Aistox : Sentiment-Aware Stock Market Analysis Platform

So, we are building a platform for stock market prediction, before predicting we have to understand what factors influence any stock price.

The official GitHub repo is :https://github.com/Notnaut77/Aistox-Sentiment-Aware-Stock-Market-Analysis-Platform.git

Factors Influencing Stock Price

(reference: Documents/Factorsaffectinginidnanstockmkt.pdf)

---

1. Fundamental Factors

- Earnings Per Share (EPS): Reflects a company's profitability and core valuation.
- Revenue Growth: Indicates business expansion and future earning potential.
- Balance Sheet Strength: A healthy ratio of assets to liabilities signals financial stability.
- Dividend Policy & Yield: Attracts long-term investors and indicates cash flow reliability.
- Price-to-Earnings (P/E) Ratio: Measures market expectations vs actual earnings.
- Company Performance: Frequently cited by investors as a major market mover.
- Government Policies: Business-friendly or hostile policies directly affect company earnings.
- Number of IPOs/New Issues: High IPO activity may dilute capital; low activity may indicate market stagnation.
- Dividend-Earnings Ratio: Reflects shareholder return vs retained profits.
- Scams/Frauds: Corporate frauds (e.g., Satyam) can trigger massive price crashes.

---

2. Technical Factors

- Price Trends: Bullish or bearish movements based on chart patterns.
- Volume: Confirms the strength of a price movement; high volume often indicates strong sentiment.
- Support and Resistance Levels: Key price thresholds where price action reverses or consolidates.
- Moving Averages (e.g., 50-day, 200-day): Smooth price trends; crossovers often signal momentum shifts.
- RSI (Relative Strength Index): Measures overbought/oversold conditions.
- MACD (Moving Average Convergence Divergence): Detects momentum and trend reversals.
- Trade Volume in Commodities: Linked to stocks in energy, metals, and agriculture sectors.

---

3. Macroeconomic & External Factors

- Foreign Institutional Investors (FII) Flow: Highest-scoring factor (87.37%)—FII inflow = bullish; outflow = bearish.
- GDP Growth Rate: Strong GDP correlates with increased investor confidence and earnings.
- Political Stability: Affects long-term investment decisions and regulatory predictability.
- Inflation Rate: High inflation reduces profits and consumer purchasing power.
- Liquidity Conditions: Availability of money in the economy affects investment flows.
- Interest Rates: High rates make bonds attractive, pulling money out of equities.
- Cash Reserve Ratio (CRR): Influences liquidity via central bank policy; scored 76.14%.
- Exchange Rates: Affects imports/exports and revenue of globalized companies.

- Oil Prices: Significant cost input; affects transportation, manufacturing, and inflation—scored 77.54%.
- Global Financial Crises (e.g., Subprime, 2008): Lead to capital outflows, recession, and panic selling—scored 76.14%.
- Government Debt Levels: High debt may affect future taxation and investor sentiment.
- Fiscal and Monetary Policies: Central bank and government actions influence the macro climate.
- Regulatory Changes: E.g., banking norms, environmental compliance, pharma approvals.

---

4. Market Sentiment & Behavioral Factors

- News & Media Coverage: Can cause immediate sentiment shifts and price volatility.
- Social Media Trends: Platforms like Twitter, Reddit (e.g., $GME) influence retail investor behavior.
- Herd Behavior: Group psychology leads to overreactions, bubbles, or panic crashes.
- FOMO (Fear of Missing Out): Leads to irrational buying at high valuations.
- Institutional Investor Activity: Large trades by mutual funds, FIIs, or pension funds can shift prices rapidly.
- Rumors & Speculation: Cause sharp short-term price movements despite fundamentals.
- Astrology (as per respondents): Cited by 36%—reflects superstitious sentiment, especially among retail traders.

---

5. Sectoral & Industry-Specific Factors

- Commodity Prices: Direct impact on sectors like oil, steel, agriculture, etc.
- Technological Innovation: Drives disruption and stock revaluation in tech, biotech, etc.
- Sectoral Cycles: Real estate, auto, IT, etc., have cyclic patterns based on economic phases.
- Regulatory Actions: Industry-specific rules can boost or break valuations (e.g., pharma, banking).
- Export Dependency: Export-heavy companies are vulnerable to global demand and forex shifts.

Most Influential Factors (From Survey Data in Study)

- FII Flow: 87.37% impact score.
- GDP Growth: 81.75% impact.
- Political Stability: 79.65%.
- Oil Prices: 77.54%.
- Liquidity: 76.84%.
- Cash Reserve Ratio: 76.14%.
- Subprime Crises (Global): 76.14%.

To effectively predict stock price movements in a
Sentiment-Aware Stock Market Analysis Platform, it is essential to integrate both **quantitative financial indicators** and **qualitative sentiment signals**. While fundamental and macroeconomic variables like **GDP growth**, **FII flows**, and **interest rates** provide long-term directional insight, short-term volatility is often driven by **news cycles**, **social media sentiment**, and **investor psychology**. Therefore, the program must incorporate real-time data streams from **financial news outlets**, **Twitter feeds**, **Reddit discussions**, and **institutional filings**. Applying **Natural Language Processing (NLP)** to extract sentiment polarity and intensity from this unstructured data will allow the model to capture market mood dynamics. Additionally, integrating **volume analysis**, **technical indicators**, and **sector-specific triggers** will strengthen the system's ability to detect actionable patterns. By fusing these diverse data layers, the platform can move from reactive analytics to proactive forecasting, making it both robust and adaptive in the face of ever-evolving market conditions.

Therefore following has to be integrated in the AI model in order to predict the prices more efficiently,

1. Real-Time Sentiment Data

- [ ] **News Feed Parser** (e.g., Reuters, Bloomberg, Economic Times API)
- [ ] **Twitter API Integration** (using hashtags, cash tags like `$TCS`, `$INFY`)
- [ ] **Reddit Sentiment Tracker** (e.g., r/IndiaInvestments, r/StockMarket)
- [ ] **Google Trends Integration** (keyword search interest spikes)

---

2. Natural Language Processing (NLP) Tools

- [ ] **Sentiment Classification** (Positive / Neutral / Negative)
- [ ] **Emotion Detection** (Fear, Greed, Panic, etc.)
- [ ] **Named Entity Recognition (NER)** to extract company names, CEOs, tickers
- [ ] **Topic Modeling** to track evolving narratives (e.g., budget, war, crisis)

---

3. Financial & Technical Indicators

- [ ] **Stock Price Time Series Data** (Open, High, Low, Close, Volume)
- [ ] **Moving Averages (SMA/EMA)**
- [ ] **RSI, MACD, Bollinger Bands**
- [ ] **Volatility Index (VIX) if available**

---

4. Fundamental Data Sources

- [ ] **Quarterly Earnings Reports (EPS, Revenue, Profit Margins)**
- [ ] **Balance Sheet Metrics (Debt, Assets, Book Value)**
- [ ] **Dividend Announcements**
- [ ] **P/E Ratio & PEG Ratio**

---

5. Macroeconomic Indicators

- [ ] **GDP Growth Rate**
- [ ] **Inflation Data (CPI/WPI)**
- [ ] **Interest Rates / RBI Policy Updates**
- [ ] **Oil & Commodity Prices**
- [ ] **Exchange Rates (USD/INR)**

---

6. Global & Geopolitical Feeds

- [ ] **International Indices (NASDAQ, S&P 500, Hang Seng)**
- [ ] **Geopolitical Event Alerts (e.g., war, elections, trade sanctions)**
- [ ] **Foreign Institutional Investment (FII) Flow Tracker**

---

7. Sectoral Monitoring

- [ ] **Sector-specific Sentiment & News**
- [ ] **Commodity Tracker (Crude oil, Gold, etc.)**
- [ ] **Regulatory Announcements (SEBI, RBI, etc.)**

---

8. Behavioral Signals

- [ ] **Retail Participation Indicators (trading app downloads, brokerage data)**
- [ ] **Google Search Trends for specific stocks**
- [ ] **Unusual Volume/Volatility Alerts**

☐ **Insider Trading / Bulk Deals Monitor**

Overall Project Flow

Data Collection → Preprocessing → Sentiment Analysis → Feature Engineering → Prediction → Visualization

Video below greatly served as a reference for me :)

https://youtube.com/playlist?list=PLBqhYPan65gjLFkhVcXRLMzeiGUkNhKIU&si=iQS04rKrPfhajjKy

Step 1: Data Collection

Task ID	Task	Source	Tools/Libraries
1.1	News Feed Parser	Economic Times, Mint, Business Standard, Reuters	`feedparser` , `newspaper3k` , `requests` , `bs4`
1.2	Twitter API Integration	Twitter	`tweepy` (or `snscrape` for no-auth scraping)
1.3	Reddit Sentiment Miner	r/IndiaInvestments, r/StockMarket, etc.	`PRAW` or `Pushshift API` , `psaw`

Task 1.1 : News Feed Parser

Objective: To collect structured, full-length financial news articles from Indian and global sources using RSS feeds and web scraping. The data will later support NLP-based sentiment analysis in the Aistox platform.

▼ Code and Architecture

Architecture & Flow

1. **Input:**

   A list of RSS feed URLs from top financial sources like Economic Times, Livemint, and Reuters.

2. **RSS Parsing:**

   The script uses the `feedparser` library to retrieve headlines, links, and publication dates from these RSS feeds.

3. **Article Extraction:**

   For each news link, the script employs the `newspaper3k` library to:

   - Download the full article
   - Parse the HTML content
   - Extract the clean, readable article text and metadata

4. **Data Structuring:**

   Each article is stored as a Python dictionary containing:

   - `title`
   - `url`
   - `published` date
   - `source` feed
   - `content` (full article text)
   - `scraped_at` (timestamp of data collection)

5. **Output:**

   All structured articles are saved as a JSON file at:

```
Data_collector/News/news_articles.json
```

🔧 Technologies Used

Component	Tool/Library
RSS Feed Parsing	`feedparser`
Article Extraction	`newspaper3k`
Data Serialization	`json`
Date Handling	`datetime`
Directory Management	`os`

📁 Output Format

Each JSON entry has the structure:

```
{
  "title": "Nifty ends higher as IT stocks rally",
  "url": "https://economictimes.indiatimes.com/.../articleshow/12345678.cms",
  "published": "Wed, 19 Jun 2025 15:23:00 GMT",
  "source": "https://economictimes.indiatimes.com/rss/markets/rssfeeds/1977021501.cms",
  "content": "The Indian benchmark indices closed higher...",
  "scraped_at": "2025-06-19T18:25:41.228Z"
}
```

Current Status

- Collects real-time financial news
- .,ṁSupports Indian + Global sources
- Stores articles in a structured JSON format

To Do: Store in MongoDB and filter by keywords (Phase 1 extensions)

Task 1.2 : Twitter API Integration

Objective

To collect real-time tweets related to the Indian stock market and specific financial instruments using the Twitter API.

▼ Code and Architecture

Architecture Diagram

```
+-----------------------+
|  Twitter Developer   |
|       Portal        |
| (App + Project Setup) |
+-----------+-----------+
            |
            v
+-----------------------+
|  Python Stream Script  |
| (tweepy.StreamingClient|
|   in stream_tweets.py) |
```

```
+-----------+------------+
          |
   Twitter Streaming API v2
          |
          v
+------------------------+
|  Rule-Based Filtering |
| ($TCS, #StockMarket,  |
|  $INFY, etc.)         |
+-----------+------------+
          |
          v
+------------------------+
| Tweet Handler         |
| (on_tweet method)     |
| Parses tweet text,    |
| timestamp, ID         |
+-----------+------------+
          |
          v
+------------------------+
| JSON Output Storage   |
| tweets_stream.json    |
| (one tweet per line)  |
+-----------+------------+
          |
          v
+------------------------+
| Phase 2: NLP Pipeline |
| Sentiment Analysis,   |
| Topic Modeling, NER   |
+------------------------+
```

Key Components

Component	Description
Twitter Developer App	Provides the Bearer Token used for authenticated streaming
StreamingClient (tweepy)	Establishes persistent connection to Twitter's filtered stream endpoint
Streaming Rule	Custom filter: `"$TCS OR $INFY OR #StockMarket OR #Nifty50"`
Tweet Parser	Converts incoming tweets into structured format (ID, content, timestamp)
Data Storage	Writes parsed tweets into `tweets_stream.json`
Downstream Use	Prepares data for NLP-based sentiment scoring, named entity recognition

Current Status

- Implemented real-time tweet collection using Twitter API v2 with streaming rules for financial keywords like `$TCS`, `$INFY`, and `#StockMarket`.
- Utilized `tweepy.StreamingClient` to ingest and store structured tweet data (ID, text, timestamp) in JSON format.
- Established a modular ingestion system to feed behavioral sentiment signals into the Aistox prediction pipeline.

Task 1.3 : Reddit Integration

Objective

To collect Reddit discussions relevant to the Indian stock market from targeted subreddits using the Reddit API and extract keyword-specific posts for sentiment analysis.

▼ Code and Architecture

Architecture

- **Input Layer**: Pulls data from Reddit using the `praw` API wrapper.
- **Filtering Layer**: Filters relevant posts using finance-specific **keywords** across curated **subreddits**.
- **Processing Layer**: Extracts structured data fields (title, self-text, score, timestamp, URL).
- **Storage Layer**: Saves all relevant posts in **JSON format** for downstream sentiment and topic modeling.

Current Status

- Implemented Reddit data collection from selected stock-related subreddits.
- Filtered and stored posts containing relevant financial keywords.
- Data saved in `reddit_posts.json` for downstream processing and NLP tasks in the Aistox pipeline.

▼ **Additional Data Sources to Add In Future**

- **YouTube** – Finance channel transcripts (Groww, Bloomberg, CNBC TV18)
- **TradingView** – User sentiments, chart ideas, comments
- **StockTwits** – Real-time stock chatter
- **Quora** – Finance discussions and opinion trends
- **ValuePickr** – Deep-dive stock analysis forum
- **Moneycontrol** – Company-specific forum and news
- **Economic Times** – Articles and expert analysis
- **Livemint** – Market and policy coverage
- **Investing.com** – Analyst ratings, macro indicators
- **Twitter Spaces** – Voice discussions on finance (stream titles + listeners)
- **Google News API** – General news with filters
- **LinkedIn** – Hiring trends at listed companies
- **SensorTower / AppStore / Play Store** – App download stats for fintechs
- **GDELT** – Global event and media monitoring
- **SEBI / NSE / BSE** – Bulk deals, announcements, filings
- **RBI** – Policy changes, repo rate, circulars
- **World Bank / IMF** – Macro data like GDP, inflation
- **CoinMarketCap** – Crypto trends (optional cross-market signal)
- **Glassdoor** – Employee sentiment and reviews

For now the work for data collection is completed with complete functioning and we'll now move to the task 2 which is Natural Language Processing.

Task 2: Natural Language Processing (NLP)

Subtask	Objective	Tools / Models	Output
2.1 Preprocessing	Clean and normalize text for NLP tasks	`re`, `NLTK`, `spaCy`	Cleaned, tokenized text
2.2 Sentiment Analysis	Identify overall sentiment of the text (positive/neutral/negative)	`VADER`, `TextBlob`, `FinBERT`, `IndicBERT`	Sentiment labels

2.3 Emotion Detection	Detect specific emotions like fear, greed, panic	NRCLex, GoEmotions, transformers	Emotion tags (optional but useful)
2.4 Named Entity Recognition (NER)	Extract named entities like company names, tickers, sectors	spaCy, Flair, HuggingFace Transformers	List of tagged entities
2.5 Topic Modeling	Identify major topics/narratives in the dataset	LDA, BERTopic, Gensim	Topic distributions and keywords

Task 2.1 : Preprocessing

Objective:

To clean and normalize text for NLP tasks. We'll use re, NLTK, spaCy for this purpose and we will get a cleaned and tokenized text.

Following will be our task:

Clean and normalize unstructured text data (from News, Twitter, Reddit) by:

- Lowercasing
- Removing links, punctuation, emojis, hashtags, mentions
- Removing stopwords
- Lemmatization

Old files like news_articles.json will be converted to a cleaned text news_articles_cleaned.json which will be readable and workable.

Task 2.2: Sentiment Classification

Step	Description	Tool	Output Field
2.2.1 Install	Install VADER lexicon and TextBlob dependencies	pip install nltk vaderSentiment textblobpython -m textblob.download_corpora	—
2.2.2 Load Data	Read in each _cleaned.json file	json	—
2.2.3 Score Text	Compute polarity scores (compound, positive, negative, neutral)	vaderSentiment.SentimentIntensityAnalyzerTextBlob	compound, pos, neu, neg
2.2.4 Label Data	Convert compound into discrete labels (POS, NEU, NEG)	simple thresholding	sentiment_label
2.2.5 Save Results	Write out new JSON files with added sentiment scores and labels	json	—

Each output JSON will now include:

```
"vader_compound": 0.6249,
"vader_label": "POS",
"textblob_polarity": 0.35,
"textblob_label": "POS"
```

Task 2.3 — Emotion Detection

Objective

- To identify **emotional context** (e.g., fear, greed, panic) in financial news and Reddit posts using **NLP techniques**.
- Enhance sentiment analysis by adding **psychological emotion tags** that reflect investor behavior.

Tools Used

- **NRCLex**: Python wrapper for the NRC Emotion Lexicon which detects emotions like:
  - *Fear, Anger, Joy, Sadness, Trust, Disgust, Surprise, Anticipation*

What the Code Does

- Loads cleaned text from:
  - `news_articles_cleaned.json`
  - `reddit_posts_cleaned.json`
- Applies NRCLex to extract the **top 3 dominant emotions** from each text entry.
- Adds a new field:
  - `"emotions": ["fear", "anticipation", "trust"]`
- Saves the enriched data to:
  - `news_articles_emotions.json`
  - `reddit_posts_emotions.json`

Task 2.4 — Named Entity Recognition (NER)

Objective

To extract **named entities** such as:

- Company Names (e.g., *Tata Consultancy Services*)
- Stock Tickers (e.g., *TCS*, *INFY*)
- Sector Names (e.g., *Finance*, *IT*)

Approach

We'll use **spaCy's** `en_core_web_sm` **model** for:

- Identifying entities like `ORG` (organization), `GPE` (locations), `PERSON`, etc.
- Annotating existing cleaned text with these tags

We'll process both:

- `news_articles_cleaned.json`
- `reddit_posts_cleaned.json`

Steps

1. Load cleaned JSON files
2. Use spaCy NER to detect entities
3. Filter relevant entity types (`ORG` , `PRODUCT` , `GPE` , etc.)
4. Add `entities` field to each data point
5. Save as:
   - `news_articles_entities.json`
   - `reddit_posts_entities.json`

2.5 Topic Modeling

Objective:

Automatically discover and track **major topics or narratives** across Reddit, Twitter, and News text datasets. This allows clustering of similar documents and tracking of evolving market discourse (e.g., inflation, interest rates, AI hype, crypto crashes).

Techniques & Tools:

Technique	Description	Libraries
LDA (Latent Dirichlet Allocation)	Probabilistic model that assigns documents to topics and topics to words	`Gensim`, `scikit-learn`
BERTopic (Using here)	Transformer-based topic modeling using embeddings + clustering	`BERTopic`, `sentence-transformers`, `UMAP`
NMF (Non-negative Matrix Factorization)	Matrix decomposition method	`scikit-learn`

🧠 Implementation Plan:

1. **Input**: Use `cleaned_text` field from Step 2.1.
2. **Vectorization**:
   - LDA: Use `CountVectorizer` or `TfidfVectorizer`.
   - BERTopic: Use `sentence-transformers` embeddings.
3. **Model Training**:
   - Set number of topics (`n_topics=10–20` as a start).
   - Train model and extract:
     - Topic-word distributions
     - Document-topic mappings
4. **Output Schema**:

```
json
CopyEdit
{
  "text": "Inflation fears rise as Fed signals rate hikes.",
  "topic_id": 3,
  "topic_keywords": ["inflation", "rate hike", "fed", "interest"]
}
```

Output:

as news_articles_topics_bertopic.json and reddit_posts_topics_bertopic.json

- **Topic Distributions**: Probabilities of each topic per document.
- **Top Keywords per Topic**: Most representative words or phrases.
- **Representative Samples**: Best matching documents per topic.

Visualization Tools:

- **pyLDAvis** (for LDA)
- **BERTopic's built-in dashboard**
- **t-SNE/UMAP** plots for embeddings

🔍 Example Topics:

Topic ID	Keywords

0	`["inflation", "fed", "rate hike", "interest"]`
1	`["layoffs", "meta", "tech", "employees"]`
2	`["earnings", "quarter", "profits", "guidance"]`
3	`["elon", "tesla", "twitter", "acquisition"]`

NOW!!!

we'll have few of the files with us

2.1 Preprocessing

Cleaned text without noise (hashtags, emojis, links, etc.)

- `Data_collector/reddit/reddit_posts_cleaned.json`
- `Data_collector/News/news_articles_cleaned.json`

2.2 Sentiment Classification

Sentiment scores and labels (from VADER/TextBlob or fine-tuned models)

- `Data_collector/reddit/reddit_posts_sentiment.json`
- `Data_collector/News/news_articles_sentiment.json`

2.3 Emotion Detection

Detected emotions like fear, greed, optimism (via NRCLex)

- `Data_collector/reddit/reddit_posts_emotions.json`
- `Data_collector/News/news_articles_emotions.json`

2.4 Named Entity Recognition (NER)

Extracted named entities (ORGs, companies, places, products)

- `Data_collector/reddit/reddit_posts_ner.json`
- `Data_collector/News/news_articles_ner.json`

🔷 2.5 Topic Modeling (BERTopic)

Topic IDs and top keywords for each entry

- `Data_collector/reddit/reddit_posts_topics_bertopic.json`
- `Data_collector/News/news_articles_topics_bertopic.json`

Now ending the step 2 we will merge the whole 5 files and will get 2 output files one csv and other json.

Sample CSV Row:

cleaned_text	source	sentiment	emotion	entities	topic_id	topic_keywords
Fed raises interest rates again...	news	{"vader_compound": -0.51, "label": "negative"}	{"fear": 0.7, "anger": 0.2}	[{"text": "Fed", "label": "ORG"}]	4	interest, fed, hike, inflation

Task 3 Overview: Text-to-Stock Feature Engineering Pipeline

We will work through the following subtasks step by step:

Task 3.1: Extract stock symbols from NER entities

Input:	Final merged files: merged.csv
Output:	Add a field `ticker` to each entry (e.g., "TSLA", "AAPL") based on entities

We used auto-match via APIs (e.g., Yahoo Finance search) and an output file as combined_with_ticker.csv is saved.

Task 3.2: Aggregate features by (stock, date)

Input:	Merged JSON with timestamps and tickers
Output:	One row per `(ticker, date)` with features like:

- Avg sentiment
- Emotion ratios
- Topic distribution
- Mention volume

Task 3.3: Fetch stock price data

Use `yfinance` or another API to get:

- Date-wise OHLC (Open, High, Low, Close) for each ticker

Task 3.4: Label data with direction (up/down)

Rule:	`target = 1` if `Close(t+1) > Close(t)` else `0`

Task 3.5: Merge text features with price labels

Final structured dataset (Pandas DataFrame):

date	ticker	avg_sentiment	fear	joy	topic_3	...	target

Task 3.5: Merge Text Features with Price Labels

We created a structured and labeled dataset that includes:

- Preprocessed text (`cleaned_text`)
- Sentiment scores (VADER, TextBlob)
- Emotions (NRC top-3 tags)
- Named entities (ORGs, etc.)
- Topics (BERTopic topic ID + keywords)
- Temporal metadata (hour, weekday)
- Ticker symbol (via Yahoo API)
- Final label: 📈 `1` for stock price up, 📉 `0` for stock price down

We saved the result as:

Task 4: Model Training Pipeline

4.1 Load and Vectorize Text

- Loaded `cleaned_text`
- Applied **TF-IDF** with `max_features=5000` , bi-grams included

4.2 Build Baseline Model

- Trained **Logistic Regression** on `cleaned_text`
- Evaluated using `classification_report` and `confusion_matrix`

4.3 Enhance Feature Set

- Added numerical features:
    - `sent_vader_compound`, `sent_textblob_polarity`
    - All `emotion_*` columns
    - `topic_id`, `hour`
- Combined with TF-IDF text vectors using `scipy.hstack`

4.4 Train Final Model

- Trained Logistic Regression on combined feature set
- Saved:
    - `models/logistic_model_full.pkl`
    - `vectorizers/tfidf_vectorizer_full.pkl` **Task 4.5: Inference Script**

Created `predict_direction.py` to:

- Preprocess new text
- Extract features (sentiment, emotion, etc.)
- Vectorize using saved TF-IDF
- Predict 📈 or 📉 using trained model

Example output:

```
Prediction: UP 📈 with confidence 0.82
```

Final Summary: Aistox — Sentiment-Aware Stock Market Prediction Platform

What We Accomplished

Over the course of this project, we designed and implemented a complete pipeline for sentiment-aware stock market direction prediction using publicly available textual data. The project involved the following key phases:

Phase 1: Data Collection

We built scrapers and collectors to gather real-time data from:

- Financial news websites
- Reddit posts from relevant communities
- Twitter content (if enabled)

Each entry included content, source, timestamp, and metadata for further processing.

Phase 2: Textual Feature Engineering

We applied a rich set of Natural Language Processing techniques to convert raw text into structured features:

- Cleaned and lemmatized text
- Sentiment scores from VADER and TextBlob
- Emotion classification using the NRC Lexicon

- Named entity recognition (e.g., companies, tickers)
- Topic modeling using BERTopic
- Time-derived features (hour, weekday, etc.)

Phase 3: Price Labeling

To create supervised labels:

- We mapped entities to stock tickers via Yahoo Finance API
- Fetched historical stock prices for each ticker around the publication time
- Labeled each sample as `1` if the price rose the following day, otherwise `0`

This resulted in a structured dataset combining textual signals with market behavior.

Phase 4: Model Training

We trained a baseline logistic regression model using:

- TF-IDF features from cleaned text
- Engineered features such as sentiment, emotions, and topic ID

The model was evaluated using standard classification metrics and saved for future inference. We also built a real-time `predict_direction.py` script to classify new text samples.

What This Model Can Do

- Analyze textual data from news or social media
- Generate a direction prediction (up/down) based on sentiment and related features
- Serve as a component in broader market sentiment analysis systems

Realistic Limitations

While the pipeline successfully integrates NLP and market data to generate directional predictions, **this model is not yet suited for real-time financial decision-making**. The stock market is influenced by numerous unpredictable factors, including macroeconomics, insider activity, and geopolitical events.

Even the most advanced sentiment models cannot consistently and accurately predict price movements in all conditions. As such, this model should be treated as **an exploratory research tool** rather than a trading signal generator.

Future Directions

- Explore ensemble and neural network models (e.g., XGBoost, BERT)
- Integrate real-time data streaming for live dashboards
- Perform backtesting with simulated trades to estimate effectiveness
- Build a feedback loop for continual learning and domain adaptation
- Add confidence thresholds to filter low-certainty predictions

Final Deliverables

- Feature-engineered dataset: `final_labeled_data.csv`
- Trained model and vectorizer
- Inference script for future predictions
- Modular pipeline for experimentation and iteration

References

Books & Research Papers

- **"Advances in Financial Machine Learning"** – Marcos López de Prado
- **"Sentiment Analysis and Opinion Mining"** – Bing Liu
- **Financial News and Stock Returns: Evidence from the Web** – Tetlock, Saar-Tsechansky & Macskassy
- **Event Extraction for Financial Forecasting** – Google Research

---

NLP Tools and APIs

- **spaCy**: https://spacy.io/
- **VADER**: https://github.com/cjhutto/vaderSentiment
- **TextBlob**: https://textblob.readthedocs.io/en/dev/
- **NRCLex**: https://github.com/metalcorebear/NRCLex
- **BERTopic**: https://github.com/MaartenGr/BERTopic
- **Yahoo Finance API** (unofficial): https://query1.finance.yahoo.com

---

GitHub Repositories

- **StockSight** – Sentiment-based stock scanner: https://github.com/shirosaidev/stocksight
- **FinBERT** – Pretrained BERT on financial sentiment: https://github.com/ProsusAI/finBERT
- **BERTopic** – Topic modeling with transformers: https://github.com/MaartenGr/BERTopic
- **News Sentiment Stock Price Prediction** – https://github.com/TechNinja101/News-Sentiment-Analysis-and-Stock-Market-Prediction

---

Video Lectures and Tutorials

- **CS224n: Natural Language Processing with Deep Learning** – Stanford University (YouTube): https://youtube.com/playlist?list=PLoROMvodv4rOABXSygHTsbvUz4G_YQhOb
- **Sentiment Analysis in Finance** – DataCamp / Coursera courses
- **Topic Modeling with BERTopic** – https://www.youtube.com/watch?v=4Zq90kVv-Is
- **Machine Learning for Trading (Georgia Tech)** – https://www.udacity.com/course/machine-learning-for-trading--ud501

Inspiration and Use Cases

- Bloomberg Terminal: Sentiment overlays for financial headlines
- RavenPack / Accern: Financial news analytics using NLP
- Kaggle Competitions: Financial Sentiment Analysis, Two Sigma, etc.

Thank you!

Signing off

Induj Tyagi