Week 5 — Engineer Features

After completing data cleaning, missing value handling, outlier correction, and variable transformations (logarithmic scaling, standardization, encoding) in Week 4, we now need to construct features useful for predicting t+1TSR. To avoid data leakage, we will only use information observable at time t. The purpose of feature engineering is both to enhance the model's predictive capability and to improve its interpretability regarding economic and company-specific drivers. Additionally, this week we will also perform data augmentation and dimensionality reduction.

1. Feature engineering

Feature engineering first requires constructing the target variable, TSR_next. For each company, we shift the next month's price and dividend to the current row in the time series, calculating according to the formula: $TSR_{t+1} = P_{t+1} - P_t + D_{t+1} / P_t$

The continuous target variable TS,R must explicitly represent “next month's” return. During construction, strict company-level grouping is maintained to prevent leakage of forward-looking information into current-period features. Using raw prices to compute TSR enhances interpretability, as percentage returns are more intuitive and easier to understand. After construction, trailing rows with uncomputable targets are removed to guarantee each training sample has a clear label. We successfully generated monthly TSR_next values from 2014 to 2025, retaining 763 observations.

We then generated 1-period, 3-period, and 6-period lags for several key variables, commonly including lags of TSR_next, cpi_yoy/cpi_mom, unemp/unemp_mom, ten_year/ten_year_mom, etc. Lagged features capture momentum (i.e., the influence of events over a period following their occurrence) and short-term memory. Financial time series often exhibit momentum or reversal effects. For instance, last month's returns or the average of the previous three months often partially explain the short-term trend in the next period. Historical inflation and interest rates may also influence rents and valuations during lagged transmission periods. Grouping by company and applying a shift ensures features reflect a company's own history rather than cross-company confounding information. Lagged values are saved under corresponding labels for easy verification.

Rolling window features compute past windows at the company level—such as 3-month or 6-month averages and standard deviations—applied to variables like TSR_next, cpi_mom, unemp, ten_year. Rolling averages smooth short-term noise to reveal recent trends, while rolling standard deviations capture data volatility, reflecting risk or uncertainty. For the real estate sector, these features distinguish

one-off events from persistent trends, helping our models learn stable signals in noisy, high-volatility environments.

Constructing interaction features involves creating cross-terms between company variables, such as cpi_mom × unemp, ten_year × dividend_ttm_log, and dividend_ttm_log / adj_price_log. In real economic relationships, variables often interact rather than being influenced by a single factor. Therefore, we need to observe how variables change in combination. Such interaction features better reveal conditional effects—for example, rising inflation may impact TSR differently in regions with declining unemployment, or higher interest rates may affect high-dividend companies more significantly. We believe constructing interaction features enhances model expressiveness and interpretability, revealing potential drivers of TSR across different economic environments.

The final step in feature engineering involves constructing time features. We extract the month from the date variable and apply one-hot encoding to the month, generating dummy variables month_1 through month_12. The significance of temporal features lies in the pronounced seasonality of the real estate market. Factors like rent adjustments, year-end tax cycles, or dividend periods all influence market dynamics. Monthly features enable the model to capture cyclical patterns, improving short-term forecasting performance during seasonal months. We use one-hot encoding instead of numerical months to avoid introducing artificial sequential relationships.

2. Data Augmentation

After preparing a clean dataset of monthly observations for six U.S. real estate firms, the sample size was still relatively limited. Each company has around a decade of history, which results in fewer than 800 usable firm-month observations after aligning prices, dividends, and macroeconomic variables. With such a modest dataset, predictive models can easily overfit by memorizing specific company-month patterns rather than learning general economic relationships.

The goal of data augmentation in this project is to artificially expand the dataset in ways that improve model robustness while preserving economic interpretability. Unlike feature engineering (which creates new predictors) or dimensionality reduction (which compresses predictors), augmentation works by creating new rows of training data from existing information. This helps the model see more diverse but still realistic inputs and learn smoother decision boundaries.

As a first step, we constructed the prediction target: the one-month-ahead total shareholder return (TSR). For each firm and month, TSR was defined as Price return plus dividend return. Grouping by company ensured that forward-looking information from other firms could not leak into the calculation. After dropping the last month of each firm's series where the forward value is missing, the base modeling dataset contained 763 observations with both predictors and target values.

Then we apply two complementary augmentation strategies.

The first strategy generated slightly perturbed versions of each observation by adding small Gaussian noise to the numeric predictors. In this strategy, noise scale was set relative to the interquartile range of each variable, which provides a robust measure of spread less sensitive to outliers and each variable was jittered within realistic bounds by clipping values between its 1st and 99th percentiles. Two noisy replicas of the dataset were created. Together with the original, this produced 2,289 rows. This step mimics measurement errors or small reporting differences that exist in real economic data, forcing the model to learn patterns that are stable under minor perturbations.

The second strategy created synthetic samples by linearly combining pairs of observations from the same company. In this strategy, for two firm-month rows, a random weight λ was drawn from a Beta distribution and we restricted mixup pairs to the same company to avoid introducing unrealistic cross-firm hybrids. We generated 381 synthetic rows via mixup, equivalent to about 50% of the base dataset size. This step smooths the relationship between inputs and outputs, reducing sharp jumps that may cause overfitting and encouraging the model to learn continuous functional forms.

The final augmented dataset combined the original rows, the noise-jittered copies, and the mixup rows.

Original observations: 763

Noise jittered copies: 1,526

Mixup samples: 381

Total after augmentation: 2,670 rows across 15 columns

The augmentation therefore more than tripled the effective training sample size without altering the statistical meaning of the variables. As we can see that the dataset increased from 763 to 2,670 rows, giving models more training material. Jittered copies make the model less sensitive to small changes in predictors, reflecting the uncertainty present in real economic indicators. Mixup forces the model to

approximate smoother, more generalizable relationships between predictors and TSR. Because augmentation was performed within companies and used reasonable bounds, synthetic samples remain plausible for real-world financial settings.

For conclusion, data augmentation successfully enlarged the dataset and introduced controlled variability. This step complements feature engineering and dimensionality reduction by directly addressing the risk of overfitting in a relatively small panel dataset. With augmentation, the predictive model for next-month TSR will be exposed to a broader range of scenarios, increasing its ability to generalize beyond the observed company histories.

3. Dimensionality reduction

The reduction step begins from the engineered week-4 splits stored in the project repository. I load the training, validation, and test tables and retain only numeric predictors, leaving out calendar and identifier fields so shapes are comparable across splits. The resulting matrices contain twenty-two columns. Row counts are 538 for training, 115 for validation, and 116 for testing. These shapes serve as the reference for compression and for checking that transformations are consistent across splits.

All numeric columns are standardized with a scaler fit on the training split and then applied to validation and test. This establishes a common scale and avoids any single variable dominating methods that rely on distances or variances. After scaling, I apply two complementary tools. Principal components analysis provides compact orthogonal features suitable for modeling. t-SNE provides a two-dimensional view that reveals neighborhood structure that a linear method may not expose.

Principal components analysis is executed in two passes that keep the selection auditable. I first fit a full PCA on the standardized training matrix and compute the cumulative explained variance curve. The goal is the smallest number of components that captures at least ninety-five percent of the variance with a soft cap at fifty components to protect against future feature growth. On this dataset the curve crosses the threshold at component thirteen, so the twenty-two original dimensions are reduced to thirteen while preserving at least ninety-five percent of training variance. I then refit PCA at that target size on the training matrix and project the validation and test matrices into the same orthogonal basis. This alignment ensures that any downstream model evaluates the three splits in a shared coordinate system.

Artifacts are saved for reuse and inspection. The notebook writes the fitted scaler and PCA object together with three transformed matrices, one for each split. A short variance report records per-component contributions and the cumulative curve. The report confirms the rapid rise in explained variance through the early components and the tapering gains thereafter. Because the components are uncorrelated by construction, the reduced design matrix is better conditioned than the raw feature block, which typically simplifies optimization and can temper instability in models sensitive to multicollinearity.

Component patterns offer quick intuition even without a full loading table. Early components draw heavily on macro-level movements such as interest-rate changes and inflation deltas, while later components pick up residual movements tied to firm-level variation. This agrees with expectations that broad economic shifts account for much of the synchronized behavior across firms and months and that a compact set of axes can summarize those shifts effectively. In practice, a thirteen-component representation strikes a good balance between compactness and fidelity for this dataset and leaves room to add a small number of original features later if a model benefits from them.

A second lens is built with t-SNE on the standardized training matrix. I use two target dimensions, initialize with the PCA projection, set perplexity to thirty, and fix the random seed for repeatability. If future versions of the dataset grow very large the code can subsample the training set before embedding to keep runtime predictable; at the present size the full 538 rows are embedded. The output is a table with two columns labeled tsne_1 and tsne_2. Although these coordinates are not used as predictors in modeling, they provide a quick way to examine structure in the feature space and to spot potential irregularities.

The resulting map is not a formless scatter. Dense zones appear where observations share similar configurations of standardized features, with gradual transitions between zones rather than abrupt jumps. This visual pattern is consistent with the PCA summary because concentration of variance in a limited number of components usually aligns with visible neighborhoods in a non-linear embedding. A small number of observations sit at the edges of the main mass, which makes them natural candidates for follow-up diagnostics such as rechecking source rows or reviewing the timing of firm events. The map can be colored by company or by month to surface additional structure when plotting outside the notebook.

These two tools are used for complementary purposes. PCA supplies compact features that can stand in for the original numeric block in baseline models. The reduction from twenty-two variables to thirteen reduces dimensionality by roughly forty percent, which can shorten training time and simplify

regularization choices without a large loss of information. The variance report makes the trade-off transparent and allows for simple sensitivity checks by adjusting the number of components and observing the change in cumulative variance. t-SNE, on the other hand, is treated as an inspection instrument rather than a modeling ingredient. Its value here is to confirm that the standardized design matrix has meaningful geometric structure and to give an accessible picture that supports narrative explanations in the appendix.

To make the stage practical for others who open the notebook, inputs are read directly from the repository so file locations are predictable, scaling is fit on the training split and applied consistently elsewhere, and outputs are written in common formats. The PCA features for train, validation, and test are saved alongside the variance report and the two-dimensional training map from t-SNE. This small set of files is enough for a reviewer to reproduce the projections, compare model performance with and without PCA, and create figures without rerunning earlier engineering steps.

Overall, the reduction stage distills the engineered predictors into a clean, orthogonal basis that preserves almost all training variance while cutting width meaningfully, and it provides a concise two-dimensional map that corroborates the presence of structure. The pair of views makes subsequent modeling more efficient and interpretation more straightforward, while keeping the workflow simple for anyone who runs the notebook from the shared repository.