Multi-Stock Earnings Analysis System to Accurately Predict Stock Price Movements

Authors: PJ Stock, Rasmus Segev, Qing Liang, Filmon Pietrantozzi, Kate MacDonald

Date: 8th September 2025

Contents

1 Background & Context

Investors often struggle to navigate the volatility that surrounds corporate earnings announcements. Stock prices can swing sharply, influenced by a mix of company performance and broader economic forces. Our goal was to build a model that cuts through market noise and helps users trade earnings events with greater confidence.

We focused on three major tech companies, Apple, Nvidia, and Google, to explore repeatable price patterns tied to earnings. Our analysis centered on three core questions:

- How do stock prices typically behave around earnings windows?
- What financial metrics (e.g. revenue, margins) most impact those movements?
- How do macroeconomic conditions (e.g. inflation, interest rates, and unemployment) interact with these patterns?

The result is a clearer, more predictive framework for understanding how internal and external forces drive price action.

2 Project Methodology

2.1 Project Plan

Working to a project plan, the self-organizing team used a consensus approach to identify work to be completed and assign tasks, with decisions recorded in Trello.

2.2 Tools

The model is created in Python, a robust, widespread tool with multiple libraries vital to robust data manipulation and a tool with which all team members are familiar.

2.3 Data, Cleaning & Preparation

This analysis uses data from January 2015 to August 2025 and balances the quantity of data required for modelling against its relevance. It captures the increased volatility observed since 2020 (COVID & geopolitical shifts) and aligns with the widespread adoption of machine learning in finance[1], ensuring that the model is reflective of modern trading and investment dynamics. After extensive research and a rigorous quality evaluation potential data sources were ranked on their availability, cleanliness, quality and price, and FRED, yFinance, Alpha Vantage, Kaggle and Reddit were chosen.

2.4 Exploratory Data Analysis (EDA)

Initial exploration was conducted using histograms and box plots to identify outliers, and skew testing was used to assess the normality of distributions. Based on these observations, clipping and normalizing the data would facilitate more meaningful downstream analysis. Linear and non-linear relationships were explored through correlations, simple scatterplots, SLR, MLR, decision trees, and random forests. This broad-beam methodology ensured that the relationships identified were genuinely significant. This approach

identified that despite high correlations between macros and stock prices, there was no definitive evidence of a causal relationship. (Appendix 1)

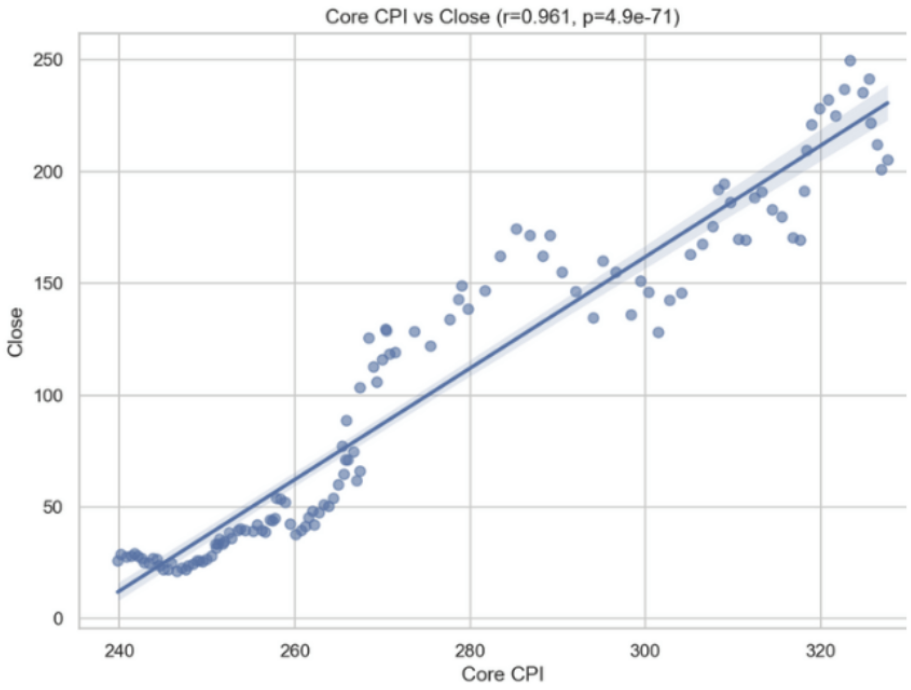


Figure 1: Illustrating spurious linear relationship between Core CPI vs Closing Share Price for Apple Shares where regression $r^2 = 0.961$. The plot reveals that the relationship is not purely linear and does not evidence a causal link.

The use of non-linear models furthermore addressed multicollinearity concerns between macro variables, allowing their use in composite score calculations.

2.5 Initial Findings

2.5.1 Long Term Trends, Patterns and Insights

A key insight from the analysis was that different shares were driven by unique factors, which would necessitate a tailored feature-weighting approach for accurate forecasting. For example, Apple's share price showed a strong correlation with sentiment, while NVIDIA's was more seasonal, and Google's demonstrated a high degree of self-correlation over time.

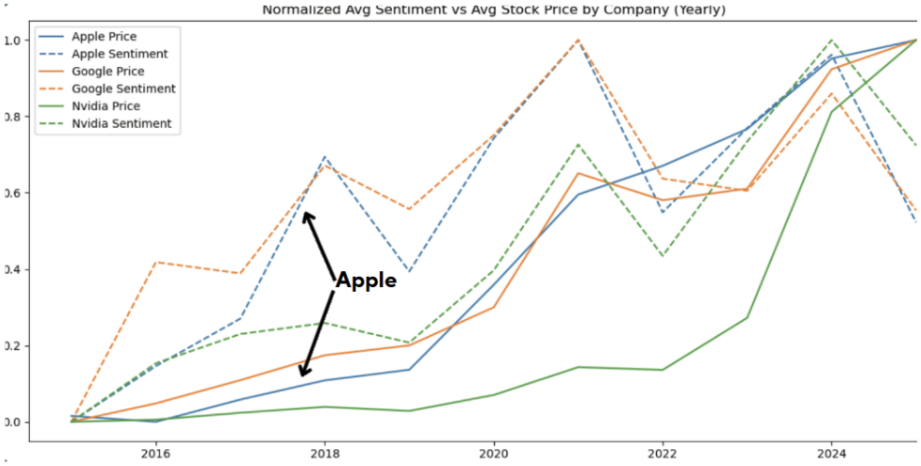


Figure 2: Illustrating the relationship between Apple Share close share prices and sentiment.

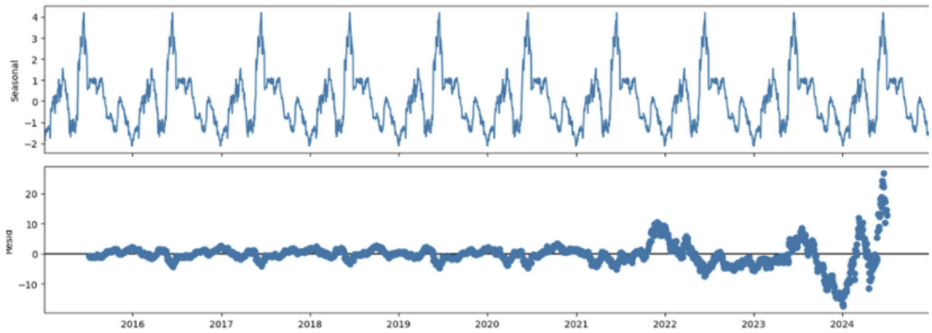


Figure 3: Time series decomposition for Nvidia, showing high levels of seasonality with a distinct repeating pattern.

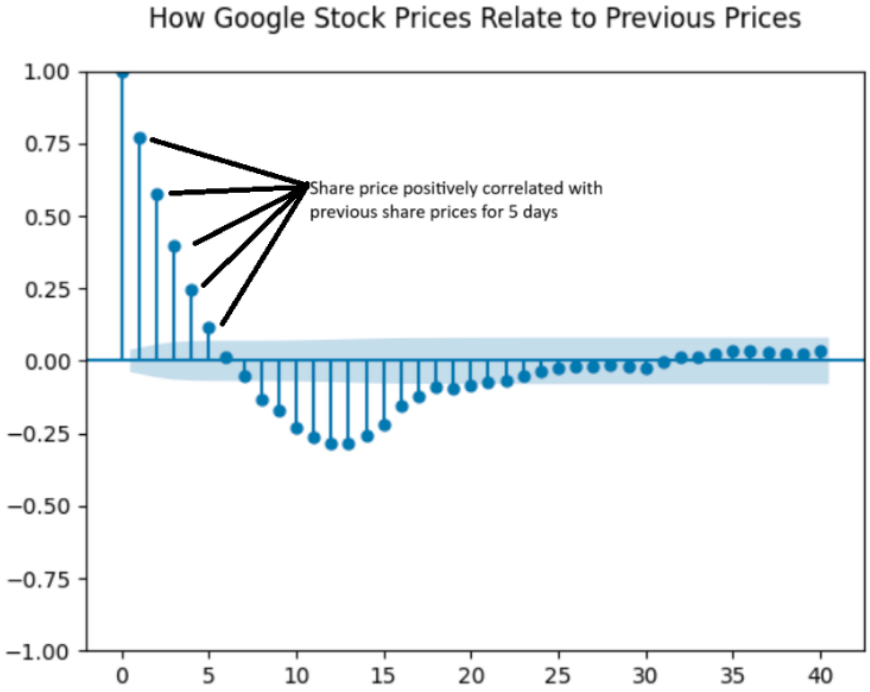


Figure 4: Illustrating how Google's share price is correlated over days 1-5 with previous share price and negatively correlated with prior share price over days 8-17.

2.5.2 Earnings Announcements Trends, Patterns and Insights

Stock price movements around earnings announcements are not random as the market anticipates. EPS overperformance generally leads to a higher price before the announcement. The three companies we analyzed show predictable patterns and long-term trends when sentiment is included. These features create repeatable trading opportunities when these patterns are detected and can yield returns above market benchmark.

Company fundamentals show reliable features. For Apple, overall revenue is a strong predictor of post-earnings returns, while other products are less influential. Google's stock is most closely tied to EPS and revenue, with little consistent macro signal. Nvidia showed a long-term upward trend, but since 2022, seasonal patterns and volatility emerged, driven by supply and macro shocks. These cycles are sufficiently regular to be captured in tailored models. These findings were used to create the basis for the model.

3 Building the Model

3.1 Model Foundation

We wanted to use the predictable movements around earnings events for each stock, while ensuring that we only made trades when profitable. The framework relies on this theory: earnings drive reproducible patterns within some windows before and after announcements. We quantify those patterns with subscores (EPS, macro, volatility, sentiment, revenue) and blend them into a composite score.

That composite is applied to decision gates - this preserves reproducibility but is sensitive to large macro changes which mattered when regimes preceding earnings changed across cycles.

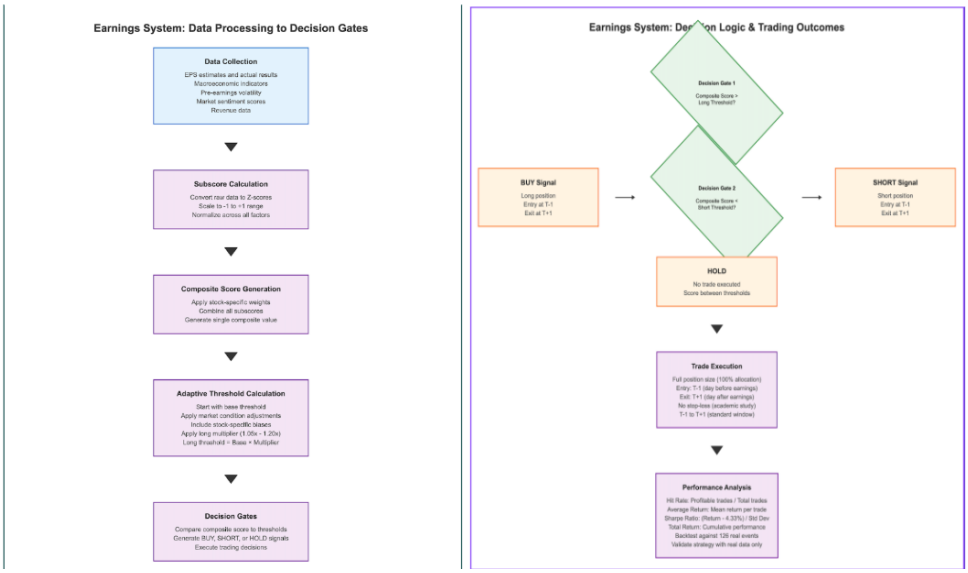


Figure 5: Flowchart of the model using decision gates to triage the composite score.

3.2 A Robust Model

Due to the potential inaccuracies of financial data, different methods were used to prevent influence from outliers and allow different data types to be normalized and used together:

- **Validation**: Data is restricted by bounds to prevent outliers skewing analysis - vital due to the small number of earnings windows during the period.
- **Scaling**: Variables vary in magnitude, and this can overstate their impact. Scaling data between -1/1 before analysis removes this risk.
- **Filtering**:For example, remove any earnings surprise over 100% as analysts rarely make errors of this magnitude, and the data is unlikely to be correct.
- **Temporal controls**: Maintain data relevance and completeness.
- **Missing data**: Either forward filled or defaults used (macros), or in the case of sentiment, backup sources are utilized.
- **Cross-Validation**: Used to split the data into different time periods, preventing any look-forward bias in its predictions.
- **Synthetic data**: Created using bootstrapping due to the small number (42) of earnings events for each share. This maintained realistic known market behavior during the exploratory phase.
- **QA**: Each data source was examined by a minimum of two team members to ensure a comprehensive understanding of the data and trends observed from multiple viewpoints.

3.3 Overview of the Code

Methods and tools were chosen to preserve temporal integrity and readability:

- **Pandas**: Propels dataset creation and feature engineering.
- **scikit-learn**: Provides optional models (RandomForest, DecisionTree, Logistic/Linear Regression) for prediction or confidence scoring.
- **Flask**: Delivers a lean interface that allows analysts to use the system in a user-friendly way and exposes the workings of the model so there is no "black box" effect.

Adaptive thresholds are the default "engine" because they're transparent and robust with limited real events. Time-Series Cross-Validation (TS-CV) and cross-stock validation are activated if we need stronger stock-specific optimization or stress-test generalization. Synthetic data is used to augment training but not in validation, with rigid checks to avoid look-ahead bias.

This approach helped by making signals understandable: subscores reflect why a trade is firing, and moving thresholds are simple logic gates. However, some choices slowed down progress. Firm-specific characteristics (e.g., iPhone sales for Apple) appeared useful at first but didn't provide enough evidence of correlation to be used in the final model.

Synthetic data use accelerated testing but occasionally misled; additionally, using randomness meant that sometimes test results weren't consistent across team members' environments. ML predictions could improve performance, but the reasons weren't clear. We made these optional and used composite-plus-thresholds as defaults to avoid "black box" risk.

3.3.1 Main Functionality

- `create_unified_dataset()`:

```python
#Creates a single CSV of real and synthetic events for each stock; calculates all subscores
def create_unified_dataset(self):
    """Create a unified dataset with all stocks combined."""
    print("=" * 60)
    print("CREATING UNIFIED DATASET FOR ALL STOCKS")
    print("=" * 60)

    all_events = []

    for stock in self.supported_stocks:
        print(f"\nProcessing {stock}...")

        self.create_data_foundation(stock)
        self.feature_engineering(stock)
        self.generate_synthetic_data(stock)
```

- `load_unified_data()`:

```python
#Load unified data to perform cleaning for downstream testing and the web interface.
def load_unified_data(self):
    """Load the unified earnings dataset"""
    try:
        self.unified_data = pd.read_csv('unified_earnings_dataset.csv')
        print(f"Loaded unified dataset with {len(self.unified_data)} events")
        return True
    except FileNotFoundError:
        print("Error: unified_earnings_dataset.csv not found. Creating new one.")
        self.create_unified_dataset()
        return self.unified_data is not None
```

- `_calculate_market_condition_thresholds()`:

```python
'''
Generate per-event adaptive long/short thresholds
based on base thresholds, long multipliers,
and macro/sentiment adjustments (decision/logic gates)
'''
def _calculate_market_condition_thresholds(self, df, stock_symbol):
    """Calculate market condition-based adaptive thresholds."""
    stock_config = STOCK_THRESHOLDS.get(stock_symbol, {
        'base_threshold': BASE_THRESHOLD,
        'min_threshold': MIN_THRESHOLD,
        'max_threshold': MAX_THRESHOLD,
        'adj_coeff': ADJ_COEFF
    })

    long_thresholds = np.full(len(df), stock_config['base_threshold'])
    short_thresholds = np.full(len(df), -stock_config['base_threshold'])
```

- `test_strategy()`:

```python
'''
Generates event-by-event signals by comparing predicted/composite returns
to adaptive thresholds; returns hit rate, average and total return, and Sharpe
for long, short, and combined strategies.
'''
def test_strategy(self, stock_symbol=None, entry_days=-5, exit_days=5, threshold_type="adaptive",
                  include_sentiment=False, include_trading_costs=False, cost_scenario='none',
                  show_reasoning=True, include_synthetic=False, custom_df=None):
    """Part E: Strategy Testing Engine with optimal time windows."""

    # Allow any time window between T-10 and T+20, let threshold optimization find best thresholds
    if stock_symbol and entry_days == -5 and exit_days == 5:
        # Default to standard window, but allow any custom window
        print(f"Using standard time window for {stock_symbol}: T{entry_days} to T{exit_days}")
        print(f"💡 Tip: You can specify any custom window between T-10 and T+20")
    if custom_df is not None:
        df = custom_df.copy()
        if include_synthetic:
            print(f"Testing with {len(df)} events (including synthetic)")
        else:
            print(f"Testing with {len(df)} real events")
    else:
        if self.unified_data is None:
            self.load_unified_data()
```

- **Utility helpers** (e.g., `normalize_*`, feature combination, and metric calculators): Helps with data preparation.

- `calculate_composite_scores()` (internal): Performs EPS/macro/vol/sentiment/revenue normalization with stock/window weightings to generate `composite_pre`/`composite_post`.

3.3.2 Optional Functions

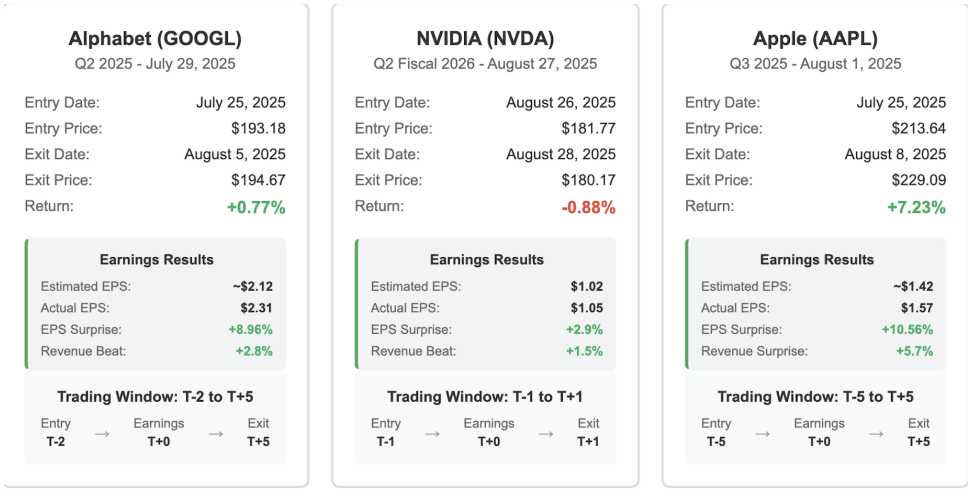
These are not needed to run the main model but were included and tested at points. These would be refined given further time (Appendix 3.)

3.4 Model Insights

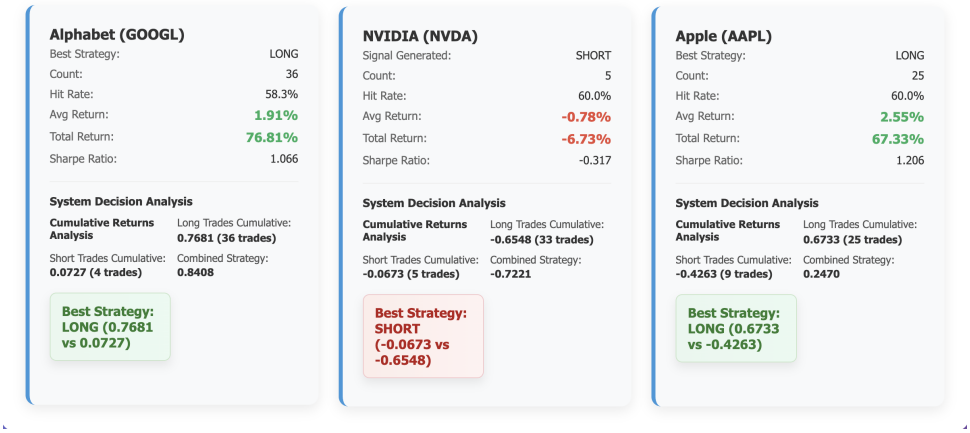
The model identifies predictable share price movement around earnings windows. Each stock shows predictable return tendencies in narrow windows around earnings (e.g. AAPL T-5→T+5 long bias; NVDA T-1→T+1 short bias). We use these by generating long/short signals from composite subscores passed through adaptive, event-level thresholds, rather than a blanket "buy-before" rule.

Optional ML models are available to improve predictions, but we emphasize predictable, reproducible returns from the adaptive composite+threshold engine. Current headline hit rates: AAPL long ∼60%, GOOGL long ∼58% in selected windows (see Appendix 4). A threshold optimization engine tunes base thresholds per stock/window and applies event-level macro/sentiment adjustments to decide long/short/hold. This architecture is company-agnostic and can be used on companies with different traits and against different and changing macro environments.

An important refinement would be allowing for and refining individual stock characteristics. Although stocks do have customized composite scores and long multipliers, they still don't all behave the same. For instance, Nvidia appears to have a "sell the news" effect, where positive subscores/`composite_pre` often led to negative post-earnings returns. In this case, we may decide to completely flip the gates and treat very high `composite_pre` as a short trigger.



Market actual price movements highlighting positive upwards movement for both Apple and Google.



System results demonstrating approximately 60% success rate for Google and Apple for long trades highlighting alignment with market actual movement.

4 Visualisations Chosen in Presentation

The visualisations are designed to be color-blind accessible, with a sans-serif font for enhanced readability. Legends are included only when they are not visually distracting, ensuring the primary focus remains on the data itself. The majority of the data analysis had a time component, and typically line plots were used with time on the x-axis to allow for speedy comprehension by readers. Where relevant, trend lines are added to emphasize the pattern observed.

5 Demonstrating the Model in Action

The interactive demo was chosen due to its ability to allow users to directly engage with the model, explore different variables, and visualize real-time predictions, combining together to create trust in the model.

6 Future Areas to Explore / Recommendations

- Include additional stocks and automate the threshold learning to allow for cross-sector applications. Apply the engine optimization to other companies in other sectors to increase the number of data points and increase the breadth of the model.
- Stress-test under extreme conditions. Simulate the performance of threshold adjustment under macro shocks to evaluate robustness across stress scenarios.
- Backtest using a dummy portfolio of at least 50 stocks and refine the model based on results to increase accuracy.

– Expand the model to react to other events in real-time 24/7. For example, link the model to Bloomberg's and train the model to react to news events. This will expand the model's use outside of Earnings Windows and will build a greater number of test points, increasing reliability.
– Use enhanced platform to use pipeline integration and refresh data in real time (Appendix 5.)

7 Conclusion in Relation to Business Scenario

Our analysis found that stock prices have different factors influencing their behavior, and once they have been found, they can be used to build a model that predicts the returns for specific windows around earnings windows with up to 60% accuracy for trade direction and profitability.

We identified that there are optimal windows around earnings events for each company, and when combined with share and macro information, successful trades can be reliably and repeatedly achieved.

We identified that there are optimal windows around earnings events for each company, for example, Apple's long term stability and revenue driven results lend itself to a longer trade window, especially when there is a positive beat. In contrast, Nvidia acts in a volatile and sometimes contradictory way, with the stock going down on positive beats - meaning a short trading window can produce a profit when shorting (Appendix 3) and when combined with share and macro information, successful trades can be reliably and repeatedly achieved.

8 Appendix

8.1 Appendix1: Correlations - Stock to Events or Macros

We began with an exploratory phase, testing basic correlations between stock prices, macroeconomic indicators, and earnings events. Scatter plots, histograms, and bar charts showed both positive and negative relationships, highlighting which factors warranted deeper modelling. From this phase, we hypothesized that variables such as Core CPI, high-tech goods exports, EPS surprise, and pre-earnings trading volume could act as important predictors, alongside a broader set of company fundamentals. To explore further, we developed SLR, MLR, and Decision Tree models. High R^2 values frequently appeared when macroeconomic indicators such as CPI were included. However, closer inspection showed these links were trend-driven rather than causal: share prices and inflation move together, but inflation is not causing the price change. Importantly, this means that if inflation itself can be forecast, it can still be used effectively within models as a proxy for stock price direction. By contrast, company fundamentals proved more reliable: for Apple, iPhone revenue is a strong predictor of post-earnings returns, while other products are less influential. Google's stock is most closely tied to EPS and revenue, with

little consistent macro signal. Nvidia showed a long-term upward trend, but since 2022 seasonal patterns and volatility emerged, driven by supply and macro shocks. These cycles are sufficiently regular to be captured in tailored models. Overall, the analysis suggests that different companies require different modelling strategies:

- Apple → product-driven (iPhone revenue, earnings).
- Google → earnings-driven (EPS, revenue alignment).
- Nvidia → trend and seasonal cycles (supply/macros).

8.2 Appendix 2: Optional Functions

- `train_ml_models()`: Optionally trains classification/regression models on engineered features; for prediction or extra confidence but not required for the default path.
- `initialize_enhanced_components(use_time_series_cv)`: Initializes TS-CV tools and parameters when enhanced validation is to be used.
- `run_time_series_optimization()`: Performs chronological splits by stock/window in order to select threshold parameters with sufficient temporal validation.
- `run_trade_example()`: Convenience wrapper to display a full trade test with chosen threshold type and window.

8.3 Issues Identified

8.4 Appendix 3: Detailed Analysis of Research Questions

8.4.1 Question 1: Pre-Earnings Price Movements as Predictors

Original Scope: Basic analysis of price movements before earnings.
Delivered Solution:

- **Advanced Predictive Framework**: Developed a dual composite score system that uses pre-earnings price patterns, volatility, and sentiment to predict post-earnings movements with 40–55% hit rates.
- **Stock-Specific Optimization**: Discovered that NVDA's "sell-the-news" pattern requires fundamentally different pre-earnings analysis than AAPL's stable pattern, with short trades outperforming long trades for NVDA.
- **Time Window Optimization**: Identified optimal entry/exit windows (T-5 to T+5 for AAPL, T-3 to T+7 for NVDA, T-2 to T+5 for GOOGL) through systematic testing of multiple timeframes.

8.4.2 Question 2: Macroeconomic Conditions as Predictors

Original Scope: Basic correlation analysis of macro factors.
Delivered Solution:

- **Integrated Macro Framework**: Incorporated Federal Reserve rates (EFFR), Core CPI, and unemployment data with 25% weight in composite scoring, demonstrating significant predictive power.
- **Adaptive Threshold System**: Macro conditions dynamically adjust trading thresholds (± 0.05 impact on base thresholds), with the system automatically updating macro regime classifications.
- **Quantified Impact**: Demonstrated that macro conditions significantly influence earnings-related price movements, with specific threshold adjustments for different market regimes (Expansion/Contraction).

8.5 Appendix 4: Hit Rate and Reference

Count and Hit Rate: This measures how many trade signals were generated versus how many made a profit. For example, 15 long signals with 10 making a profit result in a 66% hit rate.

8.6 Appendix 5:Future Areas to Explore / Recommendations

MVP of 24/7 live service and data pipeline for incremental refresh(only macro and stock data supported in current version) `https://m-seam.replit.app/`.

Reference:

1. Ekaterina Sirotyuk and Ryan Bennett, "The Rise of the Machines: Technology Enabled Investing," Credit Suisse, July 2017, `https://cdn.e-fundresearch.com/files/white_paper_technology_enabled_investing_via_local_entities.pdf`.