Week 13 — Bring it all together

1. Model Development Life Cycle for REIT Total Shareholder Return Prediction

Our project followed a complete model development life cycle (MDLC) to build a predictive model for Real Estate Investment Trust (REIT) total shareholder returns. The workflow spanned from raw data ingestion through deployment preparation, encompassing seven major phases that align with industry best practices for machine learning systems.

Phase 1: Data Ingestion and Integration

The first step involved collecting and merging multiple data sources into a unified monthly panel dataset. We gathered stock price and dividend data for six publicly traded REITs spanning major U.S. metropolitan areas: Boston Properties (BXP), SL Green Realty (SLG), Elme Communities (ELME), Equity Residential (EQR), Rexford Industrial (REXR), and Terreno Realty (TRNO). These companies operate in Boston, New York, Washington D.C., Chicago, Los Angeles, and Miami respectively.

In addition to firm-level data, we integrated macroeconomic indicators to capture regional and national economic conditions. This included metro-specific Consumer Price Index (CPI) data from the Bureau of Labor Statistics for all six markets, Metropolitan Statistical Area (MSA) unemployment rates, and the 10-year U.S. Treasury yield as a proxy for risk-free rates and broader market sentiment.

The data ingestion process involved several critical transformations. Daily stock prices were aggregated to month-end closing values to align with the monthly frequency of macroeconomic indicators. Dividend payments were summed by ex-dividend month and converted into trailing twelve-month (TTM) totals to capture the full income component of shareholder returns. All date fields were standardized to month-end format to ensure proper temporal alignment across data sources.

The final merged dataset contained one observation per company per month, with features spanning stock prices, dividend metrics, inflation measures (both year-over-year and month-over-month CPI changes), unemployment levels and changes, and Treasury yield movements. This rectangular dataset structure provided a clean foundation for subsequent modeling phases.

Phase 2: Exploratory Data Analysis and Temporal Splitting

The second phase focused on understanding data characteristics and establishing proper train-validation-test splits that respect the temporal nature of financial time series. Our prediction target was monthly Total Shareholder Return (TSR), defined as:

TSR = (Price_next - Price_current) / Price_current + Dividend_next / Price_current

This formulation captures both capital appreciation and dividend income, representing the complete return an investor would realize by holding the stock for one month. The target variable was constructed by computing next month's price and dividend for each company-month observation, ensuring that predictions would not suffer from look-ahead bias.

We implemented a chronological split strategy, allocating 70% of observations to the training set, 15% to validation, and 15% to the final test set. This time-based partitioning is crucial in financial forecasting to prevent information leakage from future periods into model training. The training set covers the earliest time period, validation includes more recent data for hyperparameter tuning and model selection, and the test set represents the most recent months to simulate true out-of-sample performance.

Exploratory data analysis revealed several important patterns. Target variable TSR exhibited moderate volatility with occasional extreme values during market stress periods. CPI showed consistent upward trends across most metros with notable acceleration in recent years. Unemployment rates displayed regional variation and cyclical patterns tied to economic conditions. Correlation analysis highlighted expected relationships such as negative correlation between TSR and unemployment, and positive relationships between different lag periods of the same variable.

Missing value analysis identified sparse gaps in macroeconomic series, primarily due to reporting lags or data collection issues in certain months. These patterns informed our subsequent data cleaning strategy. Distributional analysis showed that while most features exhibited roughly normal distributions, some momentum indicators and rate-of-change variables displayed heavier tails, suggesting the need for robust scaling methods.

Phase 3: Data Cleaning and Preprocessing

The third phase addressed data quality issues and transformed raw features into model-ready inputs. Missing value imputation followed a two-stage approach. For macroeconomic variables, we first applied

forward-fill within each metro area to propagate the most recent valid observation. This method is appropriate for slowly-changing indicators like CPI and unemployment. Any remaining missing values were filled with the median across all observations for that feature, providing a conservative central estimate.

Outlier treatment employed multiple strategies depending on feature characteristics. For bounded variables like unemployment rates, we applied hard clipping to ensure values stayed within plausible ranges (0-25% for unemployment, 0-15% for Treasury yields). For momentum and change variables such as month-over-month CPI growth and Treasury yield changes, we used Interquartile Range (IQR) based winsorization. This method identifies outliers as values falling outside 1.5 times the IQR beyond the first and third quartiles, then clips them to these boundaries. This approach preserves the overall distribution shape while limiting the influence of extreme values.

Feature transformation included logarithmic scaling for skewed variables. Stock prices and trailing dividend amounts were transformed using log(1+x) to compress their dynamic range and reduce the impact of extreme values. This transformation is standard in financial modeling as it converts multiplicative relationships into additive ones, which many machine learning algorithms handle more naturally.

Standardization was applied to all numeric features using scikit-learn's StandardScaler. Critically, the scaler was fit only on the training set, then applied to validation and test sets using the training set's mean and standard deviation. This procedure prevents test set information from leaking into the training process. After standardization, all features had zero mean and unit variance on the training set, putting them on comparable scales for modeling.

Categorical variables (company ticker and metro area) were one-hot encoded to create binary indicator variables. This transformation is necessary for most machine learning algorithms that require numeric inputs. We retained all categories without dropping any reference level to avoid information loss, resulting in six company indicators and six metro indicators.

The final preprocessing step removed redundant raw columns that had been transformed. For example, after creating log-transformed price and dividend features, the original raw values were dropped to avoid multicollinearity. Similarly, after computing rolling statistics and lag features in the next phase, we removed intermediate calculation columns to keep only final modeling features.

Phase 4: Feature Engineering

Feature engineering translated the cleaned dataset into informative predictors that capture temporal patterns and economic relationships. This phase generated 73 total features through several transformation families.

Lag features captured historical information by creating time-shifted versions of key variables. For the target variable TSR and all macroeconomic indicators (CPI year-over-year, CPI month-over-month, unemployment level, unemployment change, Treasury yield, and Treasury yield change), we generated 1-month, 3-month, and 6-month lags. These lags allow the model to learn autoregressive patterns and delayed effects of economic conditions on stock returns. The selection of these specific lag periods balances capturing short-term momentum (1 month), quarterly patterns (3 months), and longer-term trends (6 months).

Rolling window statistics quantified recent trends and volatility. For TSR, CPI month-over-month change, unemployment, and Treasury yield, we computed 3-month and 6-month rolling means and standard deviations. Rolling means capture momentum and trending behavior, while rolling standard deviations measure volatility and uncertainty. These features help the model distinguish between stable and turbulent market conditions.

Interaction features explicitly encoded domain knowledge about economic relationships. We created three key interactions: (1) inflation-unemployment interaction (CPI growth × unemployment rate) to capture stagflation or Phillips curve dynamics, (2) interest rate-dividend interaction (Treasury yield × log dividend) to represent the trade-off between fixed income yields and equity income, and (3) dividend yield ratio (log dividend / log price) to measure the income component of returns relative to valuation.

Time-based features captured seasonal patterns through month-of-year indicators. We one-hot encoded the calendar month (1-12) to allow the model to learn seasonal effects in real estate returns, such as stronger performance in certain quarters or year-end tax considerations.

After generating all features, we removed rows containing NaN values introduced by lag and rolling window operations. These missing values occur naturally in the first few months where insufficient

historical data exists to compute lags or rolling statistics. By dropping these rows, we ensured the final dataset contained complete feature vectors for all observations.

The feature engineering process was applied identically to training, validation, and test sets, with one critical distinction: lag and rolling calculations were computed separately within each company's time series to prevent cross-contamination between firms. The resulting engineered dataset contained 73 features and maintained the temporal split structure for subsequent modeling.

This comprehensive feature set provided the model with multiple representations of the same underlying information—raw values, lags, trends, volatility, interactions—allowing the algorithm to learn complex patterns while retaining interpretability through domain-meaningful transformations.

2.

In the second phase of Week 13, we systematically organized and restructured the entire project. Our goal was to establish a final model development workflow that is clearly structured, logically consistent, and capable of running independently from start to finish, all while adhering to the MDLC framework. Given that the code accumulated over the first 12 weeks evolved incrementally during the learning process, issues such as redundancy, fragmentation, high coupling, and messy dependencies inevitably arose. Consequently, this week's primary task was not introducing new content, but rather determining which components should be retained, which should be discarded, and how to distill a truly standardized pipeline from the multitude of code fragments.

During this reorganization, we first eliminated numerous redundant model training scripts generated in earlier stages. For instance, Weeks 8 and 9 repeatedly trained various models—including ElasticNet, Random Forest, and HGBR—but a production-grade ML pipeline requires only one validated optimal model. Consequently, we retained only Week 9's Lasso model as the final model, removing all training logic related to other models. This reduced code complexity and prevented dependency conflicts among redundant models. Similarly, extensive visualization code, debugging outputs, temporary file writes, and other elements generated throughout the exploration phase were removed during this consolidation. These steps are meaningful only during learning and exploration phases, are unnecessary for the final pipeline, and disrupt the overall structural coherence.

Simultaneously, we organized all data processing steps to ensure a natural progression from Week 5 through Week 6, Week 9, Week 10, Week 11, and finally to Week 12. For instance: - Week5's feature engineering and target variable construction form the foundation for all subsequent model training and must be retained; - Week6's data partitioning and schema output determine the training set structure, being central to model reproducibility; Week 9's optimal model training, as the final outcome of the model selection phase, also requires stable retention; Week 10's data governance steps represent a critical data quality component in real MDLC scenarios and are thus retained in the final version; Week 11 handles model interpretation and bias analysis, while Week 12 focuses on deployment preparation. By reconnecting these key modules, I ensured the final code flows seamlessly from raw features through training, evaluation, interpretation, and deployment.

The rationale behind these omissions is not to reduce content but to enhance professionalism in the process. The final delivery version retains only essential steps that genuinely impact model behavior, eliminating all experimental, redundant, or non-reproducible components. This streamlines the pipeline, making it clearer and more engineering-ready. This version now functions as a standard ML pipeline, aligning with the MDLC specifications required by the course and embodying the maintainability and interpretability sought in industrial model development.

3.

Following the creation of this streamlined version, our team further discussed that while concise and suitable as a final production-grade pipeline, it might have been trimmed too rigorously. We concluded that elements removed in the second version, specifically the Week 9 multi-model comparison, explanations in the model selection section, and Week 8/9 hyperparameter tuning records (including charts generated during model comparison and validation), all serve to demonstrate model justification within the strict MDLC documentation framework. Omitting these components prevented the document from fully showcasing our exploratory efforts during model selection. Consequently, we later reintroduced some content to generate a more comprehensive final version. This revised version reinstated the multi-model comparison, benchmark model performance table, concise model selection rationale, and key charts (e.g., bias plots and feature importance). This approach provides instructors with a more complete presentation of our learning journey and thought process, demonstrating that our final model selection was a deliberate choice among multiple models rather than arbitrary.