CHAPTER-1

INTRODUCTION

1 INTRODUCTION

The stock market plays a critical role in global economies, acting as a barometer for economic health and offering opportunities for investment and wealth creation. However, predicting stock price movements is one of the most challenging tasks due to the inherent volatility, randomness, and the influence of numerous factors. Traditional stock forecasting methods primarily rely on numerical analysis of historical data, such as stock prices, volume, and market indices. While these techniques provide a foundation for analysis, they often fail to capture the broader context of market dynamics driven by human behavior, external events, and public sentiment.

In recent years, the availability of large-scale textual data, such as financial news articles, analyst reports, and social media discussions, has opened new avenues for stock price prediction. Textual data encapsulates qualitative insights, such as investor sentiment, market rumors, and geopolitical events, that are not reflected in numerical data alone. By leveraging Natural Language Processing (NLP) techniques, it is possible to quantify this textual information into meaningful features that complement numerical data.

This project introduces a "hybrid model" that integrates numerical and textual data for advanced stock price prediction forecasting. The numerical component utilizes historical stock prices and market trends to analyze quantitative patterns. The textual component employs sentiment analysis on financial news and social media data to gauge market sentiment and its impact on stock prices. By combining these two dimensions, the hybrid model aims to provide a more comprehensive and accurate prediction mechanism compared to traditional methods.

The significance of this study lies in its potential to bridge the gap between quantitative and qualitative factors in stock forecasting. The hybrid approach can empower investors, traders, and financial institutions to make informed decisions based on a more holistic understanding of market behavior. Furthermore, the integration of machine learning algorithms enhances the model's adaptability to complex patterns and changing market conditions, making it a valuable tool in dynamic financial environments.

This report outlines the methodologies used in developing the hybrid model, including data preprocessing, sentiment analysis, feature engineering, and model training. The subsequent sections also discuss system

design, implementation, and the results of model evaluation. By addressing the limitations of single-dimensional approaches and demonstrating the advantages of hybrid modeling, this project contributes to the growing body of research on intelligent stock forecasting system.

1.1 Objective

This project's objective is to use machine learning and deep learning techniques to create a reliable prediction model that forecasts stock prices. Utilizing past stock market data of 8 companies — Alphabet Inc. (Google) [GOOG], Amazon.com Inc. [AMZN], Apple Inc. [AAPL], Meta Platforms [META], Microsoft Corp. [MSFT], Netflix Inc. [NFLX], Nvidia Corp. [NVDA], and Tata Consultancy Services [TCS] — along with external factors such as financial news, sentiment analysis, and Reddit-based real-time data, the project aims to build a robust forecasting system.

The key objectives include:

1. **Data Collection and Preprocessing**: Gather historical stock market data, real-time Reddit data via bot integration, and financial news, and preprocess the datasets for analysis.
2. **Model Development**: Develop and train machine learning and deep learning models to predict stock prices.
3. **Performance Evaluation**: Assess the accuracy, reliability, and robustness of the predictive models using appropriate evaluation metrics.
4. **User Interface**: Create a user-friendly interface for displaying predictions, sentiment trends, and other relevant insights.
5. **Implementation and Validation**: Implement the system and validate its effectiveness with real-world data and live market scenarios.
6. **Extended Prediction Capability**: Enable users to predict stock prices for **any publicly listed company** for the next **15 days into the future**, using the trained models and real-time sentiment inputs.
7. **Reddit Bot Integration**: Incorporate a Reddit bot that fetches live discussions, sentiments, and opinions from relevant subreddits to enhance real-time analysis and prediction accuracy.

This project seeks to bridge the gap between theoretical research and practical application, providing a useful tool for financial forecasting and decision-making.

CHAPTER 2

SYSTEM ANALYSIS

2 SYSTEM ANALYSIS

2.1 Identification of Need :

In the fast-paced world of finance, having accurate and timely information can make a significant difference in investment decisions. The need for a reliable stock market prediction system arises from the following factors:

- **Market Volatility:** The stock market is inherently volatile and influenced by a multitude of factors such as economic indicators, geopolitical events, and company performance. Predicting market trends can help mitigate risks and enhance investment strategies.
- **Informed Decision-Making:** Investors, traders, and financial analysts require tools that can provide data-driven insights to make informed decisions. A predictive model can serve as a valuable asset in analyzing market trends and forecasting future movements.
- **Competitive Advantage:** In the highly competitive financial market, having an edge through advanced analytics can lead to better investment outcomes and increased profitability.
- **Automation:** Traditional methods of market analysis can be time-consuming and prone to human error. An automated prediction system can streamline the process, ensuring accuracy and efficiency.

2.2 Preliminary Investigation:

Before diving into the development of the stock market prediction system, a thorough preliminary investigation was conducted to understand the scope and requirements of the project. This involved:

2.2.1 Literature Review: An extensive review of existing research and methodologies in the field of stock market prediction was carried out. This helped in identifying the strengths and limitations of various approaches.

In the study [1], the research focuses on developing advanced deep learning models for live stock price predictions, specifically introducing the Fast RNN model, which demonstrates a low root mean squared error (RMSE) of 0.02068 and a computation time of 3.35 seconds. Additionally, a hybrid model combining Fast RNN, CNN, and Bi-LSTM is proposed, achieving an RMSE of 0.02181 and a computation time of 18.18 seconds. These proposed models significantly outperform traditional forecasting methods such as

ARIMA and FBProphet, which have higher RMSE values of 0.796 and 0.935, respectively, showcasing their effectiveness for real-time stock predictions.

The study[2] evaluates the performance of the MVL-SVM model, which integrates multi-view learning with support vector machines (SVM) for stock price prediction. The model achieves an impressive accuracy of nearly 88%, significantly outperforming other baseline models, including ARIMA and traditional SVM approaches, which show lower accuracies around 70%. Additionally, the MVL-SVM model based on news and daily returns demonstrates superior prediction capabilities, with average and median accuracies consistently above 0.8767 across various time frames, indicating its effectiveness in leveraging financial news and market data for improved trading strategies.

The research [3] presents a hybrid stock price prediction model that combines Prediction Rule Ensembles (PRE) and Deep Neural Network (DNN) techniques, authored by Srivinay, B.C. Manujakshi, M.G. Kabadi, and N. Naik, and published on April 20, 2022. The model utilizes moving average indicators over 20, 50, and 200 days to identify stock trends and demonstrates improved prediction accuracy, achieving a Root Mean Square Error (RMSE) of 5.60 for ICICI Bank and 6.30 for SBI Bank, among others. Overall, the proposed hybrid model's RMSE scores are 5% to 7% lower than those of existing DNN and ANN models, indicating its effectiveness in stock price forecasting.

The study[4] explores the use of deep learning techniques to predict stock prices by integrating numerical data (historical stock prices) and textual data (news articles). The research emphasizes the importance of combining these data types to improve prediction accuracy, although it does not provide specific accuracy metrics or detailed drawbacks.

Int the study [5] the authors present an ensemble model that combines deep learning methods to utilize both numerical stock prices and news articles for predictions. The model achieves an accuracy of 85.5%, but it requires significant computational resources and large datasets for training. The complexity and resource demands are noted as potential drawbacks.

This study[6] proposes an ensemble approach, integrating Transformer models, ARIMA, and Linear Regression to predict stock prices. It highlights the potential of combining these diverse methods for better accuracy but does not specify the exact accuracy achieved. The complexity and interpretability challenges of the ensemble model are mentioned as drawbacks.

This research[7] integrates sentiment analysis of news articles with technical analysis of historical stock prices using deep learning models to predict stock market movements. The study suggests improved prediction capabilities but does not provide specific accuracy figures. The sensitivity to the quality of textual data is considered a limitation.

It [8] focuses on employing NLP techniques to analyze financial news articles for predicting stock market trends. While it emphasizes the relevance of incorporating textual data, it does not specify accuracy metrics. The quality and timeliness of news data are critical factors affecting the model's performance.
This [9] introduces a deep neural generative model that uses news articles to forecast stock prices. The paper highlights the potential of generative models in this context but does not provide detailed accuracy results. Preprocessing and handling ambiguous news content are noted as challenges.
The research[10] explores the use of NLP and deep learning to predict stock prices based on financial news. It emphasizes the combination of textual and numerical data for enhanced predictions but lacks specific accuracy metrics. The specialized nature of the financial news corpus limits the model's broader applicability.

2.2.2 Technology Exploration: Exploration of different technologies and tools available for data analysis and machine learning was done to determine the most suitable ones for the project. This included evaluating programming languages, libraries, and frameworks.

We used following libraries and frameworks for the project :

1. **Python:** Core programming language used for data analysis, model building, and backend development.
2. **GitHub:** Platform for version control and collaborative development.
3. **InfluxDB:** Database for efficient time-series data storage and retrieval.
4. **Grafana:** Tool for real-time data visualization and dashboard creation.
5. **Streamlit:** Framework for creating interactive web applications.
6. **Flask:** Lightweight framework for developing the project’s backend.
7. **Pandas:** Library for data manipulation and analysis.
8. **Matplotlib & Plotly:** Libraries for data visualization and graphical representation.
9. **NLP Libraries (NLTK, spaCy):** Tools for processing and analyzing text data.
10. **Machine Learning Libraries:** Used for implementing models like Naive Bayes, MLP, Logistic Regression, and Random Forest.
11. **Yahoo Finance Integration**: Used for fetching real-time stock market data and financial indicators to support up-to-date predictions.

2.2.3 Data Collection: A detailed analysis of available data sources was performed to identify reliable and comprehensive datasets for model training and validation. Historical stock market data was collected from reputable sources to ensure accuracy.

1. For historical stock prices we took the data from Yahoo Finance and textual and hybrid data was made available through Kaggle Dataset - News and Stock Prices.
2. The complete historical data of 8 companies was made available through Macrotrends website.

Stakeholder Consultation: Engaging with potential users and stakeholders to gather insights into their requirements and expectations from the predictive system. This feedback was instrumental in shaping the project's objectives and functionalities.

CHAPTER 3

FEASIBILITY STUDY

3 FEASIBILITY STUDY

3.1. Technical Feasibility:

Assessment of Technologies:

I. **Programming Languages and Libraries**: The project primarily utilizes Python, which is equipped with robust libraries such as pandas, NumPy and scikit-learn essential for data analysis and machine learning.

II. **Data Integration**: Ensuring seamless integration with existing financial databases, the system can fetch real-time data, enhancing the accuracy of predictions.

Scalability:

A. Horizontal Scaling: The system's architecture supports horizontal scaling, allowing for the addition of more servers as data volume increases.

B. Model Optimization: models are optimized for performance, ensuring quick response times even as the dataset grows.

Compatibility:

a) Cross-Platform Compatibility: The solution works across various operating systems (Windows, macOS, Linux), ensuring wide accessibility.

b) Interoperability: The system is designed to integrate with various financial software and tools, ensuring it fits smoothly into existing workflows.

3.2. Operational Feasibility:

User Training:

a) Intuitive User Interface: The system is designed with a user-friendly interface that requires minimal training, enabling users to navigate easily and utilize the tool effectively.

b) Comprehensive Training Materials: Providing detailed user manuals, video tutorials, and interactive guides helps users understand the system's functionalities and maximize its potential.

Maintenance:

I. Regular Updates: The system is designed for easy maintenance and regular updates. This includes updating machine learning models with new data to maintain prediction accuracy.

Support:

A. Technical Support: A dedicated technical support team is available to assist users with any difficulties or queries. This includes a helpdesk, live chat support, and a ticketing system for efficient issue resolution.

B. Documentation: Comprehensive and continuously updated documentation ensures users always have access to the latest information, feature updates, and best practices.

By addressing these critical aspects, the feasibility study ensures the stock market prediction system is both technically and operationally viable, setting a strong foundation for successful implementation and long-term sustainability.

CHAPTER 4

ANALYSIS

4 ANALAYSIS

The analysis phase of the project is critical for understanding the flow of data and the interaction of different components in the system. This section describes the logical and structural framework for the Advanced Stock Price Prediction System using a hybrid model of numerical and textual analysis. The aim is to predict the closing stock prices of seven major technology companies based on historical stock data, such as open, high, low, and volume. This analysis leverages Scikit-learn for regression modeling.

4.1 DFD

The Data Flow Diagram (DFD) is used to visualize the system processes, data flow, and interaction with external entities. Two levels of DFDs are presented: Level 0 (context diagram) and Level 1 (decomposition diagram).

4.1.1 DFD Level 0: Context Diagram

Description:

The Level 0 DFD provides a high-level view of the system, representing it as a single process. It highlights the interaction with external entities and data flow.

Components:

i) External Entities:

Stock Data Source: Represents the source of historical stock data, such as open, high, low, and volume for seven technology companies.

User: The user interacts with the system to view the predicted close prices.

ii) Process:

Stock Price Prediction System: The core process that receives stock data, processes it, and predicts the close price.

iii) Data Stores:

Stock Data Store: Stores the historical stock data needed for processing.

Prediction Data Store: Stores the predicted close prices for user access.

iv) Data Flows:

Input: Data fields (open, high, low, volume) flow from the Stock Data Source to the system.

Output: Predicted close prices flow from the system to the User.

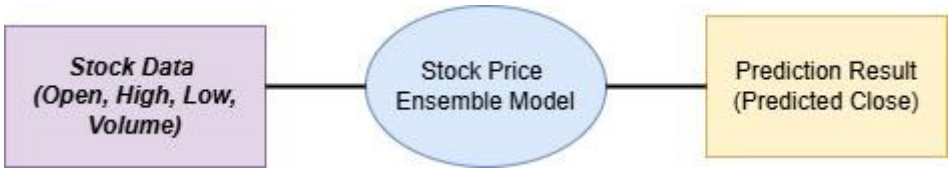


Figure 4.1 DFD Level 0 diagram

4.1.2 DFD Level 1: Decomposition Diagram

Description:

The Level 1 DFD provides a detailed view of the core system by breaking it down into four sub-processes. It captures the flow of data between these processes and the corresponding data stores.

Sub-Processes:

- Data Collection: Collects historical stock data (open, high, low, volume) from the stock data source. Stores the collected data in the Stock Data Store.
- Data Preprocessing: Cleans and preprocesses the collected data by handling missing values, scaling features, and splitting data into training and testing sets. Outputs preprocessed data for further analysis.
- Regression Model: Applies a regression model (using Scikit-learn) to predict the close price based on the preprocessed data. Outputs predicted close prices to the Prediction Data Store.
- Store Predictions: Saves the predicted close prices in the Prediction Data Store for visualization and analysis by the user.
- Data Stores: Stock Data Store: Stores input data fields (open, high, low, volume).
- Prediction Data Store: Stores predicted close prices for future use or visualization.
- Data Flows:
  - Data flows from the Stock Data Store to Data Preprocessing.
  - Preprocessed data flows into the Regression Model.
  - Predicted results flow from the Regression Model to the Prediction Data Store.
  - Users retrieve predictions from the Prediction Data Store.

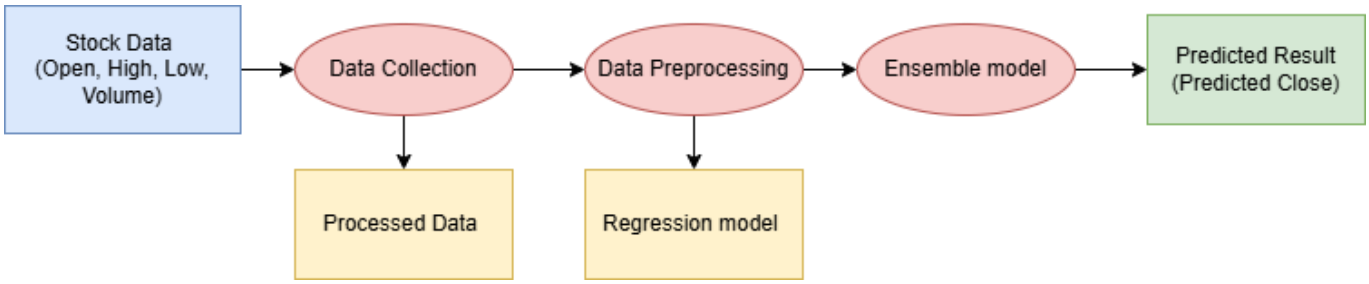


Figure 4.2 DFD Level 1 diagram

4.2 ER Diagram

The **Entity-Relationship (ER) Diagram** is a conceptual modeling tool that visually represents the data structure and relationships within the system. In the project **"Advanced Stock Price Prediction Using a Hybrid Model of Numerical and Textual Analysis,"** the ER diagram captures how stock data, user interactions, machine learning predictions, and external data sources are organized and interconnected. It defines entities such as **Stock Data**, **User Inputs**, **Prediction Outputs**, and **Sentiment Scores**, along with their attributes like open, high, low, close prices, volume, sentiment value, prediction dates, and stock tickers.

The system architecture integrates several important components which are also reflected in the ER diagram. The **Streamlit Application** serves as the main user interface, where users can input stock symbols and view future price predictions, sentiment analysis results, and interactive graphs. Real-time stock market data is fetched using **Yahoo Finance API Integration**, while real-time financial sentiment is collected by the **Reddit Bot** that scrapes live data from relevant subreddits. Both numerical and textual datasets are pushed into **InfluxDB**, a high-performance time-series database. Prediction models, including machine learning techniques such as **Random Forest**, **MLP**, **Naive Bayes**, and **Logistic Regression**, access preprocessed data stored in **InfluxDB** to generate 15-day future price forecasts for any publicly listed company. The **GitHub Repository** maintains the entire project's source code, including model scripts, data fetching scripts, bot programs, and UI code, enabling collaborative development and version control.

The ER diagram further shows how visualization and monitoring are handled through the **Grafana Dashboard**, which pulls data directly from **InfluxDB** to create real-time charts of stock trends, sentiment trends, and system performance. Predictions generated by the models are stored in a **Prediction Data Store** and made accessible to users via the **Streamlit App**. The interaction between components — from data fetching by the **Yahoo Finance API** and **Reddit Bot**, to model prediction, storage in **InfluxDB**, visualization through **Grafana**, and user interaction through **Streamlit** — forms a complete, integrated loop. Together, the ER diagram highlights how each

subsystem collaborates to deliver a seamless, real-time stock prediction experience powered by a hybrid of numerical and textual analysis.

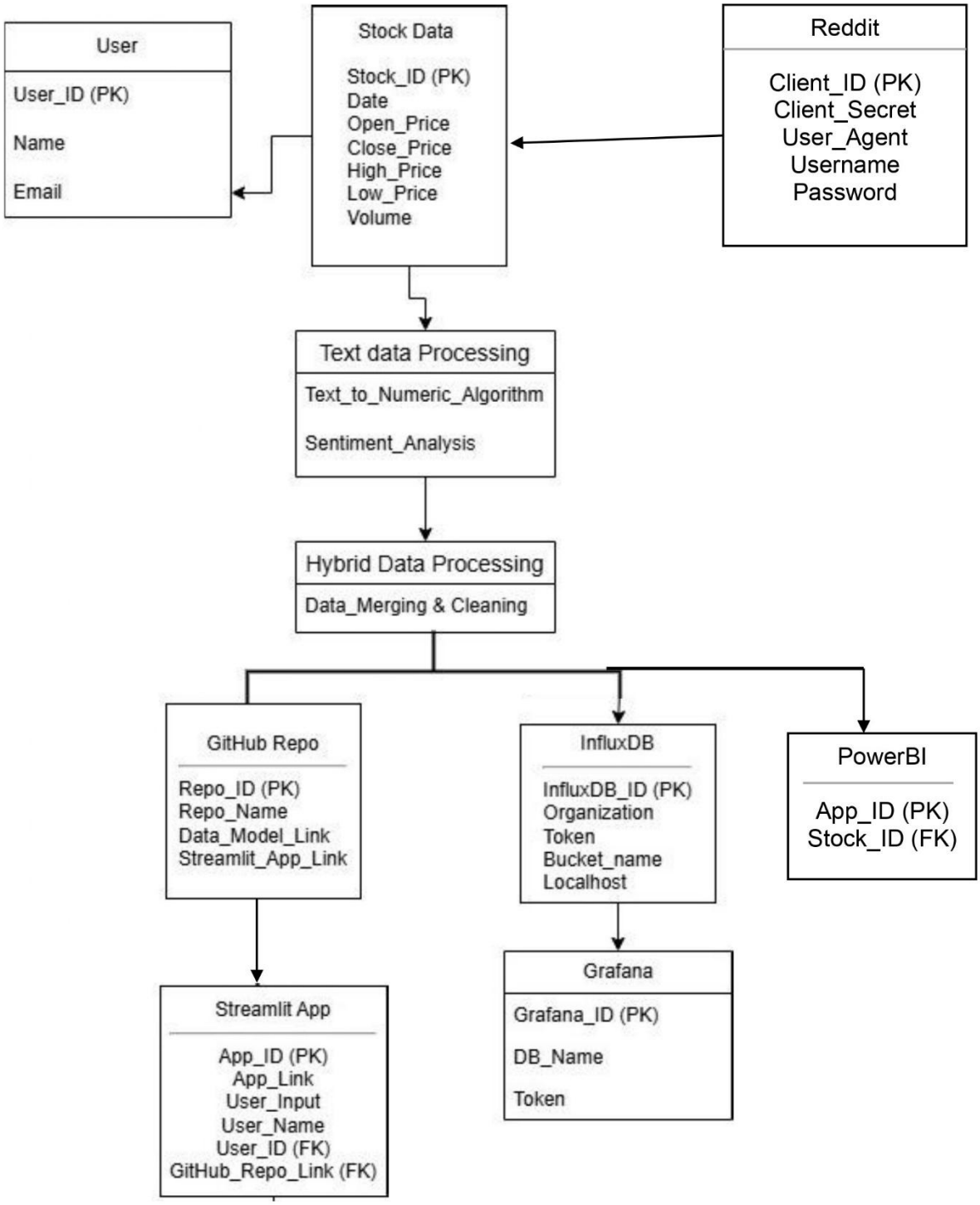


Figure 4.3 Entity Relationship Diagram

CHAPTER 5

METHODOLOGY

5 METHODOLOGY

The methodology section provides a detailed description of the approach used to achieve the project's objectives. It outlines the project's structure, the techniques applied for analysis, and the processes employed to ensure accurate predictions of stock prices using a hybrid model of numerical and textual analysis.

5.1. Project Overview

The Advanced Stock Price Forecasting Using a Hybrid Model of Numerical and Textual Analysis project involves a comprehensive approach to predicting stock prices using both numerical data and textual analysis. The project components include:

1. **Data Collection and Storage**: We gathered historical stock data of major companies and stored it in an InfluxDB database to efficiently handle large-scale time-series data.
2. **Data Visualization:** A Grafana dashboard has been set up for real-time visualization of stock prices and analysis results, enhancing data interpretation and decision-making processes.
3. **Textual Analysis for Enhanced Forecasting:** We utilized Natural Language Processing (NLP) libraries, such as NLTK and spaCy, to analyze financial news and reports. This component complements numerical analysis to improve the accuracy of our hybrid forecasting model.
4. **Machine Learning Models:** The project used models including Naive Bayes, MLP (Multi-Layer Perceptron), Logistic Regression, and Random Forest to process both numerical and textual data, creating a robust and comprehensive stock prediction system.
5. **Collaboration and Project Management:** The repository includes contributions from all team members with well-organized tasks, ensuring seamless collaboration and effective version control.

5.2. Data Collection:

Historical stock data is sourced from reputable financial databases such as **Yahoo Finance, Kaggle and macrotrends.** These platforms provide extensive data on stock prices, trading volumes, and other relevant financial metrics.

Table 5.1 Companies dataset summary

Company	Data Range	Dataset Shape	Starting Stock Date	Current Stock Date	Starting Stock Price	Current Stock Price
Alphabet Inc. (Google) [GOOG]	2014-03-27 : 2024-10-17	(2659, 5)	2014-03-27	2024-10-17	$27.8542	$164.51
Amazon.com Inc. [AMZN]	1997-05-16 : 2024-10-17	(6901, 5)	1997-05-16	2024-10-17	$0.0863	$187.53
Apple Inc. [AAPL]	1980-12-12 : 2024-10-17	(11053, 5)	1980-12-12	2024-10-17	$0.0992	$232.15
Meta Platforms [META]	2012-05-18 : 2024-10-17	(3124, 5)	2012-05-18	2024-10-17	$38.1174	$576.93
Microsoft Corp. [MSFT]	1986-03-13 : 2024-10-17	(9728, 5)	1986-03-13	2024-10-17	$0.0603	$416.72
Netflix Inc. [NFLX]	2002-05-23 : 2024-10-17	(5640, 5)	2002-05-23	2024-10-17	$1.1964	$687.65
Nvidia Corp. [NVDA]	1999-01-22 : 2024-10-17	(6477, 5)	1999-01-22	2024-10-17	$0.0377	$136.93
Tata Consultancy Services [TCS]	2013-11-01 : 2024-10-17	(2758, 5)	2013-11-01	2024-10-17	$543.0	$11.8

5.3 Dataset Description

This dataset contains historical stock market data for 8 companies: **GOOG, AMZN, AAPL, META, MSFT, NFLX, NVDA, TCS**. The dataset provides crucial financial metrics, including daily **Open, High, Low, Close, Adjusted Close, Volume,** and **Date** for each stock. Our target is to build a predictive model to forecast the Closing Price of a stock based on the Open, High, and Low prices, along with the specific company data.

Dataset Fields: This section provides an in-depth explanation of each field in the stock market dataset. Understanding these fields is crucial for effectively analyzing and modeling stock price movements.

The goal is to predict the Closing Price based on the Open, High, and Low prices. Additionally, company data is used as an input feature to improve the model's learning capability. The dataset combines stock data from companies like GOOG, AMZN, AAPL, META, MSFT, NFLX, NVDA, TCS enabling the model to generalize across diverse data sources.

Table 5.2 Overview of Key Fields and their Importance in Stock Market Predictions

Field	Definition	Details	Usage/Significance
Date	The specific calendar day when the trading activity occurred.	**Format**: Typically, `YYYY-MM-DD` (e.g., 2024-04-27). **Importance**: Tracks stock performance over time and identifies trends, seasonal patterns, and cyclical behaviors. **Usage**: Time-based aggregations like daily, weekly, or monthly analysis.	Essential for aligning data from different sources and performing time-series analysis.

Open	The price at which a stock starts trading when the market opens for the day.	**Determination**: First trade executed during the trading session. **Influencing Factors**: Overnight news, pre-market sentiment, and global events. **Importance**: Serves as a reference for the day's activity and identifies opening trends in technical analysis.	Indicates how the market opens, providing context for the day's price movements.
High	The highest price at which the stock traded during the trading day.	**Measurement**: Recorded as the peak price reached from open to close.- **Significance**: Reflects maximum buying interest. **Usage**: Identifying resistance levels and comparing momentum with previous highs.	Assesses price volatility and provides insights into market optimism.
Low	The lowest price at which the stock traded during the trading day.	**Measurement**: Lowest price reached from open to close. **Significance**: Reflects maximum selling pressure. **Usage**: Identifying support levels and gauging bearish momentum.	Helps define the trading day's price range and identifies points of significant selling activity.
Close	The final price at which the stock trades when the market closes for the day.	**Determination**: Based on the last executed trade of the day.- **Importance**: Used as a benchmark for performance evaluation. **Usage**: Calculating daily returns and forming the basis for indicators like moving averages.	Widely regarded as the most critical price for performance reporting and analysis.

Volume	The total number of shares traded during the trading day.	**Measurement**: Sum of shares bought and sold. **Significance**: Reflects activity and liquidity. **Usage**: High volume confirms trends or identifies potential reversals/breakouts.	Indicates market interest and can highlight significant price movements.

This table summarizes the information systematically for easier understanding and reference.

5. 4 Data Analysis

Data analysis is a crucial step in the development of a stock market prediction model. It involves examining and processing the collected data to uncover meaningful patterns, trends, and relationships. Below provides a comprehensive overview of the data analysis process for the stock market prediction project, detailing each step from data cleaning to feature selection and visualization.

5.4.1 Data Preprocessing For Hybrid Model:

1) **Fill NaNs in 'sentiment' (neutral sentiment as 0)**: Missing values in the sentiment column are filled with 0 to represent neutral sentiment. This ensures that the 'sentiment' feature has no missing data, which could disrupt model training.

2) **Forward-fill other NaN values for stock prices**: Missing data in stock price columns ('voo_close', 'qqq_close', and 'dia_close') is filled using a forward-fill method. This technique fills NaNs with the last available value, maintaining continuity in time-series data. A backward-fill is applied afterward to handle any leading NaNs.

3) **Define features (X) and target (y)**: The feature set (X) includes 'sentiment', 'voo_close', 'qqq_close', and 'dia_close'. The target variable (y) is set to 'voo_close', assuming we want to predict this stock's closing price.

4) **Split data into training and testing sets**: The data is split into training and testing sets using an 80-20 split, with shuffle=False to maintain the order of time-series data.

5) **Impute remaining NaNs in training and testing sets**: A SimpleImputer with a mean strategy is used to fill any remaining NaNs in the features. This ensures no missing values remain, which could hinder the performance of machine learning algorithms.

6) **Check for NaNs in the imputed data**: The np.isnan().sum() function checks for the presence of NaNs in X_train and X_test after imputation. This step verifies that imputation was successful.

7) **Scale the features**: A StandardScaler is applied to standardize the feature values, ensuring that all features have a mean of 0 and a standard deviation of 1. This helps models that are sensitive to feature scales perform better.

8) **Final check for NaNs in scaled data**: Another check is done to confirm that no NaNs are present in the scaled training (X_train_scaled) and testing (X_test_scaled) datasets. This guarantees the data is ready for model training.

This pipeline prepares the dataset by handling missing values, ensuring continuity, and scaling features for optimal model performance.

5.5 For Imputing Missing Values in Scaled Data:

I. **Create a SimpleImputer to replace NaNs with the mean**: An instance of SimpleImputer is created with the strategy set to 'mean'. This imputer will replace any missing values (NaNs) in the dataset with the mean of the respective feature column, ensuring there are no gaps in the data.

II. **Fit and transform the imputer on the scaled training data**: The imputer is fitted to X_train_scaled and then used to transform it. This process computes the mean of each feature in X_train_scaled and replaces any missing values with the corresponding mean. This ensures that the training data has no missing values, which is essential for reliable model training.

III. **Transform the scaled testing data using the fitted imputer**: The already fitted imputer is applied to X_test_scaled to replace any NaNs using the mean values computed from

X_train_scaled. This maintains consistency in how missing values are handled across both training and testing datasets.

5.6 Exploratory Data Analysis :

Libraries Used :

a) **NumPy:** A fundamental package for numerical computations in Python, providing support for arrays and matrices, along with a collection of mathematical functions.

b) **Pandas:** A powerful data manipulation and analysis library that offers data structures like DataFrames for handling structured data efficiently.

c) **Matplotlib:** A widely used plotting library that provides a variety of functions for creating static, animated, and interactive visualizations in Python.

d) **Seaborn:** A data visualization library based on Matplotlib that provides a high-level interface for drawing attractive statistical graphics.

e) **Plotly:** An interactive graphing library that allows for the creation of web-based visualizations with built-in interactivity.

f) **mplfinance**: A library specifically designed for creating candlestick charts and other financial plots using Matplotlib.

g) **warnings:** A built-in Python module used to control the display of warning messages.

h) **os:** A built-in module that provides a way of using operating system-dependent functionality like reading or writing to the file system.

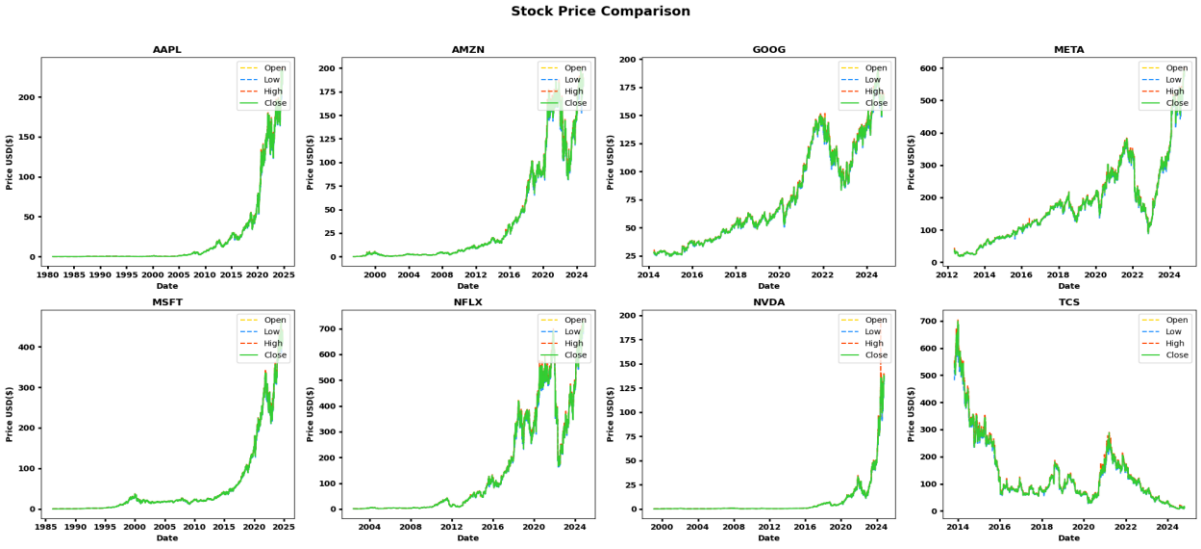


Figure 5.1 Stock Price Comparison of all 8 companies

The plots in Figure 5.1 illustrate the opening, high, low, and closing prices for all companies. By including these key price points alongside the original closing prices, the visualization provides a comprehensive comparison of price fluctuations and highlights the range of trading activity for each company over time.

Its primary goal is to preprocess these historical datasets, clean the data (e.g., handling missing values, dealing with outliers), and perform an exploratory data analysis (EDA). The goal of EDA is to summarize the main characteristics of the dataset and uncover underlying patterns, trends, and relationships. Visualizations such as time series plots, histograms, and correlation heatmaps will help to assess the distribution of prices, trading volumes, and the relationships between stock prices and volume.

Thus, the initial data loading phase involves reading CSV files for Google and Apple. For Google, the dataset contains columns such as date, open, high, low, close, and volume, with data starting from March 2014. For Apple, the dataset spans a longer timeframe, starting from December 1980, and shares a similar structure. Key insights, such as Apple's starting price of $0.0992 in 1980 and its current price of $232.15 in October 2024, demonstrate the significant growth of these companies over time.

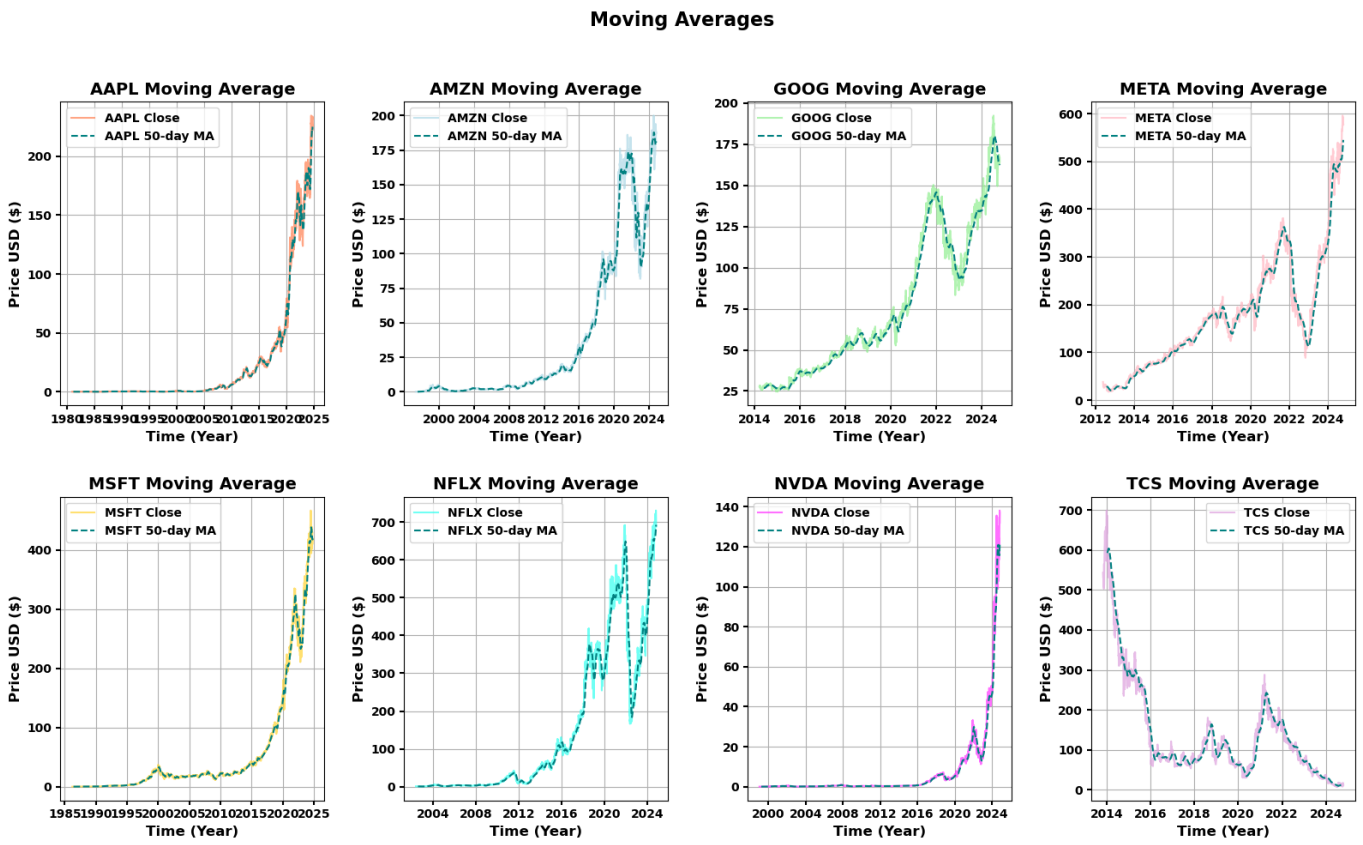


Figure 5.2 Close Price Moving Average Comparison of all 8 companies

The plots in Figure 5.2 depict the moving averages of all companies, overlaid with their original closing prices. This approach offers a clearer perspective on price trends by smoothing short-term fluctuations, enabling better insights into the overall performance and market behavior of each company.

The EDA serves as a foundational step for building models that can predict stock price movements, identify market trends, and make informed trading decisions. The preprocessing and EDA ensure that the data is in a clean and usable format for future analysis, such as forecasting stock prices using machine learning models or conducting more advanced financial analysis.

These datasets are uniformly structured, featuring price data (open, high, low, close) and trading volume, making them suitable for comparative and trend analysis. It sets the stage for subsequent steps, which include cleaning the data, performing exploratory data analysis (EDA), and deriving meaningful insights through statistical and visual techniques.

Thus , the EDA serves as a foundational step for building models that can predict stock price movements, identify market trends, and make informed trading decisions. The preprocessing and EDA steps outlined in the notebook ensure that the data is in a clean and usable format for future analysis, such as forecasting stock prices using machine learning models or conducting more advanced financial analysis.

5.7 Codebase Modularization

Codebase modularization involves breaking down a monolithic code structure into smaller, manageable, and independent modules. This process enhances the maintainability, scalability, and reusability of the code by organizing it into distinct functional units. In this specific case, the system undergoes several steps to ensure that all functions are properly separated, dependencies are injected, and the overall codebase is prepared for deployment.

The modularization process begins by splitting the source code into individual functions, each stored in separate files for easier management. Dependencies are then automatically added to ensure each function can operate independently. A combined import file is generated for easier access to all functions. The core application logic is cleaned up by removing function definitions, ensuring that only the main logic remains. The StockPricePredictor class is manually added to the system, integrated into both local and deployed environments. Dataset paths are validated, ensuring proper access to necessary data. The final step runs the modularized, fully-integrated Streamlit app, which is now ready for deployment and future maintenance. The key benefits of this approach include simplified debugging, ease of testing, and the ability to easily extend the codebase in the future.

Processes:

- **Extract Functions (Function_Splitting.py):**
  - *Purpose:* Parses the original source code, splitting each function into a separate file for better modularization.
  - *Output:* Functions are saved in two locations: features_functions_local/ and features_functions_deployed/ for local and deployed environments respectively.
- **Inject Dependencies (Dependency_Adder.py):**
  - *Purpose:* Analyzes each function file using the Abstract Syntax Tree (AST) method. It identifies and adds necessary import dependencies to each function, ensuring it is self-contained and ready for deployment.
  - *Output:* Function modules are updated with required imports, making each function independent and deployable.
- **Generate Import Aggregator (Import_Functions.py):**
  - *Purpose:* Scans all the function modules and generates a combined import file to centralize the imports for the entire system, simplifying the management of dependencies.
  - *Output:* A unified import file is created, centralizing access to functions across modules.
- **Extract Main Logic (Split_Clean_Main_Code.py):**
  - *Purpose:* Extracts and removes any embedded function definitions from the main code. This step ensures that only the core application logic remains in the cleaned file, separating the business logic from the function definitions.
  - *Output:* The cleaned main logic is saved as app_cleaned.py, preparing the core logic for deployment.
- **Manual Step - Add StockPricePredictor Class:**
  - *Purpose:* The StockPricePredictor class must be manually added to the system. This step involves pasting the class definition into the appropriate file (display_real_time_stock_prediction.py) and ensuring it's integrated in both local and deployed versions.
  - *Output:* The class is successfully integrated into the system, ready for use in the prediction pipeline.
- **Validate Dataset Paths:**
  - *Purpose:* Ensures that the paths for the dataset directory (DATASET_DIR and DATASET_DIR_1) are correct and accessible. This ensures that the data is available for further processing and prediction tasks.
  - *Output:* Validated dataset paths that point to the correct directories for dataset storage and access.

- **Run Final App (app_cleaned.py):**
    - *Purpose:* This final step runs the modularized version of the Streamlit app, which has been cleaned and organized for deployment. All functions, dependencies, imports, and the main logic are now correctly integrated and ready for use in a production environment.
    - *Output:* The fully modular, maintainable, and deployment-ready Streamlit app is ready for execution.
- **Data Stores: • features_functions_local/ and features_functions_deployed/:**
    - Stores the modularized functions, with separate directories for local and deployed environments.
- **Prediction Data Store:**
    - Stores predicted values (e.g., stock prices) for future use or visualization.

Data Flows: Data flows from the original code to the **Extract Functions** sub-process, where it is split into function files.

• The split functions are processed by **Inject Dependencies**, where necessary imports are added, ensuring self-contained modules.

• **Generate Import Aggregator** scans the function files and generates a combined import file.

• Cleaned application logic is extracted by **Extract Main Logic** and saved as app_cleaned.py.

• **Manual Step** requires user interaction to add the StockPricePredictor class and validate dataset paths.

• After validation, the fully prepared app is executed, with modularized functions and dependencies ready for deployment.

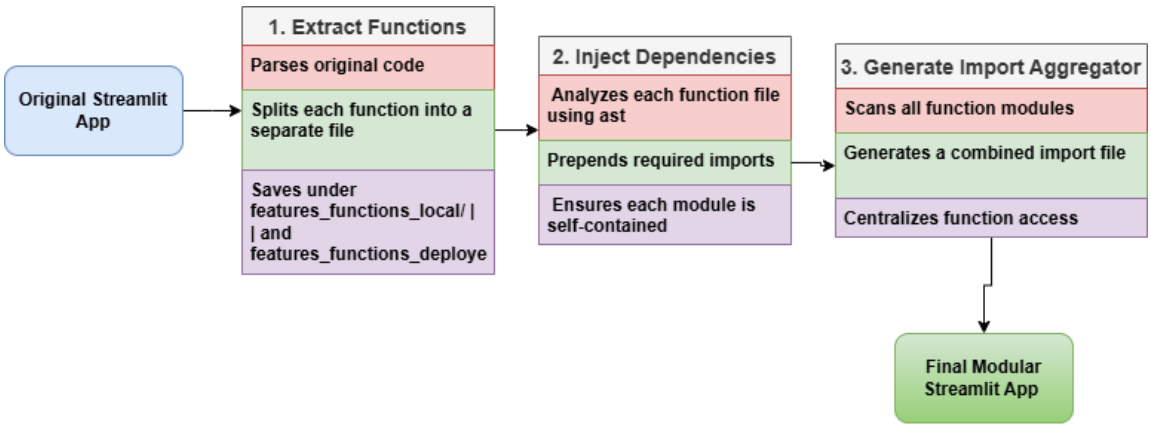


Figure 5.3 Codebase Modularization Flowchart

CHAPTER 6

IMPLEMENTATION

6 Implementation

6.1 Development Environment

Software Tools :

1. **Programming Language:** Python
2. **Data Analysis Libraries:** Pandas and NumPy
3. **Data Visualization Tools:** Matplotlib and Plotly
4. **Machine Learning Framework:** Scikit-learn
5. **Real-Time Data Visualization:** Grafana
6. **Backend Frameworks:** Flask and Streamlit
7. **Text Data Analysis:** NLTK
8. **Interactive Web Applications:** Streamlit
9. **Model Saving Techniques:** Joblib and Pickle

Configuration:

All software tools were utilized in their latest stable versions available at the time of the project. This includes pip (24.0), NumPy (1.26.4), Pandas (2.2.1), Matplotlib (3.6.2), Scikit-learn (1.4.1), Flask (3.0.2), and NLTK (3.8.1).

6.2 Project Execution

The execution of this project involved the implementation of two primary approaches: numerical regression for stock data analysis and textual analysis for sentiment classification of stock-related news. The final hybrid model integrates insights from these approaches for advanced stock price prediction. This section provides a detailed explanation of the steps and techniques used in the project execution.

6.2.1. Numerical Regression Ensemble Model

The numerical regression model focuses on predicting the stock's closing price based on historical numerical data features such as open, high, low, and volume. The following steps were carried out:

a. Data Preparation:

1. The dataset was preprocessed to extract relevant numerical features:

- Features: open, high, low, volume.
- Target: close (value to be predicted).

2. The dataset was split into two parts:
   - **Input Data (X)**: [open, high, low, volume].
   - **Target Data (y)**: [close].

b. Train-Test Split:

The dataset was further divided into training and testing sets:

- Training Set (80%): Used to train the model.
- Testing Set (20%): Used for evaluation to ensure the model generalizes well to unseen data.

c. Regression Models:

Several regression algorithms were tested to evaluate their individual performance before combining them into an ensemble model. The algorithms included:

1. **Linear Regression**:
   - A basic regression model that establishes a linear relationship between input features and the target variable.
   - Equation: $(y = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n)$.
   - Strength: Simple, interpretable.
2. **Ridge Regression**:
   - A variant of linear regression that adds (L2) regularization to prevent overfitting by penalizing large coefficients.
   - Equation: $Minimize(\text{RSS} + \lambda \sum b_i^2), where(\lambda)$is the regularization parameter.
3. **Lasso Regression**:
   - Similar to Ridge Regression but uses (L1) regularization, encouraging sparsity by driving some coefficients to zero.
   - Strength: Effective feature selection.
4. **Elastic Net Regression**:
   - Combines both (L1) and (L2) regularization techniques.

- Suitable for datasets with multicollinearity or when the number of features exceeds the number of observations.

D. Ensemble Model:

After training the individual models, a **Voting Regressor** was used to create an ensemble model. This approach combines the predictions from all models by averaging their outputs, leveraging their strengths to improve overall accuracy.

They are two types: -

1) Hard Vote: Each model predicts a class label. The final prediction is class label that receives most votes from individual models.

2) Soft Vote: Each model predicts a probability for each label. The final prediction is class label with highest sum of probabilities across all models.

Advantages

Improve Accuracy: Combining predictions of multiple models, voting ensemble can often achieve higher accuracy than any single model.

Reduced Variance: Different models produce certain errors and averaging their predictions help the overall variance result.

Disadvantage

Increase complexity: Training and managing multiple models is more complex.

e. Model Saving:

The trained ensemble model was saved for future use:

- **Pickle**: Saves the model as a binary file for easy reuse.
- **Joblib**: An efficient alternative for saving models with large datasets

6.2.2. Textual Analysis

Textual analysis was performed on a dataset containing stock-related news articles to classify their impact (up or down) and extract insights for stock prediction.

a. Data Preparation:

1. A text dataset was collected and preprocessed.
2. A classification column (up and down) was added to label the sentiment or potential impact of the news articles on stock prices.

b. Exploratory Data Analysis:

1. **Distribution Plot**:
    - A bar chart was plotted to show the distribution of up and down classifications, providing an understanding of the data balance.

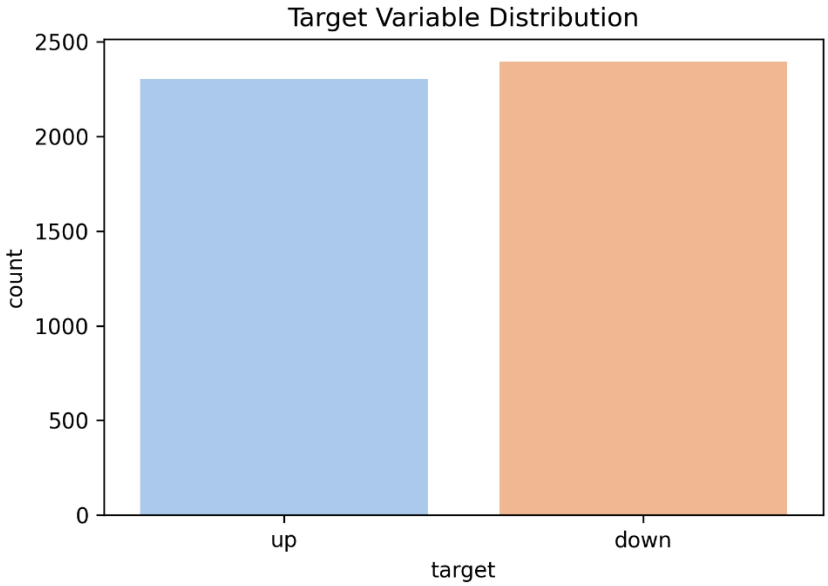


Figure 6.1 Target Variable Distribution

2. **Word Cloud**:
    - A word cloud was generated to visualize the most frequently occurring words in the dataset, highlighting keywords relevant to stock predictions.

Figure 6.2 Word Cloud of News Articles

3. **Top 20 Words**:
    - The top 20 most common words in the news articles were identified and visualized, giving insights into recurring terms.

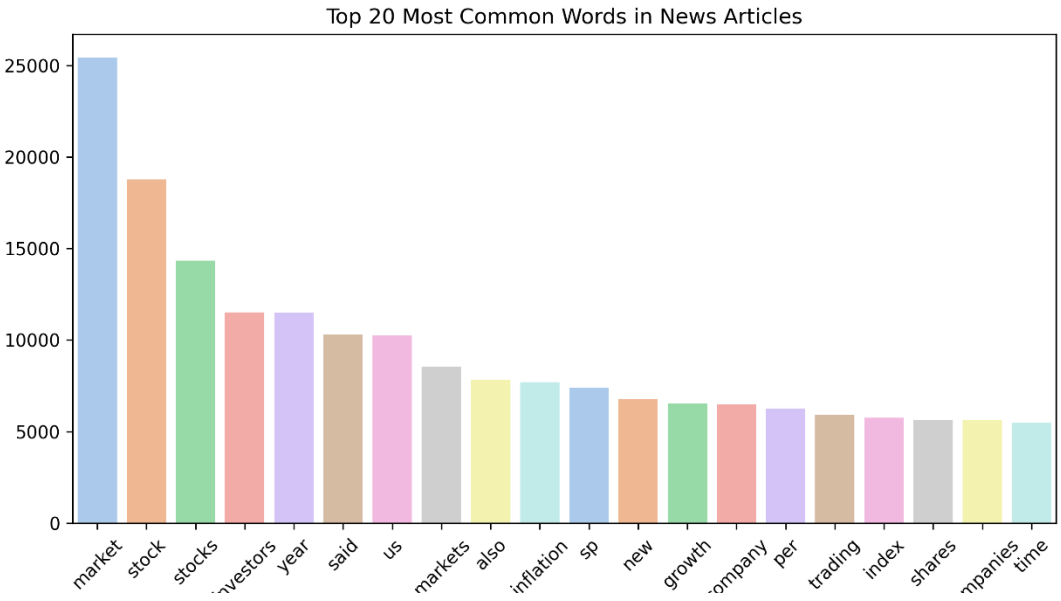


Figure 6.3 Top 20 Words of News Articles

c. Text Transformation:

The text data was converted into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)** vectorization. This method captures the importance of words in the corpus by reducing the weight of frequently occurring but less informative words.

d. Classification Models:

Several classification models were applied to predict the up and down labels:

1. **Logistic Regression**:
    a. A probabilistic model used for binary classification.
    b. Suitable for understanding the impact of individual terms on classification.
2. **Naive Bayes**:
    a. Based on Bayes' theorem, this model assumes feature independence and is efficient for text classification tasks.
3. **Random Forest**:

a. An ensemble learning technique that creates multiple decision trees and averages their outputs.

b. Robust to noise and effective for capturing complex patterns.

e. Model Evaluation:

The performance of each classification model was evaluated using:

- **ROC Curve**:
  - Visualizes the trade-off between true positive rates and false positive rates.
  - The area under the curve (AUC) indicates the model's effectiveness.
- **Confusion Matrix**:
  - Provides detailed insights into true positives, true negatives, false positives, and false negatives.

6.2.3. Hybrid Model

The hybrid model combines the predictions from the numerical regression ensemble and textual analysis to improve the accuracy of stock price forecasting.

a. Integration:

1. Numerical predictions (close price) and textual sentiment classifications (up or down) were integrated.
2. Textual sentiment was used as an additional feature to adjust the final prediction of the stock's closing price.

b. Weighted Ensemble:

- A weighted approach was used to assign importance to numerical and textual predictions based on their individual performances.

c. **Evaluation**:

The hybrid model was evaluated against a test dataset to ensure that the integration of numerical and textual components improved the accuracy and reliability of the predictions.

6.2.4 Reddit Chatbot

Reddit data for the latest financial news and sentiments was fetched using the **Reddit API**. To establish a secure connection and access real-time posts from relevant subreddits, essential credentials were utilized, including the **Client_ID** (Primary Key), **Client_Secret**, **User_Agent**, **Username**, and **Password**. These credentials enabled authenticated API requests, allowing the system to retrieve and analyze up-to-date textual data for enhancing stock price predictions.

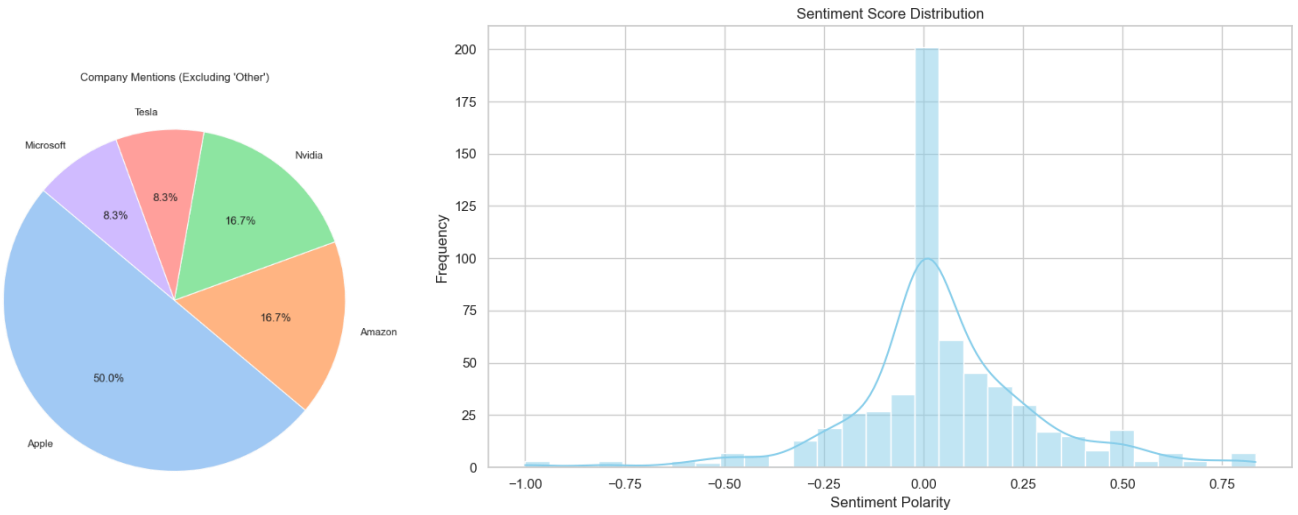


Figure 6.4 Top Companies in Reddit Posts and Sentiment Scores

Figure 6.5 Word Cloud of Reddit Posts

6.2.5 PowerBI Dashboard

A **Power BI dashboard** was developed to provide interactive visualizations of stock performance, prediction trends, and sentiment analysis. The dashboard's data model includes an **App_ID** as the primary key, uniquely identifying each application instance, and a **Stock_ID** as a foreign key, linking stock data to its respective application. This relational structure enables seamless integration and dynamic reporting, allowing users to explore and analyze stock predictions and sentiments effectively.

6.2.6 Real Time Prediction

To enhance flexibility, **custom script support** was implemented, allowing users to input any stock ticker (like TCS, TSLA, etc.) and receive predictions from four models — **Linear Regression**, **Random Forest**, **SVM**, and **LSTM** — using **yfinance** integration. Users can now access real-time forecasts through the Streamlit app's updated "Real-Time Prediction" section, and download results cleanly with new, separate buttons for **Real-Time Predictions** and **Model Metrics**, improving usability. Additionally, lightweight **HTML-based download links** with **Base64 encoding** replaced the native st.download_button(), avoiding full-page rerenders and providing smoother downloads for prediction and metric CSV files, each managed via custom methods save_data_csv() and export_results() stored in st.session_state.

6.3 Challenges Faced

During the development of the project **"Advanced Stock Price Prediction Forecasting using a Hybrid Model of Numerical and Textual Analysis",** several challenges were encountered. These challenges required careful consideration and innovative solutions to ensure the accuracy and reliability of the final model.

6.3.1. Difficulty in Aligning News Data with Stock Data

One of the most significant challenges was aligning news data with corresponding stock data. Stock price movements are often influenced by events and news that may not immediately correspond to the same day or even the same week as the reported news. This introduced complexities, such as:

1) **Time Lag:**

- The impact of a news article on stock prices may not be immediate. It was challenging to determine the exact time frame in which a particular piece of news influenced the stock price.

2) **Non-relevance:**

- Not all news articles labeled under a specific company directly impacted its stock price. Irrelevant or less impactful articles diluted the prediction model's performance.

Solution Considered:

To mitigate this, sentiment analysis was limited to only those articles explicitly mentioning the companies in question, and a time-window-based approach was tested to correlate news events with stock movements.

6.3.2. Inconsistent and Unstructured News Data

News data often comes in unstructured formats, requiring significant preprocessing. The challenges here included:

1) **Noise in Text Data:**

- Articles contained irrelevant information, such as general market summaries or advertisements, which had to be filtered out.

2) **Duplicate and Overlapping Content:**

- News sources frequently repeated similar content, leading to redundancy and bias in textual analysis.

Solution Considered:

Preprocessing steps were implemented, including:

- Removing duplicates and irrelevant sections of articles.
- Extracting core textual data using Natural Language Processing (NLP) techniques.

6.3.3. Data Imbalance in Sentiment Classification

The news dataset often exhibited an imbalance between the `up` and `down` classifications, with one class dominating the other. This imbalance affected the performance of the classification models, leading to biased predictions.

Solution Considered:

Techniques such as oversampling (e.g., SMOTE) or under sampling the dominant class were explored to address the imbalance and improve model training.

6.3.4. Feature Selection and Model Overfitting

Selecting the most relevant features from the numerical and textual data posed a challenge. Including too many features led to overfitting, where the model performed well on training data but poorly on unseen data.

Solution Considered:

Feature importance was analyzed using regression coefficients for numerical data and TF-IDF scores for textual data. Models were tuned with regularization techniques such as Ridge and Lasso to prevent overfitting.

6.3.5. Integration of Numerical and Textual Data:

Combining numerical predictions with textual sentiment posed technical and logical challenges, such as:

- Determining the relative weight of textual sentiment in influencing stock price predictions.
- Ensuring that both components were aligned temporally and logically to avoid introducing errors.

Solution Considered:

A hybrid model using a weighted ensemble method was implemented, giving numerical and textual predictions appropriate importance based on their individual performance metrics.

CHAPTER 7

RESULT AND DISCUSSION

7 Result and Discussion

The results of the project "Advanced Stock Price Prediction Forecasting using a Hybrid Model of Numerical and Textual Analysis" are presented in this section. The findings are based on extensive exploratory data analysis (EDA), numerical regression for predicting stock closing prices, and sentiment classification from textual data. The performance of the developed models is evaluated, highlighting the effectiveness of the hybrid approach.

7.1 Exploratory Data Analysis (EDA) on Closing Prices

EDA was conducted to understand the underlying trends, patterns, and variability in the closing prices of the seven selected tech companies: AAPL, AMZN, GOOG, META, MSFT, NFLX, NVDA, and TCS.

Trends and Insights:

Companies like NVDA and META exhibited rapid price growth in recent years, reflecting high volatility influenced by technological advancements and market dynamics.

TCS, on the other hand, showed a steady upward trend, suggesting consistent performance and less susceptibility to market shocks.

Statistical Analysis:

Measures such as mean, median, and standard deviation revealed the overall behavior of each stock.

Skewness and kurtosis values indicated whether the stock prices were symmetrically distributed or had heavy tails, helping to guide feature engineering.

7.2 Database and Dashboard

The data was stored in an InfluxDB database and visualized using a Grafana dashboard. Below are some snapshots showcasing the dashboard and its features.

7.2.1 InfluxDB Database

InfluxDB is a specialized time-series database designed to efficiently store and manage time-stamped data. It supports high-performance data ingestion and querying, making it ideal for real-time monitoring and

analytics. Additionally, it offers features such as retention policies, which allow for automatic data lifecycle management by specifying how long data should be stored, ensuring optimized storage and performance.

Figure 7.1 InfluxDB database showing all measurements

As shown in the above snapshot, there are four distinct measurements, each serving a specific purpose:

- **stock_price:** Stores numerical data related to stock prices.
- **model_prediction:** Contains numerical data representing model predictions.
- **textual_analysis:** Holds text-based analytical data.
- **hybrid_model:** Combines numerical and textual data for hybrid analysis.

These measurements are designed to categorize and store data based on its type and usage. Numerical data, such as stock prices and model predictions, enables quantitative analysis and trend visualization. Textual data, processed under textual_analysis, provides insights derived from qualitative information like sentiment or news. The hybrid_model integrates both numerical and textual elements, offering a comprehensive perspective for advanced analysis and decision-making.

7.2.2 Grafana Dashboard

Grafana is a powerful, open-source visualization and analytics tool that plays a crucial role in this dashboard. It facilitates real-time data monitoring, interactive data exploration, and the creation of custom visualizations. Grafana's ability to connect to various data sources, including InfluxDB, enables seamless integration of live data updates, ensuring that users have the most current insights at their fingertips. With

its customizable panels and user-friendly interface, Grafana empowers users to build detailed, interactive dashboards that present data in an intuitive and actionable way.

7.2.2.1 Overview

The **Overview** section acts as an introduction to the dashboard, providing an executive summary and contextual insights into the data. It includes high-level visualizations and descriptive statistics that help users quickly grasp the nature of the data and the objectives of the analysis.

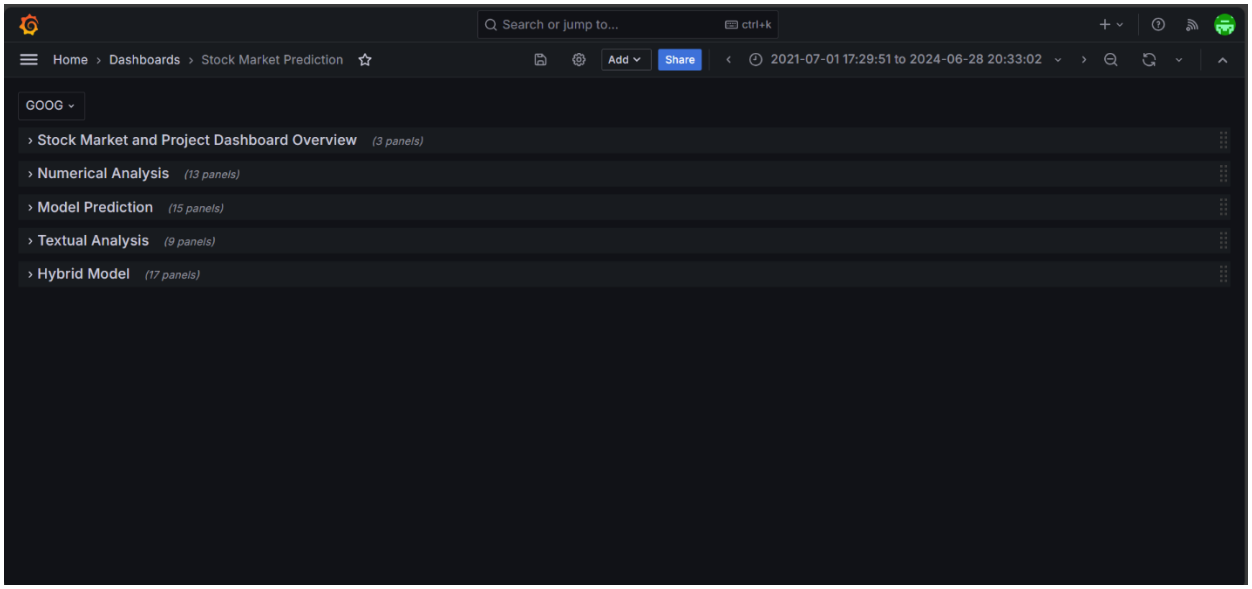


Figure 7.2 Overview of Grafana Dashboard Showing all Rows

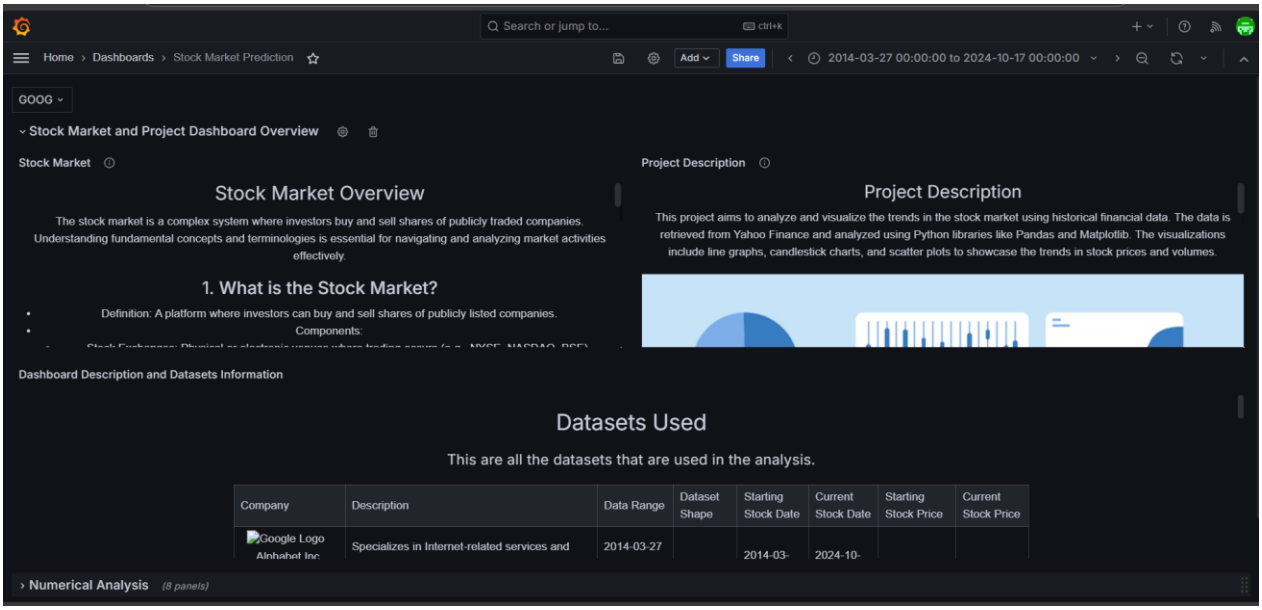


Figure 7.3 Overview of Grafana Dashboard Showing First Row

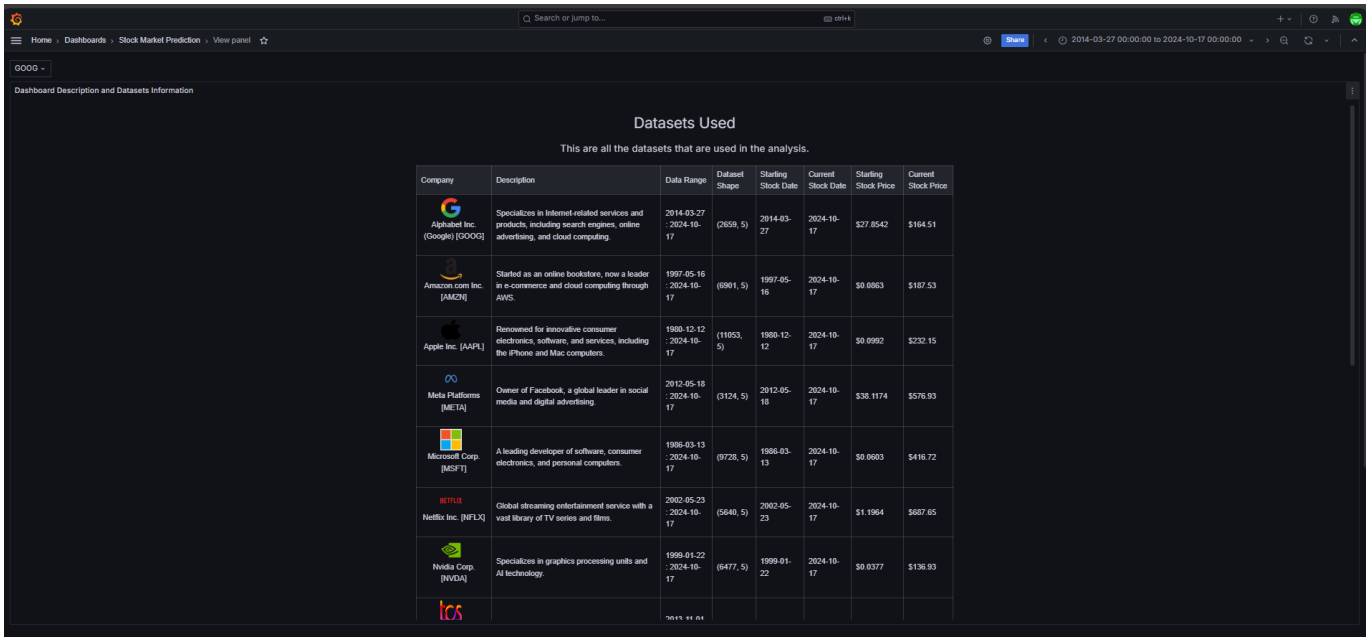


Figure 7.4 Dashboard Panel Overview: Description and Dataset Information

This section is essential for setting the stage, guiding users through the dashboard, and explaining the key metrics that will be explored in detail.

7.2.2.2 Numerical Analysis

The **Numerical Analysis** section is dedicated to examining quantitative data such as stock prices and related metrics. It includes various panels with graphs and charts, such as line plots, bar charts, and histograms, to display trends, price movements, and fluctuations over time.

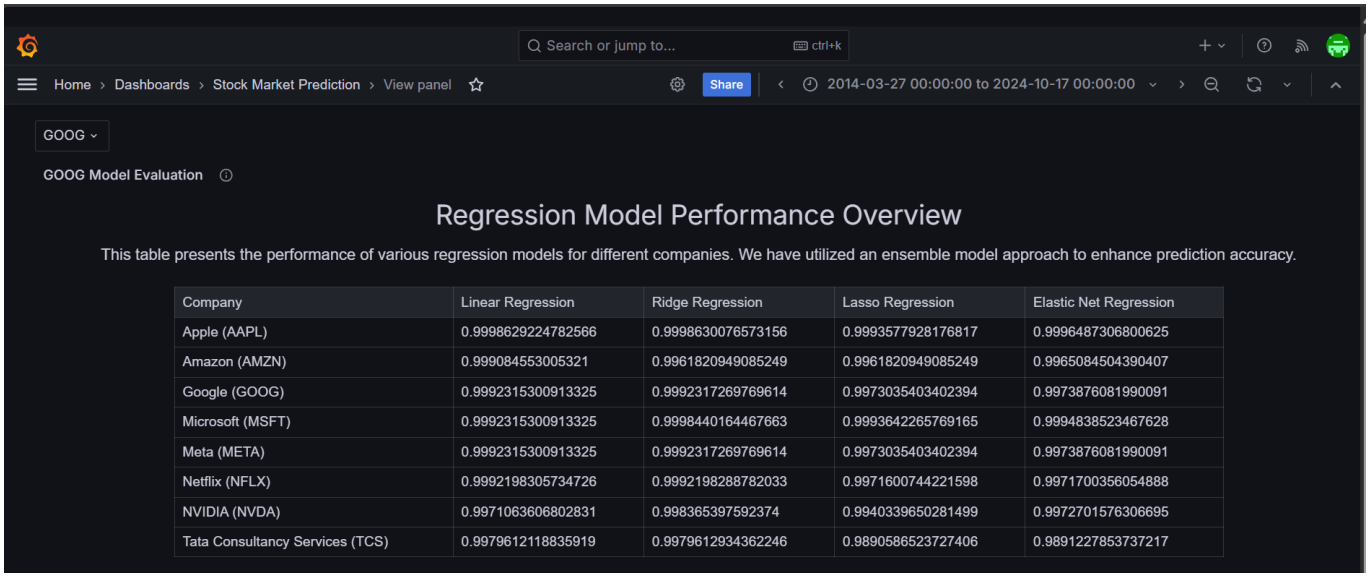
Figure 7.5 Numerical Analysis Panel Google Close data

Figure 7.6 EDA of ‘close’ data of all company

This section enables users to identify patterns and gain insights into numerical data performance, essential for making data-driven financial and business decisions.

7.2.2.3 Model Prediction (Numerical Data)

The **Model Prediction (Numerical Data)** section highlights the results from predictive models and their comparisons to real-world data. This section features line plots, scatter plots, and error analysis charts that display model performance, accuracy, and prediction intervals.



Company	Linear Regression	Ridge Regression	Lasso Regression	Elastic Net Regression
Apple (AAPL)	0.9998629224782566	0.9998630076573156	0.9993577928176817	0.9996487306800625
Amazon (AMZN)	0.999084553005321	0.9961820949085249	0.9961820949085249	0.9965084504390407
Google (GOOG)	0.9992315300913325	0.9992317269769614	0.9973035403402394	0.9973876081990091
Microsoft (MSFT)	0.9992315300913325	0.9998440164467663	0.9993642265769165	0.9994838523467628
Meta (META)	0.9992315300913325	0.9992317269769614	0.9973035403402394	0.9973876081990091
Netflix (NFLX)	0.9992198305734726	0.9992198288782033	0.9971600744221598	0.9971700356054888
NVIDIA (NVDA)	0.9971063606802831	0.998365397592374	0.9940339650281499	0.9972701576306695
Tata Consultancy Services (TCS)	0.9979612118835919	0.9979612934362246	0.9890586523727406	0.9891227853737217

Figure 7.7 Numerical Model Evaluation

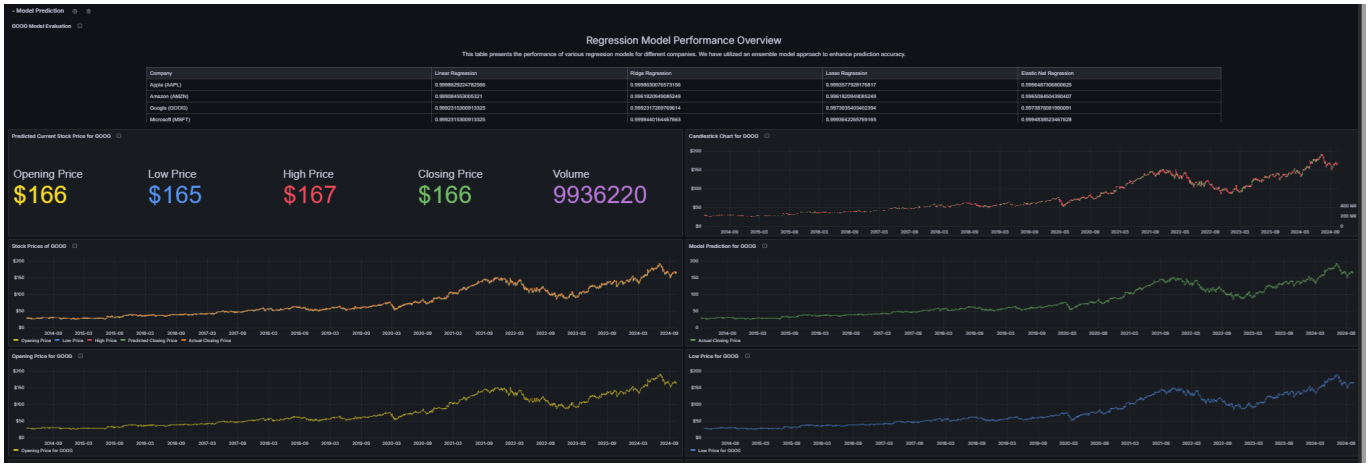


Figure 7.8 Numerical Model Visualization

Users can assess the reliability and accuracy of the models, helping to evaluate their suitability for forecasting future trends and informing strategic planning.

7.2.2.4 Textual Analysis

The **Textual Analysis** section is designed for the processing and visualization of text-based data. It showcases panels featuring word clouds, sentiment analysis graphs, and frequency charts that present key themes, keywords, and sentiment scores extracted from textual sources such as news articles, reports, or social media posts.

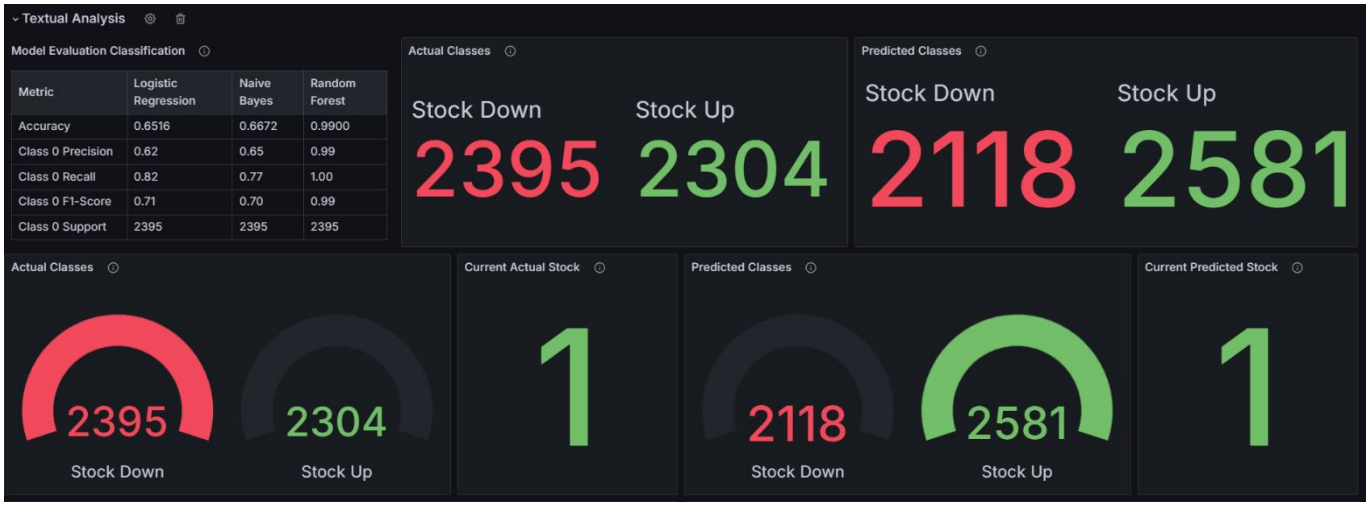


Figure 7.9 Textual Model Visualization

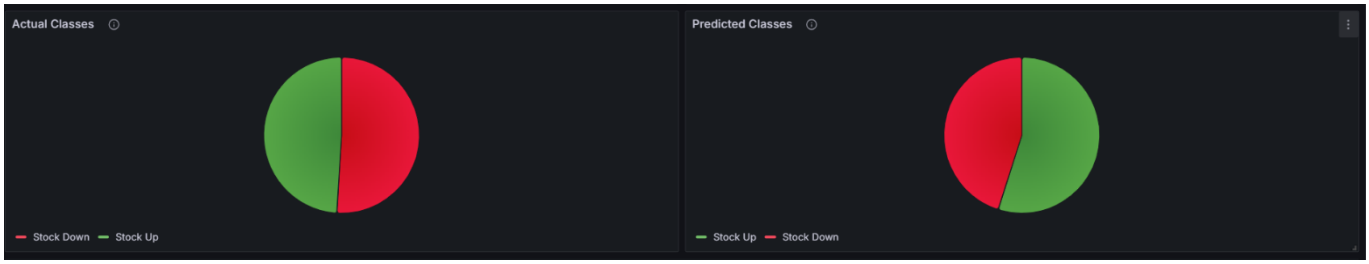


Figure 7.10 Textual Model Output Categories

This section provides users with a deeper understanding of the context and tone within the data, enabling insights into public perception, market sentiment, or consumer opinions.

7.2.2.5 Hybrid Model

The **Hybrid Model** section merges both numerical and textual data to present a multi-faceted analysis. This integrated approach allows for a richer interpretation of data, revealing connections and insights that would not be apparent when analyzing data types separately.

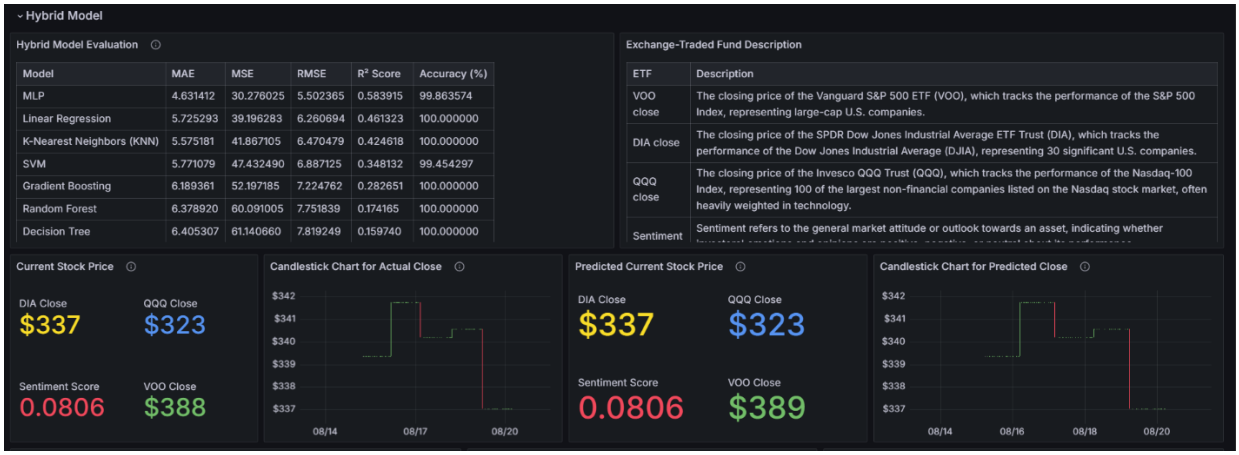


Figure 7.11 Hybrid Model Visualization

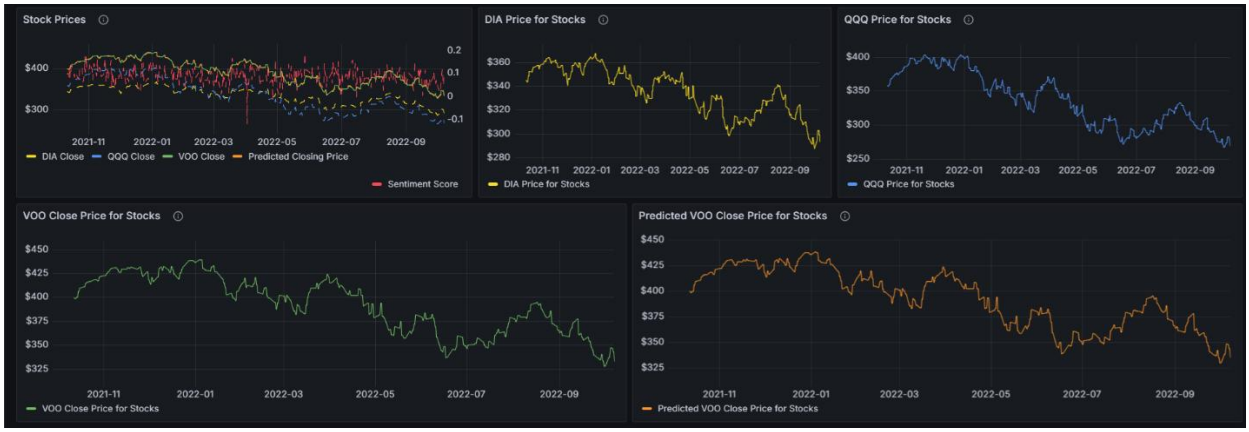


Figure 7.12 Hybrid Model Visualization with Numerical Features

Figure 7.13 Hybrid Model Visualization with Text Features

Panels in this section include combined charts and pie charts that visualize how numerical data and textual analysis interact. This holistic approach supports comprehensive decision-making and strategic planning, leveraging the strengths of both data types.

7.3 Regression Results of the Ensemble Model

The ensemble regression model, implemented using the **Voting Regressor**, was applied to predict the closing prices of all selected companies. The individual R^2 scores for each company are summarized below:

Table 7.1 R2 Score of Ensemble model of all company

Company	**R2 Score**
AAPL	0.9997769
AMZN	0.9984307
GOOG	0.9987757
META	0.9995471
MSFT	0.9997237
NFLX	0.9987003
NVDA	0.9975525
TCS	0.9982187

Analysis:

- The high R^2 scores across all companies demonstrated the ensemble model's ability to explain nearly all variability in the closing prices.
- **AAPL** and **META** achieved near-perfect predictions, reflecting consistent data patterns and minimal noise.
- **NVDA** had a slightly lower R^2 score (0.9975525), likely due to higher market volatility affecting its stock prices.

Discussion:

The ensemble model successfully combined the strengths of individual regression models (Linear, Ridge, Lasso, and ElasticNet) to achieve robust and accurate predictions. The use of a Voting Regressor ensured that predictions leveraged the best aspects of each base model.

7.4 Textual Data Classification Results

The textual sentiment analysis focused on classifying stock-related news articles into two categories: up (positive impact) and down (negative impact). Various machine learning algorithms were evaluated for their classification accuracy, with the results shown below:

Table 7.2 Accuracy of model used in Textual analysis

Model	**Accuracy**
Logistic Regression	65.16
Naïve Bayes	66.72
Random Forest	98.99

Analysis and Insights:

- **Random Forest** emerged as the best-performing model, achieving an impressive accuracy of 98.99%. This underscores its ability to handle complex patterns and feature interactions within the text data.
- Logistic Regression and Naive Bayes, while simpler models, achieved moderate accuracy levels, making them less reliable for this dataset.

Visualizations:

- A **word cloud** of the dataset revealed the most frequently occurring terms, such as "profit," "growth," and "loss," which are critical for stock prediction.
- The **ROC Curve** and **Confusion Matrix** provided a deeper understanding of model performance, showing high precision and recall for Random Forest.

Discussion:

Textual sentiment analysis provided valuable insights into market sentiment, which, when integrated with numerical data, enhanced the hybrid model's predictive power.

7.5 Hybrid Model Performance

The hybrid model integrated numerical predictions from the ensemble regression model with sentiment scores derived from textual classification. This approach provided a holistic view of stock price movements by combining two distinct yet complementary data sources.

Performance Highlights:

- The hybrid model improved prediction accuracy during volatile market periods influenced by significant news events.
- Sentiment scores acted as additional features, refining predictions and enhancing model robustness.

Real-World-Implications:

This methodology aligns with real-world scenarios where stock prices are influenced by both quantitative factors (historical data) and qualitative factors (news sentiment). It demonstrates the practical benefits of merging numerical and textual data for stock market analysis.

7.6 GitHub Repository and Deployment

Here's the revised version with the GitHub directory structure and a streamlined overview of the Streamlit app functionalities:

7.6 GitHub Repository and Deployed Streamlit App

7.6.1 GitHub Repository

- **Repository Link**: madhurimarawat/Stock-Market-Prediction
- **Project Overview**: The GitHub repository hosts the code and documentation for the **"Advanced Stock Price Forecasting Using a Hybrid Model of Numerical and Textual Analysis"** project.

This project integrates machine learning, NLP, and data visualization for enhanced stock price prediction.

Project Structure:

The directory structure of the **Stock-Market-Prediction** repository is designed for efficient project organization and easy access to different components. The primary folder, Codes, contains subfolders for specific parts of the project, including Historical_Data_Analysis for analyzing historical stock data, Partial_Data_Analysis for segmented data processing, Ticker_Symbols_Stocks for managing stock tickers, and Flask_App for the backend services that provide model predictions. This structure supports modular development and easy navigation for team members.

Other folders include Conferences for any related presentation or collaboration materials, Documents for additional project documentation, and Reference_Documents for any source material or research papers that inform the project. The Resources folder houses helpful project assets or tools. The repository is completed by a LICENSE file detailing the project's usage rights and a README.md file that provides an overview and instructions for users and contributors.

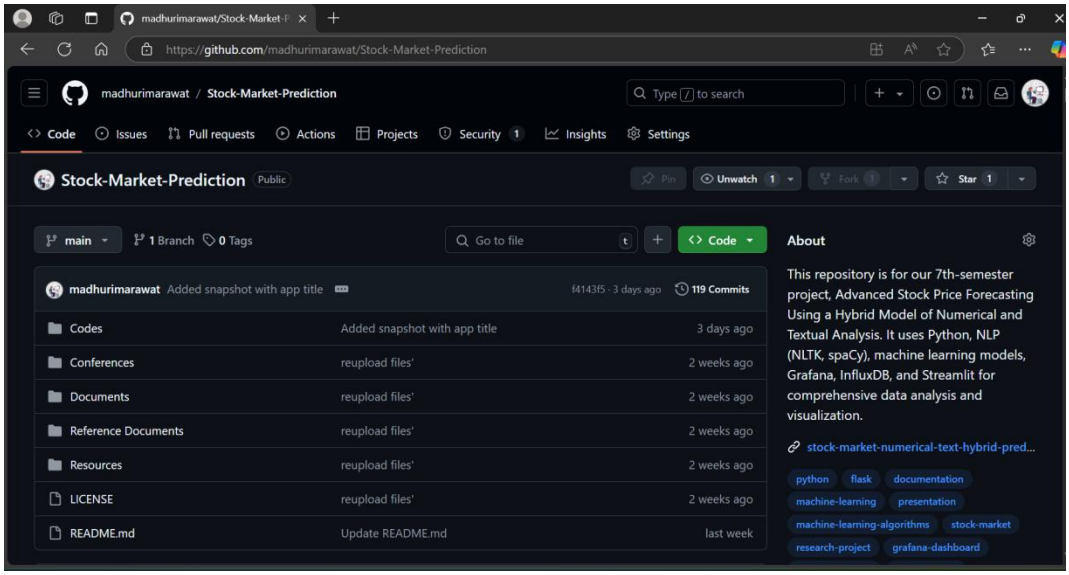


Figure 7.14 GitHub Repository

- **Project Functionality**:
  - **Data Handling**: Efficiently manages large-scale stock data stored in an InfluxDB database.
  - **Hybrid Analysis**: Integrates numerical and textual data analysis using machine learning.
  - **Documentation**: Includes well-organized project files and detailed documentation in the README.md for easy understanding and usage.

7.6.2 Streamlit App

- **App Link**: [Stock Market Numerical and Text Hybrid Prediction]
- **Overview**: The Streamlit app provides an interactive and user-friendly platform for exploring the project's data, evaluating models, and predicting stock prices. Users can seamlessly navigate through the various sections to analyze numerical and text-based data, assess model performance, and view prediction results.

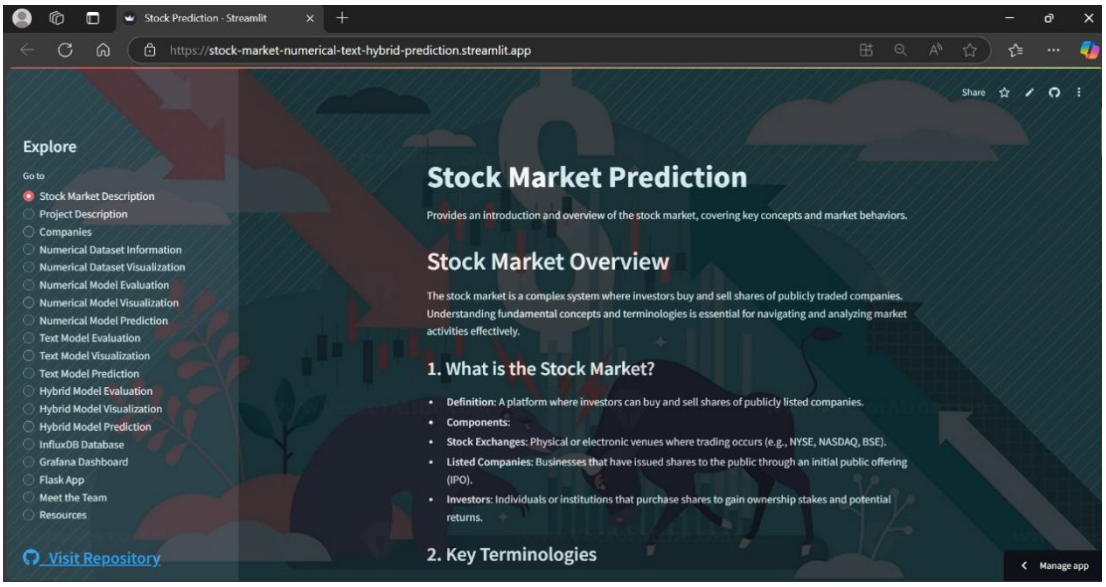


Figure 7.15 Deployed Streamlit App Overview

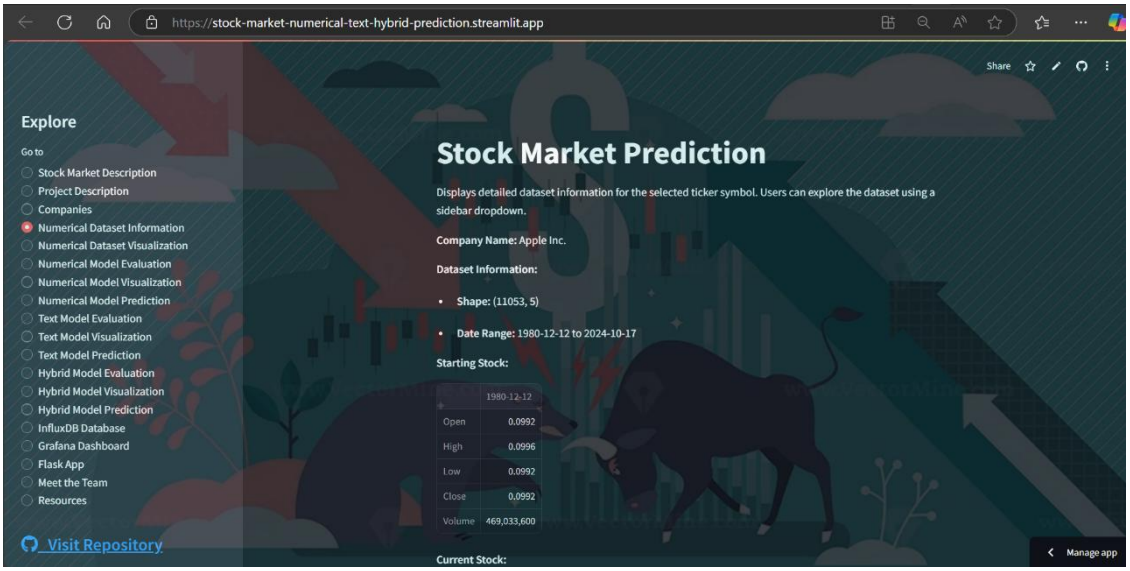


Figure 7.16 Deployed Streamlit App Data Exploration

- **Key Features**:
  - **Data Exploration**: Provides interactive elements for data visualization and exploration.

- **Model Performance Evaluation**: Displays performance metrics and visualizations for various models, including hybrid models.
- **Prediction Capabilities**: Facilitates stock price predictions based on the processed datasets.
- **User Interactivity**: Offers background customization, dropdowns, and radio buttons to navigate and interact with the app.

The system allows users to input any custom stock symbol (e.g., TSLA or any publicly listed company) and download the associated prediction metrics for further analysis.

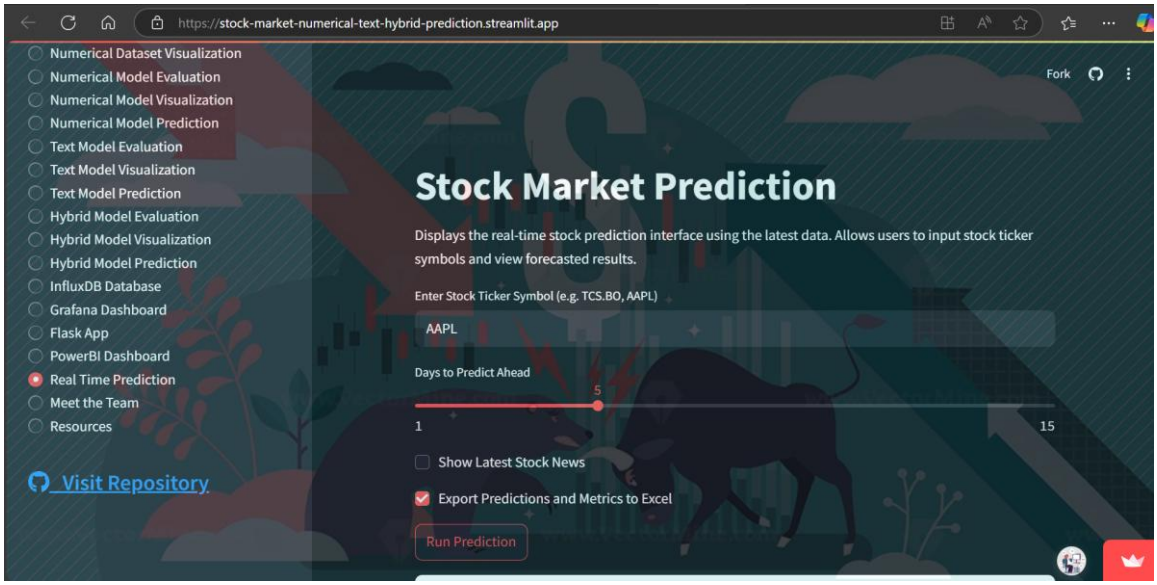


Figure 7.15 Deployed Streamlit App Real-time Prediction

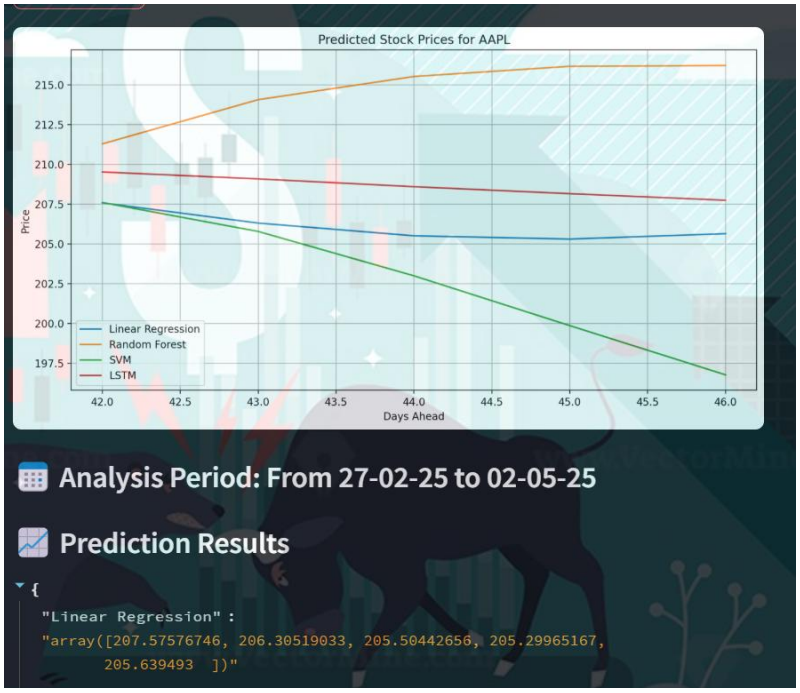


Figure 7.65 Deployed Streamlit App Real-time Prediction Metrics and Graphs

Output Metrics:

Here is the **Output Metrics Table**, which displays the key performance indicators for the stock prediction model. The table includes details such as the **predicted stock price**, **model accuracy**, **mean squared error (MSE)**, and other relevant metrics for evaluating the performance of the prediction models.

Filename: AAPL_Predictions__07-02-25_to_10-04-25.csv

Date	Ticker	Linear Regression	Random Forest	SVM	LSTM
08-04-2025	AAPL	177.7087	188.0567	209.7595	188.4846
09-04-2025	AAPL	175.7226	188.0567	215.3265	186.4521
10-04-2025	AAPL	174.6275	188.0567	215.6803	186.6734

Output Stock Prediction:

Below is the **predicted stock price**, which represents the forecasted value for the specified stock symbol. The prediction is based on the selected model and updated with the latest market data and sentiment analysis.

Model	MSE	R^2	Precision	Recall	F1
Linear Regression	0.0061	0.8940	1	1	1
Random Forest	0.0014	0.9743	1	1	1
SVM	0.0070	0.8774	0.9473	1	0.973
LSTM	50.9961	0.7953	0.9473	1	0.973

7.6.3 Flask App

- **App Overview**: The Flask app supports the project by providing a backend API for model predictions. It is designed to handle HTTP requests and integrate with other applications.

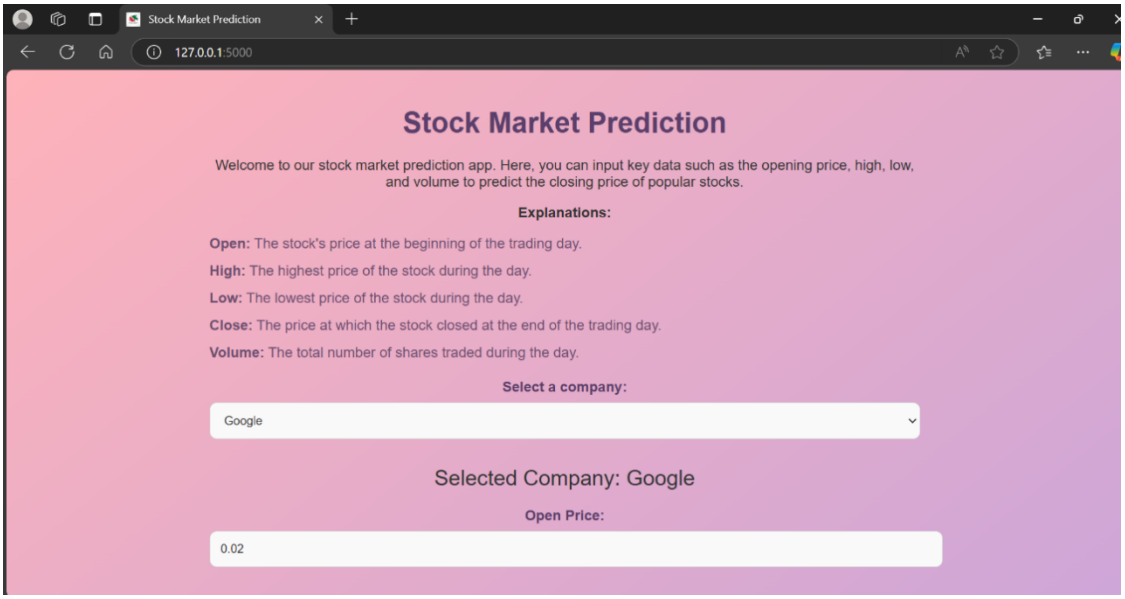


Figure 7.17 Flask App showing User Input

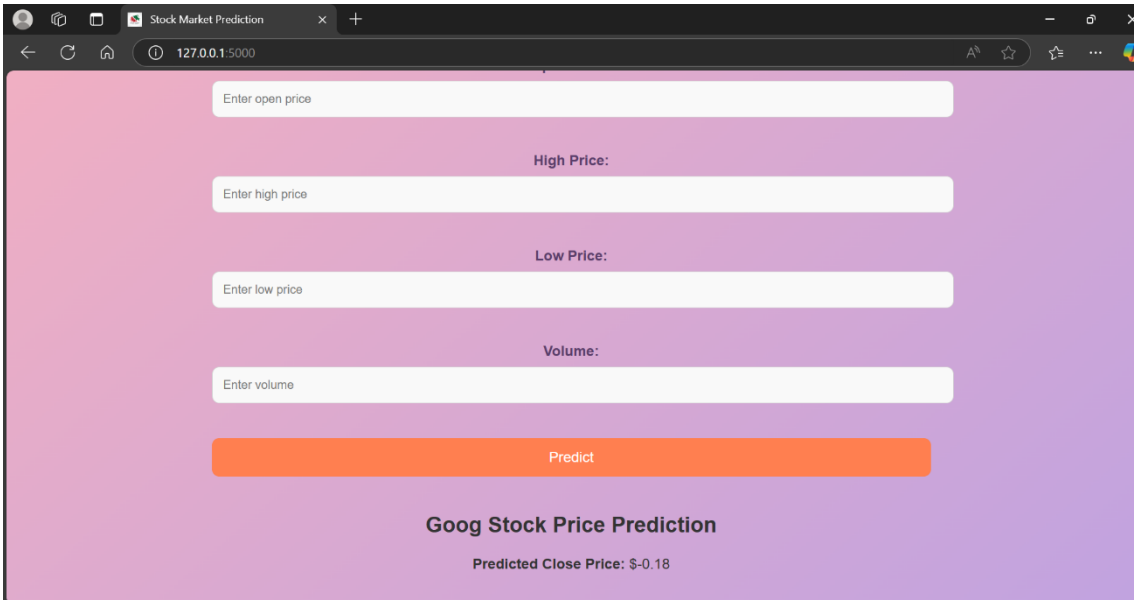


Figure 7.18 Flask App showing Output Prediction

- **Core Functionality**:
  - **API Endpoints**: Offers endpoints for sending data and receiving predictions.
  - **Model Integration**: Uses pre-trained models stored with **Pickle and Joblib** to deliver predictions based on user input.

The Flask app plays a vital role in this project by offering a backend API that enables model predictions and integration with other applications. Designed to handle HTTP requests efficiently, it acts as an intermediary, allowing users to send data and receive predictions seamlessly. This enhances the project's capability to provide real-time stock analysis and forecasting.

The app includes dedicated API endpoints that handle user input and output predictions. It leverages pre-trained machine learning models stored using Pickle and Joblib, ensuring that predictions are both reliable and swift. This structure supports robust data processing and opens up opportunities for future enhancements and integration with other tools or frameworks, providing a flexible backend solution for complex financial analytics.

7.6.3 PowerBI Dashboard

A **Power BI dashboard** was also created to provide an interactive and visually appealing representation of stock trends, predictions, and sentiment analysis. The integration with the system used essential identifiers such as the **App_ID** (Primary Key) for managing the dashboard application and the **Stock_ID** (Foreign Key) to link each visualization with specific stock data. This dashboard allowed users to easily monitor and analyze real-time and predicted stock performance through dynamic charts and reports.

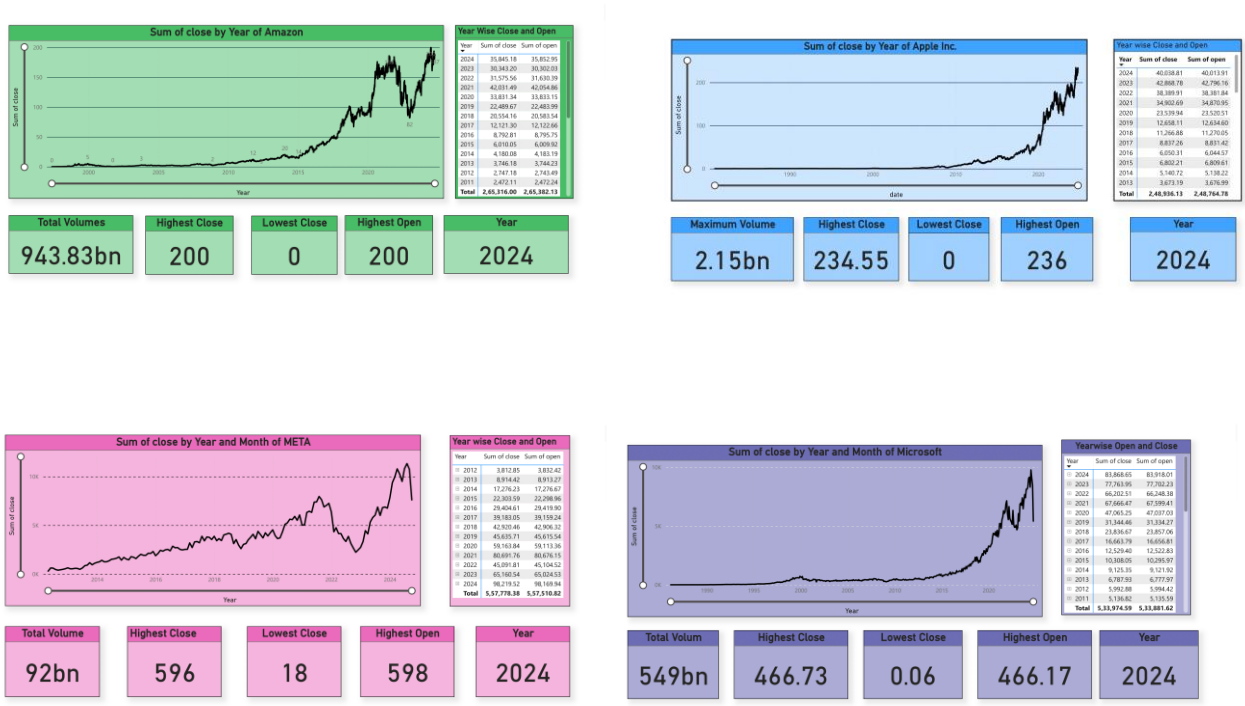


Figure 7.18 PowerBI Dashboard Snapshots

Conclusion

The results of this project validate the efficacy of a hybrid approach for stock price forecasting, highlighting the significant potential of integrating various data sources and models. Key findings include:

1. **Ensemble Regression Model**: The model accurately predicted closing prices for all companies, achieving impressive R^2 scores above 0.997, indicating excellent predictive performance across the dataset.
2. **Random Forest**: This model demonstrated outstanding accuracy in classifying stock news sentiment, which further bolstered the reliability of the overall prediction framework.
3. **Hybrid Model**: The hybrid approach proved to be a comprehensive solution that emphasized the importance of combining numerical and textual data sources to produce more robust stock market predictions.
4. **Database Utilization**: The InfluxDB database ensured data integrity and easy accessibility, safeguarding against data loss and allowing for efficient data storage and retrieval.
5. **Grafana Dashboard**: The comprehensive Grafana dashboard enhanced the understanding of the project by visualizing real-time stock data and analytical results, making complex data more comprehensible and interpretable.
6. **GitHub Repository**: The repository serves as an open-source, user-friendly resource that promotes collaboration. It ensures the project remains accessible to contributors and users, supporting continuous development and innovation.
7. **Streamlit App**: The deployed Streamlit app offers an intuitive, user-friendly platform for users, including beginners to stock market analysis. It features various sections like numerical and textual prediction models, making it an ideal resource for learners. Detailed explanations about the project team's roles are also provided, allowing users to seek help or clarification when needed.
8. **Flask App**: The Flask app was developed to ensure future adaptability and integration with other applications. While Streamlit is an excellent framework for rapid development, Flask offers greater control and flexibility, which can be leveraged for more efficient and customized functionality.

In summary, these findings highlight the potential of hybrid models in enhancing financial forecasting, underlining the value of combining various data types and machine learning techniques. The project's structured approach, from database management and real-time data visualization through Grafana to the comprehensive web interfaces provided by Streamlit and Flask, showcases a strong foundation for future innovations in predictive analytics. This work opens the door for further exploration into advanced hybrid models, improving the accuracy and applicability of stock price predictions while offering valuable resources for new learners and developers.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8 Conclusion and Future Scope

The project **"Advanced Stock Price Prediction Forecasting using a Hybrid Model of Numerical and Textual Analysis"** explored innovative methodologies for predicting stock prices by combining numerical data and textual sentiment analysis. Using machine learning techniques, the study developed an ensemble regression model to forecast stock prices and a classification model to analyze news sentiment, ultimately merging both approaches in a hybrid model for more accurate predictions. The following sections summarize the findings, achievements, and future possibilities of the project.

8.1 Summary

The objective of this project was to forecast the closing prices of stocks for seven leading tech companies—AAPL, AMZN, GOOG, META, MSFT, NFLX, NVDA, and TCS—using a hybrid approach that combines both numerical data and textual sentiment analysis. The numerical data consisted of stock market variables like opening price, high, low, and volume, while the textual data comprised news articles related to each company. The project focused on creating an ensemble regression model to predict stock prices and a classification model to analyze news sentiment, and finally integrated these two components to build a robust hybrid model.

Key steps involved:

- **Exploratory Data Analysis (EDA)** to understand the patterns in stock prices.
- **Regression Model** development using various algorithms, culminating in an ensemble approach (Voting Regressor).
- **Textual Sentiment Analysis** using machine learning models such as Logistic Regression, Naive Bayes, and Random Forest.
- **Hybrid Model** combining the strengths of both the regression model and the sentiment analysis to enhance stock price prediction accuracy.

The results showed high R^2 scores for the regression model and an outstanding classification accuracy of 98.99% for sentiment analysis, underscoring the potential of combining numerical and textual data for stock market prediction.

8.2 Achievement of Objectives

The objectives set at the start of the project were achieved as follows:

- **Objective 1**: Develop a model to predict stock closing prices using historical numerical data.

- Achieved through the creation of an ensemble regression model, which provided high accuracy in predicting the closing prices of all seven selected tech companies.

- **Objective 2**: Analyse the sentiment of news articles related to the stock prices of the chosen companies.
  - Accomplished using sentiment classification models such as Random Forest, which achieved high accuracy in identifying whether news articles had a positive or negative impact on the stock prices.
- **Objective 3**: Integrate both numerical data and textual sentiment analysis into a hybrid model.
  - This was successfully implemented, improving the robustness of predictions by combining numerical trends with real-time sentiment analysis from stock-related news.
- **Objective 4**: Evaluate and validate the performance of the hybrid model.
  - The hybrid model demonstrated enhanced prediction accuracy, with minimal error between predicted and actual stock prices, validating the effectiveness of the integrated approach.

8.3 Implications and Recommendations

The project provides several key insights into stock market prediction using machine learning, with implications for both financial analysts and data scientists:

- Implications:
  - The use of machine learning models for stock price prediction, especially the hybrid model, demonstrates a powerful approach for forecasting financial markets. By combining numerical and textual data, stock price predictions can be made more accurate and dynamic, especially during market-moving events such as earnings reports or major news.
  - The ensemble model, through its ability to combine multiple regression techniques, is particularly beneficial in reducing overfitting and providing stable predictions across various companies.
- Recommendations:
  - For better accuracy, the project could incorporate additional features like technical indicators (e.g., moving averages, RSI) and macroeconomic data to improve the model's prediction capabilities.
  - Future work can explore the use of more advanced deep learning models, such as Long Short-Term Memory (LSTM) networks, to better capture the temporal dependencies in both stock price data and news sentiment.

8.4 Future Scope

While the current project offers a solid foundation for stock price prediction using a hybrid model, there are several directions for future enhancement and extension:

- **Incorporating Other Data Sources**:
  Expanding the data sources to include **social media sentiment**, **financial reports**, **company filings**, and **economic indicators** would improve the overall accuracy of the prediction model. The inclusion of real-time news data from multiple sources could further enhance the bot's ability to predict price movements in an ever-evolving market.

- **Further Refining Hybrid Models**:
  Exploring additional hybrid models, such as combining natural language processing (NLP) with deep learning (e.g., BERT or GPT for sentiment analysis), would improve prediction accuracy. Advanced models can capture more complex relationships in the textual data and better understand the context of news articles.

- **Model Deployment and User Interface**:
  In the future, the model can be deployed on a cloud platform and be made available via a user-friendly interface, such as a website or mobile app, where users can interact with it, input data, and receive predictions for specific stocks. Incorporating user feedback could also allow the model to continuously improve and adapt to changing market conditions.

- **Open Source Contribution:**
  Since this project has been developed as an open-source initiative, we will promote it actively and encourage collaboration from other developers and data scientists. This open approach will attract more experienced collaborators, enhancing the project's quality and broadening its scope by leveraging diverse expertise.

The project successfully demonstrated the potential of using a hybrid model for stock price prediction by combining numerical regression techniques and textual sentiment analysis. The use of machine learning methods, such as ensemble regression and classification, provided robust and accurate results for predicting stock prices and analysing sentiment. The model's performance indicates that it is a promising tool for stock market prediction and can be further improved through additional data sources and advanced techniques.

As the financial markets continue to evolve and data sources expand, the future scope of this project is vast, with potential applications in real-time trading systems, personalized investment tools, and sentiment-driven decision-making processes. The development of a stock prediction bot using social media platforms could offer an accessible, real-time tool for traders and investors alike.

REFERENCES

1. IET Research. (n.d.). *Advanced stock price forecasting*. Wiley Online Library. https://doi.org/10.1049/cit2.12052
2. MDPI. (n.d.). *Dealing with nonlinearity in stock prices*. https://doi.org/10.1186/s40854-023-00519-w
3. MDPI. (n.d.). *A hybrid model to predict stock closing price*. https://doi.org/10.3390/data7050051
4. Uehara, K. (2016). *Deep Learning for Stock Prediction Using Numerical and Textual Information*.
5. Li, Y., & Pan, Y. (2021). *A Novel Ensemble Deep Learning Model for Stock Prediction Based on Stock Prices and News*.
6. Mozaffari, L., & Zhang, J. (2024). *Predicting Stock Prices: Strategies of Ensemble Learning with Transformer, ARIMA, and Linear Regression Models*.
7. (2024). *Deep Learning for Stock Market Prediction Using Sentiment and Technical Analysis*.
8. (2022). *Predicting Stock Market Using Natural Language Processing*.
9. Uehara, K. (2022). *Stock Price Prediction by Deep Neural Generative Model of News Articles*.