Predicting Short-Term FAANG Stock Price Movements

Introduction

In this project, I build and compare several machine learning models to predict the **next-day price direction** of FAANG stocks. The response variable is binary:

Target = 1 if tomorrow's adjusted closing price is higher than today's, and 0 otherwise.

Short-term direction prediction is a challenging problem because daily stock returns are noisy and close to a random walk. Therefore, the goal of this project is **not** to achieve very high accuracy, but to design a model that captures useful patterns in momentum and volatility while maintaining reasonable recall and F1-score for "up" days. This kind of model can help an investor or analyst prioritize days with higher probability of upward movement.

Data & Feature Engineering

The dataset contains daily data (Open, High, Low, Close, Adjusted Close, Volume) for several large-cap technology stocks (FAANG). After converting the Date column to a proper datetime type and sorting by Company and Date, I engineer several groups of features:

- **Returns:** 1-day, 3-day, 5-day, and 10-day percentage returns (Return1, Return3, Return5, Return10), computed within each stock using groupby and pct_change.
- **Moving averages:** 7-, 14-, and 30-day moving averages of adjusted close (MA7, MA14, MA30).
- **Volatility:** rolling standard deviations of Return1 over 7, 14, and 30 days (Volatility7, Volatility14, Volatility30).
- **Intraday spreads:** normalized high–low and open–close spreads (High_Low_Spread, Open_Close_Spread).
- **Lagged returns:** 1-, 2-, and 3-day lags of daily return (Lag1, Lag2, Lag3).

The target variable Target is defined as 1 if the next day's adjusted close is higher than the current day's adjusted close within the same stock, and 0 otherwise. Rows with insufficient history for rolling features are dropped, leaving a clean dataset for modeling.

Modeling Approach

I split the data into **70% training and 30% test sets**, keeping the original time order (shuffle=False) to respect the time-series nature of the problem.

I compare three models:

1. **Logistic Regression** – a linear baseline that is fast, simple, and interpretable.
2. **Random Forest** – a tree-based ensemble that can capture non-linear interactions and is robust to noise.
3. **Gradient Boosting** – a boosting ensemble that iteratively corrects errors and often achieves strong performance on structured tabular data.

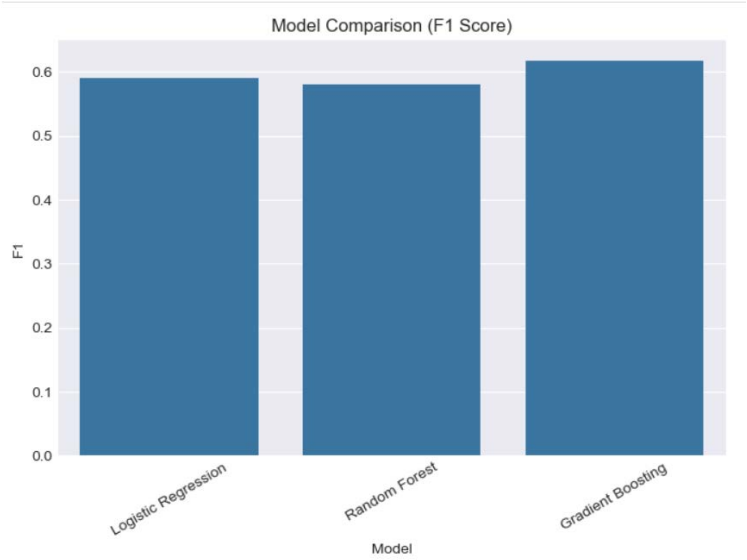
Price-level and spread features are standardized using StandardScaler for the logistic regression model, while tree-based models operate on the original (unscaled) features. All models are evaluated on the test set using four metrics: **accuracy, precision, recall, and F1-score**.

Results

The table below summarizes the test-set performance of the three models:

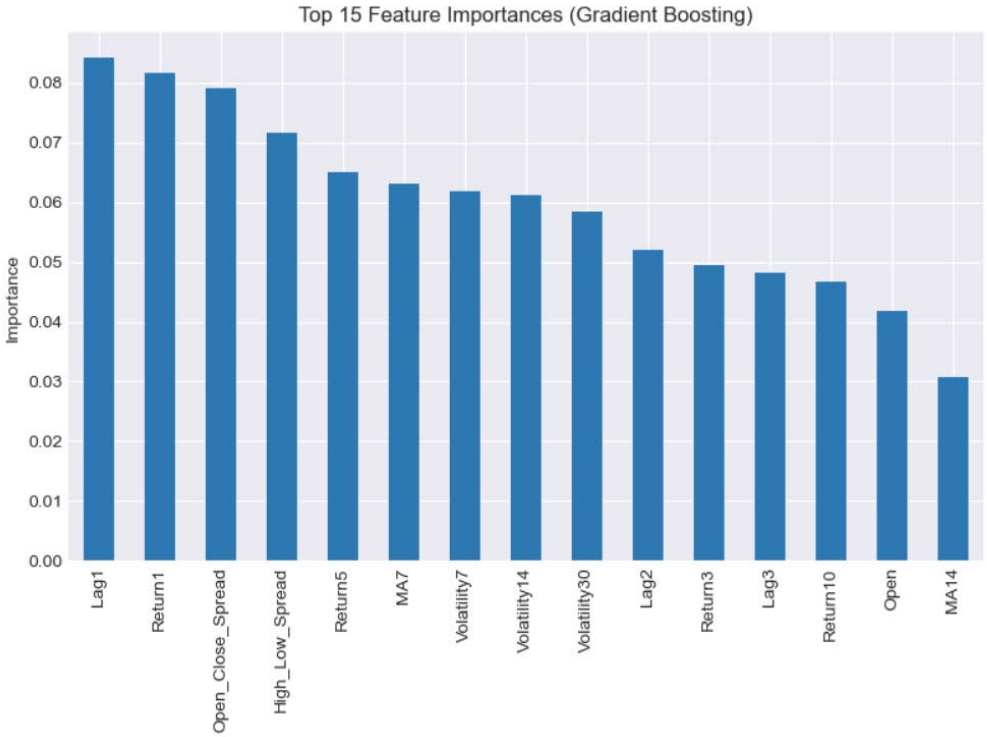
- Logistic Regression – Accuracy ≈ 0.51, F1 ≈ 0.59

- Random Forest – Accuracy ≈ 0.52, F1 ≈ 0.58
- Gradient Boosting – Accuracy ≈ 0.52, F1 ≈ 0.62



While accuracy improves only slightly across models (around 0.51–0.52), **Gradient Boosting clearly achieves the highest recall and F1-score**. This means it captures a larger share of true "up" days while keeping false positives at a reasonable level.

The feature importance plot shows that Lagged returns(Lag 1), High_Low_Spread, Open_Close_Spread,short-horizon returns (Return1, Return5) and moving averages (MA7) are among the most influential predictors. This is consistent with the intuition that short-term momentum and volatility regimes play a major role in daily direction.



Recommendations

1. **Use the Gradient Boosting model as a ranking tool for potential long entries.**
   Rather than treating the predictions as guaranteed signals, the model can be used to rank days by the probability of an upward move and focus attention on the top-scored opportunities.
2. **Combine the model with risk management and transaction-cost constraints.**
   Because daily moves are small and noisy, the model should be implemented together with strict stop-loss rules and minimum expected return thresholds to ensure that predicted edges are not wiped out by trading costs.
3. **Retrain the model regularly with a rolling window.**
   Market regimes change over time. Re-training the Gradient Boosting model on recent data (e.g., last 1–2 years) can help keep the feature relationships up to date and maintain performance.