

Predicting S&P 500 Direction with Ensemble Methods

Christian Weißmeier Farkas Tallos

Statistical and Machine Learning (2025/26)

November 13, 2025

Agenda

“Stock returns are predictable, but not by much.”

— John H. Cochrane, *Asset Pricing* (2005)

- Forecasting stock market direction is one of the most classical and challenging tasks in finance.
- Weak predictability can matter for portfolio allocation and risk management.

Our Set-Up:

- We focus on predicting whether the S&P 500 index goes **Up** or **Down** next month.
- We evaluate statistical learning methods in a time-series context.
- Introduce the role of regularization and nonlinear models.
- Compare linear vs. nonlinear classification models (Elastic Net vs. Random Forest).

- We created our own monthly dataset from S&P 500, FRED, and FRBSF data banks (1990–today).
- **Target:** Monthly S&P 500 market direction (**UP** or **DOWN**) in $t + 1$.
- **Predictors:**
 - ① **Market data:** Lagged S&P 500 returns (up to five months) and trading volume changes.
 - ② **Macroeconomic indicators:** CPI, Federal Funds Rate, NBER recession dummy.
 - ③ **Volatility:** Lagged changes in the VIX index.
 - ④ **Sentiment:** Daily News Sentiment Index.
- All features are lagged to avoid look-ahead bias.

- Most macro-financial time series are **non-stationary** in levels.
- We therefore use:
 - **Lagged returns** instead of prices.
 - **Changes** in VIX and sentiment rather than levels.
 - **Changes in macro variables** to preserve temporal causality.
- This transformation makes features approximately stationary, ensuring:
 - Stable model coefficients over time.
 - Valid cross-validation across time periods.

Hyperparameter Tuning: Time-Series Cross-Validation

Why not standard cross-validation?

- Random k -fold CV assumes i.i.d. data.
- Time-series data exhibit autocorrelation \Rightarrow temporal dependence.
- Random shuffling lets the model “see the future” \Rightarrow data leakage and over-optimism.

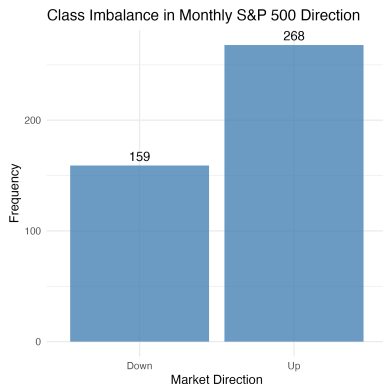
Our method: Rolling-window time-series CV

- Construct a hyperparameter grid:
 - Elastic Net: α and λ grid.
 - Random Forest: `mtry`, `min.node.size`, `max.depth`, `sample.fraction`.
- Use a chronological window structure:
 - Initial training window: 60 months (5 years).
 - Validation horizon: 12 months (1 year).
 - Fixed-length rolling window moving forward in time.
- For each fold:
 - Fit model only on past data and predict the next 12 months.
 - Record accuracy and AUC on the validation slice.

Result:

- Select the hyperparameters achieving the **highest mean AUC across all rolling folds**.
- Prevents look-ahead bias and mimics real-time forecasting.

Class Imbalance, Threshold Adjustment & AUC



Class imbalance in monthly S&P 500 direction

- Dependent variable UP_DOWN is imbalanced.
- A naive classifier that would always predict “Up” would achieve high accuracy.
- A simple threshold of 0.5 reduces the accuracy.
⇒ **Accuracy alone is misleading.**

Threshold adjustment (Youden’s J):

$$t^* = \arg \max_t \{ \text{Sensitivity}(t) + \text{Specificity}(t) - 1 \}$$

- We tune t^* on the **training set** using the ROC curve.
- Then apply this fixed threshold to the **test set** for true OoS evaluation.

AUC is robust here:

- ROC/AUC depends on the *ranking* of predicted probabilities, not on class proportions.
⇒ **AUC is invariant to class imbalance** and a suitable metric for our setting.

Model Intuition:

- We model the probability of an **Up** move using a logistic function:

$$P(Y_t = 1 \mid X_t) = \frac{1}{1 + e^{-(\beta_0 + X_t\beta)}}$$

- Coefficients β are maximum likelihood estimates under a regularization penalty to prevent overfitting and perform variable selection.

Elastic Net Regularization:

$$\min_{\beta} \left[-\ell(\beta) + \lambda \left((1 - \alpha) \frac{\|\beta\|_2^2}{2} + \alpha \|\beta\|_1 \right) \right]$$

- $\ell(\beta)$: log-likelihood of the logistic model.
- λ : overall penalty strength controlling coefficient shrinkage.
- α : mixes the two types of regularization:
 - Ridge (L2): smooth shrinkage.
 - Lasso (L1): sets some coefficients exactly to zero.

- Grid search over $\alpha \in [0, 1]$ in steps of 0.05.
- For each α , internal cross-validation via `cv.glmnet()` selects λ .
- For each fold:
 - ① Train model on training window.
 - ② Predict on validation window.
 - ③ Record Accuracy and AUC.
- The optimal (α, λ) combination is chosen by highest mean AUC.
- Final model is refitted on full training data and evaluated out-of-sample.

3. Methodology: Model Assessment Strategy

The most critical methodological slide

The Problem: Time-Series Data

Standard K -fold CV shuffles data randomly, “peeking into the future” and violating temporal order. This leads to overly optimistic results.

3. Methodology: Model Assessment Strategy

The most critical methodological slide

The Problem: Time-Series Data

Standard K -fold CV shuffles data randomly, “peeking into the future” and violating temporal order. This leads to overly optimistic results.

Our Solution: Two-Level Chronological Split

- **Level 1: Train/Test Split (for Assessment)**
 - Chronological 70/30 split.
 - **Train:** 1990–2014 (used for tuning).
 - **Test:** 2015–2025 (used once for final evaluation).

3. Methodology: Model Assessment Strategy

The most critical methodological slide

The Problem: Time-Series Data

Standard K -fold CV shuffles data randomly, “peeking into the future” and violating temporal order. This leads to overly optimistic results.

Our Solution: Two-Level Chronological Split

- **Level 1: Train/Test Split (for Assessment)**
 - Chronological 70/30 split.
 - **Train:** 1990–2014 (used for tuning).
 - **Test:** 2015–2025 (used once for final evaluation).
- **Level 2: Rolling-Window CV (for Tuning)**
 - Within the 70% training set, perform rolling-window validation.
 - Simulates real-world use: train on past, predict the future.