



Enhanced Sentiment-Based Stock Market Recommendation System Using Machine Learning

Sanapala Ramesh¹

Department of Computer Science and Systems Engineering,

Andhra University, Visakhapatnam, India¹

Abstract: Stock prices are influenced not only by historical price movement but also by public sentiment expressed through financial news. Retail investors usually consult charting platforms and news platforms separately, with no automated bridge between them. This project presents an implemented, Flask-based web application that combines quantitative technical analysis of stock prices with machine-learning based sentiment analysis of financial news to produce a transparent BUY, HOLD or SELL recommendation.

Historical price data is sourced from Yahoo Finance and used to calculate the 20-day moving average, the 14-period Relative Strength Index (RSI) and the Moving Average Convergence Divergence (MACD). Financial news is gathered from NewsAPI, cleaned with NLTK, and converted into TF-IDF and hand-crafted sentiment features. Three classifiers, namely Random Forest, Linear Support Vector Machine and Multinomial Naive Bayes, are trained, and the best is selected by test-set accuracy. Article-level predictions are aggregated into a daily Sentiment Index, which is combined with a price-trend score using the weighted formula $0.6 \times \text{Sentiment} + 0.4 \times \text{Trend}$. An RSI-based safety filter prevents extreme-market signals from being followed blindly. All data and results are stored in SQLite and displayed on an interactive Chart.js dashboard.

The system was run on 411 stored news articles covering seven US and Indian stocks. On an 80/20 split, the Linear SVM achieved the highest test accuracy (100.00%), followed by Random Forest (98.80%) and Naive Bayes (79.52%), and was selected automatically. Because the sentiment labels are generated by a financial keyword lexicon rather than by human annotators, these metrics measure agreement with that labelling heuristic and not validated real-world sentiment accuracy. This limitation is noted explicitly throughout the report.

Keywords: stock market, sentiment analysis, technical indicators, RSI, MACD, TF-IDF, Random Forest, SVM, Naive Bayes, Flask.

1. INTRODUCTION

1.1 Background

Financial markets are reactive to information. Earnings reports, corporate news and the overall market mood can influence prices over the short-term, independent of the long-term fundamentals of a company. Traditional algorithmic approaches use mostly price-derived technical indicators such as moving averages and momentum oscillators to trade; these capture trend and momentum, but not the qualitative information held in news text.

With the rise of Natural Language Processing (NLP) and freely available market and news APIs comes an opportunity to quantify the sentiment of financial news automatically and combine it with technical signals. This project implements such a system as a complete web application, from data acquisition to a stored, visualised recommendation.

1.2 Problem Statement

Investors are faced with two information streams: technical charting tools showing price trends and news platforms showing headlines. There is no simple, automated, explainable system that reads both, classifies news sentiment using



machine learning, checks it against price trend and overbought/oversold conditions, and produces a single risk-aware recommendation. This project fills that gap.

1.3 Objectives

- Obtain historical OHLCV stock data and calculate technical indicators, such as 20-day moving average, RSI, and MACD.
- Collect recent news for a stock and preprocess the text data using NLP techniques.
- Train and evaluate three classifiers (Random Forest, Linear SVM, and Multinomial Naive Bayes) on the news dataset for sentiment analysis.
- Aggregate sentiment scores at an article level to calculate a daily Sentiment Index for each stock.
- Combine the daily Sentiment Index with the price trend to formulate a recommendation (BUY/HOLD/SELL), where the recommendation is further refined using the RSI indicator to assess risk.
- Evaluate the performance of the trained classifiers using accuracy, precision, recall, F1-score, and a confusion matrix.
- Store all results in a SQLite database and visualize them on a webpage.

1.4 Scope of the Project

The system analyses one stock symbol at a time at daily granularity. The user can enter the company name, or ticker, and common aliases (for example GOOGLE to GOOGL, APPLE to AAPL, TATA to TCS.NS) are resolved automatically. Both US and Indian (NSE) tickers have been analysed. The system gives advisory recommendations only; it does not execute trades or integrate with brokerage services.

2. LITERATURE REVIEW

2.1 Technical Analysis

Technical analysis assumes that historical price and volume information contains information about future price behavior. Moving averages smooth out short-term fluctuations to show the direction of a trend. The Relative Strength Index (RSI) was developed by J. Welles Wilder and measures changes in price levels to identify when an asset is overbought or oversold. The Moving Average Convergence Divergence (MACD) uses exponential moving averages to determine changes in momentum. All indicators are relatively cheap to calculate and widely adopted, which makes them the technical foundation of this project.

2.2 NLP and Machine Learning for Financial Sentiment

Financial text data is usually concise and highly dependent on the context, e.g., "beat estimates" is positive, while "missed estimates" is negative. The bag-of-words and TF-IDF representations remain relevant for such texts since they are easily interpretable and computationally light, especially with a moderate amount of data. Naive Bayes, Support Vector Machines and Random Forests are classic classifiers for text data and were therefore selected for comparison.

2.3 Hybrid Technical-Sentiment Systems

Combining of sentiment and technical indicators generally proved to be more beneficial than using them separately: sentiment can confirm or deny the technical signal. This project implements this idea through a weighted combination of the two signals and an RSI filter to prevent sentiment from overpowering extreme technical values.

2.4 Research Gap

End-to-end, reproducible and explainable pipelines which combine technical indicators and news sentiment into one traceable recommendation, with persistent storage and an interactive dashboard, are uncommon in openly accessible



tools. This project aims to fill this gap by engineering the fusion formula, the safety filter and the storage and visualisation layers around well-established components.

3. SYSTEM ANALYSIS

3.1 Existing System

Most of the existing platforms offer either charts with technical indicators or news. The sentiment analysis is done manually, while there is no system that combines the two into one automatically, with a particular emphasis on the final score as a black-box solution.

3.2 Proposed System

The proposed system will automatize both processes, while combining them using a particular formula (not black-box), which means that every recommendation will be based on sentiment score, price trend score, and RSI value retrieved from the database

Platform Category	Technical Indicators	News Sentiment	Automated Fusion	Explainable
Charting platforms	Yes	No	No	N/A
News aggregators	No	Manual reading	No	N/A
Generic robo-advisors	Partial	Rare	Partial (opaque)	Usually no
Proposed system	Yes (MA-20, RSI, MACD)	Yes (ML-classified)	Yes (documented formula)	Yes

Table 3.1: Comparison with existing platform categories

3.3 Feasibility Study

Technical feasibility: Yahoo Finance (yfinance) and NewsAPI provide the data within the development scope, and all the models can be found in sklearn module.

Economic feasibility: The whole product will be based on free API usage and open source software, making it cost-effective.

Operational feasibility: The product's design is modular, which means that each part can be developed, tested, and replaced separately, and the interface is web-based, which does not require any specific knowledge from an end-user.

4. SYSTEM ARCHITECTURE

4.1 High-Level Architecture

The application follows a pipeline architecture. User input passes through the Flask application to two parallel branches, a technical branch and a sentiment branch, which are merged by the recommendation engine. Results are stored in SQLite and displayed on the dashboard.

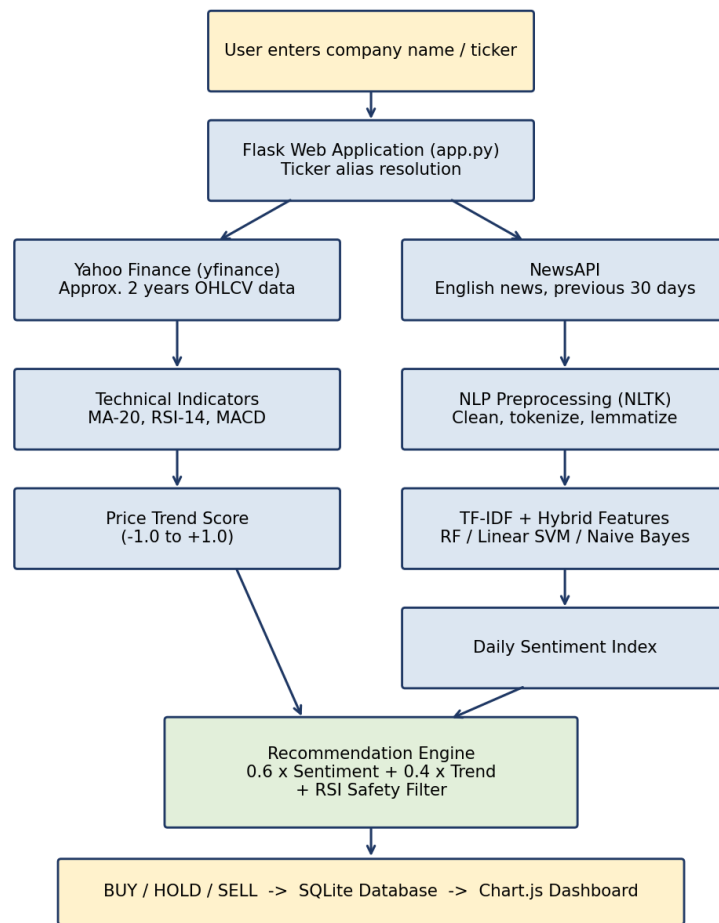


Figure 4.1: System architecture and data flow

4.2 Module Descriptions

Module	Key Component	Responsibility
1. Stock Data	yfinance, technical_indicators.py	Downloads about two years of OHLCV data and computes daily return, MA-20, RSI, MACD, MACD signal and histogram.
2. News Collection and Preprocessing	NewsAPI, NLTK	Retrieves English news for the previous 30 days and cleans the text (URL removal, lowercasing, tokenization, stopword removal, lemmatization).
3. ML Classification	sentiment_model.py	Builds hybrid features, trains Random Forest, Linear SVM and Naive Bayes, and selects the best model by test accuracy.
4. Sentiment Index	Aggregation logic	Converts article labels (+1 / 0 / -1) into a daily weighted sentiment score.
5. Recommendation Engine	recommender.py	Computes the price-trend score, combines it with sentiment, applies thresholds and the RSI safety filter.
6. Evaluation	evaluator.py	Computes Accuracy, Precision, Recall, F1-score, Confusion Matrix and per-class report.



Module	Key Component	Responsibility
7. Database	models.py (Flask-SQLAlchemy, SQLite)	Stores stock data, news, sentiment index and recommendations with duplicate protection.
8. Web Application	app.py, HTML / CSS / JS, Chart.js	Handles routes, runs the analysis pipeline and renders the dashboard.

Table 4.1: System modules

4.3 End-to-End Processing Flow

1. User submits company name or ticker, aliases resolved to valid ticker.
2. Yahoo Finance data for approx. 2 years is downloaded.
3. Daily return, MA-20, RSI (14), MACD, MACD signal and histogram are calculated for all stock records.
4. Stock records are saved to SQLite, ensuring no duplicates are saved.
5. NewsAPI provides up to 100 English articles from the previous 30 days.
6. Title and description text are combined and preprocessed using NLTK.
7. Financial keyword lexicon assigns initial Positive / Negative / Neutral labels.
8. TF-IDF and custom hand crafted features are created, 3 classifiers are trained.
9. The highest performing classifier based on test accuracy is saved.
10. The sentiments of all articles are then aggregated into a daily Sentiment Index.
11. A price trend score is calculated based on recent movements of the stock.
12. Sentiment and trend are weighted and combined, final result filtered through RSI safety score.
13. Final recommendation is saved and presented in the dashboard.

5. METHODOLOGY

5.1 Stock Data and Technical Indicators

Historical information about the stock's OHLCV is collected using the yfinance library. The indicators that are calculated are the following:

Daily Return: close price percentage change between two consecutive days. Next, there are two Moving Average indicators:

20-day Simple Moving Average:

$$MA20(t) = (1/20) \times \text{sum of Close from day } (t-19) \text{ to day } t$$

Relative Strength Index for period 14, which is an oscillator fluctuating between 0 and 100. Initially missing values are replaced with 50:

$$RSI = 100 - [100 / (1 + RS)], RS = \text{Average Gain} / \text{Average Loss}$$

MACD (fast EMA 12, slow EMA 26, signal EMA 9):

$$MACD = EMA(12) - EMA(26); \text{Signal} = EMA(9) \text{ of } MACD; \text{Histogram} = MACD - \text{Signal}$$

Example of a working case: if the average gain is 1.8 and the average loss is 1.2, then $RS = 1.5$ and $RSI = 100 - 100/2.5 = 60$, which is a neutral value.

5.2 News Collection

News is gathered from NewsAPI using the query of 'company name OR ticker stock market', filtering only English results, ordering by relevancy, searching in the last 30 days, and limiting 100 articles per request. Fields that were stored



are title, description, source, URL, publication date, raw text, cleaned text, sentiment label and sentiment score. The URL is unique, thus avoiding the gathering of duplicate articles.

5.3 Text Preprocessing

Title and description are merged into one raw text and processed with NLTK in this order: URL removal, lowercasing, special-character removal, tokenization, stopwords removal, WordNet lemmatization and removal of very short tokens.

Stage	Text
Original	Microsoft shares surged today after strong earnings!
Cleaned	microsoft share surge today strong earning

Table 5.1: Example of preprocessing

5.4 Sentiment Labelling and Feature Extraction

Initial sentiment labels (Positive, Negative, Neutral) are assigned by a financial keyword lexicon. These labels serve as training targets for the classifiers. The feature matrix combines two groups of features:

- TF-IDF word and bigram features, computed as $TF(w,d) \times \log(N / DF(w))$.
- Hand-crafted features: positive keyword count, negative keyword count, total sentiment keyword count, absolute sentiment score and total word count.

5.5 Machine Learning Models

Model	Configuration	Rationale
Random Forest	400 trees, random_state = 42, class_weight = balanced_subsample	Robust to noise and non-linear feature interactions; ensemble voting reduces overfitting.
Linear SVM	LinearSVC, class_weight = balanced, max_iter = 10000	Strong on high-dimensional sparse text features.
Multinomial Naive Bayes	MultinomialNB (default parameters)	Very fast probabilistic baseline for text classification.

Table 5.2: Classifiers used

All three models are trained on the same 80/20 train-test split. The model with the highest test accuracy is selected and saved with joblib (trained_model.pkl and tfidf.pkl), together with a metadata file recording the model version (hybrid_features_v2). If no trained model is available, the application falls back to a lexicon-based sentiment prediction.

5.6 Sentiment Index

Each article is mapped to +1 (Positive), 0 (Neutral) or -1 (Negative) and averaged per day:

Sentiment Score = (Positive - Negative) / Total articles

Example: of 10 articles, 6 Positive, 2 Neutral and 2 Negative give $(6 - 2) / 10 = 0.4$, a positive sentiment.

5.7 Price Trend Score

The trend score ranges from about -1.0 (strongly falling) to +1.0 (strongly rising). It combines the current price relative to the 20-day moving average, the latest daily return and the direction of the last three trading days. The trend is classified as rising, falling or neutral.

5.8 Recommendation Generation

Combined Score = $0.6 \times \text{Sentiment Score} + 0.4 \times \text{Price Trend Score}$



Condition	Recommendation
Combined Score > 0.15	BUY
Combined Score < -0.15	SELL
Otherwise	HOLD

Table 5.3: Decision thresholds

RSI safety filter: an RSI above 75 (overbought) changes a BUY to HOLD, and an RSI below 25 (oversold) changes a SELL to HOLD. When an override occurs the confidence is set to 0.35. The confidence value is a rule-based score and not a statistically calibrated probability.

Scenario	Sentiment	Trend	Combined	RSI	Result
A	0.40	0.20	0.32	58	BUY
B	-0.55	-0.10	-0.37	45	SELL
C	0.55	0.60	0.57	82	HOLD (override)

Table 5.4: Illustrative recommendation scenarios (hypothetical values used to demonstrate the formula)

5.9 Evaluation Methodology

Each classifier is evaluated on the held-out test set using Accuracy, Precision, Recall, F1-score, Confusion Matrix and a per-class classification report, implemented in evaluator.py.

6. SYSTEM DESIGN

6.1 Database Schema

The schema is implemented with Flask-SQLAlchemy in models.py and stored in SQLite (database/db.sqlite3). It contains four tables.

Table	Main Fields	Constraint
stocks	symbol, date, open, high, low, close, volume, daily return, MA-20, RSI, MACD, MACD signal, MACD histogram	Unique (symbol, date)
news	symbol, title, description, source, URL, published date, raw text, cleaned text, sentiment label, sentiment score	Unique URL
sentiment_index	symbol, date, positive count, negative count, neutral count, total articles, weighted sentiment score	One record per symbol per day
recommendations	symbol, date, sentiment score, price trend, recommendation, confidence, close price, moving average, RSI	Unique (symbol, date)

Table 6.1: Database tables

6.2 Data Flow

- External entity: the investor enters a company name or ticker.
- Process 1: the system downloads stock data and news for the ticker.
- Process 2: technical indicators and news sentiment are computed in parallel branches.
- Process 3: both signals are fused into a recommendation.
- Data store: intermediate and final results are written to SQLite.
- External entity: the investor receives the recommendation and charts on the dashboard.



6.3 Application Routes

Route	Purpose
/	Home page
/dashboard/<symbol>	Stock dashboard with charts
/analyze	Runs the complete analysis pipeline
/train	Trains the sentiment models
/update	Fetches new stock and news data
/api/stock/<symbol>	Stock data as JSON
/api/sentiment/<symbol>	Sentiment data as JSON
/api/recommendation/<symbol>	Recommendation as JSON
/documentation	Project documentation page

Table 6.2: Main routes and APIs

6.4 Use Cases

Actor	Use Case
Investor	Request analysis and recommendation for a stock symbol
Investor	View price, moving average, RSI and sentiment charts and latest news
Investor	View recommendation history
Administrator	Train or retrain the sentiment models and update data

7. IMPLEMENTATION DETAILS

7.1 Technology Stack

Layer	Technology
Language	Python 3
Web framework	Flask, Flask-SQLAlchemy
Database	SQLite
Stock data	Yahoo Finance via yfinance
News data	NewsAPI
NLP	NLTK (tokenization, stopwords, WordNet lemmatization)
Machine learning	scikit-learn: TF-IDF, Random Forest, LinearSVC, Multinomial Naive Bayes; joblib for persistence
Frontend	HTML templates, CSS, JavaScript, Chart.js

Table 7.1: Technology stack

7.2 Key Source Files

File	Role
app.py	Application entry point, routes and analysis pipeline
models.py	Database models (stocks, news, sentiment_index, recommendations)



File	Role
technical_indicators.py	Daily return, MA-20, RSI, MACD calculations
sentiment_model.py	Feature creation, training and selection of classifiers
recommender.py	Price-trend score, combined score, thresholds, RSI filter
evaluator.py	Accuracy, Precision, Recall, F1, confusion matrix, classification report

Table 7.2: Key source files

7.3 Dashboard

The dashboard is built with HTML templates and Chart.js and shows the stock closing price with the 20-day moving average, the RSI, the sentiment trend, the latest news articles and the recommendation history.

8. RESULTS AND DISCUSSION

8.1 Dataset Information

Parameter	Value
Stock data source	Yahoo Finance (yfinance), approximately two years per ticker
News data source	NewsAPI, English, relevancy sorted, previous 30 days, up to 100 articles per request
Stocks analysed	GOOGL, JPM, TSLA, AMZN, INTC (US); TCS.NS, WIPRO.NS (India, NSE)
Total news samples	411
Train / test split	80% / 20% (random_state = 42): 328 training and 83 test samples
Labels	Generated by a financial keyword lexicon (Positive / Neutral / Negative)

Table 8.1: Dataset summary

8.2 Stored Data

The following record counts were verified directly from the project database (database/db.sqlite3, approximately 1.28 MB).

Table	Records
Stocks	3,509
News	411
Sentiment index	122
Recommendations	7

Table 8.2: Database contents

8.3 Saved Model Artifacts

Artifact	Details
trained_model_v2.pkl	Current trained classifier (Linear SVM), saved after the latest training run
tfidf.pkl	TF-IDF vectorizer, regenerated during the latest training run
model_meta.json	Model version: hybrid_features_v2

Table 8.3: Saved model artifacts



8.4 Model Performance

The training pipeline was executed on the 411 stored news records, split 80/20 into 328 training and 83 test samples (random_state = 42). Table 8.4 gives the metrics reported by the evaluation module on the held-out test set.

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	98.80%	98.83%	98.80%	98.80%
Linear SVM	100.00%	100.00%	100.00%	100.00%
Naive Bayes	79.52%	68.85%	79.52%	72.71%

Table 8.4: Classifier performance on the held-out test set (83 samples)

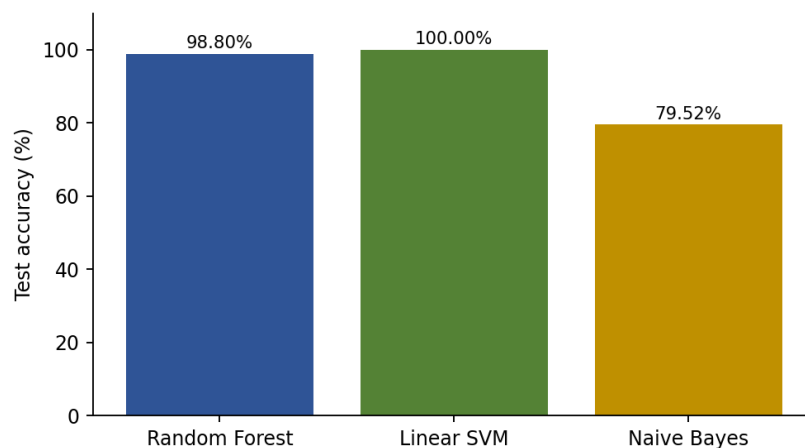


Figure 8.1: Test accuracy of the three classifiers

The Linear SVM achieved the highest accuracy and was selected automatically as the production classifier. It was saved as trained_model_v2.pkl together with the regenerated TF-IDF vectorizer, under model version hybrid_features_v2.

8.5 Discussion

The pipeline is end to end on both US and Indian stocks, from data acquisition to a stored recommendation displayed on the dashboard. Duplicate protection, the RSI safety filter and the fallback sentiment predictor make repeated runs and missing-model situations safe.

The Linear SVM achieved the best result, with Random Forest close behind. This is consistent with the known strength of linear models on high-dimensional sparse TF-IDF features and of tree ensembles on mixed feature sets. Naive Bayes was clearly weaker; its precision (68.85%) is well below its accuracy (79.52%), which suggests that it handles the minority sentiment classes poorly on this dataset. The per-class classification report can be used to confirm this.

An important point concerns interpretation. The training labels come from a financial keyword lexicon, and a portion of the feature set (keyword counts and sentiment score) is derived from the same. Thus, the reported accuracy, including the 100.00% of the Linear SVM, reflects how well the classifiers reproduce the labelling heuristic. It is not against an



independent, human-labelled financial sentiment data, and should not be read as proof of real-world sentiment accuracy or of profitable trading. The test set of 83 samples is also small.

During training the Linear SVM raised a scikit-learn convergence warning (Liblinear failed to converge). The model completed and produced the reported results, but feature scaling or a larger iteration limit should be explored to remove the warning. Validation against an independent, human-annotated dataset is left as future work.

9. TESTING STRATEGY

9.1 Test Scenarios and Expected Behaviour

Each module can be tested in isolation with known inputs, and the complete pipeline can be verified by running it on sample tickers and checking that intermediate values are stored correctly. The table lists representative scenarios and the behaviour the implementation is designed to show.

ID	Scenario	Expected Behaviour
TC-1	Same stock symbol and date fetched twice	Duplicate row is not inserted (unique symbol + date)
TC-2	Same news URL retrieved again	Article is skipped (unique URL)
TC-3	RSI in the first periods with insufficient history	Missing values set to 50
TC-4	Combined score above 0.15 with RSI above 75	Recommendation forced to HOLD, confidence 0.35
TC-5	Combined score below -0.15 with RSI below 25	Recommendation forced to HOLD, confidence 0.35
TC-6	Combined score exactly 0.15	HOLD, because the BUY rule is strictly greater than 0.15
TC-7	Input "GOOGLE" or "APPLE"	Resolved to GOOGL and AAPL
TC-8	Trained model files not available	Fallback lexicon-based sentiment prediction is used

Table 9.1: Test scenarios

9.2 Verified Test Runs

The following checks were executed on the implemented application.

Check	Result
Database initialisation	Completed successfully
Home route (GET /)	HTTP 200, content type text/html, 13,977 bytes
Model training on 411 stored news records	Completed; Linear SVM selected (100.00% test accuracy)

Table 9.2: Verified runs

During testing, the installed Werkzeug version did not expose the `__version__` attribute expected by the Flask test client. This was handled in the test harness only, without changing project files; pinning compatible dependency versions is recommended.



10. ADVANTAGES, LIMITATIONS AND FUTURE SCOPE

10.1 Advantages

- Full end-to-end Web application with persistent storage and protection against duplicates.
- Technical + sentiment information fusion into an interpretable formula-based recommendation.
- Safe RSI safety filter to avoid drawdowns in extreme market conditions.
- Three classifiers comparison and voting for the best performer.
- Hybrid TF-IDF + keyword features and fallback prediction if no model is available.
- Modular architecture and low computational footprint with free-tier data sources.
- Prices, indicators, sentiment and history visualization in interactive Chart.js dashboards.

10.2 Limitations

- Sentiment labels come from a keyword lexicon, not human annotation, hence accuracy is measured against this heuristic.
- Metrics and the winning model name are returned at training-time, but not saved in the model metadata file.
- Linear SVM warns of convergence, and the test set is small (83 samples)
- We use a single random 80/20 split rather than cross-validation
- NewsAPI and Yahoo Finance are live data sources that can vary between runs, and rate-limiting may apply
- News is summarized by title/description, not full articles
- The confidence score is heuristic and not a calibrated probability
- No backtesting is implemented, therefore trading profitability, Sharpe ratio and drawdown are not measured
- Recommendations are for informational purposes only, not financial advice or guarantees

10.3 Future Scope

- Evaluate on an independent, human-annotated financial sentiment dataset.
- Add cross-validation and store evaluation metrics and the winning model in the metadata file.
- Move from TF-IDF to transformer-based contextual embeddings.
- Add a backtesting module to measure historical performance of the recommendations.
- Extend to multi-stock portfolios and social media sentiment sources.
- Calibrate the confidence score and add live alerts.

11. CONCLUSION

This project implemented a complete hybrid stock-analysis system consisting of a Flask web application, Yahoo Finance market data, NewsAPI financial news, NLTK preprocessing, TF-IDF and hand-crafted features, Random Forest, Linear SVM and Naive Bayes classifiers, technical indicators, a Sentiment Index, a rule-based recommendation engine with an



RSI safety filter, SQLite persistence and a Chart.js dashboard. The system has been run on multiple US and Indian stocks and stores its data and recommendations for traceability.

Training on 411 news records selected the Linear SVM (100.00% test accuracy) ahead of Random Forest (98.80%) and Naive Bayes (79.52%). The system is best described as a transparent decision-support tool: these figures reflect agreement with a keyword-based labelling heuristic, so future work should focus on independent validation and backtesting before any claim about real-world predictive accuracy is made.

REFERENCES

- [1]. J. W. Wilder, New Concepts in Technical Trading Systems. Trend Research, 1978.
- [2]. yfinance: Yahoo Finance market data downloader for Python. <https://pypi.org/project/yfinance/>
- [3]. NewsAPI Documentation. <https://newsapi.org/docs>
- [4]. scikit-learn User Guide: Ensemble Methods, Support Vector Machines, Naive Bayes, Text Feature Extraction and Model Evaluation. https://scikit-learn.org/stable/user_guide.html
- [5]. NLTK Documentation: tokenization, stopwords and WordNet lemmatization. <https://www.nltk.org/>
- [6]. Flask Documentation. <https://flask.palletsprojects.com/>
- [7]. Flask-SQLAlchemy Documentation. <https://flask-sqlalchemy.palletsprojects.com/>
- [8]. SQLite Documentation. <https://www.sqlite.org/docs.html>
- [9]. Chart.js Documentation. <https://www.chartjs.org/docs/>