Predicting S&P 500 Direction with Ensemble Methods

Christian Weißmeier   Farkas Tallos

Statistical and Machine Learning (2025/26)

November 10, 2025

Agenda

1. Introduction: Task & Motivation

- **Task:** We selected a binary classification task.
- **Application Context:** Predict the monthly direction (*Up* or *Down*) of the S&P 500 index.
- **Motivation:**
  - This is a classic, challenging problem in financial econometrics.
  - We want to determine if publicly available data (market, macro, sentiment) contains a predictive signal for market direction.
  - This project allows us to apply and compare the course's key supervised learning methods to a complex, real-world time-series problem.

A Rich, High-Dimensional Time-Series Dataset

We gathered a dataset spanning over 70 years (1950 - Present) to ensure our models are robust across multiple economic regimes (e.g., high inflation, recessions).

Data Sources:

- **Market (Yahoo):** S&P 500 Price/Volume.
- **Macro (FRED):** CPI, Fed Funds Rate, NBER Recession.
- **Sentiment (FRB):** Daily News Sentiment Index (DNSI).
- **Volatility (Yahoo):** VIX Index.

Key Pre-processing Steps:

- All data aggregated to a monthly frequency.
- Predictors are **lagged** to prevent lookahead bias.
- **Target (Y):** `UP_DOWN` (factor: "Up", "Down").

1. Final Features

Market, Macro, Sentiment, and Interactions

Our final model dataset includes **16 predictors**.

- **Market Lags (5):** `lag1_return`, `lag2_return`, ..., `lag5_return`
- **Volume (1):** `volume_change_lag`
- **Macro (3):** `CPI_lag`, `FedFundsRate_lag`, `NBER_lag` (Recession binary)
- **Volatility (1):** `VIX_change_lag`
- **Sentiment (1):** `DNSI_change_lag`
- **Interactions (4):** We engineered interaction terms to capture more complex relationships (e.g., `DNSI_VIX_lag`, `VIX_CPI_lag`).

1. Final Features

Market, Macro, Sentiment, and Interactions

Our final model dataset includes **16 predictors**.

- **Market Lags (5):** `lag1_return`, `lag2_return`, ..., `lag5_return`
- **Volume (1):** `volume_change_lag`
- **Macro (3):** `CPI_lag`, `FedFundsRate_lag`, `NBER_lag` (Recession binary)
- **Volatility (1):** `VIX_change_lag`
- **Sentiment (1):** `DNSI_change_lag`
- **Interactions (4):** We engineered interaction terms to capture more complex relationships (e.g., `DNSI_VIX_lag`, `VIX_CPI_lag`).

Dataset Size

After merging and lagging, our final dataset for modeling runs from **Jan 1990 to Oct 2025**, providing **427 monthly observations**.

2. Exploratory Data Analysis (EDA)

Non-linearity is Likely

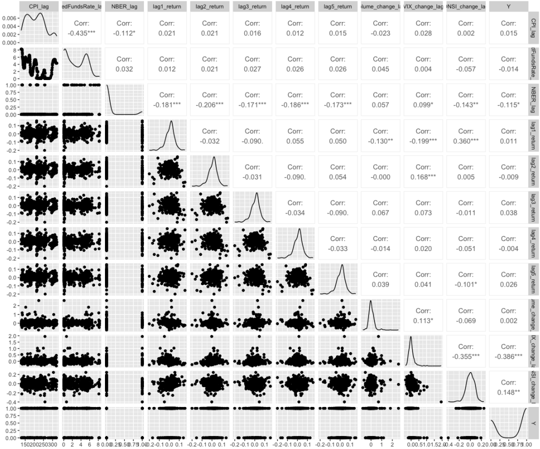


Figure: EDA: Predictor Relationships by Market Direction (Subset)

- The `ggpairs` plot shows no simple, linear "silver bullet" predictor. The distributions (histograms) for "Up" and "Down" months overlap significantly.
- This suggests that simple linear models may struggle and that predictive power, if it exists, likely comes from **non-linear relationships** or **interactions** between features.
- We also observed high correlation between lagged returns (e.g., `lag1_return`, `lag2_return`), which motivates the use of models that can handle collinearity.

3. Methodology: Model Selection

Comparing Linear, Bagging, and Boosting Methods

We selected three distinct, powerful methods covered in the course, as required:

1. Elastic Net (Regularized GLM)

`glmnet`

- A regularized logistic regression.
- Combines L_1 (Lasso) and L_2 (Ridge) penalties.
- **Why:** Excellent for variable selection (Lasso) and handling our correlated predictors (Ridge).

2. Random Forest (Bagging)

`ranger`

- Ensembles many de-correlated decision trees.
- A substantial modification of bagging.
- **Why:** Very robust, captures non-linearities, and is not prone to overfitting.

3. Gradient Boosting (GBM)

`gbm`

- Sequentially builds "weak" trees that correct prior errors.
- **Why:** Often provides top-tier accuracy and models complex interactions.

3. Methodology: Model Assessment Strategy

This is the most critical methodological slide

The Problem: Time-Series Data

We cannot use standard K-fold cross-validation. It shuffles data randomly, "peeking into the future" and violating the temporal order. This would lead to invalid, overly optimistic results.

3. Methodology: Model Assessment Strategy

This is the most critical methodological slide

The Problem: Time-Series Data

We cannot use standard K-fold cross-validation. It shuffles data randomly, "peeking into the future" and violating the temporal order. This would lead to invalid, overly optimistic results.

Our Solution: A Two-Level Chronological Split

- **Level 1: Train/Test Split (for Assessment)**
  - We split our 418 observations **chronologically** (70/30).
  - **Training Set (n=292):** 1990-2014. Used for all model tuning.
  - **Test Set (n=126):** 2015-2025. Held out completely. Used only once at the end for final, unbiased model assessment.

3. Methodology: Model Assessment Strategy

This is the most critical methodological slide

The Problem: Time-Series Data

We cannot use standard K-fold cross-validation. It shuffles data randomly, "peeking into the future" and violating the temporal order. This would lead to invalid, overly optimistic results.

Our Solution: A Two-Level Chronological Split

- **Level 1: Train/Test Split (for Assessment)**
  - We split our 418 observations **chronologically** (70/30).
  - **Training Set (n=292):** 1990-2014. Used for all model tuning.
  - **Test Set (n=126):** 2015-2025. Held out completely. Used only once at the end for final, unbiased model assessment.
- **Level 2: Rolling-Window CV (for Tuning)**
  - To tune hyperparameters (e.g., λ, `mtry`), we perform a **rolling-window validation** inside the 70% training set.
  - This simulates real-world use: we train on past data to predict the immediate future.
  - *(See Appendix for implementation code)*

4. Results: Hyperparameter Tuning Elastic Net

Best Parameters from Rolling-Window CV (Optimizing Log-Loss)

1. Tuned Elastic Net

Table: Top 5 Elastic Net tuning results (lowest log-loss)

alpha	lambda	mean_LogLoss	mean_AUC
1.00	0.0215	0.584	0.700
0.50	0.0464	0.584	0.735
0.25	0.1000	0.586	0.730
0.75	0.0464	0.587	0.714
0.25	0.0464	0.589	0.696

2. Tuned Random Forest

Table: Top 5 Random Forest tuning results (lowest log-loss)

mtry	min.node.size	sample.fraction	mean_LogLoss	mean_AUC
2	1	0.8	0.633	0.664
2	5	0.6	0.633	0.674
2	10	0.6	0.637	0.691
2	10	0.8	0.642	0.693
2	1	0.6	0.644	0.657

4. Results: Hyperparameter Tuning GBM

Best Parameters from Rolling-Window CV (Optimizing Log-Loss)

3. Tuned Gradient Boosting (GBM)

Table: Top 5 GBM tuning results (lowest log-loss)

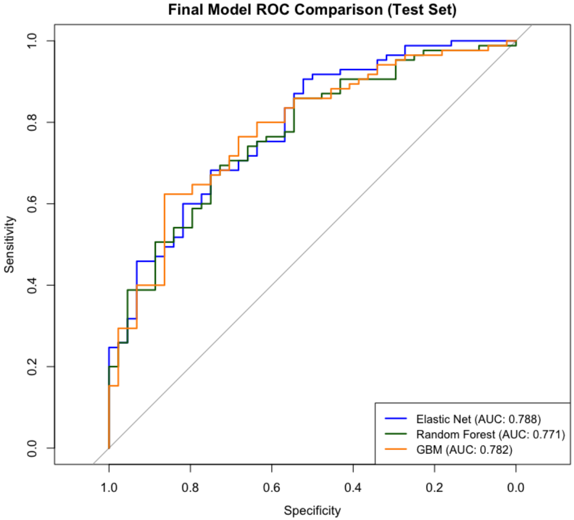
n.trees	interaction.depth	shrinkage	mean_LogLoss	mean_AUC
300	2	0.01	0.612	0.698
200	2	0.01	0.613	0.707
300	1	0.01	0.613	0.685
200	3	0.01	0.614	0.708
200	1	0.01	0.615	0.711

Table: Final Model Performance on Hold-Out Test Set

Model	Test_AUC	Test_Accuracy	Test_LogLoss
Elastic Net (Tuned)	0.7880	0.7670	0.5143
Gradient Boosting (Tuned)	0.7820	0.7210	0.5381
Random Forest (Tuned)	0.7710	0.7130	0.5518

4. Results: Final Model Assessment 2/2

Comparing Performance on the Hold-Out Test Set (2015-2025)



4. Results: Insights from Best Model

Coefficients from the Tuned Elastic Net

The **Elastic Net** was our best-performing model (Test AUC: 0.788). As a regularized GLM, its coefficients show which features were most important.

Our tuned model used $\alpha = 1.0$ (Lasso), which performed variable selection, setting 10 of 15 predictors to zero.

Final Model Coefficients (1/2)

Predictor	Coefficient
(Intercept)	0.5613
CPI_lag	.
FedFundsRate_lag	.
NBER_lag	-0.5159
lag1_return	-0.5606
lag2_return	.
lag3_return	1.4206
lag4_return	.

Final Model Coefficients (2/2)

Predictor	Coefficient
lag5_return	1.3327
volume_change_lag	.
VIX_change_lag	-5.5891
DNSI_change_lag	.
DNSI_VIX_lag	.
DNSI_FedFunds_lag	.
VIX_CPI_lag	.
DNSI_NBER_lag	.

- **Correctness of Results:**
  - We successfully implemented a **time-series-aware** validation pipeline to select and assess 3 models from the course.
  - The rolling-window tuning was essential for correctly handling the data's temporal structure and avoiding lookahead bias.
- **Final Performance:**
  - The **Tuned Elastic Net** was the best model, achieving a Test AUC of **0.788**.
  - This performance is strong and clearly better than random guessing (AUC $= 0.5$), suggesting a predictive signal exists.
- **Key Insights:**
  - The model selected only 5 predictors.
  - The most important predictor was `VIX_change_lag` with a large negative coefficient, indicating that a recent spike in volatility is a strong predictor of a 'Down' month.
  - Recent momentum is complex: `lag1_return` is negative, but `lag3_return` and `lag5_return` are positive.
  - Being in a recession (`NBER_lag`) is a strong negative predictor, as expected.

Thank You

Questions?

Appendix: Model Coefficients & Tuning Code

Tuned Elastic Net Coefficients

```
16 x 1 sparse Matrix of class "dgCMatrix"
(Intercept)          0.5613457
CPI_lag              .
FedFundsRate_lag     .
NBER_lag            -0.5158768
lag1_return         -0.5606195
lag2_return          .
lag3_return          1.4206226
lag4_return          .
lag5_return          1.3326377
volume_change_lag    .
VIX_change_lag      -5.5890975
DNSI_change_lag      .
DNSI_VIX_lag         .
DNSI_FedFunds_lag    .
VIX_CPI_lag          .
DNSI_NBER_lag        .
```

Tuned GBM Feature Importance

```
                                var    rel.inf
VIX_change_lag       VIX_change_lag 29.1156079
VIX_CPI_lag             VIX_CPI_lag 14.6891879
lag3_return             lag3_return  8.8438007
lag1_return             lag1_return  8.6205320
CPI_lag                     CPI_lag  7.4618803
lag2_return             lag2_return  7.1084220
lag5_return             lag5_return  6.7386632
FedFundsRate_lag   FedFundsRate_lag  4.6935959
DNSI_change_lag     DNSI_change_lag  3.6435704
lag4_return             lag4_return  2.9619767
volume_change_lag volume_change_lag  1.6682523
DNSI_VIX_lag           DNSI_VIX_lag  1.4948200
DNSI_FedFunds_lag DNSI_FedFunds_lag  1.2914756
NBER_lag                   NBER_lag  1.0454376
DNSI_NBER_lag         DNSI_NBER_lag  0.6227775
```

Appendix: Rolling-Window Tuning Loop (Random Forest)

RF Tuning Loop (Part 1)

```
# --- Rolling-window tuning ---
n_train <- nrow(train_df)
window_size <- floor(0.7 * n_train)
horizon <- 12
results <- data.frame()

set.seed(123)
for (i in seq(window_size, n_train - horizon,
              by = horizon)) {
  train_window <- train_df[1:i, ]
  test_window  <- train_df[(i + 1):(i + horizon), ]

  if (nrow(test_window) == 0 ||
      length(unique(test_window$Y)) < 2) next

  for (j in 1:nrow(param_grid)) {
    p <- param_grid[j, ]

    rf_model <- ranger(
      Y ~ .,
      data = train_window,
      num.trees = 500,
      mtry = p$mtry,
      min.node.size = p$min.node.size,
      sample.fraction = p$sample.fraction,
    # (Continued in next column...)
```

RF Tuning Loop (Part 2)

```
    # (...Continued from last column)
      probability = TRUE,
      seed = 123
    )
    preds <- predict(rf_model,
      data = test_window)$predictions[, "1"]

    y_true <- as.numeric(as.character(
      test_window$Y
    ))

    logloss <- -mean(
      y_true * log(preds + eps) +
      (1 - y_true) * log(1 - preds + eps)
    )

    results <- rbind(results, data.frame(
        mtry = p$mtry,
        min.node.size = p$min.node.size,
        sample.fraction = p$sample.fraction,
        LogLoss = logloss
      ))
  } # end inner loop
} # end outer loop
```