House Price Prediction on the Ames Dataset: A Comparison of Classical, Ensemble, Neural, and Cluster-Aware Models

Aniruddha Deshpande
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC, USA
Email: adeshp25@charlotte.edu

Abstract

This project presents a systematic comparison of five regression approaches for predicting house prices on the Kaggle *House Prices – Advanced Regression Techniques* dataset. A shared preprocessing and feature engineering pipeline is combined with: (i) ordinary least squares Linear Regression, (ii) Ridge Regression, (iii) a custom Neural Network regressor, (iv) an XGBoost model adapted from recent literature, and (v) a hybrid cluster-wise Ridge model inspired by a two-stage clustering framework. All models are trained on a log-transformed target and evaluated on a held-out validation split using MSE, RMSE, MAE, and R^2. Contrary to prior work that reports XGBoost as the dominant model on Ames-style data, the strongest performance in this study is achieved by the simplest method: Linear Regression, which attains the lowest error and highest R^2 among all models. The paper analyzes why the linear baseline can outperform more complex methods under a carefully designed preprocessing pipeline, discusses the strengths and limitations of the literature-inspired approaches, and reflects on practical challenges in reproducing published results in a constrained course setting.

Index Terms

Machine Learning, Regression, House Price Prediction, XGBoost, Clustering, Neural Networks, Ridge Regression, Interpretability.

I. Introduction

A. Problem Statement and Motivation

Estimating the selling price of a house from its characteristics is a classic regression problem with real economic impact. Prices depend on a mix of structural attributes (living area, number of rooms, quality), location-specific effects, and complex interactions between features. From a machine learning perspective, this setting is well suited for comparing model families: classical linear models, regularized extensions, neural networks, gradient boosting, and more structured ideas such as clustering plus local models.

This project uses the Ames housing data as a controlled environment to explore not only how far error metrics can be reduced, but also how different model classes behave when they all share the same preprocessing and feature engineering pipeline.

B. Objective

The main objectives of this capstone project are:

- Build a clean, reusable preprocessing and feature engineering pipeline for house price prediction.
- Implement and evaluate three models studied in class: Linear Regression, Ridge Regression, and a custom Neural Network regressor.
- Reproduce and adapt two research-based ideas:
  - an XGBoost-based house price prediction pipeline.
  - a cluster-wise modeling strategy inspired by a two-stage clustering paper.
- Compare all five models using MSE, RMSE, MAE, and R^2 on a held-out validation split.
- Discuss why certain models work better than others in this specific configuration and what this means for real-world deployments.

C. Dataset Summary

All experiments use the Kaggle competition dataset:

- **Name:** *House Prices – Advanced Regression Techniques*
- **Link:** https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

The training set (`train.csv`) contains 1460 houses with a target variable `SalePrice` and 79 explanatory features. These cover lot and living area sizes, numbers of rooms and

bathrooms, quality and condition ratings, garage and basement properties, neighborhood and zoning information, year built and remodeled, and other structural and location-related attributes.

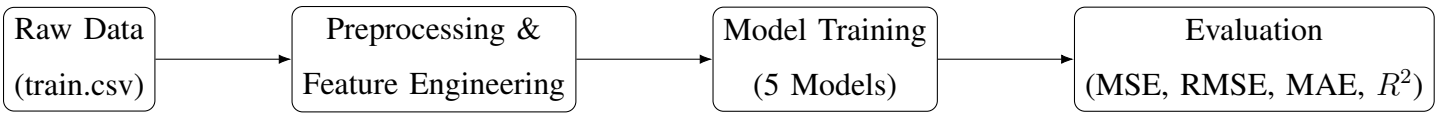

Fig. 1. High-level workflow: data loading, preprocessing, model training, and evaluation.

II. Methodology

A. Preprocessing and Feature Engineering

All five models share a common preprocessing pipeline implemented with scikit-learn transformers so that transformations are consistent across training and validation folds.

1) Train/Validation Split: The labeled `train.csv` file is split into 80% training and 20% validation data using `train_test_split` with `random_state = 42` and shuffling enabled.

2) Target Transformation: House prices are right-skewed. To dampen the impact of very expensive properties and stabilize the loss, the target is transformed using

$$y = \log(1 + \texttt{SalePrice}).$$

All models are trained on this transformed target. Predictions can be mapped back to the original dollar scale with the inverse transformation if needed.

3) Handling Missing Values: Numeric features use median imputation; categorical features use the most frequent value. This avoids dropping rows and handles missing information in a simple, robust manner.

4) Encoding and Scaling:

- **Numeric features:** standardized using `StandardScaler`.
- **Categorical features:** one-hot encoded via `OneHotEncoder` with `handle_unknown='ignore'`.

These are combined with a `ColumnTransformer` so the pipeline accepts raw `DataFrame` inputs and produces a numerical feature matrix suitable for all models.

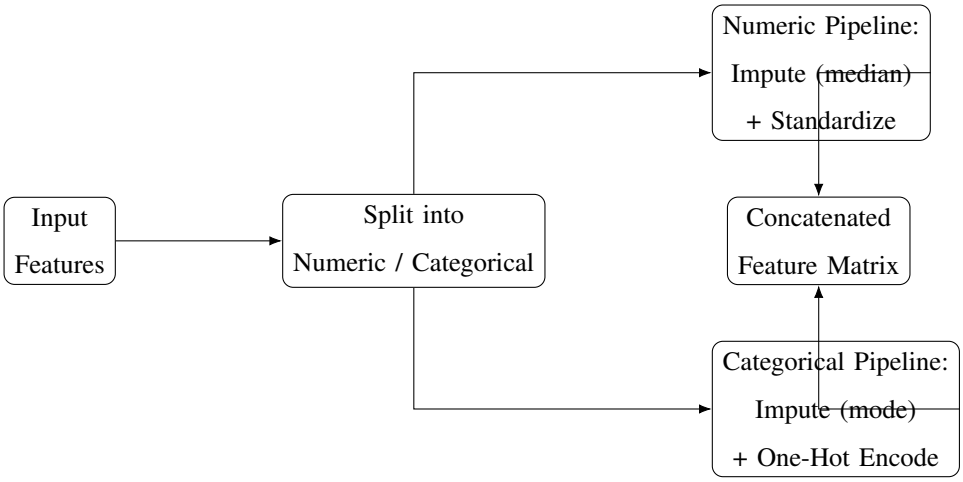

Fig. 2. Preprocessing and feature engineering pipeline shared by all models.

B. Classical Models

1) Linear Regression: The first baseline is ordinary least squares Linear Regression. It assumes a linear relationship between the log-transformed price and the encoded features. Despite its simplicity, it provides a strong and highly interpretable reference point.

2) Ridge Regression: Ridge Regression adds an L2 penalty to the sum of squared coefficients. This helps:

- stabilize the model when predictors are correlated,
- reduce variance and overfitting,
- shrink coefficients and make them easier to interpret.

The regularization strength α is tuned over a small grid of candidate values.

3) Neural Network Regressor: A custom feed-forward neural network is implemented in `03_NeuralNetwork_regression.ipynb` using NumPy. The network includes:

- multiple dense hidden layers with a non-linear activation (e.g., ReLU),
- a linear output layer for regression,
- mini-batch gradient descent,
- tracking of training and validation loss over epochs.

The emphasis is on understanding the mechanics of backpropagation and optimization on a real dataset rather than beating all other models.

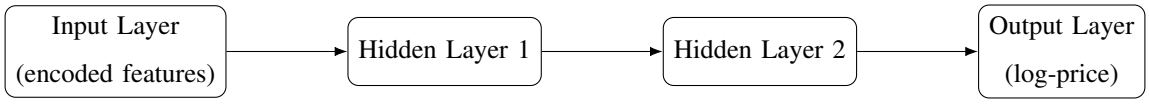

Fig. 3. Simplified architecture of the custom neural network regressor.

C. Paper-Inspired Models

1) Paper 1: XGBoost-Based House Price Prediction: Sharma. compare several regression models on the Ames dataset and conclude that XGBoost, when carefully tuned, provides the best performance. They evaluate linear regression, multilayer perceptron, random forest, SVR, and XGBoost, and analyze feature importance to identify key drivers such as overall quality and living area.

The `04_XGBoost.ipynb` notebook follows this idea by:

- applying the same preprocessing pipeline as the other models,
- training a baseline XGBoost model,
- tuning max depth, learning rate, number of estimators, subsampling, and regularization parameters,
- reporting MSE, RMSE, MAE, and R^2 on the validation set,
- visualizing feature importances and residual distributions.

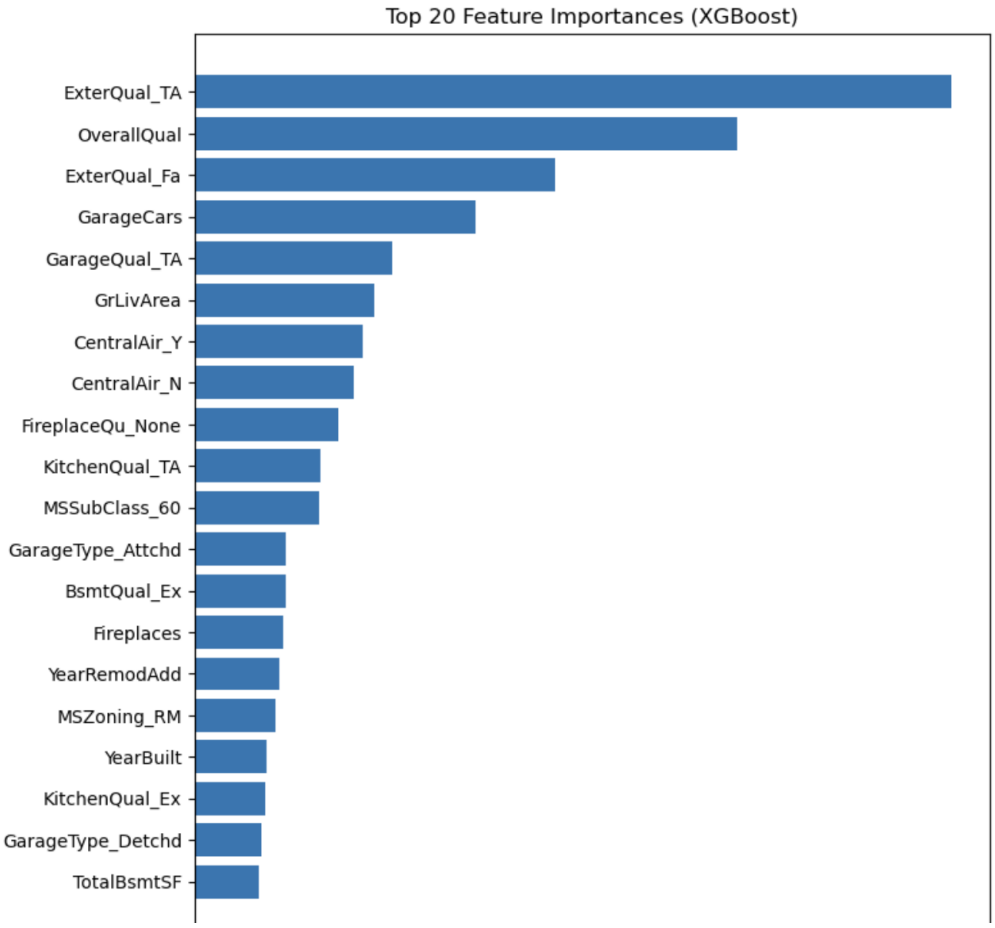

Fig. 4. Top 20 feature importances for the tuned XGBoost model.

2) Paper 2: Cluster-Aware Hybrid Ridge Modeling: Gümmer et al. focus on interpretability. They propose a two-stage clustering pipeline for real-estate listings, where:

- a first clustering stage groups properties based on location-related features,
- a second stage refines clusters using structural attributes,
- simple models (e.g., linear regression or GAMs) are trained inside each final cluster,
- the approach yields lower error and richer cluster-level insights.

The `05_Clusterwise_Ridge_Hybrid.ipynb` notebook adapts this idea to the Ames dataset in a simplified form:

- a clustering step segments houses into a fixed number of clusters using selected features,
- a Ridge Regression model is fitted separately in each cluster,
- at prediction time, each sample is assigned to the nearest cluster and the corresponding local Ridge model is used.

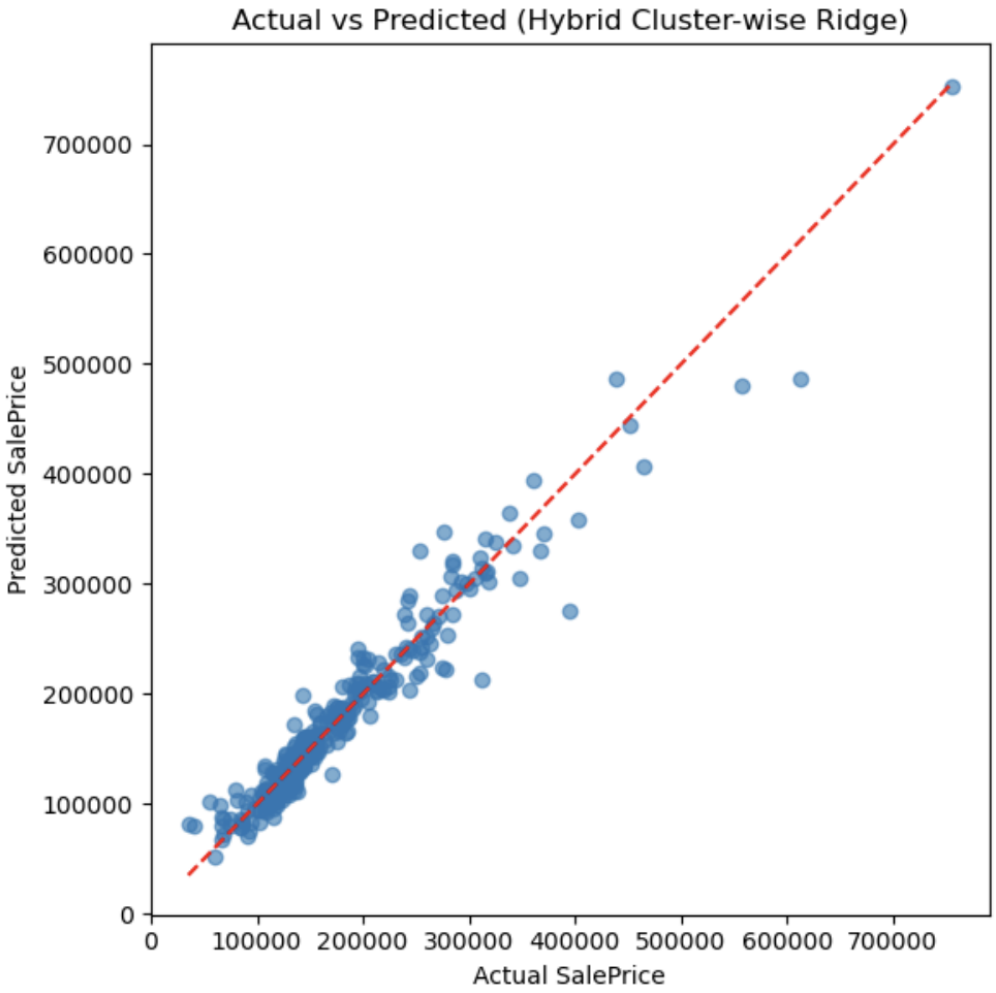

Fig. 5. Actual vs. predicted sale prices for the hybrid cluster-wise Ridge model.

III. Results and Evaluation

A. Evaluation Metrics

The following metrics are used:

- Mean Squared Error (MSE),
- Root Mean Squared Error (RMSE),
- Mean Absolute Error (MAE),
- Coefficient of Determination (R^2).

All metrics are computed on the held-out validation split using the log-transformed target.

B. Numerical Comparison of Models

Table I summarizes the final metrics for all five models based on the notebook outputs. Models are ranked primarily by MSE.

TABLE I
MODEL PERFORMANCE ON VALIDATION SET (LOG-TRANSFORMED TARGET)

Model	Rank	MSE	RMSE	MAE	R^2
Linear Regression	1	330,891,436.32	18,190.42	11,902.68	0.9569
Clusterwise Ridge	2	491,499,533.76	22,169.79	14,724.59	0.9359
Ridge Regression	3	499,570,453.33	22,351.07	14,556.06	0.9349
XGBoost (tuned)	4	647,726,550.40	25,450.47	15,303.77	0.9156
Neural Network	5	1,297,576,687.49	36,021.89	22,873.22	0.8308

The most striking result is that Linear Regression, combined with strong preprocessing and a log transformation of the target, achieves the best overall performance: it has the lowest MSE, RMSE, and MAE and the highest R^2. The cluster-wise Ridge model and global Ridge Regression form a second tier. The tuned XGBoost model is competitive but does not surpass the linear baselines in this configuration. The custom neural network performs worst among the five.

C. Plots from the Notebooks

1) Linear Regression: Figure 6 shows the actual vs. predicted sale prices for Linear Regression, with a dashed diagonal as the ideal line. The points lie close to the diagonal, matching the strong numeric metrics. Figure 7 shows the residual distribution, which is roughly centered around zero with moderate spread.

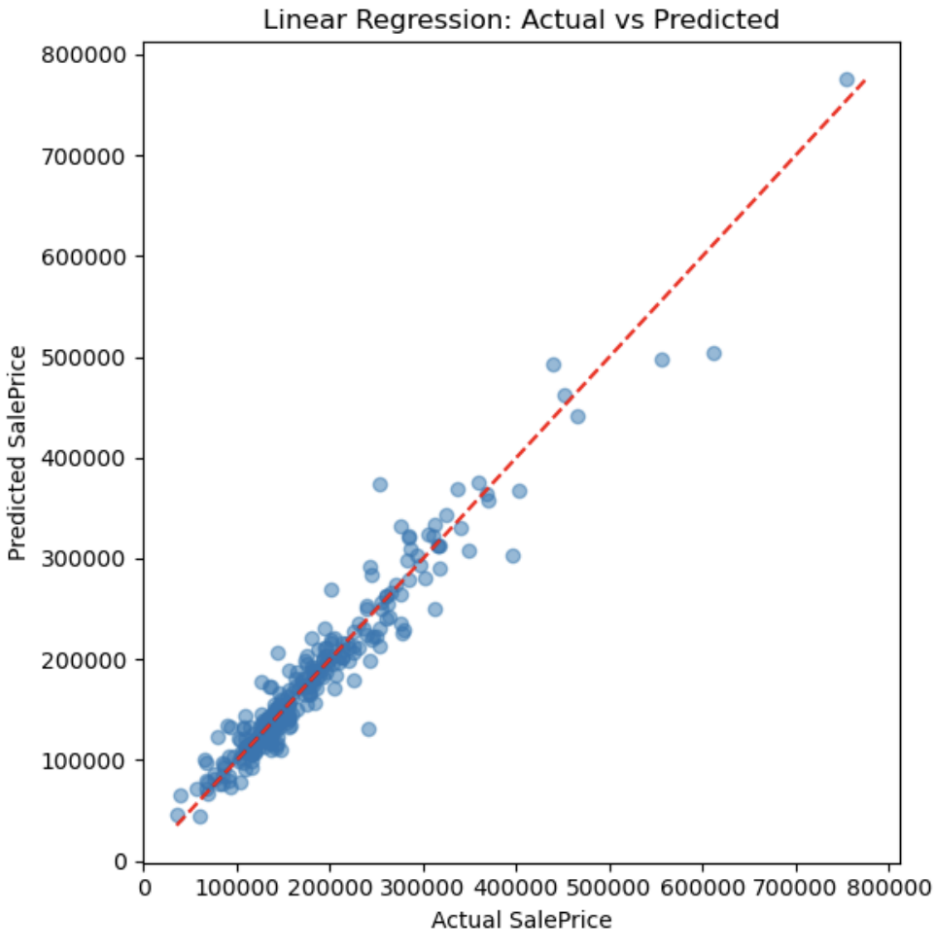

Fig. 6. Linear Regression: actual vs. predicted sale prices.

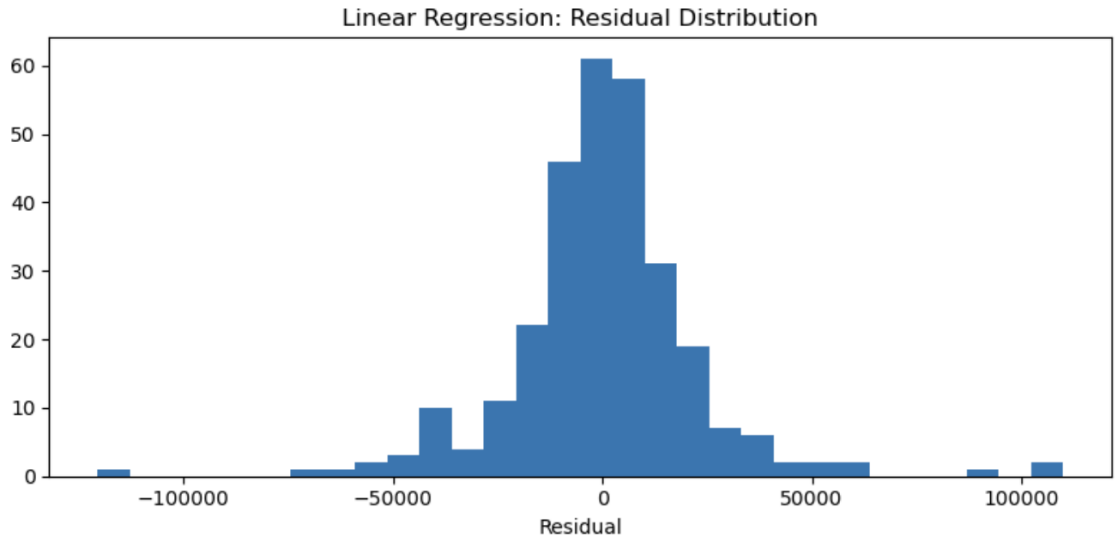

Fig. 7. Linear Regression: residual distribution on the validation set.

2) Ridge Regression: Ridge Regression shrinks coefficients and slightly trades bias for variance reduction. Figure 8 and Figure 9 show its actual vs. predicted plot and residuals vs. predicted values. The residual spread is slightly wider than for Linear Regression, which is consistent with the weaker metrics.

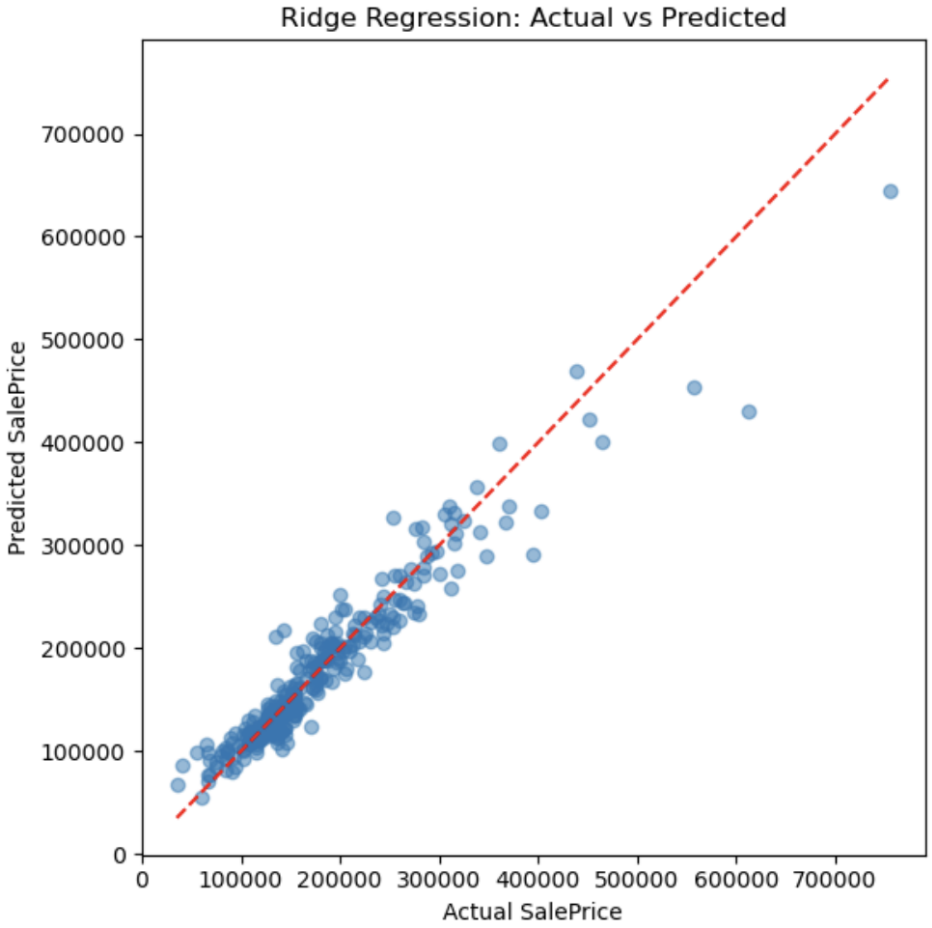

Fig. 8. Ridge Regression: actual vs. predicted sale prices.

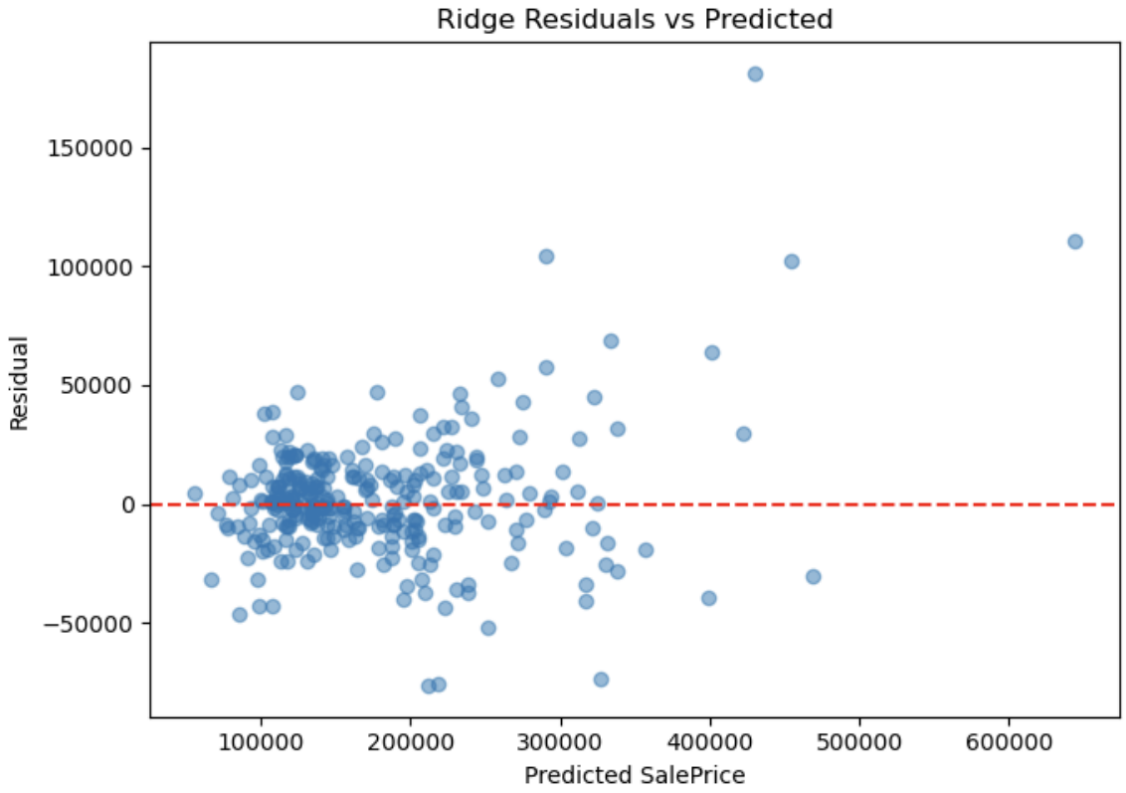

Fig. 9. Ridge Regression: residuals vs. predicted sale prices.

3) Neural Network: For the custom neural network, Figure 10 shows the actual vs. predicted plot. The points still follow the general diagonal trend but with visibly more spread. Figure 11 shows the training and validation loss over epochs. The loss drops quickly and then flattens, suggesting that the network is converging, but the final plateau is higher than the best linear model.

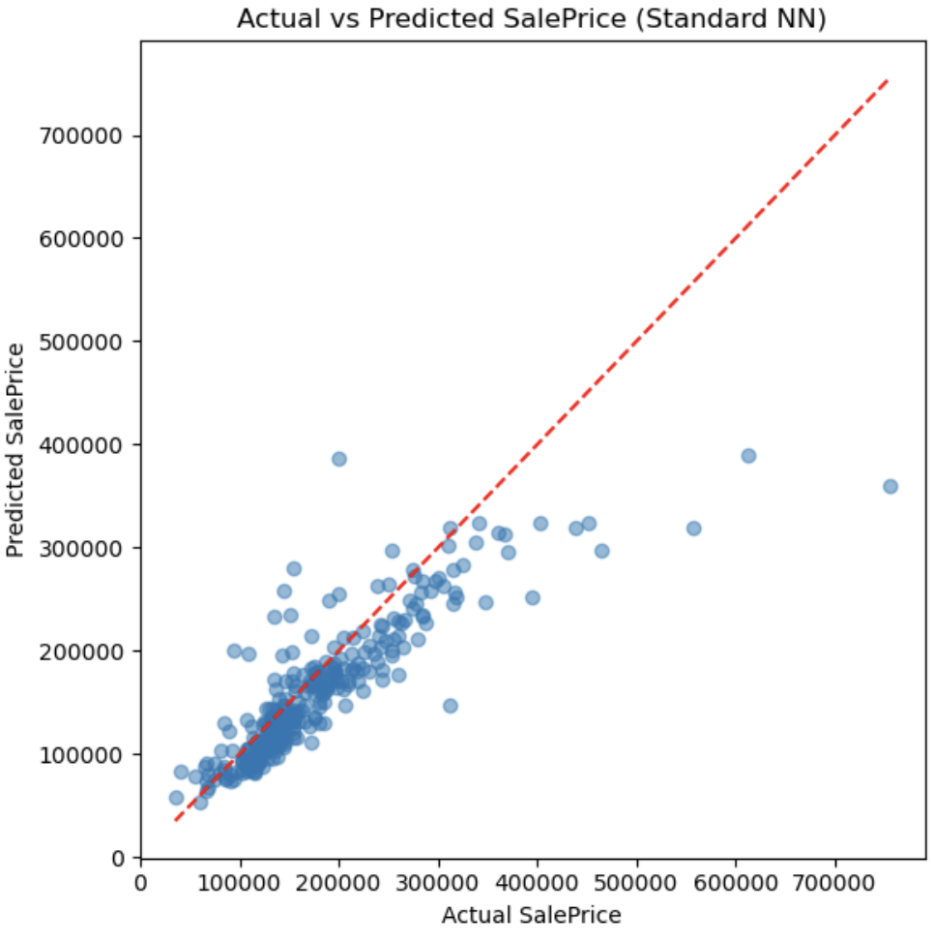

Fig. 10. Neural Network: actual vs. predicted sale prices.

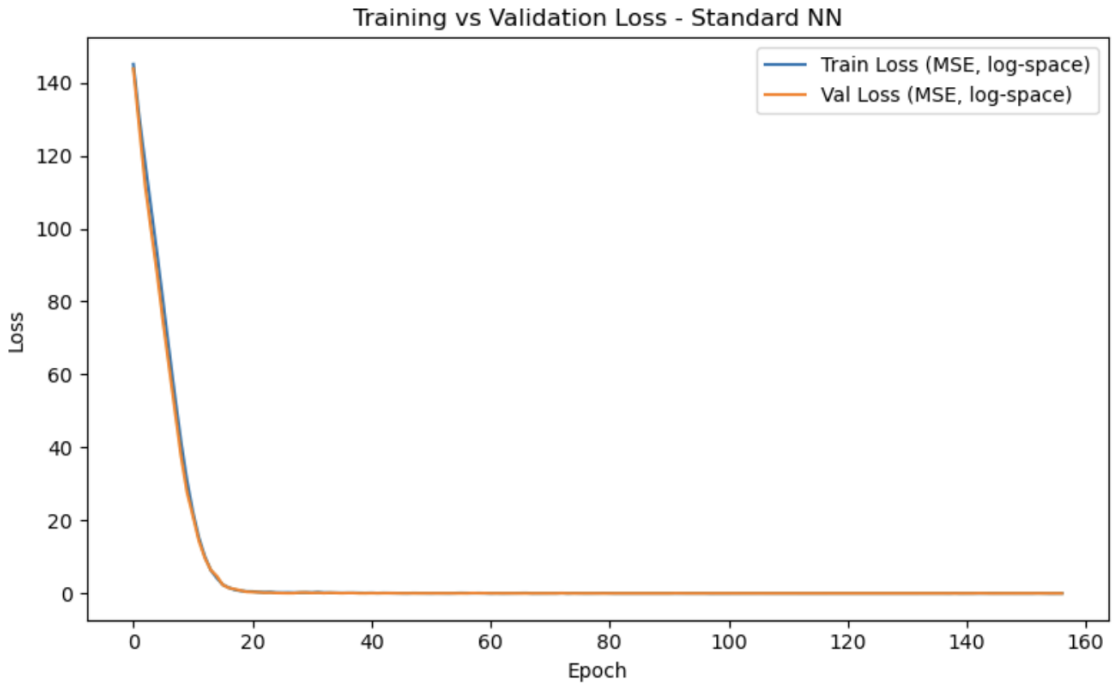

Fig. 11. Neural Network: training vs. validation loss over epochs.

4) XGBoost: For XGBoost, Figure 12 shows the actual vs. predicted plot and Figure 13 shows the residual distribution. The residuals are nicely centered around zero, but their spread remains larger than in the Linear Regression case.

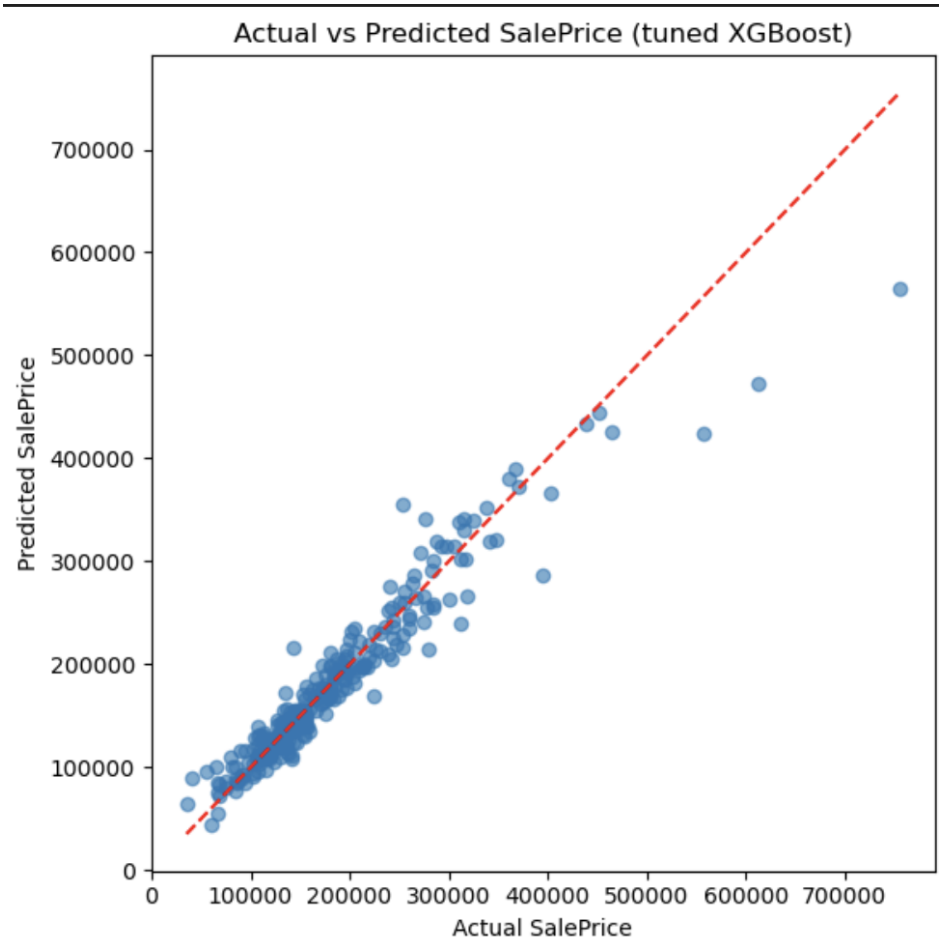

Fig. 12. Tuned XGBoost: actual vs. predicted sale prices.

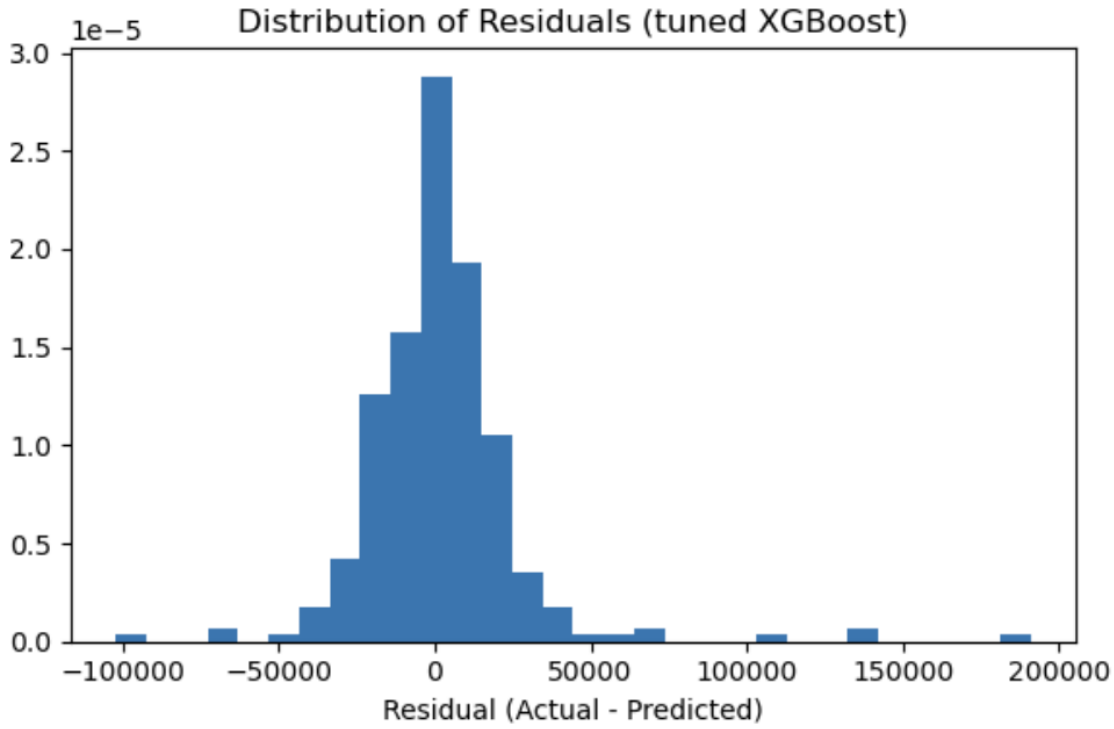

Fig. 13. Tuned XGBoost: residual distribution on the validation set.

5) Clusterwise Ridge Hybrid: Figure 5 summarizes the hybrid cluster-wise Ridge model's predictions compared to ground truth. Visually, it is comparable to Ridge Regression. The metrics place it slightly better than Ridge but still below the global Linear Regression model.

IV. Discussion

A. Why Linear Regression Won

At first glance it is surprising that Linear Regression beats XGBoost and a cluster-based model, especially since prior work reports XGBoost as the top performer on the Ames dataset. Several factors help explain this outcome:

- The log transformation of `SalePrice` makes the relationship between features and target much closer to linear, which strongly benefits linear models.
- The preprocessing pipeline is consistent and fairly strong: missing-value handling, scaling, and one-hot encoding are all handled carefully.
- The XGBoost implementation, while tuned, explores a smaller hyperparameter search space than in the paper and is constrained by project time.

- The dataset is moderately sized, so highly flexible models can overfit unless heavily regularized and tuned.

Under these conditions, a well-specified linear model can achieve very competitive performance, and in this project it becomes the best overall model.

B. Paper Methods vs. Project Results

Sharma et al. find that XGBoost outperforms other models such as linear regression and multilayer perceptron. In this project, tuned XGBoost still performs well but is clearly behind Linear Regression in terms of MSE and MAE. This does not contradict the paper; instead, it highlights how sensitive XGBoost performance is to the exact tuning strategy, choice of loss, and treatment of the target variable.

Gümmer et al. show that cluster-wise modeling can reduce error and improve interpretability in a large real-estate dataset. Here, the hybrid cluster-wise Ridge model performs slightly better than a single global Ridge model but does not beat the global Linear Regression. One likely reason is that the simplified clustering scheme and smaller dataset limit the benefits of segmentation. Nevertheless, the experiment confirms that local models can be competitive and may offer more nuanced insights in larger or more heterogeneous markets.

C. Neural Network Limitations

The custom neural network is the weakest model in this comparison. Several factors contribute:

- The architecture and learning rate schedule are relatively simple and not heavily tuned.
- Dense neural networks are not always the best choice for medium-sized tabular datasets, which often favor tree ensembles or well-prepared linear models.
- Regularization techniques (dropout, weight decay) and more advanced architectures were not fully explored due to time constraints.

The neural network experiments are still valuable from a learning perspective, as they force careful debugging of backpropagation and reveal how optimization behavior looks on real data.

D. Limitations

There are several limitations to this study:

- The hyperparameter searches for Ridge, XGBoost, and the neural network are relatively small compared to what would be used in a production system.

- Only a single train/validation split is used; cross-validation would provide a more robust estimate of generalization performance.
- The cluster-wise Ridge implementation uses a simplified clustering strategy compared to the full two-stage pipeline in.
- External data sources (e.g., additional neighborhood statistics or temporal information) are not incorporated.

These limitations should be kept in mind when interpreting the numeric results.

V. Conclusion and Future Work

This project implemented and compared five models for house price prediction on the Kaggle Ames dataset: Linear Regression, Ridge Regression, a custom Neural Network, a paper-inspired XGBoost model, and a hybrid cluster-wise Ridge model. All models shared the same preprocessing and feature engineering pipeline, which helped isolate the effect of modeling choices.

The main findings are:

- A well-preprocessed **Linear Regression** model with a log-transformed target achieved the best overall performance on the validation set and ranked first among all five models.
- Ridge Regression and the hybrid cluster-wise Ridge model formed a close second tier, confirming that linear models remain very competitive for this task.
- The tuned XGBoost implementation performed well but did not reach the performance of Linear Regression under the current tuning and data configuration.
- The custom neural network underperformed, highlighting the difficulty of applying generic dense neural networks directly to medium-sized tabular datasets without extensive tuning.

For future work, several directions are promising:

- Explore a more extensive hyperparameter search for XGBoost and compare results on both the raw and log-transformed targets.
- Refine the clustering strategy (e.g., full two-stage clustering as in Gümmer et al.) and experiment with different numbers of clusters.
- Add model-agnostic interpretation tools such as SHAP to analyze both tree-based and linear models.
- Investigate more advanced neural network architectures tailored for tabular data, such as TabNet or attention-based models.

VI. Reproducibility Notes

All experiments were run in Python using scikit-learn and XGBoost. The notebooks are organized so that each cell can be executed top-to-bottom to reproduce the reported metrics. To reproduce the results, a user should:

- install Python 3.x and the required libraries listed in the repository `README.md`,
- download the `train.csv` and `test.csv` files from the Kaggle competition page,
- place the CSV files in the project directory,
- run the notebooks in the order listed below.

Random seeds are fixed where possible (e.g., `random_state = 42`) to improve reproducibility of splits and initializations.

VII. Implementation Code

All code for this project is organized into a GitHub repository named `IntroMLCapstone`. The repository contains at least the following notebooks:

- `01_LinearRegression.ipynb`
- `02_RidgeRegression.ipynb`
- `03_NeuralNetwork_regression.ipynb`
- `04_XGBoost.ipynb`
- `05_Clusterwise_Ridge_Hybrid.ipynb`

GitHub link: https://github.com/AniruddhaDesh1609/MLCapstone

VIII. References

References

[1] H. Sharma, H. Harsora, and B. Ogunleye, "An Optimal House Price Prediction Algorithm: XGBoost," Accessed: Dec. 2025.

[2] P. Gümmer, J. Rosenberger, M. Kraus, P. Zschech, and N. Hambauer, "Unveiling Location-Specific Price Drivers: A Two-Stage Cluster Analysis for Interpretable House Price Predictions," Accessed: Dec. 2025.