ORIE 4741 Final Report

Philip Ayoub (pja66) and Kevin Cushing (kxc4)

Fall 2021

1 The Problem

The pricing models of houses are black boxes. 10,000 Sq. mansions sell for the same price as a 2 bed room ranches depending on a number of both qualitative and quantitative factors. Over the course of this project we attempt to predict the sale price of housing properties using numerous feature engineering and machine learning methods. Through this process we not only hope to produce a valid model, also gain insight into the key features that result in in the largest difference in house pricing.

2 The Data Set

2.1 Overview

We are using the data set 'house-prices-advanced.csv' to gain insight into what determines the "price-tag" of a house. The data set consists of 81 features and 1406 instances ranging in values and data-type. Among these features there are 31 real-valued columns, 26 categorical columns, 18 ordinal columns, 1 Boolean and 1 Regression column (Y-Values).

2.1.1 Real valued columns

These features consist of columns such as 'LotArea', PoolArea', and 'GarageArea'. These three features happen to be in measures of square feet, however there are also integer values representing the number of bedrooms, bathrooms, and more. All the data in these columns are integers.

'LotFrontage':	60.5
'LotArea':	1.7
'PoolArea':	200

2.1.2 Categorical columns

These features consist of columns such as 'Utilities', 'SaleCondition', and 'Heating'. These columns all have string values that describe the qualitative values of the house.

'Street':	['Pave', 'Grvl']
'LotShape':	['Reg', 'IR1', 'IR2', 'IR3']
'LandContour':	['Lvl', 'Bnk', 'Low', 'HLS']

2.1.3 Ordinal valued Columns

These features consist of columns such as 'OverallQual', 'GarageQual', and 'GarageCond'. The features in this category are strings that mainly describe the conditions of different features of the house.

'ExterQual':	['nan','Po', 'Fa', 'TA', 'Gd', 'Ex']
'KitchenQual':	['nan','Po', 'Fa', 'TA', 'Gd', 'Ex']
'HeatingQC':	['nan', 'Fa','TA', 'Gd','Ex']

2.1.4 Boolean Valued Columns

There is only one feature that falls into this category, 'CentralAir', which carries values of 'Y' and 'N' representing whether or not a house has a centralized air conditioning system.

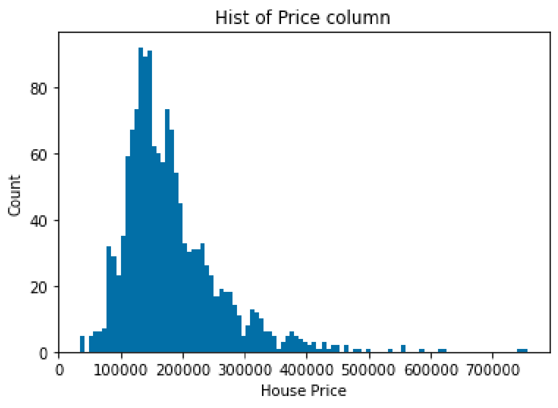
'CentralAir': ['Y', 'N']

2.1.5 Corrupted/Missing Values

There where a small but note worthy amount of missing values in our initial data set. However, there were no corrupted values which helps verify our confidence in the data set. We classify missing values as Nan or inf values and corrupted values as non-sense values such as negative numbers in a feature that describes area.

2.1.6 Y-Column

We did a number of checks to determine the validity of the "label" columns due to its critical importance for training a proper model. We came to the conclusion that the label was trustworthy largely due to the right skewed distribution shown in the histogram below. This distribution seems reasonable since there is no upper bound for the price of a house, as some can sell for many millions of dollars, however there is a fixed lower bound of 0 which is somewhat close to the median of house prices. These constraints are satisfied by the distribution of housing prices in our data, so we have no reason to believe our data is corrupted.



2.2 Data Preproccessing

2.2.1 Handling Missing Data

We encountered numerous missing entries throughout our data set. In our initial investigation, we found that these missing values are informative. For example, numerical valued features such as 'PoolArea' and 'GarageArea' have some missing values. Upon further we noticed related features like 'PoolQual' had the value 'NA.' Thus the data inspection, we noticed that the information encoded by these missing values was also encoded in another ordinal variable like 'PoolQC' and 'GarageQual', representing the quality of the pool and garage respectively. These values are missing when the values for the corresponding real valued columns are also missing. In order to extract this information and clean our data set, we replaced the missing values in both columns with '0's. This was done in order to Another possible way to remedy this issue would have been to add another feature to indicate whether the house had a pool or not. We ended up trying this type of feature transformation by encoding our ordinal variables in a one hot manner which resulted in a new feature representing whether or not each of the ordinal values had a missing value. This encoding resulted in decreased model performance so we decided against using it, and stuck with our original encoding which we describe below.

2.2.2 Numerical Values

We embed these numerical features "as is" in the training set of our models.

2.2.3 Categorical Values

For the features in our data set that fell under this category we choose to use one-hot encoding to transform these features from Strings to Integers.

'Street': 'Grvl' $\rightarrow$ 'Street': [0, 1]

2.2.4 Ordinal Values

For the features in our data set that fell under this category we choose to give integer values based on the order of the value in question. For instance, many of the features in this category describe the condition of a facet of the house, ie. ['nan','Po', 'Fa', 'TA', 'Gd', 'Ex']. Thus we transformed these values into [0,1,2,3,4,5] in order to convert the datatype of the input from Strings into Integers.

'KitchenQual': 'TA' $\rightarrow$ 'KitchenQual': 3

2.2.5 Boolean Values

For the features in our data set that fell under this category we choose to change the value of true to 1, and value of false too 0.

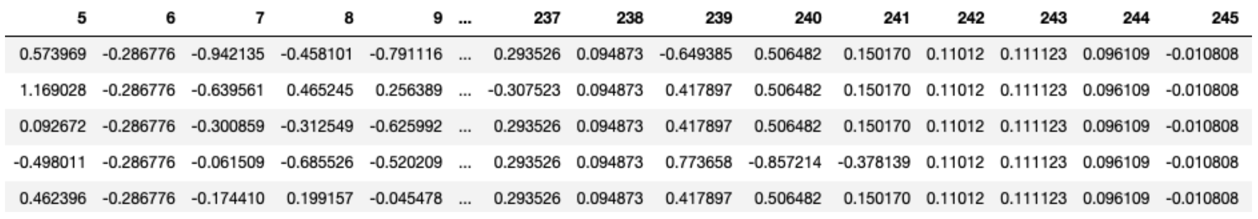
'CentralAir': 'Y' $\rightarrow$ 'CentralAir': 1

2.2.6 Standardizing Data

To standardize our data, we used sklearn's StandardScaler class which scales each feature to have a distribution with mean 0 and variance 1. This is an important step, especially for when we train our gradient boosting regressor since scaling the data in this way ensures the gradient of our loss with respect to our data is well-conditioned. Standardization is also important for interpreting the importance of different features in our models since the weights of features will be somewhat inversely proportional to the range of values they take so standardizing the data for each feature to the same mean and variance will allow us to interpret our models more accurately.

2.2.7 Final Data Set

The converted data set contains 247 features that all have datatype integer.



5	6	7	8	9	...	237	238	239	240	241	242	243	244	245
0.573969	-0.286776	-0.942135	-0.458101	-0.791116	...	0.293526	0.094873	-0.649385	0.506482	0.150170	0.11012	0.111123	0.096109	-0.010808
1.169028	-0.286776	-0.639561	0.465245	0.256389	...	-0.307523	0.094873	0.417897	0.506482	0.150170	0.11012	0.111123	0.096109	-0.010808
0.092672	-0.286776	-0.300859	-0.312549	-0.625992	...	0.293526	0.094873	0.417897	0.506482	0.150170	0.11012	0.111123	0.096109	-0.010808
-0.498011	-0.286776	-0.061509	-0.685526	-0.520209	...	0.293526	0.094873	0.773658	-0.857214	-0.378139	0.11012	0.111123	0.096109	-0.010808
0.462396	-0.286776	-0.174410	0.199157	-0.045478	...	0.293526	0.094873	0.417897	0.506482	0.150170	0.11012	0.111123	0.096109	-0.010808

3 The Solution

3.1 Initial Regression Analysis

We initially tried 4 different types of regression models in order to gain insight into our problem and ultimately lead us to the best possible solution. We used 5-Fold cross validation to help reduce the variance of our results due to random train and validation splits. We primarily used the square root of the mean squared logarithmic error to evaluate our models since our data is from a kaggle competition and this is the metric they use for the test error. We also use mean absolute error in order to allow us to better interpret the accuracy of our models. The error metrics for each initial regressor from 5-fold cross validation is presented in Table 1 and plots of each regressors' predictions versus the actual values on a validation set are in Table 2. Each of these initial models was fit using the sklearn implementation of the regressor and its default parameters.

Regression Type	Root Log Mean Squared Error	Mean Absolute Error
OLS	0.3438	21187.12
PCA	0.2436	24360.86
Random Forest	0.1457	17581.14
Gradient Boosting	0.1295	15837.84
Huber	0.1678	17608.74

Table 1: Initial Regression Model Performance on 5-fold Cross Validation

3.1.1 Linear Regression (OLS)

From initial intuition, we devised that the price of houses should be largely linear based on certain features, ie. Having a pool increases your house consistently by 30,000. Thus we felt that a good starting would be to fit a Least-Squares Linear Regression Algorithm to set a solid baseline for our final models prediction power. This model preformed rather poorly.

3.1.2 Principal Component Analysis combined with OLS

To help determine which features are the most important and to produce a more digestible dataset we decided to use PCA, with 90% of the variance explained by the number of features chosen. The number of features this resulted in was 128, which is about half as many as our initial processed dataset. We then used this new feature set as inputs to an OLS model because the 'combined' variables generated after PCA are all independent which is ideal for linear regression models. This greatly improved the performance of our original OLS model.

3.1.3 Random Forest

We decided to see how a random forest regressor would perform on this data set since we suspected that our data may be non-linear due to the large errors in the first two linear models we tried. We choose to this model due to the non-linearity of decision tress and the fact that the random forest algorithm trains each decision tree on bootstrapped samples from the data and only a subset of their features which helps reduce the variance of this model, leading it to perform rather well.

3.1.4 Gradient Boosting Regression

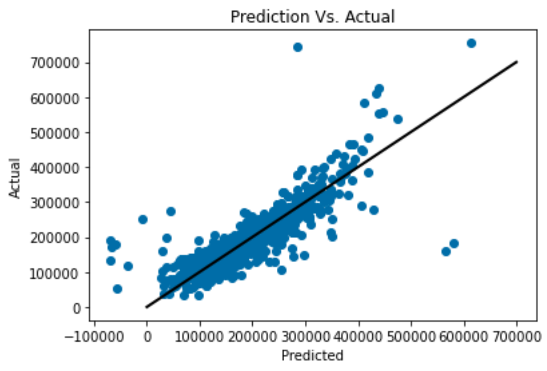
Per the success of our Random Forest Model, we choose to try another ensemble tree model. Gradient Boosting generates a prediction model in the form of an ensemble of weak models which, in our particular case, are decision trees. Due to the fact that Gradient Boosting also uses decision trees this model is also non-linear. The model preformed as expected after our great success with our previous decision tree model and we realized that our data set is most likely not linear.

3.1.5 Regularized Linear Regression

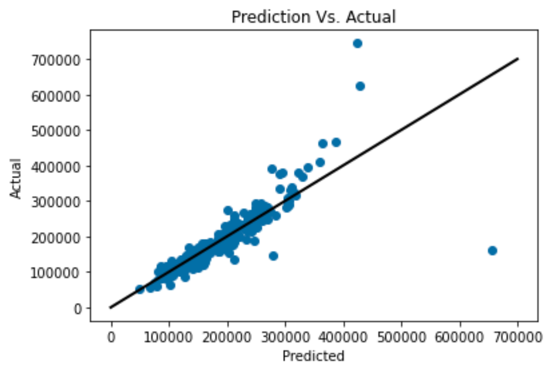
Due to the relatively large error of our original OLS model, we decided to try a few regularized linear models. We did this for ridge, lasso, and huber regularizers. We did this because regularizers increase a model's bias but decrease its variance so adding a regularize will help us minimize the bias-variance trade-off of our model when we tune for the regularization factor. The model using the huber regularizer performed the best of these regularized models, each of which performed better than the original OLS model.

3.1.6 Initial Regression Results

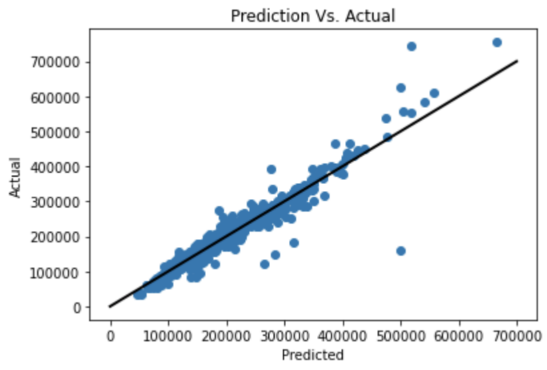
OLS



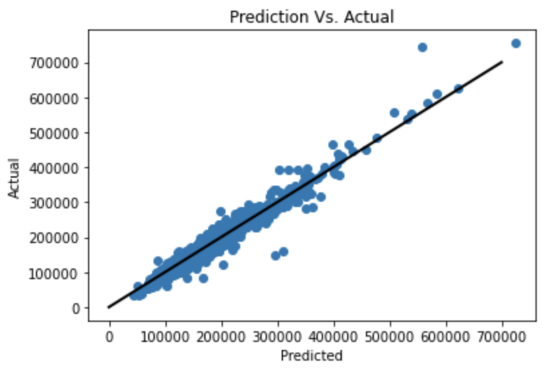
Huber



Random Forest



Gradient Boosting



PCA OLS

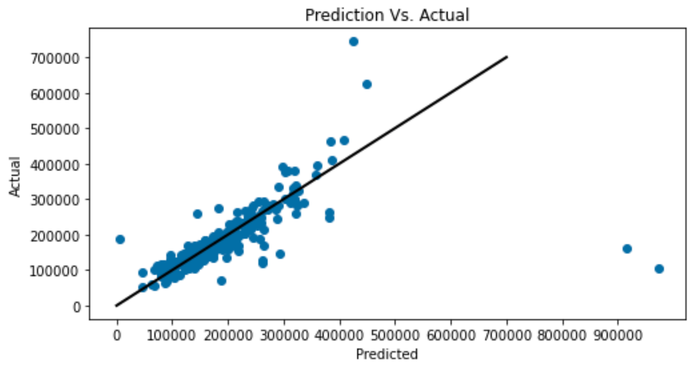


Table 2: Initial Regression Model Predictions on a Validation Split

Regression Type	Root Log Mean Squared Error	Mean Absolute Error
Random Forest	0.1539	19100.40
Gradient Boosting	0.1483	18218.12
Huber	0.1535	19252.20

Table 3: Regression Model Performance on 5-fold Cross Validation Using PCA Dimensionality Reduced Features

3.1.7 PCA Feature Transformation

After conducting the above regression analyses, we wanted to see if constructing a smaller set of features using PCA would improve the performance of our models. We found that only our huber regression model's performance improved and this improvement was very minimal. Thus, we decided to stick with our initial feature set since it seemed to yield slightly better performance and has much more interpretable features. Results from this experiment are in Table 3.

3.2 Model Tuning and Selection

After performing the above initial analyses, we decided to use sklearn's RandomSearchCV, which performs a random search of given sets of model parameter values. We did this for random forest, gradient boosting, and huber regressors and chose 25 combinations of parameters for each. The best model was then chosen from these using The 5-fold validation results for the best models are displayed in Table 4.

3.2.1 Random Forest

The parameter space we searched over for our random forest regressor consisted of n_estimators in [1,2,3,...,500], which is the number of estimators in the random forest ensemble, max_depth in [1,2,3..,19], which is the maximum depth of each tree in the ensemble, and min_samples_split in [1,2,3,4,5], which is the minimum number of samples needed to split an internal node of a decision tree. While the model has many more parameters, we believe these values give it good coverage over all possible values since some of the other parameters of the model like max_leaf_nodes are related to parameters we search over like max_depth. Our random search found that parameters of n_estimators=371, min_samples_split=2, and max_depth=19 performed best in 5-fold cross validation.

3.2.2 Gradient Boosting

To choose the best parameters for our gradient boosting regressor, we searched the parameter space containing n_estimators in [1,2,3,...,500], which is the number of base estimators in the bagged ensemble, subsample in [np.linspace(1,0.5, 100)], which is the proportion of the training data to be sampled in fitting each base estimator, min_samples_split in [1,2,3,4,5], which is the minimum number of samples needed to split an internal node of an individual tree estimator, and loss in 'squared_error', 'absolute_error', 'huber', 'quanitle', which is the loss function for which the gradient is computed when boosting. The best parameters found through our random search for this model were n_estimators=424, subsample=0.6616, min_samples_split=3 and loss='absolute_error'.

3.2.3 Huber

The parameters we searched over for our huber regression model were epsilon in [np.linspace(1.01, 5, 10)], which is a value greater than 1 that determines how many samples the model classifies as outliers, and alpha in [10^{-i} for i in range(0, 6)], which is the regularization parameter for the model. The random search found that the best performing values for these parameters were epsilon=2.7833 and alpha=0.1. These are the only parameters to tune for huber regression so we believe these possible values give us good coverage over all possible parameter values for this model.

Regression Type	Root Log Mean Squared Error
Random Forest	0.1444
Gradient Boosting	0.1244
Huber	0.1533

Table 4: Model Performance on 5-fold Cross Validation Using Optimal Parameters from Random Search Tuning

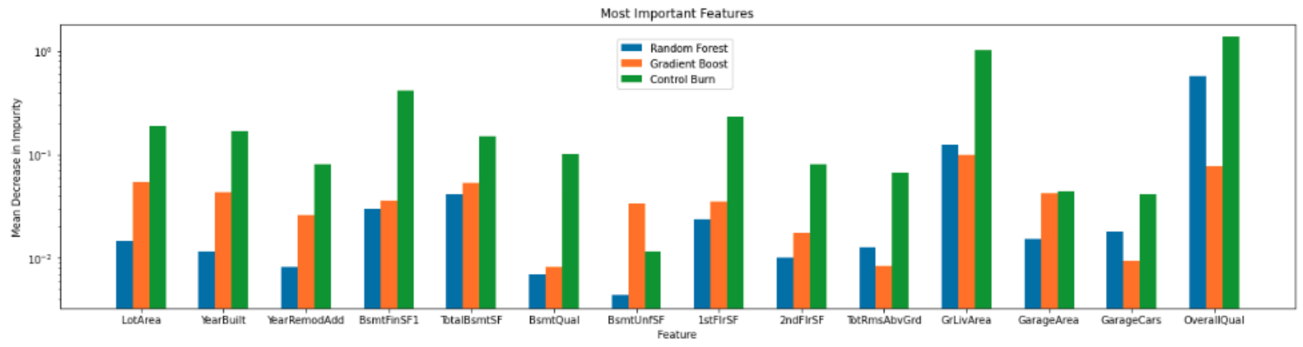


Figure 1: Feature Importance by Model. Features from left to right are LotArea, YearBuilt, YearRemodAdded, BsmtFinSf1, TotalBsmtSF, BsmtQual, BsmtUnfSF, 1stFlrSF, 2ndFlrSF, TotRmsAbvGrd, GrLivArea, GarageArea, GarageCars, OveralQual. The blue, orange and green bars represent the mean impurity decrease for each feature in the random forest, gradient boosting, and control burn models respectively. The y-axis is on a log scale.

3.3 Analyzing Feature Importance

One of the most important aspects of this model is its ability to detect which features are most important in predicting the value of a house, since this will give insight into how to best increase the value of a property. Figure 1 shows the importance of the set of features that were in the top 10 most important features in each model. There are only 14 such features and 3 models, which means the models often agreed on which features were most important. In general, we see that upgrading the living area and basement, as well as the square footage of all areas of the house affect its value the most.

Using the random forest regressor with parameters tuned, we can inspect the importance of the features of our dataset. You can see in Figure 1 that 'Overall Quality' is by far the most important feature in this data set for all models, except maybe gradient boosting. This makes sense since regardless of other features, if the house is of poor quality, it will sell for less than houses of better quality. Other features that this model seems to value most include the square footage of the basement, the finished square footage of the basement, and the grade of the living area. Similarly to our random forest regressor, we can look at the importance of features to our gradient boosting regressor. In doing this we notice that many features are more important to it than they are to the random forest model. To this model, the grade of the living area of a property is the most important feature, followed closely by overall quality and several other features.

For this section, we also decided to experiment with a control burn model. The control burn model we trained achieved a 5-fold cross validation average root log MSE of 0.1450 and an average MAE of 17595.72. This is close to the performance of our best tuned models. The benefit of control burn is that it is designed to use only a sparse subset of features in its predictions. Our control burn model ended up selecting 52 features to use in ts predictions, about 20% of the total number of features available. As can be seen in Figure 1, the most important features are generally much more important to this model relative to the other two models which is due to the sparsity of features it uses in its predictions. This sparsity helps make it more interpretable by removing sets of correlated features and essentially combining their weights into one.

4 Weapon of Math Destruction(WMD)

WMD are models that have the potential to have a negative outcome on society if they make incorrect predictions based on bias and or 'bad features' (Racist, Sexist, or Anti-Semitic). This situation could become present in out model if it is not trained carefully. Households are often the largest facet of an individuals net worth and usually are an investment that is not treated lightly. Thus, if our model makes errant predictions it could have an extremely negative effect on peoples' lives. The situation in which our model becomes an WMD is when homes prices, more specifically the impact of 'neighborhood' and similar features has on home prices, is influenced by the demographic of people living there. There are no fields that directly represent race, age, sexuality, and or religion, but 'neighborhood' can be indirectly influenced by these demographics if we do not source our data mindfully. Housing prices shouldn't be impacted by the demographic of the neighborhood but rather the physical traits, ie. is there a park. Thus, if the person training the model is not careful, this algorithm could turn into a WMD. However, due to differences that occur between neighborhoods, it is not possible to tell whether the impact of neighborhood on the predicted price is due to the demographics of the neighborhood or its location and layout of houses. A possible way to prevent our model form becoming a WMD would be to collect more data on each neighborhood including demographics of people living there and its proximity to desirable locations and ensure that demographic features do not create biases in the predictions of our model.

5 Fairness

There are many view of 'fairness' when it comes to machine learning algorithms, however we will focus on 3 the relate best to our data set and model.

5.1 Unawareness

"Unawareness" describes the feature that the model should not factor into its output. This metric often describes features based on race, religion, and or sex. As discussed above, the skin color, race, or gender of the owner of the house should never be factored into the price estimation of the house. Only features that directly connect to the house itself should be considered.

5.2 Demographic Parity

Demographic parity touches upon the same topic discussed above but in a more specific manner. The owner of the household, and the demographics of the people who live in the neighborhood should never be provided as inputs for this given model.

5.3 Equalized odds

Finally, 'equalized odds' is also a view of 'fairness' that should be taken into consideration. If a house has similar feature, 'neighborhood, 'square footage', amenities,' then the outcome of the model for both these records should be similar.

6 Conclusion

Overall, we believe that we have generated a great model that predicts the value of a property within $15000 on average! This can be a margin of negotiations on a properties price so We believe that we have properly weighed the negatives and positives of creating a model that analyzes this property prices. Based on both our models predictive success and our belief that it largely abides by the 'fairness' definitions laid out above, we would highly suggest that our model be used in the daily operations of the firm. While we may not be able to get rid of apraisers all together with this model, it will certainly aid in evaluating property values and give insight into how a given property's value could be increased in a cost-effective manner. Thank you so much for your time and consideration.

7 Sources

- https://www.kaggle.com/c/house-prices-advanced-regression-techniques
- https://scikit-learn.org/stable/
- https://datascience.stackexchange.com/questions/69572/how-to-determine-which-features-matter-the-most
- https://pypi.org/project/ControlBurn/