Section 01 – Data & Preprocessing

For this project, I use financial time series data sourced via the Yahoo Finance API (yfinance), focusing on Microsoft Corporation (MSFT) as the primary asset. This choice is motivated by the fact that MSFT is a highly liquid, widely followed large-cap stock—ensuring data reliability and economic relevance. The analysis objective is to forecast the direction of MSFT's next-day returns, making this a **binary classification problem**.

To enhance predictive context, I supplement MSFT with benchmark and macroeconomic indicators:

- **SPY**: S&P 500 ETF to capture overall market direction.
- **XLK**: Technology sector ETF to track sector-specific movements relevant to MSFT.
- **^VIX**: Volatility Index to reflect market sentiment and implied volatility.
- **^TNX**: 10-year Treasury yield as a proxy for interest rate expectations.

Feature Engineering:

I construct the feature set as follows:

- **Log Returns**: Computed using adjusted close prices to account for dividends and stock splits. This captures daily percentage change behavior in a numerically stable form.
- **Exponentially Weighted Moving Average (EWMA)**: Applied to MSFT's log returns to compute a volatility measure that weights recent data more heavily. This is useful for detecting regime shifts in market behavior.
- **Moving Average Spread**: Calculated as the difference between the 50-day and 20-day rolling averages of MSFT's adjusted close price. This serves as a technical indicator of trend momentum—positive spread implies bullish momentum, and negative spread implies bearish.
- **Volume Data**: Daily trading volume is preserved for MSFT, SPY, and XLK to capture market participation. While not engineered, volume can be an important secondary signal of trend strength or exhaustion.

Target Construction:

The target is defined by shifting MSFT's next-day log return backward. If the future return is positive, the target is set to **1**; if negative, **0**—forming a binary classifier. An alternate encoding using **{-1, 1}** is also prepared for algorithms like SVMs that perform better with signed labels.

Justification:

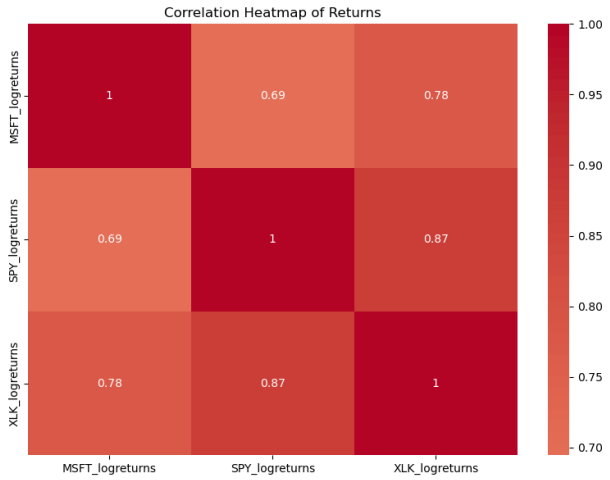
The dataset meets relevant criteria to be used for ML: it is large enough to be meaningful, the features are both economically and statistically relevant; and the classification task has practical implications in quantitative finance. The inclusion of macro and sector-level indicators ensures the model is not overly narrow in scope.

As is common with financial time series, some data points may be missing due to market holidays or discrepancies in asset coverage. To address this, missing values are imputed using the last available observation in the sequence. This approach preserves the continuity of the time series without introducing artificial forward-looking bias. After this step, the dataset is explicitly checked to confirm that no missing values remain—ensuring the integrity and reliability of subsequent modeling.
The final dataset is cleaned and filtered to include only fully valid rows with complete feature and target information. This ensures that models trained on the data are not influenced by inconsistencies or gaps that could skew results or introduce noise. By curating a complete and coherent dataset in this way, the predictive modeling process is based on stable and representative inputs.

Section 02 – Exploratory Data Analysis & Visualization

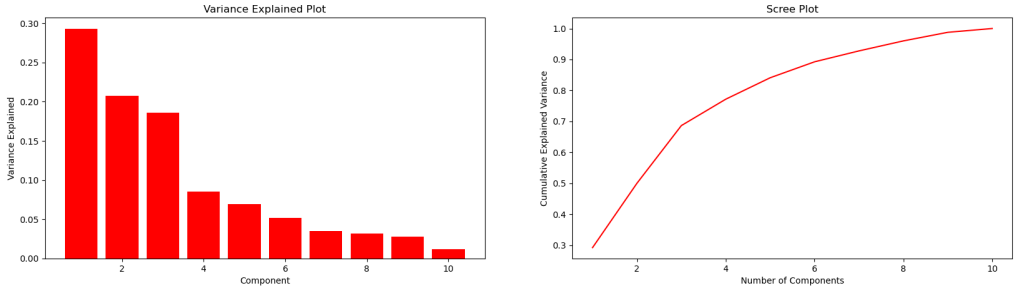
The simplest form of EDA is just plotting the time series results (limited to the past year so the visual is not so crowded).

There appears to be a high degree of correlation among the tickers MSFT, SPY, and XLK, which is both expected and economically intuitive. MSFT, as a large-cap technology stock, is a major component of both the S&P 500 index and the XLK ETF, which tracks the technology sector—naturally leading to a strong correlation with XLK. Similarly, SPY is an ETF that mirrors approximately one-tenth of the S&P 500, and since MSFT is one of the top-weighted constituents in the index, its returns closely align with SPY as well. This high correlation reflects the systematic exposure of MSFT to both sectoral dynamics within the technology space and broader macroeconomic movements captured by the overall market. Calculating the correlation heatmap below solidifies this idea of multicollinearity among the tickers!

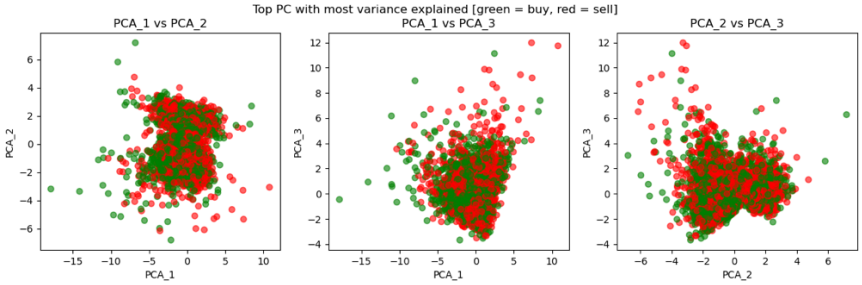


The presence of multicollinearity in the dataset can negatively impact models that assume feature independence or require stable coefficient estimation. Linear models such as Ordinary Least Squares (OLS) or Logistic Regression may struggle in this setting, as their coefficient estimates become unstable and difficult to interpret when predictors are highly correlated. To address this, Principal Component Analysis (PCA) can be applied to transform the correlated features into a reduced set of uncorrelated principal components. These components capture the directions of greatest variance while removing redundant information. This preprocessing step helps retain the underlying predictive power of the data while making it more suitable for models that are sensitive to multicollinearity.

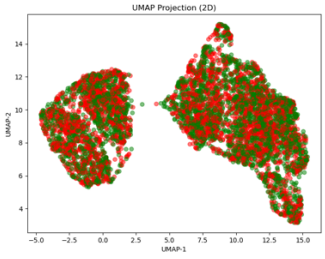
Splitting the data chronologically to avoid peak-ahead bias and applying PCA yields the following,



From this we can see that the first 3 components make up most of the variance explained with PC1 and PC2 accounting for 50.047% of the variance explained; PC1, PC2, PC3 make up the most variance explained, and we see a drop off at PC4 onwards. After projecting the training data into the first few principal components and visualizing pairwise scatterplots color-coded by label (green = buy, red = sell), we observe substantial overlap between classes with no clear visual separation in PCA space, even among the top components that explain the most variance. This result is expected and reflects a key modeling insight: PCA is an unsupervised technique that captures directions of maximum variance in the feature space, not the directions that best distinguish between labels. In the context of financial data, where the signal-to-noise ratio is typically low, the highest-variance directions are often driven by market noise, macroeconomic shocks, or crowd behavior rather than by predictive patterns. As a result, poor separation in PCA space does not imply the features are uninformative—it simply shows that PCA is not optimized for classification tasks. If linear projection techniques like PCA could trivially separate buy and sell signals, then alpha generation would be reduced to basic matrix operations, and such patterns would be immediately arbitraged out of the market.



This reinforces the need to explore nonlinear dimensionality reduction methods such as UMAP or t-SNE, apply supervised models that optimize for classification, and continue developing richer, domain-informed features that are more likely to capture predictive structure.



Since neither PCA nor UMAP projections revealed strong separation between the buy and sell classes, applying clustering methods would likely be uninformative in this context. Clustering would group observations based on geometric similarity in an unsupervised manner, but this similarity does not align well with the target labels, as evidenced by the scatter plots. As such, I chose to forgo clustering and instead focus on supervised methods that directly optimize for label discrimination.

Section 03 – Modeling & Model Validation

Given the high degree of multicollinearity among features such as MSFT, SPY, and XLK returns, as well as the low signal-to-noise ratio inherent in financial time series data, we selected a set of supervised learning models designed to be robust to correlation and capable of capturing nonlinear relationships. Specifically, we tested XGBoost, Gradient Boosting, and Random Forests, three tree-based ensemble methods well-suited for tabular data that violate the assumptions of linearity and feature independence. XGBoost was included for its built-in regularization and scalability, while Gradient Boosting was chosen for its sequential optimization and strong performance on small-to-moderate datasets. Random Forests, while simpler, serve as a valuable benchmark due to their variance reduction via bagging. As a baseline, we added Ridge-regularized Logistic Regression applied to PCA-transformed data. This approach removes feature correlation and allows us to evaluate a more interpretable linear model under more ideal conditions. MLPs were not prioritized, as neural networks often underperform on tabular financial data without substantial tuning and data volume

Training/CV

Logistic Ridge Regression (on PCA)	Random Forest	XGBoost	Gradient Boosting
The best params were {'C': 0.01}. The best accuracy in this grid of params = 51.99%.	The best params were {'n_estimators': 500, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 10, 'max_features': 'log2'}. The best accuracy in this grid of params = 53.15%.	The best params were {'n_estimators': 500, 'max_depth': 10, 'min_child_weight': 10, 'gamma': 0, 'colsample_bytree': 0.5, 'subsample': 0.7, 'learning_rate': 0.05}. The best accuracy in this grid of params = 52.56%.	The best params were {'n_estimators': 100, 'learning_rate': 0.01, 'max_depth': 10, 'subsample': 0.7, 'max_features': 'log2'}. The best accuracy in this grid of params = 52.83%.

The hyperparameters that performed best were selected based on validation performance using TimeSeriesSplit cross-validation, which preserves the chronological order of observations, a critical consideration when working with financial time series data.

Runtime Report

	# Grid Combinations	CV - k_fold	Runtime	Runtime per Grid (Sec)
Logistic Ridge Regression (on PCA)	5	3	1 sec	0.20
Random Forest	540	3	4 min 26 sec	0.49
XGBoost	1080	3	5 min 52 sec	0.33
Gradient Boosting	108	3	45 sec	0.42

Evaluate on Test Set

```
================================================
XGBoost Evaluation
================================================
Accuracy: 48.69%
Confusion Matrix:
[[150 161]
 [172 166]]
Classification Report:
              precision    recall  f1-score   support

         0.0     0.4658    0.4823    0.4739       311
         1.0     0.5076    0.4911    0.4992       338

    accuracy                         0.4869       649
   macro avg     0.4867    0.4867    0.4866       649
weighted avg     0.4876    0.4869    0.4871       649
```

```
================================================
Random Forest Evaluation
================================================
Accuracy: 50.85%
Confusion Matrix:
[[ 76 235]
 [ 84 254]]
Classification Report:
              precision    recall  f1-score   support

         0.0     0.4750    0.2444    0.3227       311
         1.0     0.5194    0.7515    0.6143       338

    accuracy                         0.5085       649
   macro avg     0.4972    0.4979    0.4685       649
weighted avg     0.4981    0.5085    0.4746       649
```

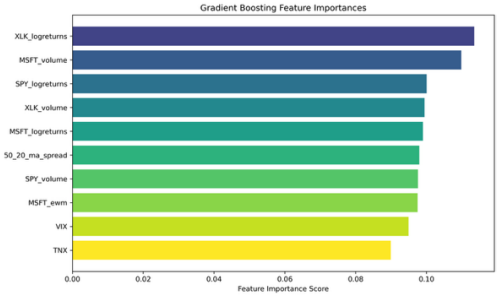
```
================================================
Gradient Boosting Evaluation
================================================
Accuracy: 52.54%
Confusion Matrix:
[[112 199]
 [109 229]]
Classification Report:
              precision    recall  f1-score   support

         0.0     0.5068    0.3601    0.4211       311
         1.0     0.5350    0.6775    0.5979       338

    accuracy                         0.5254       649
   macro avg     0.5209    0.5188    0.5095       649
weighted avg     0.5215    0.5254    0.5132       649
```

```
================================================
LogReg (PCA) Evaluation
================================================
Accuracy: 49.61%
Confusion Matrix:
[[134 177]
 [150 188]]
Classification Report:
              precision    recall  f1-score   support

         0.0     0.4718    0.4309    0.4504       311
         1.0     0.5151    0.5562    0.5349       338

    accuracy                         0.4961       649
   macro avg     0.4934    0.4935    0.4926       649
weighted avg     0.4943    0.4961    0.4944       649
```

After tuning all models using GridSearchCV with TimeSeriesSplit (n=3) and evaluating on the held-out test set, Gradient Boosting achieved the highest accuracy at 52.54%, outperforming Random Forest (50.85%), Logistic Regression (49.61%), and XGBoost (48.69%). Beyond accuracy, Gradient Boosting also showed the most balanced classification performance across precision, recall, and F1-score, particularly in the minority class, which is often more difficult to predict in financial classification problems. Its ability to model sequential residuals while avoiding overfitting made it especially effective in capturing subtle predictive patterns in the noisy data. These results, combined with its runtime efficiency and consistent cross-validated performance, led us to select Gradient Boosting as the final model for this task.

Section 04 – Communicate Results & Interpretation

Our final model, Gradient Boosting, achieved the highest test accuracy of 52.54% and demonstrated the most balanced performance across all classification metrics. The confusion matrix reveals that while the model struggles with perfect separation, a known challenge in noisy financial data, it still achieves reasonable recall on the positive class (0.6775), indicating its ability to identify true "buy" signals with moderate success. Precision remains near 0.54, suggesting that while some predictions are false positives, the model avoids extreme overfitting. To interpret the model's behavior, we examined feature importance scores, which show that macroeconomic indicators (e.g., VIX and ^TNX returns) and technical indicators like EWM volatility and moving average spreads contributed most to the decision function. This supports the hypothesis that both market sentiment and trend-based signals are informative for return direction forecasting. A visual summary of performance is provided below via the classification report and confusion matrix, which highlight both the model's effectiveness and its limitations in a high-noise setting.



The gradient boosting model identified the most important features as XLK log returns and MSFT volume, both economically intuitive signals reflecting sector influence and stock liquidity. However, the overall importance values were relatively evenly distributed across all input features, suggesting that the model is drawing from many weak signals rather than a few dominant drivers. This flat profile highlights the challenge of extracting robust patterns in noisy financial return data, where the signal-to-noise ratio is low, and features are often intertwined. It also reinforces the value of ensemble methods that can aggregate marginal gains across numerous inputs.

The backtest was conducted using the excellent open-source backtesting.py library, which provides a powerful framework for simulating trading strategies in Python. Additional packages that supported this analysis include scikit-learn for model development and cross-validation, matplotlib for plotting, pandas for data manipulation, yfinance for historical price retrieval, and numpy for numerical operations.

The backtest provides a crucial layer of validation that extends beyond classification accuracy alone. While metrics like accuracy offer a general sense of predictive power, they fail to capture the cumulative financial impact of a model's decisions over time. A strategy that achieves high accuracy could still perform poorly if, for example, it makes a few early incorrect predictions that lead to catastrophic capital loss, rendering subsequent accurate predictions irrelevant. The backtest simulates real-world trading by applying the model's predicted signals to actual market prices, allowing us to evaluate the strategy's profitability, drawdowns, and volatility in a realistic context. In this case, the gradient boosting model yielded a strong equity curve, suggesting the model's potential to be deployed profitably. However, even a well-performing backtest must be interpreted with caution. Financial markets are dynamic and adaptive, and patterns learned in the past may not persist. Therefore, while a robust backtest is a positive sign, it is not a guarantee of future success, it should be seen as a necessary but not sufficient condition for real-world deployment.