Week 10 — Data Centric AI

We kept the modeling family fixed and focused on the data. The pipeline loads the enhanced full table produced earlier in the project, preserves the original time-based partition identities that were established at the beginning, and reconstructs three refreshed views that all subsequent steps consume: a cleaned training view, a validation view locked to the same calendar window used throughout the project, and an untouched test window. We enforced a single target column across views, restricted features to aligned numerics, and wrote out checkpoints and artifacts so every plot and metric is reproducible from the notebook. By holding the hypothesis class constant (the L1-regularized linear model we selected last week), any performance movement this week is attributable to data interventions rather than to algorithm changes. The three refreshed files—training, validation, and test—are saved explicitly so that readers can re-run our evaluation on identical windows.

1) Three new data improvements we implemented

First, we eliminated row-level duplication before any split logic. The enhanced table contained exact and near-exact duplicates inherited from earlier joins; such duplication implicitly up-weights specific regimes and biases coefficients toward over-represented slices. We surfaced counts, dropped duplicates deterministically, and confirmed that the training/validation/test identities still map to the same calendar periods after projecting the cleaned data back onto the original time split index. Removing duplication prevents the model from collecting “free” low-error repeats and yields more honest residuals.

Second, we standardized missing-value handling for all numeric features with a single, robust policy: median imputation. Prior work tolerated NaNs through library defaults and occasional per-column fixes, which created inconsistency across features and introduced avoidable variance. Median imputation is stable under skew and outliers and avoids creating feature-specific behavior that the optimizer can overfit. We printed per-column missingness before and after imputation to document how much mass each feature recovers and to ensure that no column silently collapses to a constant.

Third, we enforced leakage-safe temporal reconstruction. Cleaning can change row membership and order; if we were to split after cleaning, calendar drift and subtle label leakage could infect validation or test. Instead, we preserved the original train/validation/test identities created at the start of the project and projected the cleaned table onto those identities. This keeps the holdout windows identical to prior weeks and ensures that week-over-week differences reflect data hygiene rather than moving time boundaries.

Two small engineering refinements sit under these three changes. We added directory creation and file-existence assertions so the run is error-free end-to-end, and we made target-column discovery explicit (pattern search with validation) to avoid accidental shifts if column names vary across sources. These do not change the statistics, but they eliminate a class of operational failures and make the notebook easier for a fresh reader to execute without surprises.

2) Error analysis on the training split and the kinds of problems it revealed

We re-examined model errors on the cleaned training data using three standard diagnostics saved in the outputs directory: actual-versus-predicted scatter with a 45-degree reference line, residual distribution (histogram/density), and residuals-versus-predicted scatter with a zero baseline. Compared with last week's visuals, three patterns emerged that map directly to the data work.

The actual-versus-predicted scatter lost the suspicious micro-clusters that previously hugged the 45-degree line. Those clusters were a tell-tale sign of duplicated or nearly duplicated rows granting the model artificially easy points. After duplicate removal, the cloud is more diffuse and the slope symmetry looks more realistic across the range of fitted values. Points at higher predicted values no longer collapse into tight stacks; instead they distribute around the diagonal in a way that reflects genuine variability rather than repetition.

The residual distribution is centered and smooth but slightly broader than before, which is what we expect when the model can no longer collect repeat wins. We traded artificially tight tails for an honest spread. Importantly, the histogram no longer exhibits sharp, isolated spikes, consistent with a uniform imputation policy that prevents a small subset of rows from receiving unusually favorable default values.

The residuals-versus-predicted plot shows reduced curvature and less funnel-shaped heteroscedasticity, but not a perfectly flat band. The remaining curvature is a bias signal rather than a hygiene signal: a strictly linear hypothesis with L1 shrinkage cannot fully capture relationships that bend with the level of the prediction. In short, the data is now clean enough that the main limitation we see is model capacity, not quirks or leakage.

3) Retraining the selected model on the updated training data and reporting split metrics

We kept the model family and general configuration from last week—Lasso regression with a small alpha and ample iterations—and re-fit on the cleaned Week 10 training view. Feature scope stayed numeric and column-aligned to the saved training matrix to guarantee that evaluation matrices match

model expectations. The artifact is serialized to disk so that evaluation and comparison steps always reference the exact same set of weights.

We evaluated the refit on training, validation, and test with a consistent metric block that prints and saves MSE, RMSE, MAE, MAPE, and R^2. The notebook's metrics table for this Week 10 refit is:

• training: MSE 0.8291, RMSE 0.9106, MAE 0.7050, MAPE 139.68%, R^2 0.0456

• validation: MSE 1.6312, RMSE 1.2772, MAE 1.0267, MAPE 114.50%, R^2 0.0089

• test: MSE 1.0944, RMSE 1.0461, MAE 0.8168, MAPE 267.38%, R^2 −0.0677

Two aspects deserve emphasis. First, the gap between training and validation is modest, indicating that regularization is functioning correctly and that the model is not memorizing the training view even after cleaning. Second, R^2 hovers near zero on validation and dips slightly negative on test. That pattern is consistent with a capacity-limited linear-sparse hypothesis facing a noisy target: after we removed duplication and standardized missingness, the task became more honest, and the model's ceiling became visible. The data-centric work stabilized behavior; it did not, by itself, add functional-form flexibility.

4) Comparing this week's model against last week's best on the updated validation window and selecting a final model

To make a fair comparison, we reused the updated validation dataset as the common yardstick and contrasted two candidates: the best model selected last week (same family and comparable configuration) and the new refit trained on the cleaned data. Because the calendar window is identical, any performance movement can be discussed as a data effect rather than a window effect.

On that updated validation set, last week's model remains the better performer. Its validation error is lower and its R^2 remains visibly positive, whereas this week's refit posts a higher RMSE with R^2 near zero. The explanation aligns with the error analysis: when duplicated rows and inconsistent missingness are removed, the validation window becomes a harder, more honest target for a linear-sparse model. The prior model benefited from quirks that we have now corrected; the refit cannot claim those gains.

Given the head-to-head, we select last week's model as the final model. The decision criterion is performance on the updated validation window, and the baseline wins on that yardstick. At the same time, we keep using the cleaned views for evaluation and diagnostics because they provide a more truthful read

on what the model can and cannot do. This separation—clean data for evaluation, prior weights for the deployed function—preserves reliability and maintains comparability across weeks.

5) Using the final model to report test-set performance and relating test to validation and training

We evaluated the selected model on the Week 10 test view to keep comparability with the rest of the outputs. The structure of the errors mirrors the validation behavior more closely than the training behavior: the magnitude of test RMSE is in the same neighborhood as validation RMSE, and the sign and size of test R^2 are consistent with the idea that a simple linear-sparse function struggles to dominate a horizontal baseline in that holdout period. The diagnostic plots retain the shapes discussed earlier—centered residuals, no pathological spikes, no micro-cluster artifacts—and do not show signs of overfitting to the training period.

In relative terms, training error is smaller than validation error, as expected for a properly regularized setup, and validation and test errors live close together. That alignment is the desirable regime for risk management: we are not seeing brittle validation wins that collapse on test, and we are not seeing training error that is implausibly tiny compared with what the model achieves out of sample. The similarity between validation and test also confirms that the original time split index is functioning as intended: the two holdouts represent comparable difficulty, and improvements on one translate to the other.

6) Insights from the training, validation, and test errors of the selected model

The first insight is that the dominant source of improvement opportunity has shifted from hygiene to functional form. Our interventions did exactly what they were designed to do: remove duplicated leverage, unify imputation, and preserve strict temporal identities so that all windows are comparable. The result is lower variance and more stable behavior across splits. However, because the refit's R^2 is near zero on validation and slightly negative on test, we can see the ceiling of a linear-sparse model against a noisy target on honest data. The error shape—especially the mild curvature in residuals versus predictions—indicates structure that a strictly linear function cannot capture without feature engineering or nonlinearity.

The second insight is that duplicates were masking uncertainty. The disappearance of tight micro-clusters around the 45-degree line in the actual-versus-predicted plot confirms that repeated rows previously granted artificially low squared errors in a few regimes. Their removal widened the residual distribution but made it more symmetric and better centered. That change makes the reported metrics

better calibrated for downstream use, including the construction of prediction intervals from residual quantiles.

The third insight concerns split drift. Validation and test windows are close in error magnitude but not identical in explanatory power; one can produce a barely positive R^2 while the other dips slightly negative even when RMSE is comparable. This pattern implies that the mean level and variance of the target move around from window to window in ways that a linear-sparse model cannot easily normalize. Monitoring baseline shifts and re-centering features or the target when appropriate remains important when we keep the model family simple.

The fourth insight is about interpretability under sparsity. Re-fitting on the cleaned training view still yields a sparse coefficient vector with a small set of large-magnitude drivers and many near-zero weights. That structure is valuable for explaining behavior and for diagnosing drift: when only a handful of features carry most of the weight, we can track their distributions across splits and verify whether their ranges and relationships remain stable. Even though the selected artifact comes from the prior week, the coefficient table from the refit helps pinpoint pressure points and complements the residual diagnostics.

The fifth insight is a bias–variance framing of what changed. Previously there was a mixture of variance (instability driven by inconsistent handling of NaNs and duplicated leverage) and bias (a linear-sparse form applied to relationships with curvature). This week reduced variance through consistent imputation and deduplication, which is why training and validation now sit closer together and why test aligns with validation. Bias remains because the function class was intentionally held constant. Seeing the two components cleanly separated is the purpose of a data-centric cycle with a fixed hypothesis family.

7) How the narrative ties back to our code path

We introduced three new data improvements implemented directly in the notebook before model training: duplicate removal, uniform median imputation for all numerics, and leakage-safe reconstruction that maps the cleaned table back to the original time split index. We performed training-set error analysis with actual-versus-predicted, residual distribution, and residuals-versus-predicted views, and we used the shapes of those plots to explain why duplicates and inconsistent missingness previously drove flattering metrics. We re-trained the same model family selected last week on the updated training dataset, saved the artifact, and reported performance on training, validation, and test with a unified metric block. We compared the new refit against last week's best on the updated validation window, determined that the prior model performs better on that common yardstick, and selected it as the final model. Using that final model, we reported test behavior and discussed how test error compares to validation and training,

highlighting that validation and test live close together while training stays lower due to regularization. We then articulated insights grounded in the split errors of the selected model—variance reduction from data hygiene, residual curvature indicating remaining bias, duplicated-row effects, split drift, and the interpretability afforded by sparsity—strictly as they appear in the notebook's outputs.