Preprocessing and Feature Engineering Report

Predicting Next-Day S&P 500 Direction Using Price and Volume Data

Group A

Team Lead: Ross Schanck

Recorder: Auriana Anderson

Spokesperson: Chase Golden

November 24, 2025

Background

Our project examines whether short-term movements in the S&P 500, such as recent returns or basic technical indicators, actually offer any real meaningful clues about what the market might do the next day. We initially framed this in broad terms, but after some discussion we refined it into a more direct question, which felt more practical for what we needed.

Research Question:

Can historical S&P 500 price and volume data help predict whether the next day's closing prices will rise or fall?

Hypothesis and prediction:

We're keeping expectations realistic. Short horizon signals like momentum, brief volatility spikes, and moving average behavior might contain just enough structure for a model to beat chance by a small margin. If logistic regression or a random forest achieves even 52-60% accuracy, that would be great. It would suggest these engineered features capture a modest but real predictive pattern.

Methods

Overview of Our Plan

Last week we aimed to wrap up preprocessing, handle missing rows caused by rolling windows, add a set of new features, and decide whether PCA would be useful. After revisiting our exploratory work, we followed most of that plan and even added a few extra features along the way. Interestingly, both PCA and KMeans turned out to fit that dataset better than expected.

Preprocessing

We pulled S&P 500 data from 2010-2025 using the yfinance API, which provided standard fields (Open, High, Low, Close, Adjusted Close, and Volume). The data was already fairly clean, so no formatting fixes were really needed. We reset the index, so the data became a column and then sorted everything chronologically.

Missing Data

Missing values appeared only in features we created ourselves:

- Percent Change (missing first row)
- Rolling means (MA_3, MA_5, MA_10)
- Rolling volatility metrics
- Lagged returns (1-7 days)

- Sharpe-like ratio

These gaps were expected because rolling windows require prior observations. Once all features were generated, we dropped rows with missing values.

Outliers

We didn't remove outliers since large jumps are normal in market data and removing them risks distorting what we are trying to prove with the model.

Train Test Split

We used an 80/20 chronological split, so the model evaluated only future data

- Training: 2010-2021 (3019 rows)
- Testing: 2021-2024 (755 rows)

We double-checked the date ranges just to be safe and to prevent overlap or leakage.

Feature Engineering

We engineered several features that capture short-term momentum, volatility, trend behavior, and risk adjusted performance.

1. Percent Returns and Lagged Features

We calculated daily percent returns and created lagged versions at 1, 2, 3, 5, and 7 days. Short lags capture immediate market reactions, while the 7-day lag often reflects weekly behavior. These features are pretty common in predictive finance research, so including them made sense and didn't feel forced.

2. Rolling Price Indicators

Moving averages (MA_3, MA_5, MA_10) helped smooth daily noise and highlight short-term directional trends, which was helpful.

3. Volatility Measures

We added two volatility-related features:

- Rolling standard deviation of closing prices
- Rolling standard deviation of returns

Both provide snapshots of short-term market turbulence, at least in the near term.

4. Trend Based Return Indicator

The 10-day rolling mean of returns was included to pick up mid momentum effects, at least to some extent.

5. Sharpe-Like Ratio

We constructed a simplified return to volatility measure (sharpe_like_10d). It isn't a full Sharpe ratio, but it still reflects short-term risk-adjusted performance.

6. Target Variables

We labeled each row Up if the next day's closing price was higher than the current day's, otherwise Down. For modeling, this became a binary variable (1 or 0)

Results and Interpretations

Cleaning and Transformation

After removing rolling related rows, the dataset contained no missing values. We kept outliers to preserve the authenticity of market movements, even though it makes the data a little messier. The binary target variable worked as intended for classification.

Processing Outcomes

We double-checked the timeline boundaries to ensure no leakage. With time series data, this mistake is common, so verifying it will help save a lot of headaches further on in the project.

Unsupervised Feature Engineering

Even though we weren't sure at first if PCA would help, our results showed it was appropriate.

Principal Component Analysis (PCA)

We standardized all numeric features and applied PCA. Keeping components that explained 95% of the variance gave us 9 principal components. Because many of our engineered features were correlated, PCA helped reduce redundancy. Honestly, the cumulative variance plot made the cutoff surprisingly easy to choose.

KMeans Clustering

We tested cluster counts from 2 to 7 and selected k=3, mainly because inertia flattened noticeably after that point. These cluster labels were added to both the training and test sets and may serve as additional model features later.

Supervised Feature Engineering Plan

Our baseline models are logistic regression and random forest, mostly because they give quick insight into feature importance.

From here, we plan to:

- Examine logistic regression coefficients
- Review random forest feature importances
- Use recursive feature elimination (RFE)
- Test interaction terms

Discussion and Next Steps

Key Takeaways

The dataset is now fully cleaned and structured for the most part. Our feature set covers momentum, volatility, short-term behavior, and some basic risk-adjusted measures. It's well aligned with our goal of predicting next-day direction. PCA and KMeans were more useful than expected. Early model tests show logistic regression slightly outperformed random forest, at least in the initial runs.

Next Steps

Our upcoming work includes a few things we still need to check:

- Training logistic regression, random forest, and gradient boosting models
- Evaluating accuracy, precision, recall, and F1-score
- Using model outputs to refine feature selection
- Testing cluster labels as additional predictors
- Checking for class imbalance
- Expanding the indicator set and re-running PCA if needed

GitHub Repository

https://github.com/aurianaanderson/StockMarketTrends/tree/main