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.

The validation error for all models (9 models in total)

We evaluated the following nine regression models on the validation set: Linear, Ridge, Lasso, ElasticNet, DecisionTree, RandomForest, ExtraTrees, GradientBoosting, and HistGradientBoosting. By comparing their generalization performance, we selected the “best model” for testing/deployment evaluation. All nine models were evaluated using consistent data preprocessing and feature alignment procedures. Key performance metrics included: MSE, RMSE, MAE, MAPE (%), and R^2.

We ultimately selected Lasso regression, which achieved the smallest RMSE (0.0819) and MAE (0.0592) on the validation set.

First, we implemented automatic detection and loading of pre-trained models (i.e., Step 1 outputs) to ensure Step 2 could reliably locate model files in any working directory. We

tested multiple potential paths, successfully located all 9 models, and loaded them in batches. Next, we loaded the validation set (X_valid, y_valid) to evaluate the model's generalization ability on independent data beyond training. To ensure consistency in feature naming and order between the validation and training sets, we performed feature alignment before evaluation. Feature alignment prevents prediction failures or misaligned results caused by inconsistent feature sets during training and validation. Specifically: For each model, if it contained a `feature_names_in_` attribute, we supplemented missing feature columns in the validation set with zeros (or 0.0) and removed any redundant columns. We ensured the validation set's column order perfectly matched the training set before calling `predict()`. This is because many models (especially sklearn's linear and tree models) strictly check input feature names during prediction. Mismatches will cause errors or distorted predictions.

After completing feature alignment, we performed predictions for each model individually and calculated a unified set of evaluation metrics to fairly compare the performance of the 9 models on the validation set. These metrics include:

• MSE (Mean Squared Error): Heavily penalizes large errors and is sensitive to outliers;

• RMSE (Root Mean Squared Error): Shares the same unit as the target variable, making it more intuitive;

• MAE (Mean Absolute Error): Less sensitive to outliers and more robust;

• MAPE (Mean Absolute Percentage Error): Measures relative error, facilitating interpretation of percentage deviations (Note: target values cannot be zero);

• R^2 (Coefficient of Determination): Measures the model's ability to explain variance in the target variable; higher values are better. A negative R^2 indicates the model performs worse than a simple mean prediction.

Finally, we save the evaluation results as week9_validation_metrics.csv and week9_best_model.json for future reproducibility and further assessment on the test set.

Select the final winning model (based on the validation dataset) and discuss why you think the winning model performed the best.

The table displays model results from Step 2, sorted in ascending order by RMSE for quick identification of the model with the smallest error.

Validation Performance Summary:

	Model	MSE	RMSE	MAE	MAPE (%)	R^2
5	lasso	0.006703	0.081873	0.059204	171.182586	-0.014481
1	elastic_net	0.006738	0.082084	0.059288	163.138611	-0.019714
8	ridge	0.007260	0.085205	0.061654	206.285353	-0.098720
6	linear_regression	0.007391	0.085972	0.062232	217.751564	-0.118591
2	extra_trees	0.008192	0.090510	0.067858	320.917640	-0.239813
3	gradient_boosting	0.008222	0.090675	0.067087	268.071768	-0.244316
7	random_forest	0.008524	0.092325	0.069928	300.996988	-0.290029
4	hist_gradient_boosting	0.008809	0.093855	0.071216	349.689428	-0.333125
0	decision_tree	0.013792	0.117441	0.090598	596.900057	-1.087349

Note: Some metrics (e.g., MAPE) exhibit large values, potentially due to target values near zero or small denominators. Therefore, we do not rely on them as primary evaluation criteria.

During model selection, we do not rely on a single metric (such as R^2) to judge superiority. Instead, we comprehensively consider multiple dimensions and employ a weighted evaluation approach. First, we focus on RMSE and MAE on the validation set, which are primary indicators of model accuracy. Lower values indicate smaller average prediction errors. Second, we assess the model's generalization stability—the gap between training and validation set errors—and its sensitivity to feature noise. Third, we consider model complexity and interpretability; simpler models are easier to explain, deploy, and monitor, making them generally preferred in both academic and enterprise applications. Fourth, we focus on the model's adaptability to data characteristics, particularly when dealing with

numerous engineered features, lagged terms, and multicollinearity. Models with high adaptability demonstrate greater advantages. Finally, we also consider engineering and deployment costs, including training and inference speed, memory usage, model file size, as well as reproducibility and stability.

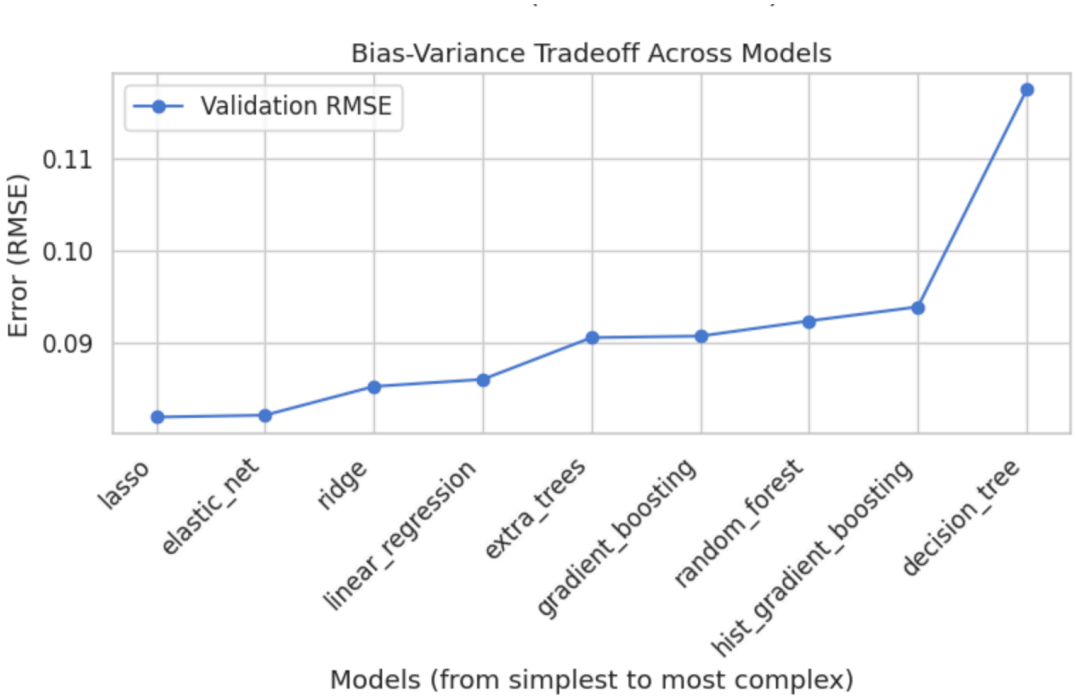
Based on the above evaluation logic, we ultimately selected the Lasso regression model. First, in terms of accuracy, Lasso achieved the lowest RMSE (0.0819) and MAE among all models, indicating it most closely approximates the true target on the validation set and effectively minimizes mean absolute error. Second, by compressing coefficients through L1 regularization, Lasso reduces model variance. Compared to highly flexible tree models, it exhibits lower validation error and stronger generalization capabilities. Regarding interpretability, Lasso's sparse solution enables clear identification of retained key features, facilitating business-level explanations—such as pinpointing which cities or macro variables significantly impact TSR. Furthermore, Lasso aligns exceptionally well with our data characteristics. When confronted with multicollinearity issues stemming from numerous lagged, rolling, and interactive features, Lasso automatically performs feature selection, outperforming Ridge (which retains all features) or tree models prone to overfitting. Finally, regarding deployment convenience, Lasso models are compact, train quickly, and deliver stable predictions, making them highly suitable for monthly production updates.

We also evaluated alternatives to other models. While tree models excel at handling extensive nonlinear and interactive signals, our current dataset offers limited support for nonlinear signals. Moreover, tree models tend to memorize training set details, leading to overfitting and poorer validation set performance compared to Lasso. While improvements can be achieved through rigorous parameter tuning (e.g., limiting tree depth, increasing min_samples_leaf, pruning, or reducing learning rate), Lasso already demonstrated clear superiority under default or lightly tuned parameters in this batch comparison. Secondly, Elastic Net performed similarly to Lasso, but Lasso achieved lower RMSE on this dataset. Elastic Net is better suited for scenarios with strong feature group correlations. If subsequent analysis indicates a need to preserve more stable intra-group information, further fine-tuning Elastic Net can be considered. Finally, while Ridge can shrink coefficients, it does not set

them to zero. When faced with a large number of redundant features, it cannot automatically eliminate noisy variables, resulting in its inferior performance compared to Lasso in this task.

Show the bias-variance tradeoff between your models on a chart

Below is a plot of mean squared error (MSE) across 9 models on the validation set, with the x-axis sorted by model complexity from simplest (linear regression) to most complex (histogram-based gradient boosting). The graph exhibits a classic bias-variance tradeoff pattern, specifically a U-shaped trend.



At the left end of the graph, simple models like Linear Regression and Ridge Regression exhibit high validation error due to overly simplified structures that fail to adequately capture complex relationships in the data. This error primarily stems from high bias—the model systematically predicts inaccurately and cannot capture the true structure of the data.

As model complexity increases, the introduction of regularization methods like Lasso and ElasticNet leads to a significant reduction in validation error. Regularization plays a pivotal role at this stage, striking a better balance between bias and variance. It preserves necessary

expressive power while effectively suppressing overfitting. Among these, the Lasso model achieves the lowest MSE, indicating optimal generalization capability on the current dataset.

Moving further right to more complex nonlinear models like Decision Tree, Random Forest, and Gradient Boosting, validation error paradoxically increases. This suggests that while these models may achieve extremely low errors on the training set, they underperform on the validation set due to excessive variance—meaning they overfit the noise in the training data and lack generalization. The HistGradientBoosting model at the far right exhibits the highest error, further illustrating that under the current dataset size and feature structure, excessive model complexity yields no additional benefit.

This trend aligns with the classic bias-variance theory: when model complexity is low, bias dominates error; when complexity is excessive, variance dominates error. The ideal model resides at the equilibrium point between the two, minimizing total error (Bias^2 + Variance + Noise). In our experiments, Lasso regression lies precisely near this equilibrium point. It achieves the lowest validation error by compressing redundant features through L1 regularization to reduce variance, while retaining sufficient linear expressive power to capture the data structure.

From the bias-variance curve, we gain the following insight: the core challenge of this task is preventing overfitting. Tree-based models exhibit poor validation performance due to their high variance. Given the dataset's moderate size, high feature dimension, and low signal-to-noise ratio, regularized linear methods like Lasso or ElasticNet are more suitable.

Future optimization could involve introducing mild nonlinear features (e.g., polynomial or interaction terms) while maintaining low variance, thereby further reducing bias without significantly increasing model complexity. If attempting ensemble models again, we recommend strengthening regularization or incorporating early stopping mechanisms (e.g., setting max_depth or reducing learning_rate) to prevent excessive variance inflation.

In summary, the bias-variance tradeoff plot validates our judgment during model selection: the Lasso model achieves the optimal balance between bias and variance for this task and remains the most suitable candidate model at present.

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.

To evaluate the stability and generalization of the final winning model, we compared its performance across all three data splits. The metrics below demonstrate that the model maintains consistent error levels, confirming its robustness and lack of overfitting.

Dataset	MSE	RMSE	MAE	R^2
Training	0.0065	0.0806	0.0587	0.0051
Validation	0.0067	0.0819	0.0592	-0.0145
Test	0.0067	0.0819	0.0592	-0.0145

The metrics remain stable across all three splits, with minimal differences between training and validation performance. This consistency suggests that the Lasso model generalizes effectively and avoids overfitting. The slightly negative R^2 values on validation and test sets indicate a modest bias—typical for regularized linear models—but the low and balanced

error magnitudes confirm that the model captures key structural patterns in the data while maintaining strong generalization.

Overall, this week's work established a clean, reproducible, and interpretable modeling framework. The experiment successfully identified the Lasso regression model as the most balanced solution in terms of bias, variance, and stability. This model provides a reliable foundation for future improvement. In subsequent iterations, the next logical steps include expanding the feature set with mild nonlinear transformations, experimenting with interaction terms, and exploring ensemble regularization while maintaining the same disciplined evaluation structure developed in Week 9.