## Week 9 — Select the Winning Model

This week we completed the model training layer so the rest of the pipeline can move directly into validation, selection, testing, and reporting. We trained a coherent suite of nine baseline regressors on the same feature matrix and target with fixed parameters and a single random seed. The models are Linear Regression, Ridge, Lasso, Elastic Net, Decision Tree, Random Forest, Extra Trees, Gradient Boosting, and Histogram Gradient Boosting. Together they cover a practical spectrum of capacity from linear baselines to tree ensembles and boosting methods, which prepares a meaningful comparison for the next stage.

Before fitting anything, we enforced strict alignment between the feature matrix and the target. We verified that the two objects refer to the exact same rows, removed rows with missing values in either features or target, and confirmed equal sample counts. This small but crucial step ensures that every model is trained on identical data, which prevents hidden discrepancies when we later compare validation errors across models. It also guarantees that the winning model can be reproduced exactly when we score the test set.

All models were trained with fixed parameters by design. Our goal for this week is to deliver stable, comparable baselines rather than chase performance through hyperparameter search. Fixing parameters and the random seed creates a level playing field. Any improvements observed later during tuning will be attributable to the tuning process rather than accidental variation in training.

We saved each fitted estimator as a Joblib artifact under a single folder for week nine and wrote a compact registry in JSON that records the file path, model class, module path, fixed parameters, and seed. This registry acts as a single source of truth for downstream steps. A teammate can iterate over it to load each artifact, generate predictions on the validation set, and compute errors. The registry also stabilizes collaboration because it removes guesswork about where models are stored and how they were configured.

These choices directly support the written deliverables. The professor asks for validation errors for nine models, a final winning model based on the validation set, discussion of the

bias variance tradeoff, test metrics for the selected model, and a table that shows how metrics change across training, validation, and test splits. Our work provides clean inputs for each of these items. Since all models were trained on the same aligned training split, the validation teammate can compute a fair error table without worrying about mismatched sample counts. Because the nine models form a natural sequence from simple to complex, the bias variance chart can place model names on the x axis and error on the y axis to show how performance evolves with capacity. The registry guarantees that the winning model can be reloaded to score the test set and produce final metrics with full reproducibility. It also makes it straightforward to assemble a compact table that lists metrics for the winning model across the three splits.

Our approach also prepares for packaging and basic monitoring. Housing all artifacts in a predictable folder with consistent names creates a stable target for a deployment script that picks up the winning model. The registry supplies the metadata needed to verify that the model being deployed matches the one that was validated and tested. If we later add feature schema checks or drift monitors, the same structure can serve as the anchor for those components.

After training was completed, we evaluated all nine models on the validation dataset using the same features and target structure. This step allowed us to measure their relative performance under identical conditions and identify the most stable generalizer. The comparison revealed a clear trend in model behavior. Among all models, the Lasso Regression achieved the lowest validation RMSE (≈ 0.0819) and MAE (≈ 0.059), performing slightly better than Elastic Net and Ridge while maintaining lower variance than tree-based models. The results show that simpler linear regularization methods provided better generalization on unseen data than more complex ensemble approaches. This outcome is common when the feature space is moderately correlated and the dataset is not extremely large.

The bias–variance trade-off across the nine models further confirmed this observation. When models were ordered from the simplest (Lasso) to the most complex (Decision Tree and

Boosting), validation error gradually increased rather than decreased. This pattern indicates that additional complexity did not reduce bias but instead introduced unnecessary variance. In other words, the dataset does not contain strong nonlinear relationships that require deep trees or iterative boosting to capture. Instead, the signal is largely linear, and adding capacity only amplifies noise. This is visible in the bias-variance chart, where the RMSE curve rises steadily toward the right side. The flatter lower region on the left shows that linear models are sufficient to explain the available structure, and that heavier models begin to overfit the training noise.

Based on these results, Lasso Regression was selected as the final winning model. It balances predictive accuracy and simplicity, producing consistent results across folds and runs. Because Lasso includes an L1 penalty, it also performs implicit feature selection by shrinking weak coefficients toward zero. This improves interpretability and guards against instability in multicollinear data. The compactness of the model makes it easy to reproduce, deploy, and monitor in later stages without extensive retraining or hyperparameter maintenance.

We then evaluated the selected Lasso model on the independent test dataset to measure true out-of-sample performance. The test results were: MSE = 0.0067, RMSE = 0.0819, MAE = 0.0592, MAPE ≈ 1.71 %, $R^2$ = –0.0145, and Adjusted $R^2$ = –0.0456. These numbers show that the model's prediction errors remain small in absolute magnitude, but its $R^2$ value is close to zero, indicating limited explanatory power. In practical terms, this means that while the model can approximate the average direction of TSR changes, it struggles to capture the full volatility of monthly returns. This is reasonable because TSR contains considerable random market movement that is difficult to model purely from macro-level variables. The residual distribution plot supports this interpretation: the residuals follow a near-normal curve centered around zero, suggesting that the model is unbiased but slightly underfits the tails.

The "Actual vs Predicted" scatter plot also demonstrates that most predictions cluster tightly around the mean, with few extreme deviations. This reflects the conservative nature of the Lasso penalty—it prefers smaller coefficients and smoother responses. The overall pattern

implies that our current feature set explains part of the variation in TSR but not all of it, and that additional nonlinear or interaction features may be required to improve $R^2$ in the future. Still, the model's low error and stable distribution make it a reliable baseline for subsequent enhancement.

When we compare performance across the three splits—training, validation, and test—the RMSE remains consistently near 0.08 in all cases. This stability confirms that the model generalizes well and does not rely on spurious patterns. The small difference between training and validation error suggests low variance, while the modest $R^2$ reflects high bias, which is a known trade-off for regularized linear models. From a bias-variance perspective, this position is acceptable for our current stage because we prioritize consistency and interpretability over short-term predictive gain.

In summary, the Lasso model emerges as the most balanced option in this week's experiment. It achieves low and stable prediction errors, avoids overfitting, and provides interpretable coefficients that can be examined in later analysis. Although its $R^2$ score is modest, the model sets a clear and reproducible benchmark for evaluating more advanced architectures in the following phases. Future work can focus on expanding the feature set, exploring interaction terms, or integrating non-linear kernels while preserving the same disciplined evaluation framework established this week.