Caroline Schwengler and Jakob Garcia
CSC 480
Final Project Report
Kaggle Competition: House Prices - Advanced Regression Techniques | Kaggle

Introduction

The competition we selected to work with is called 'House Prices - Advanced Regression Techniques'. This competition is about using machine learning techniques to help identify what factors and variables influence the final price of homes in Ames, Iowa. This competition uses the Root-Mean-Square-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price. This is similar to the MSE we have covered in class but it is specifically the root of that value. The logarithm is taken of the predicted value and the observed sales price to help address scaling and ensure that the classification of cheaper homes is as important as more expensive homes. This problem benefits everyone who is interested in the housing market or seeking to buy a home (specifically in Ames, Iowa or in similar areas). This dataset and effective algorithms can help these people identify which factors influence the prices of homes and juxtapose this with their own desires to find a valuation for a home that they deem is reasonable. It can also help to identify when a house on the market with certain traits is overvalued. This can save a lot of time, labor, and money for people seeking to buy homes because they will not have to rely on a realtor with this background information (information that may not even be rooted in data).

Background

The task is to use a selection of the 79 explanatory variables to predict the observed sales price. The dataset includes a lot of variables that you might expect but also some that are a bit more unusual like the general shape of the property. There are a lot of categorical, continuous, and binary variables in the dataset. Key variables likely to impact sale price include OverallQual (overall material and finish quality), GrLivArea (above-ground living area square feet), and TotalBsmtSF (total basement square feet), as these directly influence the size, usability, and perceived value of the home. Additionally, Neighborhood, YearBuilt, and GarageCars are important since location, the age of the home, and parking capacity can significantly affect market demand and pricing. The data is already split into the training and test set but we further split the training set using cross validation so that we can have validation data.

Our approaches

Data preprocessing

We used a StandardScalar to standardize and normalize the data. This will be necessary to use ML techniques like LASSO but will likely be helpful and certainly not harmful for the other techniques we will be using. Random forest requires very minimal preprocessing and since nothing is strictly required we can test a variety of transformations and see which ones perform

the best on our validation data. Some of the variables are text based, and so we needed to do transformations such as one-hot encoding in order for it to work with our regressions. The data set does have a lot of continuous variables as well as categorical variables (typically a ranking from 1-10) and balancing these so that they are equally important to the algorithm when making predictions will also require preprocessing. In addition to these preprocessing steps we tried out feature selection techniques, namely principal component analysis, and data imputation for missing values.

Feature Selection

Many of the previous submissions for this competition did some feature engineering, oftentimes with the stepwise feature selection but sometimes other feature selection algorithms. In the end we decided to do **Principal Component Analysis (PCA)** for our method of feature reweighting and selection. This was the first novel feature engineering step we took for our project. Initially, before doing extensive hyperparameter tuning, improved a basic support vector regression's performance from a root mean square error (RMSE) of about 0.198 to 0.196. Considering our outcome is log scaled this was actually a fairly large improvement. We had high hopes for this technique based on this model and everything we had learned in class but oddly after doing more extensive hyperparameter training, PCA resulted in a worse score for every single base model, including the support vector. Oftentimes the score was not much lower though, and since PCA does help reduce dimensionality and computational complexity we determined it could still be useful for this project.

Two of the models we used for this project, random forest and extreme gradient boosting, both work by training many different trees. Unfortunately, we were unable to hyperparameter tune values for these models after PCA because they took an unreasonable amount of time to run. It would work and was able to get through a few iterations but we didn't have the resources to allow them to run for hours like they would've needed. Instead, for these two models, we had to use a lot of the basic parameters. These models still achieved good results but it is important to note that with better computing power and the ability to hyperparameter tune, their results could improve further.

Data Imputation

Nearly everyone who received a moderately good score in the past on this competition and shared their process in the discussion on kaggle, was doing some amount of imputation to handle the missing values in the dataset. There are many different ways to handle imputation but we focus solely on imputing the numerical values. There are different techniques like imputing based on the mean but no matter what we will be adding noise and bias as we impute. We closely examined all missing values in this dataset and many were data input errors that could be replaced with zeros, but a few columns proved to be more challenging. For houses that seemed to have correct data in all regards except one missing value in an essential column (like one house was missing an electrical value but was complete in all other regards) we concluded that the data

was missing on accident and we decided to impute in these cases. When imputing features like electrical that are installed in a house upon construction, we decided to impute based on the neighborhood because it is highly likely that all the homes in any given neighborhood were constructed by the same builders in a similar time frame and would have been constructed the same way. In these cases we imputed using a majority vote classifier for that column in the same neighborhood as the house missing a value. Some columns that were particularly difficult to impute were the year built for the garage and the lot frontage size.

It was extremely difficult to come up with what could be an accurate predictor for the year the garage was built in cases where this value was missing, even for houses without garages because imputing a 0 would be unreasonable. In this case we decided to remove the garage year built column entirely because there were many other columns relating to garages. These other columns had high correlation with garage year built and even higher correlations with the outcome, sales price. This implied to us that most of the important predictive information in the garage year built was accounted for in the other garage columns.

For the lot frontage size we know that every house must have some amount of frontage. So, for this column in particular we decided to predict and **impute the missing values by training a model**, specifically in this case we used an XGB model. The linear regression model was fitted with all other predictors in the data set. This improved the score for all but one of our models (see ablation study below) and is definitely a novel way of performing feature engineering on this dataset.

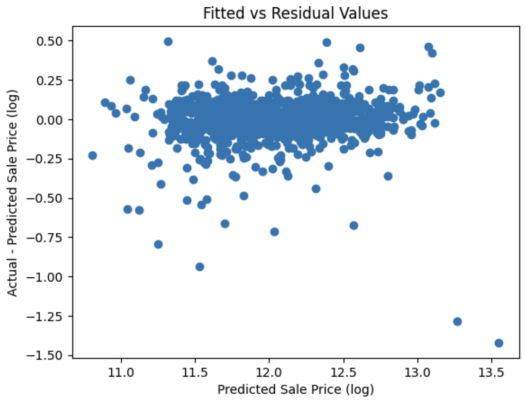
Models

The machine learning algorithms that we used are LASSO, Ridge regression, the regression variation of the support vector machine called support vector regression (SVR), an elastic net, a random forest, and an extreme gradient boosting (XGB) model. We predicted LASSO would be useful for this competition because it allows us to do a form of feature weighting and selection which became necessary because of the large amount of features in this dataset (especially after one-hot encoding). We also predicted Ridge and SVR would be useful if data violates some of the assumptions of the linear data like multicollinearity or if the data itself is not linear. Random forest seemed like a very cool technique that we are both excited to try to implement but also has performed well for this competition in the past. Using a random forest algorithm will allow us to learn very complex relationships in the data that can help us predict the price of houses. We used bayes search to train the hyperparameters for all of these models; it was fast and effective in finding the optimal values. After our first pass through the project the elastic net was performing the best on its own but then after discovering the XGB model it skyrocketed to be the best. So, XGB was our best standalone model with a RMSE of 0.125216, and after more hyperparameter tuning, the support vector regression was fairly close behind with a RMSE of 0.127325 before PCA.

Additionally, we tried combinations of the models mentioned above as a novel algorithm for this data. We figured that each of these models has its own way of analyzing the data and is

likely using different features and metrics to predict the sale price. Therefore, it seemed reasonable to us that finding a way to combine models would improve our predictive abilities. So, after training several models we also tested the **average prediction for several models**, and it performed quite well.

After running all of the steps mentioned above (particularly feature selection with PCA, imputing lot frontage using XGB, and averaging several models to get the best performance), our best model that combined all these steps, was a model which took the averaged result of a LASSO model after PCA and a support vector after PCA and then averaged this result with XGB and support vector (before PCA) for a final root mean square error of 0.124863. For better or for worse though, all of these unique steps combined were not the model we used to achieve our best results overall. None of the models that were trained with principal component analysis contributed to our best model but this was still a valuable and novel way of feature engineering as evidenced by the fact that one of our five best models relied on models that had used PCA. Our overall best model was a combination of the support vector regression and the extreme gradient boosting model which was able to achieve an RMSE of 0.121653 as a cross validation score using the training set. **After fine-tuning this model, it obtained a root mean squared error of 0.12042 on the test set which earned us rank of 318 out of 4675 in the competition overall (as of writing this report).**



Above are two plots that demonstrate the predictive ability of our best model on the training set. We see that it is doing quite a good job at predicting the sale price. There are a lot of points that are able to be predicted exactly or with very small amounts of error. We see that there is a pretty even distribution of points that were overestimated and others that are underestimated, which is a good sign. The mostly random scatter in the Fitted vs Residual plot also implies that our approach which used mostly linear models was a good idea because there are not any trends that would encourage a transformation or more nonlinear models. We do see a few major outliers. These might be impacting our score drastically and it would be worth examining what makes these points different. If they have anything in common, this information could help us to improve our model even further.

Ablation Study

Our first method of introducing novelty to this project was the principal component analysis. This was originally improving our results for the support vector regression, as previously mentioned. However, as we continued cleaning our data and training more sophisticated models, its usefulness decreased. The increase in error for these models, though, was not very large. It was able to improve several of the models we tried, including one of our top five, but was not contributing to either of our best models. Since it does not always do worse and the performance does not lower by much, in some cases it would be preferred to use PCA for the dimensionality reduction and complexity benefits. To be fair, as also mentioned previously, we were not able to complete hyperparameter tuning for two of our models after PCA was implemented. As of now though, for the ablation study we see that removing feature selection with PCA is causing most of our models to improve. Our best model, which was a combination of the support vector and XGB, worsens from 0.121653 to 0.147363 when we implement PCA.

Training a model to impute the missing lot frontage values was a really interesting process and initially improved the model by quite a bit. When replacing this technique with a rudimentary version for the ablation study we decided to impute all of the missing values with the mean of the lot frontage column instead. The model trained to impute the data improved all of our initial models quite a bit; like XBG which improved from 0.125486 to 0.125216. Our best model saw a decrease in the RMSE of 0.001 which isn't very large but still means that the model-based imputation is preferred. Interestingly, the RMSE for the support vector regression changed from 0.127325 to 0.127290 meaning that it had a slight (very small) preference for the average lot frontage as opposed to our model based imputation. These opposite trends in the XGB and SVR could explain why they are able to achieve the best results together and certainly explains why the overall score does not change much when we change how we impute the lot frontage.

Another interesting thing we learned while performing the ablation study on our lot frontage model was that the models that were trained with PCA tended to perform worse in this case as opposed to when the missing values were filled in with the mean lot frontage. For example our two best combined models after PCA, the averaged LASSO and SVR model and the averaged ElasticNet and SVR model resulted in root mean squared errors that were 0.000358 and 0.001073 lower, respectively when the lot frontage was imputed by the mean. The fact that the PCA models tend to dislike the lot frontage XGB model implies that this feature is likely being dropped or given a very low weight in the end.

After training all of our initial models we decided to average them as well to achieve improved performance, which we did. Without implementing this novel way of using machine learning, our best model is the XGB as previously mentioned. At the time we decided to average models though we had not yet discovered XBG (RMSE=0.125216) and the best standalone model was the support vector regression, achieving a RMSE of 0.127325. The best average model before we discovered XBG was a combination of the support vector regression and a random forest which obtained a RMSE of 0.127128 and we repeated this process after implementing XBG to find that support vector regression averaged with XGB was now the best

combined model with a RMSE of 0.121653. See supplementary materials for tables of our best performing averaged models and their corresponding root mean square errors.

Future Studies

As previously mentioned, we weren't able to effectively tune the hyperparameters for the XBG and random forest models on the PCA data because of the computational requirements. It would be nice in the future, if we have the computing power or learn of a more efficient process, to fully tune the hyperparameters for these two models and see if we are able to achieve better performance. We would also like to try other feature engineering methods, such as different methods of scaling and kernel functions. At this point we have only used the standard scaler in Scikit learn and trying other methods better suited for our data could be beneficial.

In the future we would like to research and explore other models to try. The XGB model was one we found in our research very very late into the project and it was able to achieve very high performance with less effort than some of the other models. We would like to research other existing models to see if there are any others that perform exceptionally well for this dataset. It is likely that the model would be some sort of tree based model as these were performing the best due to the more complex relationship between the data and the outcome. Also, averaging the models worked quite well for us and led to quite impressive improvements. It would be interesting to look into and try out other, potentially more sophisticated, ways of combining multiple models. One strategy we have heard of in the past that may lead to interesting results for this data is training one large model that based on the presence of certain features choses a specific model within itself to use. This strategy seems as though it'd be fun to implement in general but we also believe that it would be successful because averaging models worked well for us. The only potential issue is that the training data may not be large enough or segmented enough to split it in a way that best takes advantage of this strategy.

Supplementary Material

Top ten models (average or standalone) that did not use PCA

sv+xgb	0.121653
(rf+s)+xgb	0.123141
rf+s+xgb	0.123807
xgb	0.125216
sv+rf	0.127128
(rf+e)+xgb	0.127166
sv	0.127325
rf+xgb	0.128338
e+xgb+rf	0.129209
all	0.129515

Top ten PCA models (though not explicitly marked as PCA, each model listed other than XGB are trained and fit on PCA data)

lasso+sv	0.138182
elastic+sv	0.138269
e+rd+sv	0.140672
ridge+sv	0.140899
sv_pca	0.141612
e+lasso+xgb	0.145134
lasso_pca	0.146009
elastic+lasso	0.146060
e+rd+xgb	0.146065
elastic_pca	0.146116

Top ten models that combine PCA and non-PCA models

(lasso_pca+sv_pca)+sv+xgb	0.124863
lasso_pca+sv_pca+sv+xgb	0.127382
(lasso_pca+sv_pca)+(sv+xgb)	0.127382
(e_pca+rd_pca+sv_pca)+xgb	0.128454
sv+sv_pca	0.132004
e_pca+rd_pca+sv_pca+xgb	0.133545
all	0.135552
lasso+lasso_pca	0.141534
elastic+elastic_pca	0.141726
ridge+ridge_pca	0.145823