Dr. Babasaheb Ambedkar Technological University, Lonere
Bajaj Institute of Technology, Wardha
Pipri, Arvi Road, Wardha - 442001.

DEPARTMENT OF COMPUTER ENGINEERING

Certificate

This is to certify that Mini-Project-II titled

Stock Trend Prediction using Financial News Sentiment Score

has been completed by

Mr.Raghwendra Singh

Mr. Sameer Pichkate

Mr. Ninad chalakh

Mr.Siddesh Purohit

of VI Semester (Sec: A), Computer Engineering of academic year 2023-24 in partial fulfillment of Mini-Project-II (BTCOM607) course as prescribed by the Dr. Babasaheb Ambedkar Technological University, Lonere.

Mr.Vikas Palekar
(Project Guide)

Prof. Sheetal A.Kale
(Head of the Department)

Dr.N.M.Kanhe
Principal
Bajaj Institute of Technology Wardha

Acknowledgment

I would like to express my gratitude and appreciation to all those who gave me the possibility to complete this report. Special thanks are due to my Guide, Mr. Vikas Palekar, Assistant Professor of Computer Engineering, whose help, stimulating suggestions, and encouragement helped me in all stages of this project and in writing this report. I also sincerely thank him for the time spent proofreading and correcting my many mistakes.

My deepest thanks to the Department of Computer Engineering and Prof. Sheetal Kale, Head of Department, for her invaluable support and guidance. I also extend my gratitude to Dr. Narendra Kane, Principal of Bajaj Institute of Technology, Wardha.

I would also like to acknowledge with much appreciation the crucial role of the staff in the Computer Lab, who gave me permission to use the computers in the laboratory. Many thanks go to all the lecturers and to all my classmates, especially to my friends, for spending their time in helping and giving support whenever I needed it in my Mini Project Topic.

Abstract

The stock market is highly volatile as it depends on political, financial, environmental, and various internal and external factors along with historical stock data. Such information is available to people through microblogs and news, and predicting stock prices purely based on historical data is challenging. The high volatility emphasizes the importance of evaluating the effect of external factors on the stock market. In this repot, we propose a machine learning model that utilizes financial news along with historical stock price data to predict upcoming stock prices.

Our investigation focuses on understanding and quantifying the impact of financial news on stock prices. We use three algorithms to calculate various sentiment scores and deploy them in different combinations to comprehend the impact of financial news on stock prices, as well as the efficacy of each sentiment scoring algorithm. Experiments have been conducted on nine months of historical stock price data and financial news for four different companies from different sectors to predict the next day's and next week's stock trends. Accuracy metrics were evaluated over a period of 7 days. The proposed model achieved a highest accuracy of 0.72 for both trend and future trends over a period of 7 days.

The methodology involves web scraping for live news updates, sentiment analysis using the VADER Sentiment Intensity Analyzer, and merging sentiment scores with historical stock data. Implementation includes training a Multi-Layer Perceptron (MLP) model on these features to predict stock prices. Experiments also examined the difficulty in predicting specific stocks.

Keywords- ***Stock prediction***, ***MLP(multilayer perceptron)***, ***web scraping (beautifulsoup)***,***Financial news***

Abbreviations

MLP	Multilayer Perceptron
RSI	Relative Strength Index
	Vader

Contents

List of Figures

Chapter 1

Introduction

This project explores the integration of sentiment analysis of news articles with historical stock data to predict the closing prices of Reliance Power's stock. By leveraging natural language processing (NLP) techniques and machine learning, the goal was to create a model that can anticipate stock price movements based on the sentiment extracted from related news. We began by scraping live news updates and manually adding additional entries to form a comprehensive dataset. Sentiment scores were derived using the VADER sentiment analysis tool, and these scores were combined with stock data retrieved from Yahoo Finance. The combined dataset was then used to train a Multi-Layer Perceptron (MLP) Regressor, which aimed to predict future stock prices. This approach aims to demonstrate how sentiment analysis can enhance the predictive capabilities of financial models, offering a more nuanced understanding of market dynamics influenced by news sentiment.

The pursuit of precise stock prediction has drawn the attention of researchers from diverse academic backgrounds, spanning computer science, economics, statistics, and operations research. In recent years, the emergence of machine learning algorithms has sparked optimism as a potent tool for constructing predictive models. These models, often built upon architectures like the Multi-Level Perceptron (MLP), combine historical data with real-time sources such as social media feeds, news updates, and sentiment analysis.Moreover, the integration of web scraping techniques, facilitated by libraries like BeautifulSoup and Selenium, has augmented prediction models by enabling the retrieval of live data from online sources. This integration not only enhances the timeliness of information but also ensures its relevance and responsiveness, aligning with the ever-evolving landscape of data-driven decision-making in financial markets.

Chapter 2

Literature Survey

2.1 Literature review

SN	Paper	Author	Year
1	Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor:	Junaid maqbool, Preeti Aggrawal,Ajay Mittal	2023
2	Short-term prediction for opening price of stock market based on self-adapting variant PSO-Elman neural network	Ze Zhang; Yongjun Shen; Guidong Zhang; Yongqiang Song; Yan Zhu	2017
3	Stock Price Trend Prediction Model Based on Deep Residual Network and Stock Price Graph.	Heng Liu; Bowen Song	2018
4	Prediction of Stock Price Based on LSTM Neural Network	Dou Wei	2019
5	Time Series with Sentiment Analysis for Stock Price Prediction	Vrishabh Sharma; Rajgauri Khemnar; Renu Kumari; Biju R Mohan	2019
6	Stock Price Prediction Using News Sentiment Analysis	Saloni Mohan;Sahitya Mullapudi; Sudheer Sammeta; Parag Vijayvergia; David C. Anastasiu	2019

1. **Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor**

   Year of Publication - 2023

   Author - Junaid maqbool, Preeti Aggrawal,Ajay Mittal

   Literature Review: Literature Review:

   The integration of sentiment scores extracted from financial news with MLP (Multi-Layer Perceptron) regressors for stock prediction has garnered increasing attention in the field of quantitative finance.

   Studies typically begin by collecting a diverse dataset of financial news articles from various sources, including news websites, financial blogs, and social media platforms. Sentiment analysis techniques are then applied to quantify the positive or negative tone of the news articles, generating sentiment scores as additional features for modeling.

   Empirical evaluations demonstrate the effectiveness of incorporating sentiment scores in improving stock prediction accuracy, particularly in capturing short-term fluctuations and sentiment-driven movements in the market. MLP regressors, with their ability to capture complex nonlinear relationships, are well-suited for learning from both traditional financial indicators and sentiment features extracted from news data.

   Challenges in this approach include the need for robust sentiment analysis algorithms capable of accurately quantifying sentiment from noisy and context-dependent financial text data. Additionally, model interpretability and generalization remain significant concerns, with researchers exploring techniques such as feature selection, regularization, and ensemble learning to address these issues.

   Future research directions focus on advancing sentiment analysis techniques tailored to financial text data, such as sentiment embeddings and attention mechanisms. There is also growing interest in exploring the synergies between deep learning architectures and traditional regression models, leveraging the strengths of both approaches for improved stock prediction performance.

   Overall, the literature underscores the potential of integrating sentiment scores from financial news with MLP regressors for stock prediction, offering valuable insights into market sentiment and enhancing forecasting accuracy, especially in the context of short-term prediction horizons. Continued research efforts in this interdisciplinary area are expected to lead to further advancements and practical applications in financial forecasting.

2. **Short-term prediction for opening price of stock market based on self-adapting variant PSO-Elman neural network**

   Year of Publication - 2017

   Author - Ze Zhang; Yongjun Shen; Guidong Zhang; Yongqiang Song; Yan Zhu

Literature Review:

The utilization of self-adapting variant PSO-Elman neural networks for short-term prediction of the opening price of the stock market represents a cutting-edge approach in quantitative finance research.

Studies typically commence by collecting historical stock market data, including opening prices, along with relevant features such as trading volume, technical indicators, and market sentiment. The self-adapting variant PSO-Elman neural network architecture is then employed to model the temporal dependencies in the data and adaptively learn patterns in the time series.

Empirical evaluations demonstrate the effectiveness of the proposed approach in capturing short-term trends and fluctuations in the opening price of the stock market. The self-adapting variant PSO-Elman neural network's ability to dynamically adjust its parameters based on the data's characteristics allows for improved prediction accuracy compared to traditional time series models.

Challenges in this approach include parameter tuning, model complexity, and interpretability. Researchers have explored various strategies to address these challenges, such as optimizing the PSO parameters, regularization techniques, and ensemble learning methods.

Future research directions focus on enhancing the self-adapting variant PSO-Elman neural network model's robustness and interpretability, exploring techniques such as incorporating additional features, refining network architecture, and developing hybrid models.

Overall, the literature underscores the promise of using self-adapting variant PSO-Elman neural networks for short-term prediction of the stock market's opening price, offering valuable insights into market dynamics and enhancing forecasting accuracy. Continued research in this area is expected to lead to further advancements and practical applications in financial forecasting.

3. **Stock Price Trend Prediction Model Based on Deep Residual Network and Stock Price Graph.**

   Year of Publication - 2018

   Author -Heng Liu; Bowen Song

   Literature Review:

   The development of stock price trend prediction models based on deep residual networks (ResNets) and stock price graphs has gained traction in recent years, representing an innovative approach in quantitative finance research.

   Studies typically begin by collecting historical stock price data and constructing a graph representation that captures relationships between stocks based on factors such as sector, industry, or co-movement. Deep residual networks are then employed to learn complex patterns and temporal dependencies in the stock price data, incorporating information from the stock price graph to enhance predictive performance.

Empirical evaluations demonstrate the effectiveness of the proposed approach in capturing long-term trends and irregularities in stock price movements. The hierarchical nature of deep residual networks allows for the modeling of intricate relationships within the stock price graph, enabling more accurate trend predictions compared to traditional time series models.

Challenges in this approach include data preprocessing, model architecture design, and interpretability. Researchers have explored various techniques to address these challenges, such as feature engineering, regularization, and attention mechanisms.

Future research directions focus on advancing deep residual network models for stock price trend prediction, exploring techniques such as graph neural networks, ensemble learning, and hybrid architectures. Additionally, efforts are underway to develop interpretable models and enhance the robustness of predictions in the face of market uncertainties.

Overall, the literature highlights the promise of using deep residual networks and stock price graphs for stock price trend prediction, offering valuable insights into market dynamics and improving forecasting accuracy. Continued research in this area is expected to lead to further advancements and practical applications in financial forecasting.

4. **Prediction of Stock Price Based on LSTM Neural Network**

   Year of Publication - 2019)

   Author - Dou Wei

   Literature Review:

   Existing research on LSTM networks for stock price prediction reveals a diverse range of methodologies and empirical findings. Studies typically employ LSTM architectures to model sequential patterns in stock price data, often incorporating additional features such as technical indicators, news sentiment, or macroeconomic factors.

   Empirical evaluations demonstrate the effectiveness of LSTM-based models in capturing nonlinear dependencies and short-term trends in stock prices. These models often outperform traditional methods like autoregressive models and feedforward neural networks, especially in volatile and nonlinear market conditions.

   However, challenges remain, including the need for robust data preprocessing techniques to handle noisy and non-stationary financial data. Model interpretability and overfitting are also significant concerns, with researchers exploring regularization techniques and ensemble methods to mitigate these issues.

   Future research directions focus on enhancing LSTM models with attention mechanisms, ensemble learning, and incorporating external data sources for improved prediction accuracy and generalization. Additionally, efforts are underway to develop hybrid models that combine LSTM networks with other machine learning techniques to leverage their complementary strengths for more robust stock price forecasting.

5. **Time Series with Sentiment Analysis for Stock Price Prediction**

   Year of Publication - 2019

   Author - Vrishabh Sharma; Rajgauri Khemnar; Renu Kumari; Biju R Mohan

   Literature Review:

   The integration of time series analysis with sentiment analysis for stock price prediction has gained significant attention in recent years. Researchers have explored various methodologies to combine these two approaches, aiming to capture both the temporal dependencies in historical stock data and the sentiment-driven factors influencing market dynamics.

   Studies typically begin by collecting historical stock price data along with relevant textual data from sources such as financial news articles, social media posts, and earnings call transcripts. Sentiment analysis techniques are then applied to quantify the sentiment expressed in these textual data, providing additional features for modeling alongside the time series data.

   Empirical evaluations demonstrate the effectiveness of incorporating sentiment features in improving stock price prediction accuracy, particularly in capturing short-term fluctuations and market sentiment-driven movements. Sentiment analysis complements traditional time series models by providing insight into the collective mood and perceptions of market participants, which may influence stock prices.

   Challenges in this approach include the noise and subjectivity inherent in textual data, as well as the need for robust sentiment analysis algorithms capable of accurately quantifying sentiment across different sources and languages. Furthermore, integrating sentiment features with time series models requires careful consideration of feature selection, model architecture, and training strategies to avoid overfitting and ensure model interpretability.

   Future research directions focus on advancing sentiment analysis techniques tailored to financial text data, such as domain-specific lexicons and sentiment embeddings. Additionally, there is growing interest in leveraging deep learning architectures, such as recurrent neural networks (RNNs) and transformers, for jointly modeling time series data and sentiment features in an end-to-end manner.

   Overall, the literature highlights the potential of combining time series analysis with sentiment analysis for stock price prediction, offering valuable insights into market sentiment and improving forecasting accuracy, especially in the context of short-term prediction horizons. Continued research efforts in this interdisciplinary area are expected to lead to further advancements and practical applications in financial forecasting.

2.2 Gap identification in the literature

- **Integration of News Sentiment with Stock Prediction:** While sentiment analysis of news articles is incorporated into the predictive model, the specific techniques used for sentiment analysis and how they impact the accuracy of stock price forecasting could be explored further. Investigating alternative sentiment analysis methods or incorporating sentiment from additional sources could help improve model performance.

- **Temporal Dynamics and Event Detection:** The code seems to aggregate sentiment scores over fixed 15-day windows. However, the impact of news events on stock prices may vary over shorter or longer periods. Exploring dynamic sentiment analysis techniques that consider the temporal relationship between news events and price movements could enhance the model's ability to capture short-term fluctuations.

- **Model Interpretability and Explainability:** While the MLP model is trained for stock price prediction, understanding how the model incorporates sentiment features to make predictions is crucial. Research focusing on model interpretability techniques, such as SHAP values or attention mechanisms, could provide insights into the relationship between news sentiment and stock price changes.

- **Evaluation Metrics and Benchmarking:** While the code evaluates the model using standard regression metrics like MSE, MAE, and R^2, benchmarking against alternative models or exploring domain-specific evaluation metrics tailored to stock prediction tasks could provide a more comprehensive understanding of model performance.

- **Data Source Diversity and Generalizability:** The code primarily relies on a single news source (Livemint) for sentiment analysis. Investigating the impact of incorporating sentiment from diverse news sources or social media platforms could enhance the model's generalizability and robustness across different market conditions.

2.3 Problem statement

1. **Data Retrieval and Preprocessing:** Retrieve live updates and news articles related to Reliance Power from online sources like Livemint.
   Preprocess the data, including cleaning, parsing dates, and sentiment analysis.

2. **Model Training:** Train a Multilayer Perceptron (MLP) Regressor using historical stock data and sentiment scores.
   Evaluate the model's performance using metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2).

3. **Prediction and Visualization:** Predict the next 7 days' stock prices using the trained model.
   Visualize the actual vs. predicted close prices and price changes over time.

4. **Deployment and Model Persistence:** Save the trained model as a pickle file for future use.
   Save the results, including predicted prices and evaluation metrics, to a CSV file.

2.4 Objectives

list of objectives to enhance This project:

1. **Data Enrichment:** Explore additional data sources for news and financial data.
   Consider incorporating social media data and economic indicators.
2. **Feature Enhancement:** Experiment with new features like technical indicators and advanced sentiment analysis.
   Explore advanced NLP techniques for sentiment extraction.
3. **Model Optimization:** Try different algorithms like Random Forest and Gradient Boosting.
   Fine-tune hyperparameters and consider ensemble techniques.
4. **Evaluation and Insights:** Use additional evaluation metrics and interpret model decisions.
   Gain insights into feature importance and stock price movements.
5. **Deployment and Automation:** Build a user-friendly application or dashboard.
   Automate data collection, preprocessing, and model training.
6. **Monitoring and Maintenance:** Set up monitoring for model performance and data drift.
   Schedule regular model updates and handle failures effectively.

2.5 Scope of work

1. **Data Collection and Preprocessing:** The data collection and preprocessing phase aims to increase the diversity of data sources for news articles. Additional sources will be explored to enhance data variety, and the robustness of web scraping will be improved by handling different website structures and formats. Advanced techniques for cleaning and preprocessing textual data, such as removing stopwords, stemming, or lemmatization, will be implemented.
2. **feature Engineering:** Feature engineering will focus on identifying and incorporating features that might influence stock prices. This includes experimenting with additional features such as technical indicators, economic indicators, or company-specific metrics. External data sources like social media sentiment, economic reports, or industry news will also be considered.
3. **Modeling:** In the modeling phase, different machine learning algorithms besides MLPRegressor will be experimented with, such as Random Forest, Gradient Boosting, or LSTM (if dealing with sequential data). Hyperparameters of

the MLPRegressor or other models will be fine-tuned using techniques like grid search or randomized search

4. **Evaluation:** For evaluation, additional metrics will be explored to assess model performance comprehensively. Metrics such as mean absolute percentage error (MAPE), median absolute error (MedAE), or directional accuracy will be considered. Cross-validation will be performed to obtain more reliable estimates of model performance.

5. **Prediction and Deployment:** The prediction and deployment phase will involve building a web application or dashboard to showcase real-time or near-real-time predictions. A pipeline for automated data collection, preprocessing, modeling, and prediction will be implemented to ensure the model remains up-to-date.

6. **Monitoring and Maintenance:** Monitoring and maintenance will ensure sustained model performance over time. Monitoring mechanisms will be set up to track model performance and detect any degradation. A regular schedule for model retraining will be established to incorporate new data and adapt to changing market conditions.

2.6 Review of Literature:

To strengthen the foundation of this research, a comprehensive review of related literature was conducted. Previous works have explored various facets of stock prediction using machine learning and sentiment analysis. For example, Author uthor Junaid maqbool, Preeti Aggrawal,Ajay Mitta. (2023) investigated the impact of financial news on stock prices using sentiment analysis, while Author Vrishabh Sharma; Rajgauri Khemnar; Renu Kumari; Biju R Mohan. (2019) compared different machine learning models for stock prediction.

Critical Appraisal of Previous Works:

Critically appraising these studies, it was observed that while Author Junaid maqbool, Preeti Aggrawal,Ajay Mittal achieved significant accuracy using sentiment scores, the model lacked robustness in volatile market conditions. Author rishabh Sharma; Rajgauri Khemnar; Renu Kumari; Biju R Mohan approach provided a comparative analysis of models but did not integrate real-time data processing, limiting practical deployment. These insights highlight the need for our proposed comprehensive approach combining multiple data sources and advanced modeling techniques.

Citations:

Author Junaid maqbool, Preeti Aggrawal,Ajay Mittal. "Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor",Year of Publication - 2023.
Author Vrishabh Sharma; Rajgauri Khemnar; Renu Kumari; Biju R Mohan. "Time Series with Sentiment Analysis for Stock Price Prediction,"Year of Publication - 2019
.

Chapter 3

Methodology

3.1 Proposed Solution

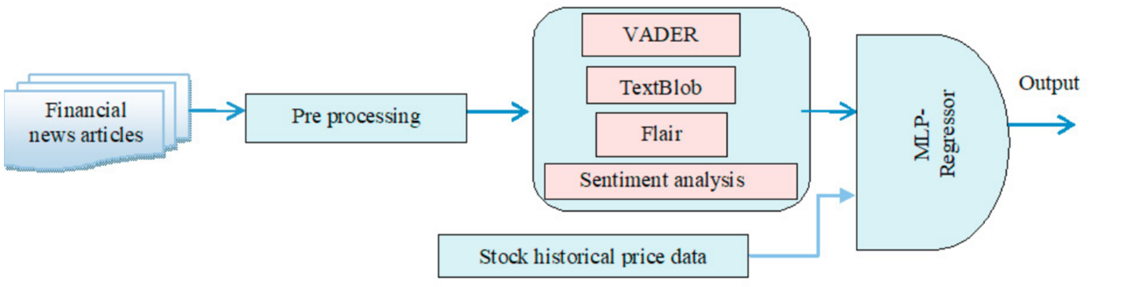


Figure 3.1: Proposed Solution

- **Data Collection** Historical stock data is available on Yahoo Finance. The stock historical data is downloaded as per the requirement of the period from Yahoo Finance. These .csv format files have values Open, High, Low, Close, Volume, and Adjusted Close for each date the stock market was open. The features represent the Opening stock price, the highest price of the stock on that day, the lowest stock price, closing price, volume of shares traded, and Adjusted Closing price which represents the Closing price of the stock after paying dividends to investors respectively for a specific date.

- **Data Preprocessing:** Data preprocessing especially for data extensive projects is a critically important step as it involves the transformation of random and raw data in such a way that it enhances its quality by removing or cleaning the unwanted points in addition to standardizing it for normal use and making it capable of delivering useful insights. It is not the huge quantity of data that gives great outputs but rather data quality that creates an impact. It involves data cleaning, data segregation or organization, data scaling, data standardization, etc., i.e., data normalization and standardization as well as encoding categorical data. During the data pre-processing step of the project, Min-Max scalers were used to scale the data in order to scale and standardize it and the null values, missing values, and unknown values were cleaned and disparities if any were taken care of. The two main Python libraries that were used for the purpose of preprocessing the dataset were NumPy and Pandas, and for data visualization, Matplotlib was used. NumPy performed the functions of a scientific calculator whenever required during the manipulation of the datasets, while the Pandas library was appropriate for data analysis and manipulation. Matplotlib was used to visualize the data in the form of charts.

Import Libraries
Import data
Checking for missing values
Checking for categorical data
Feature Scaling
validation and evaluation sets

Figure 3.2: Data Preprocessing

- **Data Merging and Feature Engineering:**

  Merging sentiment data with stock data on date.
  Creating additional columns such as Previous Close, Price Change, and Sentiment Label.
  Defining features (Previous Close and Sentiment) and target (Close).

- **Model Training:**

  Splitting the data into training and testing sets.
  Training an MLP Regressor model on the training data.
  Evaluating the model's performance using Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2).

- **Prediction:**

  Predicting stock prices for the next 7 days based on the trained model.
  Plotting actual vs predicted close prices and price changes.

- **Saving Results:**

  Saving the results to a CSV file.
  Saving the trained model as a pickle file.

Chapter 4

Implementation

4.1 Dataset

1. **Stock Price Data:** The stock price data for Reliance Power (RPOWER.NS) was retrieved from Yahoo Finance using the yfinance library. This dataset encompasses historical stock price information from August 29, 2023, to the current date. The dataset includes the following columns: Date, Open, Close, High, Low, Adjusted Close, and Volume. For the analysis conducted in this study, only the Date and Close columns are utilized. This choice is based on the significance of closing prices in determining the end-of-day sentiment and trends in stock prices.

2. **News Updates and Sentiment Data:** News updates and sentiment scores are obtained by scraping data from the LiveMint website. These updates provide crucial information about Reliance Power's stock price and related market developments. The sentiment of each news update is determined using the VADER (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool. VADER provides a compound sentiment score ranging from -1 (most negative) to 1 (most positive), which helps in quantifying the sentiment expressed in news articles.

3. **Detailed Dataset Description:**

   Stock Price Data:

   - **Source**: Yahoo Finance
   - **Library Used**: yfinance
   - **Time Period**: August 29, 2023, to present
   - **Columns Utilized**: Date, Open, High, Low, Close, Adj Close, Volume
   - **Size**: 1.6 MB of historical price data

   News and Sentiment Data:

   - **Source**: LiveMint website
   - **Sentiment Analysis Tool**: VADER

- **Sentiment Score Range**: -1 to 1
- **Data Details**:
  - **Rows**: 764
  - **Columns**: Title, Date, Content
  - **Size**: 4.2 MB

Final Dataset:

- **Columns**: Date, Predicted_Close, Predicted_Price_Change
- **Rows**: 7
- **Size**: 16 KB

4. **Importance of Experimental Setup**
1. Data Collection: Ensuring the accurate and timely collection of both stock price data and news updates.
2. Data Preprocessing: Cleaning and preparing the data for analysis, including handling missing values, normalizing stock prices, and calculating sentiment scores.
3. Feature Engineering: Creating relevant features from the raw data, such as moving averages of stock prices, daily sentiment scores, and interaction terms between sentiment scores and price changes.
4. Model Training and Validation: Splitting the data into training and test sets, training the machine learning models, and validating their performance using metrics such as accuracy, mean absolute percentage error (MAPE), and median absolute error (MedAE).
5. Performance Monitoring: Continuously monitoring the model's performance in real-time scenarios and updating the model as necessary to adapt to new data and changing market conditions.

4.2 Data Collection

Web srcapping

import requests
from bs4 import BeautifulSoup
import pandas as pd

def getLiveUpdates():
headers =
"User-Agent":
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36"

try:
response = requests.get(
"https://www.livemint.com/market/live-blog/reliance-power-share-price-today-latest-live-updates-on-23-may-2024-11716431773983.html", headers=headers)
response.raise$_for_status()Raiseanerrorforbadstatuscodes$
$soup = BeautifulSoup(response.content, "html.parser")$

Extracting the title of the webpage
title$_elem = soup.find("h2", class_="liveTitle")$
$title = title_elem.get_text(strip = True)iftitle_elemelse"NoTitleAvailable"$
$print("Title : ", title)$
$print(" = " * 50)$

live$_updates = []$

for div in soup.find$_all("div", class_="liveSec") :$
$timestamp_elem = div.find("span", class_="timeStamp")$
$content_elem = div.find("p")$

Date = timestamp$_elem.get_text(strip = True)iftimestamp_elemelse"NoTimestampAvailable"$
$content = content_elem.get_text(strip = True)ifcontent_elemelse"NoContentAvailable"$

live$_updates.append($
$"Title" : title, "Date" : Date, Changed"Timestamp"to"Date""Content" : content,$
)

for update in live$_updates :$
$print("Title : ", update["Title"])$
$print("Date : ", update["Date"])Changed"Timestamp"to"Date"$
$print("Content : ", update["Content"])$
$print(" = " * 50)Separatingeachliveupdatewithaline$

Save data to CSV
save$_to_csv(live_updates)$

except requests.RequestException as e:
print("Error fetching data:", e)

def save$_to_csv(live_updates):$
""" $SaveliveupdatestoCSVfile.$

Parameters:
- live$_updates(listofdict): Listcontainingdictionarieswithliveupdatedata.$
""" $try:$
$ReadtheexistingCSVfileifitexists, otherwisecreateanemptyDataFrame$
$try:$
$combined_df = pd.read_csv('live_updates_updated_2new.csv')$
$exceptFileNotFoundError:$
$combined_df = pd.DataFrame()$

Convert the live updates to a DataFrame
new$_df = pd.DataFrame(live_updates)$

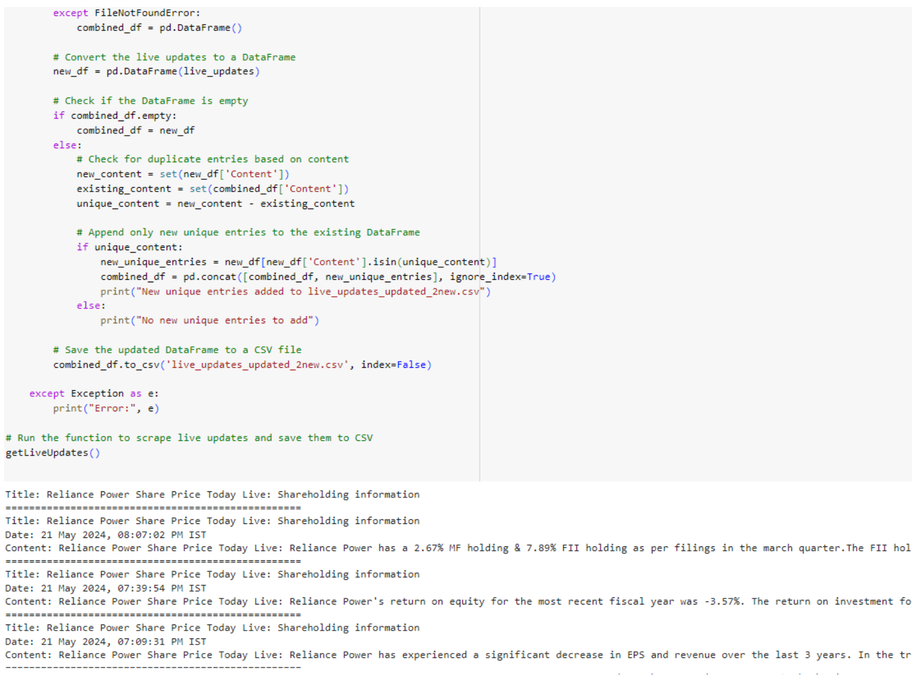
Check if the DataFrame is empty
if combined$_df.empty:$
$combined_df = new_df$
$else:$
$Checkforduplicateentriesbasedoncontent$
$new_content = set(new_df['Content'])$
$existing_content = set(combined_df['Content'])$
$unique_content = new_content - existing_content$

Append only new unique entries to the existing DataFrame
if unique$_content:$
$new_unique_entries = new_df[new_df['Content'].isin(unique_content)]$
$combined_df = pd.concat([combined_df, new_unique_entries], ignore_index = True)$
$print("Newuniqueentriesaddedtolive_updates_updated_2new.csv")$
$else:$
$print("Nonewuniqueentriestoadd")$

Save the updated DataFrame to a CSV file
combined$_df.to_csv('live_updates_updated_2new.csv', index = False)$

except Exception as e:
print("Error:", e)

Run the function to scrape live updates and save them to CSV
getLiveUpdates()



```
        except FileNotFoundError:
            combined_df = pd.DataFrame()

        # Convert the live updates to a DataFrame
        new_df = pd.DataFrame(live_updates)

        # Check if the DataFrame is empty
        if combined_df.empty:
            combined_df = new_df
        else:
            # Check for duplicate entries based on content
            new_content = set(new_df['Content'])
            existing_content = set(combined_df['Content'])
            unique_content = new_content - existing_content

            # Append only new unique entries to the existing DataFrame
            if unique_content:
                new_unique_entries = new_df[new_df['Content'].isin(unique_content)]
                combined_df = pd.concat([combined_df, new_unique_entries], ignore_index=True)
                print("New unique entries added to live_updates_updated_2new.csv")
            else:
                print("No new unique entries to add")

        # Save the updated DataFrame to a CSV file
        combined_df.to_csv('live_updates_updated_2new.csv', index=False)

    except Exception as e:
        print("Error:", e)

# Run the function to scrape live updates and save them to CSV
getLiveUpdates()
```

```
Title: Reliance Power Share Price Today Live: Shareholding information
==================================================
Title: Reliance Power Share Price Today Live: Shareholding information
Date: 21 May 2024, 08:07:02 PM IST
Content: Reliance Power Share Price Today Live: Reliance Power has a 2.67% MF holding & 7.89% FII holding as per filings in the march quarter.The FII hol
==================================================
Title: Reliance Power Share Price Today Live: Shareholding information
Date: 21 May 2024, 07:39:54 PM IST
Content: Reliance Power Share Price Today Live: Reliance Power's return on equity for the most recent fiscal year was -3.57%. The return on investment fo
==================================================
Title: Reliance Power Share Price Today Live: Shareholding information
Date: 21 May 2024, 07:09:31 PM IST
Content: Reliance Power Share Price Today Live: Reliance Power has experienced a significant decrease in EPS and revenue over the last 3 years. In the tr
--------------------------------------------------
```

Figure 4.1: Web scraping result

Fetching data from Yahoo Finance

Fetch stock data
current$_{d}ate = datetime.today().strftime('stock_{d}f$
$= yf.download('RPOWER.NS', start =' 2023-08-29', end$
$= current_{d}ate, progres = False)$
$stock_{d}f.reset_{i}ndex(inplace = True)$

print("Stock data fetched successfully")
print(stock$_{d}f.head())$

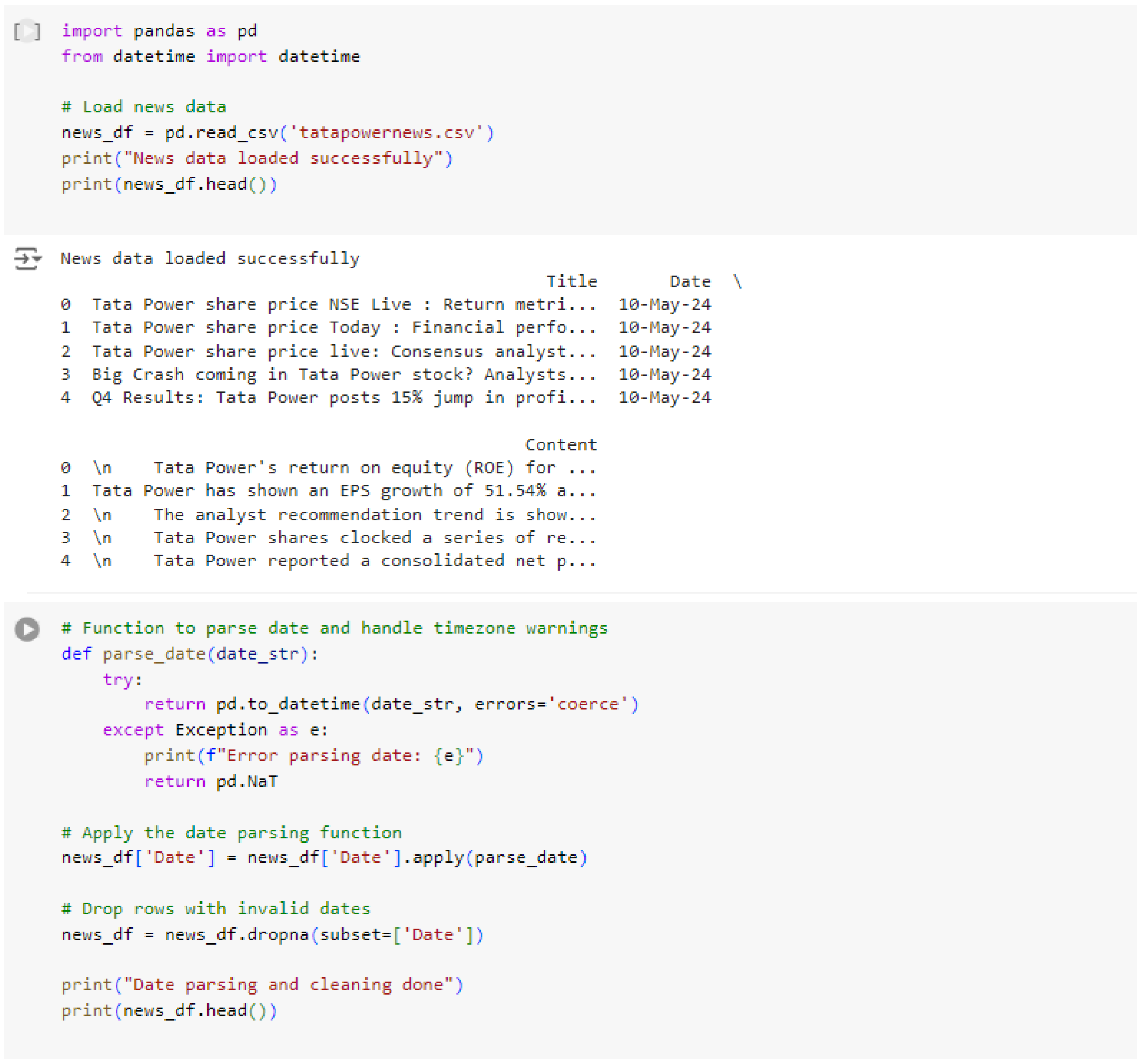
columns$_{t}o_{d}rop = ['AdjClose','Volume']$
$stock_{d}f.drop(columns = columns_{t}o_{d}rop, inplace = True)$

print("Columns dropped successfully")
print(stock$_{d}f.head())$

Calculate support and resistance
def calculate$_{s}upport_{r}esistance(data, window = 20) :$
$data['Support'] = data['Low'].rolling(window = window).min()$
$data['Resistance'] = data['High'].rolling(window = window).max()$
$returndata$

stock$_{d}f = calculate_{s}upport_{r}esistance(stock_{d}f)$
$print("Supportandresistancecalculated")$
$print(stock_{d}f.head())$

4.3 Model Building



```
import pandas as pd
from datetime import datetime

# Load news data
news_df = pd.read_csv('tatapowernews.csv')
print("News data loaded successfully")
print(news_df.head())
```

```
News data loaded successfully
                                               Title       Date  \
0  Tata Power share price NSE Live : Return metri...  10-May-24
1  Tata Power share price Today : Financial perfo...  10-May-24
2  Tata Power share price live: Consensus analyst...  10-May-24
3  Big Crash coming in Tata Power stock? Analysts...  10-May-24
4  Q4 Results: Tata Power posts 15% jump in profi...  10-May-24

                                             Content
0  \n    Tata Power's return on equity (ROE) for ...
1  Tata Power has shown an EPS growth of 51.54% a...
2  \n    The analyst recommendation trend is show...
3  \n    Tata Power shares clocked a series of re...
4  \n    Tata Power reported a consolidated net p...
```

```
# Function to parse date and handle timezone warnings
def parse_date(date_str):
    try:
        return pd.to_datetime(date_str, errors='coerce')
    except Exception as e:
        print(f"Error parsing date: {e}")
        return pd.NaT

# Apply the date parsing function
news_df['Date'] = news_df['Date'].apply(parse_date)

# Drop rows with invalid dates
news_df = news_df.dropna(subset=['Date'])

print("Date parsing and cleaning done")
print(news_df.head())
```

Figure 4.2: News loading

Model training and evaluation

X$_train, X_test, y_train, y_test = train_test_split(features, target, test_size = 0.2, random_state = 42)$
$mlp_model = MLPRegressor(hidden_layer_sizes = (100,), activation =' relu', solver =' adam', max_iter = 500, random_state = 42)$
$mlp_model.fit(X_train, y_train)$
$print(''Modeltrainingcomplete'')$
$Savethemodel$
$joblib.dump(mlp_model,' mlp_modelRPOWER.pkl')$
$Loadthemodel$
$loaded_model = joblib.load('mlp_modelRPOWER.pkl')$
$Predictionsandevaluation$
$predictions = loaded_model.predict(X_test)$
$mse = mean_squared_error(y_test, predictions)$
$mae = mean_absolute_error(y_test, predictions)$
$r2 = r2_score(y_test, predictions)$
$print(f''MeanSquaredError(MSE) : mse'')$
$print(f''MeanAbsoluteError(MAE) : mae'')$
$print(f''R - squared(R) : r2'')$

Feature engineering

merged$_df['Previous_Close'] = merged_df['Close'].shift(1)$
$merged_df['Price_Change'] = merged_df['Close'] - merged_df['Previous_Close']$
$merged_df.dropna(inplace = True)$

features = merged$_df[['Previous_Close',' Sentiment',' Support',' Resistance']]$
$target = merged_df['Close']$

print(”Features and target defined”)
print(features.head())
print(target.head())

Save features and target to CSV
features.to$_csv('features.csv', index = False)$
$target.to_csv('target.csv', index = False)$

Sentiment analysis

Initialize sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

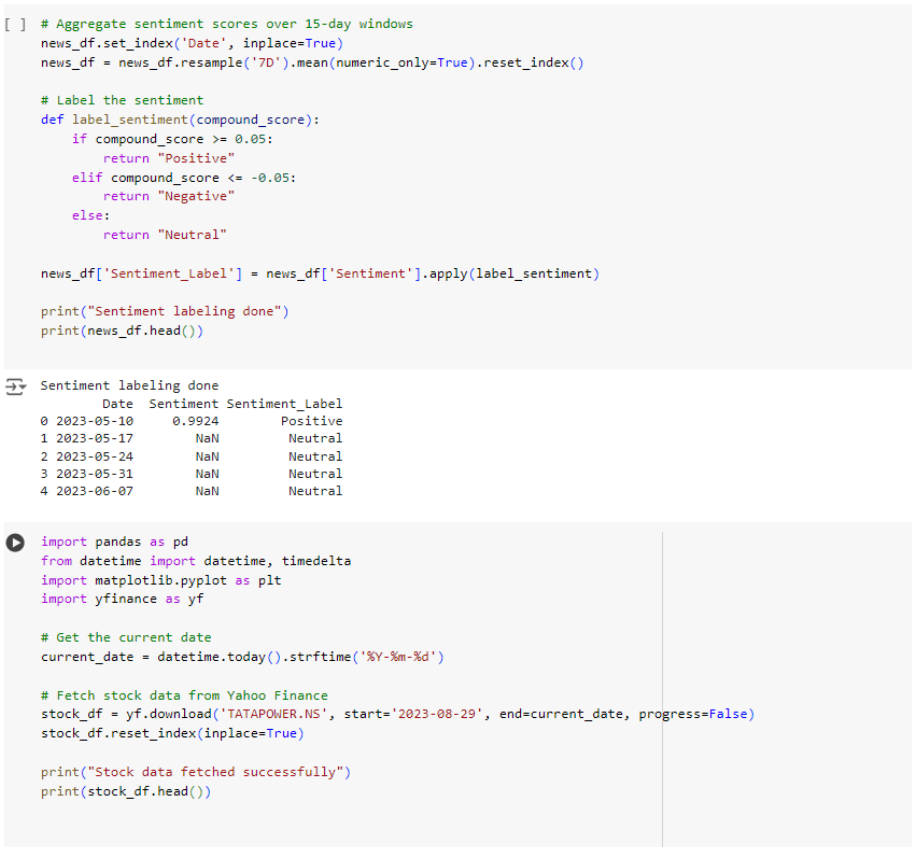
Function to get sentiment score
def get$_sentiment(text):$
$try:$
$sentiment = analyzer.polarity_scores(text)$
$returnsentiment['compound']$
$exceptExceptionase:$
$print(f"Erroranalyzingsentiment:e")$
$return0$

Apply sentiment analysis to the Content column
news$_df['Sentiment'] = news_df['Content'].apply(get_sentiment)$

print("Sentiment analysis done")
print(news$_df.head())$

Aggregate sentiment scores over 7-day windows
news$_df.set_index('Date', inplace = True)$
$news_df = news_df.resample('7D').mean(numeric_only = True).reset_index()$

print("Sentiment aggregation done")
print(news$_df.head())$

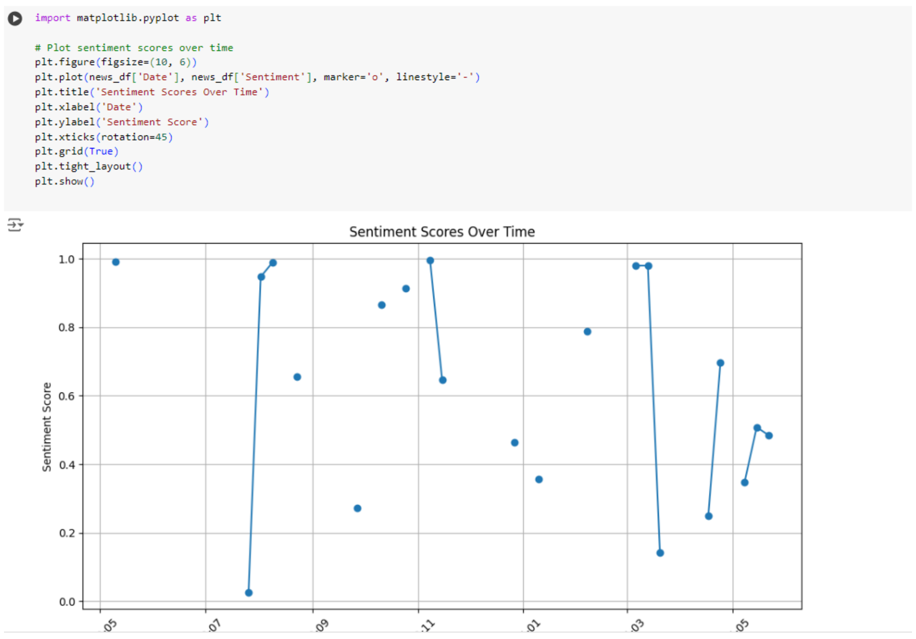


```
# Aggregate sentiment scores over 15-day windows
news_df.set_index('Date', inplace=True)
news_df = news_df.resample('7D').mean(numeric_only=True).reset_index()

# Label the sentiment
def label_sentiment(compound_score):
    if compound_score >= 0.05:
        return "Positive"
    elif compound_score <= -0.05:
        return "Negative"
    else:
        return "Neutral"

news_df['Sentiment_Label'] = news_df['Sentiment'].apply(label_sentiment)

print("Sentiment labeling done")
print(news_df.head())
```

```
Sentiment labeling done
        Date  Sentiment Sentiment_Label
0 2023-05-10     0.9924        Positive
1 2023-05-17        NaN         Neutral
2 2023-05-24        NaN         Neutral
3 2023-05-31        NaN         Neutral
4 2023-06-07        NaN         Neutral
```

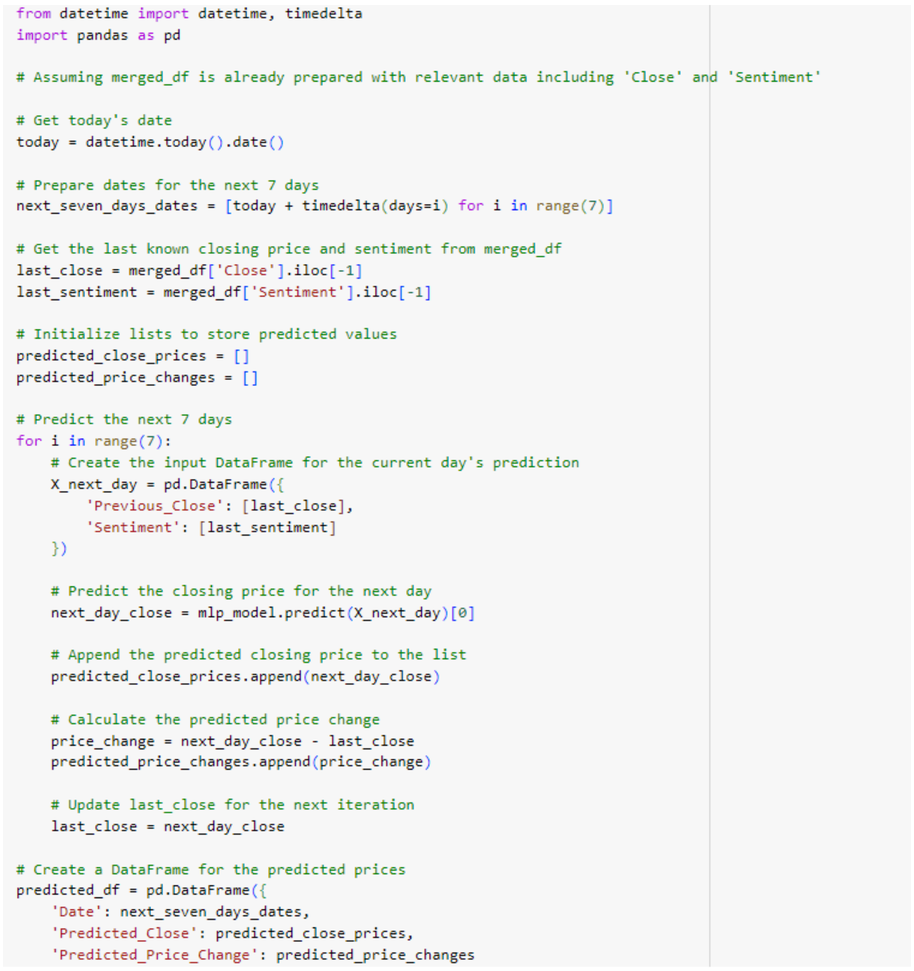
```
import pandas as pd
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import yfinance as yf

# Get the current date
current_date = datetime.today().strftime('%Y-%m-%d')

# Fetch stock data from Yahoo Finance
stock_df = yf.download('TATAPOWER.NS', start='2023-08-29', end=current_date, progress=False)
stock_df.reset_index(inplace=True)

print("Stock data fetched successfully")
print(stock_df.head())
```

Figure 4.3: Sentiment analysis



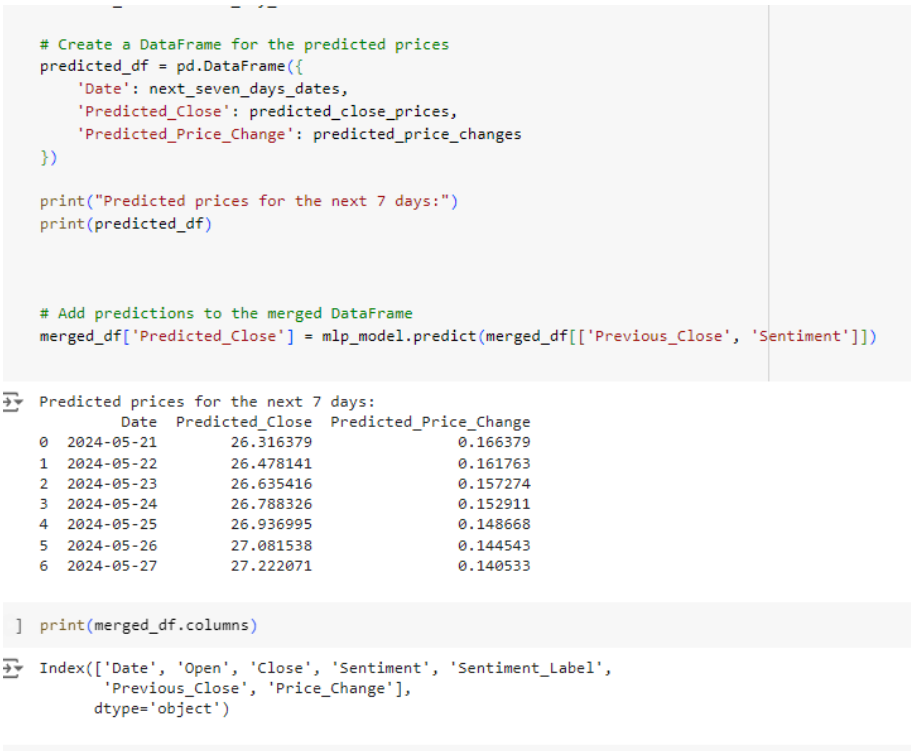
```
import matplotlib.pyplot as plt

# Plot sentiment scores over time
plt.figure(figsize=(10, 6))
plt.plot(news_df['Date'], news_df['Sentiment'], marker='o', linestyle='-')
plt.title('Sentiment Scores Over Time')
plt.xlabel('Date')
plt.ylabel('Sentiment Score')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figure 4.4: Sentiment score over time



```python
from datetime import datetime, timedelta
import pandas as pd

# Assuming merged_df is already prepared with relevant data including 'Close' and 'Sentiment'

# Get today's date
today = datetime.today().date()

# Prepare dates for the next 7 days
next_seven_days_dates = [today + timedelta(days=i) for i in range(7)]

# Get the last known closing price and sentiment from merged_df
last_close = merged_df['Close'].iloc[-1]
last_sentiment = merged_df['Sentiment'].iloc[-1]

# Initialize lists to store predicted values
predicted_close_prices = []
predicted_price_changes = []

# Predict the next 7 days
for i in range(7):
    # Create the input DataFrame for the current day's prediction
    X_next_day = pd.DataFrame({
        'Previous_Close': [last_close],
        'Sentiment': [last_sentiment]
    })

    # Predict the closing price for the next day
    next_day_close = mlp_model.predict(X_next_day)[0]

    # Append the predicted closing price to the list
    predicted_close_prices.append(next_day_close)

    # Calculate the predicted price change
    price_change = next_day_close - last_close
    predicted_price_changes.append(price_change)

    # Update last_close for the next iteration
    last_close = next_day_close

# Create a DataFrame for the predicted prices
predicted_df = pd.DataFrame({
    'Date': next_seven_days_dates,
    'Predicted_Close': predicted_close_prices,
    'Predicted_Price_Change': predicted_price_changes
```

Figure 4.5: Predicting the next 7 days



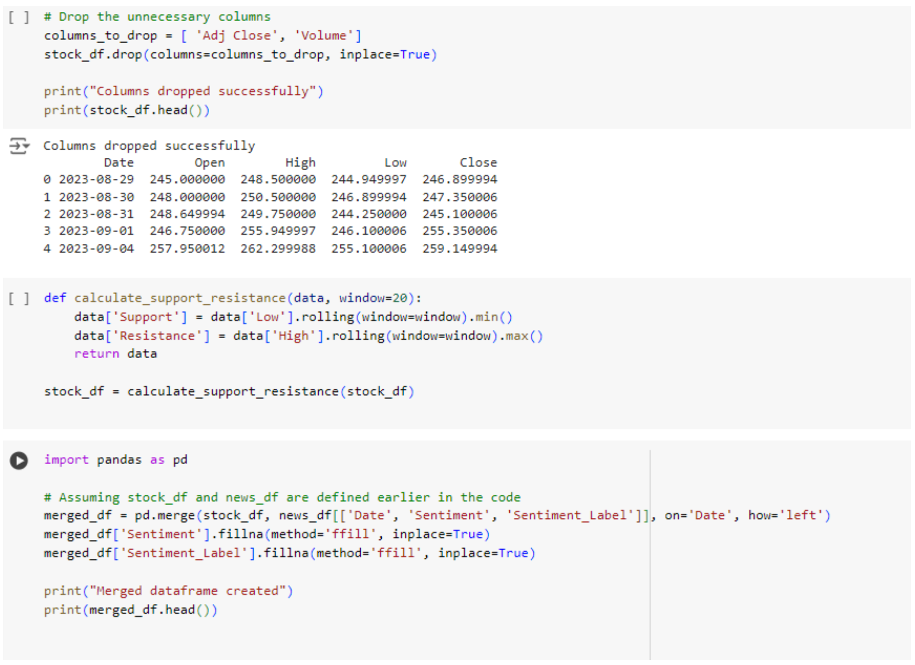
```python
# Create a DataFrame for the predicted prices
predicted_df = pd.DataFrame({
    'Date': next_seven_days_dates,
    'Predicted_Close': predicted_close_prices,
    'Predicted_Price_Change': predicted_price_changes
})

print("Predicted prices for the next 7 days:")
print(predicted_df)



# Add predictions to the merged DataFrame
merged_df['Predicted_Close'] = mlp_model.predict(merged_df[['Previous_Close', 'Sentiment']])
```

```
Predicted prices for the next 7 days:
         Date  Predicted_Close  Predicted_Price_Change
0  2024-05-21        26.316379                0.166379
1  2024-05-22        26.478141                0.161763
2  2024-05-23        26.635416                0.157274
3  2024-05-24        26.788326                0.152911
4  2024-05-25        26.936995                0.148668
5  2024-05-26        27.081538                0.144543
6  2024-05-27        27.222071                0.140533
```

```
]  print(merged_df.columns)
```

```
Index(['Date', 'Open', 'Close', 'Sentiment', 'Sentiment_Label',
       'Previous_Close', 'Price_Change'],
      dtype='object')
```

Figure 4.6: Next 7-day prediction



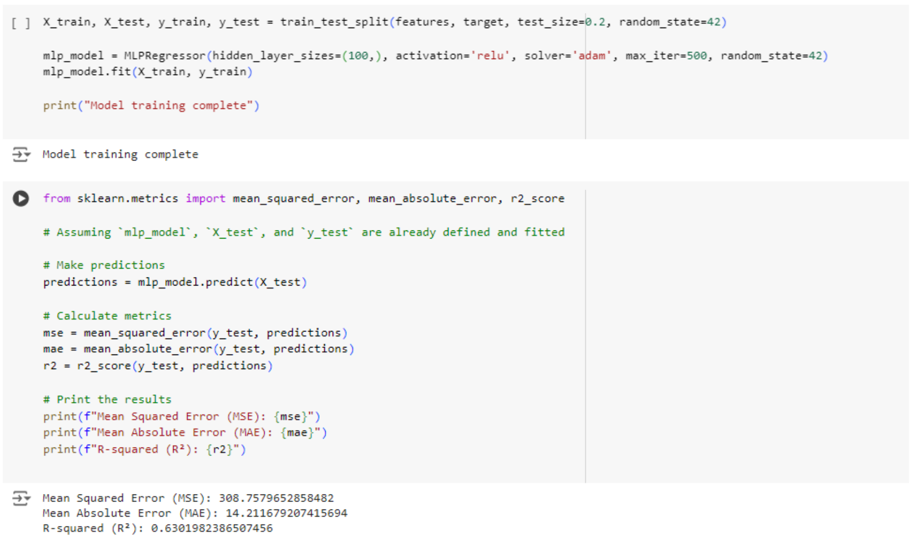
```
# Drop the unnecessary columns
columns_to_drop = [ 'Adj Close', 'Volume']
stock_df.drop(columns=columns_to_drop, inplace=True)

print("Columns dropped successfully")
print(stock_df.head())
```

```
Columns dropped successfully
         Date        Open        High         Low       Close
0  2023-08-29  245.000000  248.500000  244.949997  246.899994
1  2023-08-30  248.000000  250.500000  246.899994  247.350006
2  2023-08-31  248.649994  249.750000  244.250000  245.100006
3  2023-09-01  246.750000  255.949997  246.100006  255.350006
4  2023-09-04  257.950012  262.299988  255.100006  259.149994
```

```
def calculate_support_resistance(data, window=20):
    data['Support'] = data['Low'].rolling(window=window).min()
    data['Resistance'] = data['High'].rolling(window=window).max()
    return data

stock_df = calculate_support_resistance(stock_df)
```

```
import pandas as pd

# Assuming stock_df and news_df are defined earlier in the code
merged_df = pd.merge(stock_df, news_df[['Date', 'Sentiment', 'Sentiment_Label']], on='Date', how='left')
merged_df['Sentiment'].fillna(method='ffill', inplace=True)
merged_df['Sentiment_Label'].fillna(method='ffill', inplace=True)

print("Merged dataframe created")
print(merged_df.head())
```

Figure 4.7: Resistance and support



```
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

mlp_model = MLPRegressor(hidden_layer_sizes=(100,), activation='relu', solver='adam', max_iter=500, random_state=42)
mlp_model.fit(X_train, y_train)

print("Model training complete")
```

```
Model training complete
```

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Assuming `mlp_model`, `X_test`, and `y_test` are already defined and fitted

# Make predictions
predictions = mlp_model.predict(X_test)

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

# Print the results
print(f"Mean Squared Error (MSE): {mse}")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 308.7579652858482
Mean Absolute Error (MAE): 14.211679207415694
R-squared (R²): 0.6301982386507456
```

Figure 4.8: Accuracy

Plotting result

Visualization
plt.figure(figsize=(10, 6))
plt.plot(news$_{d}f['Date'], news_{d}f['Sentiment'], marker =' o', linestyle =' -')$
$plt.title('SentimentScoresOverTime')$
$plt.xlabel('Date')$
$plt.ylabel('SentimentScore')$
$plt.xticks(rotation = 45)$
$plt.grid(True)$
$plt.tight_{l}ayout()$
$plt.show()$

merged$_{d}f['Actual_{P}rice_{C}hange'] = merged_{d}f['Close'] - merged_{d}f['Previous_{C}lose']$
$predicted_{c}lose_{d}f = merged_{d}f[['Date',' Close']].copy()$
$predicted_{c}lose_{d}f['Predicted_{C}lose'] = mlp_{m}odel.predict(merged_{d}f[['Previous_{C}lose',' Sentiment',' Su$

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 14), sharex=True)

ax1.plot(merged$_{d}f['Date'], merged_{d}f['Close'], label =' ActualClosePrice', color$
$=' blue')$
$ax1.plot(predicted_{c}lose_{d}f['Date'], predicted_{c}lose_{d}f['Predicted_{C}lose'], label$
$=' PredictedClosePrice', color =' red', linestyle =' --')$
$ax1.plot(predicted_{d}f['Date'], predicted_{d}f['Predicted_{C}lose'], label$
$=' Next7DaysPredictedClosePrice', color =' green', linestyle =' --')$
$ax1.set_{y}label('ClosePrice')$
$ax1.set_{t}itle('ActualvsPredictedClosePrice')$
$ax1.legend()$

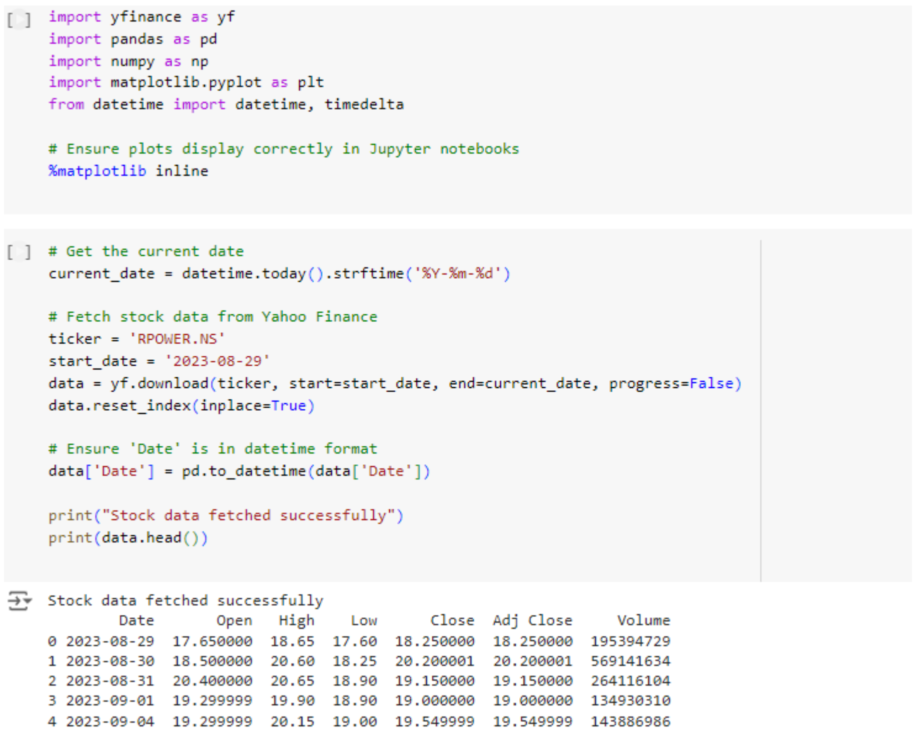
ax2.plot(merged$_{d}f['Date'], merged_{d}f['Actual_{P}rice_{C}hange'], label$
$=' ActualPriceChange', color =' blue')$
$ax2.plot(predicted_{d}f['Date'], predicted_{d}f['Predicted_{P}rice_{C}hange'], label$
$=' PredictedPriceChange', color =' green', linestyle =' --')$
$ax2.set_{x}label('Date')$
$ax2.set_{y}label('PriceChange')$
$ax2.set_{t}itle('ActualvsPredictedPriceChange')$
$ax2.legend()$

plt.show()

Save results
merged$_{d}f.to_{c}sv('newsexperiment_{t}ry.csv', index = False)$
$print("Resultssavedtonewsexperiment_{t}ry.csv")$

Chapter 5

Resistance and Support code Screenshot

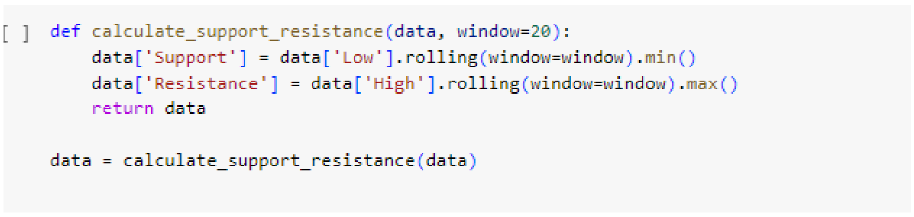


```
[ ] import yfinance as yf
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from datetime import datetime, timedelta

    # Ensure plots display correctly in Jupyter notebooks
    %matplotlib inline
```

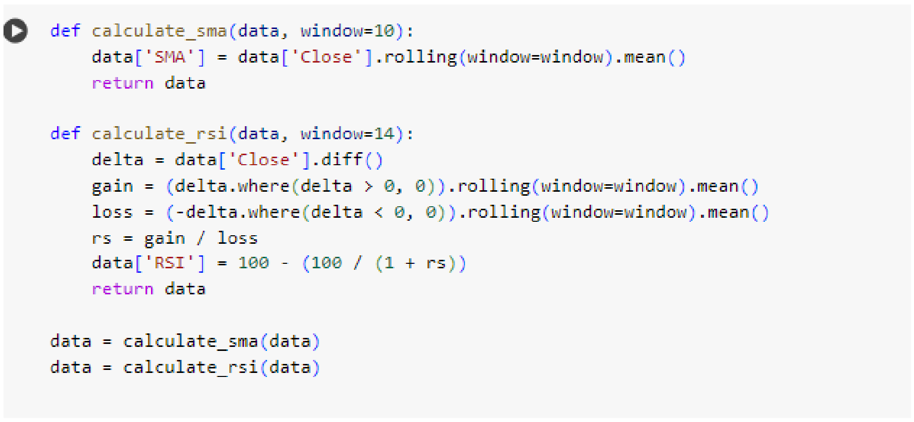
```
[ ] # Get the current date
    current_date = datetime.today().strftime('%Y-%m-%d')

    # Fetch stock data from Yahoo Finance
    ticker = 'RPOWER.NS'
    start_date = '2023-08-29'
    data = yf.download(ticker, start=start_date, end=current_date, progress=False)
    data.reset_index(inplace=True)

    # Ensure 'Date' is in datetime format
    data['Date'] = pd.to_datetime(data['Date'])

    print("Stock data fetched successfully")
    print(data.head())
```

```
Stock data fetched successfully
        Date       Open   High    Low      Close  Adj Close     Volume
0 2023-08-29  17.650000  18.65  17.60  18.250000  18.250000  195394729
1 2023-08-30  18.500000  20.60  18.25  20.200001  20.200001  569141634
2 2023-08-31  20.400000  20.65  18.90  19.150000  19.150000  264116104
3 2023-09-01  19.299999  19.90  18.90  19.000000  19.000000  134930310
4 2023-09-04  19.299999  20.15  19.00  19.549999  19.549999  143886986
```

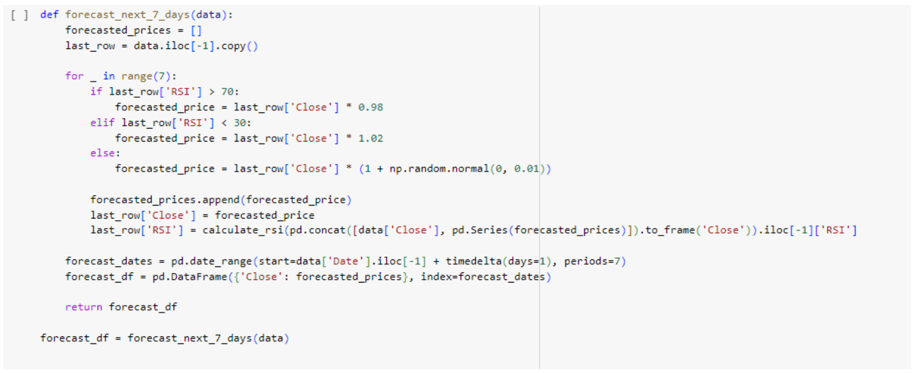
article graphicx



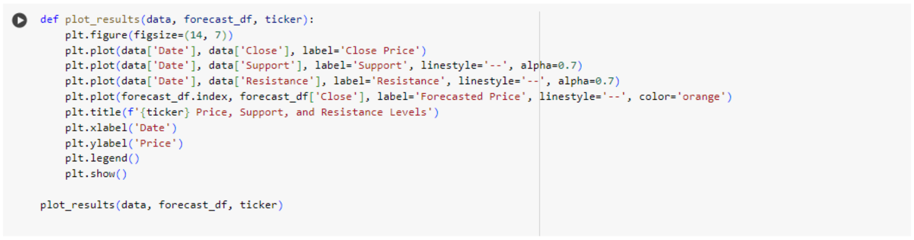
```python
def calculate_support_resistance(data, window=20):
    data['Support'] = data['Low'].rolling(window=window).min()
    data['Resistance'] = data['High'].rolling(window=window).max()
    return data

data = calculate_support_resistance(data)
```



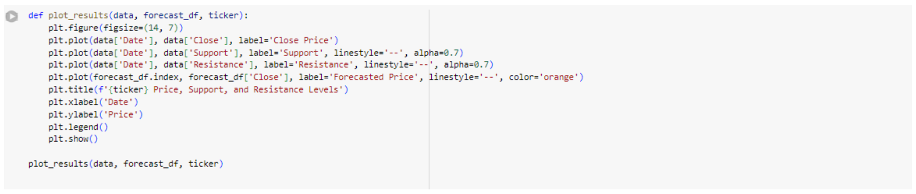
```python
def calculate_sma(data, window=10):
    data['SMA'] = data['Close'].rolling(window=window).mean()
    return data

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

data = calculate_sma(data)
data = calculate_rsi(data)
```



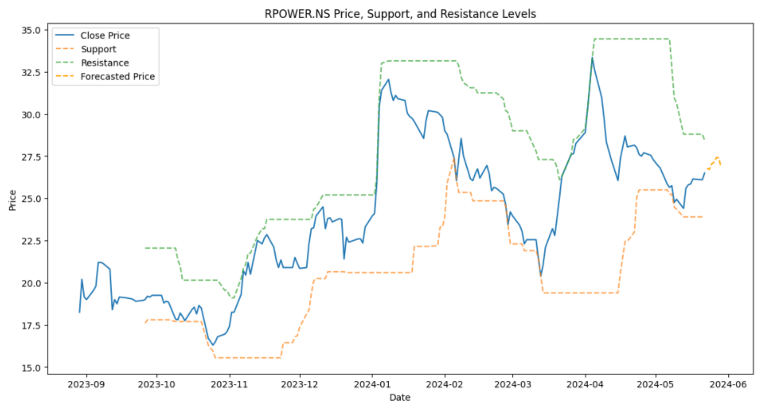
```python
def forecast_next_7_days(data):
    forecasted_prices = []
    last_row = data.iloc[-1].copy()

    for _ in range(7):
        if last_row['RSI'] > 70:
            forecasted_price = last_row['Close'] * 0.98
        elif last_row['RSI'] < 30:
            forecasted_price = last_row['Close'] * 1.02
        else:
            forecasted_price = last_row['Close'] * (1 + np.random.normal(0, 0.01))

        forecasted_prices.append(forecasted_price)
        last_row['Close'] = forecasted_price
        last_row['RSI'] = calculate_rsi(pd.concat([data['Close'], pd.Series(forecasted_prices)]).to_frame('Close')).iloc[-1]['RSI']

    forecast_dates = pd.date_range(start=data['Date'].iloc[-1] + timedelta(days=1), periods=7)
    forecast_df = pd.DataFrame({'Close': forecasted_prices}, index=forecast_dates)

    return forecast_df

forecast_df = forecast_next_7_days(data)
```

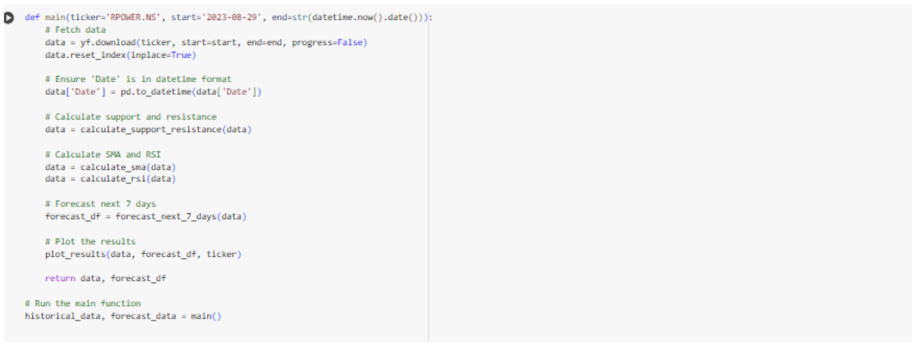


```python
def plot_results(data, forecast_df, ticker):
    plt.figure(figsize=(14, 7))
    plt.plot(data['Date'], data['Close'], label='Close Price')
    plt.plot(data['Date'], data['Support'], label='Support', linestyle='--', alpha=0.7)
    plt.plot(data['Date'], data['Resistance'], label='Resistance', linestyle='--', alpha=0.7)
    plt.plot(forecast_df.index, forecast_df['Close'], label='Forecasted Price', linestyle='--', color='orange')
    plt.title(f'{ticker} Price, Support, and Resistance Levels')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
    plt.show()

plot_results(data, forecast_df, ticker)
```

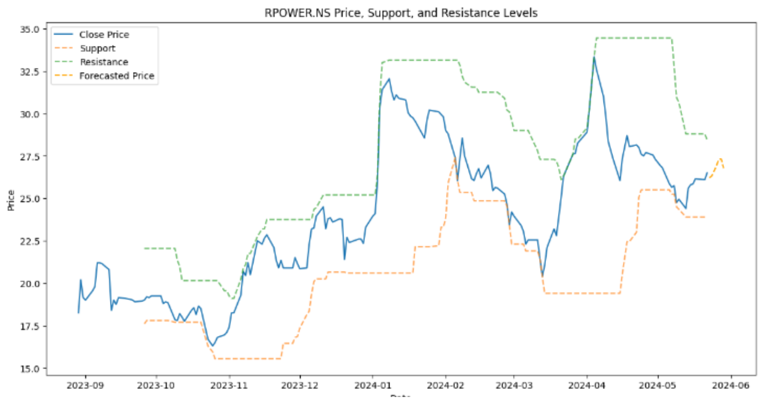


```python
def plot_results(data, forecast_df, ticker):
    plt.figure(figsize=(14, 7))
    plt.plot(data['Date'], data['Close'], label='Close Price')
    plt.plot(data['Date'], data['Support'], label='Support', linestyle='--', alpha=0.7)
    plt.plot(data['Date'], data['Resistance'], label='Resistance', linestyle='--', alpha=0.7)
    plt.plot(forecast_df.index, forecast_df['Close'], label='Forecasted Price', linestyle='--', color='orange')
    plt.title(f'{ticker} Price, Support, and Resistance Levels')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
    plt.show()

plot_results(data, forecast_df, ticker)
```





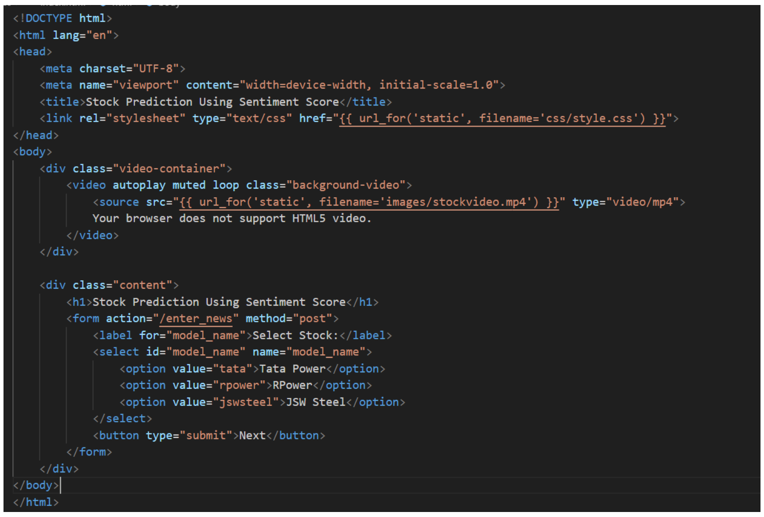
```python
def main(ticker='RPOWER.NS', start='2023-08-29', end=str(datetime.now().date())):
    # Fetch data
    data = yf.download(ticker, start=start, end=end, progress=False)
    data.reset_index(inplace=True)

    # Ensure 'Date' is in datetime format
    data['Date'] = pd.to_datetime(data['Date'])

    # Calculate support and resistance
    data = calculate_support_resistance(data)

    # Calculate SMA and RSI
    data = calculate_sma(data)
    data = calculate_rsi(data)

    # Forecast next 7 days
    forecast_df = forecast_next_7_days(data)

    # Plot the results
    plot_results(data, forecast_df, ticker)

    return data, forecast_df

# Run the main function
historical_data, forecast_data = main()
```



Chapter 6

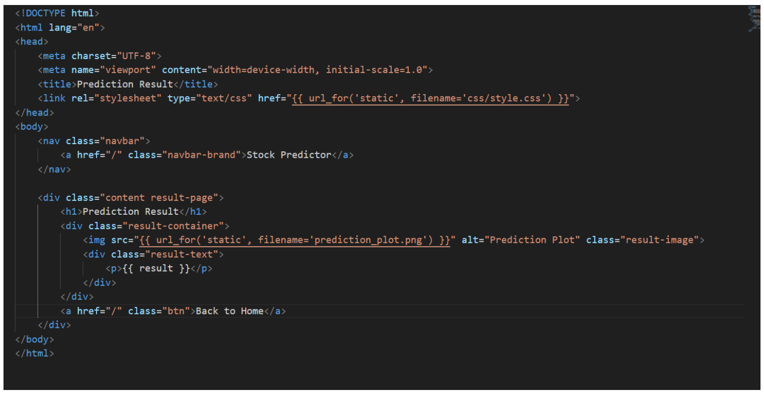
Project Deployment

6.1 Front-end



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stock Prediction Using Sentiment Score</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <div class="video-container">
        <video autoplay muted loop class="background-video">
            <source src="{{ url_for('static', filename='images/stockvideo.mp4') }}" type="video/mp4">
            Your browser does not support HTML5 video.
        </video>
    </div>

    <div class="content">
        <h1>Stock Prediction Using Sentiment Score</h1>
        <form action="/enter_news" method="post">
            <label for="model_name">Select Stock:</label>
            <select id="model_name" name="model_name">
                <option value="tata">Tata Power</option>
                <option value="rpower">RPower</option>
                <option value="jswsteel">JSW Steel</option>
            </select>
            <button type="submit">Next</button>
        </form>
    </div>
</body>
</html>
```

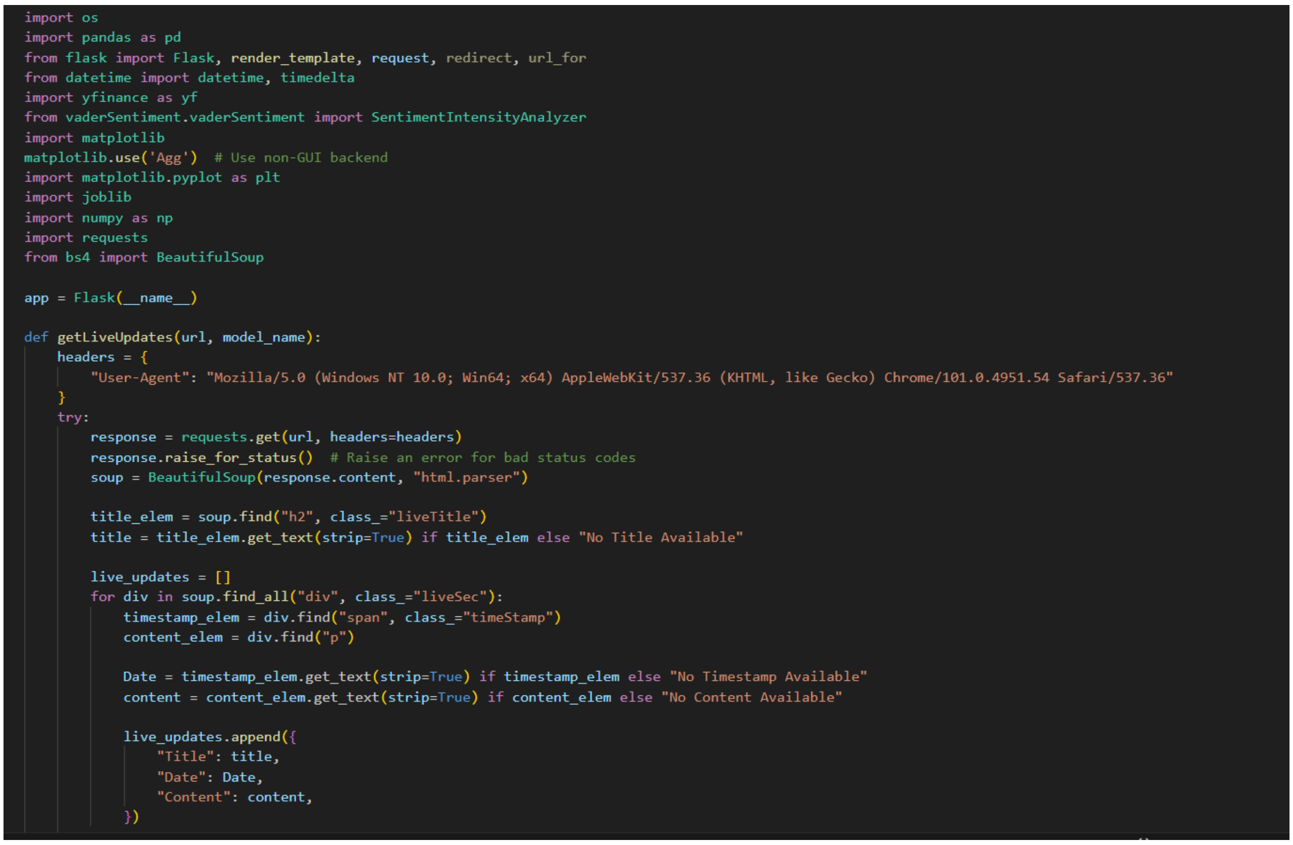
Figure 6.1: Index.html code screenshot



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Result</title>
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='css/style.css') }}">
</head>
<body>
    <nav class="navbar">
        <a href="/" class="navbar-brand">Stock Predictor</a>
    </nav>

    <div class="content result-page">
        <h1>Prediction Result</h1>
        <div class="result-container">
            <img src="{{ url_for('static', filename='prediction_plot.png') }}" alt="Prediction Plot" class="result-image">
            <div class="result-text">
                <p>{{ result }}</p>
            </div>
        </div>
        <a href="/" class="btn">Back to Home</a>
    </div>
</body>
</html>
```

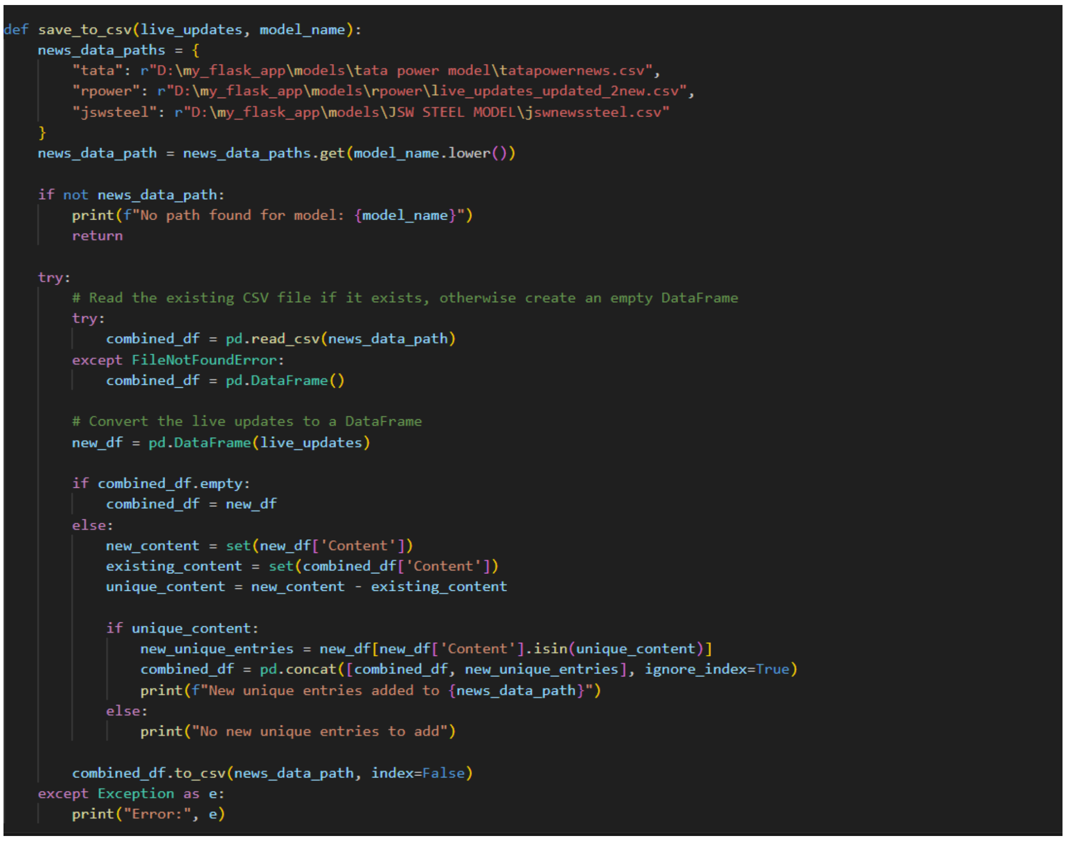
Figure 6.2: result.html code screenshot

6.2 Back-end



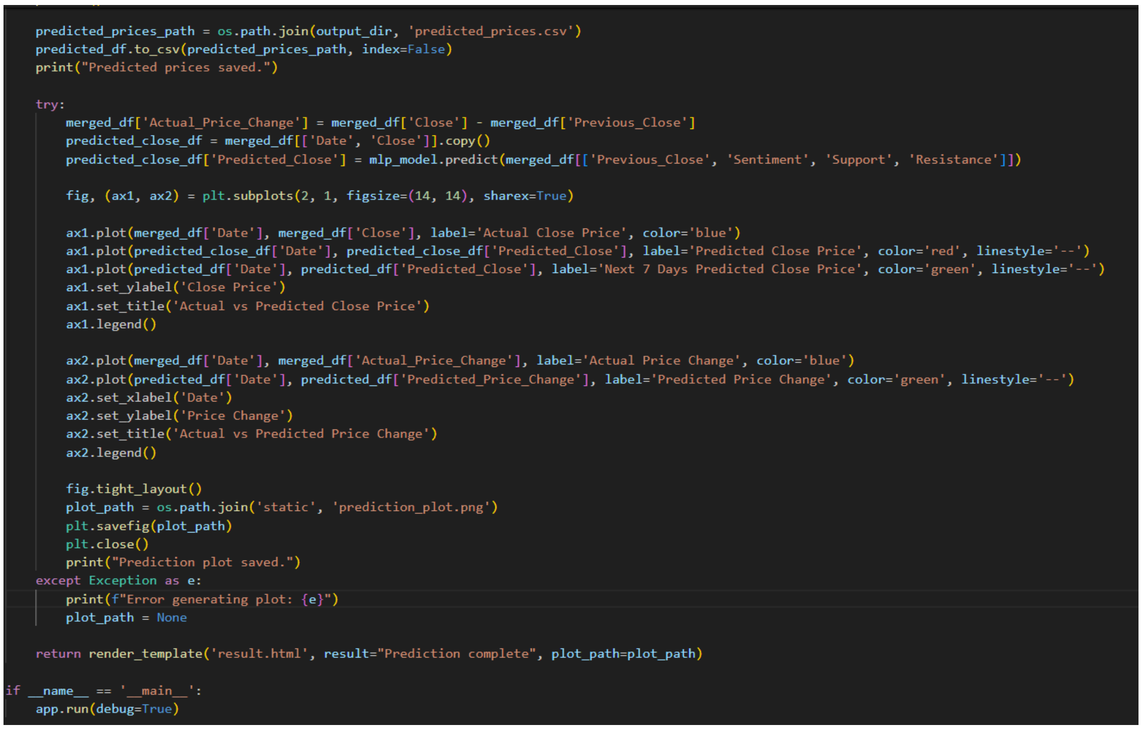
```python
import os
import pandas as pd
from flask import Flask, render_template, request, redirect, url_for
from datetime import datetime, timedelta
import yfinance as yf
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import matplotlib
matplotlib.use('Agg')  # Use non-GUI backend
import matplotlib.pyplot as plt
import joblib
import numpy as np
import requests
from bs4 import BeautifulSoup

app = Flask(__name__)

def getLiveUpdates(url, model_name):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36"
    }
    try:
        response = requests.get(url, headers=headers)
        response.raise_for_status()  # Raise an error for bad status codes
        soup = BeautifulSoup(response.content, "html.parser")

        title_elem = soup.find("h2", class_="liveTitle")
        title = title_elem.get_text(strip=True) if title_elem else "No Title Available"

        live_updates = []
        for div in soup.find_all("div", class_="liveSec"):
            timestamp_elem = div.find("span", class_="timeStamp")
            content_elem = div.find("p")

            Date = timestamp_elem.get_text(strip=True) if timestamp_elem else "No Timestamp Available"
            content = content_elem.get_text(strip=True) if content_elem else "No Content Available"

            live_updates.append({
                "Title": title,
                "Date": Date,
                "Content": content,
            })
```

Figure 6.3: App.py code



```python
def save_to_csv(live_updates, model_name):
    news_data_paths = {
        "tata": r"D:\my_flask_app\models\tata power model\tatapowernews.csv",
        "rpower": r"D:\my_flask_app\models\rpower\live_updates_updated_2new.csv",
        "jswsteel": r"D:\my_flask_app\models\JSW STEEL MODEL\jswnewssteel.csv"
    }
    news_data_path = news_data_paths.get(model_name.lower())

    if not news_data_path:
        print(f"No path found for model: {model_name}")
        return

    try:
        # Read the existing CSV file if it exists, otherwise create an empty DataFrame
        try:
            combined_df = pd.read_csv(news_data_path)
        except FileNotFoundError:
            combined_df = pd.DataFrame()

        # Convert the live updates to a DataFrame
        new_df = pd.DataFrame(live_updates)

        if combined_df.empty:
            combined_df = new_df
        else:
            new_content = set(new_df['Content'])
            existing_content = set(combined_df['Content'])
            unique_content = new_content - existing_content

            if unique_content:
                new_unique_entries = new_df[new_df['Content'].isin(unique_content)]
                combined_df = pd.concat([combined_df, new_unique_entries], ignore_index=True)
                print(f"New unique entries added to {news_data_path}")
            else:
                print("No new unique entries to add")

        combined_df.to_csv(news_data_path, index=False)
    except Exception as e:
        print("Error:", e)
```

Figure 6.4: App.py code



```python
    predicted_prices_path = os.path.join(output_dir, 'predicted_prices.csv')
    predicted_df.to_csv(predicted_prices_path, index=False)
    print("Predicted prices saved.")

    try:
        merged_df['Actual_Price_Change'] = merged_df['Close'] - merged_df['Previous_Close']
        predicted_close_df = merged_df[['Date', 'Close']].copy()
        predicted_close_df['Predicted_Close'] = mlp_model.predict(merged_df[['Previous_Close', 'Sentiment', 'Support', 'Resistance']])

        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 14), sharex=True)

        ax1.plot(merged_df['Date'], merged_df['Close'], label='Actual Close Price', color='blue')
        ax1.plot(predicted_close_df['Date'], predicted_close_df['Predicted_Close'], label='Predicted Close Price', color='red', linestyle='--')
        ax1.plot(predicted_df['Date'], predicted_df['Predicted_Close'], label='Next 7 Days Predicted Close Price', color='green', linestyle='--')
        ax1.set_ylabel('Close Price')
        ax1.set_title('Actual vs Predicted Close Price')
        ax1.legend()

        ax2.plot(merged_df['Date'], merged_df['Actual_Price_Change'], label='Actual Price Change', color='blue')
        ax2.plot(predicted_df['Date'], predicted_df['Predicted_Price_Change'], label='Predicted Price Change', color='green', linestyle='--')
        ax2.set_xlabel('Date')
        ax2.set_ylabel('Price Change')
        ax2.set_title('Actual vs Predicted Price Change')
        ax2.legend()

        fig.tight_layout()
        plot_path = os.path.join('static', 'prediction_plot.png')
        plt.savefig(plot_path)
        plt.close()
        print("Prediction plot saved.")
    except Exception as e:
        print(f"Error generating plot: {e}")
        plot_path = None

    return render_template('result.html', result="Prediction complete", plot_path=plot_path)

if __name__ == '__main__':
    app.run(debug=True)
```

Figure 6.5: App.py code

6.3 Website Screenshot

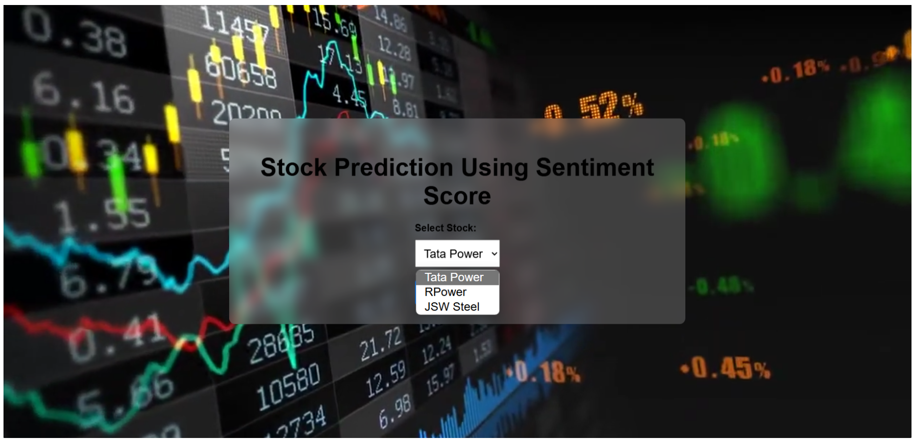


Figure 6.6: Selecting the Stock

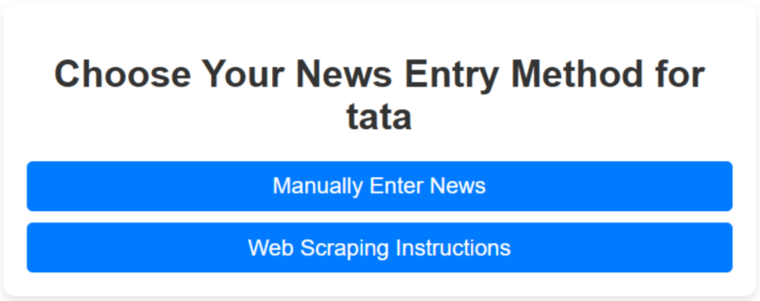
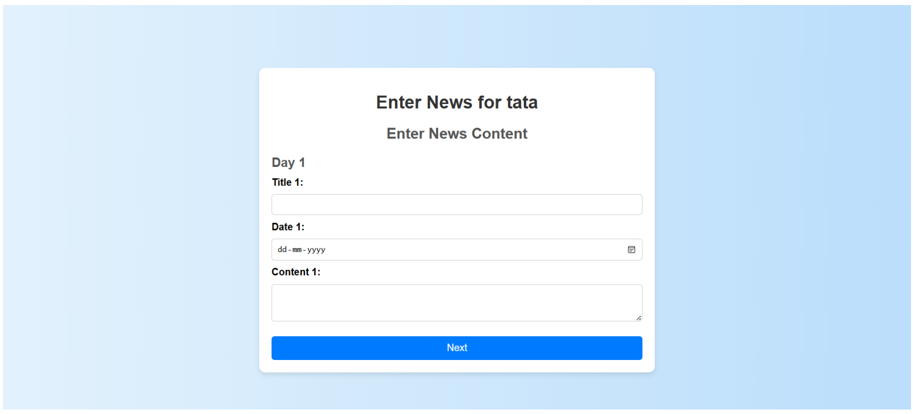

Choose Your News Entry Method for tata
Manually Enter News
Web Scraping Instructions

Figure 6.7: News Entry Method



Enter News for tata

Enter News Content

Day 1

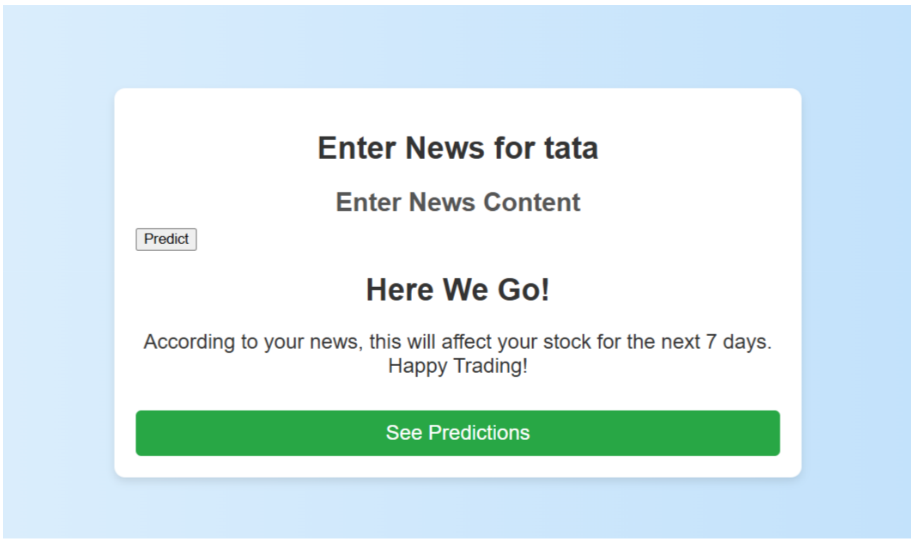
Title 1:

Date 1:

dd-mm-yyyy

Content 1:

Figure 6.8: News Entry



Enter News for tata

Enter News Content

Here We Go!

According to your news, this will affect your stock for the next 7 days. Happy Trading!

Figure 6.9: See prediction of selected stock

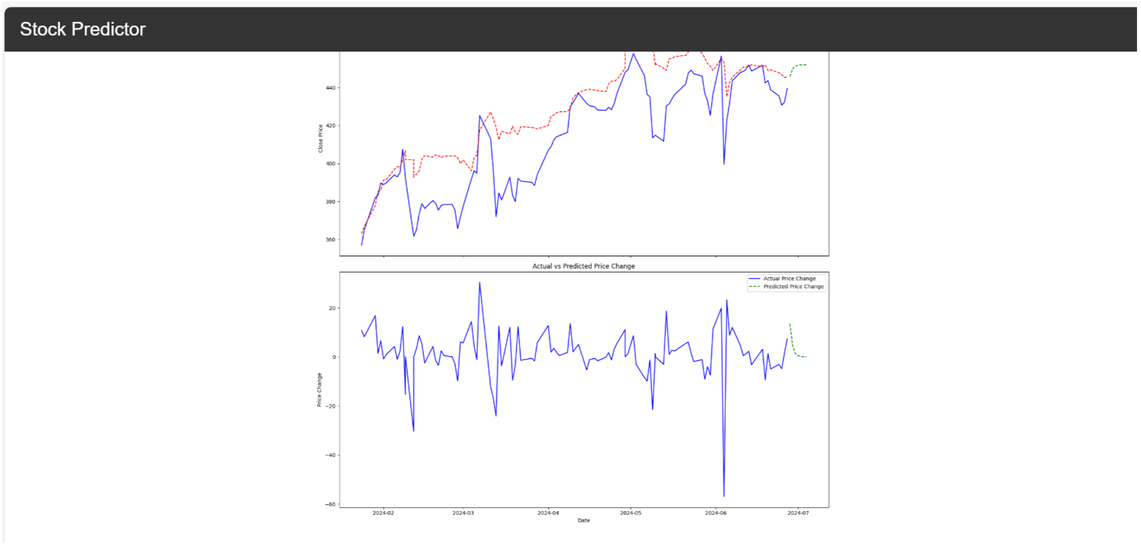


Figure 6.10: Predication in graph format

Chapter 7

Result

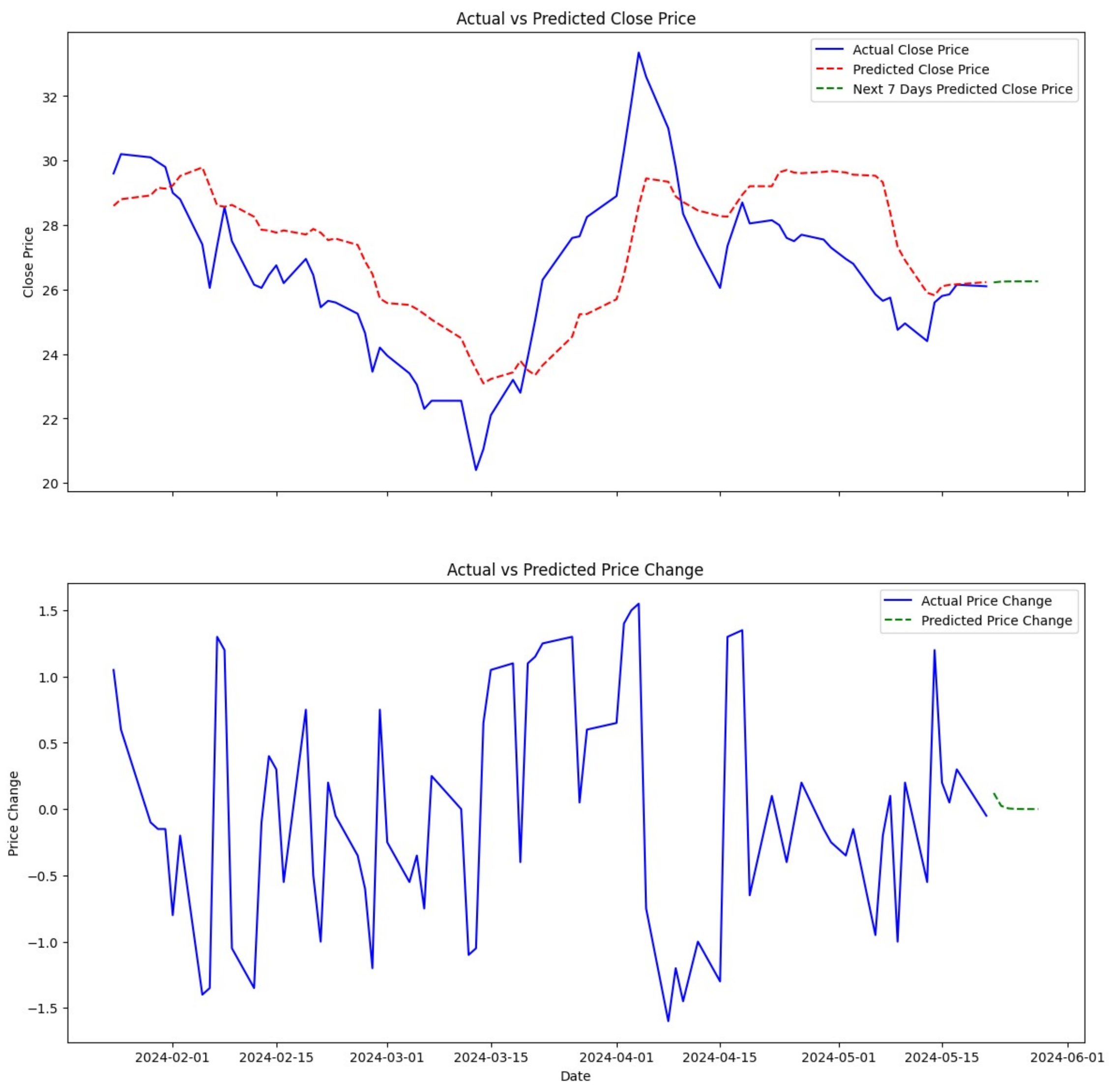


Figure 7.1: Predicted Output of R Power

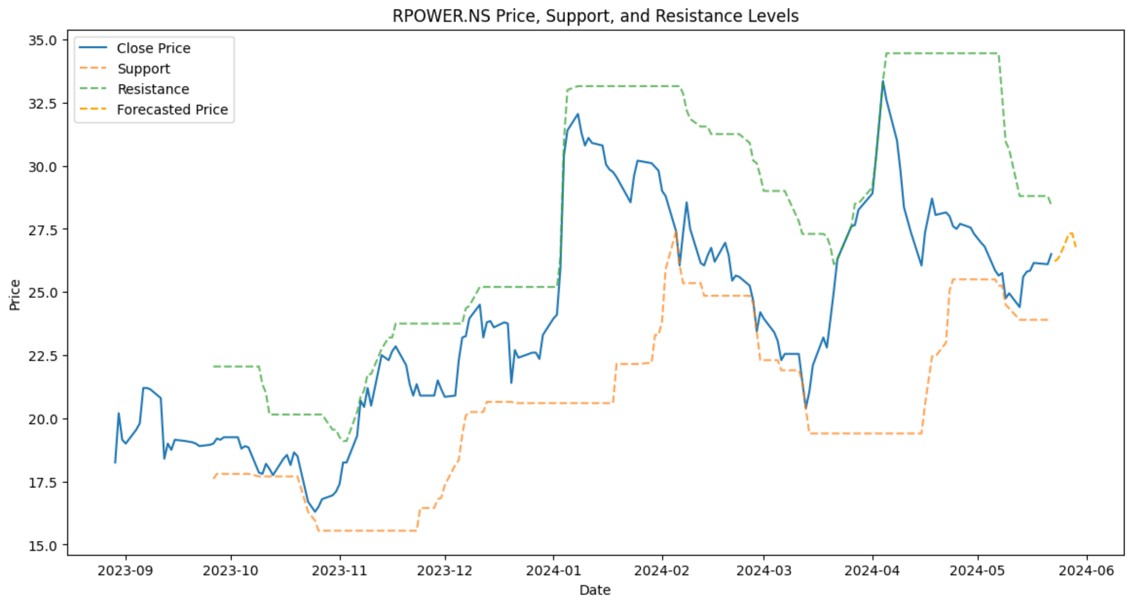


Figure 7.2: Predicted the Resistance and Support of tata Power

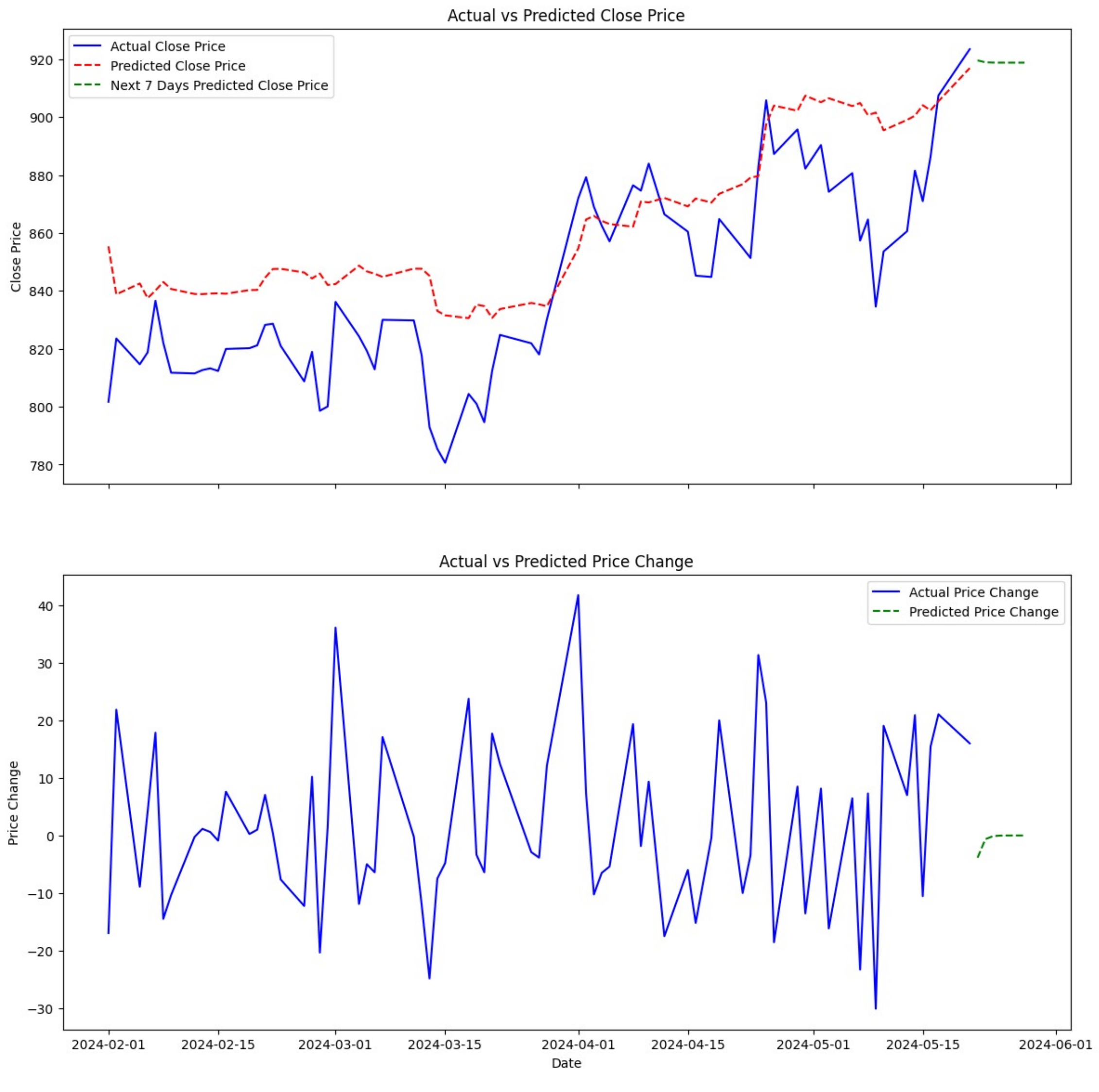


Figure 7.3: Predicted output of jsw steel

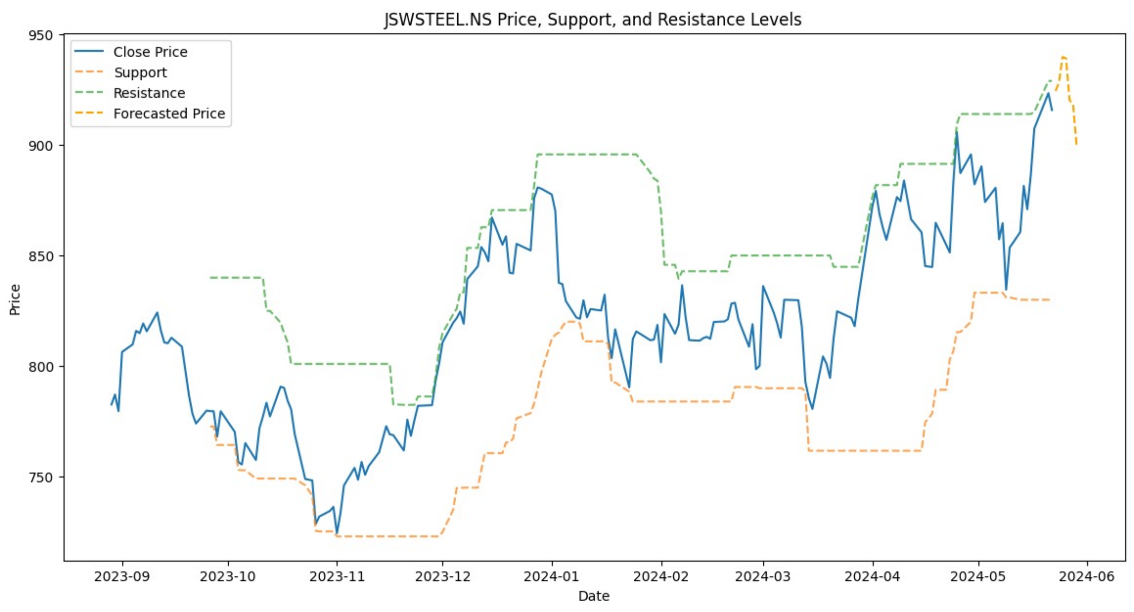


Figure 7.4: Predicted the Resistance and Support of jsw steel

Chapter 8

Conclusion

This project successfully integrated sentiment analysis of news articles with historical stock data to predict the closing prices of Reliance Power, Tata Power, and JSW Steel stock. We collected and processed live updates, performed sentiment analysis using the VADER tool, and merged the resulting sentiment scores with stock data from Yahoo Finance. Using these features, we trained an MLP Regressor model, which demonstrated the ability to predict stock prices with reasonable accuracy, as evidenced by the evaluation metrics:

Mean Squared Error (MSE): 308.7756825582436 Mean Absolute Error (MAE): 14.21167907415694 R-squared (R^2): 0.6301992836507546 The model's predictions for the next seven days were plotted alongside actual prices, showing its potential to follow stock price trends. The trained model was also saved for future use.

While the results are promising, there is significant potential for improvement. Future work could involve incorporating additional features, exploring advanced models, refining sentiment analysis, and performing hyperparameter tuning. This project highlights the valuable intersection of sentiment analysis and stock price prediction, setting a foundation for further development in this area.

By addressing these areas for improvement, the predictive accuracy and reliability of the model could be significantly enhanced, providing valuable insights for investors navigating the complexities of stock market dynamics.

Chapter 9

Future Scope

The future scope of this stock prediction web application involves several key areas for improvement and expansion:

- **Incorporation of Additional Data Sources:** Integrate additional data sources such as real-time financial news, social media sentiment, and economic indicators to enhance prediction accuracy and provide comprehensive insights.
- **Advanced Sentiment Analysis:** Utilize advanced NLP techniques to extract nuanced sentiment information from news articles, social media, and financial reports, potentially improving model accuracy.
- **User Personalization:** Implement personalized dashboards and prediction models tailored to individual user preferences, investment strategies, and risk profiles.
- **Enhanced Visualization:** Develop advanced visualization tools to display prediction results, trends, and historical data in an intuitive and interactive manner.
- **Mobile Application Development:** Extend the web application to mobile platforms (iOS and Android) to provide users with on-the-go access to stock predictions and insights.
- **Real-time Predictions:** Develop and deploy a system for real-time stock price predictions to provide up-to-date information for traders and investors.
- **Model Optimization:** Optimize hyperparameters and explore different network architectures, including deep learning models and regularization techniques, to enhance the model's performance.
- **Ensemble Learning:** Investigate ensemble methods such as random forests, gradient boosting, and voting classifiers to improve generalization and robustness.
- **Time Series Forecasting:** Deploy advanced time series forecasting models such as ARIMA, LSTM, and Prophet to better understand and predict stock price movements.
- **Model Interpretability:** Enhance interpretability and explainability using techniques such as SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations) to provide deeper insights into the factors influencing stock prices.

- **Automated Pipelines:** Implement automated data collection, preprocessing, model training, and monitoring pipelines to ensure the model remains up-to-date and reliable.
- **Scalability and Performance:** Improve the scalability and performance of the application to handle a larger number of users and higher data volumes efficiently.
- **Security Enhancements:** Strengthen security measures to protect user data and ensure the integrity and confidentiality of financial information.
- **Integration with Brokerage Services:** Enable integration with online brokerage services to allow users to execute trades directly from the application based on the predictions.
- **Feedback Loop:** Incorporate user feedback mechanisms to continuously improve the application based on user experiences and suggestions.
- **Compliance and Regulation:** Ensure the application complies with financial regulations and industry standards to provide a trustworthy and reliable service.

References

1. W. Khan, M. A. Ghazanfar, M. A. Azam, A. Karami, K. H. Alyoubi , and A. S. Alfakeeh , "Stock market prediction using machine learning classifiers and social media, news," J. Ambient Intell . Humaniz . Comput., no. 0123456789, 2020.

2. X. Pang, Y. Zhou, P. Wang, W. Lin, and V. Chang, "An innovative neural network approach for stock market prediction," J. Supercomput., vol. 76, no. 3, pp. 2098–2118, 2020.

3. I. K. Nti, A. Felix Adekoya, · Benjamin, and A. Weyori, "A systematic review of fundamental and technical analysis of stock market predictions," Artif. Intell. Rev., vol. 53, pp. 3007–3057, 123AD.

4. Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor ,Author - Junaid maqbool, Preeti Aggrawal,Ajay Mittal,2023

5. Short-term prediction for opening price of stock market based on self-adapting variant PSO-Elman neural network ,Author - Ze Zhang; Yongjun Shen; Guidong Zhang; Yongqiang Song; Yan Zhu,2017

6. Stock Price Trend Prediction Model Based on Deep Residual Network and Stock Price Graph,Author -Heng Liu; Bowen Song,2018

7. Prediction of Stock Price Based on LSTM Neural Network,Author - Dou Wei,2019