Project Report: Sentiment-Driven Stock Market Prediction

Prepared By: Sateesh N

Project: Capstone Development

Subject: S&P 500 Directional Movement Analysis via XGBoost

GitHub repository link

1. Executive Summary

This report outlines the development and evaluation of a machine learning pipeline designed to predict whether the S&P 500 index (^GSPC) will move **Up** or **Down** on any given trading day. The core hypothesis was that combining **financial sentiment analysis** (derived from news headlines) with **quantitative technical indicators** (momentum and trend data) would yield a predictive edge over random market chance.

While the technical pipeline—spanning data ingestion, natural language processing (NLP), and gradient boosting classification—was successfully implemented, the model achieved a testing accuracy of **45.13%**. This performance is slightly below that of a random guess, indicating that the current prototype serves as a functional architectural "Proof of Concept" but requires significant refinement in data sourcing (specifically replacing synthetic news with real-time APIs) before it is viable for live trading.

2. Methodology & System Architecture

The project utilizes a multi-layered approach to feature engineering, merging qualitative text data with quantitative price data.

2.1 Data Acquisition

- **Market Data:** Historical OHLCV (Open, High, Low, Close, Volume) data for the S&P 500 was sourced via the yfinance library for the period 2020–2024.
- **News Data:** A dataset of financial headlines was generated and processed to simulate the flow of market-moving information.

2.2 Feature Engineering Pipeline

To transform raw data into predictive signals, two categories of features were engineered:

- **NLP Sentiment (The "Mood"):** Headlines were analyzed using the **NLTK VADER Lexicon**. A "Compound Score" was generated for each day, ranging from -1 (Negative) to +1 (Positive).

- **Technical Indicators (The "Math"):**
    - **Trend:** Simple Moving Averages (SMA) and Exponential Moving Averages (EMA) at 5, 10, and 20-day intervals to smooth out volatility.
    - **Momentum:** The **Relative Strength Index (RSI)** was calculated to detect overbought (>70) or oversold (<30) conditions.
    - **Convergence: MACD** (Moving Average Convergence Divergence) was implemented to identify trend reversals.

3. The Mechanics of Prediction (How the Model Works)

The model functions as a **Binary Classifier**, trained to answer a simple "Yes/No" question: *Will the stock price close higher tomorrow than it did today?*

To arrive at this prediction, the system does not simply "look" at a raw price chart. Instead, it processes data through a three-stage "inference engine" that mimics the decision-making process of a quantitative analyst.

Stage 1: Feature Extraction (The "Senses")

Before the model makes a decision, it converts raw data into interpretable signals:

- **Decoding Sentiment:** The model uses **VADER**, a rule-based sentiment analysis tool, to "read" news headlines. It assigns a mathematical score to the text, allowing the computer to quantify abstract concepts like market fear and greed.
- **Decoding Momentum:** Using the `pandas_ta` library, the model calculates technical indicators. These indicators provide context—telling the model if the market is "overheated" or "cheap" relative to recent history, regardless of the absolute dollar price.

Stage 2: Temporal Alignment (The "Shift")

This is the most critical logic step. If we feed the model *today's* price and ask it to predict *today's* movement, the prediction is useless for trading.

- **The Logic:** The code applies a **Target Shift (-1)**.
- **The Effect:** It explicitly aligns *Today's* inputs (e.g., Today's RSI is 75, Today's News is Positive) with *Tomorrow's* outcome (Did the price go up?). This forces the model to learn predictive patterns rather than just describing the present state.

Stage 3: Gradient Boosting (The "Brain")

The **XGBoost** algorithm utilizes an "ensemble" approach. It does not rely on a single equation but instead builds hundreds of Decision Trees.

1. It creates a simple decision tree (e.g., "If RSI > 70, predict Down").
2. It tests this tree and identifies where it made mistakes.

3. It builds a *second* tree specifically designed to correct the errors of the first tree.
4. It repeats this process iteratively, "boosting" the model's performance by combining hundreds of weak learners into one strong predictive model.

Finally, the model outputs a probability score. If the probability of an increase is >50%, it outputs a **1 (Buy/Up)**; otherwise, it outputs a **0 (Sell/Down)**.

4. Model Performance Evaluation

The model was evaluated on a held-out test set comprising 195 trading days (20% of the dataset). The results indicate a struggle to differentiate between market noise and genuine signals.

Metric	Score	Interpretation
Accuracy	**45.13%**	The model is incorrect more often than random chance.
Precision (Class 1 - Up)	0.48	When it predicts "Buy", it is correct only 48% of the time.
Recall (Class 1 - Up)	0.49	It fails to catch 51% of actual market rallies.
Precision (Class 0 - Down)	0.42	High false positive rate when predicting market drops.

Performance Summary:

The confusion matrix reveals that the model suffers from high False Positives—predicting an upward trend when the market actually falls. In a real-world trading scenario, this is a high-risk failure mode as it would trigger buy orders immediately preceding a loss.

5. Critical Failure Analysis

A crucial requirement of this project is to analyze **why** the model failed to achieve high accuracy. The low performance is not a result of code errors, but rather specific structural contradictions in the data strategy.

Failure 1: The "Groundhog Day" Contradiction

- **The Scenario:** The code generates dummy news using a repeating pattern of 4 headlines (e.g., *"Economy shows strong growth"*).
- **The Failure:** The model sees the exact same headline—and thus the exact same Sentiment Score—on days the market goes **UP** and on days it goes **DOWN**.
- **Impact:** This forces the algorithm to "learn" that news sentiment has **zero correlation** with price. The model effectively discards the NLP component because the input (X) remains constant while the output (y) flips randomly.

Failure 2: The Lagging Indicator Trap

- **The Scenario:** The model relies heavily on Moving Averages (SMA/EMA). These are mathematically based on *past* prices.
- **The Failure:** By the time a Moving Average "crosses over" to signal a buy, the move has often already occurred.
- **Specific Example:** During a V-shaped recovery, the SMA might still be trending down (calculating the average of the crash) while the price is rocketing up. The model predicts "Down" based on the SMA, missing the entire rally.

Failure 3: The RSI Overextension Error

- **The Scenario:** The model treats high RSI (>70) as a sell signal (Overbought).
- **The Failure:** In a strong Bull Market (like the 2020-2021 recovery present in the dataset), the RSI can stay "overbought" for weeks while prices continue to skyrocket.
- **Impact:** The model bets against the momentum, predicting a reversal that doesn't happen. This explains the low precision (0.42) for the "Down" class—the model kept trying to short a rising market.

6. Conclusion & Recommendations

The project successfully established a technical framework for financial machine learning, demonstrating proficiency in data manipulation (`pandas`), natural language processing (`nltk`), and advanced modeling (`xgboost`). However, the predictive utility is currently limited by the quality of input data.

Strategic Roadmap (Next Steps):

1. **Data Integrity (Immediate Fix):** Replace the dummy news generator with a live Financial News API (e.g., AlphaVantage or NewsAPI). This will ensure every trading day has unique, time-stamped headlines, allowing the model to learn actual sentiment correlations.
2. **Model Evolution:** Transition from the static XGBoost classifier to **LSTM (Long Short-Term Memory)** networks. LSTMs are specifically designed for time-series data

and can "remember" sequences of days, whereas XGBoost treats every day as an isolated event.

3. **Advanced Features:** Incorporate **Volatility Indicators** (like Bollinger Bands or ATR). This would help the model distinguish between a "quiet" market and a "volatile" crash, reducing false signals during chop.

Final Verdict: The system is a robust **Proof of Concept** requiring data-stream integration to achieve statistical significance.