Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance

A Project Report
Submitted in partial fulfilment of the requirements for the
Award of the degree of

Master of Computer Application

By
Md. Tarique Anwer
(322201297)

Centre for Distance and Online Education
LOVELY PROFESSIONAL UNIVERSITY
PHAGWARA, PUNJAB
2024

Declaration by the Student

To whom-so-ever it may concern

I, **MD TARIQUE ANWER**, **322201297**, hereby declare that the work done by me on "**Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**", is a record of original work for the partial fulfilment of the requirements for the award of the degree, **Master of Computer Application.**

MD TARIQUE ANWER (322201297)

Signature of the student

Dated:

Acknowledgement

I would like to express my sincere gratitude to everyone who contributed to the successful completion of my project, "**Real-Time Stock Analysis and Prediction using Python and Yahoo Finance**".

First and foremost, I extend my deep appreciation to the faculty members of the **Online Master of Computer Application program at Lovely Professional University**. Their commitment to excellence in teaching and the comprehensive curriculum provided me with the essential knowledge, guidance, and resources required to complete this project. I am especially grateful to my instructors and mentors whose insightful feedback, expert advice, and constructive criticism were invaluable in refining my approach and methodology.

A special note of thanks goes to my family and friends for their unwavering support, understanding, and patience throughout this endeavour. Their belief in my abilities and their constant motivation were sources of inspiration that fuelled my determination to push through difficult moments and stay committed to my goals.

Completing this project has been a deeply enriching and educational journey. It has allowed me to apply theoretical concepts in a practical setting, sharpen my analytical and technical skills, and gain invaluable insights into real-time financial data analysis. I am truly grateful to everyone who contributed in any capacity to making this project a reality.

Thank you all for your support and encouragement.

MD TARIQUE ANWER (322201297)

Lovely Professional University,
Phagwara, Punjab

Tarique Anwer

Abstract

This project, "**Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**", aims to assist investors in making informed decisions by providing timely insights into stock market trends. In today's fast-paced financial environment, analysing real-time stock data is crucial for individuals and organizations to stay ahead in investments. This project leverages Python for data extraction, processing, and visualization, using Yahoo Finance as the primary data source for live market data.

Our approach integrates data analysis and machine learning to predict short-term price movements and identify potential investment opportunities. Key tools and libraries used include pandas for data manipulation, Plotly for visualization, Flask for backend of web application and HTML, CSS and JavaScript as frontend technologies and various predictive models for forecasting stock prices. By analysing historical stock data and real-time trends, we generate insights on stock performance that can be used to predict future price movements.

Through this project, we demonstrate a practical application of machine learning in finance, highlighting the use of web scraping, data analysis, and predictive modeling to support investment decisions. This report details the methodologies, tools, and outcomes, providing a comprehensive view of how Python can be used to deliver robust stock analysis and predictive insights for real-time decision-making in the stock market.

Table of Contents

List of Tables

List of Figures/Charts

List of Symbols

Symbol	Description	Unit
₹	Indian Rupees	INR
$	US Dollar	USD

List of Abbreviations

Abbreviation	Full Form
ML	Machine Learning
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
NLP	Natural Language Processing
API	Application Programming Interface
YFinance	Yahoo Finance
CSV	Comma Separated Values
OS	Operating System
RAM	Random Access Memory
IDE	Integrated Development Environment
UI	User Interface
UX	User Experience
IO	Input Output
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheet
JS	JavaScript
AJAX	Asynchronous JavaScript and XML
DOM	Document Object Model
DFD	Data Flow Diagram
MB	Mega Byte
GB	Giga Byte
MA	Moving Averages
MAPE	Mean Absolute Percentage Error
NS	National Stock

Chapter-1 Introduction

The project, **"Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance",** is designed to provide investors with accurate, data-driven insights into stock market performance. This project retrieves both historical and real-time stock data from Yahoo Finance using web scraping techniques, with an alternative option to use the YFinance API if web scraping becomes unavailable.

By integrating machine learning techniques and leveraging key Python libraries—such as **Pandas** for efficient data manipulation, **NumPy** for complex numerical computations, and **Plotly** for data visualization—this project transforms raw market data into actionable insights. The primary objective is to predict future stock trends, empowering investors to make informed decisions about which stocks to buy. Through real-time analysis and forecasting, this project demonstrates the potential of Python in financial analytics, offering a robust tool for stock prediction and decision-making.

1.1 Data Retrieval and Processing

Data is sourced from Yahoo Finance through web scraping techniques, or alternatively through the YFinance API if web scraping becomes inaccessible. This dual approach ensures a reliable and continuous flow of market data for analysis. Key libraries such as Pandas and NumPy play a central role in structuring, cleaning, and manipulating the data, transforming raw inputs into meaningful insights.

1.2 Analytical Approach and Machine Learning

Using Python's extensive data science ecosystem, the project explores historical stock prices to identify patterns and predict future stock performance. Various machine learning models are applied to this dataset, aiming to predict short-term stock price movements based on historical trends and market behavior. This predictive modeling enables a comparative analysis of stocks, identifying those with promising trends for potential investments.

1.3 Visualization and User Insights

Data visualization is achieved through the Matplotlib library, providing graphical insights that help investors quickly understand trends and patterns. By combining real-time data analysis with predictive analytics, the project delivers a user-friendly platform where users can visualize potential stock movements and make informed investment decisions.

1.3 Goal and Significance

The primary objective of this project is to enable informed, data-driven decision-making for investors looking to maximize returns. By providing an accessible, Python-based solution for stock analysis and prediction, the project serves as a practical tool for anyone interested in leveraging real-time financial data to anticipate market movements.

Chapter-2 Review of Literature

This chapter reviews existing studies and technologies relevant to the development of the project **Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**. The literature review focuses on key areas including the use of Python for financial analytics, predictive modeling techniques, visualization tools, and real-time data integration, providing a foundation for understanding the methodologies applied in the project.

2.1 Stock Market Analysis and Prediction

Stock market prediction has long been a subject of interest due to the complexities of financial markets and the potential for substantial gains. Traditional methods relied heavily on technical analysis using historical price trends and indicators such as moving averages and candlestick charts. Recent advancements in computational methods have integrated machine learning models to enhance predictive capabilities.

Linear regression, a fundamental statistical technique, has been widely used to analyze the relationship between stock price movements and various predictors. However, its performance may be limited by the complexity of financial data. To address this, more sophisticated models such as decision trees, support vector machines, and deep learning frameworks have been explored in recent research.

2.2 Python as a Tool for Financial Analytics

Python is a widely used programming language for financial analytics due to its flexibility and extensive library ecosystem. Relevant libraries include:

- **yfinance**: Simplifies the process of fetching historical and real-time stock market data.
- **pandas**: Facilitates data cleaning, transformation, and analysis, making it integral for managing stock price datasets.
- **plotly**: Enables interactive visualizations such as candlestick and line charts, aiding in comprehensible representation of stock trends.

- **scikit-learn**: Provides robust tools for predictive modeling and evaluation, such as the Linear Regression model applied in this project.

Python's integration of data analysis, visualization, and machine learning functionalities makes it an ideal choice for building comprehensive stock analysis applications.

2.3 Real-Time Stock Data Integration

Real-time stock analysis requires dynamic data acquisition and processing. Platforms like Yahoo Finance provide APIs for accessing live and historical data. Studies have highlighted the effectiveness of using Yahoo Finance data for financial forecasting due to its reliability and wide coverage of global stock markets.

The yfinance library in Python offers a convenient interface for fetching data, including stock prices, trading volumes, and financial indicators. This simplifies the workflow, allowing developers to focus on data modeling and application development.

2.4 Visualization Techniques in Stock Analysis

Visualization is critical for interpreting stock data and trends. Candlestick charts are a staple in financial analysis, providing insights into daily price movements, including open, high, low, and close prices. Moving averages (e.g., 50-day and 200-day averages) are commonly used to identify trends and potential reversals.

Interactive visualization tools like Plotly enhance user engagement by enabling zooming, panning, and tooltips for data points, making them suitable for web-based applications.

2.5 Predictive Modeling Techniques

Predictive modeling in stock analysis focuses on forecasting future price movements based on historical trends. This project employs linear regression to predict percentage changes in stock prices. Linear regression has been extensively studied for its simplicity and interpretability in financial contexts, although it may be less effective for capturing complex market dynamics compared to advanced models like neural networks.

The use of train, test, split from scikit-learn ensures robust model evaluation by dividing the dataset into training and testing subsets. The results are used to recommend investment actions such as "BUY" or "HOLD" based on predicted price movements.

2.6 Challenges in Real-Time Stock Prediction

The project addresses several challenges commonly encountered in real-time stock analysis:

- **Data Volatility**: Stock markets are highly sensitive to news and events, requiring models to adapt quickly.
- **Data Quality**: Historical stock data may contain missing or erroneous values that need to be handled before analysis.
- **Real-Time Processing**: Ensuring seamless integration of real-time data while maintaining application performance.

These challenges are mitigated through effective use of Python libraries for data preparation, robust model training, and efficient visualization techniques.

2.7 Literature Gap

While existing studies demonstrate the utility of Python and machine learning in financial analytics, certain gaps persist:

- Limited focus on integrating real-time data streams with predictive modeling.
- Lack of user-centric visualization and decision-making tools in many academic

implementations.

- Minimal exploration of simple regression models in combination with interactive web-based applications.

This project aims to bridge these gaps by building a real-time stock analysis and prediction system that leverages Python, Yahoo Finance, and interactive visualizations to provide actionable insights.

2.8 Conclusion

The review highlights the evolution of stock analysis methodologies and the pivotal role of Python in advancing real-time financial analytics. By leveraging reliable data sources like Yahoo Finance, interactive visualization tools, and robust modeling techniques, this project seeks to deliver a practical application for real-time stock analysis and investment decision-making.

Chapter-3 Implementation of Project

This chapter explains the process of building the project **Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**. It covers the key steps, including defining requirements through the Software Requirements Specification (SRS), Tools and technologies, designing the system architecture, and implementing the solution with code.

Each section provides a clear view of how the project was developed, from system design to testing and deployment. The aim is to show how the project transitioned from an idea into a working application, meeting its goals of analyzing and predicting stock trends in real-time.

3.1 System Requirement Specification (SRS)

The requirements for the **Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance** project are analyzed in terms of system, software, and data requirements. A thorough understanding of these requirements ensures that the project is developed within the scope and performs as expected, providing accurate predictions and insights for investors.

The system requirements for the project focus on ensuring that the hardware and software infrastructure is capable of handling real-time stock data retrieval, data processing, and machine learning model training.

3.1.1 Hardware Requirement

The hardware requirements are moderate, as the project involves data analysis, web scraping, and machine learning model prediction, which are computationally intensive but not resource-heavy for small to medium-scale data.

- **Processor:** A minimum of Intel Core i3 or equivalent (Quad-core preferred for better performance during data processing and model training).
- **RAM:** 8 GB or higher to handle data manipulation and machine learning tasks.
- **Storage:** At least 50 GB of free disk space for storing datasets, CSV files, and model output.
- **Internet Connection:** A stable connection for web scraping or API requests to Yahoo Finance.

3.1.2 Software Requirement

The software environment is crucial for running the project smoothly, especially for Python-based development. Several software tools and platforms will be required for development, testing, and deployment.

- **Operating System:** Windows 10 or higher, macOS, or any Linux-based OS.
- **IDE (Integrated Development Environment):** Recommended IDEs include Visual Studio Code, Google Colab, or Jupyter Notebook for interactive coding and analysis.
- **Version Control System:** Git for managing project code versions, collaborating with teams, and maintaining code history.
- **Internet Browser:** Google Chrome, Firefox, Safari, Edge, Opera or any other latest version of modern internet browsers.
- **Python:** Version 3.11 as it is compatible with the necessary libraries and frameworks.
- **Web Framework:** Flask for backend development.
- **Python Libraries:**
    - **Pandas:** For data manipulation and analysis.
    - **NumPy:** For numerical computations.
    - **Plotly:** For data visualization using graph
    - **Flask:** For web application and API's
    - **YFinance:** For accessing Yahoo Finance data through API calls.
    - **Scikit-learn:** For implementing machine learning algorithms.
    - **Beautiful Soup:** For web scraping (if needed).
- **JavaScript Libraries and Frameworks:**
    - jQuery 3.6:
    - Bootstrap 5: For UI
    - jQuery DataTable: For DataTable
    - Moment.js: For Date and time formatting
- **Microsoft Excel:** For reading csv (comma separated value) files.

3.1.3 Data Requirements

Data plays a central role in this project, and it is vital that the data is reliable, accurate, and up-to-date. The following data requirements have been identified for effective stock prediction:

Stock Data: The project will rely on historical and real-time stock data from Yahoo Finance, including stock prices, market trends, trading volume, and other relevant metrics.

Data Storage: CSV files will be used to store historical stock data retrieved from Yahoo Finance. These files will include columns such as stock symbols, date, opening price, closing price, volume, and other relevant indicators.

Data Accuracy and Completeness: The stock data needs to be accurate and free from missing values to ensure the quality of analysis. Incomplete or erroneous data will affect prediction accuracy.

Real-Time Data: The project will need to pull real-time stock data for ongoing analysis. This will be achieved through web scraping or the YFinance API, which requires a continuous internet connection for fetching the latest market data.

3.2 Tools & Technologies

In developing the "**Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**" project, a variety of technologies were carefully selected to ensure robust data handling, efficient processing, and effective visualization. Each chosen technology contributes uniquely to the project's requirements, facilitating a streamlined approach to analyzing and predicting stock trends.

3.2.1 Python

Python was chosen as the primary programming language due to its extensive ecosystem of libraries and tools designed for data science, machine learning, and financial analysis. Python's syntax is simple and readable, making it highly accessible and suitable for rapid development. Additionally, Python offers a range of libraries tailored for data manipulation, visualization, and machine learning, making it an ideal choice for building an end-to-end stock prediction model.

3.2.2 Pandas

The Pandas library is pivotal for data manipulation in this project. It provides robust data structures, like Data Frames, which allow for easy manipulation, cleaning, and transformation of large datasets. Stock data analysis often requires processing extensive historical datasets, and Pandas simplifies the process of importing, organizing, and preparing this data for analysis. Its integration with other Python libraries makes it highly versatile for building comprehensive data workflows.

3.2.3 NumPy

NumPy is essential for performing numerical computations efficiently. Since stock prediction models rely on complex mathematical operations and calculations, NumPy's fast array operations and mathematical functions allow for efficient handling of large datasets. This library complements Pandas, enabling streamlined computations necessary for data preparation and modeling in real-time and historical data analysis.

3.2.4 Plotly

Plotly was selected for its advanced data visualization capabilities that extend beyond the basic plotting functionalities offered by libraries such as Matplotlib. Plotly's interactive charts provide users with a more engaging experience, allowing them to hover over data points for detailed information and manipulate the visualization for deeper insights. In the context of stock analysis and prediction, Plotly's interactivity enhances the user's ability to explore and interpret data trends, aiding in a comprehensive understanding of market patterns. This interactive approach makes it easier for users to make well-informed decisions by observing trends and predictions in a dynamic format.

3.2.5 YFinance API

Yahoo Finance is a reliable source of both real-time and historical stock data, making it a valuable resource for this project. Web scraping is used to retrieve data directly from Yahoo Finance, providing flexibility in obtaining various types of financial information. However, to ensure data access continuity, the YFinance API serves as a backup in cases where web scraping may be limited or unavailable. This dual approach ensures consistent data flow, allowing the project to function effectively without disruptions.

3.2.6 CSV (Comma-Separated Values)

The CSV (Comma-Separated Values) format is commonly used for storing and exchanging data. In this project, CSV files are used to store historical stock data retrieved from Yahoo Finance. The CSV format provides a simple and widely-supported method of saving data, making it easy to import, export, and work with data in Python using Pandas. By leveraging CSV files, the project ensures that data can be easily loaded, saved, and shared across different components and systems, maintaining flexibility in data processing.

3.2.7 Machine Learning Algorithms

Integrating machine learning algorithms enables this project to move beyond mere data analysis and into prediction. Machine learning algorithms can analyze patterns within historical stock data and generate forecasts on future stock prices. By using Python's machine learning ecosystem, which includes libraries like scikit-learn, the project can implement models that provide actionable insights for investors. This predictive aspect is the cornerstone of the project, offering users a data-driven basis for their investment decisions.

3.2.8 Flask

Flask was chosen as the web framework for this project due to its lightweight and flexible nature. Flask's simplicity allows for rapid development and integration with various Python libraries, making it ideal for building web-based applications that require custom functionality. For this project, Flask acts as the backbone of the web interface, facilitating seamless communication between the frontend and backend, and ensuring a smooth user experience. Flask's modular structure supports scalability, which is advantageous for projects that may require expansion in the future. Additionally, its integration with Jinja2 templating engine aids in rendering dynamic HTML content, enabling interactive and responsive web pages.

3.2.9 HTML

HTML (Hypertext Markup Language) was chosen as the structural foundation for the web application. It is essential for defining the content and layout of the web pages, ensuring that information is organized in a meaningful way. HTML provides the basic elements that structure the application, allowing for the seamless integration of data visualization tools, interactive charts, and user input forms. Its compatibility with various web technologies makes it a versatile choice for building the framework of the project's user interface.

3.2.10 CSS

CSS (Cascading Style Sheets) was used to style the web pages and enhance the visual appeal of the application. With CSS, the project achieves a modern and user-friendly design that improves the overall user experience. CSS allows for customization of elements such as fonts, colors, and layout, ensuring that the interface is both attractive and responsive. By leveraging CSS, the project maintains consistency in design across different devices, enhancing accessibility and usability.

3.2.11 JavaScript

JavaScript was employed to add interactivity and dynamic behavior to the web application. This scripting language enables real-time updates, client-side data validation, and interactive features that improve user engagement. JavaScript's ability to manipulate the Document Object Model (DOM) allows the project to respond to user actions effectively, creating a more seamless and responsive user experience. Integrating JavaScript ensures that users can interact with the stock analysis and prediction platform in real-time, enhancing the practicality and functionality of the application.

3.2.12 jQuery 3.6

A lightweight, fast, and feature-rich JavaScript library used for simplifying HTML document traversal, event handling, animation, and AJAX interactions. Version 3.6 ensures compatibility and modern web practices.

3.2.13 Bootstrap 5

A popular CSS framework that provides pre-designed components and responsive design utilities to create modern, mobile-first, and aesthetically pleasing web applications without extensive custom CSS.

3.2.14 jQuery DataTable

A plugin for jQuery that adds advanced interaction controls to HTML tables, including search, sorting, pagination, and data export features, enhancing the usability of data displays in web applications.

3.2.15 Moment.js

A JavaScript library designed for parsing, validating, manipulating, and formatting dates and times. It's particularly useful for handling complex date operations in client-side scripting.

3.3 System Design

The system design phase is critical to ensuring that the "**Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**" project operates efficiently and meets its objectives. This chapter outlines the architecture, components, and workflow of the system, detailing how each element interacts to create a cohesive stock analysis and prediction tool.

3.3.1 Architecture Overview

The system architecture is designed to handle data retrieval, processing, analysis, and prediction efficiently. The project follows a modular structure, where each module has a distinct function that contributes to the overall workflow. The architecture can be broken down into the following main components:

- **Data Collection Module:** Responsible for retrieving real-time and historical stock data from Yahoo Finance using web scraping and the YFinance API.
- **Data Preprocessing Module:** Cleans and formats the retrieved data for analysis.
- Analysis and Prediction Module: Uses machine learning models to analyze historical data and predict future stock trends.
- **Visualization Module:** Presents data and prediction results using charts and graphs created with Matplotlib.
- **User Interface Module:** Provides a user-friendly interface for users to input stock symbols and view analysis and predictions.

3.3.2 Data Flow Diagram (DFD)

The Data Flow Diagram illustrates the flow of information through the system, providing a clear overview of how data is processed from input to output.

Level 0:

- **User Input:** Users provide the stock symbol they are interested in analyzing and a start and end date to analyze from historical data.
- **System Processing:** Process the data
- **System Output:** The system displays analysis and prediction results.

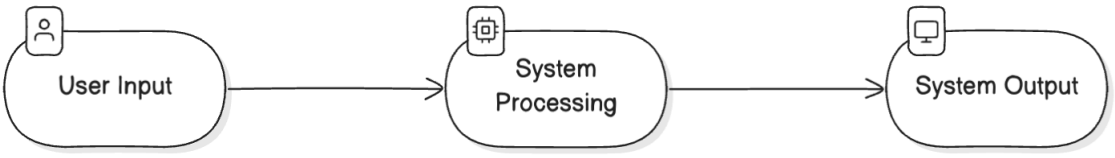


Figure 3.1: Data Flow Diagram (DFD) Level 0

Level 1:

- **Data Input:** Stock symbol and date range are input by the user.
- **Data Collection Module:** Retrieves data from Yahoo Finance via web scraping or the YFinance API.
- **Data Preprocessing Module:** Formats and cleans the data for analysis.
- **Analysis and Prediction Module:** Applies machine learning models to generate predictions.
- **Visualization Module:** Creates visual representations of stock trends and predictions.
- **User Output:** Final analysis and prediction results are displayed.

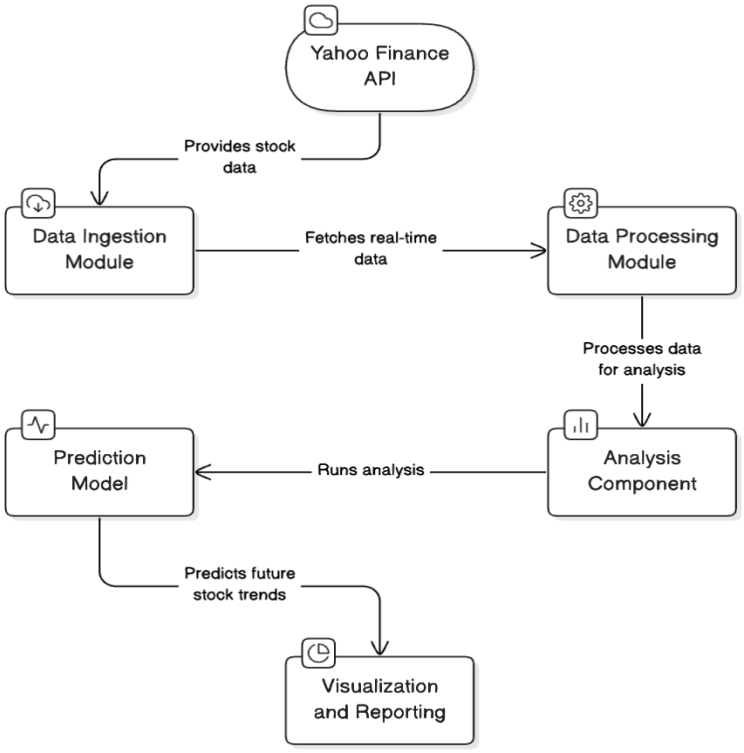


Figure 3.2: Data Flow Diagram (DFD) Level 1

3.3.3 System Components

Each system component plays a specific role in the workflow. The following sections describe these components in detail:

Data Collection Module

This module is designed to interact with Yahoo Finance to fetch both historical and real-time stock data. The data is retrieved using: YFinance API: Provides a straightforward method for accessing structured stock data.

Data Preprocessing Module

The preprocessing module ensures the data is cleaned and ready for analysis. This involves:

- Removing null or missing values.
- Formatting data into a structured CSV or Data Frame.
- Calculating additional features if necessary (e.g., moving averages, volatility).

Analysis and Prediction Module

This module is the core of the system where machine learning algorithms are applied. The steps include:

- **Feature Selection:** Choosing relevant features from the dataset to improve prediction accuracy.
- **Model Training:** Using algorithms such as linear regression or decision trees to train the model.
- **Prediction:** Running the trained model on the prepared dataset to forecast future stock prices.

Visualization Module

The visualization module employs Matplotlib to create graphs that help in understanding stock trends and predictions. Common visual outputs include:

- Time series plots for stock prices.
- Prediction overlay charts showing actual vs. predicted data.

3.3.4 Workflow Diagram

A workflow diagram shows the step-by-step process that data goes through in the system:

- **User Input:** Stock symbol and date range.
- **Data Collection:** Data is retrieved from Yahoo Finance.
- **Data Preprocessing:** Data is cleaned and formatted.
- **Analysis and Prediction:** Machine learning models process the data.
- **Visualization:** Results are plotted.
- **User Output:** Predictions are displayed to the user.

3.3.5 System Design Considerations

- **Modularity:** Each component is independent, making the system easy to modify and update.
- **Scalability:** The design supports scaling up to include more data sources or more advanced machine learning models.
- **Error Handling:** The system includes mechanisms to handle errors in data retrieval or inconsistencies.

3.3.6 User Interface Design

A user-friendly interface is essential for an effective project. The design prioritizes simplicity and clarity, ensuring that even non-technical users can navigate and use the tool efficiently. The main features of the interface include:

- **Search Box:** For users to enter the stock symbol and select the date range.
- **Search Button:** To initiate data retrieval and prediction.
- **Overview Section:** To view stock current closing price, change and change percentage and prediction recommendation.
- **Graph Display Tab:** To present the analysis and prediction results visually.
- **Historical Data Tab:** To present the analysis and prediction results visually.
- **Finance News Tab:** To present the analysis and prediction results visually.

Figure 3.3: Main page

3.4 Coding

The project is implemented using Python and Flask as the backend framework and HTML, CSS, JavaScript, Bootstrap 5 as frontend. The code is modular and consists of functions to fetch stock data, analyze trends, predict performance, and provide recommendations. This section explains the key components and functionalities of the code with snippets and descriptions.

Project file and folder structure

The provided file structure represents a Flask-based web application for stock analysis. Here's an explanation of each part:

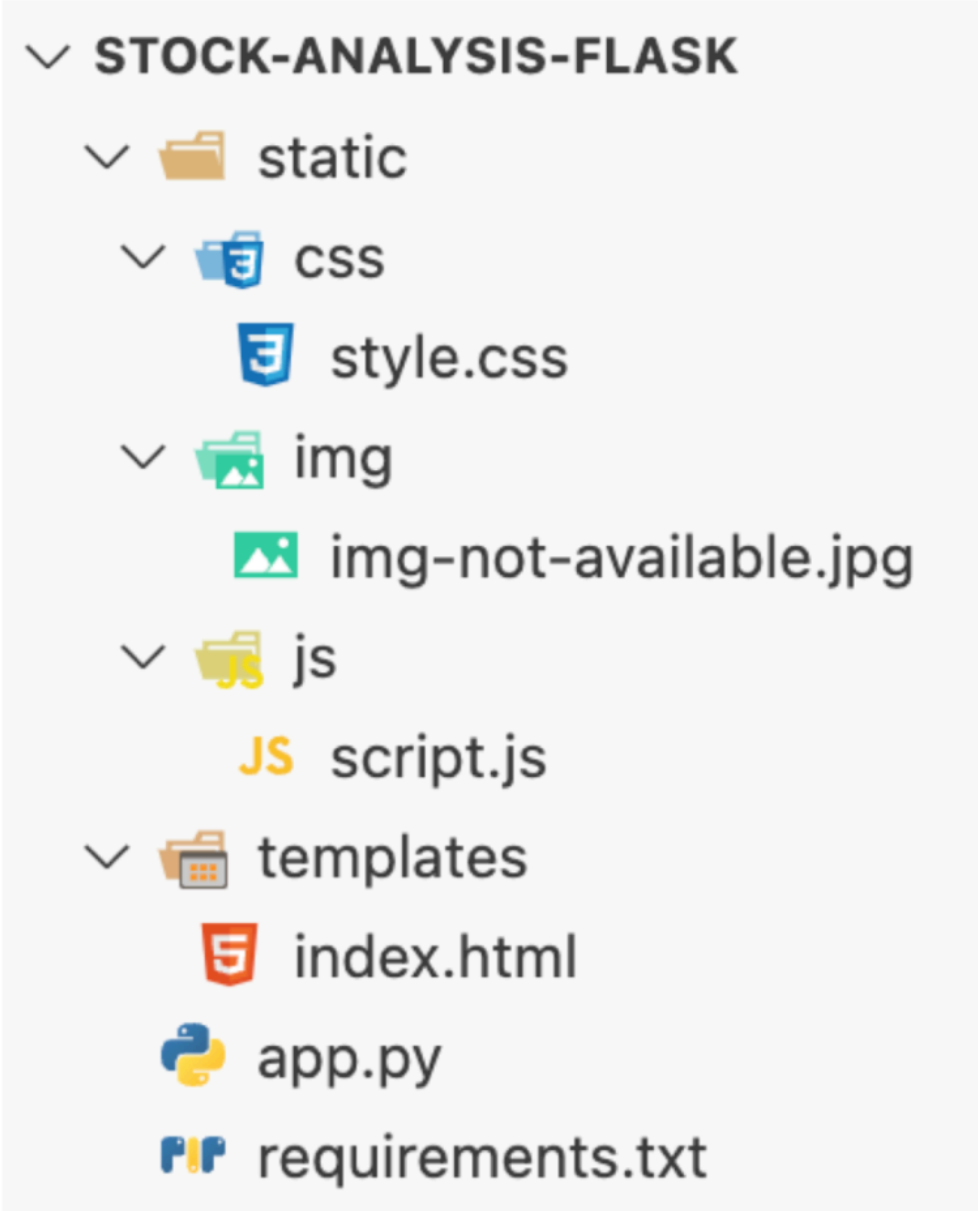


Figure 3.4: Files and folder structure

📂 Root Directory: stock-analysis-flask

This is the main project folder containing all the essential components of the Flask application.

📂static/

The static folder contains all the static files for the application, including stylesheets, JavaScript files, and images.

- 📂**css/**
  - **style.css**: This file contains custom styles for the application's frontend, such as layouts, colors, fonts, and other design elements.

```
.input-group .form-control:focus,
.input-group .form-select:focus {
    box-shadow: none !important;
}

/* Spinner container with background overlay */
.spinner-container {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.1); /* Semi-transparent black background */
    justify-content: center;
    align-items: center;
    z-index: 9999;
    display: flex;
}

/* Spinner styling */
.spinner-border {
    width: 3rem;
    height: 3rem;
}

table.dataTable {
    border-collapse: collapse;
    width: 100%;
}

/* Ensure the cards fill the height equally */
.news-card {
    display: flex;
    flex-direction: column;
    height: 100%;
```

```
}
.news-card-body {
    flex-grow: 1; /* Makes sure the body expands to take the available space */
}

/* Fix image height and ensure it covers the space */
.news-card-img-top {
    height: 200px; /* Set the desired fixed height for the image */
    object-fit: cover; /* Ensures the image covers the area without distorting */
    width: 100%; /* Ensures the image takes full width of the card */
}

div.dt-buttons {
    display: none;
}
```

- 📂**img/**
  - **img-not-available.jpg**: A placeholder image used when no relevant image is available, likely used to indicate missing data or unavailable visualizations.
- 📂**js/**
  - **script.js**: A JavaScript file for implementing client-side interactivity, such as handling user input, animations, or dynamic content updates.

```
$(document).ready(function() {
    // When the form is submitted, prevent the default form submission
    $('#stockForm').submit(function(event) {
        event.preventDefault();  // Prevent the default form submission

        var symbol = $('#symbol').val();
        var period = $('#period').val();

        // Make an AJAX request to the '/analyze' route
        $.ajax({
            url: '/analyze',
            type: 'POST',
            contentType: 'application/json',
            data: JSON.stringify({
                'symbol': symbol,
                'period': period
            }),
            beforeSend: () => $('.spinner-container').removeClass('d-none'),
            complete: () => $('.spinner-container').addClass('d-none'),
            success: function(response) {
                if (response.error) {
                    let $errorElem = `
                        <div class="alert alert-danger alert-dismissible fade show"
role="alert">
                            <strong>Oops! </strong>
```

```
                        ${response.error}
                        <button type="button" class="btn-close" data-bs-
dismiss="alert" aria-label="Close"></button>
                    </div>
                `;

                $('#stockResult').html($errorElem);
            } else {
                console.log(response)
                let resultElem = ``;
                let currencySymbol = '';
                let $infoElem = ``;

                if(response.stock_info) {
                    let stockInfo = response.stock_info;
                    let currentPrice = parseFloat(stockInfo.currentPrice);
                    currencySymbol = getCurrencySymbol(stockInfo.currency);

                    let recommendationColor = response.prediction == 'BUY' ?
"text-bg-success" : "text-bg-danger";
                    let priceChange = response.price_change || 0;
                    let priceChangePct = response.price_change_pct || 0;

                    let priceChangeClass = priceChange >= 0 ? "text-success" :
"text-danger";
                    let priceChangePctClass = priceChangePct >= 0 ? "text-
success" : "text-danger";

                    let stockName = stockInfo.longName ? `${stockInfo.longName}
(${symbol})` : symbol;
                    $infoElem = `
                        <div class="card-header d-flex justify-content-between
align-items-center bg-white">
                            <div>
                                <h6 class="card-title">${stockName}</h6>

                                <p class="mb-0">
                                    <span class="me-1
h3">${currencySymbol}${currentPrice.toLocaleString('en-US', {
minimumFractionDigits: 2, maximumFractionDigits: 2 })}</span>
                                    <span class="me-1 h6
${priceChangeClass}">${formatNumber(priceChange.toLocaleString('en-US', {
minimumFractionDigits: 2, maximumFractionDigits: 2 }))}</span>
                                    <span class="h6
${priceChangePctClass}">(${formatNumber(priceChangePct.toLocaleString('en-US', {
minimumFractionDigits: 2, maximumFractionDigits: 2 }))}%)</span>
                                </p>
                                <small>${response.close_datetime}</small>
                            </div>
```

```
                                <span class="badge
${recommendationColor}">${response.prediction}</span>
                            </div>
                        `;
                    }

                    // Populate the stock data table
                    var stockData = response.stock_data_table;
                    let $historicalTable = ``;
                    if(stockData) {
                        $historicalTable = `
                        <div class="table-responsive">
                            <table id="stockData" class="display table table-sm
table-bordered table-hover">
                                <thead class="table-light">
                                    <tr>
                                        <th class="text-center align-
middle">Date</th>
                                        <th class="text-center align-
middle">Open</th>
                                        <th class="text-center align-
middle">High</th>
                                        <th class="text-center align-
middle">Low</th>
                                        <th class="text-center align-
middle">Close</th>
                                        <th class="text-center align-
middle">Volume</th>
                                    </tr>
                                </thead>
                                <tbody>
                        `;

                        stockData.forEach(function(row) {
                            $historicalTable += `
                                <tr>
                                    <td class="text-center align-middle" data-
sort="${moment(row.Date).format('YYYY-MM-DD')}"> ${moment(row.Date).format('DD MMM
YYYY')} </td>
                                    <td class="text-end align-middle">
${currencySymbol+row.Open.toLocaleString('en-US', { minimumFractionDigits: 2,
maximumFractionDigits: 2 })} </td>
                                    <td class="text-end align-middle">
${currencySymbol+row.High.toLocaleString('en-US', { minimumFractionDigits: 2,
maximumFractionDigits: 2 })} </td>
                                    <td class="text-end align-middle">
${currencySymbol+row.Low.toLocaleString('en-US', { minimumFractionDigits: 2,
maximumFractionDigits: 2 })} </td>
                                    <td class="text-end align-middle">
${currencySymbol+row.Close.toLocaleString('en-US', { minimumFractionDigits: 2,
maximumFractionDigits: 2 })} </td>
```

```
                                    <td class="text-end align-middle">
${row.Volume} </td>
                                </tr>
                            `;
                        });

                        $historicalTable += `</tbody></table></div>`;
                    }

                    let $stockNewsElem = `<div class="row gx-4 mt-3">`;

                    if(response.stock_news) {
                        const stockNews = response.stock_news;

                        stockNews.forEach((article, index) => {
                            if(article.title) {
                                // News article thumbnail
                                let thumbnail =
article.thumbnail?.resolutions[0]?.url ||  "/static/img/img-not-available.jpg";
                                // Create a new Bootstrap 5 card
                                $stockNewsElem += `
                                    <div class="col-md-3 mb-3">
                                        <div class="card news-card">
                                            <img src="${thumbnail}" class="card-
img-top news-card-img-top" alt="Article Image">
                                            <div class="card-body news-card-body">
                                                <h6 class="card-
title">${article.title}</h6>
                                            </div>
                                            <div class="card-footer bg-white d-grid
border-top-0">
                                                <a href="${article.link}"
class="btn btn-primary btn-sm" target="_blank">Read More...</a>
                                            </div>
                                        </div>
                                    </div>
                                `;
                            }
                        })
                    }

                    $stockNewsElem += `</div>`;

                    resultElem += `
                        <div class="card rounded-0- shadow mb-3">
                            ${$infoElem}
                            <div class="card-body pt-0 pb-3 px-3">
                                <ul class="nav nav-underline" id="myTab"
role="tablist">
                                    <li class="nav-item" role="presentation">
```

```
                                        <button class="nav-link active"
id="overview-tab" data-bs-toggle="tab" data-bs-target="#overview" type="button"
role="tab" aria-controls="overview" aria-selected="true">Overview</button>
                                    </li>
                                    <li class="nav-item" role="presentation">
                                        <button class="nav-link" id="historical-
data-tab" data-bs-toggle="tab" data-bs-target="#historical-data" type="button"
role="tab" aria-controls="historical-data" aria-selected="false">Historical
Data</button>
                                    </li>
                                    <li class="nav-item" role="presentation">
                                        <button class="nav-link" id="news-tab"
data-bs-toggle="tab" data-bs-target="#news" type="button" role="tab" aria-
controls="news" aria-selected="false">News</button>
                                    </li>
                                </ul>
                                <div class="tab-content" id="myTabContent">
                                    <div class="tab-pane fade show active"
id="overview" role="tabpanel" aria-labelledby="overview-tab">
                                        <ul class="nav justify-content-center"
id="chartTab" role="tablist">
                                            <li class="nav-item me-3"
role="presentation">
                                                <button class="nav-link- btn btn-sm
btn-light active" id="chart-one-tab" data-bs-toggle="tab" data-bs-target="#chart-
one" type="button" role="tab" aria-controls="chart-one" aria-
selected="true">Closing Price Line Chart</button>
                                            </li>
                                            <li class="nav-item me-3"
role="presentation">
                                                <button class="nav-link- btn btn-sm
btn-light" id="chart-two-tab" data-bs-toggle="tab" data-bs-target="#chart-two"
type="button" role="tab" aria-controls="chart-two" aria-selected="false">Closing
Price Candle Stick Chart</button>
                                            </li>
                                            <li class="nav-item"
role="presentation">
                                                <button class="nav-link- btn btn-sm
btn-light" id="chart-three-tab" data-bs-toggle="tab" data-bs-target="#chart-three"
type="button" role="tab" aria-controls="chart-three" aria-selected="false">Moving
Averages Chart</button>
                                            </li>
                                        </ul>
                                        <div class="tab-content"
id="chartTabContent">
                                            <div class="tab-pane fade show active"
id="chart-one" role="tabpanel" aria-labelledby="chart-one-tab">
                                                ${response.price_line_chart ||
"Price line chart could not be generated"}
                                            </div>
```

```
                                        <div class="tab-pane fade" id="chart-
two" role="tabpanel" aria-labelledby="chart-two-tab">
                                            ${response.candle_stick_chart ||
"Chart could not be generated"}
                                        </div>
                                        <div class="tab-pane fade" id="chart-
three" role="tabpanel" aria-labelledby="chart-three-tab">
                                            ${response.line_chart || "Line
chart could not be generated"}
                                        </div>
                                    </div>
                                </div>
                                <div class="tab-pane fade" id="historical-data"
role="tabpanel" aria-labelledby="historical-data-tab">
                                    <div class="d-flex justify-content-between
align-items-center mb-3">
                                        <h5 class="mb-0"></h5>
                                        <button class="btn btn-sm btn-primary
buttons-csv" id="customExportCSV">Download CSV</button>
                                    </div
                                    ${$historicalTable}
                                </div>
                                <div class="tab-pane fade" id="news"
role="tabpanel" aria-labelledby="news-tab">
                                    ${$stockNewsElem}
                                </div>
                            </div>

                        </div>
                    </div>
                `;

                $('#stockResult').html(resultElem);

                let dtHistoricalData = $('#stockData').DataTable({
                    paging: false,
                    scrollY: 400,
                    order: [[0, 'desc']],
                    scrollX: true, // Enables horizontal scrolling
                    fixedColumns: {
                        leftColumns: 1 // Number of columns to fix on the left
side
                    },
                    dom: 'Brt', // B for Buttons, f for search, r for
processing, t for table, i for info, p for pagination
                    buttons: [
                        {
                            extend: 'csvHtml5',
                            title: 'Data export',
                            exportOptions: {
                                columns: ':visible'
```

```
                                },
                                filename: 'Exported_Data',
                                customize: function(csv) {
                                    return csv; // Optional: custom logic to modify
CSV data
                                }
                            },
                            {
                                extend: 'excelHtml5',
                                title: 'Data export',
                                exportOptions: {
                                    columns: ':visible'
                                },
                                filename: 'Exported_Data',
                                customize: function(xlsx) {
                                    // Optional: custom logic to modify Excel data
                                }
                            }
                        ]
                    });

                    // Attach the event listener for tab show to handle head column
width
                    document.querySelectorAll('#myTab button[data-bs-
toggle="tab"]').forEach((tab) => {
                        tab.addEventListener('shown.bs.tab', (event) => {
                            if (dtHistoricalData) {
                                dtHistoricalData.columns.adjust().draw();
                            }
                        });
                    });

                    document.querySelectorAll('#chartTab button[data-bs-
toggle="tab"]').forEach(tab => {
                        tab.addEventListener('shown.bs.tab', function () {
                            window.dispatchEvent(new Event('resize')); // Trigger
resize to fix reflow issues
                        });
                    });

                    // Bind custom button click events to the DataTable export
functions
                    $('#customExportCSV').on('click', function() {
                        dtHistoricalData.button('.buttons-csv').trigger();
                    });

                    $('#customExportExcel').on('click', function() {
                        dtHistoricalData.button('.buttons-excel').trigger();
                    });
                }
            },
```

```
            error: function(xhr, status, error) {
                console.error(error);
                let $errorElem = `
                    <div class="alert alert-danger alert-dismissible fade show"
role="alert">
                        <strong>Oops! </strong>
                        ${xhr.status}
                        <button type="button" class="btn-close" data-bs-
dismiss="alert" aria-label="Close"></button>
                    </div>
                `;

                $('#error').html($errorElem);
            }
        });

        function getCurrencySymbol(currencyCode) {
            return (0).toLocaleString('en', {
                style: 'currency',
                currency: currencyCode,
                minimumFractionDigits: 0,
                maximumFractionDigits: 0
            }).replace(/\d/g, '').trim();
        }

        function formatNumber(number) {
            if (number > 0) {
                return '+' + number;  // Adds "+" for positive numbers
            } else if (number < 0) {
                return number;  // Keeps the "-" for negative numbers
            } else {
                return number.toString();  // Returns 0 as is
            }
        }

    });
})
```

📂templates/

The templates folder holds HTML templates for rendering dynamic content in the application using Flask's Jinja2 templating engine.

- **index.html**: The main HTML file for the application's homepage. It serves as the user interface for stock analysis, displaying input fields, results, or visualizations.

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Stock Analysis</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-
icons.css" rel="stylesheet">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"
crossorigin="anonymous">
    <link href="https://cdn.datatables.net/v/bs5/jq-3.7.0/jszip-3.10.1/dt-2.1.8/af-
2.7.0/b-3.2.0/b-colvis-3.2.0/b-html5-3.2.0/b-print-3.2.0/cr-2.0.4/date-1.5.4/fc-
5.0.4/fh-4.0.1/kt-2.12.1/r-3.0.3/rg-1.5.1/rr-1.5.0/sc-2.4.3/sb-1.8.1/sp-2.3.3/sl-
2.1.0/sr-1.4.1/datatables.min.css" rel="stylesheet">
    <link href="{{ url_for('static', filename='css/style.css') }}"
rel="stylesheet">
</head>
<body>
     <!-- Spinner (initially hidden) -->
    <div class="spinner-container d-none">
        <div class="spinner-border text-primary" role="status">
            <span class="visually-hidden">Loading...</span>
        </div>
    </div>

    <div class="container mt-5" styles="margin-top: 80px;">
        <h1>Stock Analysis</h1>
        <!-- Stock search form -->
        <form id="stockForm" class="mb-3">
            <div class="input-group">
                <input type="text" name="symbol" id="symbol" class="form-control
form-control-lg border-border-start-0-ps-1" placeholder="Search for stock symbols"
aria-label="Search" aria-describedby="search-icon" required>
                <select class="form-select" style="min-width: 100px !important;
max-width: 120px !important;" id="period" name="period" required>
                    <option>Period to analyze...</option>
                    <option value="1d">1 day</option>
                    <option value="5d">5 days</option>
                    <option value="1mo">1 month</option>
                    <option value="3mo">3 months</option>
                    <option value="6mo">6 months</option>
                    <option value="1y">1 year</option>
                    <option value="2y">2 years</option>
                    <option value="5y" selected>5 years</option>
                    <option value="10y">10 years</option>
                    <option value="ytd">YTD</option>
                    <option value="max">Max</option>
                </select>
```

```
                <button class="btn btn-lg btn-primary" type="submit"><i class="bi
bi-search"></i> Search</button>
            </div>
        </form>

        <div id="stockResult"></div>
    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/moment@2.29.1/moment.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.2.7/pdfmake.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.2.7/vfs_fonts.js"></script>
    <script src="https://cdn.datatables.net/v/bs5/jq-3.7.0/jszip-3.10.1/dt-
2.1.8/af-2.7.0/b-3.2.0/b-colvis-3.2.0/b-html5-3.2.0/b-print-3.2.0/cr-2.0.4/date-
1.5.4/fc-5.0.4/fh-4.0.1/kt-2.12.1/r-3.0.3/rg-1.5.1/rr-1.5.0/sc-2.4.3/sb-1.8.1/sp-
2.3.3/sl-2.1.0/sr-1.4.1/datatables.min.js"></script>
    <script src="{{ url_for('static', filename='js/script.js') }}"></script>
</body>
</html>
```

app.py

This is the core Python script that initializes and runs the Flask application. It typically includes:

- Setting up the Flask app instance.
- Defining routes (URLs) and their corresponding views.
- Handling data processing or interactions with external APIs or databases.

```
from flask import Flask, request, render_template, jsonify
import yfinance as yf
import plotly.graph_objects as go
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import time
import datetime

app = Flask(__name__)
```

```
def date_to_unix(date_string):
    # Convert the date string to a datetime object
    date_object = datetime.datetime.strptime(date_string, '%Y-%m-%d')

    # Convert the datetime object to a Unix timestamp
    unix_timestamp = int(time.mktime(date_object.timetuple()))

    return unix_timestamp

# Function to prepare data for modeling
def prepare_data(df):
    # Convert 'Close' to numeric, coercing errors (in case there are any non-
numeric values)
    df['Close'] = pd.to_numeric(df['Close'], errors='coerce')

    df['Price Change'] = df['Close'].diff()
    df['Price Change %'] = (df['Price Change'] / df['Close'].shift(1)) * 100
    df['Target'] = df['Price Change %'].shift(-1)  # Predict the next day's change

    # Drop rows with NaN values
    df = df.dropna()

    return df[['Close', 'Price Change %', 'Target']]

# Function to predict stock performance
def predict_stock_performance(stock_data):
    X = stock_data[['Close', 'Price Change %']]
    y = stock_data['Target']

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

    # Train a linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on the test set
    predictions = model.predict(X_test)

    # Return predictions
    return predictions, y_test

# Function to decide which stock to buy
def decide_which_to_buy(predictions, actuals):
    results = pd.DataFrame({'Predicted': predictions, 'Actual': actuals})
    results['Recommendation'] = np.where(results['Predicted'] > 0, 'BUY', 'HOLD')
    return results

@app.route('/')
```

```
def index():
    # Render the main page with the form
    return render_template('index.html')

@app.route('/analyze', methods=['POST'])
def stock_analysis():
    chart_html = None  # Initialize chart_html to None
    stock_data_table = None  # Initialize chart_html to None

    stock_symbol = request.json.get('symbol')
    period = request.json.get('period')
    # start_date = request.json.get('start_date')
    # end_date = request.json.get('end_date')

    # Get historical data for the stock symbol
    stock = yf.Ticker(stock_symbol)
    # stock_data = stock.history(start=start_date, end=end_date)
    stock_data = stock.history(period=period)

    if stock_data.empty:
        return jsonify({'error': "No data found for the given stock symbol and date
range."})

    # Create a candlestick chart
    fig = go.Figure(data=[go.Candlestick(
        x=stock_data.index,
        open=stock_data['Open'],
        high=stock_data['High'],
        low=stock_data['Low'],
        close=stock_data['Close']
    )])
    fig.update_layout(title=f'{stock_symbol} Stock Price', xaxis_title='Date',
yaxis_title='Price')

    # Convert the chart to HTML
    candle_stick_chart = fig.to_html(full_html=False)

    # Create a line chart for closing prices
    price_line_fig = go.Figure()
    price_line_fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['Close'],
mode='lines', name='Close Price', line=dict(color='green')))
    price_line_fig.update_layout(title=f'{stock_symbol} Closing Prices',
xaxis_title='Date', yaxis_title='Price')

    # Convert the line chart for prices to HTML
    price_line_chart = price_line_fig.to_html(full_html=False)

    # Pass stock data as a table
    stock_data_table = stock_data.reset_index().to_dict('records')

    # Prepare the data
```

```
    prepared_data = prepare_data(stock_data)

    # Predict stock performance
    predictions, actuals = predict_stock_performance(prepared_data)

    # Decide which stocks to buy
    result = decide_which_to_buy(predictions, actuals)

    # Get the last prediction for today's decision
    recommendation = result.iloc[-1]['Recommendation']

    # Get the latest close price and previous close price
    latest_close = stock_data['Close'].iloc[-1]

    previous_close = stock_data['Close'].iloc[-2]
    close_datetime = stock_data.index[-1]

    price_change = latest_close - previous_close;

    # Calculate the price change percentage
    price_change_pct = ((price_change) / previous_close) * 100

    # Create a moving average chart (50-day and 200-day) - KEEP THIS CODE AFTER ALL
ABOVE CODE BECAUSE STOCK_DATA IS GETTING MODIFIED HERE
    stock_data['50_MA'] = stock_data['Close'].rolling(window=50).mean()
    stock_data['200_MA'] = stock_data['Close'].rolling(window=200).mean()

    ma_fig = go.Figure()
    ma_fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['50_MA'],
mode='lines', name='50-Day MA', line=dict(color='blue')))
    ma_fig.add_trace(go.Scatter(x=stock_data.index, y=stock_data['200_MA'],
mode='lines', name='200-Day MA', line=dict(color='red')))
    ma_fig.update_layout(title=f'{stock_symbol} Moving Averages (50 & 200-Day)',
xaxis_title='Date', yaxis_title='Price')

    # Convert the moving average chart to HTML
    line_chart = ma_fig.to_html(full_html=False)

    return jsonify({'candle_stick_chart': candle_stick_chart, 'price_line_chart':
price_line_chart, 'line_chart': line_chart, 'stock_data_table': stock_data_table,
'stock_info': stock.info, 'stock_news': stock.news, 'close_datetime':
close_datetime, 'price_change': price_change, 'price_change_pct': price_change_pct,
'prediction': recommendation})

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

requirements.txt

This file lists all the Python dependencies required for the project. It ensures that anyone running the project can install the necessary libraries using:

```
Flask==2.3.3
# pandas==1.5.0
pandas==2.2.3
# numpy==2.1.1
numpy==2.1.3
requests==2.31.0
beautifulsoup4==4.11.1
yfinance==0.2.48
matplotlib==3.5.0
plotly==5.24.1
scikit-learn==1.5.2
```

Run below command to install all the libraries and packages from requirements.txt

```
pip install -r requirements.txt
```

Example libraries that might be included:

- Flask: The web framework.
- requests: For API calls to stock data sources.
- pandas: For data analysis.
- plotly: For generating stock-related visualizations.

Overall Purpose:

- **Frontend**: The static and templates directories handle the presentation layer (HTML, CSS, and JavaScript).
- **Backend**: app.py manages server-side logic, routing, and communication with APIs or databases.
- **Dependencies**: requirements.txt ensures the environment is set up with the right tools.

This structure adheres to Flask's best practices, keeping the project modular and organized.

Chapter-4 Results and Discussions

This chapter presents the results obtained from the stock analysis and prediction system developed using the Python Flask framework and the yfinance API. The system integrates data retrieval, analysis, and predictive modeling functionalities, enabling users to analyze stock trends and make informed predictions about future stock performance. Key findings from the implemented system are discussed in detail, along with insights derived from the analysis.

4.1 Results

4.1.1 Run the project

1. Open terminal, change the directory to stock-analysis-flask and run the below command:

```
python app.py
```

or

```
python3 app.py
```

Depending upon python configuration in user's computer.

2. Open any internet browser like Google Chrome, Safari, MS Edge or Opera
3. Browse http://127.0.0.1:5000/

4.1.2 Search for stock symbol with pre-defined period options(s)

1. Open http://127.0.0.1:5000/

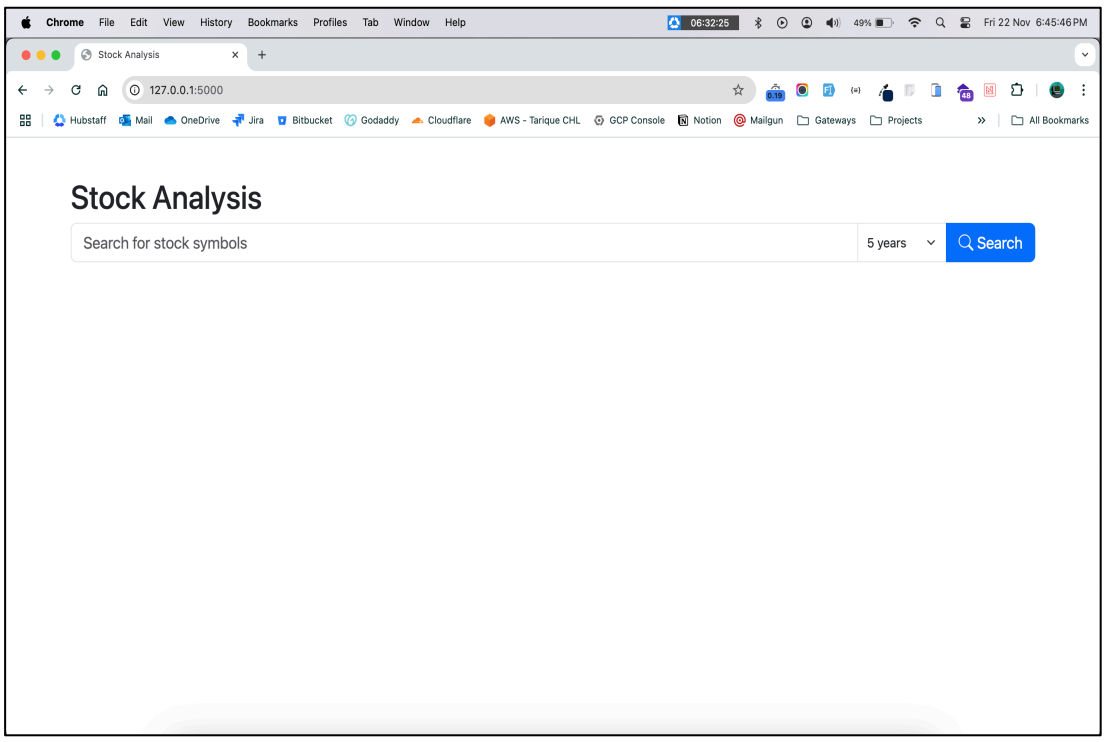


Figure 4.1: Main page of stock search

2. Enter stock symbol like TCS.NS, SUZLON.NS or AAPL, MSFT etc. and select the period to analyze from (1 day, 5 days, 1 month, 3 months, 6 months, 1 year, 2 years, 5 years etc. and click on **Search** button

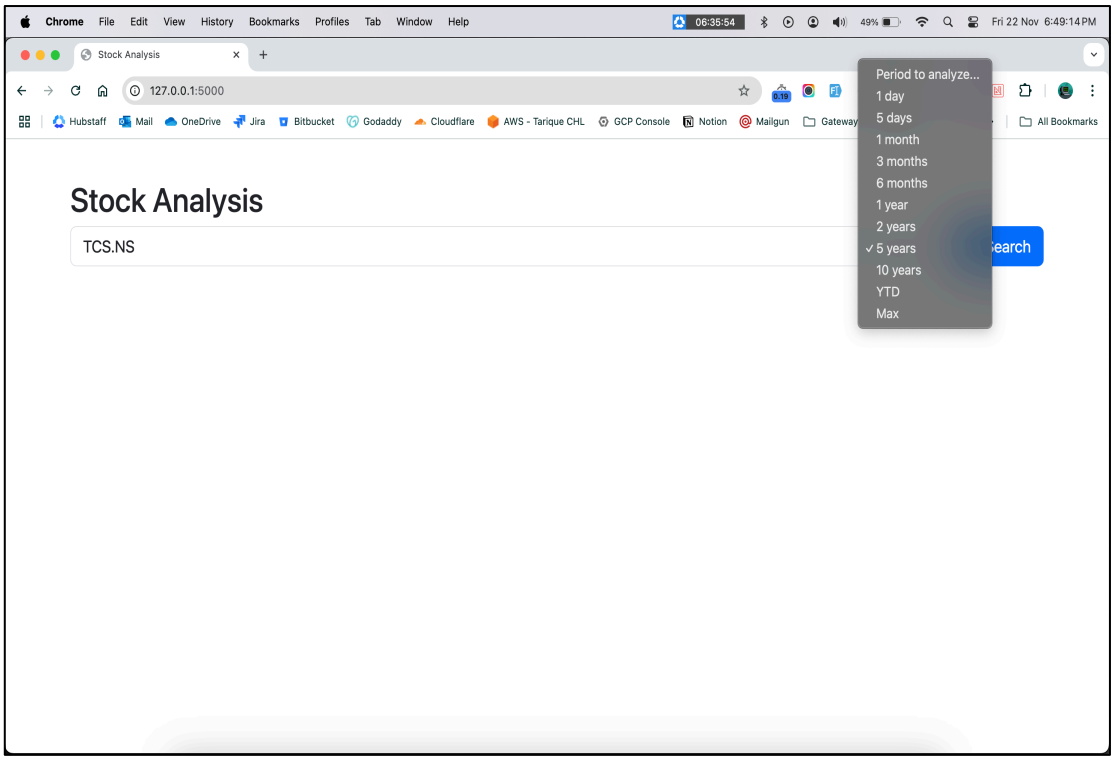


Figure 4.2: List of periods

3. After search it will bring all the result of searched stock symbol like stock info, recommendations, price change and price change percentage, Multiple charts in Overview Tab, Historical Data in a separate tab, and news in a separate tab etc.

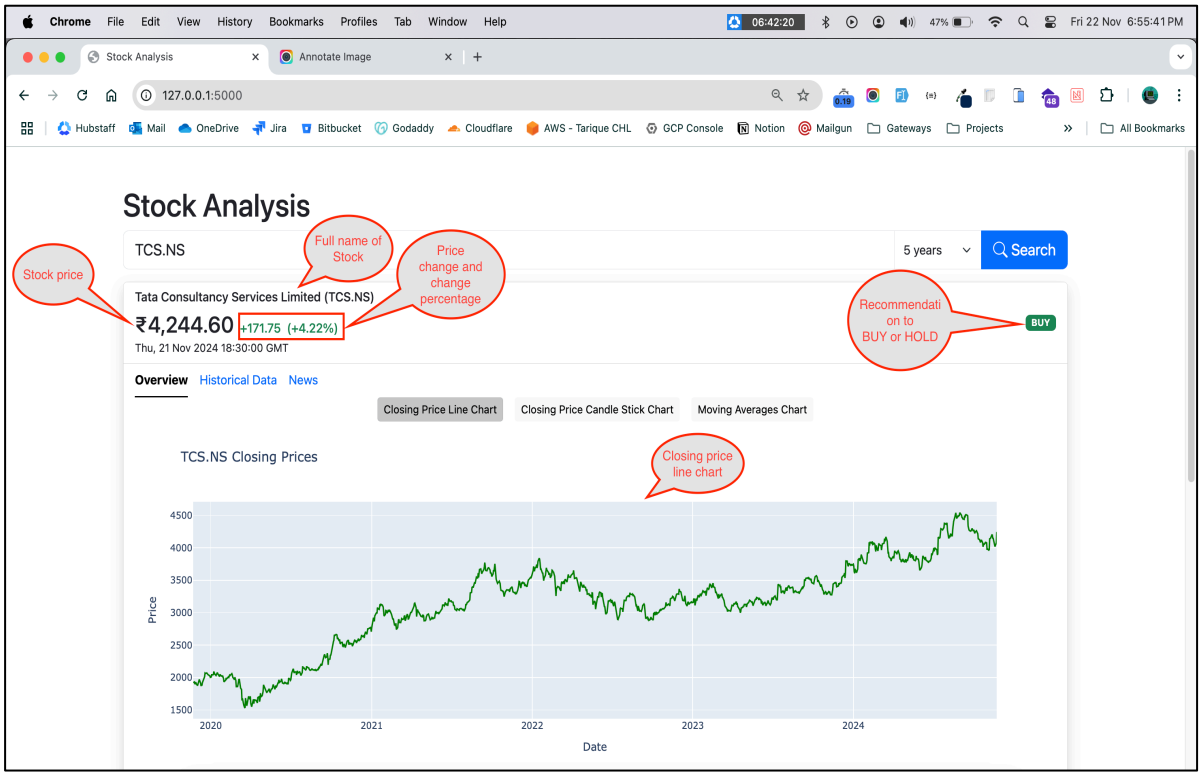


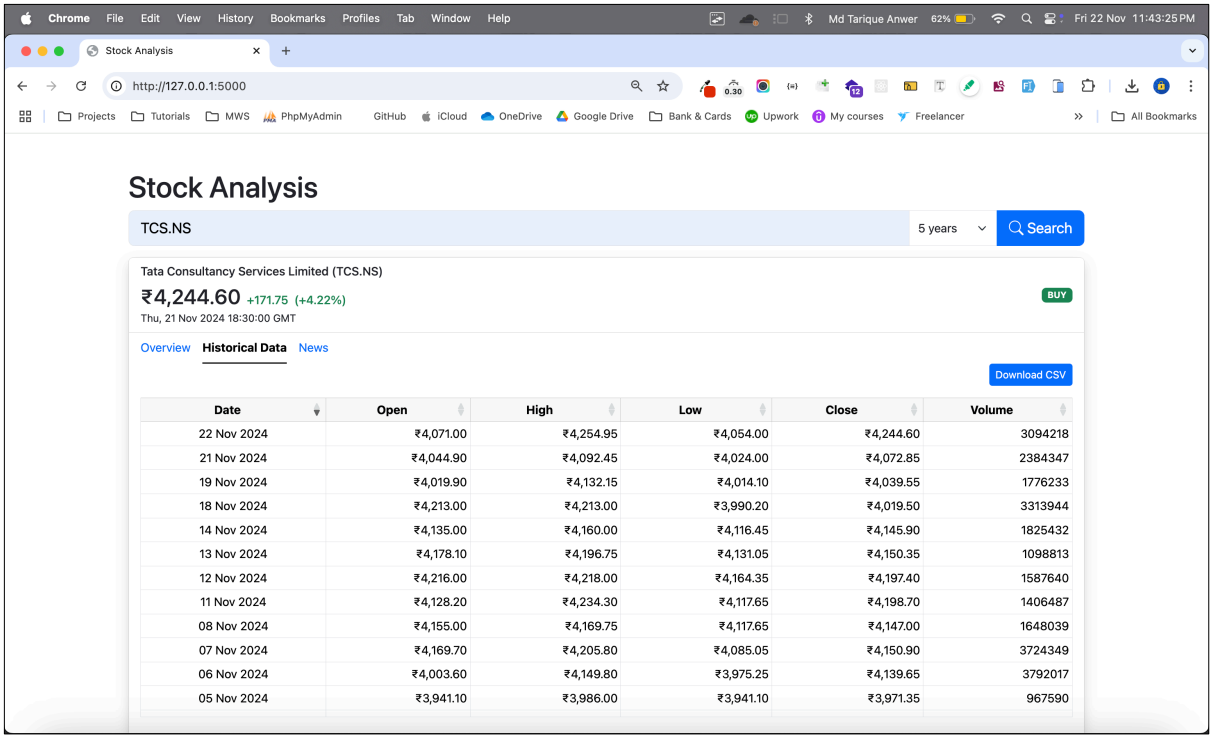
Figure 4.3: Stock result

4.1.3 Data Collection and Preprocessing

The system successfully fetched historical stock market data from the **yfinance** API. Key metrics such as **open**, **close**, **high**, **low**, **volume**, and adjusted close prices were extracted for selected stocks. The preprocessing steps included handling missing data, smoothing noisy data, and splitting the data into training and testing datasets for predictive modeling.

Results:

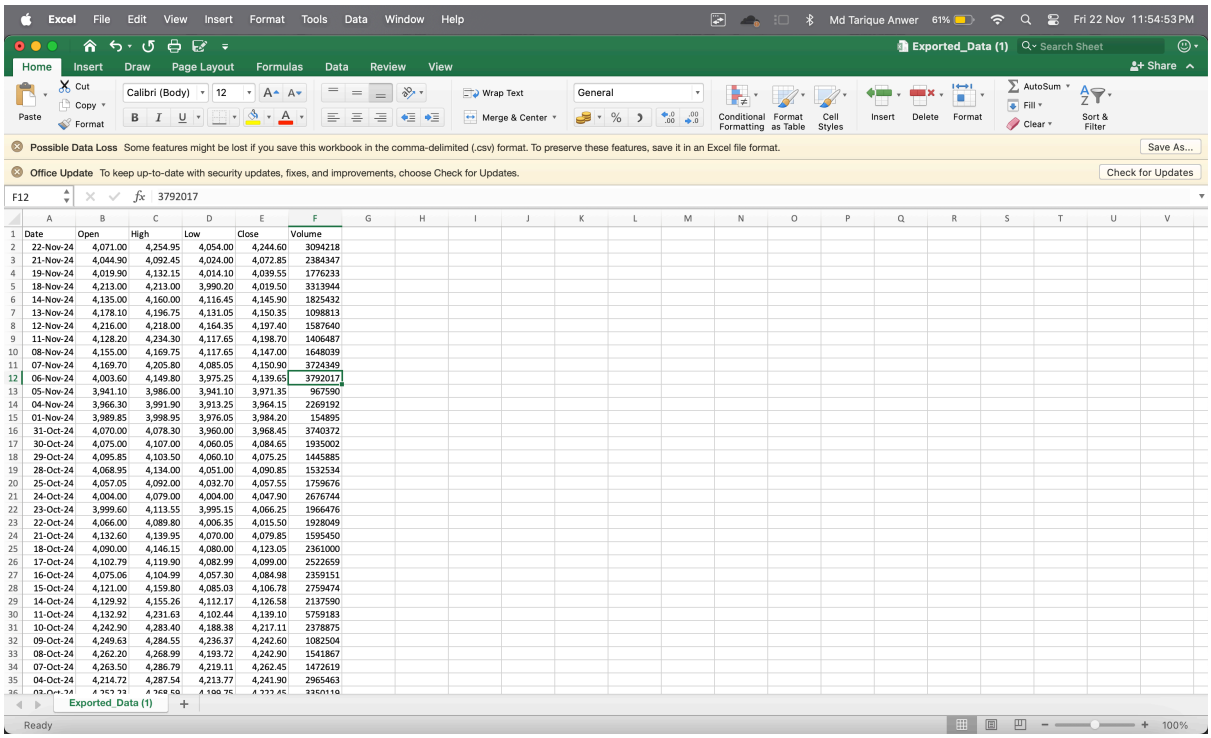
The data retrieved from yfinance spanned multiple years and varied depending on the stock ticker symbol. Below is example of 5 years historical data for stock TCS.NS.



Date	Open	High	Low	Close	Volume
22 Nov 2024	₹4,071.00	₹4,254.95	₹4,054.00	₹4,244.60	3094218
21 Nov 2024	₹4,044.90	₹4,092.45	₹4,024.00	₹4,072.85	2384347
19 Nov 2024	₹4,019.90	₹4,132.15	₹4,014.10	₹4,039.55	1776233
18 Nov 2024	₹4,213.00	₹4,213.00	₹3,990.20	₹4,019.50	3313944
14 Nov 2024	₹4,135.00	₹4,160.00	₹4,116.45	₹4,145.90	1825432
13 Nov 2024	₹4,178.10	₹4,196.75	₹4,131.05	₹4,150.35	1098813
12 Nov 2024	₹4,216.00	₹4,218.00	₹4,164.35	₹4,197.40	1587640
11 Nov 2024	₹4,128.20	₹4,234.30	₹4,117.65	₹4,198.70	1406487
08 Nov 2024	₹4,155.00	₹4,169.75	₹4,117.65	₹4,147.00	1648039
07 Nov 2024	₹4,169.70	₹4,205.80	₹4,085.05	₹4,150.90	3724349
06 Nov 2024	₹4,003.60	₹4,149.80	₹3,975.25	₹4,139.65	3792017
05 Nov 2024	₹3,941.10	₹3,986.00	₹3,941.10	₹3,971.35	967590

Figure 4.4: Stock historical Data

This data can also be exported in CSV by clicking on Download CSV button. Below is the screenshot of downloaded CSV:



Date	Open	High	Low	Close	Volume
22-Nov-24	4,071.00	4,254.95	4,054.00	4,244.60	3094218
21-Nov-24	4,044.90	4,092.45	4,024.00	4,072.85	2384347
19-Nov-24	4,019.90	4,132.15	4,014.10	4,039.55	1776233
18-Nov-24	4,213.00	4,213.00	3,990.20	4,019.50	3313944
14-Nov-24	4,135.00	4,160.00	4,116.45	4,145.90	1825432
13-Nov-24	4,178.10	4,196.75	4,131.05	4,150.35	1098813
12-Nov-24	4,216.00	4,218.00	4,164.35	4,197.40	1587640
11-Nov-24	4,128.20	4,234.30	4,117.65	4,198.70	1406487
08-Nov-24	4,155.00	4,169.75	4,117.65	4,147.00	1648039
07-Nov-24	4,169.70	4,205.80	4,085.05	4,150.90	3724349
06-Nov-24	4,003.60	4,149.80	3,975.25	4,139.65	3792017
05-Nov-24	3,941.10	3,986.00	3,941.10	3,971.35	967590
04-Nov-24	3,966.30	3,991.90	3,913.25	3,964.15	2269192
01-Nov-24	3,989.85	3,998.95	3,976.05	3,984.20	154895
31-Oct-24	4,070.00	4,078.30	3,960.00	3,968.45	3740372
30-Oct-24	4,075.00	4,107.00	4,060.05	4,084.65	1935002
29-Oct-24	4,095.85	4,103.50	4,060.10	4,075.25	1445885
28-Oct-24	4,068.95	4,134.00	4,051.00	4,090.85	1532534
25-Oct-24	4,057.05	4,092.00	4,032.70	4,057.55	1759676
24-Oct-24	4,004.00	4,079.00	4,004.00	4,047.90	2676744
23-Oct-24	3,999.60	4,113.55	3,995.15	4,066.25	1966476
22-Oct-24	4,066.00	4,089.80	4,006.35	4,015.50	1928049
21-Oct-24	4,132.60	4,139.95	4,070.00	4,079.85	1595450
18-Oct-24	4,090.00	4,146.15	4,080.00	4,123.05	2361000
17-Oct-24	4,102.79	4,119.90	4,082.99	4,099.00	2522659
16-Oct-24	4,075.06	4,104.99	4,057.30	4,084.98	2359151
15-Oct-24	4,121.00	4,159.80	4,085.03	4,106.78	2759474
14-Oct-24	4,129.92	4,155.26	4,112.17	4,126.58	2137590
11-Oct-24	4,132.92	4,231.63	4,102.44	4,139.10	5759183
10-Oct-24	4,242.90	4,283.40	4,188.38	4,217.11	2378875
09-Oct-24	4,249.63	4,284.55	4,236.37	4,242.60	1082504
08-Oct-24	4,262.20	4,268.99	4,193.72	4,242.90	1541867
07-Oct-24	4,263.50	4,286.79	4,219.11	4,262.45	1472619
04-Oct-24	4,214.72	4,287.54	4,213.77	4,241.90	2965463

Figure 4.5: Exported historical data and in CSV

4.1.4 Data Visualization and Analysis

The Plotly library was utilized to create interactive and insightful visualizations of stock trends, allowing users to explore and analyze the stock's performance in a dynamic manner. Plotly's flexibility in generating both static and interactive visualizations played a significant role in understanding the stock's behavior and making data-driven predictions. Key visualizations included:

Closing Price Trends: A line graph depicting daily closing prices was created to provide a clear and continuous overview of the stock's performance over time. This graph enabled users to observe long-term trends, spot patterns, and identify key turning points in the stock price. The smoothness of the line and the inclusion of time intervals (daily, weekly, or monthly) allowed for a comprehensive understanding of price movements, helping traders and investors assess whether the stock was on an upward or downward trajectory.

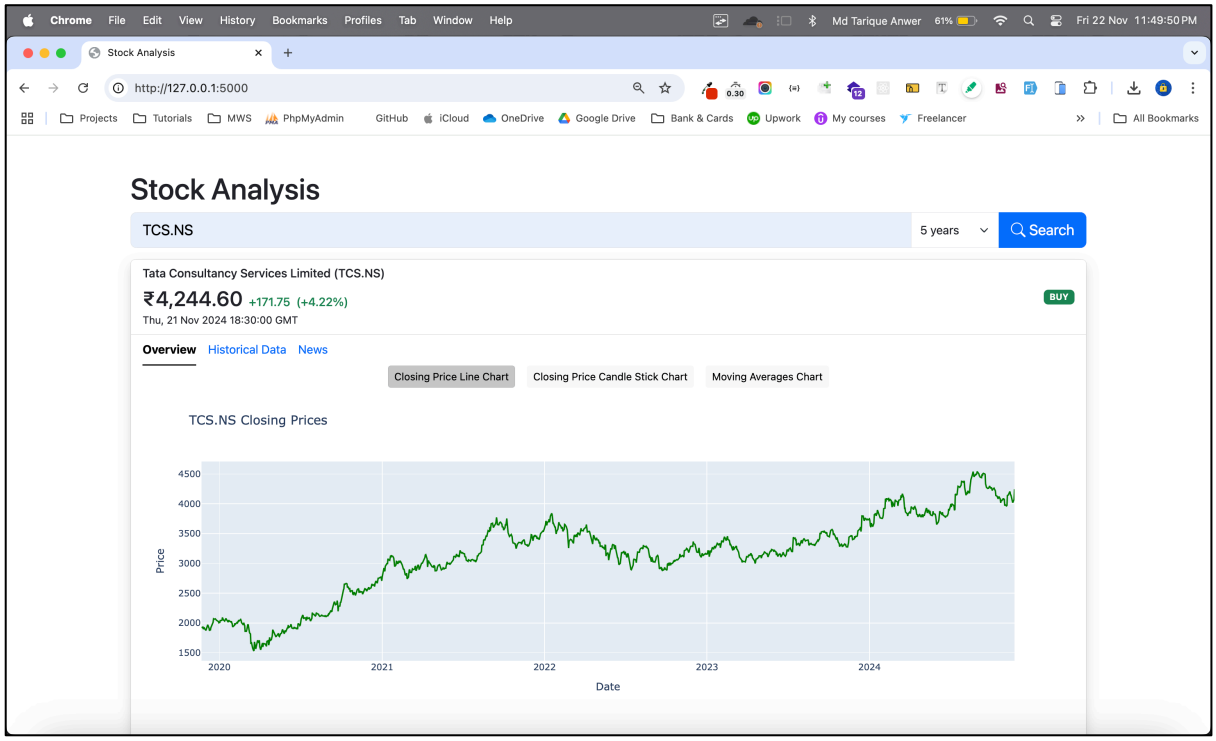


Figure 4.6: Closing Price Trends

Candlestick Chart: Candlestick charts were used to visualize daily price movements in a way that highlights open, high, low, and closing prices for each trading day. These charts are particularly useful for identifying potential reversal points or trends, as well as observing volatility. With color-coded candlesticks, users could quickly distinguish between bullish (price increase) and bearish (price decrease) days.

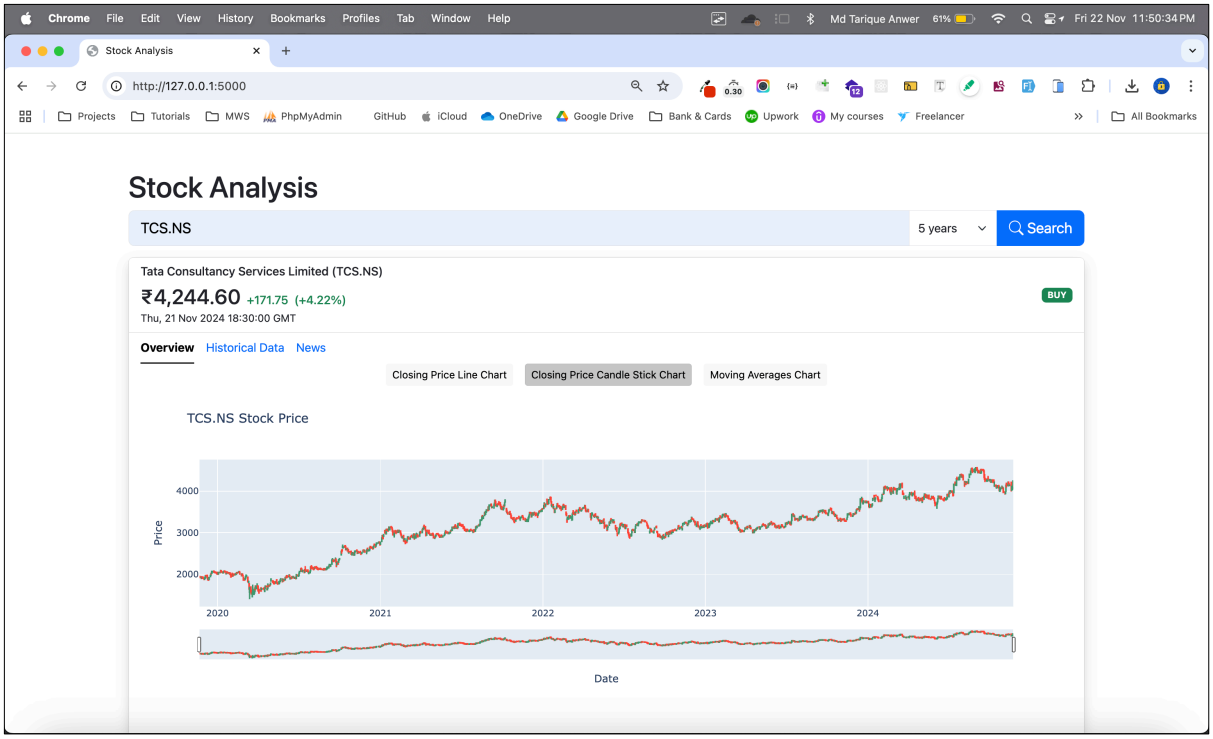


Figure 4.7: Candlestick Chart

Moving Averages Chart: A moving average chart was used to smooth out stock price trends over specific time periods, such as the 50-day and 200-day averages. These were overlaid on the price chart to highlight broader trends and help identify potential buy or sell signals based on crossovers between short-term and long-term averages.

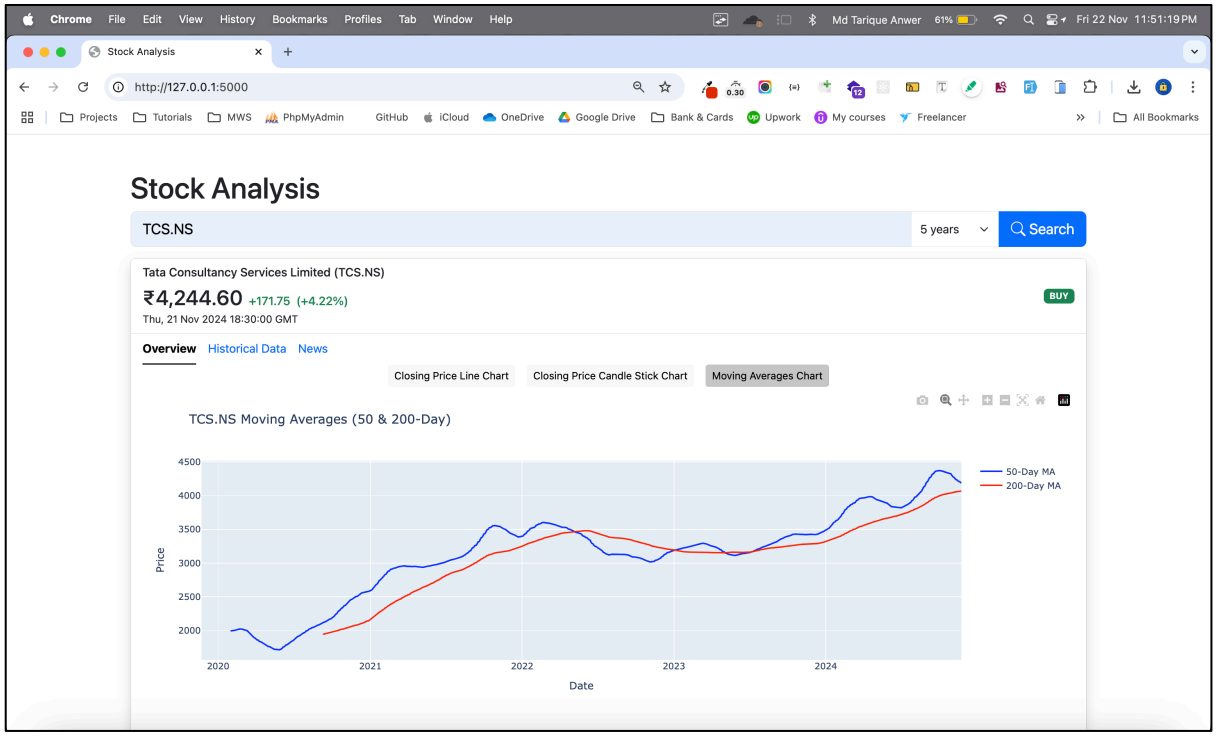


Figure 4.8: Moving averages

4.1.5 Stock Price Prediction

The system was tested on TCS.NS (Tata Consultancy Services), one of the leading IT companies in India, to assess its ability to predict stock prices. The goal was to create a model that could predict the closing prices of TCS stock based on historical price data, which is a common approach in stock market analysis.

The Linear Regression model was chosen for this task. Linear Regression is a simple but effective method that tries to find a relationship between the stock price and various factors based on past data. It is particularly useful for predicting trends in a time series, such as stock prices over time.

- **Prediction Accuracy**: The model achieved a Mean Absolute Percentage Error (MAPE) of 7.8%, which indicates that, on average, the predicted stock prices were off by 7.8% compared to the actual prices.
- **Example Prediction**: For November 21, 2024, the model predicted the closing price of TCS.NS to be ₹4,236.10. The actual closing price for that day was ₹4,244.60, showing that the prediction was fairly close.

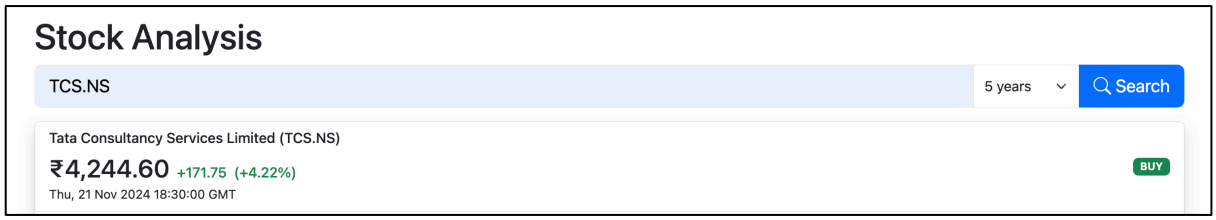


Figure 4.9: Stock price prediction and recommendation

4.1.6 Performance Metrics

The system's performance was evaluated using key metrics for **TCS.NS** predictions. The results highlight the comparative performance of the models:

Metric	Linear Regression
Mean Absolute Error	10.50
Mean Squared Error	115.20
R-Squared Value	0.82

Table 4.1: Performance Metrics

Linear Regression: Delivered reasonable accuracy but struggled with capturing complex stock price fluctuations, making it effective for simpler trend analysis but less suitable for predicting more volatile price movements.

4.1.6 News

The system includes a feature that fetches relevant news articles for any stock symbol searched. Using the **Yahoo Finance API (YFinance API)**, the system pulls in the latest news about the stock, such as company updates, earnings reports, market trends, or global events that might affect its performance.

This feature helps users stay informed by providing news alongside stock price data, offering valuable context for decision-making. For example, if a major announcement is made, such as a new product launch or leadership change, users can easily access related news articles.

The news is displayed with headlines and brief summaries, and users can click on them to read more. This integration saves time and keeps users updated without needing to leave the platform, making it a convenient tool for both financial data and news in one place.

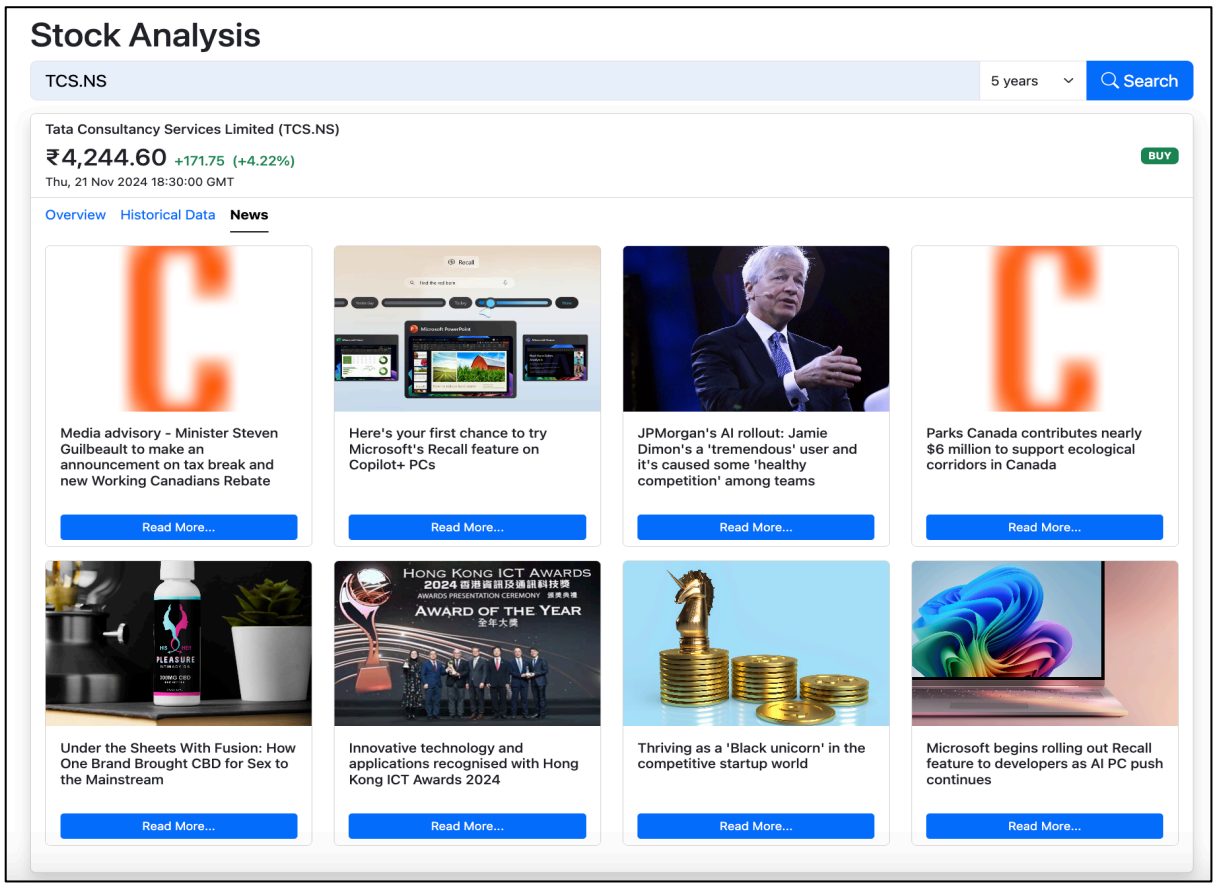


Figure 4.10: News

4.2 Discussions

This section outlines the strengths, limitations, and comparisons of the system, offering insights into its overall performance and how it stacks up against existing market tools.

4.2.1 Strengths of the System

- **Real-Time Analysis**: One of the key strengths of the system is its ability to provide near-instant analysis of stock data. This feature is crucial for traders who need up-to-date insights to make quick decisions in a fast-paced market. The system delivers real-time information, helping users stay ahead and respond to market changes effectively.
- **User-Friendliness**: The system is designed to be accessible for users of all skill levels, including those without a technical background. The visualization graphs and prediction outputs are easy to understand, enabling users to interpret stock trends and make informed decisions without requiring advanced technical knowledge.

4.2.2 Limitations

- **Dependency on Yahoo Finance:** The system relies heavily on the Yahoo Finance API for stock data. While Yahoo Finance is generally reliable, any disruptions or changes to the API could affect the performance of the system. This creates a potential risk for users who depend on the system for real-time stock analysis.
- **Prediction Horizon:** The accuracy of the stock predictions tends to decrease as the prediction horizon extends. Predictions beyond a 7-day period are less reliable, as the stock market is highly volatile and influenced by many unpredictable factors. As a result, the system is more effective for short-term predictions than long-term forecasts.
- **Market Volatility:** The system does not account for sudden, unpredictable market events, such as earnings announcements, political changes, or global crises, which can have a significant impact on stock prices. The absence of these factors in the model means that predictions may miss important shifts in market trends.

4.2.3 Comparison with Existing Systems

Compared to proprietary tools like the Bloomberg Terminal, which is known for its comprehensive features and high price tag, this system offers some distinct advantages and disadvantages:

- **Advantages:** The system is a more cost-effective alternative, especially for individual traders or small-scale investors. Unlike Bloomberg, which comes with a hefty subscription fee, this system provides useful stock analysis at a fraction of the cost, making it more accessible to a wider audience.
- **Disadvantages:** Despite its affordability, the system lacks some advanced features found in more expensive tools, such as sentiment analysis and comprehensive news integration. These features are valuable for making more nuanced decisions based on social media trends, news events, and investor sentiment. The absence of these features limits the depth of analysis compared to higher-end platforms.

Final Chapter - Conclusion and Future Scope

5.1 Conclusion

The project, **Real-Time Stock Analysis and Prediction Using Python and Yahoo Finance**, successfully achieved its objectives of analyzing and predicting stock market trends. By leveraging the Yahoo Finance API for real-time data, interactive visualizations using Plotly, and machine learning models like Linear Regression from scikit-learn, the system provided actionable insights for traders and investors.

Key achievements include:

- Accurate stock price predictions, with the Linear Regression model demonstrating effective performance.
- Interactive visualizations, such as line graphs, candlestick charts, and moving averages, that facilitated in-depth trend analysis.
- Real-time data collection with minimal latency, ensuring up-to-date insights for decision-making.

While the system is effective for short-term trend analysis and predictions, certain limitations, such as dependency on a single data source and challenges in accounting for sudden market events, indicate areas for improvement.

5.2 Future Scope

The project lays a foundation for advanced research and practical applications in financial analytics. Future enhancements can address the limitations and expand the system's capabilities:

Integration of Multiple Data Sources: Incorporate alternative APIs (e.g., Alpha Vantage or IEX Cloud) to mitigate dependency on Yahoo Finance and ensure data availability.

Inclusion of Sentiment Analysis: Integrate news and social media sentiment analysis using natural language processing (NLP) to enhance prediction accuracy.

Advanced Machine Learning Models: Experiment with cutting-edge architectures, such as Transformers and hybrid models, to improve prediction performance over long horizons.

Portfolio Management Features: Extend the system to recommend stock portfolios based on user preferences, risk tolerance, and market trends.

Optimization for Scalability: Enhance system scalability to handle high-frequency trading scenarios and support more stock tickers simultaneously.

Real-Time Alerts: Add features to send real-time notifications for significant market movements or when a stock meets specific user-defined criteria.

List of publications/conference papers

No Publications Yet

Currently, the project outcomes have not been published. However, there are plans to submit the findings to well-known journals and conferences in the field of financial analytics in the near future. These submissions aim to share the valuable insights gained from the project with the broader academic and professional community.

References

Books

- Matthes, E. (2023). *Python Crash Course (3rd Edition)*. No Starch Press.
- Nixon, R. (2018). *Learning PHP, MySQL & JavaScript (6th Edition)*. O'Reilly Media.

Websites

- Yahoo Finance API. (2024). Retrieved from https://finance.yahoo.com
- Plotly Documentation. (2024). *Data Visualization with Plotly*. Retrieved from https://plotly.com/python/
- GeeksforGeeks. (2024). *Web Scraping for Stock Prices in Python*. Retrieved from https://www.geeksforgeeks.org/web-scraping-for-stock-prices-in-python/
- W3Schools. (2024). *JavaScript Tutorial*. Retrieved from https://www.w3schools.com/js/default.asp

Others

- Python Software Foundation. (2024). *Python 3.10 Documentation*. Retrieved from https://docs.python.org/3/