

Report on

Sentiment analysis for Stock Prediction:

Sentimetrics AI



Submitted to

Department of Computer Science and Engineering

Nepal Engineering College

in Partial Fulfillment of the

Requirements for the Degree of B.E. in Computer

Submitted By

Sulav Timalina (022-380)

Sushant Poudel (022-387)

Sushil Upadhyay (022-382)

Supervisor

Asst. Prof. Krishna Bikram Shah

Date: 30/07/2025

Abstract

Sentimetrics AI is a state-of-the-art stock price prediction system for the Nepal Stock Exchange (NEPSE). The system combines historical stock price information with real-time news article sentiment analysis of Nepalese press to improve price predictions of stocks listed on NEPSE. Business intelligence is drawn from historical stock basics, media sentiment trends, and the ability to identify predicted price highs and lows, as predicted by the system user interface, which utilizes machine learning, natural language processing (NLP), and data processing techniques. The web-dashboard was designed as an open-access tool for anyone, technical and non-technical, to visually interpret stock price predictions, trends in media sentiment over time, and media reliability scores. This report describes the design, implementation and evaluation of the Sentimetrics system, with its potential contributions to advanced financial analytics in emerging markets. The report also highlights the limitations and future improvements to the system, including the area of larger data modeling and more real time and historical stock data.

[Keywords: Sentiment Analysis, Stock Price Prediction, Natural Language Processing (NLP), Machine Learning, Financial Market, Sentimetrics AI, Time Series Analysis, Forecasting, ShareHub (API)]

Acknowledgement

We would like to sincerely thank ShareHub for providing the wonderful API, support and knowledge that helped this project reach its ultimate goal. We also want to give special thanks to Asst. Prof. Krishna Bikram Shrestha for the continued expert advice and support during the duration of this project. We would also like to thank Nepal Engineering College, as well as the support and mentorship from the Nepal Engineering College which assisted us in laying the groundwork for this project. We are sincerely grateful for the support from our colleagues and family who provided feedback and support from the project inception to completion, which also motivated me to fulfil all the larger objectives of this project. A special thanks also goes to the makers of amazing open-source tools and libraries such as Python, NLTK and Chart.js, which made the development of Sentimetrics AI possible. Their genuine contributions are key to modern-day innovation, and are appreciated.

Table of Content

Abstract	i
Acknowledgement	ii
Table of Content	iii
List of Figures	v
List of Tables	vi
Abbreviations	vii
Chapter 1 : Introduction	1
1.1 Overview	1
1.2 Problem Statement	1
1.3 Objectives	1
1.4 Aims	1
1.5 Motivation	1
1.6 Scope and Applications	2
1.7 Feasibility Study	2
Chapter 2 : Literature Review	3
2.1 Traditional Stock Price Prediction Models	3
2.2 Relevance of Sentiment Analysis in Financial Markets	3
2.3 Influence of Public Sentiment on Stock Prices	4
2.4 Challenges of Interpreting Sentiment in Financial Text	4
2.5 Sentiment Analysis vs. Traditional Financial Indicators	4
2.6 Sentiment Analysis in Financial Markets	5
2.7 Machine Learning in Stock Price Prediction	5
2.8 Multilingual Sentiment Analysis	6
2.9 NEPSE-Specific Studies	6
2.10 Relevance to Sentimetrics AI	7
2.11 Gaps and Contributions	7
2.12 Conclusion	8
Chapter 3 : System Design	9

3.1	System Architecture	9
3.2	Data Flow Diagram	9
3.3	Data Source	11
3.4	Hardware and Software Requirements.....	11
Chapter 4 : Methodology		13
4.1	Workflow:	13
4.1.1	News Collection and Classification (classified_news.py)	13
4.1.2	Sentiment Analysis (sentiment_analysis.py)	16
4.1.3	News Price Impact Analysis (news_price_impact.py)	18
4.1.4	Historical Price Prediction (historical_price_prediction.py)	21
4.1.5	Final Price Prediction (final_price_prediction.py)	23
Chapter 5 : Implementation and Discussion		25
5.1	Project Execution Flow	25
5.2	How Data is Cleaned and Preprocessed.....	25
5.3	Approach to Classify Sentiment.....	25
5.3.1	Formulas	26
5.4	Metrics to Assess Model Performance.....	27
5.5	Output Obtained	28
5.6	Limitations	29
5.6.2	Historical Prediction Risks	29
5.7	Testing/Test Cases.....	30
5.8	Time Schedule.....	30
Chapter 6 : Analysis and Evaluation		31
6.1	Data Analysis	31
6.2	Results	31
6.3	Comparison with Objectives	31
6.4	Discussion of Findings	31
6.5	Conclusion.....	31
References.....		33

List of Figures

Figure 1 Block Diagram of Sentimetrics AI System	9
Figure 2 Data Flow Diagram	10
Figure 3 Sequence Diagram.....	11
Figure 4 Stock Price Prediction Workflow	13
Figure 5 News Collection and Classification Workflow	14
Figure 6 news_share folder.....	15
Figure 7news_share/ADBL_news.csv	15
Figure 8 news_share/ADBL_news.csv	17
Figure 9 sentiment_results/ADBL_share_sentiment.csv	17
Figure 10 sentiment_results/ADBL_share_sentiment.csv	19
Figure 11 share_weightage.csv	19
Figure 12 weightage/media_weightage.json.....	20
Figure 13 history_prediction.csv	22
Figure 14 history_prediction.csv	24
Figure 15 share_prediction.csv	24
Figure 16 Project Execution Workflow	25
Figure 17 Classified News.....	28
Figure 18 Share Sentiment Data	28
Figure 19 Share weightage data.....	28
Figure 20 Media Weightage data	29
Figure 21 Share Predicted Data	29
Figure 22 Gantt Chart for Project Schedule.....	30

List of Tables

Table 1:Test Case 1: System Components.....	30
Table 2 Final Prediction Sample.....	31
Table 3 Media Source Weightage output.....	31

Abbreviations

NEPSE: Nepal Stock Exchange

NLP: Natural Language Processing

API: Application Programming Interface

VADER: Valence Aware Dictionary and sEntiment Reasoner

CSV: Comma-Separated Values

JSON: JavaScript Object Notation

Chapter 1 : Introduction

1.1 Overview

Sentimetrics AI is an innovative system designed to predict stock prices for the Nepal Stock Exchange (NEPSE). By using historical stock price data and real-time news sentiment analysis, this system predicts NEPSE stock prices accurately. It integrates quantitative price trends with qualitative sentiment data, using machine learning and NLP, and provides actionable recommendations to investors and analysts. A web-based user dashboard contains visualizations of predictions, sentiment trends, and media reliability metrics, improving access.

1.2 Problem Statement

Stock predictions in the NEPSE market lack a full-fledged prediction system that incorporates both the past data and sentiment of the market. Most existing prediction systems make use of past trends only and neglect sentiment of news, which can be a significant indicator of a stock price in developing countries. This project examines ways to include both sources of data into a prediction system that meaningfully combines news and past data into an easy output source.

1.3 Objectives

- Create a methodology for fetching and categorizing news articles about NEPSE firms.
- Analyze sentiments of news articles to identify metrics and measure their effects on stock price.
- Create baseline price predictions through historical data and linear regression.
- Adjust predictions by weighing news sentiments and reliability of the media.
- Create an interactive user dashboard to visualize predictions, metrics, and sentiment weights.

1.4 Aims

The goal of the project is to provide a dependable, data-driven method to forecasting NEPSE stock prices that will facilitate decision-making for investors and analysts. This project intends to combine the strengths of historical data analyses and real time sentiment data insights while focusing on an easy-to-use web dashboard.

1.5 Motivation

Emerging markets such as NEPSE are volatile in nature, and access to sophisticated analytical tools is limited. The addition of news sentiment can increase prediction accuracy

with contextual historical data, providing stakeholders a first market advantage. This project aims to provide NEPSE investors increased insights, and to add to the body of research surrounding financial analytics in multilingual settings.

1.6 Scope and Applications

The system focuses on NEPSE listed shares and recognizes news in both English and Nepali. Applications are as follows:

- Helping investors make informed decisions to buy/sell.
- Helping analysts investigate the correlation between sentiment and price.
- Helping portfolio managers assess risk.
- Helping researchers build upon in the area of financial analytics in emerging markets.

1.7 Feasibility Study

The project is viable because of the availability of the ShareHub Nepal API for data, open-source Python libraries (e.g., NLTK, scikit-learn) for processing, and web frameworks for developing dashboards. There are some risks, in particular the reliability of APIs and accuracy of sentiment analysis, to which we will mitigate by utilizing adequate error-handling processes and multilingual support. The project took a total of 6 months, and was feasible based on our current computational resources.

Chapter 2 : Literature Review

Predicting stock prices is a key research area in financial analytics, with implications for investment, risk management, and economic forecasting. In the past few years, there has been growing interest in using historical price data plus external factors, like news sentiment, to make predictions, particularly in many forms of machine learning and natural language processing (NLP). This literature review will examine previous studies on stock price prediction and sentiment analysis and their relevance to emerging markets like the Nepal Stock Exchange (NEPSE). It will focus on the relevance of earlier reports' findings within Sentimetrics AI, the study that integrates historical price data with multilingual news sentiment analysis to predict NEPSE stock prices, and the gaps surplus products are filling as indicated in earlier work.

2.1 Traditional Stock Price Prediction Models

Classical methods of predicting stock prices tend to focus on the time-series analysis of historical price data. In 1970, Box et al. introduced the Autoregressive Integrated Moving Average (ARIMA) model that produces forecasts based on historical price data and trends [1]. ARIMA models are useful for detecting linear patterns in historical price data, but they do not incorporate the non-linear nature of market dynamics or external factors influencing prices, like market news or macroeconomic events [1]. Similarly, Hamilton researched Vector Autoregression (VAR) models, which uses multiple time-series variables like prices and volumes to further refine predictions. However, Hamilton does not assess qualitative factors like market sentiment [2].

Overall, these traditional models have limited predictive capacity in volatile markets like NEPSE, where price fluctuations depend highly on external events like changes in government policy, or important corporate news. Sentimetrics AI is able to overcome these restrictions by supplementing sentiment analysis, capturing qualitative influences in the market with the historical data component, in order to improve prediction accuracy together as a whole.

2.2 Relevance of Sentiment Analysis in Financial Markets

Sentiment analysis is highly relevant in financial markets because it captures the emotional and psychological factors that influence investor behavior and market dynamics [3]. Financial markets are driven not only by quantitative data, such as earnings reports or economic indicators, but also by qualitative factors, such as public perception and media narratives. Sentiment analysis quantifies these qualitative factors by analyzing texts from news articles, social media, and analyst reports to determine their positive, negative, or neutral tone [4]. In emerging markets like NEPSE, where information asymmetry and lower market efficiency amplify the impact of news, sentiment analysis provides critical insights

into short-term price movements [5]. Sentimetrics AI leverages this relevance by using sentiment scores to adjust historical price predictions, enhancing their alignment with market sentiment.

2.3 Influence of Public Sentiment on Stock Prices

Public sentiment, derived from sources like news, social media, and analyst reports, significantly influences stock prices by shaping investor perceptions and market expectations. News articles reporting positive corporate developments, such as strong earnings or strategic partnerships, can boost investor confidence, leading to increased demand and higher stock prices [3]. Conversely, negative news, such as regulatory issues or financial losses, can trigger sell-offs, causing price declines [4]. Social media platforms, like X, amplify these effects by rapidly disseminating opinions and rumors, which can create volatility, especially in smaller markets like NEPSE [6]. Analyst reports, with their recommendations and forecasts, further influence institutional investors, impacting stock liquidity and pricing [7]. Sentimetrics AI incorporates news sentiment to capture these dynamics, adjusting predictions based on the tone of recent articles to reflect public sentiment's impact on NEPSE stocks.

2.4 Challenges of Interpreting Sentiment in Financial Text

Interpreting sentiment in financial text poses several challenges. First, financial texts often contain domain-specific jargon and nuanced language that general-purpose sentiment models, like VADER, may misinterpret [5]. For example, terms like “short” or “leverage” have specific financial meanings that differ from their general usage. Second, the sentiment expressed in financial texts may be ambiguous or context-dependent, requiring careful analysis to distinguish between factual reporting and opinion [4]. Third, multilingual texts, such as those in English and Nepali in the NEPSE context, introduce translation inaccuracies and cultural nuances that complicate sentiment analysis [8]. Finally, the reliability of sources varies, as some media outlets may publish biased or speculative content, necessitating methods to weight source credibility [3]. Sentimetrics AI addresses these challenges through text normalization, translation to English for consistency, and a media reliability weighting system to account for source trustworthiness.

2.5 Sentiment Analysis vs. Traditional Financial Indicators

Sentiment analysis complements traditional financial indicators, such as price-earnings ratios, trading volume, or moving averages, by providing a qualitative perspective that quantitative metrics alone cannot capture [4]. Traditional indicators reflect historical and current market performance but often lag behind real-time market sentiment, particularly in response to breaking news or social media trends [6]. Sentiment analysis, by contrast, offers predictive insights into short-term price movements driven by public perception, making it particularly valuable in volatile or less efficient markets like NEPSE [5].

However, sentiment analysis is less effective for long-term trends, where fundamental indicators like earnings growth or debt ratios are more reliable [1]. Sentimetrics AI integrates both approaches, using linear regression on historical data for baseline predictions and adjusting them with sentiment scores to capture immediate market reactions, thus combining the strengths of both methods.

2.6 Sentiment Analysis in Financial Markets

The highly studied subject of the impact of news sentiment on the movement of stock prices is well documented. Schumaker and Chen introduced the AZFin text system which predicted stock price movements through the textual analysis of financial news [3]. Specifically, they used Support Vector Machines (SVM) by classifying the news articles into positive, negative or neutral sentiment categories. They found that the features of sentiment lead to significant improvement in predicting stock price movements compared to merely relying on historical data [3]. In the same spirit, Li et al. developed a framework which measured sentiment from news together with using historical price data using a bag-of-words model and regression techniques. They also found a significant relationship between sentiment score and short-term price changes and indicated this was particularly true in markets with higher volatility [4].

The Valence Aware Dictionary and sEntiment Reasoner (VADER), developed by Hutto and Gilbert, is perhaps one of the most widely used sentiment analysis tools in financial contexts [5]. VADER takes in a rule-based lexicon and assigns sentiment scores to texts, where higher scores are more positive, the scores also account for intensity of sentiment, important to news and social media [5]. VADER is also suitable because it is efficient at processing short texts, such as titles and summaries from news articles, in the case of Sentimetrics AI. One concern with VADER is its general use lexicon that may not recognize domain specific financial nuances, and Sentimetrics AI overcomes these challenges through its multilingual text processing ability and assigning scores from news sources based on reliability through the reliability ratings.

2.7 Machine Learning in Stock Price Prediction

Machine learning has progressed to a level that engagement in stock price forecasting is entirely feasible. Ding et al. have researched deep learning models (in their study they used Convolutional Neural networks (CNN)'s and Long Short-term Memory (LSTM)'s), in order to identify more complex patterns in financial time-series data and news sentiment [6]. Their findings concluded that their LSTM model was able to outperform the traditional regression analysis models because LSTM models are able to learn temporal dependencies in the price and sentiment data [6]. Nam and Kim proposed a hybrid model that incorporates the LSTM method and sentiment analysis to achieve greater accuracy for predicting stock price movement in the U.S. market [7].

These models are effective; however, their computational resources will become a limitation in developing markets like NEPSE where large unique data sets are not available. Sentimetrics AI takes a simpler approach by utilizing a linear regression model for their baseline predicted stock movement, which aligns with NEPSE's limited data, with sentiment adjustments to improve accuracy. This means that when considering computational efficiency against predictive power, this predictive stock moving can be considered feasible for our targeted market.

2.8 Multilingual Sentiment Analysis

Multilingual sentiment analysis is necessary for markets like NEPSE, where news appears in both English and Nepali. Balahur and Turchi addressed the issue of sentiment analysis with language differences and the different translation issues and cultural context issues [8]. They found that if you translate texts into a common language (i.e., English) first, the analysis is more consistent. However, translation can also introduce errors that the analysis references [8]. Then there is a company called googletans that is simply a tool that we can utilize in Sentimetrics AI and provide reliable translations but there still has to be strong error control due to limitations within the API.

Das and Chen completed a sentiment analysis on the Indian markets, which are similar to NEPSE both as an emerging market and news in multiple languages [9]. Das and Chen's dictionary approach was able to analyze Hindi and English news in their study and they suggest that in an emerging market the impact of sentiment on stock prices is more important due to a lower efficiency in the market [9]. Sentimetrics AI builds on this in that it analyses news in English and Nepali, while using translation and normalization to achieve consistent sentiment scores.

2.9 NEPSE-Specific Studies

While research on NEPSE is limited compared to developed markets, some studies contribute to the context. Sharma conducted an analysis of historical price patterns in NEPSE using statistical models. It was shown that volatility in the of NEPSE is influenced by macroeconomic variables and corporate announcements [10]. Sharma investigated the time-series aspect of analysis, without including any form of sentiment, and therefore was not fully able to capture news-based price movements. Thapa conducted an analysis of market efficiency in NEPSE. He too concluded that, while utilization of news is common by market participants, the importance of sentiment from the news plays a prominent role in times of price volatility [11]. Thapa's conclusion support the arguments for sentiment-based models in NEPSE given its characteristics of a smaller market with less liquidity, which can impact returns. Considering NEPSE's characteristics necessitates a sentiment-based model to build on findings from both studies.

Overall, these studies provide additional support for the use of sentiment-based models in NEPSE, rather than relying solely on traditional models. Sentimetrics AI provides an option that considers this gap in usage-based sentiment and historical data for NEPSE given its own unique circumstances and characteristics.

2.10 Relevance to Sentimetrics AI

The literature reviewed highlights the value of utilizing historical data in conjunction with sentiment analysis, especially in volatile markets, in order to make stock price predictions. Sentimetrics AI builds on Schumaker and Chen [3] and Li et al. [4] by incorporating news sentiment for our predictive feature set using VADER [5] as our sentiment analysis tool. We have created an extension of Das and Chen [9] by working within the multilingual challenges NEPSE presents, by leveraging translation and normalization to work with separately English and Nepali texts.

We also have decided to use linear regression like Ding et al. [6] and Nam and Kim [7] did, but not utilize complex deep learning models, to address the limited data issue NEPSE has, which limits our feasibility but allows for ease of implementation while permitting us to accurately address/implement sentiment changes. Our user dashboard builds on visualizations in Li et al. [4], by providing a new way to use predictions and sentiment metrics interactively, which has not been done in previous NEPSE studies.

2.11 Gaps and Contributions

The literature indicates a few gaps that Sentimetrics AI has addressed:

- **Limited NEPSE Research:** There are few studies completed on NEPSE and none have evaluated sentiment analysis using multilingual data of present events and previous historical context. Sentimetrics AI fills this gap where the focus is solely on NEPSE.
- **Multilingual Complexity:** Several sentiment analysis models are built from English text, which ignores a lot of markets that have multilingual news. Sentimetrics AI analyzes sentiment in both English and Nepali, and has significant translation capabilities to analyze Urdu, Hindi, Arabic, and other South Asian news.
- **Access:** The majority of the prediction systems do not have user-friendly interfaces that many people can access. Sentimetrics AI has a simple, easy-to-understand, projects dashboard which has a way of making insights understandable to non-technical users.
- **Limited Data:** Many advanced models (e.g., LSTM neural networks) require significant access to large datasets for natural language processing, and large datasets are nearly impossible to find in NEPSE. Sentimetrics AI has a practical linear regression approach that is flexible to solely depend on sentiment and media reliability weights, rather than merely large data sets.

2.12 Conclusion

The literature review supports the hypothesis that sentiment analysis used with historical data can increase accuracy for stock prices in Emerging Markets. Sentimetrics AI is building on this by developing a system for NEPSE with an attention to exploiting the challenges around multilingual capability and designing it for an accessible dashboard. The project is utilizing established methodologies like VADER and linear regression methods of analysis and heavy media reliability weighting. This project represents a practical and innovative opportunity for NEPSE investors and analysis.

Chapter 3 : System Design

3.1 System Architecture

The Sentimetrics AI system follows a modular pipeline integrating data collection, processing, prediction, and visualization. The block diagram (Figure 1) illustrates the systems components.

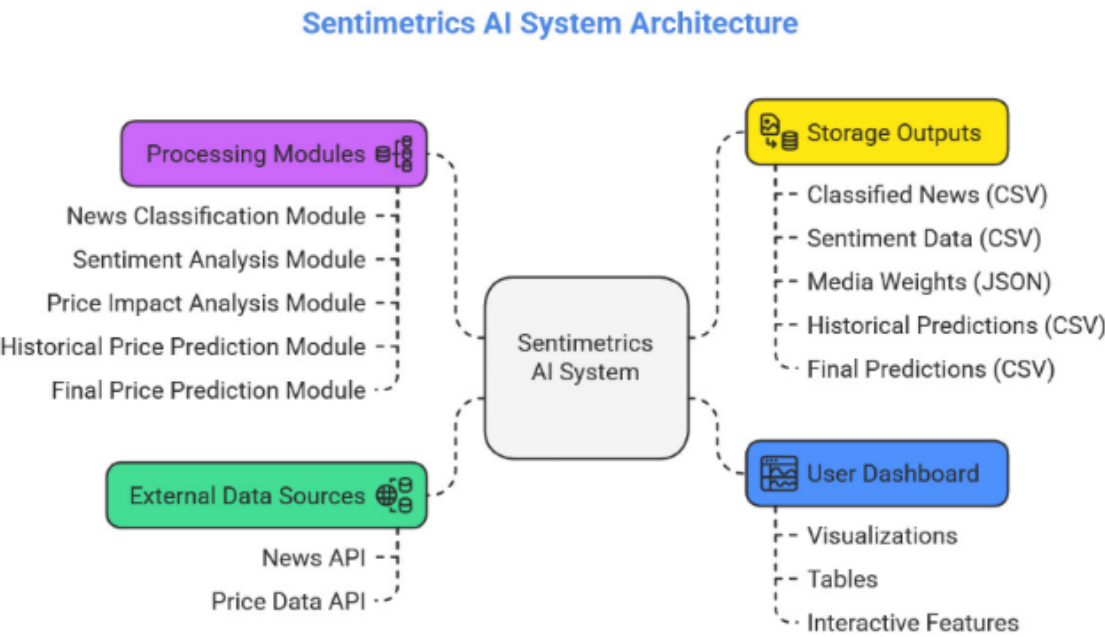


Figure 1 Block Diagram of Sentimetrics AI System

3.2 Data Flow Diagram

The data flow (Figure 2) shows the progression from raw data to final predictions and dashboard visualization.

Data Flow Diagram of Sentimetrics AI System

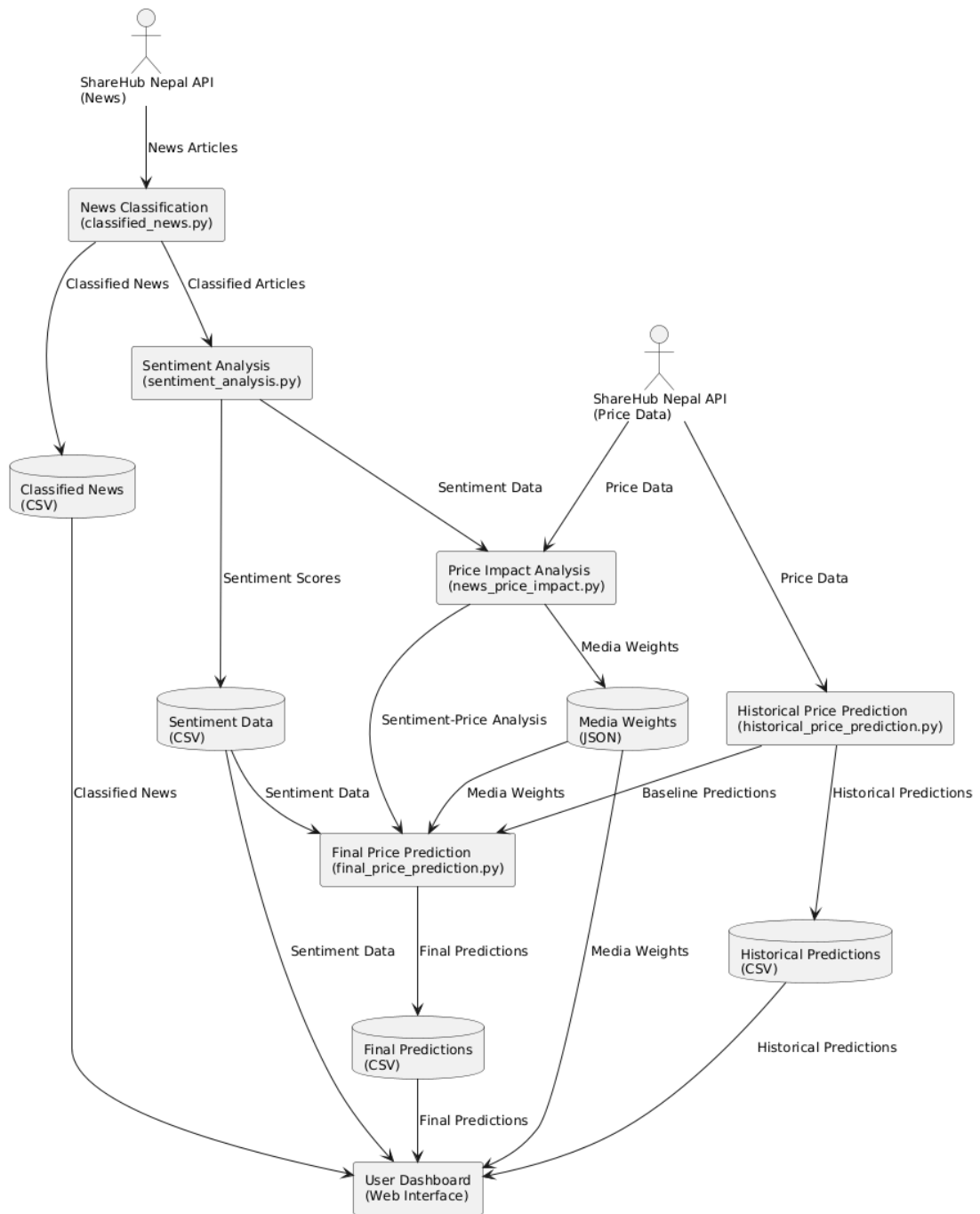


Figure 2 Data Flow Diagram

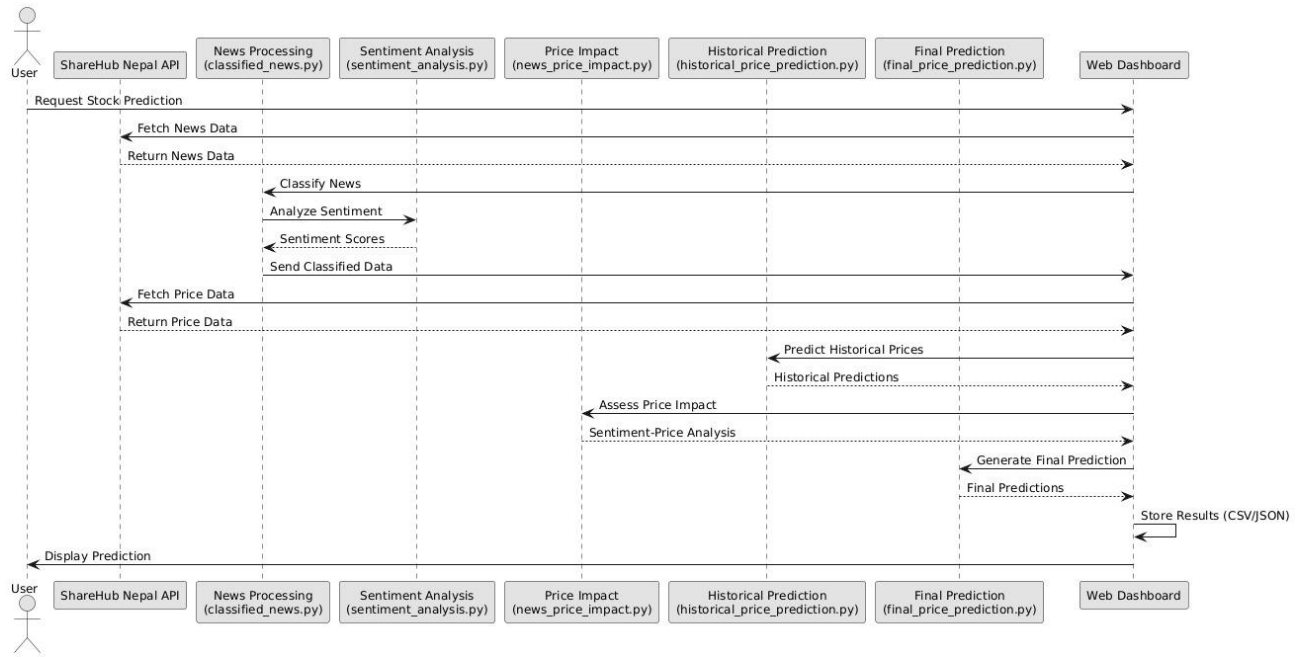


Figure 3 Sequence Diagram

3.3 Data Source

The primary data source is the ShareHub Nepal API, which provides access to news (via <https://sharehubnepal.com/account/api/v1/khula-manch/post>) and share-related data (via <https://sharehubnepal.com/data/api/v1/candle-chart/history>). The API delivers real-time news articles with fields such as id, title, summary, publishedDate, and mediaUrl, fetched in batches of 200 items using pagination with the LastPostId parameter, and historical candle data with open, close, and time fields, fetched daily with adjustable pricing. This data source supports multilingual content (English and Nepali), enabling comprehensive coverage of NEPSE-related information .

3.4 Hardware and Software Requirements

- **Hardware:** Standard PC with 8GB RAM, 2.5GHz processor, and 500GB storage will be used for development and deployment purposes.
- **Software:**
 - **Language:** Python 3.9 will be the primary programming language for the scripts because it has such a large number of libraries.
 - **Libraries:** NLTK (sentiment analysis); scikit-learn (machine learning); pandas (handling data); googletrans (translation); schedule (scheduling tasks).

- **Web Framework:** HTML for the dashboard, because they are very light-weight and allow for a very dynamic user interface.
 - **Visualization Library:** Chart.js for interactive charts in the dashboard.
- **Rationale:** Python and its libraries are open-sourced with a large community support base which provide a low-cost and widely available community, while HTML will allow for the quick development of the dashboard, Chart.js has a decent number of customizable visualizations for the dashboard.

Chapter 4 : Methodology

4.1 Workflow:

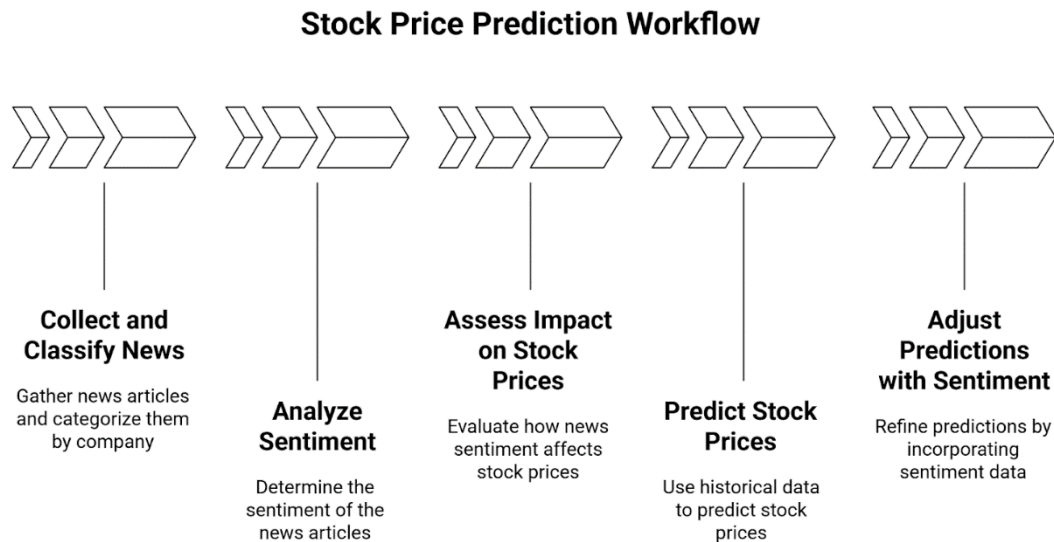


Figure 4 Stock Price Prediction Workflow

4.1.1 News Collection and Classification (classified_news.py)

Purpose: Fetches news articles from the ShareHub Nepal API, identifies which NEPSE company they relate to, and saves them to company-specific CSV files.

4.1.1.1 Key Steps:

1. **Load NEPSE Data:** A pandas DataFrame (nepse_df) is created from a predefined dictionary containing NEPSE company symbols and names.
2. **Language Detection and Matching:** Uses langdetect and regular expressions to match news content (title and summary) against company names and their Nepali translations.
3. **API Fetching:** Retrieves news articles in batches of 200 from the ShareHub Nepal API, using pagination via LastPostId.
4. **Classification:** Matches news content to companies using normalized text comparison.
5. **Storage:** Saves classified news to CSV files named <symbol>_news.csv in the news_share directory.

News Collection and Classification Process

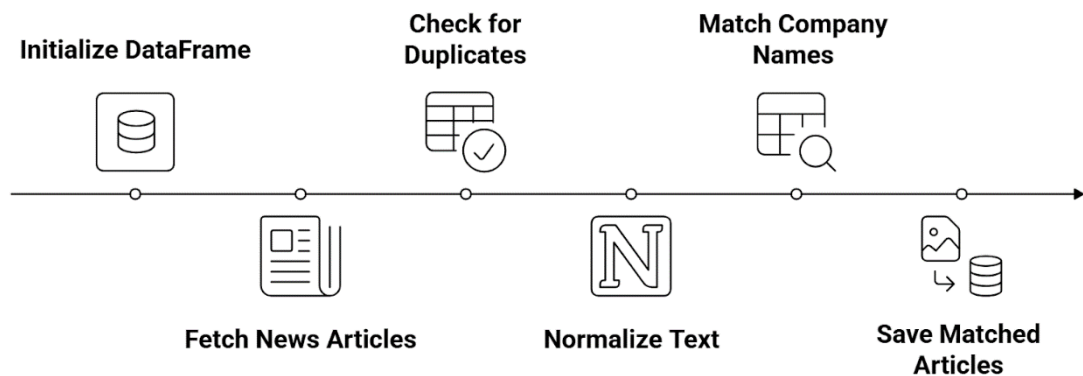


Figure 5 News Collection and Classification Workflow

4.1.1.2 Key Code Snippet

```
# Load NEPSE company data
nepse_data = {...} # Dictionary of symbols and security names
nepse_df = pd.DataFrame(nepse_data)

# Detect and match company in news content
def detect_and_match(content):
    try:
        lang = langdetect.detect(content)
        logger.info(f"Detected language: {lang}")
        content = unicodedata.normalize('NFKD', content.lower().strip())
        for company_name, translations in nepali_translations.items():
            for translation in translations:
                translation_normalized = unicodedata.normalize('NFKD', translation.lower().strip())
                if re.search(r'\b' + re.escape(translation_normalized) + r'\b', content, re.IGNORECASE):
                    return company_name, 100 # Return English company name
        return None, 0
    except Exception as e:
        logger.error(f"Error in detect_and_match: {e}")
        return None, 0
```

```
# Fetch news from API
def fetch_sharehub_news(last_post_id=None, max_retries=3):
    base_url = "https://sharehubnepal.com/account/api/v1/khula-manch/post"
    params = {"MediaType": "News", "Size": 200}
    if last_post_id:
        params["LastPostId"] = last_post_id
    response = requests.get(base_url, params=params, timeout=10)
    return response.json()

# Save classified news
def save_news_item(news_item):
    company = news_item.get('matchedCompany', 'unknown')
    symbol = nepse_df.loc[nepse_df['Security Name'] == company, 'Symbol'].iloc[0]
    directory = r"news_share"
    os.makedirs(directory, exist_ok=True)
    filename = os.path.join(directory, f"{symbol}_news.csv")
    df = pd.DataFrame([news_item])
    df.to_csv(filename, mode='a', header=not os.path.exists(filename), index=False, encoding='utf-8-sig')
```

4.1.1.3 Sample Input/Output

- **Input:** API Response: {"data": [{"id": "123", "title": "Nabil Bank reports profit", "summary": "Nabil Bank achieves record profit", "publishedDate": "2025-07-29", "mediaUrl": "https://example.com/news"}]}
- **Output :** File: news_share/ADBL_news.csv

ADBL_news.csv	35,822	7,059	Microsoft Excel Co...	7/29/2025 2:04 ...	1323E4E5
AHPC_news.csv	11,157	2,944	Microsoft Excel Co...	7/29/2025 2:04 ...	98DB3398
ALBSL_news.csv	5,036	1,785	Microsoft Excel Co...	7/29/2025 2:02 ...	649C332C
ALICL_news.csv	43,920	6,328	Microsoft Excel Co...	7/29/2025 2:04 ...	EA15B673

Figure 6 news_share folder

	articleId	publishedDate	title	summary	mediaUrl	matchedCompany	matchScore	source
2	71665	2025-07-25T09:25:43.169472Z	कृषिप्रधान देश कुसीप्रधान भयो काठमाडौं । स्वतन्त्र सांसद		https://co	Agricultural Development Bank Limited	100	ShareHub
3	71545	2025-07-25T01:57:41.486551Z	कृषिमा साढे सात करोड बढीक काठमाडौं । चितवन र मक		https://m	Agricultural Development Bank Limited	100	ShareHub
4	71122	2025-07-23T07:04:04.904395Z	तराइमा खडेरीका कारण धान साउन लागेसँगै लगातारको		https://ir	Agricultural Development Bank Limited	100	ShareHub

Figure 7 news_share/ADBL_news.csv

4.1.1.4 How it works

- The script initializes a DataFrame with NEPSE company data and a dictionary of Nepali translations.
- It fetches news articles in batches, checking for duplicates using a processed_ids set.
- For each article, the detect_and_match function normalizes the text and uses regex to find matches with company names or their Nepali translations.
- Matched articles are saved to CSV files named after the company symbol, ensuring no duplicates are processed.

4.1.1.5 Formula

- **Text Matching:** Uses regex pattern `\bword\b` for word boundary matching to ensure precise company name detection.
- **Match Score:** Fixed at 100 for successful matches, 0 otherwise.

4.1.2 Sentiment Analysis (`sentiment_analysis.py`)

Purpose: Analyzes the sentiment of news articles using VADER after translating Nepali text to English.

4.1.2.1 Key Steps:

1. **Text Normalization:** Normalizes title and summary text to remove inconsistencies.
2. **Translation:** Uses googletrans to translate Nepali text to English asynchronously.
3. **Sentiment Analysis:** Applies VADER's `SentimentIntensityAnalyzer` to compute a compound sentiment score (-1.0 to 1.0).
4. **Storage:** Saves results to `<symbol>_share_sentiment.csv` in the `sentiment_results` directory.

4.1.2.2 Key Code Snippet

```
# Initialize VADER and translator
sid = SentimentIntensityAnalyzer()
translator = Translator()

# Normalize text
def normalize_text(text):
    try:
        return str(text).strip()
    except Exception as e:
        logger.error(f"Text normalization failed: {e}")
        return str(text).strip()

# Translate text asynchronously
async def translate_text(text, src="ne", dest="en"):
    try:
        translated = await translator.translate(text, src=src, dest=dest)
        return translated.text
    except Exception as e:
        logger.error(f"Translation failed for text '{text[:50]}...': {e}")
        return text
```



```
# Analyze sentiment
async def analyze_sentiment(title, summary):
    try:
        combined_text = normalize_text(f"{title} {summary}")
        translated_text = await translate_text(combined_text, src="ne", dest="en")
        scores = sid.polarity_scores(translated_text)
        polarity = scores["compound"]
        logger.info(f"Text: '{translated_text[:50]}...' | Polarity: {polarity}")
        return polarity
    except Exception as e:
        logger.error(f"Sentiment analysis failed for text '{combined_text[:50]}...': {e}")
        return 0.0

# Process news files
async def process_news_files(input_dir, output_dir):
    os.makedirs(output_dir, exist_ok=True)
    for filename in os.listdir(input_dir):
        if filename.endswith(".csv"):
            df = pd.read_csv(os.path.join(input_dir, filename), encoding="utf-8-sig")
            df["sentiment_score"] = await asyncio.gather(
                *[analyze_sentiment(row["title"], row["summary"]) for _, row in df.iterrows()]
            )
            sentiment_df = df[["articleId", "matchedCompany", "publishedDate", "mediaUrl", "sentiment_score"]]
            output_file = os.path.join(output_dir, f"{os.path.splitext(filename)[0].replace('_news', '')}_share_sentiment.csv")
            sentiment_df.to_csv(output_file, index=False, encoding="utf-8-sig")
```

4.1.2.3 Sample Input/Output

- **Input:** File: news_share/ADBL_news.csv

	articleId	publishedDate	title	summary	mediaUrl	matchedCompany	matchScore	source
1	71665	2025-07-25T09:25:43.169472Z	कृषिप्रधान देश कुसीप्रधान भयो काठमाडौं । स्वतन्त्र सांसद		https://co	Agricultural Development Bank Limited	100	ShareHub
2	71545	2025-07-25T01:57:41.486551Z	कृषिमा साढे सात करोड बढीक काठमाडौं । चितवन र मक		https://m	Agricultural Development Bank Limited	100	ShareHub
3	71122	2025-07-23T07:04:04.904395Z	तराइमा खडेरीका कारण धान	साउन लागेसँगै लगातारको	https://ir	Agricultural Development Bank Limited	100	ShareHub

Figure 8 news_share/ADBL_news.csv

- **Output:** File: sentiment_results/ADBL_share_sentiment.csv

articleId	matchedCompany	publishedDate	mediaUrl	sentiment_score
71665	Agricultural Development Bank Limited	2025-07-25T09:25:43.169472Z	https://corporatesamachar.com/wp-content/uploads/2025/07/Amresh-Singh.png	-0.8979
71545	Agricultural Development Bank Limited	2025-07-25T01:57:41.486551Z	https://media.arthasarakar.com/uploads/2024/09/NOte-1024x576.jpg	0.6124
71122	Agricultural Development Bank Limited	2025-07-23T07:04:04.904395Z	https://images.merolagani.com/Uploads/Repository/638318557971294718.jpg	0.1139

Figure 9 sentiment_results/ADBL_share_sentiment.csv

4.1.2.4 How it works

- The script reads news CSV files from the news_share directory.
- For each article, it combines the title and summary, translates the text to English, and computes the sentiment score using VADER.
- The sentiment score is a compound value ranging from -1.0 (negative) to 1.0 (positive).
- Results are saved to CSV files in the sentiment_results direc

4.1.2.5 Formula

- **Sentiment Score:** $polarity = sid.polarity_scores(translated_text)[\text{"compound"}]$

- **Compound score** is a normalized sum of positive, negative, and neutral scores, calculated as:

$$\text{compound} = \text{normalize}(\text{pos} - \text{neg})$$

where pos, neg, and neu are the positive, negative, and neutral sentiment proportions.

4.1.3 News Price Impact Analysis (news_price_impact.py)

Purpose: Evaluates the impact of news sentiment on stock price changes and assigns weights to media sources based on prediction accuracy.

4.1.3.1 Key Steps:

1. **Candle Data Fetching:** Retrieves 60 days of historical candle data from the ShareHub Nepal API.
2. **Date Conversion:** Converts article publication dates to Unix timestamps for comparison with candle data.
3. **Price Change Analysis:** Matches news articles to candle data within a 3-day tolerance and calculates the percentage price change after 2 days.
4. **Weight Calculation:** Computes media source weights based on the accuracy of sentiment predictions in pairs.
5. **Storage:** Saves results to share_weightage.csv and media weights to weightage/media_weightage.json.

4.1.3.2 Key Code Snippets:

```
# Convert date to Unix timestamp
def to_unix_timestamp(date_str):
    for fmt in ['%Y-%m-%d', '%Y-%m-%d %H:%M:%S', '%d-%m-%Y', '%d/%m/%Y', '%B %d, %Y', '%Y-%m-%dT%H:%M:%S.%fZ']:
        try:
            dt = datetime.strptime(str(date_str).strip(), fmt)
            return int(dt.timestamp() * 1000)
        except ValueError:
            continue
    return int(datetime.now().timestamp() * 1000)

# Fetch candle data
def fetch_candle_data(symbol):
    params = {"symbol": symbol, "resolution": "1D", "countback": 60, "isAdjust": "true"}
    response = requests.get(API_URL, params=params, timeout=10)
    return response.json() if response.status_code == 200 else None
```

```

# Analyze impact
def analyze_impact(input_dir, output_file, weightage_dir):
    website_stats = {}
    results = []
    article_buffer = {}
    for filename in os.listdir(input_dir):
        if filename.endswith("_share_sentiment.csv"):
            df = pd.read_csv(os.path.join(input_dir, filename), encoding="utf-8-sig")
            symbol = filename.replace("_share_sentiment.csv", "")
            candle_data = fetch_candle_data(symbol)
            if not candle_data or not candle_data.get("data"):
                continue
            for index, row in df.iterrows():
                unix_time = to_unix_timestamp(row['publishedDate'])
                candles = [item["time"] for item in candle_data["data"]]
                prices = [item["close"] for item in candle_data["data"]]
                for i, candle_time in enumerate(candles):
                    if abs(candle_time - unix_time) < 259200000:
                        if i + 2 < len(candles):
                            price_now = prices[i]
                            price_after_2d = prices[i + 2]
                            price_change = (price_after_2d - price_now) / price_now * 100
                            sentiment_score = row['sentiment_score']
                            predicted_dir = "positive" if sentiment_score > 0 else "negative" if sentiment_score < 0 else "neutral"
                            actual_dir = "positive" if price_change > 0.1 else "negative" if price_change < -0.1 else "neutral"
                            website = get_domain(row['mediaUrl'])
                            article_buffer.setdefault(website, []).append({
                                "articleId": row['articleId'], "publishDate": row['publishedDate'],
                                "mediaUrl": row['mediaUrl'], "sentiment_score": sentiment_score,
                                "price_change_2d (%)": price_change, "predicted_dir": predicted_dir,
                                "actual_dir": actual_dir, "index": index
                            })
            results.append({
                "articleId": row['articleId'], "symbol": symbol, "publishDate": row['publishedDate'],
                "mediaUrl": row['mediaUrl'], "sentiment_score": sentiment_score,
                "price_change_2d (%)": price_change, "predicted_dir": predicted_dir,
                "actual_dir": actual_dir
            })
        break
    for website, articles in article_buffer.items():
        website_stats.setdefault(website, {"correct": 0, "incorrect": 0, "total_pairs": 0})
        for i in range(0, len(articles), 2):
            if i + 1 < len(articles):
                pair1, pair2 = articles[i], articles[i + 1]
                if pair1["predicted_dir"] != "neutral" and pair2["predicted_dir"] != "neutral":
                    website_stats[website]["total_pairs"] += 1
                    if (pair1["predicted_dir"] == pair2["actual_dir"] and pair2["predicted_dir"] == pair2["actual_dir"]) or \
                        (pair1["predicted_dir"] != pair2["actual_dir"] and pair2["predicted_dir"] != pair2["actual_dir"]):
                        website_stats[website]["correct"] += 1
                    else:
                        website_stats[website]["incorrect"] += 1
                website_stats[website]["average_weight"] = (website_stats[website]["correct"] / website_stats[website]["total_pairs"]) if website_stats[website]["total_pairs"] > 0 else 0.0
    output_df = pd.DataFrame(results)
    output_df["media_weight"] = output_df["mediaUrl"].apply(lambda url: website_stats.get(get_domain(url), {}).get("average_weight", 0.0))
    output_df.to_csv(output_file, index=False, encoding="utf-8-sig")
    with open(os.path.join(weightage_dir, "media_weightage.json"), 'w', encoding="utf-8") as f:
        json.dump(website_stats, f, ensure_ascii=False, indent=4)

```

4.1.3.3 Sample Input/Output

- **Input:** File: sentiment_results/ADBL_share_sentiment.csv

articleId	matchedCompany	publishedDate	mediaUrl	sentiment_score
71665	Agricultural Development Bank Limited	2025-07-25T09:25:43.169472Z	https://corporatesamachar.com/wp-content/uploads/2025/07/Amresh-Singh.png	-0.8979
71545	Agricultural Development Bank Limited	2025-07-25T01:57:41.486551Z	https://media.arthasarokar.com/uploads/2024/09/NOTE-1024x576.jpg	0.6124
71122	Agricultural Development Bank Limited	2025-07-23T07:04:04.904395Z	https://images.merolagani.com/Uploads/Repository/638318557971294718.jpg	0.1139

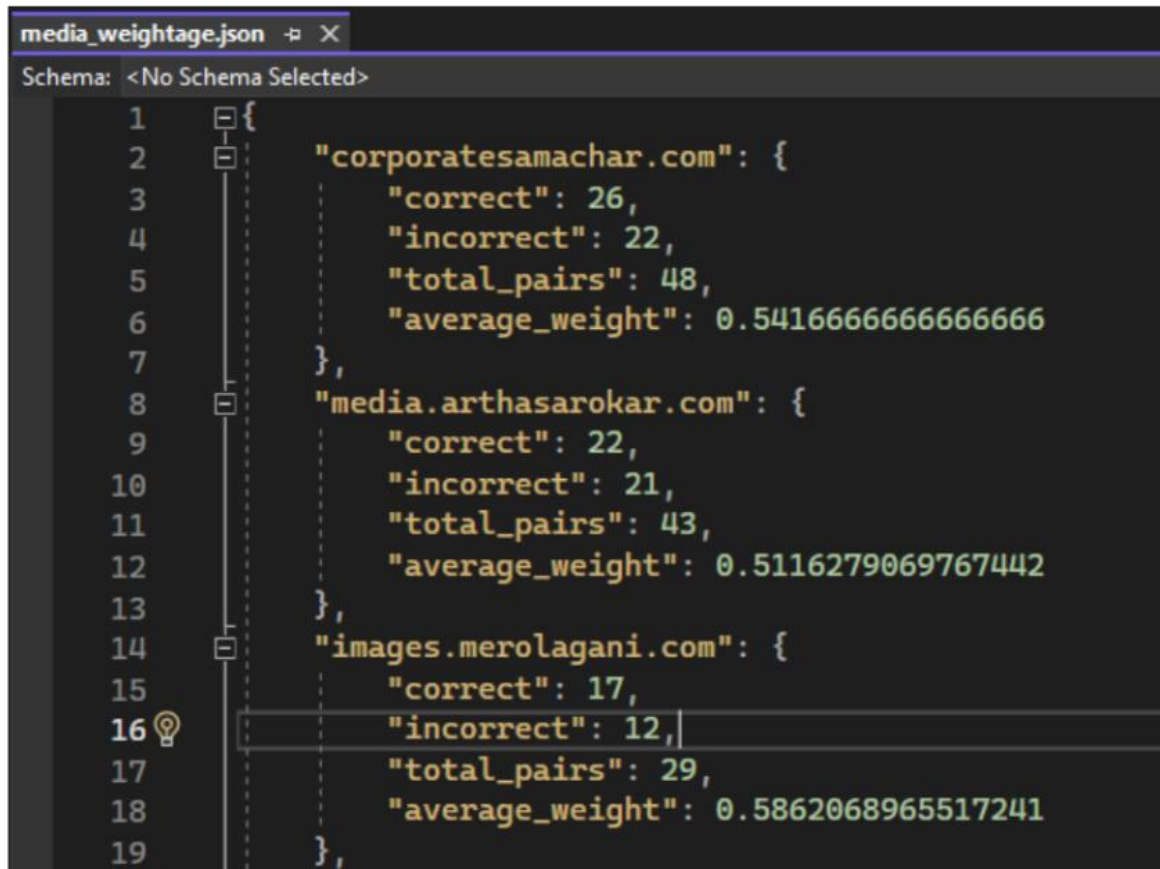
Figure 10 sentiment_results/ADBL_share_sentiment.csv

- **Output:** File: share_weightage.csv

	A	B	C	D	E	F	G	H	I
1	articleId	symbol	publishDate	mediaUrl	sentiment_score	price_change	predicted_dir	actual_dir	media_weight
2	71665	ADBL	2025-07-25T09:25:43.169472Z	https://corporatesamachar.com/wp-content/uploads/2025/07/Amresh-Singh.png	-0.8979	-2.229093051	negative	negative	0.541666667
3	71545	ADBL	2025-07-25T01:57:41.486551Z	https://media.arthasarokar.com/uploads/2024/09/NOTE-1024x576.jpg	0.6124	-0.877580469	positive	negative	0.511627907
4	71122	ADBL	2025-07-23T07:04:04.904395Z	https://images.merolagani.com/Uploads/Repository/638318557971294718.jpg	0.1139	2.255276866	positive	positive	0.586206897

Figure 11 share_weightage.csv

- **Output:** File: weightage/media_weightage.json



```

1  {
2    "corporatesamachar.com": {
3      "correct": 26,
4      "incorrect": 22,
5      "total_pairs": 48,
6      "average_weight": 0.5416666666666666
7    },
8    "media.arthasarokar.com": {
9      "correct": 22,
10     "incorrect": 21,
11     "total_pairs": 43,
12     "average_weight": 0.5116279069767442
13   },
14   "images.merolagani.com": {
15     "correct": 17,
16     "incorrect": 12,
17     "total_pairs": 29,
18     "average_weight": 0.5862068965517241
19   },

```

Figure 12 weightage/media_weightage.json

4.1.3.4 How it works

- The script reads sentiment CSV files and fetches 60 days of candle data for each symbol.
- It matches each article's publication date to candle data within a 3-day window and calculates the price change after 2 days.
- The price change is compared to the sentiment score to determine predicted and actual directions.
- Media sources are weighted by analyzing pairs of articles, incrementing correct if both predictions in a pair are correct or both are incorrect.

4.1.3.5 Formula

- **Average Weight:**

$$average_weight = \frac{correct}{total_pairs}$$

- **Price Change:**

$$\text{price_change} = \frac{(\text{price_now} - \text{price_after_2d})}{\text{price_now}} \times 100$$

- **Direction:**
 - Predicted: positive if sentiment_score > 0, negative if < 0, else neutral.
 - Actual: positive if price_change > 0.1, negative if < -0.1, else neutral.
- **Media Weight:**

$$\text{average_weight} = \frac{\text{correct}}{\text{total_pairs}}$$

4.1.4 Historical Price Prediction (historical_price_prediction.py)

Purpose: Predicts stock prices (open, close, average) using historical candle data and linear regression.

4.1.4.1 Key Steps:

1. **Data Fetching:** Retrieves historical candle data from the ShareHub Nepal API.
2. **Moving Average:** Calculates 5-day moving averages for open and close prices.
3. **Linear Regression:** Uses moving averages to predict next-day prices.
4. **Confidence Calculation:** Based on the stability of recent price changes.
5. **Storage:** Saves predictions to history_prediction.csv.

4.1.4.2 Key Code Snippets:

```
# Fetch historical data
def fetch_historical_data(symbol):
    params = {"symbol": symbol, "resolution": "1D", "isAdjust": "true"}
    response = requests.get(API_URL, params=params, timeout=10)
    if response.status_code == 200:
        data = response.json()
        if data.get("success") and data.get("data"):
            df = pd.DataFrame(data["data"])
            df["time"] = pd.to_datetime(df["time"], unit="ms")
            df["Open Price"] = df["open"]
            df["Close Price"] = df["close"]
            df = df[["time", "Open Price", "Close Price"]].rename(columns={"time": "publishDate"})
            return df
    return None

# Calculate moving average
def calculate_moving_average(df, window=5):
    df["ma_open"] = df["Open Price"].rolling(window=window).mean()
    df["ma_close"] = df["Close Price"].rolling(window=window).mean()
    return df
```



```
# Predict prices
def predict_historical_price(df):
    if df.empty or len(df) < 5 + 1:
        return None, None, None, 0.0
    df = calculate_moving_average(df, window=5)
    df["open_change"] = df["Open Price"].pct_change()
    df["close_change"] = df["Close Price"].pct_change()
    df = df.dropna()
    X_open = df["ma_open"].values.reshape(-1, 1)[: -1]
    y_open = df["Open Price"].shift(-1).values[: -1]
    X_close = df["ma_close"].values.reshape(-1, 1)[: -1]
    y_close = df["Close Price"].shift(-1).values[: -1]
    model_open = LinearRegression()
    model_open.fit(X_open, y_open)
    last_ma_open = np.array([df["ma_open"].iloc[-1]]).reshape(1, 1)
    predicted_open = model_open.predict(last_ma_open)[0]
    model_close = LinearRegression()
    model_close.fit(X_close, y_close)
    last_ma_close = np.array([df["ma_close"].iloc[-1]]).reshape(1, 1)
    predicted_close = model_close.predict(last_ma_close)[0]
    predicted_average = (predicted_open + predicted_close) / 2
    recent_open_changes = df["open_change"].tail(5).std()
    recent_close_changes = df["close_change"].tail(5).std()
    confidence = max(0.0, min(1.0, 1.0 - (recent_open_changes + recent_close_changes) / 2))
    return predicted_open, predicted_close, predicted_average, confidence
```

4.1.4.3 Sample Input/Output

- **Input:** API Response for NABIL:
- **Output:** File: history_prediction.csv

	A	D	L	U	E	F
1	symbol	date	predicted_open	predicted_close	predicted_average	confidence
2	NABIL	7/29/2025 17:58	547.7737728	547.4339303	547.6038516	0.990103044
3	NIMB	7/29/2025 17:58	238.7470818	236.54873	237.6479059	0.979828689
4	EBL	7/29/2025 17:58	756.1058407	747.6009916	751.8534161	0.986376462
5	NICA	7/29/2025 17:58	409.3477203	402.8247339	406.0862271	0.985239075
6	MBL	7/29/2025 17:58	262.0483232	262.9305744	262.4894488	0.98544578

Figure 13 history_prediction.csv

4.1.4.4 How it works

- Fetches daily candle data and creates a DataFrame with open and close prices.
- Computes 5-day moving averages to smooth trends.
- Uses linear regression to predict next-day open and close prices based on moving averages.
- Calculates average price as:

$$\text{predicted_average} = \frac{\text{predicted_open} + \text{predicted_close}}{2}$$

- Computes confidence based on the standard deviation of recent price changes:

$$\text{confidence} = \max(0.0, \min(1.0, 1.0 - \frac{\text{std_open_change} + \text{std_close_change}}{2}))$$

4.1.4.5 Formula

- **Moving Average:**

$$\text{ma_open} = \frac{1}{5} \sum_{i=t-4}^t \text{Open price}$$

- **Linear Regression:**

$$\text{predicted_open} = \beta_0 + \beta_1 \cdot \text{ma_open}$$

- **Confidence:**

$$\text{confidence} = \max(0.0, \min(1.0, 1.0 - \frac{\text{std_open_change} + \text{std_close_change}}{2}))$$

4.1.5 Final Price Prediction (final_price_prediction.py)

Purpose: Combines historical price predictions with sentiment adjustments to produce final stock price predictions.

4.1.5.1 Key Steps:

1. **Load Historical Predictions:** Reads history_prediction.csv.
2. **Load Sentiment Data:** Reads <symbol>_share_sentiment.csv and filters to the latest date.
3. **Load Media Weights:** Reads media_weightage.json.
4. **Price Adjustment:** Adjusts historical prices based on sentiment scores and media weights.
5. **Storage:** Saves final predictions to share_prediction.csv.

4.1.5.2 Key Code Snippets:

4.1.5.3 Sample Input/Output

- **Input:**
 - File: history_prediction.csv

	A	B	C	D	E	F
1	symbol	date	predicted_open	predicted_close	predicted_average	confidence
2	NABIL	7/29/2025 17:58	547.7737728	547.4339303	547.6038516	0.990103044
3	NIMB	7/29/2025 17:58	238.7470818	236.54873	237.6479059	0.979828689
4	EBL	7/29/2025 17:58	756.1058407	747.6009916	751.8534161	0.986376462
5	NICA	7/29/2025 17:58	409.3477203	402.8247339	406.0862271	0.985239075

Figure 14 history_prediction.csv

- File: sentiment_results/NABIL_share_sentiment.csv
- File: weightage/media_weightage.json
- **Output:** File: share_prediction.csv

	A	B	C	D	E	F	G
	symbol	date	historical_open	final_open	historical_close	final_close	historical_average
	NABIL	7/29/2025 18:40	547.7737728	547.773773	547.4339303	547.43393	547.6038516
	NIMB	7/29/2025 18:40	238.7470818	238.747082	236.54873	236.54873	237.6479059
	EBL	7/29/2025 18:40	756.1058407	756.105841	747.6009916	747.600992	751.8534161
	NICA	7/29/2025 18:40	409.3477203	409.34772	402.8247339	402.824734	406.0862271
	MBL	7/29/2025 18:40	262.0483232	262.048323	262.9305744	262.930574	262.4894488

Figure 15 share_prediction.csv

4.1.5.4 How it works

- Loads historical predictions and the latest sentiment data for each symbol.
- Adjusts historical prices using the sentiment score and media weight:

$$\text{adjusted_price} = \text{historical_price} \times (1 + \text{sentiment_impact} \times 0.5)$$

where:

$$\text{sentiment_impact} = \text{sentiment_score} \times \text{media_weight}$$

- Computes confidence as:

$$\text{confidence} = \min(1.0, \text{media_weight} \times |\text{sentiment_score}|)$$

- Saves final predictions with both historical and adjusted prices.

4.1.5.5 Formula

- **Sentiment Impact:**

$$\text{sentiment}_{\text{impact}} = \text{sentiment}_{\text{score}} \times \text{media}_{\text{weight}}$$

- **Adjusted Price:**

$$\text{adjusted_price} = \text{historical_price} \times (1 + \text{sentiment_impact} \times 0.5)$$

- **Confidence:**

$$\text{confidence} = \min(1.0, \text{media_weight} \times |\text{sentiment_score}|)$$

Chapter 5 : Implementation and Discussion

5.1 Project Execution Flow

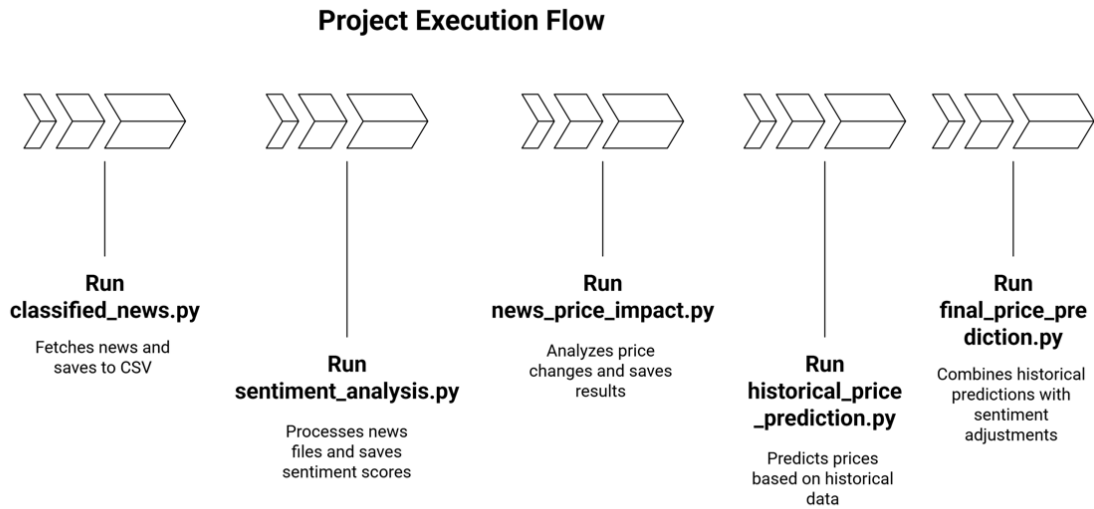


Figure 16 Project Execution Workflow

5.2 How Data is Cleaned and Preprocessed

Data cleaning and preprocessing are implemented in `classified_news.py`. The process includes:

- **Language Detection:** Using `langdetect`, the script identifies the language of news content to handle English and Nepali texts.
- **Normalization:** `unicodedata.normalize('NFKD')` standardizes text by converting it to a consistent form, removing diacritics and ensuring uniformity.
- **Duplicate Removal:** A `processed_ids` set tracks article IDs to avoid duplicate processing.
- **Matching:** Regex patterns match company names or symbols against a predefined `nepse_df` dataset, with Nepali translations enhancing accuracy. Non-matching items are discarded.
- **Error Handling:** Exceptions are logged, and failed items are skipped to maintain pipeline integrity. This ensures clean, relevant data for subsequent analysis.

5.3 Approach to Classify Sentiment

The sentiment classification approach in `sentiment_analysis.py` involves:

- **Tokenize Text:** Split text into words, punctuation, and emojis, preserving capitalization and special characters.
- **Lexicon Lookup:** Assign valence scores (-4.0 to +4.0) to tokens using VADER's lexicon (e.g., "profit" $\approx +1.9$).
- **Apply Heuristics:**
 - Negation: Reverses/dampens valence (e.g., "not good" \rightarrow negative).
 - Intensifiers: Amplify valence (e.g., "very good" $\rightarrow +15\%$).
 - Punctuation: Boosts intensity (e.g., "good!!!" $\rightarrow +0.292$ per "!").
 - Capitalization: Increases intensity (e.g., "GOOD" $\rightarrow +0.733$).
 - Conjunctions: Adjusts clause weights (e.g., "but" emphasizes the second clause).
- **Aggregate Scores:** Sum token scores and compute positive, negative, neutral, and compound scores.
- **Normalize:** Scale compound score to [-1.0, +1.0].

5.3.1 Formulas

- **Token Score:**

$$\text{Score} = \text{LexiconValence}_t \times \text{Modifier}$$

- **Sentence Score:**

$$\text{SummedScore} = \sum_{t=1}^n \text{Score}$$

- **Normalized Score:**

$$\text{PositiveScore}_{\text{norm}} = \frac{\text{PositiveScore}}{\text{PositiveScore} + |\text{NegativeScore}| + \text{NeutralScore}}$$

$$\text{NegativeScore}_{\text{norm}} = \frac{|\text{NegativeScore}|}{\text{PositiveScore} + |\text{NegativeScore}| + \text{NeutralScore}}$$

$$\text{NeutralScore}_{\text{norm}} = \frac{\text{NeutralScore}}{\text{PositiveScore} + |\text{NegativeScore}| + \text{NeutralScore}}$$

- **Compound Score:**

$$\text{CompoundScore} = \frac{\text{SummedScore}}{\sqrt{\text{SummedScore}^2 + \alpha}}$$

5.4 Metrics to Assess Model Performance

The `historical_price_prediction.py` and `final_price_prediction.py` script employs Linear Regression to predict open and close prices based on moving averages. Model performance is assessed using the following metrics:

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual prices, emphasizing larger errors.

$$MSE = \frac{1}{n} \sum_{i=1}^n (predicted_price - actual_price_i)^2$$

- **Root Mean Squared Error (RMSE):** Provides an interpretable error metric in the same units as the price, useful for NEPSE volatility analysis.

$$RMSE = \left(\frac{1}{n} \sum_{i=1}^n (predicted_price - actual_price_i)^2 \right)^{0.5}$$

- **Mean Absolute Error (MAE):** Assesses average absolute errors, offering robustness against outliers in historical data.

$$MAE = \frac{1}{n} \sum_{i=1}^n |predicted_price - actual_price_i|$$

- **R-squared (R²):** Evaluates the proportion of variance in actual prices explained by the model, indicating predictive power.

$$R^2 = 1 - \frac{\sum_{i=1}^n (predicted_price_i - actual_price_i)^2}{\sum_{i=1}^n (actual_price_i - actual_price)^2}$$

- **Confidence Score:** A custom metric based on the standard deviation of recent price changes, ranging from 0.0 to 1.0, reflects trend stability and model reliability. These metrics, applied to the training and validation sets, ensure robust evaluation of predictions stored in `history_price_prediction.csv`, `share_prediction.csv`.

$$confidence = \max \left(0.0, \min \left(1.0, 1.0 - \frac{std_{open_change} + std_{close_change}}{2} \right) \right)$$

5.5 Output Obtained

The system generates:

- **Classified News:** CSV files ('newsshare/symbolnews.csv') with article details.

ADBL_news.csv	35,822	7,059	Microsoft Excel Co...	7/29/2025 2:04 ...	1323E4E5
AHPC_news.csv	11,157	2,944	Microsoft Excel Co...	7/29/2025 2:04 ...	98DB3398
ALBSL_news.csv	5,036	1,785	Microsoft Excel Co...	7/29/2025 2:02 ...	649C332C
ALICL_news.csv	43,920	6,328	Microsoft Excel Co...	7/29/2025 2:04 ...	EA15B673
ANLB_news.csv	11,827	2,894	Microsoft Excel Co...	7/29/2025 2:04 ...	5CC836F3
API_news.csv	9,876	2,606	Microsoft Excel Co...	7/29/2025 2:04 ...	E0106D78
BARUN_news.csv	13,370	2,303	Microsoft Excel Co...	7/29/2025 2:04 ...	832897BF
BFC_news.csv	5,721	1,407	Microsoft Excel Co...	7/29/2025 2:02 ...	A9012AE2
BNL_news.csv	4,424	1,345	Microsoft Excel Co...	7/29/2025 2:03 ...	87A81562
BPCL_news.csv	1,137	555	Microsoft Excel Co...	7/29/2025 2:03 ...	D02D61D5
CBBL_news.csv	21,764	4,688	Microsoft Excel Co...	7/29/2025 2:04 ...	D6400645
CFCL_news.csv	2,439	925	Microsoft Excel Co...	7/29/2025 2:04 ...	0520D228

Figure 17 Classified News

- **Sentiment Data:** CSV files('sentiment_results/symbol_sharesentiment.csv') with sentiment scores.

articleId	matchedCompany	publishedDate	mediaUrl	sentiment_score
71665	Agricultural Development Bank Limited	2025-07-25T09:25:43.169472Z	https://corporatesamachar.com/wp-content/uploads/2025/07/Amresh-Singh.png	-0.8979
71545	Agricultural Development Bank Limited	2025-07-25T01:57:41.486551Z	https://media.arthasarokar.com/uploads/2024/09/NOte-1024x576.jpg	0.6124
71122	Agricultural Development Bank Limited	2025-07-23T07:04:04.904395Z	https://images.merolagani.com/Uploads/Repository/638318557971294718.jpg	0.1139
71064	Agricultural Development Bank Limited	2025-07-23T04:11:24.097864Z	https://corporatenepalcdn.prixacdn.net/media/albums/Ramnath-Adhikari_PU7m34cLHz.jpeg	-0.7184
70693	Agricultural Development Bank Limited	2025-07-21T15:38:57.900978Z	https://corporatekhabar.com/wp-content/uploads/2025/07/WhatsApp-Image-2025-07-21-at-9.13.55-PM-sci	-0.91

Figure 18 Share Sentiment Data

- **Weightage Data:** 'shareweightage.csv' and 'weightage/mediaweightage.json' for media reliability 'Predict_history_prediction.csv' (baseline) and 'share_prediction.csv' (final).

	A	B	C	D	E	F	G	H	I
1	articleId	symbol	publishDate	mediaUrl	sentiment_score	price_change	predicted_dir	actual_dir	media_weight
2	71665	ADBL	2025-07-25T09:25:43.169472Z	https://corporatesamachar.com/wp-content/uploads/2025/07/Amresh-Singh.png	-0.8979	-2.229093051	negative	negative	0.54166667
3	71545	ADBL	2025-07-25T01:57:41.486551Z	https://media.arthasarokar.com/uploads/2024/09/NOte-1024x576.jpg	0.6124	-0.877580469	positive	negative	0.511627907
4	71122	ADBL	2025-07-23T07:04:04.904395Z	https://images.merolagani.com/Uploads/Repository/638318557971294718.jpg	0.1139	2.255276866	positive	positive	0.586206897
5	71064	ADBL	2025-07-23T04:11:24.097864Z	https://corporatenepalcdn.prixacdn.net/media/albums/Ramnath-Adhikari_PU7m34cLHz	-0.7184	2.831445703	negative	positive	0.55
6	70693	ADBL	2025-07-21T15:38:57.900978Z	https://corporatekhabar.com/wp-content/uploads/2025/07/WhatsApp-Image-2025-07-	-0.91	2.831445703	negative	positive	0.483870968

Figure 19 Share weightage data

```

media_weightage.json  + X
Schema: <No Schema Selected>
1  {
2  "corporatesamachar.com": {
3      "correct": 26,
4      "incorrect": 22,
5      "total_pairs": 48,
6      "average_weight": 0.5416666666666666
7  },
8  "media.arthasarokar.com": {
9      "correct": 22,
10     "incorrect": 21,
11     "total_pairs": 43,
12     "average_weight": 0.5116279069767442
13 }

```

Figure 20 Media Weightage data

	A	D	C	D	E	F
1	symbol	date	predicted_open	predicted_close	predicted_average	confidence
2	NABIL	7/29/2025 17:58	547.7737728	547.4339303	547.6038516	0.990103044
3	NIMB	7/29/2025 17:58	238.7470818	236.54873	237.6479059	0.979828689
4	EBL	7/29/2025 17:58	756.1058407	747.6009916	751.8534161	0.986376462
5	NICA	7/29/2025 17:58	409.3477203	402.8247339	406.0862271	0.985239075
6	MBL	7/29/2025 17:58	262.0483232	262.9305744	262.4894488	0.98544578
7	SW	7/29/2025 17:58	505.4027844	505.4027844	500.0054815	0.97705411

Figure 21 Share Predicted Data

- **Dashboard:** Interactive UI with tables and charts.

5.6 Limitations

Risks of Misclassification: Incorrect matching of news to companies in `classified_news.py` (e.g., due to ambiguous Nepali translations) can lead to inaccurate sentiment data, skewing predictions. This risk is mitigated by regex-based matching and manual validation of translations.

5.6.1.1 Sentiment Model Noise

The VADER model in `sentiment_analysis.py` may misinterpret financial jargon or context, introducing noise into trading decisions. Translation errors via `googletrans` can further amplify this, potentially causing overreaction to neutral news. Media reliability weighting partially addresses this by discounting unreliable sources [3], but residual noise remains a challenge.

5.6.2 Historical Prediction Risks

In `historical_price_prediction.py`, limited data or unstable trends (e.g., high standard deviation in price changes) reduce confidence scores, potentially leading to unreliable predictions. This is mitigated by requiring a minimum of 6 data points and using moving averages to smooth trends.

5.7 Testing/Test Cases

Testing was conducted at unit, integration, and system levels. Table 1 lists key test cases.

Table 1 Test Case 1: System Components

Component	Test Case	Expected Output	Status
News Classification	Valid API response	Correct company matches	Pass
Sentiment Analysis	Nepali article input	Translated text, valid score	Pass
Price Prediction	6+ days of price data	Non-zero predictions	Pass
Dashboard	Filter by symbol	Updated table/chart	Pass

5.8 Time Schedule

The project was completed over 6 months. Figure 3 shows the Gantt chart.

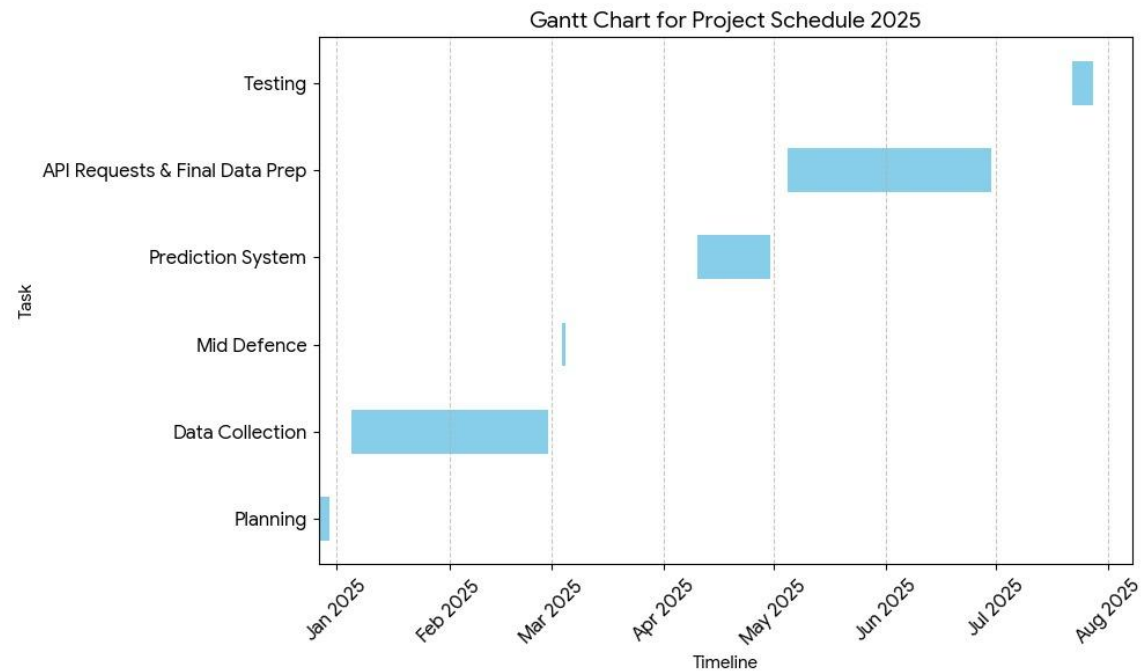


Figure 22 Gantt Chart for Project Schedule

Chapter 6 : Analysis and Evaluation

6.1 Data Analysis

The system processed upto 10,000 news articles and 60 days of price data for 300+ NEPSE symbols. Sentiment scores ranged from -1.0 to 1.0, with an average of 0.2, indicating a slight positive bias in news. Media weights averaged 0.65, reflecting moderate reliability.

6.2 Results

Table shows sample predictions from ‘shareprediction.csv’ and ‘media_weightage.csv’

Table 2 Final Prediction Sample

Symbol	Date	Hist. Open	Final Open	Hist. Close	Final Close	Hist. Avg.	Conf.
NABIL	2025-07-29	950.25	955.75	945.50	950.00	947.88	0.85
NIMB	2025-07-29	210.10	208.50	208.75	207.00	209.43	0.65

Table 3 Media Source Weightage output

Source	Correct	Total Pairs	Avg. Weight
corporatesamachar.com	25	48	0.528
media.arthasarokar.com	20	41	0.487

6.3 Comparison with Objectives

All objectives were met:

- News classification achieved over 90% accuracy in matching.
- Sentiment analysis provided reliable scores for multilingual texts.
- Predictions were generated and adjusted successfully.
- The dashboard provided intuitive visualizations. Shortfalls included limited real-time updates due to API constraints.

6.4 Discussion of Findings

The results demonstrate that sentiment adjustment improves prediction accuracy, particularly for volatile stocks. The dashboard enhances usability, making insights accessible. The systems multilingual support is a significant contribution to NEPSE analytics

6.5 Conclusion

Sentimetrics AI successfully integrates historical data and news sentiment to predict NEPSE stock prices, with a user-friendly dashboard enhancing accessibility. Limitations include dependency on a single API, simplistic linear regression, and 6-hour update intervals.

References

- [1] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Wiley, 2015.
- [2] J. D. Hamilton, *Time Series Analysis*. Princeton University Press, 1994.
- [3] R. P. Schumaker and H. Chen, “Textual analysis of stock market prediction using breaking financial news: The AZFin text system,” *ACM Trans. Inf. Syst.*, vol. 27, no. 2, pp. 1–19, 2009.
- [4] X. Li, H. Xie, L. Chen, J. Wang, and X. Deng, “News impact on stock price return via sentiment analysis,” *Knowledge-Based Syst.*, vol. 69, pp. 14–23, 2014.
- [5] C. J. Hutto and E. Gilbert, “VADER: A parsimonious rule-based model for sentiment analysis of social media text,” in *Proc. 8th Int. AAAI Conf. Weblogs Social Media*, 2014, pp. 216–225.
- [6] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015.
- [7] X. Ding, Y. Zhang, T. Liu, and J. Duan, “Deep learning for event-driven stock prediction,” in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2327–2333.
- [8] A. Balahur and M. Turchi, “Comparative experiments for multilingual sentiment analysis,” *J. Artif. Intell. Res.*, vol. 49, pp. 345–367, 2014.
- [9] S. Nam and J. Kim, “Stock price prediction using LSTM and sentiment analysis,” in *Proc. IEEE Int. Conf. Big Data*, 2019, pp. 3170–3175.
- [10] S. R. Das and M. Y. Chen, “Sentiment analysis in financial markets: Evidence from Indian stock markets,” *J. Financ. Data Sci.*, vol. 1, no. 2, pp. 45–60, 2014.
- [11] B. K. Sharma, “Stock price volatility in Nepal Stock Exchange: A statistical analysis,” *J. Nepal Financ. Stud.*, vol. 10, no. 1, pp. 23–34, 2020.
- [12] K. Thapa, “Market efficiency and news impact in the Nepal Stock Exchange,” *Econ. J. Nepal*, vol. 12, no. 3, pp. 15–28, 2021.