House Prices Prediction using Linear Regression with Cross Validation

Paula Lozano Gonzalo

February 12, 2025

1 Introduction

This report details the process of developing and evaluating regression models for predicting house prices using the Kaggle House Prices dataset. The goal was to train multiple regression models, evaluate them using cross-validation, and submit predictions to the Kaggle competition.

2 Data Understanding

The dataset was obtained from Kaggle's House Prices Advanced Regression Techniques competition. Initial exploration revealed:

- Approximately 80 feature columns
- Mix of numerical and categorical data
- Every column (except the label) contains missing values
- Data was pre-split into training and testing sets and the testing set was missing the label

3 Data Exploration and Visualization

Visualizing the data in this assignment was definitely more difficult since there are around 80 features and we can't plot them in the same page so its harder to compare them to each other. Examples of generated visualizations include:

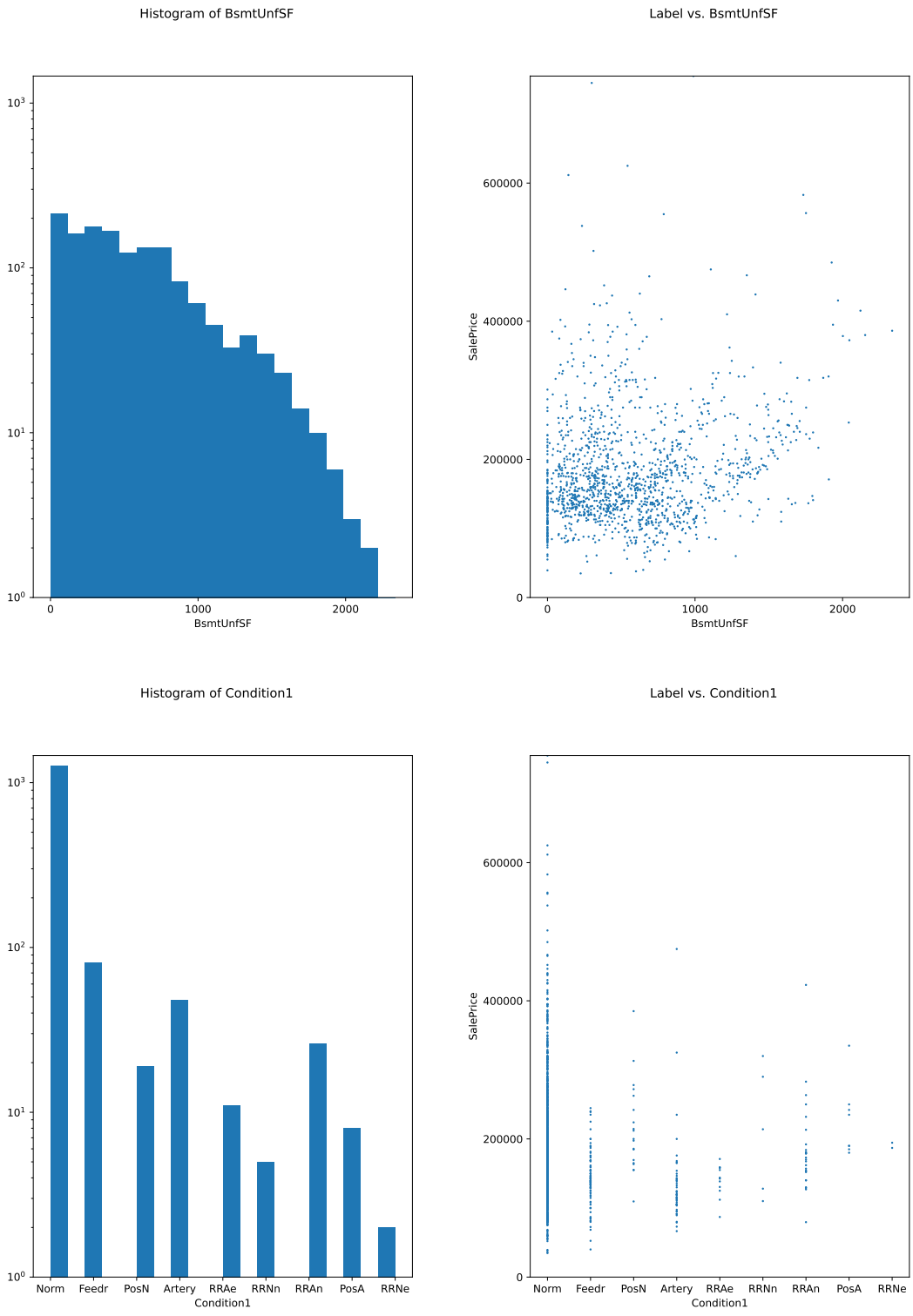

Figure 1: Feature Visualization Examples

4 Data Preparation and Pipeline Development

A pipeline of pipelines was developed using the Data Frame Selector:

- **Numerical features**: Processed using median imputation, scaling, and polynomial features.
- **Categorical features**: Handled via one-hot encoding.

The pipelines were combined using `FeatureUnion` to generate a unified dataset for model training.

5 Model Development and Evaluation

Multiple models were trained and evaluated using 3-fold cross-validation. Below is a summary of performance metrics:

Table 1: Model Performance Comparison

Model	Cross-Val R^2	Cross-Val MAE	Train R^2	Kaggle Score
SGDRegressor	-2.48e+20	-1.08e+15	N/A	N/A
Ridge	0.816	18,689.34	0.934	N/A
RidgeCV	0.820	18,441.60	0.934	N/A
RidgeCV (fine-tuned)	0.839	17,402.39	0.915	0.14651
Lasso	0.813	18,673.52	0.934	N/A
Random Forest	0.719	26,592.31	1.000	0.21058

The SGDRegressor wasn't really trained because, after looking at its cross-validation results, it was clear that it wasn't a good fit at all.

Among the other models, I trained all of them but only submitted the fine-tuned RidgeCV and Random Forest. RidgeCV fine-tuned was my pick since it performed the best (or slightly better) among the linear models, and I had put the most effort into optimizing it. Lasso and Ridge were close contenders, but RidgeCV fine-tuned edged them out. Random Forest was included purely out of curiosity (I wanted to see how badly an extremely overfitted model would perform on Kaggle).

6 Model Analysis

6.1 Initial Findings

- SGDRegressor performed poorly with extreme negative R^2 scores.
- Ridge and RidgeCV yielded comparable results with R^2 around 0.82. They also overfitted the training data.

6.2 Fine-Tuning RidgeCV

RidgeCV was fine-tuned using:

- alphas: logspace(-3, 3, 50)
- cv: 5-fold
- gcv_mode: 'svd'

The fine-tuned RidgeCV model:

- Improved cross-validation R^2 to 0.839.

- Reduced training R^2 to 0.915, indicating slightly better generalization.
- Achieved better MAE (-17,402.39). Which, if I dont understand wrong means that we are off by 17,400 dollars.

6.3 Alternative Models

- Lasso performed similarly to Ridge.
- Random Forest exhibited extreme overfitting with training R^2 of 1.0.

7 Challenges and Solutions

- **Data Visualization**: There are around 160 plots in between histograms and scatter plots so its very hard to compare them all and visualize everything.
- **Missing Values**: Median strategy used for numerical features; categorical features handled in pipeline.
- **Model Convergence**: Increased iterations to ensure Lasso convergence.
- **Computational Efficiency**: Avoided polynomial features due to high dimensionality. I did try once doing cross validation with polinomial degree 2 and the first cross validation took 3 minutes and the results were not improving at all so I stopped there.

8 Final Results

The fine-tuned RidgeCV model was chosen for final submission:

- Kaggle Score: 0.14651
- Best balance of performance and generalization (even though I still think is very overfit)

9 Conclusions and Future Work

- Linear models (Ridge, RidgeCV, Lasso) performed better than tree-based models.
- Fine-tuning RidgeCV enhanced performance and generalization.
- Overfitting remained a challenge, requiring careful parameter tuning.
- Future improvements:
  - Feature selection to reduce dimensionality.
  - Removing irrelevant or low-impact features.
  - Experimenting with polynomial features on a subset of features.