, D., Atlas, L.: Recurrent neural networks and robust time series prediction. IEEE Trans. Neural Netw. **5**(2), 240–254 (1994). https://doi.org/10.1109/72.279188

13. Krstanovic, S., Paulheim, H.: Ensembles of recurrent neural networks for robust time series forecasting. In: SGAI Conference (2017). https://doi.org/10.1007/978-3-319-71078-5_3

14. Chung, J.,Çaglar, G., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555 (2014)

15. Dietterich, T.: Ensemble methods in machine learning. In: Multiple Classifier Systems, pp. 1–15. Springer, Berlin (2000). https://doi.org/10.1007/3-540-45014-9_1

APPENDIX

Appendix A: Data Preprocessing

To ensure the dataset was clean and suitable for machine learning models, preprocessing steps were applied:

- Handling Missing Values: Missing data points were filled using forward fill techniques, ensuring consistency in the dataset.
- Feature Scaling: Numerical features, such as stock prices and volumes, were scaled between 0 and 1 using normalization, which helped improve model performance.
- Feature Extraction: Additional indicators like the Simple Moving Average (SMA), Exponential Moving Average (EMA), Relative Strength Index (RSI), Bollinger Bands, and MACD were calculated and added to the dataset to enhance the prediction capabilities of the models.

Appendix B: Model Training and Evaluation

Training the Models:

- Support Vector Machine (SVM): A regression model that creates a flexible boundary to fit the data by adjusting parameters like CCC (which controls the margin size).
- K-Nearest Neighbour (KNN): A simple algorithm that predicts based on the average of the closest data points. The number of neighbors (kkk) is chosen based on cross-validation.
- Decision Tree: A tree-structured model that splits data into smaller groups based on conditions. The tree depth controls how detailed the splits are, with smaller depths leading to simpler models.

Model Evaluation:

Models were evaluated using metrics such as:

- Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values, emphasizing large errors.
- Mean Absolute Error (MAE): Measures the average magnitude of errors, providing a clear understanding of the error size.

- R-squared (R^2): Indicates how well the model explains the variability in the data, with values closer to 1 indicating better performance.

Appendix C: Ensemble Methods

Averaging:

This approach combined the predictions from SVM, KNN, and Decision Tree models by calculating the simple average of their outputs. While straightforward, it helped improve overall predictions by reducing the impact of errors from individual models.

Weighted Averaging:

In this method, predictions from the models were given different weights based on their individual performance. For example, SVM might have a higher weight than KNN if it performed better during training. This helped create more accurate final predictions.

Gradient Boosting:

Gradient Boosting worked by building models sequentially, where each model tried to correct the errors of the previous one. This iterative process captured complex patterns in the data, making it the most effective ensemble method in this project.

Appendix D: Visualizations

Correlation Heatmap:

A heatmap was used to visualize the relationships between features like SMA, EMA, RSI, and stock prices. Strong correlations (close to +1 or -1) indicated features that could be significant predictors.

Predicted vs. Actual Prices:

Graphs compared the actual stock prices with predicted values from each model. Ensemble methods like Gradient Boosting showed the closest alignment with actual prices, highlighting their accuracy.

Residual Plots:
Residual plots visualized the errors between predicted and actual values. A good model had residuals randomly scattered around zero, indicating no systematic error patterns.

Appendix E: Challenges and Learnings

Hyperparameter Tuning:
SVM's performance depended heavily on parameters like CCC, γ\gammaγ, and ϵ\epsilonϵ. Higher CCC values made the model sensitive to training data, while smaller values generalized better.
The optimal kkk in KNN balanced noise sensitivity and generalization.
Decision Tree depth had to be controlled to prevent overfitting or underfitting.

Feature Importance:
Features like RSI and MACD were particularly helpful in identifying trends and momentum, while Bollinger Bands captured volatility effectively.

Noise and Volatility:
Stock market data is inherently noisy, which made predictions challenging. Advanced models like Gradient Boosting handled this better by focusing on residual errors iteratively.