

**Capstone Project Phase A**

**Stock Trend Prediction Using Sentiment Analysis**

**Project code: 25-2-R-6**

**Shelly Trifonov  
Guy Naeh**

**Git repository link:**[**https://github.com/shellytrifonov/braude-stock-predictor**](https://github.com/shellytrifonov/braude-stock-predictor)

**Supervisor:  
Mr. Ilya Zeldner**

**Abstract 4**

**1 Introduction** **4**

**2 Background and Related Work** **5**

2.1 Financial Forecasting 5

2.1.1 Characteristics of Financial Time Series 5

2.1.2 Importance of Predictive Models in Finance 6

2.2 Deep Learning Models 6

2.2.1 Recurrent Neural Networks (RNN) and LSTM 6

2.2.2 Gated Recurrent Unit (GRU) 7

2.2.3 Convolutional Neural Networks (CNN) 8

2.3 Feature Engineering 8

2.3.1 Historical Price Data 9

2.3.2 News Features 9

2.3.3 Twitter Features 9

2.3.4 Technical Indicators 10

2.4 Sentiment Analysis 10

2.4.1 Role of Social Media 10

2.4.2 NLP Techniques for Sentiment Extraction 11

2.4.3 Integration of Sentiment and Market Data 11

2.4.4 News-Based Sentiment Analysis 11

2.5 Forecasting Challenges 12

2.6 Forecasting Evaluation Metrics 12

2.6.1 Root Mean Squared Error (RMSE) 12

2.6.2 Mean Absolute Error (MAE) 13

2.6.3 Mean Absolute Percentage Error (MAPE) 13

2.6.4 Directional Accuracy 13

2.6.5 Sharpe Ratio 13

**3 Expected Achievements 14**

3.1 Potential Problems 14

3.2 Success Criteria 15

**4** [**Engineering Process**](https://docs.google.com/document/d/1vkwSqJhJ_-abRNFm7SAKJc2mq8UmdIv9/edit#heading=h.b3re5hjlsh8t) **16**

4.1 Project Workflow 16

4.2 Data Acquisition and Preprocessing 17

4.2.1 Historical Stock Data 17

4.2.2 News Data 17

4.2.3 Twitter Data 17

4.2.4 Dataset Selection Criteria 18

4.2.5 Data Synchronization and Temporal Alignment 18

4.3 Sentiment Feature Extraction 19

4.3.1 Text Preprocessing 19

4.3.2 Keyword and Entity Filtering 19

4.3.3 Sentiment Scoring 20

4.3.4 Temporal Aggregation 20

4.4 Feature Engineering 20

4.4.1 Model Selection Rationale 20

4.4.2 Stock Features 22

4.4.3 Sentiment LSTM Encoding 22

4.5 Multimodal Fusion 23

4.6 Prediction Module 24

4.6.1 Feature Refinement Layer 24

4.6.2 Output Layer and Decision Rule 24

4.6.3 Training Procedure 25

4.6.4 Loss Function and Optimization 26

4.6.5 Classification Metrics 26

4.7 Tooling and Environment 27

4.7.1 Languages & Frameworks 27

4.7.2 Selection of Data Storage and Processing Framework 28

**5 Product** **30**

5.1 Architecture overview 30

5.2 Requirements 30

5.2.1 Functional Requirements 30

5.2.2 Non-functional Requirements 31

5.3 System Use Case 33

5.4 System Activity 34

5.5 Screens 34

**6 Evaluation Plan** **35**

6.1 Scope 35

6.2 Objectives 35

6.3 Test Cases 35

**7 References** **38**

**8 AI Prompts 39**

**Abstract**

Accurately forecasting stock prices remains a complex and dynamic challenge due to the constant impact of external factors such as geopolitical events, macroeconomic trends, and rapidly shifting public sentiment. This innovative project introduces a hybrid deep learning-based forecasting system that integrates historical financial data with sentiment signals to enhance stock market prediction. The system leverages both numerical stock data and unstructured textual content from news sources and social media platforms like X (formerly Twitter), uncovering subtle correlations between market movements and public discourse. By analyzing these diverse data streams in parallel, the model captures complex patterns that traditional models often miss. Ultimately, the system has the potential to significantly influence economic activity by improving investment strategies, enhancing market efficiency, and contributing to more stable financial ecosystems.

**1 Introduction**

Predicting stock prices has long been a major challenge for investors, researchers, and financial institutions. Accurate forecasts can reduce investment risk and support data-driven decision-making. Traditional forecasting methods rely primarily on historical financial data, such as price, volume, and technical indicators. While these methods are effective for identifying recurring patterns, they often fall short in capturing the real-time factors that drive sudden market changes.

In today’s interconnected world, stock markets are heavily influenced by two powerful forces: breaking news events and public sentiment. News reports, ranging from geopolitical conflicts to corporate announcements, can rapidly shift investor expectations and cause dramatic price movements. In parallel, social media platforms like X (formerly Twitter) serve as a real-time pulse of public opinion, offering insights into how people perceive companies, sectors, or global events.

Sentiment analysis, powered by natural language processing (NLP), enables machines to interpret emotional signals embedded in public commentary. Meanwhile, financial news analysis focuses on extracting factual insights and detecting the relevance of events to specific assets. By combining these two sources, objective news and subjective opinion, stock prediction models can capture a broader, more nuanced picture of market behavior.

Previous research has shown that combining sentiment analysis with historical data improves stock price prediction performance [4]. In parallel, LSTM-based models have demonstrated superior accuracy over traditional statistical methods when working solely with historical financial data [1]. Building on these findings, our project introduces a deep learning-based forecasting system that fuses historical stock data, financial news, and sentiment from X. By analyzing both numerical and textual information in parallel, the system uncovers patterns that reflect not only past performance, but also current events and collective reactions. The result is a hybrid model that enhances prediction accuracy and provides more reliable, context-aware insights to investors.

**2 Background and Related Work**

**2.1 Financial Forecasting**

Financial market forecasting represents a foundational yet formidable challenge within the domain of quantitative finance. The goal of such forecasting is to anticipate future price movements of assets, enabling investors to make more informed decisions, enhance returns, and manage risk more effectively. Over time, the forecasting discipline has evolved from purely statistical approaches to hybrid and learning-based models capable of processing vast and complex datasets.

Classical statistical models, such as ARIMA (AutoRegressive Integrated Moving Average), have been widely used for time series forecasting. These models require the data to be stationary, meaning that the statistical properties such as mean and variance remain constant over time, and assume linear relationships in the data. However, financial time series are often noisy, non-stationary, and nonlinear, which limits the effectiveness of such models. To address these limitations, recent approaches employ deep learning techniques that can capture complex temporal patterns directly from raw data [2].

Contemporary forecasting techniques utilize models like Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN), which have been shown to outperform traditional models in various empirical studies. LSTM, in particular, excels at learning long-range temporal dependencies, a critical capability when working with sequential stock data [6]. In conjunction with these models, the integration of technical indicators and sentiment-based inputs further improves model accuracy and robustness [1][4].

The growing complexity and noise in financial data underscore the need for sophisticated predictive systems that combine multiple data modalities and adapt dynamically to changing market conditions. These advancements are instrumental in areas such as high-frequency trading, portfolio optimization, and risk analytics.

**2.1.1 Characteristics of Financial Time Series**

Financial time series present unique challenges for forecasting due to their non-linear and non-stationary nature. In other words, the data's statistical properties, such as the mean and variance, tend to shift over time. This makes it difficult for traditional parametric models to accurately capture long-term patterns or trends in stock market behavior [1].

Additionally, the stock market is very volatile and reacts to many different factors. Some of them come from inside the market, like company earnings or supply and demand, while others are external, like political events or natural disasters. These things add a lot of noise to the data, which can make it hard to spot real patterns. On top of that, many features in the data are connected to each other; for example, technical and macroeconomic indicators often influence one another in ways that are hard to separate using traditional forecasting methods [6].

Given these attributes, there is an increasing reliance on advanced models capable of detecting latent patterns and adapting to dynamic environments. LSTM networks, for instance, have been employed successfully in multiple studies for modeling such complex sequences with promising results [2][6].

**2.1.2 Importance of Predictive Models in Finance**

Predictive models serve multiple strategic roles in modern financial systems. They are used to support algorithmic trading, guide asset allocation, and help manage financial risk. In addition to their operational value, predictive analytics also play an important role in long-term planning and investment research.

The ability to forecast asset behavior with reasonable accuracy helps investors and fund managers make better decisions, minimize losses, and aim for stronger returns. Models that are capable of handling complex and noisy data are especially useful in volatile markets, where traditional forecasting methods often fall short [6].

Furthermore, integrating additional data sources such as social media sentiment can enhance the model's responsiveness to market psychology. Recent work combining sentiment scores with price data has demonstrated improved prediction outcomes, showcasing the complementary role of Natural Language Processing in financial modeling [4].

**2.2 Deep Learning Models**

Recent developments in deep learning have brought meaningful improvements to financial forecasting, especially when dealing with stock market data that tends to be non-linear, noisy, and difficult to model using traditional tools. Deep learning models are particularly effective at identifying complex patterns in large, high-dimensional datasets, something that's crucial for tasks like predicting stock prices, assessing risk, or optimizing portfolios. Unlike more conventional approaches that often depend on predefined features and assume data stationarity, these models can automatically learn useful representations and detect time-based relationships in the data, which often translates into better forecasting performance.

**2.2.1 Recurrent Neural Networks (RNN) and LSTM**

Recurrent Neural Networks (RNN) are built to process sequential data by preserving information from previous steps in the form of a hidden state. This makes them suitable in theory for time-dependent data like stock prices. However, in practice, standard RNNs often struggle when the sequences are long, due to the vanishing gradient problem, where the network fails to retain relevant signals from earlier time steps as the sequence grows.

To address this, the Long Short-Term Memory (LSTM) architecture was introduced. LSTM networks include internal mechanisms known as gates, specifically, forget, input, and output gates, which help the model decide what information to keep, discard, or pass on at each step. These gates allow the network to maintain and update a form of memory over longer time periods, making it more effective for complex time series tasks.

In financial forecasting, LSTM has proven especially useful. For instance, one study applied LSTM models to data from the Tehran Stock Exchange, combining them with a set of technical indicators such as RSI and MACD. The results showed a clear improvement in prediction accuracy compared to a basic RNN, suggesting that LSTM’s memory mechanisms provide an advantage when dealing with financial data that contains both trends and short-term fluctuations [1]. These findings align with previous work comparing LSTM and RNN models in stock prediction tasks. In those studies, LSTM consistently achieved higher accuracy, while RNN struggled due to memory limitations and the vanishing gradient problem [2].

In another case, researchers explored how combining LSTM and sentiment scores derived from Twitter posts could enhance stock price prediction. The goal was to evaluate whether investor sentiment, often expressed in real time and informally, adds value when used alongside historical price data. While both LSTM and XGBoost were applied to the same enriched dataset, the results showed that XGBoost achieved higher prediction accuracy across multiple evaluation metrics. Nonetheless, the study demonstrated LSTM’s strength in processing sequential and unstructured data sources, reinforcing its role as a flexible modeling approach [4].

LSTM models have also been used to help with portfolio optimization. In one study, the model predicted how stocks might perform in the future, and those predictions were used to decide how to build the investment portfolio. This approach didn’t just improve expected returns, but also helped reduce risk. It was especially helpful in markets that are more volatile, where traditional methods usually have a hard time [6].

**2.2.2 Gated Recurrent Unit (GRU)**

The Gated Recurrent Unit (GRU) is a streamlined alternative to LSTM that was designed to retain many of its advantages while reducing architectural complexity. Unlike LSTM, which uses three gates, GRU relies on just two: the update gate and the reset gate. These components help the model decide how much of the past information to carry forward and when to reset memory, allowing GRU to handle sequential data efficiently without the overhead of managing a separate cell state. Thanks to its simpler structure, GRU tends to train faster and requires fewer resources, qualities that can be particularly useful in financial contexts where models need to be retrained frequently as new data becomes available.

In terms of performance, GRU has shown itself to be a strong contender. In one study that examined stock price prediction for Tesla, Ferrari, and Walmart, GRU and LSTM were both tested on the same data. While LSTM produced better forecasts for Tesla and Walmart, GRU had a slight edge when predicting Ferrari’s stock, suggesting that the relative strength of each model may vary depending on the unique characteristics of a specific stock [9]. Another comparison involved adding technical indicators, such as moving averages and momentum oscillators, to the input data. In that case, both GRU and LSTM improved significantly over traditional forecasting models, demonstrating their capacity to incorporate multiple signals and uncover patterns that simpler models might miss [5].

**2.2.3 Convolutional Neural Networks (CNN)**

Although primarily used in image processing, Convolutional Neural Networks (CNN) have also been adapted for financial forecasting. By transforming time series data into two-dimensional visual representations, CNNs can capture spatial features and temporal correlations that might be overlooked by sequence-based models.

In one such implementation, researchers used 2D histograms of quantized financial data and applied CNNs to forecast stock prices. The CNN model achieved an accuracy of 98.92%, outperforming traditional neural networks and demonstrating the method’s effectiveness in identifying complex market patterns [3].

Another study explored the use of CNNs within a sliding window setup to predict stock prices based on data from a national stock exchange. By analyzing short sequences of historical data, the model was able to detect local patterns and capture brief market movements. When compared to other deep learning approaches such as MLP, RNN, and LSTM, the CNN performed particularly well in volatile periods, suggesting its strength in handling short-term fluctuations [7].

These examples highlight the growing importance of CNNs in stock price prediction, particularly for capturing local and short-term patterns that traditional models often miss. As financial data becomes increasingly complex and dynamic, CNN-based architectures offer a powerful tool for uncovering subtle structures in time series data.

**2.3 Feature Engineering**

Feature engineering is a crucial process in the development of predictive models in finance. It involves transforming raw financial data into meaningful features that can enhance model performance and improve interpretability. In stock price prediction models, particularly those based on deep learning, the quality and selection of input features often determine the effectiveness of the prediction. This section presents a detailed overview of the most commonly used feature engineering approaches in financial modeling.

**2.3.1 Historical Price Data**

Historical stock data, including daily open, close, high, low prices, and traded volumes, forms the cornerstone of most predictive financial models. Feature engineering in this context involves deriving meaningful variables that capture the structure and behavior of the market over time. Common transformations include generating lagged values to reflect previous days performance, computing moving averages over short or long windows to smooth out noise, and calculating daily returns or percentage changes to emphasize volatility and directional shifts.

These engineered features are essential in helping models identify temporal patterns, such as trends, momentum, and reversals, that may not be evident in raw data. Deep learning models, particularly those designed for sequential inputs like Long Short-Term Memory (LSTM) networks, are well-suited to exploit these patterns, as they can model dependencies across time steps. Research has demonstrated that such preprocessing steps, when guided by financial insight, substantially improve model generalization, especially in volatile or regime-shifting market environments [1][2][6].

**2.3.2 News Features**

News data offers essential qualitative context to financial markets by highlighting real-world events such as economic reports, earnings releases, or political developments that often trigger immediate investor reactions. Feature engineering from news sources typically starts with preprocessing techniques like tokenization and named entity recognition (NER) to extract relevant financial terms. These texts are then analyzed for sentiment, often using models tailored to financial language, producing scores that reflect the tone of each news item.

Such sentiment scores are usually aggregated over specific time intervals and synchronized with stock price data to form predictive features. In some cases, topic modeling is also applied to group news articles by theme, helping the model distinguish between different market drivers (e.g., interest rates vs. corporate earnings). Studies have shown that combining these engineered features with historical stock data improves the performance of models like LSTM and GRU, particularly during volatile periods when market sentiment plays a critical role [1][2][6].

**2.3.3 Twitter Features**

Social media, especially Twitter (now X), has emerged as a powerful source for gauging market sentiment in real time. Feature engineering for Twitter data involves cleaning raw tweets, applying sentiment analysis, and aggregating sentiment scores over defined time intervals. These aggregated metrics are then aligned with stock price data to serve as predictors. Some models utilize embedding layers or recurrent neural architectures to capture nuanced sentiment trajectories. The integration of Twitter-based sentiment features has been shown to improve forecasting accuracy by capturing investor mood and public opinion shifts before they manifest in stock prices [4].

**2.3.4 Technical Indicators**

Technical indicators are mathematical constructs derived from historical price and volume data, frequently used to assess trends, momentum, and volatility in financial markets. Common indicators include Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Bollinger Bands, and stochastic oscillators. These tools provide traders and analysts with condensed representations of underlying market behavior, often highlighting overbought or oversold conditions and potential trend reversals.

In the context of deep learning models for financial forecasting, technical indicators are widely used as engineered features that enhance the model’s ability to capture complex market patterns. They are typically integrated directly into the input feature set or as part of hybrid architectures combining machine learning and statistical techniques. Studies have shown that incorporating technical indicators significantly improves predictive accuracy in models such as LSTM and GRU, owing to their ability to represent short-term dynamics and market microstructures [1][3][5]. For instance, research comparing LSTM and GRU architectures concluded that the inclusion of selected technical indicators reduced error rates and improved overall forecasting performance [5]. Similarly, other works validated that deep learning models incorporating a broad set of indicators outperformed traditional RNNs in both predictive accuracy and robustness across market conditions [1].

**2.4 Sentiment Analysis**

Sentiment analysis is increasingly utilized in financial prediction as a means of capturing the emotional tone of market participants. By leveraging natural language processing (NLP) techniques, it is possible to quantify public sentiment expressed in unstructured text sources such as social media and news articles. The combination of sentiment signals with historical price data has demonstrated an ability to enhance forecasting performance for stock price movements [4].

**2.4.1 Role of Social Media**

Social media platforms, particularly Twitter (now X), provide a substantial stream of real-time opinions and emotional expressions from investors, analysts, and the general public. These platforms have become essential sources for sentiment extraction due to their accessibility and volume. Studies have shown that integrating Twitter sentiment with financial time series data improves the accuracy of stock price prediction models. For instance, research combining LSTM with sentiment scores derived from tweets about companies like Apple, Google, and Tesla found that models incorporating sentiment significantly outperformed those based solely on historical data. Specifically, the sentiment-enhanced model achieved RMSE values of 8.17 for Apple, 12.46 for Google, and 13.65 for Tesla, while models relying only on historical prices yielded higher RMSE values, indicating reduced accuracy [4].

**2.4.2 NLP Techniques for Sentiment Extraction**

The extraction of sentiment from textual data involves several key natural language processing (NLP) steps aimed at converting unstructured text into meaningful numerical representations. This process typically begins with preprocessing operations such as converting text to lowercase, removing stop words, and applying stemming or lemmatization to standardize the content and reduce noise.

Once the text is cleaned, it is transformed into vectorized representations using methods such as bag-of-words, TF-IDF (Term Frequency–Inverse Document Frequency), or word embeddings like Word2Vec or GloVe, which capture contextual or semantic relationships between words.

In financial applications, especially those involving social media data, sentiment is often quantified through polarity scores, numerical values that reflect whether a text conveys a positive, negative, or neutral emotion. These scores are derived from lexicon-based approaches or trained sentiment classifiers and serve as the core features extracted for further predictive modeling [4].

**2.4.3 Integration of Sentiment and Market Data**

The integration of sentiment analysis with conventional financial indicators represents a hybrid modeling strategy. Instead of relying solely on technical or historical data, modern approaches incorporate features such as daily sentiment scores, tweet volume, and polarity averages. This multi-source approach has demonstrated improvements in predictive performance metrics such as RMSE and MAE.

For example, the study in [4] examined how the inclusion of Twitter-based sentiment scores affects stock price prediction. The results showed that the LSTM model trained with both historical and sentiment data significantly outperformed the same LSTM model trained only on historical data. These findings underscore the value of integrating sentiment information for improving model accuracy.

**2.4.4 News-Based Sentiment Analysis**

In addition to social media, financial news articles and corporate press releases are commonly used for sentiment analysis in stock forecasting. These sources typically feature more formal and structured language, often conveying institutional viewpoints and broader economic indicators. Sentiment derived from news, such as the tone of earnings reports or analyst commentary, can serve as a valuable supplement to traditional price-based models, helping to improve prediction accuracy and model robustness. Some forecasting approaches combine news sentiment with historical stock data, enabling the model to capture both market perception and underlying trends.

**2.5 Forecasting Challenges**

Forecasting stock prices remains a formidable task due to the inherently volatile and nonlinear behavior of financial markets. A primary challenge is the high degree of noise and unpredictability in financial time series, which often masks meaningful patterns and increases the risk of misleading model outcomes. This is especially critical in deep learning models like LSTM and CNN, which, despite their power, are susceptible to overfitting and reduced performance under volatile conditions [6].

Another significant difficulty is the integration of heterogeneous data sources such as historical prices, technical indicators, and textual sentiment from news or social media. Each of these data types differs not only in format and structure, but also in frequency, latency, and semantic content. For example, price and volume data are typically recorded at fixed intervals (e.g., minute, hourly, or daily), whereas news events occur irregularly and may have a delayed or anticipatory effect on market behavior. Social media signals, on the other hand, are often high-frequency but noisy, and their relevance to financial outcomes may depend on context, language, or user credibility.

Synchronizing these sources temporally poses a major challenge. Misalignments in timestamp granularity or delays between an event and its market impact can distort causality and reduce the model’s predictive power. This is especially critical for models targeting short-term or intraday forecasting, where precision in timing is essential.

Moreover, the feature engineering phase involves its own set of complications. Redundant or highly correlated indicators can lead to multicollinearity, reducing model robustness. On the other hand, overlooking subtle but influential features might limit the predictive power. Striking a balance between model complexity and generalizability remains a persistent design challenge [5].

**2.6 Evaluation Metrics**

Accurate evaluation metrics are crucial for assessing the performance of predictive models in financial applications. These metrics not only measure how close the predictions are to actual values but also provide insights into risk-adjusted returns, model robustness, and market responsiveness. The following subsections describe the most widely used metrics in stock price prediction and trading strategy evaluation.

**2.6.1 Root Mean Squared Error (RMSE)**

The Root Mean Squared Error (RMSE) is one of the most frequently used metrics for evaluating the accuracy of numerical predictions. It calculates the square root of the average of the squared differences between predicted and actual values. Because it squares the errors before averaging, RMSE gives more weight to larger deviations, which is useful when large errors are especially undesirable. In the context of financial forecasting, RMSE serves as a reliable indicator for comparing the performance of different predictive models on time-series data such as stock prices or returns.

**2.6.2 Mean Absolute Error (MAE)**

Mean Absolute Error (MAE) is a simple and intuitive metric that measures how far predictions are from actual values, on average. Unlike RMSE, which gives more weight to large errors, MAE treats all errors equally by taking the absolute difference between each predicted and actual value and then averaging them. This makes MAE more stable and less sensitive to outliers. It is particularly useful when a balanced and consistent measure of prediction accuracy is desired, without disproportionately penalizing occasional large deviations.

**2.6.3 Mean Absolute Percentage Error (MAPE)**

Mean Absolute Percentage Error (MAPE) measures the average size of the prediction error as a percentage of the actual values. It is widely used because it is easy to interpret and does not depend on the scale of the data, which makes it useful when comparing forecasting performance across different stocks or time periods. However, one limitation of MAPE is that it can become unstable when actual values are very close to zero, since dividing by small numbers can exaggerate the error.

**2.6.4 Directional Accuracy**

Directional Accuracy is a metric that evaluates whether a model correctly predicts the direction of price movement, whether the price will go up or down, without considering the size of the change. This metric is especially useful in financial applications where the correct trend matters more than the exact predicted value. For example, traders who make buy or sell decisions based on anticipated market direction benefit from models with high directional accuracy, even if the numerical price predictions are slightly off.

**2.6.5 Sharpe Ratio**

The Sharpe Ratio is a common measure used to assess the performance of a model or investment strategy in terms of both return and risk. Specifically, it compares the excess return (over a risk-free rate) to the variability of those returns. A higher Sharpe Ratio indicates that the model or strategy provides better returns relative to the amount of risk taken, making it a key metric for evaluating profitability under uncertainty.

where is the average return of the portfolio or strategy, is the risk-free rate, and ​ is the standard deviation of the portfolio’s returns.

**3 Expected Achievements**

This project aims to design and implement a multi-source stock market forecasting system based on deep learning and sentiment analysis. Upon completion, we expect to achieve the following outcomes:

* **Accurate Stock Movement Forecasting**  
  A trained and validated model capable of predicting short-term stock price direction (up or down) with improved accuracy over traditional models. This enables more informed trading strategies and enhances investor confidence in model-based decision-making.
* **Data-Driven Sentiment Integration**  
  By incorporating sentiment extracted from both Twitter and financial news, the system demonstrates how qualitative signals can strengthen quantitative models.
* **Actionable Forecasting Tool**  
  The system is designed not just as a proof-of-concept, but as a functional tool for real-world financial analysis. It enables practical applications such as trend monitoring, event-driven portfolio adjustments, and strategy backtesting, especially useful in dynamic markets where sentiment changes rapidly.
* **User-Friendly and Modular Design**  
  Throughout the development process, emphasis will be placed on creating a system that is both easy to understand and flexible to use. Its modular architecture will allow researchers or financial practitioners to extend the pipeline to new stocks or data types without major reconfiguration, making the tool accessible even to those with limited machine learning experience.

**3.1 Potential Problems**

While the system is designed for robustness, several challenges may arise:

* **Risk of Overfitting to Narrow Market Scenarios**  
  There is a risk that the model will become too tailored to specific stocks or time periods used during training. This could limit its usefulness when applied to new or unexpected market conditions, such as economic crises or regulatory changes.
* **Uncertainty in Sentiment Interpretation**  
  Even with advanced models like FinBERT or RoBERTa, interpreting the true tone of tweets or financial headlines is not always straightforward. Sarcasm, slang, or vague wording can result in incorrect sentiment readings, which may lead the model to make poor predictions.
* **Difficulty in Generalization Across Assets and Timeframes**  
  A model that works well for one stock in a specific time frame might not perform as well on other stocks or during different market cycles. This lack of generalizability poses a challenge for applying the system broadly in dynamic financial environments.
* **Computational Complexity**  
  The system's reliance on deep learning techniques such as LSTM and transformers increases the computational cost. Training and deploying the model may require hardware resources that are not always available, particularly in settings where speed and scalability are important.
* **Systemic Interdependencies and Hidden Influences**Financial markets are highly interconnected and influenced by complex global dynamics. A seemingly unrelated event, such as geopolitical news, a change in interest rates, or a viral social media post, can unexpectedly impact a specific stock, even if it is not mentioned explicitly. These indirect effects are difficult to capture through standard modeling pipelines, and they pose a fundamental challenge for training models that rely on localized or stock-specific input features. Teaching a machine to infer such latent relationships remains an open research problem in financial AI.

**3.2 Success Criteria**

The project will be considered successful if the following technical and practical goals are met:

* **Improved Forecasting Accuracy**  
  The final model should consistently achieve high predictive performance in terms of RMSE and Directional Accuracy across multiple test sets. This reflects its ability to capture meaningful patterns in the data and make reliable short-term predictions.
* **Superiority over Historical-Only Models**  
  The model should outperform baseline models that rely solely on numerical historical stock data. Demonstrating a measurable improvement over these traditional approaches will validate the added value of incorporating sentiment signals from financial news and social media.
* **Modular and Reusable Components**  
  Each part of the system from data collection to sentiment modeling and prediction should be structured as a reusable module. This makes the system flexible for future extensions, whether for different assets, new data types, or integration with other financial tools.
* **User Feedback and Satisfaction**  
  Preliminary user testing with finance or data science students and practitioners should indicate at least 80% satisfaction with the system’s usability, clarity of results, and practical relevance.
* **System Reliability and Uptime**  
  The system should demonstrate a high level of reliability, including at least 99% operational uptime during extended test periods. It must remain stable under large-scale inputs and handle typical usage without failure.

**4** [**Engineering Process**](https://docs.google.com/document/d/1vkwSqJhJ_-abRNFm7SAKJc2mq8UmdIv9/edit#heading=h.b3re5hjlsh8t)

**4.1 Project Workflow**

The project is structured as a multi-stage process, beginning with a review of related work in financial forecasting and sentiment analysis. The core problem addressed is the limited accuracy of traditional models that rely solely on numerical stock data, neglecting the impact of real-time public sentiment and news events on short-term market behavior. To overcome this limitation, we focused on designing a system architecture that integrates three primary data sources: historical stock prices, financial news headlines, and Twitter posts. Special emphasis was placed on ensuring that all data sources are chronologically aligned, as temporal consistency is essential for modeling financial trends accurately.

The proposed architecture includes a dedicated processing pipeline for each data type. Stock price data is to be handled using time-series encoding via an LSTM model trained on 60-day sequences. News and Twitter data are each processed through separate sentiment analysis pipelines involving FinBERT and RoBERTa, respectively, followed by temporal aggregation and encoding using independent LSTM models. These three LSTM models are designed to capture the distinct temporal dynamics within each modality.

The outputs of the three pipelines are 128-dimensional feature vectors intended to be concatenated into a unified multimodal representation. This vector will be passed through fully connected layers to produce a binary prediction indicating the expected market direction (Up or Down).

**4.2 Data Acquisition and Preprocessing**

This stage establishes the foundational dataset infrastructure used for all downstream analysis, modeling, and prediction components. The project focuses on two companies, Tesla (TSLA) and Apple (AAPL), selected due to their high trading volume, strong presence in financial news and social media, and historical relevance in stock prediction research. These characteristics ensure a rich and diverse dataset across all three modalities: historical stock prices, financial news headlines, and Twitter posts. Each data stream is processed independently according to its nature, with a consistent focus on chronological alignment across all sources.

**4.2.1 Historical Stock Data**

Historical stock price data will be collected using Python’s yfinance library [25], which retrieves time-series financial data from Yahoo Finance [24]. The dataset will include the following standard indicators: Date, Open, High, Low, Close, Volume, and Adjusted Close. These columns are selected because they represent the fundamental elements used in technical stock analysis. They provide key insights into market behavior for each trading day and are commonly used to derive various technical indicators.

Once retrieved, the data will be processed using the pandas library [17] to handle missing values via forward filling. Additional preprocessing steps will include converting the date column to datetime format, sorting the data chronologically, setting the date as the index, and removing any irrelevant columns. These steps ensure the dataset is clean, well-structured, and temporally consistent. This is an essential foundation for accurate time-series analysis. Finally, the cleaned data will be exported to CSV format and ingested into Apache Spark [10] as distributed DataFrames for scalable preprocessing and integration with additional data modalities.

**4.2.2 News Data**

To enrich the temporal context surrounding stock movements, the project will collect financial news headlines related to Tesla and Apple using publicly available libraries that provide access to financial news sources, such as Yahoo Finance [24]. The data collection process will include filters based on ticker symbols ("TSLA", "AAPL") and finance-related keywords to minimize irrelevant content and ensure dataset relevance. Headlines and their associated timestamps will serve as the core features for downstream sentiment analysis. While preliminary keyword-based filtering may be applied at this stage, more sophisticated filtering strategies such as Named Entity Recognition and domain-specific lexicons are described in Section 4.3.2. The preprocessed data will be stored in CSV format and loaded into Apache Spark [10] as DataFrames to enable scalable processing and alignment with corresponding stock price events.

**4.2.3 Twitter Data**

The project will also integrate sentiment signals from social media by collecting Twitter data referencing Tesla and Apple. While the official Twitter API was initially considered, its rate limits and restricted historical access make it impractical for large-scale collection. Instead, the project will utilize twscrape [23], an open-source tool for retrieving historical tweets without API limitations.

Tweets will be retrieved using queries that include stock tickers, relevant hashtags, and general finance-related keywords. Basic filtering will be performed at this stage to exclude irrelevant or non-English content. More advanced filtering strategies such as the use of a curated keyword list and Named Entity Recognition are described in Section 4.3.2.

Metadata such as timestamp, tweet ID, retweet count, and hashtags will be preserved. Preprocessing will include mention and URL removal, lowercasing, and basic tokenization. The cleaned tweets will be saved in CSV format and loaded into Apache Spark [10] to facilitate scalable processing and synchronization with corresponding stock price data.

**4.2.4 Dataset Selection Criteria**

The selection of Tesla (TSLA) and Apple (AAPL) as target companies for this project is based on a combination of practical and analytical considerations. Both firms are among the most actively traded stocks in the U.S. market and have substantial media exposure and social media presence, ensuring a sufficient volume of both news articles and tweets. Their price movements are often influenced by a mix of macroeconomic events, corporate actions, and public sentiment, making them suitable case studies for evaluating multimodal prediction models. Additionally, their inclusion in major indices such as the S&P 500 and NASDAQ-100 ensures availability of high-quality historical stock data. These factors make TSLA and AAPL ideal candidates for testing the performance and generalizability of sentiment-driven financial forecasting models.

**4.2.5 Data Synchronization and Temporal Alignment**

In multimodal modeling, temporal synchronization of heterogeneous data streams is a critical challenge. While stock prices are available in structured, time-indexed formats with daily resolution, unstructured data such as news headlines and tweets are produced asynchronously and at irregular intervals. To align these modalities effectively, the project adopts a fixed-window aggregation strategy. Specifically, sentiment scores derived from news and Twitter content are aggregated within sliding windows of 72 hours preceding each stock price observation.

Each headline or tweet is assigned a Polarity Score using pre-trained sentiment analysis models (FinBERT for news, RoBERTa for tweets). These individual scores are then grouped by hour, producing 72 hourly sentiment values for each 72-hour window. Within each hour, the average sentiment score is calculated, with the option to include the standard deviation as an additional feature. This creates a time-ordered sequence that captures the evolution of public and media sentiment over the three days prior to each price point.

For modeling stock prices, a 60-day input window is used. This duration aligns with common financial practice, as it approximately spans one fiscal quarter and allows models to capture both short-term momentum and multi-week structural trends. The window length has also been shown to offer practical effectiveness in recent LSTM-based forecasting studies [8].

This asymmetric temporal framing, short-term sentiment vs. mid-term pricing, is intentionally designed to exploit the complementary dynamics of public opinion and market behavior.

**4.3 Sentiment Feature Extraction**

Sentiment extraction refers to the process of identifying and quantifying the emotional tone expressed in text, typically along a positive–neutral–negative spectrum. In financial contexts, public sentiment captured through news headlines and social media can provide valuable insights into investor psychology, market expectations, and reactions to corporate events. This project incorporates sentiment features as a complementary modality to numerical price data, under the hypothesis that short-term shifts in sentiment may precede or co-occur with market movements.

To model sentiment dynamics, the system processes unstructured textual data from news and Twitter and transforms it into structured numerical representations that can be input into machine learning models. The sentiment extraction pipeline is applied separately to each text stream and consists of several sub-stages, including text preprocessing, keyword and entity filtering, sentiment scoring using transformer-based models, temporal aggregation, and sequence encoding via LSTM networks.

**4.3.1 Text Preprocessing**

Text data is cleaned through a standardized preprocessing pipeline implemented with the spaCy NLP library [22]. This includes lowercasing all text, removing punctuation, URLs, and extra whitespace, and applying tokenization using spaCy’s built-in language model. For tweets, additional preprocessing steps remove user mentions and hashtags, which often introduce noise without contributing to sentiment interpretation.

Language filtering is also performed using spaCy’s language detection utilities to retain only English-language entries, ensuring consistency across all textual data sources.

**4.3.2 Keyword and Entity Filtering**

To ensure relevance, text entries are filtered using a hybrid approach. A manually curated keyword list, supplemented with the SentiBigNomics lexicon [21] is used alongside Named Entity Recognition (NER) via spaCy. This step ensures that only text referring to financial concepts or the target companies (Tesla and Apple) is retained for sentiment scoring.

**4.3.3 Sentiment Scoring**

Pre-trained transformer models are used to compute sentiment polarity for each text instance: FinBERT [13] for news and RoBERTa (cardiffnlp/twitter-roberta-base-sentiment) [11] for tweets. The resulting probability distributions over sentiment classes are converted into scalar polarity scores using the formula:

This formula calculates the sentiment polarity score by subtracting the probability of negative sentiment from the probability of positive sentiment. The resulting scalar ranges from –1 (strongly negative) to +1 (strongly positive), with values near 0 indicating neutral sentiment. These scores enable consistent sentiment representation across both news and tweet texts, facilitating their integration in downstream predictive models.

**4.3.4 Temporal Aggregation**

For each modality, polarity scores are grouped into 72 hourly intervals prior to each stock price observation. Within each hour, the mean of the sentiment scores are computed, forming a 72-length sequence that captures sentiment dynamics over three days.

**4.4 Feature Engineering**

This stage focuses on transforming the raw and preprocessed data into structured feature representations that can serve as input to machine learning models. The system incorporates two primary types of features: numerical time-series derived from historical stock data, and sentiment-based sequences extracted from textual sources. Feature extraction is performed separately for each modality using specialized pipelines tailored to the structure of the input data.

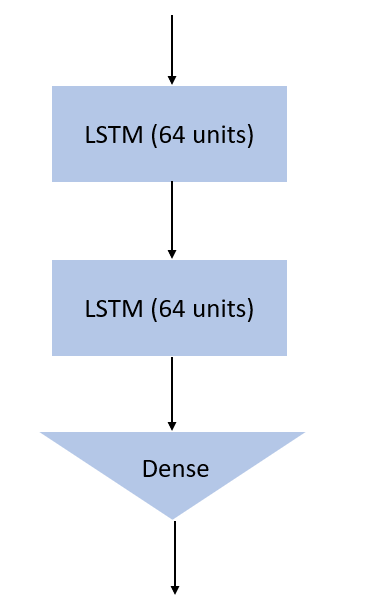
**4.4.1 Model Selection Rationale**

Modeling financial time-series data requires architectures that can effectively capture sequential patterns, temporal dependencies, and dynamic shifts in the underlying data distribution. Recurrent Neural Networks (RNNs) are well-suited for such tasks due to their ability to process input sequences of variable length while retaining information across time steps. Unlike feedforward networks, RNNs maintain internal hidden states, allowing them to learn from temporal structures inherent in financial signals such as price trends, momentum shifts, and sentiment changes.

Among RNN variants, the decision to adopt Long Short-Term Memory (LSTM) networks instead of Gated Recurrent Units (GRUs) is based on both theoretical properties and empirical evidence. LSTM networks offer improved capabilities in capturing long-term temporal dependencies, which are critical in modeling financial time-series data characterized by delayed effects and volatility. Their gated architecture provides enhanced memory control, making them more robust to noisy and non-stationary input patterns.

Empirical studies have shown that LSTM models tend to outperform GRUs in stock market prediction tasks, especially when the input sequences incorporate technical indicators. In particular, one study demonstrated that LSTM models achieved lower RMSE and MAE scores compared to GRUs across multiple stock datasets and configurations, highlighting the advantage of LSTM's gating mechanism in capturing temporal dependencies [5]. Another benchmark comparison focusing on high-volatility stocks such as Tesla, Ferrari, and Walmart confirmed that LSTM models offered better generalization across different market conditions [9].

Based on these findings, LSTM was selected as the primary architecture for encoding both historical stock sequences and sentiment trajectories in this project. The model architecture consists of two stacked LSTM layers, each with 64 hidden units, followed by a fully connected Dense layer that projects the final hidden state into a 128-dimensional feature vector. This configuration is applied uniformly across all sequence encoding modules, including stock price sequences and sentiment sequences derived from news and Twitter. The shared design ensures consistency in the latent representation of different modalities, facilitating their integration in downstream prediction layers. While the architecture remains consistent, each module is trained with a distinct set of hyperparameters tailored to the statistical characteristics of its respective input. The diagram below illustrates the general structure of this recurrent encoder module.



*Figure 1: Shared LSTM encoder with two 64-unit layers and a 128-dim Dense output*

**4.4.2 Stock Features**

The input for this stage consists of the preprocessed historical stock data described in Section 4.2.1, which includes the columns Open, High, Low, Close, Volume, and Adjusted Close. Before sequence modeling, all numerical features are normalized to the [0,1] range using the MinMaxScaler from scikit-learn [16]. Normalization is essential because raw financial indicators vary widely in magnitude, stock prices may reach hundreds of dollars while trading volumes span millions. Without scaling, larger-valued features could dominate the learning process and impair the convergence of gradient-based optimization in neural networks.

Once normalized, the dataset is segmented into fixed-length sequences using a 60-day sliding window. Each sequence captures two months of trading activity, providing sufficient context to model recent market dynamics and volatility patterns.

These sequences are fed into a two-layer LSTM network with 64 units per layer, implemented in PyTorch [19]. The model outputs a 128-dimensional feature vector derived from the final hidden state, encoding latent temporal dependencies in the stock’s historical behavior. This vector serves as the stock-specific representation for multimodal fusion.

The following hyperparameters were selected based on common practice and empirical validation:

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Justification** |
| Sequence window | 60 trading days | Reflects quarterly market cycles |
| Epochs | 50 | Allows convergence on mid-sized datasets without overfitting |
| Batch size | 32 | Standard choice; balances convergence speed with stability |
| Learning rate | 0.001 (Adam) | Default value for Adam; adaptive and efficient optimizer |
| Dropout | 0.2 | Prevents overfitting in recurrent layers |
| Early stopping | Patience = 5 | Stops training early if no improvement in validation loss |

*Table 1: LSTM hyperparameters for stock feature extraction*

**4.4.3 Sentiment LSTM Encoding**

Following temporal aggregation, the sentiment sequences, each consisting of 72 hourly-aggregated polarity scores, are processed using separate LSTM encoders for the news and Twitter modalities. These encoders share a common architecture: two stacked LSTM layers with 64 units each, followed by a dense projection layer that outputs a 128-dimensional feature vector.

The choice of a 72-hour input window reflects the short-term and highly reactive nature of public sentiment. Despite differences in source, both encoders operate on the same temporal resolution and output format, enabling seamless integration of both sentiment sources in the subsequent layers of the prediction architecture.

The hyperparameters for both sentiment LSTM models were tuned to reflect the smaller size and higher noise level of text-derived features, and are summarized below:

|  |  |  |
| --- | --- | --- |
| **Hyperparameter** | **Value** | **Justification** |
| Sequence window | 72 hours | Captures short-term sentiment dynamics in the days preceding stock price movements |
| Epochs | 30 | Sufficient for convergence given smaller and noisier datasets |
| Batch size | 16 | Lower batch size to improve generalization on limited textual input |
| Learning rate | 0.001 (Adam) | Adaptive optimizer with proven convergence in NLP and time-series tasks |
| Dropout | 0.2 | Regularization to prevent overfitting in recurrent layers |
| Early stopping | Patience = 3 | Stops training when validation performance plateaus |

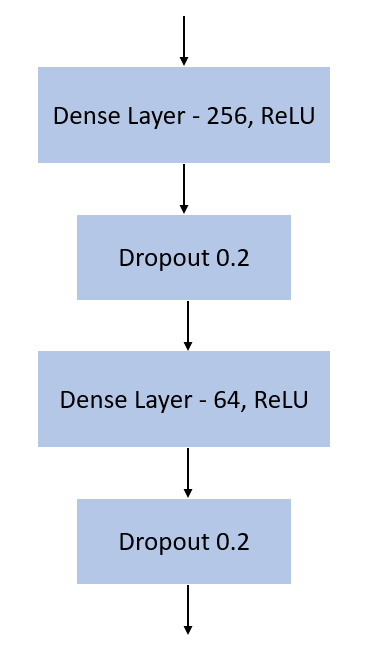
*Table 2: LSTM hyperparameters for sentiment feature extraction (news and Twitter)*

**4.5 Multimodal Fusion**

The multimodal fusion stage integrates the independently extracted features from historical stock prices, financial news sentiment, and Twitter sentiment to form a unified representation suitable for prediction. Each modality contributes a 128-dimensional feature vector, generated by its respective LSTM encoder described in Sections 4.4.2 and 4.4.3. These vectors are concatenated to form a single 384-dimensional composite vector that encapsulates long-term market trends and short-term public sentiment signals.

At this stage, feature fusion is performed using a simple concatenation of the three modality-specific vectors, without applying any weighting or attention mechanisms. This design choice enables a straightforward baseline for integration. In future iterations, we plan to experiment with alternative fusion strategies, such as weighted averaging, gated fusion, or attention-based mechanisms to evaluate their impact on predictive performance.

To reduce dimensionality and model cross-modal interactions, the fused vector is passed through a sequence of fully connected layers: a 256-unit dense layer followed by a 64-unit dense layer, both using ReLU activation functions. Dropout layers with a rate of 0.2 are applied after each dense layer to mitigate overfitting and enhance generalization. The output is a 64-dimensional fused embedding vector that encodes the combined temporal and semantic information from all three input streams, serving as the final representation for downstream classification.



*Figure 2: Multimodal fusion architecture*

**4.6 Prediction Module**

The final stage of the modeling pipeline translates the fused feature representation into a binary prediction indicating the expected direction of stock price movement. This stage is composed of two sub-layers: a refinement layer and a classification layer.

**4.6.1 Feature Refinement Layer**

The 64-dimensional multimodal embedding vector obtained from the fusion stage is passed into a dense (fully connected) layer with 32 units and ReLU activation. This transformation serves to compress and refine the integrated features, enhancing discriminative power and removing any residual noise from the fusion process.

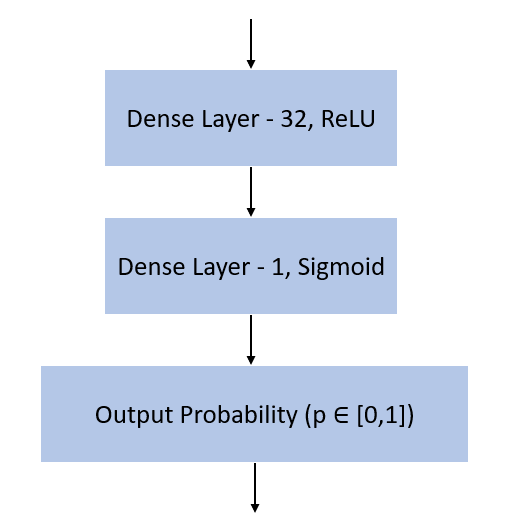
**4.6.2 Output Layer and Decision Rule**

The refined 32-dimensional vector is fed into a final dense layer with one neuron and a sigmoid activation function. This outputs a scalar probability p ∈ [0,1] which reflects the model’s confidence that the stock price will increase.

The binary prediction is determined by thresholding this probability:

* If p ≥ 0.5: predict Up
* If p < 0.5: predict Down

This formulation frames the problem as a binary classification task, suitable for daily directional forecasting of stock prices.



*Figure 3: Final classification*

**4.6.3 Training Procedure**

The model is trained on a rolling-window dataset constructed from synchronized historical stock prices and sentiment sequences. For each training instance, the input comprises:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Modality** | **Sequence Length** | **Feature Dimensions** | **Shape** | **Description** |
| Stock Prices | 60 days | 6 features | 60×6 | Daily Open, High, Low, Close, Volume, Adjusted Close |
| News Sentiment | 72 hours | 1 features | 72×1 | Hourly aggregated sentiment polarity scores |
| Twitter Sentiment | 72 hours | 1 features | 72×1 | Hourly aggregated sentiment polarity scores |

*Table 3: Input structure for each training instance*

For each instance, a 60-day sequence of stock prices is collected up to the prediction date, while sentiment sequences are aggregated over the preceding 72 hours. Each input stream is independently processed through a dedicated LSTM encoder, producing a 128-dimensional latent vector per modality. These vectors are concatenated to form a unified 384-dimensional representation, which is passed through fully connected layers for further refinement and ultimately produces a binary classification output: Up or Down, indicating the expected direction of the stock price movement.

During training, the model's prediction is compared to the true label using binary cross-entropy loss. Optimization is performed using the Adam optimizer with a learning rate of 0.001. The training process is conducted in mini-batches of size 32, with each epoch consisting of all training instances being seen once. To prevent overfitting, dropout is applied with a rate of 0.2 in the dense layers, and early stopping is triggered if the validation loss does not improve for 5 consecutive epochs. A total of 50 epochs is set as the maximum, although training may terminate earlier if convergence is reached.

**4.6.4 Loss Function and Optimization**

During training, the model is optimized using the binary cross-entropy loss function, which is appropriate for probabilistic binary classification:

where y ∈ {0,1} is the true label and p is the predicted probability. Optimization is performed using the Adam optimizer with a learning rate of 0.001.

**4.6.5 Classification Metrics**

To assess the model’s performance, standard classification metrics are used. These metrics are computed on a held-out test set that was not seen during training, ensuring an unbiased evaluation of generalization performance.

Let:

* TP (True Positive) – the model correctly predicts an upward movement.
* TN (True Negative) – the model correctly predicts a downward movement.
* FP (False Positive) – the model predicts an upward movement, but the stock goes down.
* FN (False Negative) – the model predicts a downward movement, but the stock goes up.

The following metrics are used:

* Accuracy: Proportion of correct predictions.
* Precision and Recall: Useful in measuring class-specific performance, especially under imbalance.
* F1-Score: Harmonic mean of precision and recall.
* ROC-AUC: Evaluates the model’s ability to rank predictions correctly across thresholds.

**4.7 Tooling and Environment**

This section outlines the technical tools, programming frameworks, and computational infrastructure that will be used throughout the project, from data collection and preprocessing to model training and evaluation.

**4.7.1 Languages & Frameworks**

The project will be implemented in Python [18], chosen for its extensive ecosystem of data science libraries and ease of integration with machine learning frameworks. All code will be developed and executed within Jupyter Notebook environments [15], which provide an interactive and modular workflow ideal for experimentation and documentation.

Key frameworks and libraries include:

|  |  |
| --- | --- |
| **Library / Tool** | **Purpose** |
| Apache Spark | Scalable processing and alignment of large sentiment and stock datasets [10] |
| Jupyter Notebook | Interactive environment for developing, testing, and documenting code [15] |
| pandas | Data manipulation and preprocessing for structured stock and sentiment data [17] |
| PyTorch | Deep learning framework for implementing LSTM encoders and fusion/prediction layers [19] |
| scikit‑learn | Preprocessing tasks such as normalization and evaluation metric [20] |
| spaCy | NLP preprocessing: tokenization, stopword removal, named entity recognition [22] |
| Transformers (Hugging Face) | Pre-trained transformer models for sentiment scoring (FinBERT, RoBERTa) [14] |
| twscrape | Twitter scraping tool without API restrictions [23] |
| yfinance | Retrieval of historical stock data from Yahoo Finance [25] |

*Table 4: Core libraries and frameworks*

**4.7.2 Selection of Data Storage and Processing Framework**

To manage and process the multimodal datasets collected for this project, Apache Spark [10] will be used as the primary data handling framework. Spark was selected over traditional storage and processing solutions due to its superior scalability, speed, and distributed architecture. Given the substantial volume and variety of data, Spark’s in-memory computation and parallel processing capabilities make it particularly effective for aligning and transforming time-sensitive data across modalities.

Unlike traditional single-machine processing tools, Spark can efficiently manage the integration of stock, news, and Twitter datasets while preserving chronological alignment. Its support for DataFrames and compatibility with Python via PySpark make it seamlessly integrable with the rest of the project’s machine learning pipeline.

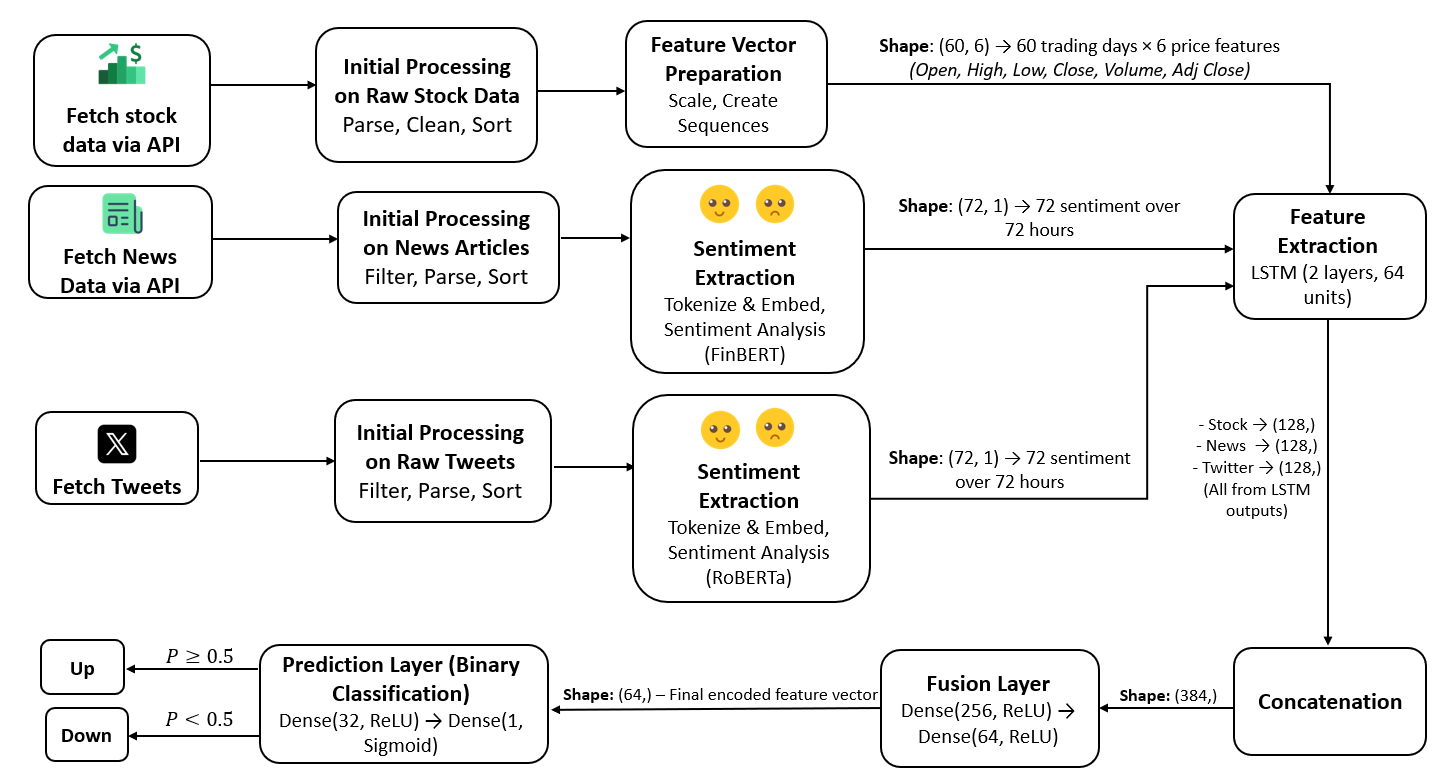
To validate this choice, a comparative analysis was conducted between Apache Spark and MongoDB [16], the latter being a common alternative for handling unstructured data. The key differences in scalability, data processing capabilities, and suitability for time-indexed, multimodal fusion are summarized in Table 5. Based on this comparison, we concluded that Apache Spark is better aligned with the computational and temporal demands of the project.

|  |  |  |
| --- | --- | --- |
| **Criterion** | **Apache Spark** | **MongoDB** |
| Processing Model | Distributed, in-memory batch/stream | NoSQL document store |
| Scalability | High (horizontal via clusters) | High (but less optimized for analytics) |
| Data Handling | Ideal for large-scale analytics | Optimized for fast read/write of records |
| Chronological Alignment | Supports time-indexed transformations | Manual handling required |
| Integration with ML Pipelines | Strong via PySpark | Requires external tools/connectors |
| Suitability for Sentiment Fusion | Excellent for joining multi-source temporal data | Limited, not designed for time-series fusion |

*Table 5: Comparison Between Apache Spark and MongoDB*

**5 Product**

**5.1 Architecture overview**

****

*Figure 4: Architecture*

Figure 5 illustrates the overall system architecture, integrating all key components described in Sections 4.2 through 4.6. The pipeline begins with the retrieval and preprocessing of raw data from three sources: historical stock prices, financial news articles, and tweets (see Section 4.2).

Each text source then undergoes sentiment extraction using pre-trained transformer models: FinBERT for news and RoBERTa for tweets, resulting in polarity scores over time (see Section 4.3).

Following sentiment extraction and feature preparation, the three input streams, stock prices, news sentiment, and Twitter sentiment, are each processed by an independent LSTM encoder, which learns to capture the temporal structure of the data (see Section 4.4).

The resulting representations are combined into a single 384 dimensional vector (see Section 4.5), refined through fully connected layers, and then passed into a final prediction module that outputs a binary classification of the expected stock price movement (see Section 4.6).

**5.2 Requirements**

**5.2.1 Functional Requirements**

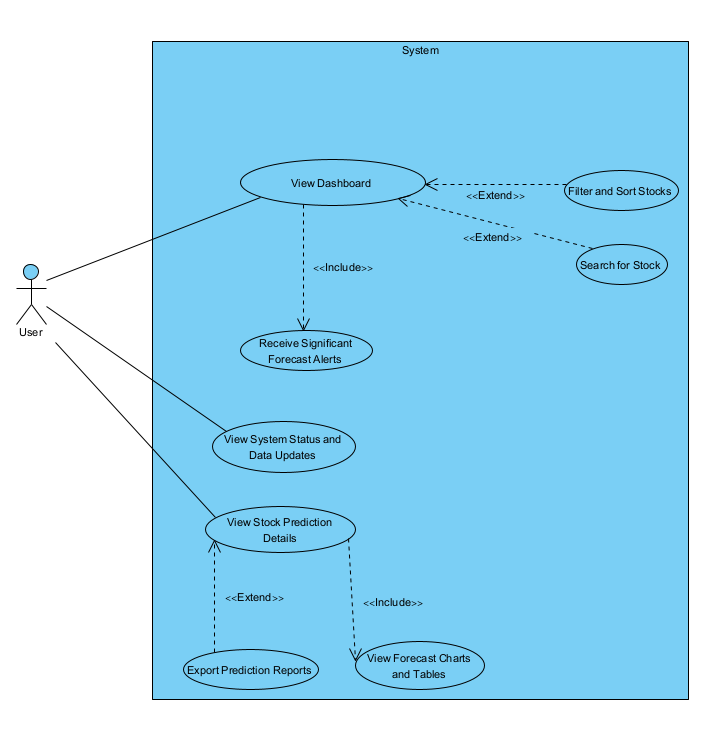
|  |  |
| --- | --- |
| **No.** | **Requirement** |
| 1 | The system shall display a dashboard listing target stocks with predicted movement direction (Up/Down). |
| 2 | The system shall update stock forecasts automatically in the background without user input. |
| 3 | The system shall retrieve and process financial data automatically from external sources. |
| 4 | The system shall persist processed data in structured formats suitable for analysis, audit, and re-use. |
| 5 | The system shall perform sentiment analysis on both news headlines and tweets, generating temporally aligned sentiment scores. |
| 6 | The system shall use machine learning models to generate stock movement predictions. |
| 7 | The system shall highlight stocks with significant forecast changes. |
| 8 | The system shall display system status and data update progress indicators. |
| 9 | The system shall handle errors gracefully and inform the user of data retrieval or processing issues. |
| 10 | The system shall allow users to search for specific stocks by ticker or company name. |
| 11 | The system shall provide filtering and sorting options by sector, industry, prediction direction, and confidence level. |
| 12 | The system shall present interactive visualizations and tables for each stock, including visual charts of price history and forecast trends, as well as tables showing forecast direction, confidence score, and supporting sentiment data. |
| 13 | The system shall allow exporting prediction results and system logs in formats such as CSV or PDF. |

*Table 6: Functional requirements*

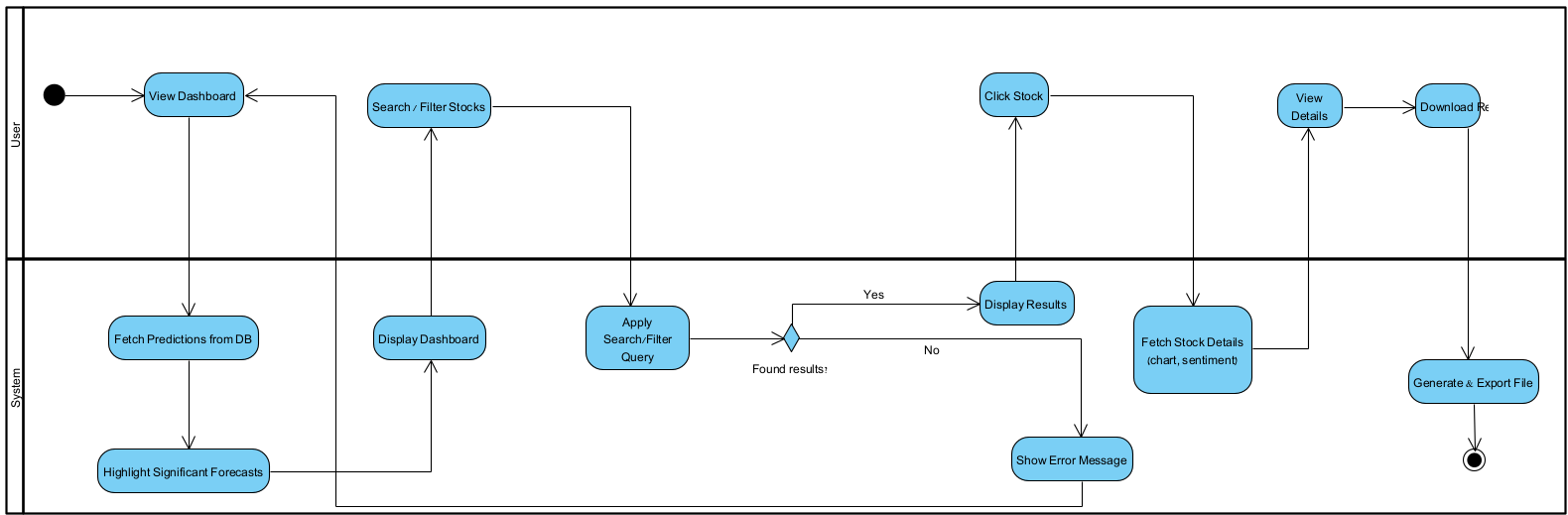
**5.2.2 Non-functional Requirements**

|  |  |  |
| --- | --- | --- |
| **No.** | **Requirement** | **Type** |
| 1 | The system should provide prediction results with minimal latency to ensure a responsive user experience. | Performance |
| 2 | The system must handle large volumes of historical data and real-time updates without performance degradation. | Scalability |
| 3 | The system should maintain high accuracy in stock movement predictions to minimize false positives and negatives. | Accuracy |
| 4 | The system should provide an intuitive and user-friendly interface for non-expert users. | Usability |
| 5 | The system should be reliable and available, with minimal downtime during data updates. | Reliability |
| 6 | The system should implement robust error handling and recovery mechanisms for data retrieval and processing failures. | Reliability |
| 7 | The system should support multiple devices and browsers for accessing the dashboard. | Compatibility |
| 8 | The system should be easy to maintain and update, with clear modular architecture and documentation. | Maintainability |
| 9 | The system shall optimize data inputs to match model requirements and improve prediction accuracy. | Optimization |
| 10 | The system shall employ LSTM-based deep learning models and transformer-based sentiment scoring to generate high-quality forecasts. | Technology |
| 11 | The system should integrate seamlessly with external data providers via APIs. | Integration |
| 12 | The system should provide consistent prediction quality across varying market conditions and data volatility. | Robustness |
| 13 | The system should perform background updates and data processing without interrupting the user interface. | Performance |
| 14 | The system should be scalable to support additional stocks, data sources, or analytical modules in the future. | Scalability |
| 15 | The system should comply with relevant data privacy and financial industry regulations. | Compliance |

*Table 7: Non-functional requirements*

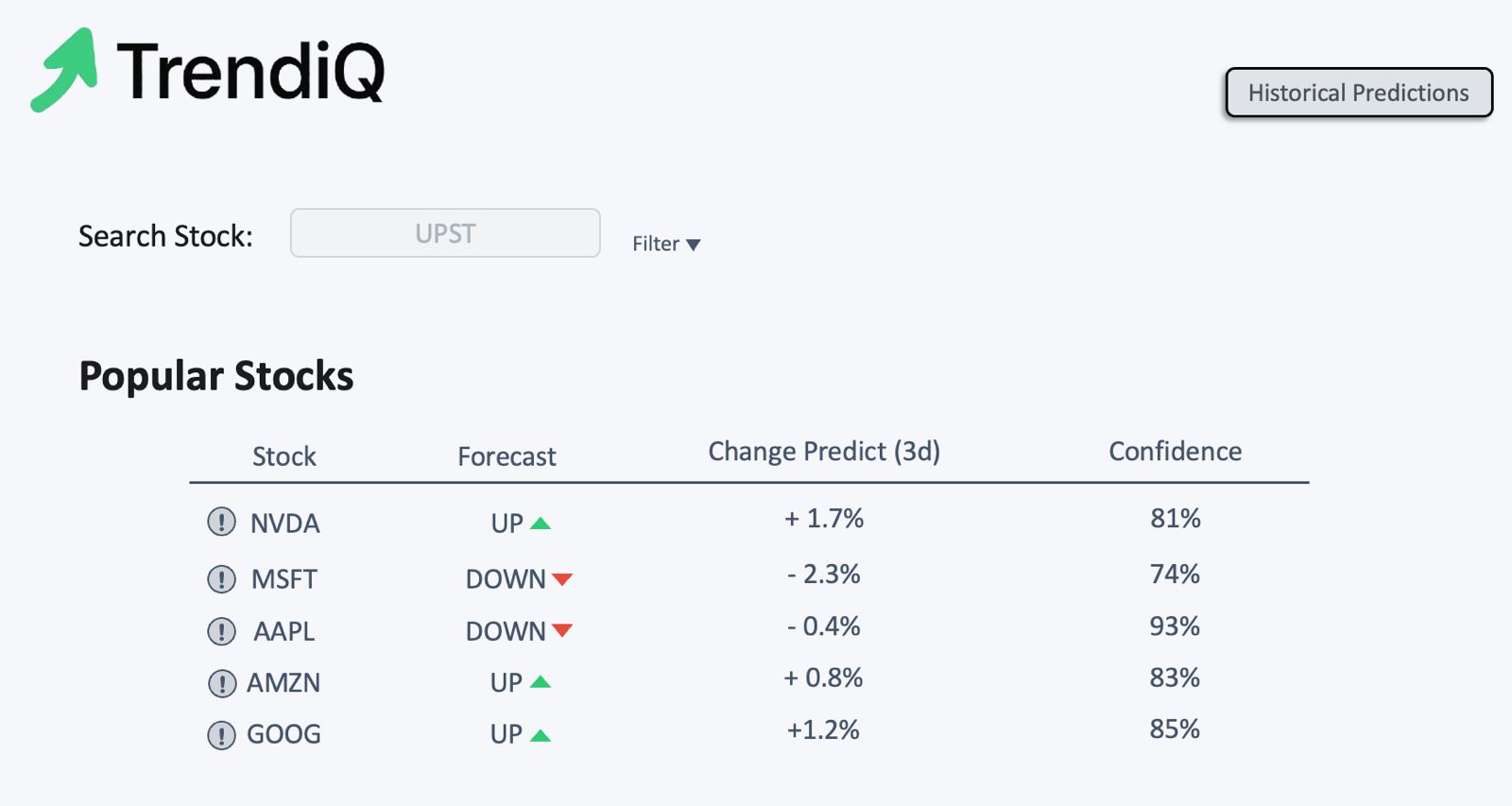
**5.3 System Use Case**

*Figure 5: System Use Case Diagram*

**5.4 System Activity**

*Figure 6: System Activity Diagram*

**5.5 Screens**

*****Figure 7: Screen 1, Dashboard*

*****Figure 8: Screen 2, Stock Data*

**6 Evaluation Plan**

**6.1 Scope**

This testing plan covers the validation of all major components in the prediction pipeline, including data acquisition, preprocessing, feature extraction, model training, prediction output, and system integration. Testing also includes performance evaluation of models and consistency across input sources (news, social media, historical data).

**6.2 Objectives**

* Verify the functional accuracy of each module: data ingestion, sentiment scoring and model outputs.
* Assess the prediction accuracy of the models (LSTM).
* Validate the fusion mechanism and the final prediction layer.
* Ensure robustness across different time periods and market conditions.
* Evaluate system latency and scalability with large datasets.

**6.3 Test Cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test Case** | **Description** | **Procedure** | **Expected Result** |
| 1 | Full\_Pipeline\_Run | Run full prediction pipeline with valid, clean data | 1. Load complete datasets (stock, news, Twitter)   |  | | --- | | 2. Run full pipeline from preprocessing to prediction |  |  | | --- | |  | | System runs end-to-end without errors; outputs valid forecast. Stores the prediction output in the designated format   |  | | --- | |  | |
| 2 | Multi\_Source\_Alignment | Verify time alignment across stock/news/tweets | 1.  Use synchronized inputs with known time stamps  2. Run feature extraction  3. Inspect aligned feature vectors | Features across all modalities align properly; no shift or data loss |
| 3 | Temporal\_Gap\_Scenario | Simulate missing days | 1. Remove multiple consecutive days from stock/news  2. Run pipeline  3. Observe fusion and sequence handling | Model handles gaps with interpolation, padding, or skips safely |
| 4 | Contradictory\_Sentiment | Test conflicting news and tweets | 1. Use positive tweet and negative news on same date  2.  Extract sentiment  3. Run Model | Fusion handles contradiction without instability; output is still coherent |
| 5 | Missing\_Stock\_Values | Test with missing Open/Close/Volume | 1. Create stock CSV with missing values  2. Run preprocessing  3. Run prediction | Missing values handled via imputation or flagged; model does not fail |
| 6 | Outlier\_Price\_Day | Input extreme spike/drop in stock price | 1. Inject 100x change in a day  2. Extract features  3. Run model | Model remains stable; predictions do not overreact |
| 7 | Empty\_Tweet\_Text | Test tweet with no content | 1. Insert tweet with “”  2. Run sentiment pipeline  3. Continue to prediction | Tweet is ignored or marked neutral; no influence on results |
| 8 | Non\_English\_Tweet | Insert tweet in unsupported language | 1. Use tweet in Hebrew  2. Run NLP module  3. Pass to sentiment layer | Handled safely; either skipped, labeled uncertain, or not included in features |
| 9 | Unseen\_Stock\_Symbol | Use stock not in training set | 1. Input data for new stock  2. Run full pipeline | System flags the input as out-of-scope or unknown |
| 10 | Error\_Handling\_BadInput | Input corrupted CSV / invalid tweet format | 1. Load malformed file  2. Start pipeline | System detects and reports input error |
| 11 | Negative\_Stock\_Price | Simulate a bug with negative price | 1. Insert -100 for “Close”  2. Run full pipeline | System flags error or corrects without failure |
| 12 | Ambiguous\_Sentiment | Tweet like “That was unexpectedly fine I guess" | 1. Run through sentiment engine  2. Check fusion | Marked neutral, model behavior remains stable |

*Table 8: Test cases*

**7 References**

[1] Boroumand, O., & Doaei, M. (2024). Developing a stock market prediction model by deep learning algorithms. Journal of Information Technology Management, 16(3), 115–131.

[2] Gupta, H., & Jaiswal, A. (2024). A study on stock forecasting using deep learning and statistical models. arXiv preprint, arXiv:2402.06689.

[3] Mukherjee, S., Sadhukhan, B., Sarkar, N., Roy, D., & De, S. (2021). Stock market prediction using deep learning algorithms. CAAI Transactions on Intelligence Technology, 8, 82–94.

[4] Ouf, S., El Hawary, M., Aboutabl, A., & Adel, S. (2024). A deep learning-based LSTM for stock price prediction using Twitter sentiment analysis. International Journal of Advanced Computer Science and Applications, 15(12), 207–218.

[5] Patel, V., & Tandel, S. (2025). A comparative analysis of LSTM and GRU models enhanced with technical indicators for stock forecasting. ResearchGate.

[6] Sebastian, A., & Tantia, V. (2024). Deep learning for stock price prediction and portfolio optimization. International Journal of Advanced Computer Science and Applications, 15(9), 926–934.

[7] Selvin, A., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In International Conference on Advances in Computing, Communications and Informatics (ICACCI) (pp. 1643–1647).

[8] Tarek, M. A., Osman, H., & Elgohary, A. F. M. (2024). Integrating LSTM and transformer-based sentiment for stock price forecasting. Journal of Economic Analysis, 4(3), Article 109.

[9] Wan, X. (2023). Stock price prediction of high-tech industry based on LSTM and GRU: Tesla, Ferrari, and Walmart. Highlights in Science, Engineering and Technology, 76, 585–590.

[10] Apache Spark. (n.d.). Apache Spark. <https://spark.apache.org/>

[11] CardiffNLP. (n.d.). Twitter RoBERTa – Sentiment. Hugging Face. <https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment>

[12] ChatGPT. (n.d.). <https://chat.openai.com/>

[13] FinBERT (ProsusAI/finbert). (n.d.). FinBERT. Hugging Face. <https://huggingface.co/ProsusAI/finbert>

[14] Hugging Face. (n.d.). Transformers documentation. <https://huggingface.co/docs/transformers>

[15] Jupyter Notebook. (n.d.). <https://jupyter.org/>

[16] MongoDB Documentation. (n.d.). MongoDB Manual. <https://www.mongodb.com/docs/manual/>

[17] Pandas Documentation. (n.d.). pandas. <https://pandas.pydata.org/>

[18] Python Software Foundation. (n.d.). Python programming language. <https://www.python.org/>

[19] PyTorch Documentation. (n.d.). <https://pytorch.org/docs/>

[20] scikit-learn. (n.d.). Machine learning in Python. <https://scikit-learn.org/>

[21] SentiBigNomics GitHub. (n.d.). <https://github.com/consose/SentiBigNomics>

[22] spaCy. (n.d.). Industrial-strength NLP. <https://spacy.io/>

[23] twscrape GitHub. (n.d.). <https://github.com/vladkens/twscrape>

[24] Yahoo Finance. (n.d.). <https://finance.yahoo.com/>

[25] yfinance GitHub. (n.d.). <https://github.com/ranaroussi/yfinance>

**8 AI Prompts**

Throughout the development of this project, ChatGPT [4] was employed as a supportive tool to enhance productivity, validate technical decisions, assist in drafting and editing text, and accelerate the research process. The model proved especially helpful during early ideation phases, literature review generation, and in refining the language and structure of technical content. Below are several prompts that were used during different stages of the project.

* Explain Limitations of Sentiment-Based Financial Forecasting  
  <https://chatgpt.com/share/688a2dfc-eea0-8007-99ee-a6dc58de2dc0>
* Retrieve academic articles on stock prediction using sentiment analysis  
  <https://chatgpt.com/share/6868331d-199c-8007-9d48-eee4cbc4806a>
* Generate a structured summary for the paper “A Deep Learning-Based LSTM for Stock Price Prediction Using Twitter Sentiment Analysis”  
  <https://chatgpt.com/share/68683252-d218-8007-b9cd-6a9131605578>
* Compare deep learning models used in stock forecasting  
  <https://chatgpt.com/share/6868313f-0848-8007-b277-1743a5cfd2a8>
* Extract evaluation metrics and explain them clearly  
  <https://chatgpt.com/share/686830b4-5ffc-8007-b301-ad04082f3370>
* Pros and Cons of Using Pre-Trained Transformers in Financial Sentiment Analysis  
  <https://chatgpt.com/share/688a2ec7-6d60-8007-995b-e32ea1bd77dc>