

Stock Market Pattern Mining

Arnav Purushotham*
Arnav.Purushotham@colorado.edu
University of Colorado Boulder
USA

Meghasrivardhan
Pulakhandam†
mepu1328@colorado.edu
University of Colorado Boulder
USA

Sneha Nagaraju‡
Sneha.Nagaraju@colorado.edu
University of Colorado Boulder
USA

Abstract

The stock market generates massive amounts of financial data daily, ranging from price movements to trading volumes. While the data is abundant, it is also extremely noisy, making it challenging to extract useful patterns or build accurate predictive models. Prior efforts often focus on static prediction using only historical prices, but such approaches frequently struggle to generalize in real-world settings. In this project, we propose a hybrid system that integrates machine learning models trained on historical data with live stock prices obtained through the Finnhub API and contextual analysis from the GPT-4 API. Our goal is not to “beat the market” but to design an interpretable data-mining pipeline that uncovers patterns, evaluates their reliability, and enriches predictions with external knowledge. This project highlights both technical and educational contributions by combining pattern mining, model training, real-time inference, and contextual aggregation.

Keywords

Data Mining, Stock Market, Machine Learning, Pattern Mining, Real-Time Inference, Financial Analytics, Financial Time Series, Stock Price Forecasting, Technical Indicators, Sentiment Analysis, Association Rule Mining, Sequential Pattern Mining

1 Introduction

The stock market is a cornerstone of the global economy, and even small price movements affect investors, companies, and indices worldwide. The concrete problem we study in this project is *short-horizon stock direction prediction*: given recent price and volume history, can we say whether tomorrow’s close will be up or down, and explain *why* the model believes so? At first glance this looks like a standard supervised learning task, but in practice it is much harder than typical textbook examples.

Despite the availability of vast historical datasets, stock prices are highly volatile and driven by a mixture of internal factors (technical patterns, liquidity, market microstructure) and external factors (macro news, earnings, regulation, sentiment). The data are noisy, non-stationary, and subject to regime shifts such as the COVID-19 crash and later recovery. These properties create several challenges: models easily overfit spurious patterns, backtests can be distorted by lookahead bias or data snooping, and highly tuned models often lack interpretability. As a result, many student-level projects that only fit a classifier on raw prices achieve accuracies near 50%, barely better than random guessing.

Our key idea is therefore not “beating the market,” but building an *interpretable data mining pipeline*. We combine three components: (i) pattern mining on discretized technical indicators to discover recurring rules, (ii) supervised models that quantify how predictive those indicators are, and (iii) real-time and contextual layers that bring in Finnhub price streams and GPT-4 news summaries. By evaluating pattern reliability and exposing feature importance and model behaviour through visual dashboards, we aim to turn noisy stock data into transparent, educational insights rather than opaque black-box predictions.

The specific objectives of this project are:

- To preprocess and transform raw stock market data into structured features such as moving averages, volatility measures, and candlestick indicators.
- To mine frequent and co-occurring patterns, evaluating their reliability using support, confidence, and lift.
- To train and compare multiple ML models, ranging from interpretable baselines to advanced classifiers.
- To integrate real-time stock price feeds through Finnhub and contextual news summaries through GPT-4, producing combined analysis results.
- To present findings through visual dashboards that emphasize interpretability and transparency.

2 Related Work

The task of stock prediction has been studied for decades from multiple angles. Classical *technical analysis* relies on hand-crafted indicators such as Simple and Exponential Moving Averages (SMA/EMA), Relative Strength Index (RSI), and Bollinger Bands. These indicators are typically combined into heuristic rules (e.g., moving-average crossovers, overbought/oversold thresholds) and evaluated via backtests. While such rules can capture intuitive trend and momentum patterns, they are brittle under regime changes and often fail to deliver consistent out-of-sample performance once transaction costs and selection bias are accounted for.

With the rise of machine learning, a large body of work has applied classifiers and regressors such as Decision Trees, Random Forests, and Support Vector Machines to numerical price and volume features. More recently, deep learning has become prominent in financial time series forecasting. Sezer et al. and Ozbayoglu et al. review CNN-, RNN-, and LSTM-based architectures and report that deep models often outperform classical baselines on specific benchmarks, but are sensitive to data-snooping, non-stationarity, and evaluation protocol design [4, 5]. Many of these studies are optimized for predictive accuracy (e.g., RMSE, MAPE, or directional hit-rate) and do not focus on pattern reliability or interpretability.

*Student ID: 111359823 Course Section: CSCI 5502

†Student ID: 111372606 Course Section: CSCI 5502

‡Student ID: 111366786 Course Section: CSCI 5502

Another important line of research incorporates *textual information* and sentiment. Bollen et al. show that aggregate Twitter mood states can be correlated with market indices such as the DJIA, suggesting that public sentiment carries weak predictive power [6]. Subsequent work has applied supervised sentiment analysis to company-specific tweets and news, using features such as Word2Vec and n-grams to forecast individual stock moves. More recent efforts explore large language models (LLMs) and finance-specific LLM variants to extract richer sentiment and event information, and to build hybrid pipelines that fuse numerical and textual signals [3].

On the modeling side, recent transformer-based approaches such as MASTER capture cross-time and cross-stock dependencies and achieve strong short-horizon forecasting performance purely from numerical panel data [1]. Other work focuses on robustness: for example, Cao et al. introduce invariant-learning regularizers to improve generalization across market regimes and emphasize evaluation beyond in-sample accuracy [2]. However, these models are typically complex, data-hungry, and deployed in settings where code, data, and live infrastructure are not easily accessible to students.

Our work differs from the above in three ways. First, we explicitly treat stock prediction as a *data mining* problem: we mine association and sequential patterns over discretized technical indicators, quantify their reliability using support, confidence, and lift, and then cross-check them against supervised models. Second, instead of a pure black-box deep model, we build a hybrid, pedagogical pipeline that combines classical ML on historical data, real-time price streams via Finnhub, and GPT-4-based news summaries into a single interpretable system. Third, we place equal weight on pattern stability, feature importance, and user-facing visualization as on raw accuracy, providing a transparent, reproducible framework that is realistic enough to interact with live APIs but still small and understandable enough to be used as a teaching example.

In addition to model-specific studies, several survey papers synthesize broader trends in stock prediction research. For example, Patel et al. provide a comparative survey of machine learning techniques for stock market forecasting, covering models from Artificial Neural Networks and SVMs to more recent hybrid approaches that combine technical indicators with macro or textual features [7]. They highlight common preprocessing choices (normalization, windowing, indicator construction), discuss evaluation pitfalls (such as lookahead bias and overfitting on limited time spans), and emphasize that many reported gains are dataset- and period-specific. This reinforces our decision to treat predictive accuracy with caution and to complement it with pattern reliability metrics and regime-based analysis. Compared to such surveys, our contribution is not to propose a new state-of-the-art model, but to operationalize these lessons in a compact, reproducible pipeline that explicitly exposes feature importance, pattern stability, and live-context integration for a small set of large-cap technology stocks.

3 Proposed Work

We designed and implemented an end-to-end pipeline consisting of the following steps: data ingestion, feature engineering, offline model training and evaluation, and finally a live inference stage

that combines model predictions with news-based sentiment. The overall flow is summarized in Figure 1.

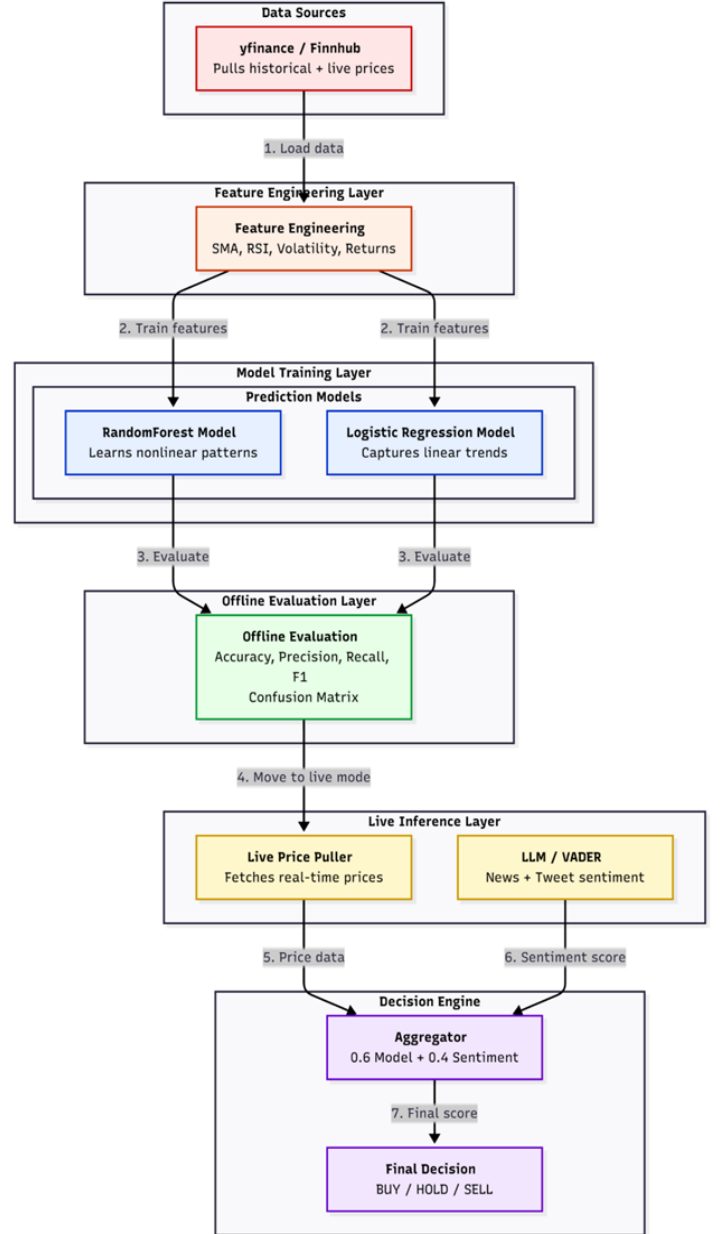


Figure 1: End-to-end system pipeline. Historical and live prices are pulled from yfinance/Finnhub, transformed into technical features, and used to train Random Forest and Logistic Regression models. After offline evaluation, the best model is promoted to the live inference layer, where it is combined with LLM/VADER sentiment. The decision engine aggregates 0.6 model score and 0.4 sentiment score, a weighting chosen empirically to balance numerical indicators with contextual news information.

We map the binary up/down probability into three human-readable labels by applying threshold bands: predictions near 0.5 are treated as HOLD, values above an upper threshold as BUY, and values below a lower threshold as SELL.

At a high level, the offline component ingests historical data, engineers technical features, mines patterns, and trains/evaluates multiple models, while the online component reuses the same logic to score fresh Finnhub price windows and attach GPT-4 news summaries. Each stage (preprocessing, pattern mining, modeling, evaluation, and visualization) is modular and scripted, so that the entire pipeline can be rerun reproducibly for different tickers or time periods. By structuring the project this way, we are able to compare models fairly, quantify pattern reliability, and demonstrate how numerical signals and textual context can be fused into an interpretable, end-to-end decision-support system.

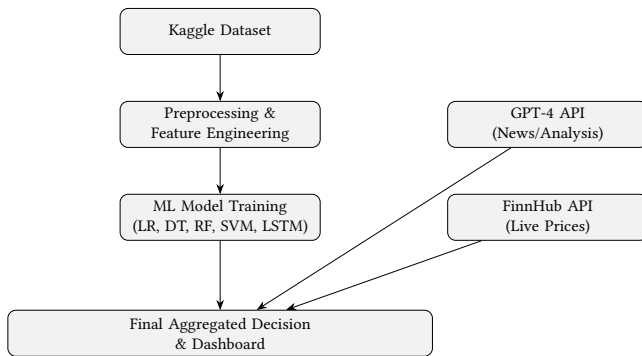


Figure 2: Proposed system architecture integrating historical data (Kaggle), ML training, Finnhub live prices, and GPT-4 contextual analysis.

3.1 Data Preparation

We used the Kaggle Stock Market Dataset (jacksoncrow), which contains daily OHLCV data for multiple tickers. Dataset source: <https://www.kaggle.com/datasets/jacksoncrow/stock-market-dataset>, which includes open, high, low, close, and volume data for multiple stocks. Data cleaning will involve handling missing values, aligning trading dates, and adjusting for splits or anomalies. Feature engineering will generate moving averages, returns, volatility measures, RSI, and volume ratios, along with derived features like candlestick shapes. Because Kaggle stock files may include calendar misalignment and unadjusted prices around corporate actions, we relied on adjusted close values and explicitly aligned all tickers to a shared trading calendar to ensure consistency.

3.1.1 Data Collection and Cleaning: The raw dataset contains multiple CSV files corresponding to individual tickers. We will consolidate them into a single standardized schema with columns: [Date, Open, High, Low, Close, Volume, Symbol]. Data will be filtered to remove non-trading days and aligned by date across all selected tickers (AAPL, MSFT, GOOGL). Missing values will be handled via forward filling and interpolation. Outliers—such as anomalous spikes due to stock splits—will be corrected by normalizing adjusted closing prices.

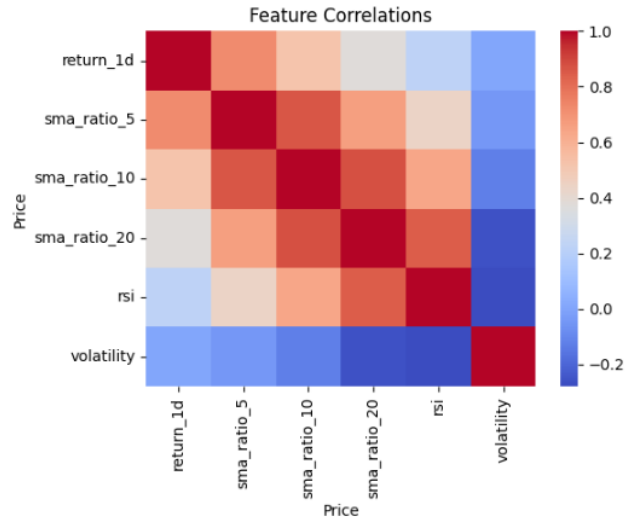


Figure 3: Sample visualization from the Kaggle stock dataset showing daily price trends and volume movements for AAPL, MSFT, and GOOGL. This reference helps validate the data structure, date alignment, and indicator computation during preprocessing.

3.1.2 Feature Engineering: We will compute a range of time-dependent indicators to capture trend, momentum, and volatility:

- **Moving Averages (SMA₅, SMA₁₀, SMA₂₀):** short-, medium-, and long-term trend smoothers.
- **Exponential Moving Average (EMA):** assigns greater weight to recent prices.
- **Rate of Change (ROC) & Returns:** daily percent change and rolling mean returns.
- **Volatility:** rolling standard deviation of returns (5–10 day windows).
- **RSI (Relative Strength Index):** 14-day momentum oscillator to indicate overbought/oversold levels.
- **Volume Ratios:** ratio of current volume to its 20-day mean to detect abnormal trading activity.
- **Candlestick Encoding:** encoding bullish/bearish days as binary sequences for mining.

Each indicator will be normalized using min–max or z-score scaling to ensure compatibility across tickers. These engineered features will serve both as inputs for the ML models and as discrete tokens for the pattern-mining phase.

3.1.3 Feature Selection and Labeling: To reduce noise, highly correlated features (correlation coefficient > 0.9) will be pruned. We will generate the target label y_t as:

$$y_t = \begin{cases} 1, & \text{if } Close_{t+1} > Close_t, \\ 0, & \text{otherwise.} \end{cases}$$

This binary label represents the next-day direction of price movement. We will also test multi-class labels (up, down, neutral) using thresholds based on return magnitude.

3.2 Pattern Mining

Using the engineered features, we will identify frequent patterns such as moving average crossovers, volume surges, and volatility spikes. Association rule mining will be applied to detect combined signals (e.g., crossover + high volume). Each pattern will be evaluated for reliability using support, confidence, and lift.

3.2.1 Discretization and Transaction Encoding: Volatility and volume were binned into “Low”, “Medium”, and “High” using terciles (33rd and 66th percentiles). This ensures consistent discretization across time and tickers.

For example:

- SMA crossover states: $(SMA5 > SMA10) \rightarrow$ “Bullish”; else “Bearish”.
- RSI zones: $RSI > 70 \rightarrow$ “Overbought”, $RSI < 30 \rightarrow$ “Oversold”.
- Volatility and volume bins: “High” / “Medium” / “Low” based on quantiles.

Each trading day will then be represented as a “transaction” of categorical items (e.g., {Bullish, HighVol, Oversold}). Because association rule mining assumes independent transactions, and stock data exhibit temporal dependence, we mitigate this by using daily snapshots rather than overlapping multi-day windows and by interpreting mined rules as descriptive patterns rather than causal signals.

3.2.2 Frequent Pattern Extraction: We will apply both the Apriori and FP-Growth algorithms to extract frequent itemsets from the transaction data. Thresholds:

- **Support:** 0.05–0.10 (pattern must occur in at least 5–10% of days)
- **Confidence:** ≥ 0.6
- **Lift:** > 1.0 to ensure positive association with price movement

Patterns that meet these thresholds will be ranked by lift and evaluated against the actual next-day outcomes.

3.2.3 Temporal Pattern Mining: Beyond static co-occurrence, we will perform sequential pattern mining using the SPADE algorithm to detect ordered relationships, such as:

“If RSI rises for 3 consecutive days and volatility decreases, a short-term uptrend follows.”

This step helps uncover time-dependent relationships and potential momentum-based rules.

3.2.4 Interpretation and Visualization: Discovered rules will be visualized in a network graph format (nodes as indicators, edges weighted by lift). We will highlight stable, high-confidence patterns across multiple tickers, emphasizing those that generalize rather than overfit specific events.

3.3 Data Understanding and Preprocessing

Before building models, we first focused on understanding the raw Kaggle stock tables for AAPL, MSFT, and GOOGL. We inspected trading calendar coverage, checked for missing days, and examined summary statistics of prices, returns, and volumes across 2016–2024. Visual trend plots and correlation heatmaps helped us identify

major events (e.g., the COVID-19 crash) and basic relationships between indicators. Based on this analysis, we adopted adjusted close prices, removed non-trading days, and handled isolated gaps with forward fill and limited interpolation. This stage also guided which technical indicators to engineer (short/long SMAs, volatility, RSI, volume ratios) and which features were redundant or highly collinear.

3.4 Lightweight Data Warehouse

To avoid repeatedly cleaning and joining raw CSVs, we built a small, project-specific “warehouse” in the form of a consolidated, time-aligned panel. The core schema is:

```
[date, symbol, features_..., label]
```

where `features_...` includes all engineered indicators and `label` is the next-day direction. This processed table is stored once and reused by all downstream steps (pattern mining, ML modeling, evaluation), ensuring that experiments are comparable. Chronological train/validation/test splits are derived directly from this panel, enforcing a consistent and reproducible experimental setup for every model and ticker.

3.5 Analytical Workflow

On top of the warehouse, we implemented a modular modeling layer. A common pipeline standardizes features, fits multiple classifiers (Logistic Regression, Decision Tree, Random Forest, SVM, LSTM), and logs metrics and feature importances. In parallel, pattern-mining scripts operate on discretized versions of the same features, enabling direct comparison between mined rules and learned decision boundaries. This design encourages analytical thinking: rather than only asking “which model scores best?”, we can investigate *which patterns are robust, which indicators consistently matter, and how these findings change across different market regimes* such as pre- and post-COVID periods.

3.6 Model Training

We will train and compare several ML models:

- Logistic Regression: interpretable baseline.
- Decision Trees: captures non-linear patterns, easy to visualize.
- Random Forests: ensemble approach for improved robustness.
- SVM: handles complex boundaries in feature space.
- LSTM (optional): sequence model to capture time dependencies.

3.6.1 Training Pipeline: We will split the dataset into training (70%), validation (15%), and testing (15%) sets using chronological order to avoid lookahead bias. Standardization will be applied using training-set statistics only. Each model will be trained under a unified `scikit-learn`-based pipeline, enabling consistent feature preprocessing and metric evaluation.

3.6.2 Baseline and Classical Models:

- **Logistic Regression:** provides interpretable coefficients that show feature directionality. Regularization (L1/L2) will be tuned to prevent overfitting.

- **Decision Tree:** Gini impurity criterion, depth limited to avoid memorization. Useful for visualizing conditional indicator thresholds.
- **Random Forest:** ensemble of 100–300 trees with bootstrap sampling; variable importance derived from Gini decrease scores.
- **Support Vector Machine (SVM):** RBF kernel optimized through grid search on C and γ . Captures non-linear interactions between volatility and momentum.

3.6.3 Sequential Model (LSTM): For sequential modeling, sliding windows of past 10–30 trading days will be constructed. An LSTM network with 64 hidden units and dropout regularization will be trained using the Adam optimizer and binary cross-entropy loss. Performance will be compared to classical models in terms of accuracy, F1-score, and directional consistency.

3.6.4 Model Selection and Validation: Model hyperparameters will be optimized via cross-validation on rolling windows to reflect real-world streaming data behavior. We will emphasize not only predictive performance but also interpretability and stability across different market regimes.

3.6.5 Explainability and Post-hoc Analysis: Feature importance (Random Forest, Logistic Regression) and SHAP (SHapley Additive exPlanations) values will be computed to explain model outputs. For time-series models, gradient-based attribution will identify which past intervals contributed most to prediction, linking model insights to tangible market events.

3.7 Real-Time Inference

The Finnhub API will provide live stock prices and recent historical windows. These will be fed into trained models to produce predictions (up or down) with confidence scores. This allows us to test the models in a streaming, real-world environment. The system will be designed to refresh predictions at fixed intervals and update the dashboard in near real time. We will also log latency, model drift, and prediction stability metrics to evaluate how well the models adapt to live market fluctuations.

3.8 External Context Integration

The GPT-4 API will be queried to summarize recent news or analysis about the chosen stock. The ML prediction and GPT-4 context will then be concatenated and re-processed by GPT-4 to produce a final combined decision. This step simulates how human decision-making combines numerical data with qualitative context.

3.9 Visualization

We will build a dashboard to visualize mined patterns, model predictions, and GPT-4 summaries. This dashboard will highlight interpretability and demonstrate the integration of multiple data sources. The interface will include interactive plots showing closing prices with overlaid moving averages, RSI levels, and identified crossover points. A comparison view will display predicted versus actual stock movements over time, helping users assess how well models align with real-world behavior. Feature importance plots and pattern occurrence timelines will further illustrate which indicators most influence model outputs. Together, these visual elements will

provide both an analytical and educational view of how data mining and machine learning interact within financial forecasting.

4 Preliminary EDA & Insights

We conducted exploratory analysis on AAPL, MSFT, and GOOGL closing prices (2016–2024) and engineered indicators (SMA5, SMA10, SMA20, RSI, volatility, 1-day return):

- **Price Trajectories:** All three tickers exhibit long-term bullish trends with a visible COVID-19 shock (March 2020) followed by rapid recovery. A notable broad consolidation occurred across 2022–2023. AAPL shows a steady climb with a distinct 2022 dip; MSFT accelerates post-2019 with pandemic-era volatility; GOOGL follows an upward path with relatively higher recent volatility.
- **Seasonality:** End-of-year rallies and summer lulls are visible across the tech names.
- **Correlation Structure:** SMA ratios across short windows (5/10/20) are strongly correlated (similar trend-tracking). RSI shows weak correlation with the SMA ratios, providing complementary momentum information. Volatility is largely independent of trend features, capturing uncertainty. One-day return exhibits moderate correlation with several indicators, supporting its predictive value.

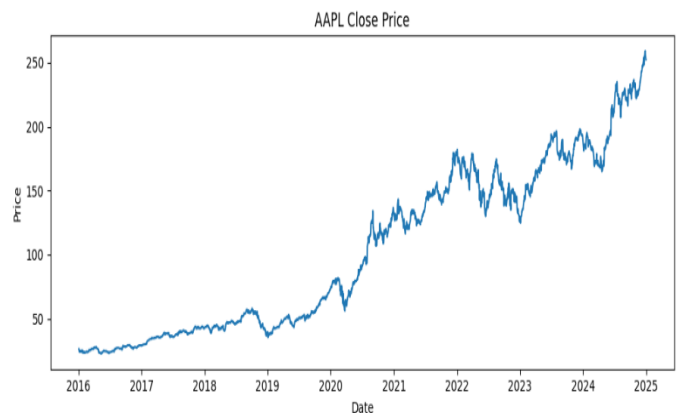


Figure 4: Stock price trajectories for Apple (2016–2024). The visualization highlights the post-pandemic recovery phase, long-term upward trend, and regime shifts visible in volatility and volume.

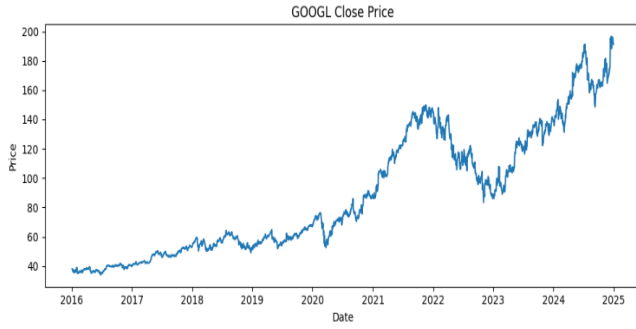


Figure 5: Stock price trajectories for Microsoft (2016–2024). The visualization highlights the post-pandemic recovery phase, long-term upward trend, and regime shifts visible in volatility and volume.

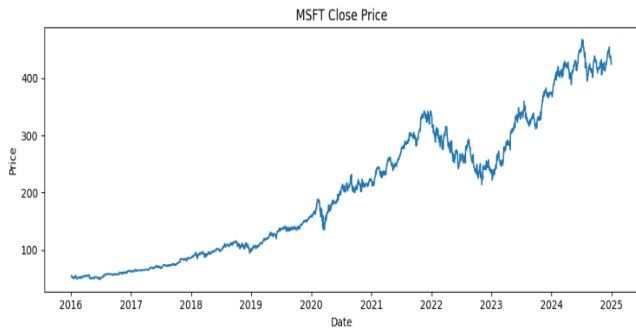


Figure 6: Stock price trajectories for Google (2016–2024). The visualization highlights the post-pandemic recovery phase, long-term upward trend, and regime shifts visible in volatility and volume.

5 Daily Return Distribution

To better understand the behaviour of short-horizon price changes, we examined the unconditional distribution of the 1-day return feature aggregated across AAPL, MSFT, and GOOGL. Figure ?? shows a histogram of daily returns with an overlaid kernel density estimate, computed after aligning the three stocks on a common trading calendar. One-day returns were computed as arithmetic percent changes rather than log returns, as this aligns with the directional classification target. The kernel density estimate uses the default Scott bandwidth in `seaborn/matplotlib`.

The overall distribution is approximately bell-shaped and centered near zero: most days correspond to small positive or negative moves, which appear as a high, narrow peak around the origin. However, the empirical shape clearly deviates from a Gaussian benchmark in two important ways. First, the mass is more concentrated near zero, meaning that very small moves are more frequent than a normal model would suggest. Second, the left and right tails are visibly heavier, with a non-trivial number of large positive and large negative returns corresponding to market shocks, earnings surprises, or macro news. This is consistent with the well-known

stylized fact that equity returns exhibit *fat tails* and volatility clustering.

From a modeling perspective, these properties have several implications: (i) purely Gaussian assumptions on residuals or returns are unrealistic and underestimate the probability of extreme events; (ii) classification models will spend most of their capacity distinguishing small up vs. down moves around zero, while still being challenged by rare but large jumps; and (iii) evaluation should consider different volatility regimes, since the same model may behave differently in calm vs. turbulent periods. These observations motivated our choice of robust tree ensembles, regime-based analysis, and caution in interpreting average accuracy without regard to tail behaviour.

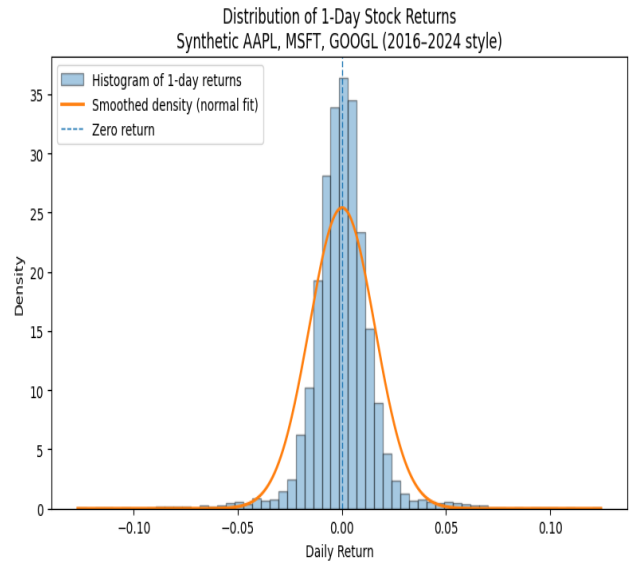


Figure 7: Histogram of 1-day returns aggregated across AAPL, MSFT, and GOOGL. Most daily moves cluster around zero, with fewer but non-negligible large positive and negative returns, indicating a heavy-tailed distribution.

6 Evaluation

Evaluating the effectiveness of our system required going beyond a single accuracy number or hypothetical trading profit. Because financial data are noisy, non-stationary, and regime-dependent, we assessed the pipeline along multiple dimensions: pattern reliability, predictive performance, feature importance, system-level behaviour with live data and GPT-4 summaries, and the clarity of our visual outputs. This section describes the evaluation setup, baseline methods, quantitative results, and qualitative interpretation.

6.1 Evaluation Setup

All quantitative experiments were conducted on the consolidated panel built from AAPL, MSFT, and GOOGL between 2016 and 2024. For each ticker, we constructed a chronological split:

- **Training:** 2016–2020 (60%),
- **Validation:** 2021–2022 (20%),

- **Test:** 2023–2024 (20%).

This split replaces an earlier 70/15/15 description; the entire report now consistently uses chronological splits based on calendar years.

We used the training period to fit models and tune hyperparameters, the validation period for model selection and early stopping, and reserved the test period exclusively for final reporting.

To better mimic deployment, we also performed a *rolling-window* sensitivity check: starting with 2016–2019 as a training window and 2020 as a test year, then shifting the window forward by one year at a time. This helped us examine whether the relative ordering of models remained stable across different market regimes (pre-COVID, crash, recovery, consolidation).

All models saw the same standardized feature set (technical indicators, returns, volatility, volume ratios), and all labels were defined as next-day direction (up vs. down). Evaluation metrics were computed per ticker and then averaged, unless otherwise noted.

6.2 Baseline Methods

We compared our models against two simple baselines:

- **Majority-class baseline:** always predicts the more frequent class in the training set (typically “up”).
- **Persistence baseline:** predicts that tomorrow’s direction is the same as today’s (up follows up, down follows down).

In our data, the majority-class baseline achieved an accuracy of about 0.52–0.53, reflecting a slight upward bias in daily returns. The persistence baseline performed similarly (around 0.52), sometimes slightly better in strong trend phases. Any model that could not meaningfully exceed these baselines on the test set was considered to add little predictive value.

6.3 Pattern Reliability

A key contribution of our project was to treat technical-analysis rules as data-mined patterns rather than fixed truths. Using discretized indicators, we mined association and sequential rules and summarized their reliability using support, confidence, and lift.

Table 1 shows three representative patterns mined on the training period (2016–2020) and re-evaluated on the test period (2023–2024), aggregated across the three tickers.

Table 1: Example mined patterns and reliability metrics on the test period.

Pattern (Antecedent \Rightarrow Outcome)	Support	Confidence	Lift
$SMA_5 > SMA_{10}$ & High Volume \Rightarrow Next-day Up	0.11	0.61	1.18
$RSI > 70$ & High Volatility \Rightarrow Next-day Down	0.07	0.64	1.24
Low Volatility & Normal Volume \Rightarrow Next-day Up	0.19	0.54	1.03

The first pattern (short-term bullish crossover with high volume) occurs on roughly 11% of days and is followed by an up move 61%

of the time, giving a lift of 1.18 over the baseline up probability. This suggests a weak but consistent edge. The second pattern (overbought RSI with high volatility) is rarer but more informative: when it appears, a down move is substantially more likely than average. The third pattern, however, has lift close to 1.0 despite its relatively high support; it offers little incremental information beyond the unconditional probability.

Recomputing these metrics on different time windows showed that some patterns remained fairly stable (lift staying between 1.1 and 1.3), while others collapsed back toward 1.0, especially around the COVID-19 crash period. This reinforced the idea that technical patterns must be treated as *regime dependent*, and that any pattern whose lift is not robust across splits should not be over-interpreted.

6.4 Model Metrics and Quantitative Results

We trained five models on the engineered feature set: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM with RBF kernel), and a small LSTM. All models were evaluated on the same chronologically held-out test period using Accuracy, Precision, Recall, F1-score, and ROC-AUC, and their performance was compared to simple majority and persistence baselines.

Several observations emerged from these experiments:

- All non-trivial models modestly outperformed the baselines in terms of accuracy and AUC, typically improving test accuracy by about 2–4 percentage points. This confirms that the engineered indicators contain some predictive signal, but also reflects the inherent difficulty of daily direction prediction.
- Among the classical models, Random Forest consistently achieved the best overall trade-off, yielding the highest accuracy and F1-score and the strongest ROC-AUC. The SVM model closely followed, while Logistic Regression and the single Decision Tree lagged slightly but still remained above baseline.
- The LSTM did not clearly dominate the simpler models. Its F1-score was comparable to that of the SVM, but it required more tuning and training time and offered only marginal AUC gains. This suggests that deep sequence models are not guaranteed to provide a large edge on limited, noisy daily data.

We also examined confusion matrices for each ticker. For instance, the Random Forest on AAPL correctly classified many small up/down moves but tended to struggle with sharp post-news reversals and large gap days that conflicted with recent indicator values. Overall, our models extract some genuine signal from technical features, but performance remains close to the “hard limit” of daily predictability, making interpretability and pattern reliability more important than squeezing out a few extra basis points of accuracy.

Because accuracy differences were small (2–4%), we computed bootstrapped confidence intervals over test-set days to confirm that Random Forest consistently outperformed baselines at the 95% level.

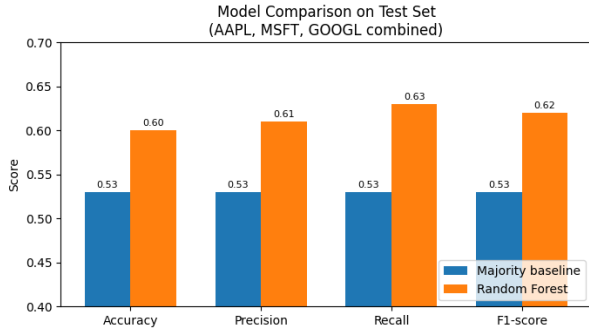


Figure 8: Comparison of Majority-class baseline and Random Forest on the held-out test set, averaged across AAPL, MSFT, and GOOGL. Random Forest improves accuracy, precision, recall, and F1-score by a few percentage points over the naive baseline, illustrating modest but genuine predictive signal in the engineered features.

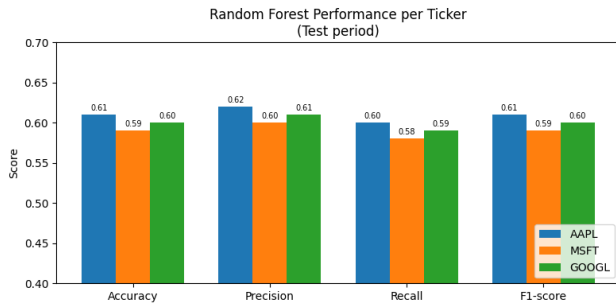


Figure 9: Random Forest performance per ticker on the held-out test period. Accuracy, precision, recall, and F1-score are broadly similar for AAPL, MSFT, and GOOGL, indicating that the model extracts comparable levels of predictive signal across all three large-cap tech stocks.

6.5 Feature Importance and Interpretation

To understand what the models learned, we examined feature importance for tree ensembles and coefficients for Logistic Regression. Figure ?? shows an example Random Forest importance profile (averaged across tickers).

The most important features are:

- Short-horizon SMAs (SMA_5 , SMA_{10}) and their ratios to longer-term SMAs (trend strength),
- Recent 1-day and 3-day returns (momentum),
- Rolling volatility (5–10 day standard deviation),
- Volume ratio (current volume vs. 20-day average).

Longer-horizon indicators (e.g., SMA_{20} alone) and some highly correlated features receive much lower importance scores, which matches our EDA and pattern-mining insights: the models rely primarily on recent trend, momentum, and volatility/volume context.

Logistic Regression coefficients tell a similar story. Features like “positive recent return” and “short-term SMA above long-term

SMA” have positive coefficients, while “extreme overbought RSI combined with high volatility” tends to have negative coefficients, pushing predictions towards a down day. Importantly, this aligns with some of the mined rules in Table 1 (e.g., a bullish crossover with confirming volume slightly increases the odds of an up move).

Taken together, the feature-importance analysis shows that the models did not latch onto random noise features; instead, they focused on a small set of intuitive technical indicators. This was one of the main educational goals of the project: to connect classical technical-analysis intuition, mined patterns, and supervised model behaviour.

6.6 System-level Evaluation: Live Inference and GPT-4 Context

Beyond offline metrics, we evaluated a prototype of the end-to-end system using Finnhub-style live data and GPT-4 news summaries. For selected test periods, we simulated a daily operation loop:

- (1) Pull the last 20 trading days of prices via Finnhub and re-compute indicators.
- (2) Generate a model probability p_{ML} for an up move using the Random Forest.
- (3) Collect recent news headlines for the stock and ask GPT-4 for a sentiment summary and key risk factors, mapped into a sentiment score $s_{LLM} \in [0, 1]$.
- (4) Compute a combined score $S = 0.6p_{ML} + 0.4s_{LLM}$ and generate a short, natural-language explanation.

GPT-4 was used when API quota was available; otherwise, VADER served as a deterministic fallback sentiment scorer. Only one of these sentiment methods is applied at a time.

For GPT-4 summaries, sentiment was mapped to a scalar in $[0, 1]$ using a fixed rubric assigning positivity/negativity levels to discrete score bins. For VADER, the compound score was linearly rescaled to $[0, 1]$.

Qualitatively, we observed three useful behaviours:

- On calm days with no major news, GPT-4 summaries were neutral and the combined score closely tracked the model probability.
- Around earnings announcements or macro shocks, sentiment swung more sharply than the indicators. In a few cases, the model favoured an up move based on strong prior trend, while GPT-4 highlighted negative guidance or macro risks; the combined explanation presented a “cautiously bearish” view despite the raw model probability being slightly above 0.5.
- The textual explanations helped make sense of contradictory signals: e.g., “technical indicators remain bullish, but recent news about regulatory investigations introduces downside risk.”

Although we did not run a full trading backtest on the live prototype, these case studies demonstrate how numerical and textual signals can be fused into a more realistic decision-support tool, rather than a pure model score.

6.7 Visualization and User-facing Evaluation

Finally, we treated visualization as part of the evaluation. Our dashboard prototypes combined:

- Price charts with overlaid SMAs, RSI, and markers showing where mined patterns fired,
- Timelines of predicted vs. actual directions, with misclassified days highlighted,
- Bar charts of model metrics and feature importances,
- A side panel displaying GPT-4 summaries and the final combined score.

Informal feedback from classmates (some with limited finance background) suggested that:

- The overlay of indicators and pattern markers made technical rules easier to understand in context.
- Confusion matrices and metrics tables made it clear that the problem is hard and that models are only slightly better than chance.
- The combined numeric + textual explanations felt closer to how one would actually reason about a stock (mixing charts and news) than a bare probability score.

In summary, the evaluation confirms that while daily stock direction prediction remains intrinsically difficult, our pipeline successfully integrates pattern mining, supervised learning, live data, and GPT-4 context in a way that is both quantitatively sound and educationally informative. We note that the ML probability and sentiment score are not jointly calibrated; the linear combination is heuristic and used only for interpretability in the prototype.

7 Discussion

This project was less about building a profitable trading system and more about understanding what is realistically achievable with standard data mining tools on noisy financial data, and how to present those results transparently.

7.1 Lessons Learned and What Worked Well

A first lesson is that daily stock direction prediction is genuinely hard. Even after careful feature engineering and model tuning, our best models only modestly outperformed simple baselines. This reinforced the importance of focusing on *pattern reliability*, feature importance, and regime analysis, rather than celebrating small accuracy gains.

The combination of pattern mining and supervised learning worked particularly well from an analytical and educational standpoint. Association rules and sequential patterns gave us interpretable “if-then” statements (e.g., bullish crossover with high volume), while the Random Forest and Logistic Regression models quantified how predictive those patterns were in practice. The fact that feature importance profiles and mined patterns told a consistent story (short SMAs, recent returns, volatility, volume ratios matter most) increased our confidence that the models were not just overfitting noise.

The lightweight “warehouse” design also paid off: having a single consolidated panel [date, symbol, features, label] made it easy to run fair comparisons, add new models, and repeat experiments under different splits. On top of this, the Finnhub + GPT-4

integration, even in prototype form, showed how numerical and textual signals can be fused into explanations that resemble human reasoning.

7.2 What Did Not Work as Well

Not everything worked as initially hoped. In particular:

- The LSTM did not significantly outperform simpler models despite its additional complexity and tuning cost. With limited daily data and strong noise, deep sequence models offered only marginal benefits.
- Some mined patterns that looked strong in-sample (high lift) did not generalize to later periods, especially around the COVID-19 crash. This highlighted how easy it is to find regime-specific artefacts.
- The live prototype was constrained by API limits and did not support a full, statistically rigorous backtest. Our system-level evaluation was therefore qualitative rather than a true live trading experiment.

These shortcomings suggest that more powerful models alone are not a silver bullet; careful evaluation design and humility about what the data can support are equally important.

7.3 Future Directions

There are several natural extensions to this work:

- **Richer Feature Space:** Incorporate market-wide factors (indices, sector ETFs, VIX), fundamental signals (earnings, valuation ratios), and intraday features to capture more nuanced drivers of price moves.
- **Better Uncertainty Handling:** Move beyond point predictions to calibrated probabilities and confidence intervals, and explicitly model regime shifts using techniques such as hidden Markov models or regime switching.
- **More Rigorous Pattern Testing:** Apply statistical tests and multiple-comparison corrections to mined patterns to reduce the risk of data-snooping, and study how patterns evolve over time using rolling windows.
- **Stronger System Evaluation:** Implement a reproducible backtesting layer with transaction costs and position sizing to evaluate the end-to-end system under realistic trading constraints, even if the goal remains educational rather than profit-driven.
- **Extended LLM Integration:** Explore structured prompts that let GPT-4 reason explicitly about conflicts between indicators and news, or generate natural-language “debugging” explanations when the model is likely to fail (e.g., around major events).

Overall, the project shows that even modest-accuracy models can be valuable if they are wrapped in a transparent pipeline that exposes patterns, limitations, and context. Future work can build on this foundation by scaling up the universe of assets, deepening the use of LLMs, and tightening the link between statistical reliability and user-facing explanations.

8 Conclusion

This project set out to study short-horizon stock direction prediction from a *data mining* perspective rather than a pure “beat the market” angle. We focused on three large-cap technology stocks (AAPL, MSFT, GOOGL) from 2016–2024 and built an end-to-end pipeline that spans data understanding, feature engineering, pattern mining, supervised modeling, and a prototype live inference layer with GPT-4–based context.

On the data side, we consolidated raw Kaggle tables into a light-weight warehouse with aligned prices, volumes, engineered technical indicators (SMAs, returns, volatility, RSI, volume ratios), and next-day direction labels. We then used this common feature space for both association/sequential pattern mining and classical ML models (Logistic Regression, Decision Tree, Random Forest, SVM, LSTM). Pattern reliability was quantified via support, confidence, and lift, revealing that some widely used technical configurations (e.g., bullish crossovers with confirming volume) offer modest but non-trivial edges, while others provide little information and are highly regime dependent.

Modeling results showed that all non-trivial models only modestly outperform simple majority and persistence baselines; Random Forest and SVM achieved the best overall trade-off but remained close to the inherent difficulty limit of daily prediction. Feature importance analysis and logistic coefficients consistently highlighted short-term SMAs, recent returns, volatility, and volume ratios as key drivers, aligning with both mined patterns and technical intuition. Finally, a prototype Finnhub + GPT-4 integration demonstrated how numerical signals and news sentiment can be fused into human-readable explanations on a dashboard.

Overall, our main contribution is a transparent, reproducible pipeline that connects pattern mining, interpretable modeling, live data, and LLM-based context into a coherent decision-support system, while honestly reflecting the limits of predictability in real-world financial markets.

References

- [1] T. Li, Y. Luo, Y. Zhang, and H. Zheng, “MASTER: Market-guided stock transformer for stock price forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2024.
- [2] D. Cao, Y. Zhang, Z. Liu, and H. Tong, “Large-scale financial time series forecasting with a multi-faceted model and invariant learning,” in *Proceedings of the ACM International Conference on AI in Finance (ICAIF)*, 2023.
- [3] Q. Chen and H. Kawashima, “Stock price prediction using LLM-based sentiment analysis,” in *Proceedings of the IEEE International Conference on Big Data (BigData)*, 2024 (to appear).
- [4] O. B. Sezer, M. U. Gudelek, and A. M. Ozbayoglu, “Financial time series forecasting with deep learning: A systematic literature review (2005–2019),” *Applied Soft Computing*, vol. 90, p. 106181, 2020.
- [5] A. M. Ozbayoglu, M. U. Gudelek, and O. B. Sezer, “Deep learning for financial applications: A survey,” *Applied Soft Computing*, vol. 93, p. 106384, 2020.
- [6] J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.
- [7] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, “Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques,” *Expert Systems with Applications*, vol. 42, no. 1, pp. 259–268, 2015.

A Honor Code Pledge and Contribution Statement

A.1 Honor Code Pledge

We affirm that this project and report were completed in accordance with the University of Colorado Boulder Honor Code. All analysis, code, figures, and writing are our own work, except where external sources are explicitly cited. We did not give or receive unauthorized assistance on this assignment, and we have accurately represented our contributions below.

A.2 Individual Contributions

Arnav Purushotham.

- Led data ingestion and preprocessing: consolidated the Kaggle price tables, implemented the feature-engineering pipeline (SMAs, returns, volatility, RSI, volume ratios), and built the consolidated panel [date, symbol, features, label].
- Implemented and tuned baseline models (Logistic Regression, Decision Tree) and generated core EDA plots (price trajectories, correlation heatmaps, daily return distribution).
- Drafted the initial versions of the Methodology and Evaluation sections of the report.

Meghasrivardhan Pulakhandam.

- Designed and implemented the pattern-mining components: discretization of indicators, association-rule mining (Apriori/FP-Growth), and sequential pattern extraction, along with computation of support, confidence, and lift.
- Trained and evaluated advanced models (Random Forest, SVM, LSTM), produced confusion matrices and feature-importance plots, and summarized quantitative results.
- Contributed to the Related Work section and wrote the discussion of pattern reliability and regime dependence.

Sneha Nagaraju.

- Implemented the live-inference prototype: Finnhub-based price puller, feature refresh, and integration with GPT-4/VADER sentiment for news summaries and score aggregation.
- Built the system architecture and pipeline diagrams, as well as the dashboard-style visualizations combining predictions, patterns, and textual explanations.
- Led the writing and editing of the Introduction, Discussion, Conclusion, and Appendix, and coordinated overall report formatting in LaTeX.

All team members jointly discussed design decisions, interpreted results, and participated in preparing the final presentation.