Predicting Home Prices in Ames, Iowa Using Advanced Regression Techniques

Team: Lars Bosgraaf, Bill Sendewicz, Stas Sochynskyi

Introduction

Residential real estate prices are based on dozens, if not hundreds, of factors, including dwelling characteristics (e.g., number of bedrooms, area), characteristics about the surrounding property (e.g., lot size, the presence of other structures), and location characteristics (e.g., neighborhood and distance to public transit). A tool that can predict sale prices based on a large number of factors can benefit both home buyers and real estate agents.

Aim

Our aim is to explore the given dataset and build the best possible regression models to predict the sale prices of houses in Ames, Iowa. Additionally, we had the goal of placing in the top 20 of the House Prices: Advanced Regression Techniques Kaggle competition on Kaggle.

Data & Methodology

The dataset for this project is the Ames, Iowa housing dataset detailing houses that were sold in Ames between 2006 and 2010. It includes 81 variables (35 numeric, 23 categorical and 23 ordinal variables) and 2919 observations, and was already split into separate train and test datasets (1460 and 1459 observations, respectively).

Based on the high correlation matrix *(Figure 1)* 9 columns were dropped (e.g., *id, GarageCars, GarageCond*).

In the dataset, all the NAs were filled either with mean value or 0. Additional feature called “NeighborhoodClass” were created using K-means clustering with values from 0 to 2 *(Figure 2)*. It represents cluster where house is situated.

Logarithms of six variables (e.g., *SalePrice, LotArea)* that appeared to be lognormally distributed were used to reduce the effect of outliers and improve the model.

All the rest 37 categorical variables were one-hot-encoded.

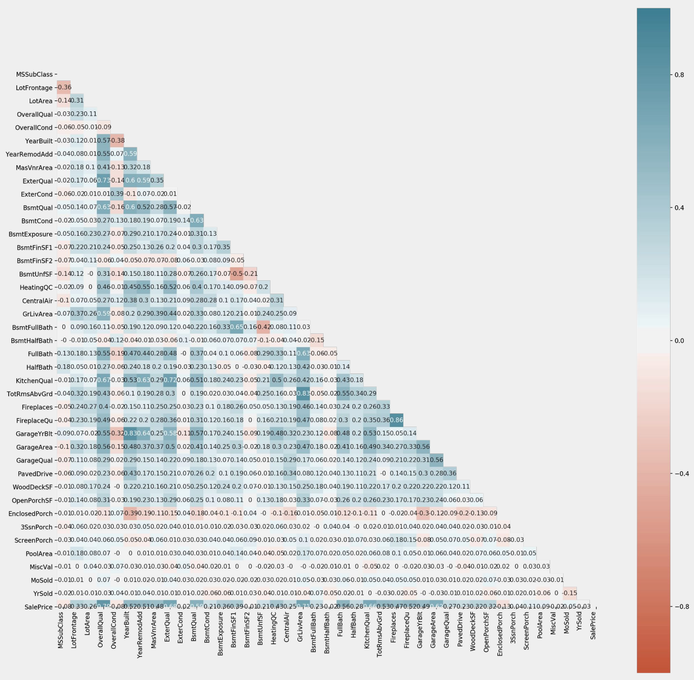

Figure 1. Correlation matrix

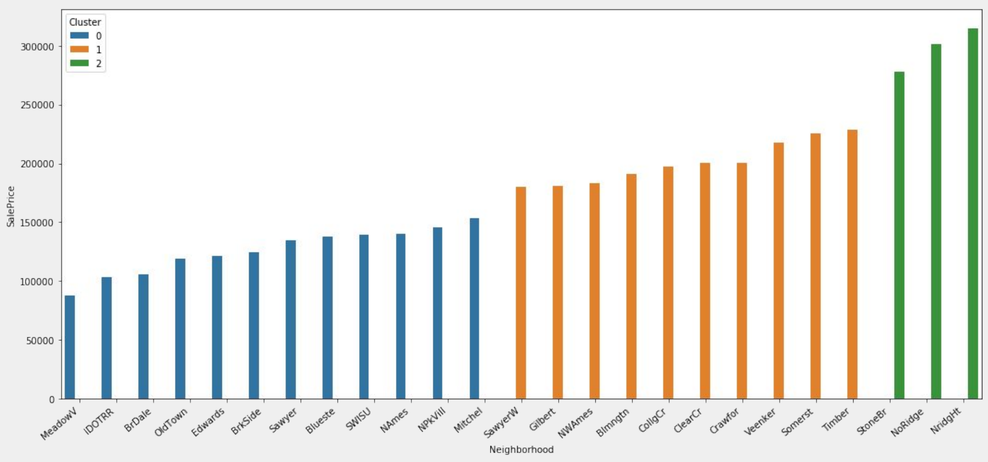

Figure 2. Clustered neighborhoods

Principal component regression (PCR) method was applied on the cleaned dataset to reduce the number of features. Many of one-hot-encoded features had standard deviation near 0 thus the results weren’t meaningful on the entire dataset. Even after dropping the one-hot-encoded features, there was no obvious subset of features that could be dropped from the model.

Following these changes, our datasets consisted of 243 numerical features and one label, *SalePrice*, on 1456 observations in train dataset (no observations were removed from the test dataset).

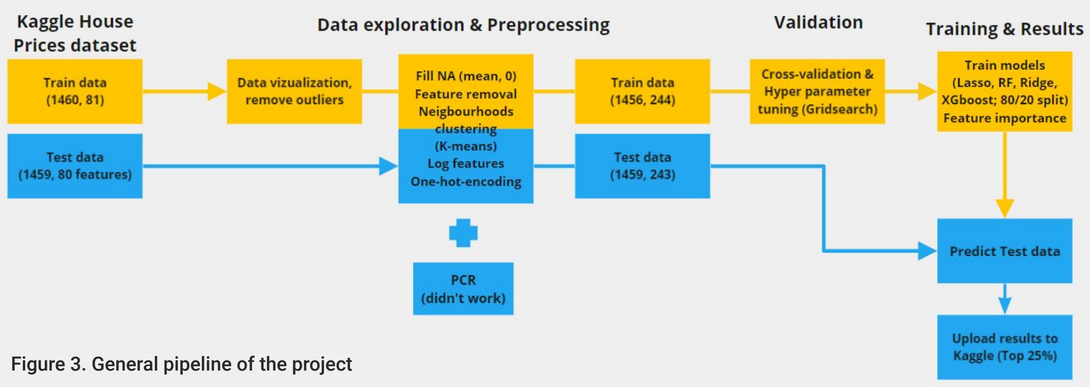

Figure 3. General pipeline of the project

Results

Our main goals were twofold: find the best-performing model with any number of features, and then find the best-performing model with a reduced number of features, 10-20 in total.

For the validation of different models we used 5-fold cross validation data, which creates 80/20 splits in the data. Variance for each feature/model combination was below 0.4%.

Feature amount	1	5	10	50	70	100	200	243
Model								
Lasso	0.39649	0.39649	0.31711	0.27725	0.27725	0.27725	0.27725	0.27725
RandomForest	0.31446	0.16160	0.13933	0.13894	0.13891	0.13884	0.13932	0.13926
Ridge	0.27010	0.16159	0.13381	0.11527	0.11475	0.11564	0.11725	0.11839
XGboost	0.28016	0.15224	0.12689	0.11947	0.11971	0.11956	0.11966	0.11992
XGboost-Ridge ensemble	0.27284	0.15188	0.12519	0.11379	0.11354	0.11394	0.11490	0.11553
XGboost-Ridge ensemble custom	0.27273	0.15211	0.12569	0.11360	0.11393	0.11429	0.11546	0.11631

Figure 4. Average model cross-validation RMSLE scores.

During the validation process, Lasso regression showed the worst result. It has the same errors for all versions of the model with different amounts of features. Another finding was that the random forest model with very few features also performed poorly. RFs with lots of features performed much better.

For hyperparameter tuning GridSearchCV function was used from Scikit-learn. For every hyperparameter in the model, input a list of possible parameter values, and the function returns the best possible combination of parameters.

After the cross-validation process it was clear that the Ridge and XGboost models performed best. A visual analysis of the model predictions compared to the ‘True’ values revealed that the XGboost model was better at predicting large values and the Ridge regression was predicting the low values better. XGboost and Ridge regression models were combined to form an ensemble model that gives more weight to the Ridge predictions for sale prices below 200.000 dollars, and more weight to XGboost predictions for sale prices above 200.000 dollars.

Another problem was that both models consistently overpredicted low and underpredicted high housing prices. The custom ensemble model also introduces an extra penalty for predictions that have a very low value and increases the value of predictions that have a high value.

The initial ensemble model outperformed both the individual Ridge and XGboost regression models. Adding the extra custom penalties to the ensemble model did not improve the RMSLE score of the ensemble model.

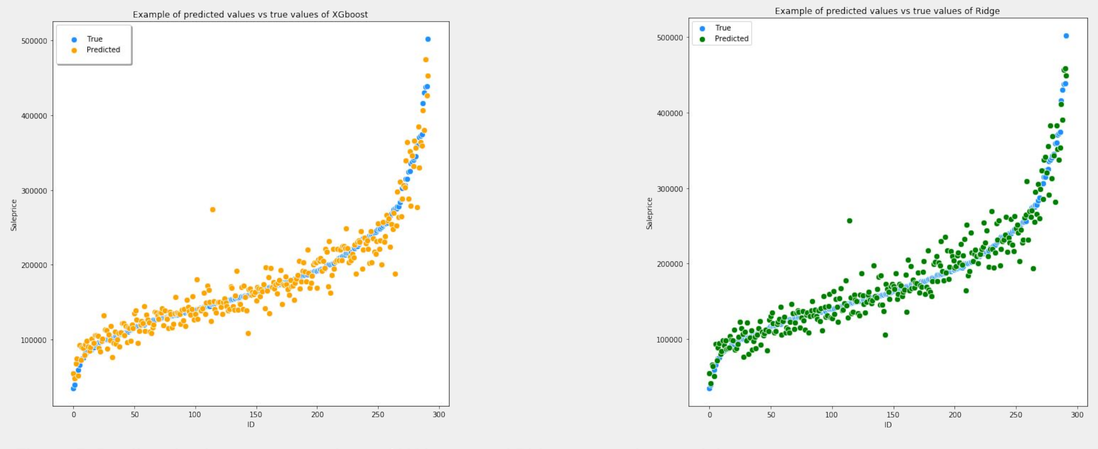

Figure 5. Predicted values vs. true values of XGboost and Ridge regression models

Feature amount	10	50	70	100	243
Model					
Ridge	0.14278	0.12605	0.12654	0.12635	0.12750
XGboost	0.13491	0.12241	0.12319	0.12246	0.12428
Ridge-XGboost ensemble	0.14393	0.12342	0.12382	0.12267	0.12291
Ridge-XGboost ensemble custom	0.14485	0.12359	0.12526	0.12467	0.12441

Figure 6. RMSLE scores of predictions on test data

Conclusions

1. Ridge and XGboost regression models vastly outperformed RF and Lasso models
2. When there are very few observations removing outliers can worsen prediction accuracy.
3. Predicting extreme values can be problematic and may require alternative approaches.
4. Our team was in the Top 25% of the overall Kaggle leaderboard
5. The 10-feature Ridge regression model with the lowest RMSLE (0.13381) is:

SalePrice = -1.258 + 0.388**GrLivArea* + 0.091**OverallQual* + 0.017**GarageArea* + 0.071**N_Class* + 0.017**BsmtFinSF1* + 0.041**Fireplaces* + 0.120**LotArea* + 0.002**YearRemodAdd* + 0.003**YearBuilt*+0.043**OverallCond*