House Price Prediction Using Regression Models

Arunava Mukherjee (24BM6JP11)
Bhavya Arya (24BM6JP14)
Harshal Bhagwat Ugalmugle (24BM6JP20)
Pranjal Chakraborty (24BM6JP41)

April 20, 2025

1 Introduction

In this project, we aim to predict house prices using regression-based techniques. The dataset used is the Kaggle House Prices dataset (see references), which contains information on house sales in Ames, Iowa. The goal is to build a predictive model that estimates the `SalePrice` of a house based on various features such as location, size, quality, and condition etc.

2 Dataset Description

The dataset consists of 1460 observations and 81 variables. The target variable is `SalePrice`, a continuous numeric variable representing the final price of the house.

3 Methodology

3.1 Train-Test Split

We split the data into training and test sets using an 80-20 ratio. The training set was used to fit the models, while the test set was used to evaluate the performance.

3.2 Missing Value Imputation

We performed Imputation under the MAR (Missing at Random Assumption) using the following process:

- We removed features that have more than 40% missingness.
- We created a dataframe that has features with no missingness.
- We then added one by one the numerical variables to the completed dataframe and performed Linear Regression to impute the values. (Note we calculated 10 fold Cross-Validated RMSE to check whether imputation is acceptable)

- With the imputed numerical variables we append it to the completed dataframe and then start imputing categorical variables using Logistic Regression (checking 10 fold cross-validated accuracy for acceptable imputation; eg 0.50 accuracy is not acceptable for imputation)

3.3 Data Preprocessing

- Categorical variables were encoded using one-hot encoding.
- Numerical variables were standardized using StandardScaler.

3.4 Principle Component Analysis

We tried out PCA for dimensionality reduction but only 40-50% of variance in the data (only numerical columns) by the elbow obtained so the approach of PCA is dropped.

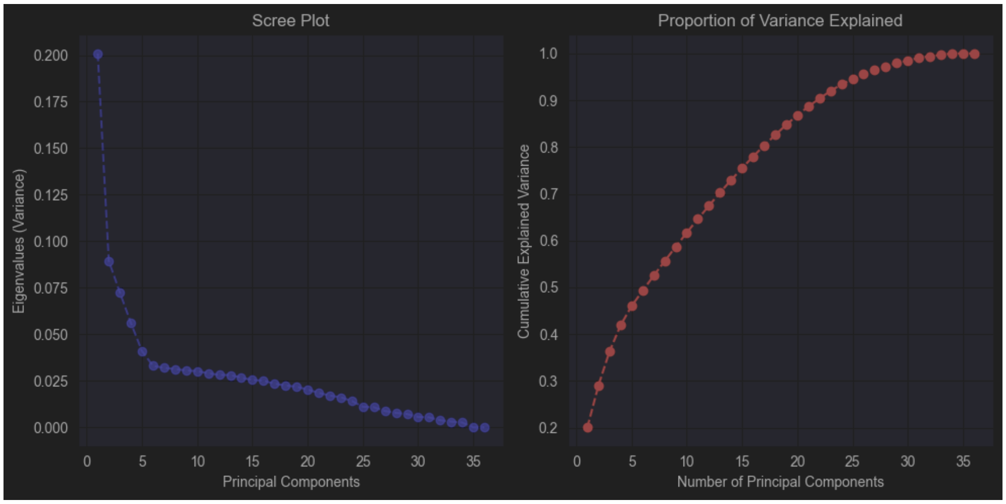

Figure 1: Principal Component Analysis

3.5 Multiple Linear Regression

We will proceed with Multiple Linear Regression as our attempts for PCA were spoiled but with the following three optimisations in our mind:

- We will first remove Multicollinearity between features by removing variables with VIF greater than 10.
- We will then perform regression and remove features that have p values greater than 0.05.
- We will then remove data points iteratively using Cook's Distance until R(square)adj does not decrease.

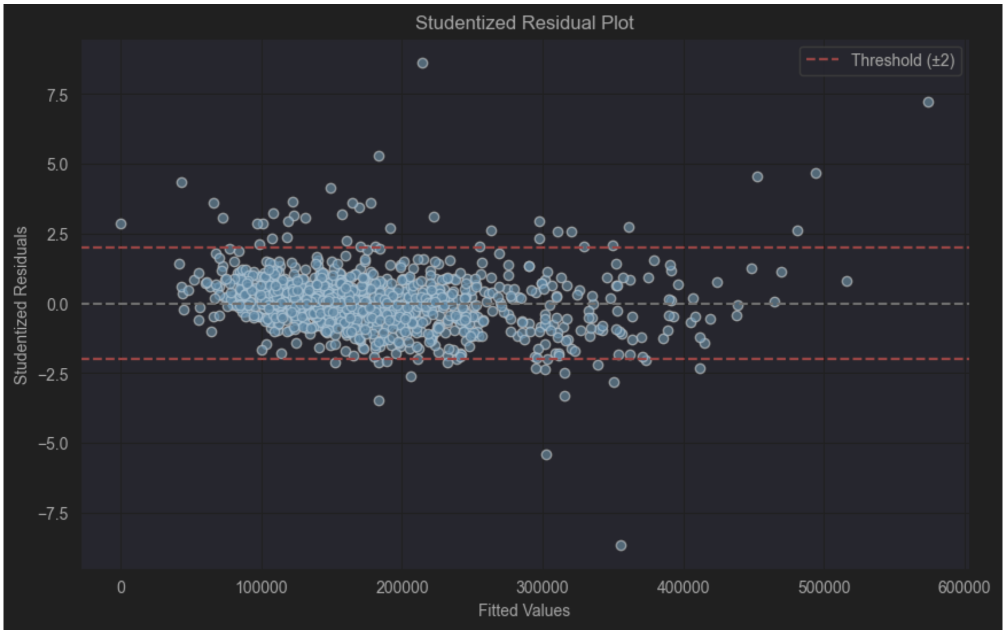

Figure 2: Residuals of Multiple Linear Regression

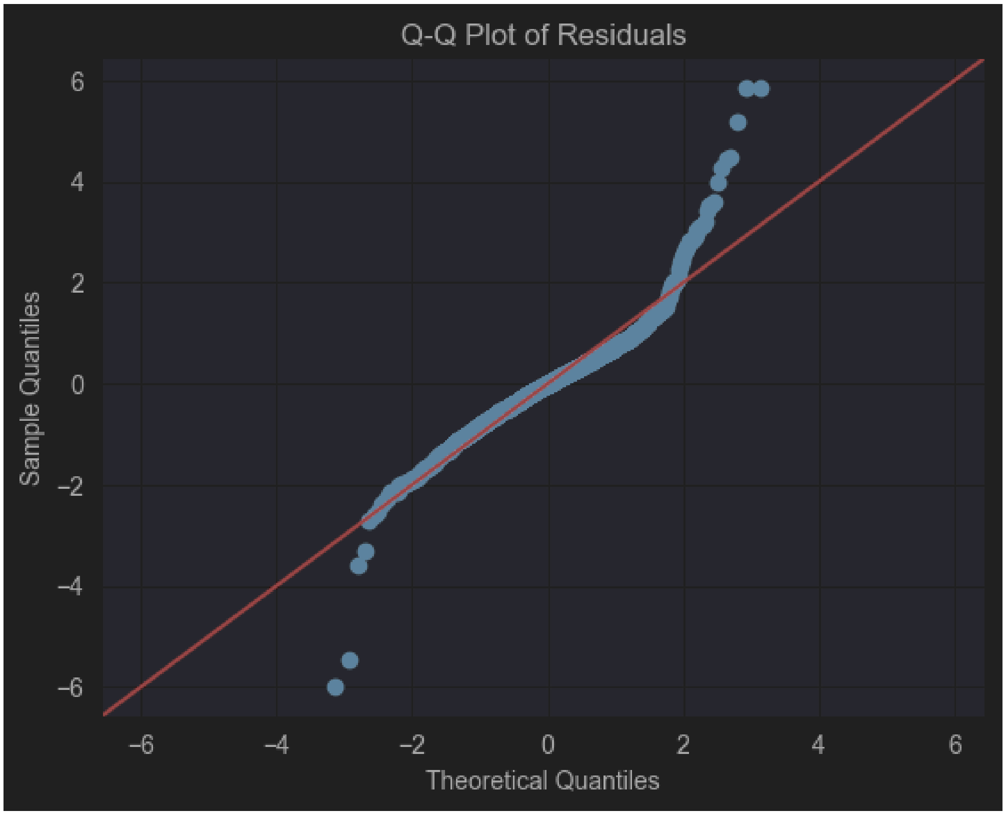

Figure 3: Q-Q plot of Residuals of Multiple Linear Regression

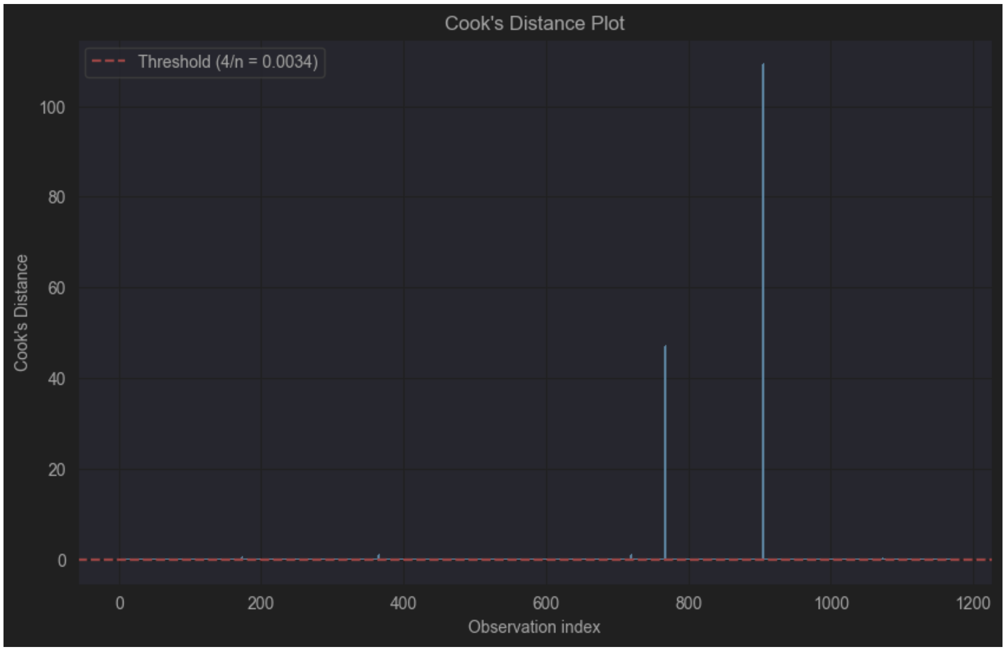

Figure 4: Cook's Distance Plot of Multiple Linear Regression

Model	Log RMSE (Train)	R^2 Score (Train)
MLR	0.979	0.977

Table 1: MLR Model Performance on Train Set

Model	Log RMSE (Test)	R^2 Score (Test)
MLR	0.239	0.815

Table 2: MLR Model Performance on Test Set

3.6 Important Comment 01

Although our MLR model performed greatly in the training dataset they performed abysmally in the test performance, this is because we have tried to maximize the performance in the train set based on the results of MLR regression table but when we are performing multiple linear regression our confidence of the p-values are $(0.95)^p$ resulting in very poor confidence of our regression results. To overcome this we need to perform F-tests but to know the absolute true association between features and response we need to perform 2^p regressions which is infeasible. As a result we will move on towards feature selection techniques like Forward Selection, Ridge Regression, Lasso Regression.

3.7 Forward Selection Procedure

We will iteratively choose the best single feature out of all the features (i.e. which gives lowest MSE) now keeping that obtained feature constant we will try out higher number of features each time selecting the best addition to the regressors. Essentially we are choosing greedily until I exhaust all the features. Then we will select the best set of Features for which the CV MSE was the lowest.

Model	Log RMSE (Test)	R^2 Score (Test)
Forward Selection	0.159	0.890

Table 3: Forward Selection Performance on Test Set

3.8 Ridge Regression

We will perform GRIDSEARCHCV using an alpha parameter space and then create a tolerance variable where after getting the best GRIDSEARCHCV Ridge regression model we will drop those features that have weights less than the tolerance we will then again perform GRIDSEARCHCV to get the solution. We will iterate until no features can be dropped.

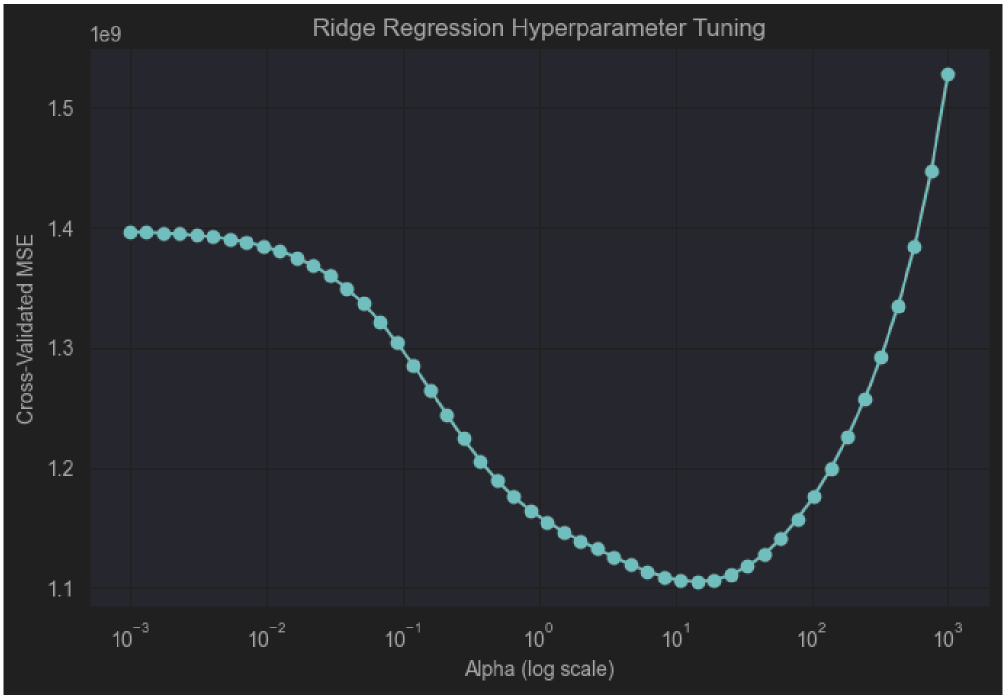

Figure 5: Loss function over the parameter space for one pass of Ridge Regression

Model	Log RMSE (Test)	R^2 Score (Test)
Ridge Regression	0.155	0.879

Table 4: Ridge Regression Model Performance on Test Set

3.9 Important Comment 02

The performance is a bit worse off than Forward Selection maybe it is picking up more noise features as it does not regularise features to 0.

3.10 Lasso Regression

The idea for Lasso Regression is same as for Ridge Regression where we will iteratively call GRIDSEARCHCV until no features are dropped. (One small change is that the Lasso inherently drops features and we do not need to define a tolerance variable as in the case of Ridge Regression)

Model	**Log RMSE (Test)**	**R^2 Score (Test)**
Lasso Regression	0.147	0.891

Table 5: Lasso Regression Model Performance on Test Set

4 Result Summary

Model	**Log RMSE (Test)**	**R^2 Score (Test)**
MLR	0.239	0.815
Forward Selection (best α)	0.159	0.890
Ridge Regression (best α)	0.155	0.879
Lasso Regression	0.147	0.891

Table 6: Model Performance Summary on Test Set

5 Conclusion

In Conclusion The Lasso Regression performs ever so slightly better than Forward Selection. Although Forward Selection has lower number of columns than Lasso Regression which allows for better Interpretability.

References

- Kaggle: House Prices - Advanced Regression Techniques
  `https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques`