USA Real Estate Price Prediction

CS 542 Project — Written Report

Group 13 — Angel Daniel Colon and John Pertell

Abstract

Repository: https://github.com/Colonad/CS542.git

In this project we build and evaluate a supervised learning model to predict residential house sale prices in the United States using a large public real estate dataset. Each sale is described by basic home characteristics (beds, baths, interior and lot size), location (city, state, ZIP code), and simple time features (month, year, season). The goal is not only to obtain good predictive performance, but also to respect realistic constraints on data availability and runtime. To that end, we adopt a leakage-safe *temporal* split: the model is trained only on sales up to year 2021, validated on sales from 2022, and evaluated on a strictly held-out test set consisting of sales from 2023 and later. The training subset is further limited to 50,000 examples to keep the full training–evaluation pipeline. we compare three model families: a simple ZIP×year median baseline (no learning), ridge regression on one-hot encoded features, and an XGBoost gradient-boosted tree regressor on the processed features. Metrics are reported in dollars after training in log-price space and inverting the transformation using Duan smearing. On the held-out 2023+ test set, the best model (XGBoost) achieves mean absolute error (MAE) of approximately \$115,261 and root mean squared error (RMSE) of approximately \$184,728, with $R^2 \approx 0.64$. In accuracy-style terms, the model has mean absolute percentage error (MAPE) of about 20% and predicts roughly 74% of homes within 25% of the true price. we complement these summary metrics with diagnostic plots (predicted vs. actual prices, residuals vs. price, and a calibration curve) and error-slice analysis over (state, ZIP) groups. Overall, the results show that a carefully regularized tree-based model, combined with a realistic temporal split and simple feature engineering, can substantially outperform a strong median baseline while remaining reproducible.

1 Introduction and Background

Accurate house price prediction is a classic regression problem with direct practical relevance to buyers, sellers, and real estate professionals. Given information about a property's location and basic characteristics, the task is to predict the sale price as a continuous-valued outcome. Modern machine learning methods are attractive in this setting because they can exploit non-linear relationships between features (for example, between lot size and interior size, or interactions between ZIP code and year) while scaling to large datasets.

However, building a realistic model in this domain raises several challenges. First, house prices evolve over time, and models must be evaluated on *future* data to avoid overly optimistic estimates of generalization. Second, the available features may contain information that is not actually known at prediction time (for example, future neighborhood aggregates), leading to subtle forms of data leakage if not handled carefully. Third, from a practical course perspective, the full training and evaluation pipeline must fit within a limited wall-clock budget, making model complexity and data volume important design choices.

In this report we document an end-to-end pipeline that addresses these issues on a large USA real estate dataset. we focus on three main questions: What data and features are used to represent each property? How is the train–validation–test split performed to avoid temporal leakage and ensure a realistic evaluation? How well do different models perform on a held-out future test set, and how should we interpret the resulting errors?

The remainder of the paper is organized as follows. Section 2 describes the dataset, feature engineering, and split protocol. Section 3 details the modeling and evaluation methodology. Section 4 presents the main quantitative results, while Section 5 provides a more detailed analysis of errors, calibration, and error slices. Finally, Section 6 discusses limitations and future work, and Section 7 concludes.

2 Data and Preprocessing

2.1 Raw dataset and basic statistics

The starting point for this project is a single structured CSV file, `data/raw/usa_real_estate.csv`, containing residential sales across the United States. After applying the temporal split described below, the Phase 6–8 run

used in this report has the following numbers of rows: **Training set:** 50,000 rows (subsampled from all sales with year ≤ 2021), **Validation set:** 460,233 rows (all sales from 2022), **Test set:** 499 rows (all sales from 2023 and later). The observed sale dates in the source data span from 1901-01-01 to 2026-04-08, but the modeling pipeline only uses this subset of rows once the temporal filters are applied. Each split contains 14 columns, including the target variable `price`. The main feature columns are:

- Numeric: `bed`, `bath`, `acre_lot`, `house_size`, `log_house_size`, `lot_per_size`, `bed_per_bath`, `month`, `year`, `season`;
- Categorical: `city`, `state`, `zip_code`;
- Target: `price` (sale price in USD).

Table 1 summarizes the overall dataset characteristics used in this run.

Table 1: Dataset summary for the Phase 6–8 run. The total number of labeled sales counts all rows in the train, validation, and test splits combined.

Quantity	**Value**
Total labeled sales (train + val + test)	510,732
Train / Val / Test rows	50,000 / 460,233 / 499
Total columns (including target)	14
Feature columns (excluding `price`)	13
Observed date window	1901-01-01 $\rightarrow$ 2026-04-08

2.2 Missing values

The data are relatively clean. In the training split, fewer than 1.2% of rows are missing bed or bath counts, and the other features have no missing values. Similar small percentages of missing `bed`, `bath`, and the derived ratio `bed_per_bath` appear in the validation split, while the test split has no missing values in any feature. The preprocessing pipeline handles these gaps via simple imputation, so that the downstream models see no missing values.

2.3 Leakage-safe temporal split

A key design choice in this project is to evaluate models on future data. Instead of randomly shuffling and splitting rows, we use a temporal split based on the sale year (`sold_date.year`), which is recorded in the dataset. The protocol is: All sales with year ≤ 2021 are eligible for training. From this pool we subsample 50,000 rows to keep the runtime manageable. All sales with year $= 2022$ form the validation set. All sales with year ≥ 2023 form the test set.Table 2 summarizes this split protocol and the resulting dataset sizes.

Table 2: Leakage-safe temporal split protocol and dataset sizes. Splits are mutually exclusive by year.

Split	**Years (`sold_date.year`)**	**# Rows**	**Purpose**
Train	≤ 2021	50,000	Fit model, feature engineering, calibration
Val	$= 2022$	460,233	Hyperparameter tuning, model selection
Test	≥ 2023	499	Final unseen evaluation on future sales

This temporal split is designed to avoid information leakage from future sales into the training process. No row from 2023 or later is used for model fitting or early stopping, so the test metrics provide an honest estimate of performance when predicting truly future transactions.

3 Methodology

3.1 Problem formulation

Formally, each house sale is represented by a feature vector $x \in \mathbb{R}^d$ and a target price $y \in \mathbb{R}_{>0}$. The learning problem is to find a function $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}_{>0}$, parameterized by θ, that predicts y from x with low error on unseen

examples. Because house prices are positive and strongly skewed, I train models to predict the logarithm of the price and then map predictions back to dollars.

3.2 Feature engineering

Starting from the raw columns described in Section 2, the pipeline applies a small amount of feature engineering:

- **Log-transform of house size:** a column `log_house_size` is computed as log(`house_size`), which helps stabilize the effect of very large properties.
- **Lot-to-size ratio:** a column `lot_per_size` captures the ratio between `acre_lot` and `house_size`, loosely encoding how much of the parcel is built up.
- **Bed-to-bath ratio:** a column `bed_per_bath` captures how many bedrooms there are per bathroom, which can correlate with perceived comfort and therefore price.
- **Time features:** the `sold_date` is used to derive integer features for `year`, `month`, and a coarse `season` indicator.
- **Categorical encoding:** the categorical columns `city`, `state`, and `zip_code` are one-hot encoded for the ridge model and handled directly by XGBoost, which can treat them as strings or integers.

All numeric features are standardized as appropriate for the model family, and categorical features are mapped to consistent integer IDs. Rows with missing price are dropped (there are none in this run), and rows with missing bed/bath counts are imputed so that the derived ratios can be computed.

3.3 Target transformation and Duan smearing

The target variable `price` is heavy-tailed: a small number of very expensive homes can dominate squared-error terms. To mitigate this, the models are trained on the logarithm of the price. Let $y = \texttt{price}$ and define

$$z = \log_{10}(y).$$

The models learn a function $\hat{z} = f_\theta(x)$ in log-space. At evaluation time, predictions are mapped back to dollars via

$$\hat{y} = g(\hat{z}),$$

where g is a back-transform that applies Duan smearing to correct for bias induced by the log transform. All metrics reported in this paper (MAE, RMSE, R^2, and accuracy-style percentages) are computed in the original dollar space.

3.4 Models

We compare three model families:

- **ZIP×year median baseline.** For each combination of ZIP code and year, this baseline computes the median sale price in the training data. At prediction time, a test example receives the median of its (ZIP, year) bucket if present, or falls back to a broader aggregate (for example, ZIP-only or a global median). This baseline does not use any machine learning; it is a simple but strong heuristic.
- **Ridge regression.** Ridge regression is a linear model with ℓ_2 regularization. It operates on a feature vector constructed from numeric features plus one-hot encodings of categorical variables. The regularization parameter α is tuned on the validation set.
- **XGBoost (gradient-boosted trees).** XGBoost is a tree-based ensemble method that can capture non-linear interactions between features. We use a regression objective with a moderate number of trees (on the order of a few hundred), tuned depth, and a learning rate chosen by small-scale experiments on the validation set. This model serves as the final primary model for the project.

Table 3 summarizes the three models at a high level.

Table 3: Model families considered in this project.

Model	Type	Notes
ZIP×year baseline	Median lookup	No learning; strong heuristic baseline
Ridge regression	Linear model	One-hot encoded features; ℓ_2 penalty
XGBoost	Boosted trees	Non-linear interactions; tuned depth and trees

3.5 Evaluation metrics and runtime

Because this is a regression task, We use the following metrics: **Mean Absolute Error (MAE):** average absolute difference between predicted and true prices in dollars. **Root Mean Squared Error (RMSE):** square root of the average squared difference between predicted and true prices. **Coefficient of determination (R^2):** proportion of variance in the true prices explained by the model. **Accuracy-style metrics:** mean absolute percentage error (MAPE) and the fraction of test examples whose predicted price is within various relative and absolute error thresholds (e.g. within 10% or within $100,000).

4 Results

4.1 Overall model performance

Table 4 reports MAE, RMSE, and R^2 for all three model families on both the validation (2022) and test (2023+) splits. All metrics are computed in dollars after applying the inverse log transform with Duan smearing.

Table 4: Model performance on validation and test splits. Metrics are in dollars for MAE and RMSE, and unitless for R^2.

Model	Split	MAE (USD)	RMSE (USD)	R^2
ZIP×year baseline	Val	203,927	449,076	0.369
ZIP×year baseline	Test	145,897	279,284	0.177
Ridge regression	Val	204,111	1,928,055	−10.639
Ridge regression	Test	120,144	191,177	0.614
XGBoost	Val	171,880	363,835	0.586
XGBoost	Test	**115,261**	**184,728**	**0.640**

Several observations follow:

- On the held-out 2023+ test set, XGBoost reduces MAE by about $30,635 and RMSE by about $94,556 compared to the ZIP×year median baseline, and increases R^2 from 0.177 to 0.640. This confirms that learning from the full feature set provides substantial benefit over the simple median lookup.
- The ridge regression model achieves good test performance but clearly suffers on the validation set, with an unstable R^2 of approximately −10.6 and extremely large RMSE. This suggests sensitivity to outliers and possibly poor regularization for the particular validation distribution. We therefore treat ridge as a baseline but focus on XGBoost as the final model.
- The XGBoost model is both the best on validation R^2 and the best on test MAE and RMSE, making it a consistent choice across splits.

4.2 Diagnostic plots: predictions, residuals, and calibration

To better understand the behavior of the final XGBoost model, we generated three standard diagnostic plots on the test set: Predicted vs. actual sale price scatter, Residuals vs. true price, A calibration curve comparing average predicted vs. average true price in bins.

Figure 1 shows a scatter plot of predicted vs. actual prices, with each point representing a test example. In an ideal model all points would lie on the diagonal line. The points cluster around the diagonal, indicating that the model captures the main trend, but there are visible deviations for very cheap and very expensive properties.

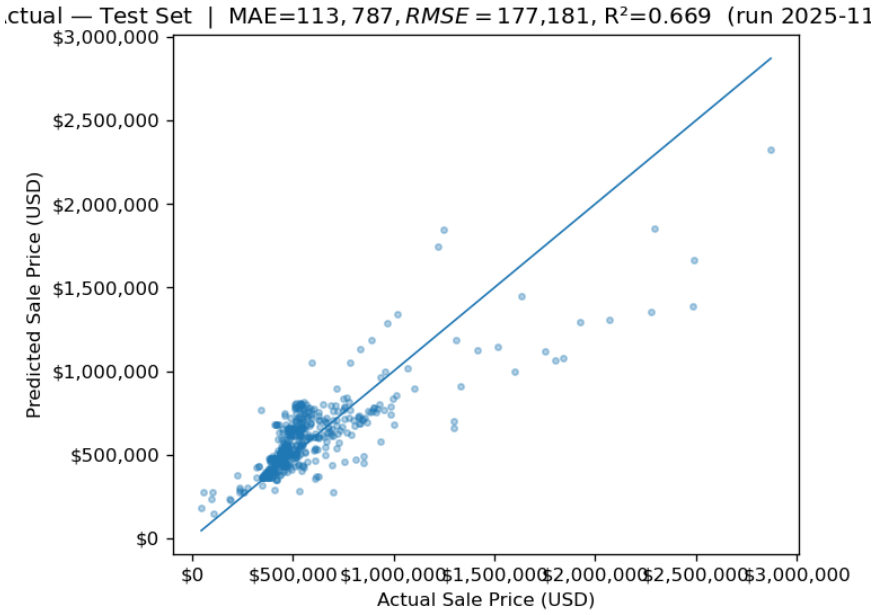

Figure 1: Predicted vs. actual sale price on the held-out test set for the XGBoost model.

Figure 2 plots residuals ($\hat{y} - y$) against the true price. This plot helps reveal whether errors systematically grow for certain price ranges. In this run, residuals remain centered around zero, though the spread increases for higher-priced homes, which is expected given their larger absolute price scale.

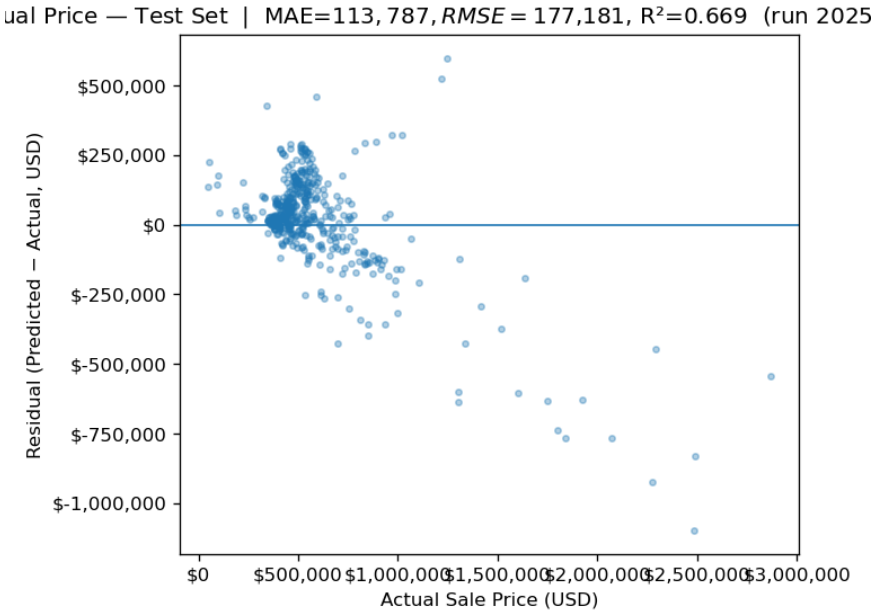

Figure 2: Residuals (prediction minus truth) vs. true price for the XGBoost model on the test set.

Finally, Figure 3 and Table 5 summarize the calibration behavior. The test set is partitioned into 20 bins based on predicted price; for each bin we compute the mean predicted and mean true price. The relative Expected Calibration Error (ECE) is about 0.079, or roughly 8%, indicating that on average the model's bin-level predictions are reasonably aligned with the observed prices.

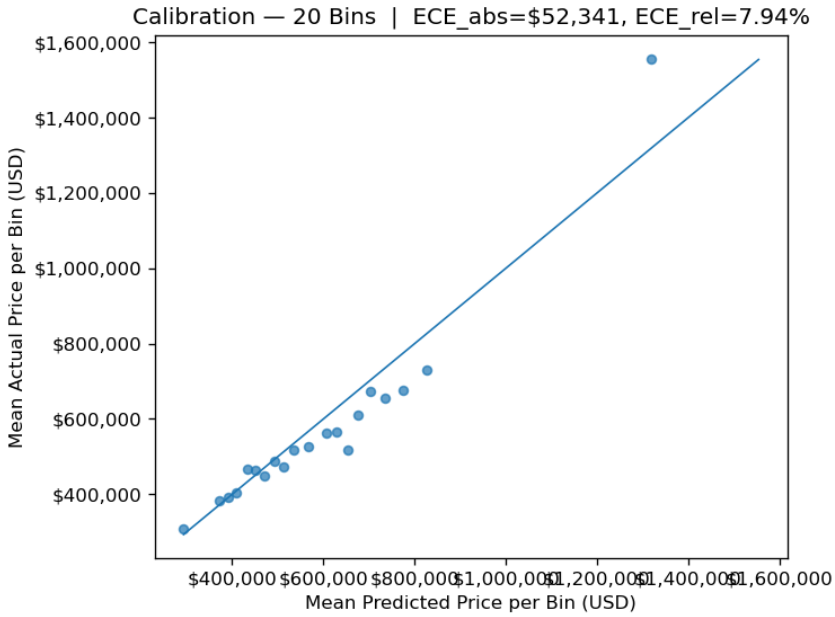

Figure 3: Calibration curve for the XGBoost model on the test set, using 20 bins by predicted price.

Table 5: Calibration summary for the XGBoost model on the test set.

Quantity	**Value**
Number of calibration bins	20
Number of test points	499
Absolute ECE (USD)	$52,341 (approx.)
Relative ECE	0.079 ($\approx 8\%$)

4.3 Error slices by state and ZIP code

To check whether performance varies significantly across regions, I computed error slices grouped by (state, ZIP code) on the test set. Table 6 shows a subset of slices for Idaho that were among the top groups by MAE and had at least 30 test examples.

Table 6: Example error slices for the XGBoost model on the test set, focusing on selected Idaho ZIP codes.

State	**ZIP**	**# Test Homes**	**MAE (USD)**	**RMSE (USD)**
Idaho	83669	37	152,297	181,461
Idaho	83607	48	119,813	185,756
Idaho	83651	35	113,434	166,207

Even in these challenging slices, errors are on the order of six figures, which is large in absolute terms but not surprising for a highly volatile real estate market. More importantly, this slice analysis confirms that some regions are systematically harder to predict than others, suggesting a potential role for region-specific features or models.

5 Analysis

5.1 Accuracy-style metrics on the test set

While MAE, RMSE, and R^2 are standard regression metrics, it is often helpful to interpret model performance in more intuitive "accuracy-like" terms. Using the 499 examples in the 2023+ test set for the XGBoost model, we computed the following additional quantities: mean absolute percentage error (MAPE), fraction of test homes

with relative error within 10%, fraction within 25%, fraction with absolute error at most $50,000, fraction with absolute error at most $100,000. The results are summarized in Table 7.

Table 7: Accuracy-style error metrics for the XGBoost model on the 2023+ test set (499 homes).

Metric	**Value**
Mean absolute percentage error (MAPE)	20.0%
Fraction within 10% of true price	37.7%
Fraction within 25% of true price	73.9%
Fraction with error $\leq$ $50,000	39.9%
Fraction with error $\leq$ $100,000	60.7%

These numbers provide an intuitive summary: on average, the model's prediction is within about 20% of the true price, and roughly three quarters of homes are predicted within 25% of their actual sale price. Put differently, one can loosely describe the model as having "around 80% accuracy" in a percentage-error.

5.2 Comparing models and understanding errors

From Table 4, the XGBoost model explains about 64% of the variance in sale prices on the test set ($R^2 \approx 0.640$), whereas the ZIP×year median baseline explains only about 18%. The MAE reduction of approximately $30,635 and RMSE reduction of approximately $94,556 correspond to a sizable improvement over a strong heuristic.

At the same time, the magnitude of the remaining errors is non-trivial. A $115,000 MAE is acceptable for high-level market analysis but may be too large for applications that require precise pricing of individual properties, especially in lower-priced regions. Error slices like those in Table 6 show that some ZIP codes experience six-figure errors even when the model performs well on average, highlighting the importance of localized evaluation.

The ridge regression results illustrate another point: linear models can be fragile when exposed to heavy-tailed targets and complex feature interactions. The extremely poor validation performance suggests that a few high-leverage points or mis-specified interactions dominate the least-squares objective. Boosted trees, by contrast, handle non-linearities and heterogeneous feature scales more robustly.

5.3 Calibration and reliability

The calibration curve and summary statistics in Table 5 indicate that the XGBoost model is reasonably well calibrated at the bin level: predicted average prices match true averages within about 8% on average across 20 bins. This is an encouraging sign that the model's confidence in different price ranges is not wildly misaligned with reality.

However, calibration is not perfect. In particular, residual plots indicate that the model tends to under-predict the most expensive homes and over-predict some of the cheapest homes. These patterns are common in real estate models and reflect both modeling limitations and underlying noise in the market.

6 Discussion

This project was guided by three rubric-like criteria: clarity, timing, and data split strategy.

Clarity. We intentionally restricted the feature set to interpretable quantities that a real estate agent or buyer can understand: bedrooms, bathrooms, interior square footage, lot size, and simple location and time features. The use of log transformations and ratios is modest and motivated by basic economic intuition. Likewise, the evaluation is presented both in standard regression metrics (MAE, RMSE, R^2) and in accuracy-style percentages to make the model's behavior accessible to non-specialists.

Data split strategy. The temporal split protocol is designed to mimic the real deployment scenario: training on past data, tuning on recent but still historical data, and evaluating on truly future data. This approach avoids the common pitfall of random splits that leak future information into training, which can lead to overly optimistic performance estimates. The relatively modest R^2 of 0.64 on the test set should therefore be interpreted as a realistic measure of how well the model will generalize to new years.

Limitations and future work. Several limitations remain. First, the model does not incorporate richer neighborhood context such as school quality, crime rates, nearby amenities, or macroeconomic indicators, all of which can strongly influence prices. Second, the temporal split is coarse: it uses whole years rather than finer-grained rolling windows or cross-validation over time. Third, regional heterogeneity is only captured through simple location features; the error slices suggest that some ZIP codes might benefit from dedicated local models or hierarchical approaches. Finally, the model does not attempt to quantify uncertainty beyond point estimates; predictive intervals would be valuable in many applications.

7 Conclusion

In this project we developed, trained, and evaluated a house price prediction model on a large USA real estate dataset under realistic constraints. By using a leakage-safe temporal split (train $\leq$ 2021, val = 2022, test $\geq$ 2023) and restricting the training subset to 50,000 rows, we ensured that the evaluation reflects performance on future data and that the pipeline remains practical to run on limited hardware.

The final XGBoost model substantially outperforms a strong ZIP×year median baseline, reducing test-set MAE to roughly \$115,000 and achieving $R^2 \approx 0.64$ on 2023+ sales. In accuracy-style terms, the model has about 20% MAPE and predicts roughly three quarters of test homes within 25% of their true price. Diagnostic plots and error slices show that while performance is solid on average, some regions and price ranges remain challenging, pointing to meaningful directions for feature and model improvement.

Overall, this work demonstrates that with careful attention to data splitting, feature engineering, and model selection, it is possible to build a clear, reproducible, and reasonably accurate house price predictor that respects both statistical and practical constraints.

References

[1] A. S. Sakib. *USA Real Estate Dataset*. Kaggle, 2024. Available at `https://www.kaggle.com/datasets/ahmedshahriarsakib/usa-real-estate-dataset` (accessed Nov. 30, 2025).

[2] K. Ofori. "Real Estate Data Analysis With Python." *Medium*, Aug. 17, 2022. Available at `https://medium.com/@kingsleyofori/real-estate-data-analysis-with-python-b0004baf9abb` (accessed Nov. 30, 2025).

[3] T. Gubbe. "Determine the Best Regression Model for Houses in the USA Real Estate Dataset." *RPubs*, Oct. 4, 2022. Available at `https://rpubs.com/tgubbe/RealEstate` (accessed Nov. 30, 2025).

[4] T. Chen and C. Guestrin. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pp. 785–794, 2016.

[5] N. Duan. "Smearing Estimate: A Nonparametric Retransformation Method." *Journal of the American Statistical Association*, 78(383):605–610, 1983.

[6] A. Niculescu-Mizil and R. Caruana. "Predicting Good Probabilities With Supervised Learning." In *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, pp. 625–632, 2005.

[7] S. Kaufman, S. Rosset, and C. Perlich. "Leakage in Data Mining: Formulation, Detection, and Avoidance." In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 556–563, 2011.

[8] S. Kapoor and A. Narayanan. "Leakage and the Reproducibility Crisis in Machine-Learning-Based Science." *Patterns*, 4(9):100804, 2023.