

# House Prices Predictions: Advanced Regression Techniques

Author: **Mhlengeni Miya (363729)**, Group Members: Thabani Jali (1876297) and George Marantos (2116287)  
School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg, 2050, South Africa  
ELEN4025 – Introduction to Machine Learning  
02/06/2023

**Abstract**—This report presents a comprehensive analysis of developing a regression model for housing price prediction, focusing on data preprocessing, feature engineering, model selection, and ensemble modeling. The primary objective was to achieve accurate predictions and a low RMSE on the Kaggle leaderboard. Thorough data cleaning and feature engineering techniques were applied to improve data quality and capture complex relationships. A wide range of regression models, including both linear and tree-based algorithms, were evaluated based on performance metrics such as R-squared scores. The top-performing models, including Ridge, Lasso, CatBoost, and XGBoost, were combined in an ensemble approach to leverage their individual strengths. The ensemble model effectively reduced biases and errors, resulting in a remarkable RMSE and a high position on the Kaggle leaderboard. The success of the project can be attributed to meticulous data preprocessing, insightful feature engineering, and the thoughtful ensemble modeling strategy, emphasizing the importance of data quality and leveraging diverse models for regression tasks.

**Index Terms**—House prices, SalePrice, Linear Regression, Ensemble Models

## I. INTRODUCTION

This report documents the design, implementation, and analysis of a Machine Learning (ML) solution to a Kaggle competition [1]. This Kaggle competition requires the prediction of the final price of each home in Ames, Iowa, using 79 explanatory variables/features describing every aspect of the home [1]. The aim of this project is to predict the prices of the home using Machine Learning models. The predictions are achieved by training multiple models on the data and then blending them to perform the final prediction. This method of prediction is chosen because it takes advantage of strengths from different models and gives better predictions.

## II. DATA EXPLORATION AND CLEANING

Section discusses the data exploration process performed on the dataset for predicting house prices in Ames, Iowa. This section was carried out by all group members.

### A. Relationship between features and the target variable

The dataset consists of 79 variables, and the main objective is to gain insights into the relationships between these variables and the target variable, "SalePrice". To begin with, a log transformation is applied to the target variable,

"SalePrice". Log transformations are commonly used when dealing with skewed distributions, as they can help normalize the data and make it more suitable for modeling. This transformation can also alleviate the impact of extreme values on the analysis. After transformation, the "SalePrice" is distributed normally.

After the log transformation, the numerical variables are analyzed in the dataset. Among these variables, four stood out due to their similar distribution to the transformed "SalePrice" variable. These variables are "Lotfrontage", "TotalBsmtSF", "1stFlrSF", and "GrLivArea". The similarity in distribution indicates a potential linear relationship between these features and house prices.

Correlation between the features and the 'SalePrice' was evaluated, and it was observed that the features "3Ssn Porch", "BsmtFin SF 2", "Low Qual Fin SF", "Misc Val", and "Pool Area" have no correlation with the target variable, "SalePrice", in the analysis. This indicates that the size of the three-season porch, the finished square footage of basement type 2, the low-quality finished square footage, the value of miscellaneous features, and the area of the pool do not significantly impact the determination of the house sale price. While they may still have non-linear or complex relationships, their linear correlation with the target variable is negligible.

The features "OverallQual", "GrLivArea", "TotalBsmtSF", and "1stFlrSF" display a strong correlation with the target variable, "SalePrice". This indicates that the overall quality of the house, above-ground living area, total basement area, and first-floor area have a significant linear relationship with the house sale price. Higher overall quality, larger living areas, and increased basement and first-floor sizes tend to correspond to higher sale prices. These features play a crucial role in determining the house sale price and provide valuable insights into the housing market dynamics in Ames, Iowa. These correlations correspond to the observations made on the plots.

## B. Data preprocessing

Preprocessing the dataset before developing a model is an important step in machine learning. It involves transforming and cleaning the data to improve the performance and accuracy of the models [2]. Each feature in the dataset is analyzed using the `analyse_feature()` function, this helps understand the characteristics of each feature, its relationship with the target variable, and any discrepancies between the training and test data.

1) *Handling Missing Values:* Handling missing data is crucial in data analysis and machine learning for several reasons. Firstly, missing data can introduce bias and affect the quality and reliability of the analysis or model [2]. Ignoring missing values may lead to inaccurate conclusions or biased predictions [2]. By addressing missing data, we aim to minimize the impact of missing entries and ensure the integrity of the analysis.

Secondly, missing data can disrupt the computations and algorithms used in data analysis and modeling. Many machine learning algorithms cannot handle missing values directly, and they require complete datasets for training and prediction. Therefore, filling in missing values allows us to utilize the complete dataset and make the most effective use of the available information.

Handling missing data is carried out in both the training and test datasets. Handling missing entries in both the train and test datasets increases consistency, fairness, and real-world application. It ensures that the model is trained and evaluated on datasets with identical preparation methods, eliminating potential biases and enhancing the model's capacity to handle missing values in unseen data.

Various strategies have been employed to fill the missing values. After analyzing each feature using `analyse_feature()`, the missing values in that feature are filled based on the analysis. For example, missing entries in the "MSZoning" column in "test\_b" are filled with the value "C (all)", this is because the average sale price for properties with "MSZoning" classified as "C (all)" is the lowest among all the categories, and we assume that the missing values are more likely to belong to properties with lower sale prices.

Similarly, missing entries in the "Functional" and "SaleType" columns are filled with "Typ" and "WD" respectively. The missing entries in the "Fence" column in "data\_b" and "test\_b" is filled with "None", indicating the absence of a fence. Other categorical columns like "Electrical", "FireplaceQu", "GarageType", "GarageQual", "GarageCond", "GarageFinish", "MasVnrType", "BsmtQual", "BsmtCond", "BsmtExposure", and "Utilities" are also filled with appropriate values based on domain knowledge or by

considering "None" as a reasonable default value.

2) *Fixing Outliers:* Outliers are extreme values that deviate significantly from the normal pattern of the data and can adversely affect the analysis and modeling process [2]. Fixing outliers is important in data preprocessing for several reasons. Outliers can distort statistical properties, bias analysis, and hinder model performance [2]. They violate assumptions and indicate data quality issues. By addressing outliers, we ensure reliable analysis, build robust models, adhere to assumptions, improve data quality, and gain meaningful insights. It allows for more accurate results, better generalization, and informed decision-making based on a representative dataset.

After analyzing the data, it is identified that some features are prone to outliers, such as "GarageYrBlt," "LotArea," "1stFlrSF," "GrLivArea," "TotRmsAbvGrd," "BsmtFinSF1," "TotalBsmtSF," "GarageCars," and "GarageArea," and correction techniques to address these outliers are applied. For instance, in the "GarageYrBlt" column, there is an entry with a value of 2207. This value is unlikely to be accurate, so it is replaced with 2007, which is a more reasonable assumption.

Similarly, for "LotArea," "LotFrontage," "1stFlrSF," "GrLivArea," "TotRmsAbvGrd," "BsmtFinSF1," "TotalBsmtSF," "GarageCars," and "GarageArea," values exceeding certain thresholds are adjusted to more reasonable limits. This helps in ensuring that the dataset remains consistent and suitable for subsequent analysis and modeling tasks. Additionally, the missing values in the "LotFrontage" column are handled by utilizing a simple linear regression approach. The missing values in this column are estimated based on the relationship between "LotArea" and "LotFrontage" using a linear regression equation. This enables the missing values to be filled with reasonable estimates, enhancing the completeness of the dataset for further analysis.

3) *Feature Extraction:* Feature extraction is the method used to create new features based on the existing features in the dataset [2]. This helps capture additional information and potentially improve the predictive power of the data. Firstly, the "Age" feature is extracted by subtracting the "YearBuilt" column from 2011. This represents the age of the house at the time of the dataset. Similarly, the "RemodAfter" feature is computed by subtracting the "YearBuilt" from the "YearRemodAdd" column, indicating the number of years that elapsed between remodeling and the construction of the house. The "AgeRemodAdd" feature represents the age of the house at the time of remodeling. Next, the "Age\_Sold" feature is calculated by subtracting the "YearBuilt" from the "YrSold" column, indicating the age of the house at the time of sale. The "Sold\_before" feature represents the number of years before the dataset's

reference year (2011) that the house was sold.

Furthermore, we compute the "Age\_Garage" feature by subtracting the "GarageYrBlt" from 2011, representing the age of the garage at the time of the dataset. The "Age\_Garage\_Sold" feature represents the number of years between the garage's construction and the house's sale. Lastly, we create the "TotalPorch" feature by summing up the porch-related columns, including "EnclosedPorch," "OpenPorchSF," "ScreenPorch," and "3SsnPorch." This provides a combined measure of the total porch area. The "TotalBath" feature is computed by summing up the bathroom-related columns, including "FullBath," "HalfBath," "Bsmt-FullBath," and "BsmtHalfBath," representing the total number of bathrooms in the house. By incorporating these additional features, the dataset is enhanced with more meaningful information, potentially improving the performance of future analysis and modeling tasks. The features with missing entries or are insignificant according to the analysis, are dropped, this reduces the dimensionality of the dataset and focuses on the most relevant features for the subsequent modeling tasks.

### III. EXPERIMENTS

This section was my individual work.

#### A. Cross-validation

The approach taken for selecting the best models was to train eight different models and assess their performances and then decide which ones to use based on the results. The models that were trained are Elastic Net (ENet), Ridge, Lasso, KernelRidge (KRR), LightGBM (LGBM), XGBoost (XGB), CatBoost (CatB), and Gradient Boosting Regressor (GBR) regression models. The models are trained using k-fold cross-validation, specifically with "cv=10". This means that the data is split into 10 equal-sized folds, and the model is trained and evaluated 10 times. In each iteration, one fold is used as the validation set, and the remaining nine folds are used as the training set. This process is repeated 10 times, with each fold serving as the validation set exactly once. The performance scores from each iteration are then averaged to obtain the overall performance estimate. Table I shows the performance of each model.

TABLE I  
PERFORMANCES OF REGRESSION MODELS

Regression Models Performance	
Model	Average Score
Elastic Net (ENet)	0.907747969310624
Ridge	0.9030237016062781
Lasso	0.9056746316323258
KernelRidge (KRR)	0.895561030177632
LightGBM (LGBM)	0.8998234725965346
XGBoost (XGB)	0.9112048594146847
CatBoost (CatB)	0.9145825780429611
Gradient Boosting Regressor (GBR)	0.9075380942665318

A higher score represents better performance. Based on the average scores in Table I, CatBoost (CatB) shows the highest performance, followed by XGBoost (XGB) and ElasticNet (ENet).

#### B. Train and validation errors

Cross-validation is a technique used in machine learning to assess the performance and generalization ability of a model. To further evaluate the performances of the models, the training error and validation error for each model are computed using mean squared error (MSE) as the evaluation metric. For each model, the model is fitted to the training data and makes predictions on both the training and testing data. The mean squared error (MSE) is calculated between the true target values and the predicted values for both the training and testing sets.

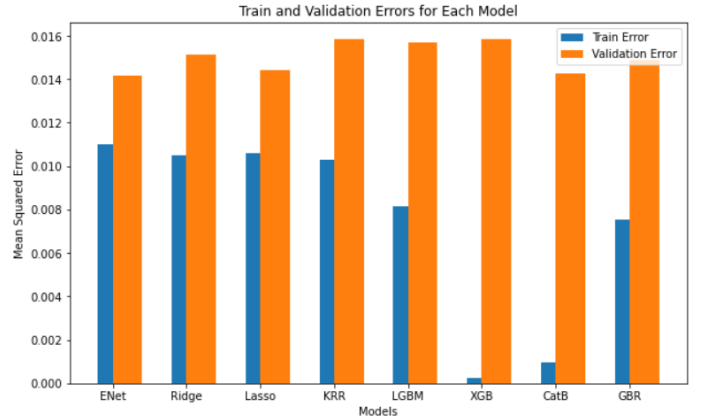


Fig. 1. Train and validation errors for each model.

These errors represented by the bar plot in Figure 1 provide an indication of how well each model performs on the training and validation sets. Lower values indicate better performance, as they represent smaller differences between predicted and true values. XGBoost and CatBoost models have the lowest validation errors, indicating better performance on unseen data.

#### C. Train and validation accuracy

Train and validation accuracy are crucial metrics for evaluating the performance of machine learning models. The train accuracy reflects how well the model fits the training data. A high train accuracy suggests that the model has successfully learned the patterns and relationships present in the training set. However, excessively high train accuracy can be an indication of overfitting, where the model becomes too specialized to the training data and fails to generalize well to new, unseen data.

Validation accuracy, on the other hand, measures the model's performance on a separate validation dataset that simulates real-world scenarios. It provides an estimate of

how well the model can generalize to new examples. A high validation accuracy implies that the model is capable of effectively applying the learned patterns to previously unseen data. When the validation accuracy is similar to the train accuracy, it suggests that the model has achieved a good balance between capturing the training data's patterns and generalizing to new instances.

Comparing the train and validation accuracy is essential for assessing the model's behavior. A significant disparity between the two metrics can provide insights into potential issues. If the train accuracy is much higher than the validation accuracy, it indicates overfitting, where the model has memorized the training data but struggles to generalize. Conversely, if the validation accuracy is substantially lower than the train accuracy, it suggests underfitting, meaning the model is too simplistic and fails to capture the underlying patterns. Striking a balance between high train accuracy and reasonable validation accuracy is crucial to ensure that the model learns meaningful patterns from the training data while being able to generalize effectively to new, unseen data.

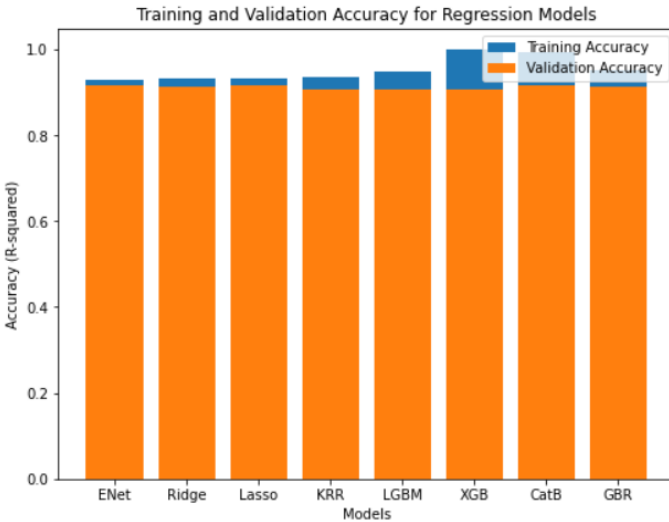


Fig. 2. Train and validation accuracy for each model.

Figure 2 shows train and validation bar plots. The models generally perform well, with  $R^2$  scores ranging from approximately 0.91 to 0.99. Kernel Ridge Regression (KRR), LGBM, CatBoost (CatB), and Gradient Boosting (GBR) models consistently show good performance across both the training and validation sets. XGBoost (XGB) shows extremely high accuracy on the training set ( $R^2$  of 0.998) but a lower score on the validation set ( $R^2$  of 0.907). This suggests that the model may be overfitting to the training data.

#### IV. RESULTS

From the experiment results in Section III, we select Ridge Regression, Lasso Regression, CatBoostRegressor, and XGBoostRegressor as the models to make the

final predictions of the "SalePrice". The models are combined, and their logarithmic predictions are averaged and converted back to the original scale. The resulting predictions are assigned to the 'SalePrice' column of a submission Dataset, which is saved as a CSV file named "submission.csv". This function allows for the combination of tree-based and linear models, providing an ensemble approach to prediction generation for submission.

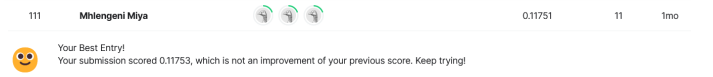


Fig. 3. Leaderboard place.

Figure 3 shows the leaderboard position on Kaggle, this is the top 2.3% position. The achieved high position and low RMSE on the Kaggle leaderboard can be attributed to effective data cleaning, thoughtful feature engineering, and the decision to combine four models. These steps collectively enhanced the models' ability to capture relevant patterns and relationships in the data, resulting in accurate predictions. The combination of tree-based and linear models allowed for a more comprehensive and robust modeling approach. The success of this solution highlights the importance of thorough data preprocessing, feature engineering, and leveraging the strengths of different models in machine learning. The reflection on group work is presented in Appendix A, and a print of submissions on Kaggle is shown in Appendix B.

#### V. CONCLUSION

This report details the development of a regression model for housing price prediction, focusing on data preprocessing, feature engineering, model selection, and ensemble modeling. By addressing missing values, outliers, and inconsistencies in the dataset and creating informative features, the models were able to capture complex relationships and improve predictive accuracy. The evaluation of various linear and tree-based models resulted in the selection of the four best-performing models, which were then combined using an ensemble approach. This strategic combination of models enhanced accuracy and stability, leading to a low RMSE and a high position on the Kaggle leaderboard. The success of this project underscores the importance of data quality, feature engineering, and leveraging diverse models in achieving accurate predictions for regression tasks.

#### REFERENCES

- [1] A. Montoya. "House Prices - Advanced Regression Techniques.", 2016. URL <https://kaggle.com/competitions/house-prices-advanced-regression-techniques>.
- [2] P. Deitel and H. Deitel. *Intro to Python for Computer Science and Data Science*. United Kingdom: Pearson Education Limited, first ed., 2022.

## APPENDIX A

### *A. Reflection on the team's work*

Working on this project as a group of three individuals provided valuable insights into the benefits and challenges of collaborative work. One of the key takeaways from our group experience was the advantage of having diverse perspectives and expertise. Each team member brought their unique skills and background to the project, enriching our discussions and decision-making process. The diversity within the group allowed us to approach problems from multiple angles, fostering innovation and creative problem-solving.

Effective communication and coordination were crucial aspects of our group work. We recognized that clear and open communication was essential for maintaining a shared understanding and ensuring everyone was aligned with project goals. Regular team meetings and discussions allowed us to exchange ideas, provide progress updates, and assign tasks. By actively engaging in communication, we were able to leverage each other's strengths and optimize our collective output.

Dividing tasks and responsibilities among team members proved to be an effective strategy. This approach enabled us to make efficient use of our individual skills and expertise. By assigning specific roles based on our strengths and interests, we fostered a sense of ownership and accountability within the group. Each team member took responsibility for their assigned tasks, contributing to the overall progress of the project. This division of tasks also reduced duplication of efforts and ensured that all aspects of the project were adequately covered.

Moreover, our group experience facilitated a collaborative learning environment. We embraced the opportunity to learn from each other's experiences and knowledge. Sharing coding techniques, analytical approaches, and problem-solving strategies allowed us to expand our individual skill sets. The collective learning enhanced our capabilities as a team and contributed to the overall success of the project. Through collaboration, we were able to leverage the strengths of each team member, fostering a culture of continuous learning and growth.

In summary, the group project provided valuable insights into the benefits of collaboration. We discovered the power of diverse perspectives, effective communication, coordinated task management, and collaborative learning. These takeaways have equipped us with valuable skills and knowledge that we can apply to future group projects, enabling us to leverage the collective intelligence of the team for greater success.

## APPENDIX B









	<b>submission.csv</b> Complete · Mhlengeni Miya · 1mo ago	<b>0.11753</b>
	<b>submission.csv</b> Complete · Mhlengeni Miya · 1mo ago	<b>0.11751</b>
	<b>KNNSubmission.csv</b> Complete · George Marantos · 1mo ago	<b>0.22403</b>
	<b>submission_RFR.csv</b> Complete · Thabani melusi · 1mo ago	<b>0.14276</b>
	<b>submission_RFR.csv</b> Complete · Thabani melusi · 1mo ago	<b>0.16869</b>
	<b>submission_RFR.csv</b> Complete · Thabani melusi · 1mo ago	<b>0.14732</b>
	<b>KNNSubmission.csv</b> Complete · George Marantos · 1mo ago	<b>0.49128</b>
	<b>KNNSubmission.csv</b>	

Fig. 4. Submissions on Kaggle.