# Predicting House Prices in Ames, Iowa

Machine Learning Project

Jay Cherukuri
School of Computing and Information
University of Pittsburgh
Pittsburgh, PA USA
jac426@pitt.edu

Divya Ghandi
School of Computing and Information
University of Pittsburgh
Pittsburgh, PA USA
dig22@pitt.edu

## ABSTRACT

The value of machine learning is the ability to harness computer technology for the processing of complex statistical learning. Machine learning excels in analyzing large datasets to define what features matter and to predict a result based on those variables. In our report we provide details about how we used various machine learning techniques and algorithms to complete the Kaggle competition, "House Prices: Advanced Regression Techniques."[1] This paper goes into detail about the data cleaning process, including relevant data exploration of the 2,919 observations and 81 features provided by Kaggle over 2 data sets. Various machine learning models, including a neural network with TensorFlow, were evaluated as candidate models on the basis of testing Root Mean Squared Error (RSME) to determine the best model for prediction and ultimate submission. The models, trained for optimal parameter selection with cross-validation, were run on different subsets of the data to explore the potential of feature selection and feature engineering. Our work demonstrates the value in the data analysis process, from cleaning to modeling. The result was a score in the Top 35% of competition participants at the time of submission.

## 1 INTRODUCTION AND PROBLEM DESCRIPTION

A common dataset used in machine learning education is the Boston housing dataset. [2] The Boston dataset provides information about the suburbs of Boston, MA. The response variable is that of median value of homes for those neighborhoods. Predictive features include the crime rate, the tax rate and the average number of rooms in each household, among 10 others.

The Ames dataset is in the mold of that Boston dataset. The Ames dataset is a survey of home sales in the Ames, Iowa area from January 2006 through July 2010. The broader dataset is separated into two sub-datasets, the train and the test set. In the training set, 1,460 observations accompany 81 features. Beyond the difference in features, the Ames dataset differs from the Boston set in that it includes categorical variables in addition to continuous features. Of the 81 variables, 80 are meaningful; a unique 'Id' is provided for each observation. One feature is also the response of 'SalePrice.' The testing set is nearly the same. The key difference is that the response variable is not provided. The competition participant is to predict the 'SalePrice' for each testing set 'Id' and submit it to Kaggle, where it will be evaluated on the basis of minimizing RSME of the predicted response and that of the dependent variable of the testing set.

## 2 RELATED WORK

House price prediction has been a topic for many academic works. This popularity may be a function of it being a topic that doesn't necessitate a wealth of domain knowledge. For example, it stands to reason that a large house with a substantial number of rooms will go for a comparatively high price relative to a small house with a limit number of rooms.

L. Yu et al. explored the prediction of homes in Beijing, China based on a variety of features derived from publicly available data, including geospatial

[1] https://www.kaggle.com/c/house-prices-advanced-regression-techniques

[2] https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

information. What was novel about their approach was the usage of deep learning for a regression problem of this type. The authors built several models, including a convolutional neural network (CNN), to predict prices. Their results were good, minimizing error to a satisfactory extent for that aspect of their implementation. Relevant to this problem type was their success in reducing accuracy with an increasing the number of iterations over the data. The error reduced with additional training time. Also of note, the authors preprocessed the data, normalizing features to a variance of 1 and a mean of 0, or a standard scaler (Yu, Jiao and Xin).

Zurada et al. also explored a neural network. The authors built several models to compare house price predictions for the Louisville, KY area, including a neural network and advanced regression techniques like gradient boosting. They found that the neural network generally performed worse relative to regression models when the data was similar in nature. Effectively, the neural network used didn't scale well to what was generally linear data. Another interesting aspect of their work was the use of 10-fold cross validation. The authors argued that building a model without cross-validation would result in a prediction that wouldn't generalize well to new data (Zurada, Levitan and Guan).

Mu et al. discussed Support Vector Machines (SVM) and Support Vector Regression (SVM.) SVM is a technique that addresses non-linear data well, but can handle linear functions. Potentially valuable in this context is its ability to get accuracy from a limited number of training samples. Utilizing the Boston dataset, the authors explored SVR in the context of predicting house prices. Accuracy was satisfactory, but processing time is a disadvantage (Mu, Wu and Zhang).

Relevant work unrelated to housing data was that of Kaneko et al. Their approach to Japanese supermarket data involved deep learning. Building a 3-layer neural network, sales data over a 3-year period was analyzed with H2O to predict the highest selling items in categories organized by product specificity. Relevant to our work was the usage of regularization on the hidden neural network layers. AUC, a measure of classification prediction accuracy, showed increased performance with the constraint on the layers. Regularization can help prevent overfitting. An interesting part of the work is the performance degradation the authors noticed as the number of samples increased (Kaneko and Yada).

Modeling is only one part of this problem. The data itself is a challenge. Chandrashekar et al, argued that features with limited relevance to the problem can add unnecessary noise to the problem and thus, effect performance. The authors suggested several methods for selecting features to reduce dimensionality. Feature selection can be done by ranking. One ranking methodology is correlation. The danger in using the methodology is that important features aren't necessary strongly correlated with the response. Stripping them out of the model may reduce performance. Filtering algorithms may not be useful in conjunction with algorithms that learn from the data. Indirectly, a feature selection component is already a part of the underlying methodology for boosting techniques. That logic is applicable to SVM, as well as neural networks (Chandrashekar and Sahin).

Feng et al. discussed the dangers of log transformation of skewed variables. The authors suggest that taking the logarithm can reduce skewness, making the data more normal. However, it is not a perfect means of doing so. Overcorrecting for skewing can result in data that is even more skewed and create greater variability in the data than is already present for certain types of distributions. A log transformation of a right skew can result in a left skew for the reshaped data. The authors cautioned against using transformation for potentially inaccurate results. A log transformed model may not generalize well to new data, given potential issues with variability (Feng, Wang and Lu).

Abedin et al., discussed approaches to null values and outliers. In sensitive medical experiments, accuracy is of the upmost importance. Many machine learning techniques are sensitive to outliers. Too many null values can also throw off model performance.

Relevant to a dataset with null values, the authors suggested that filling in with mean values is not the optimal solution. Replacing values with the median value for a feature was shown to provide a more accurate result in their studies (Abedin, Rahman and El-Baz).

## 3 DATA CLEANING AND TRANSFORMATION

### 3.1. HANDLING OF NULL VALUES

The Ames dataset, both training and testing, is rife with errors. The primary issue is that of missing or null values. 19 of the 81 variables in the training set and 33 of the 80 features in the testing set contain null values. The missing values are of two different types. The bulk of the missing data is a result of the feature not existing for that observation. For example, 1453 values are missing for 'PoolQC', which measures the quality of the pool, if it exists. In lieu of the feature having a value of 'NA', which corresponds to no pool, it was left empty during collection. The other null values are simple data errors unrelated to the feature not existing. For example, 'Id' 1380 has a null value for its electrical system. There is no corresponding categorical value for no electrical system and every other home has one.

In the process of encoding categorical variables, which will be described in the next section, it was discovered that there were some spelling errors in the dataset. For the feature 'Exterior2nd' in the training set, the tag 'CemntBld' was misspelled as 'CmentBld.' Those data errors were comparatively small in number.

The handling of null values was done on a case by case basis. Where there were a large number of missing values for a variable, the feature was just dropped. In most cases, there was another variable to account for its presence as part of the home sale. For example, the feature 'PoolArea' captures the information about whether a pool exists in a similar manner as the categorical variable 'PoolQC'; if the home doesn't have a pool, the pool doesn't have an area. For variables with smaller amounts of null values, each case was analyzed to identify the proper course of action. Foe example, 'Id' 2611 has a null value for its 'MasVnrType,' despite having an area. In an isolated case such as this one, the median value was filled in. Great care was taken to clean at a micro level in these cases.

While the approach seems inconsistent, it was done for a purpose. Observations could have been dropped, but valuable information, such as the sale price and the size of living area would have been lost. It was important to maintain as many observations as possible. The features with the smaller amounts of null values may not end up being used during modeling regardless.

### 3.2 HANDLING OF CATEGORICAL VARIABLES

While null values posed a challenge, the other came from categorical variables. The aforementioned Boston dataset contains a categorical variable, but it has already been converted into a continuous integer attribute. String values are not compatible with scikit-learn's packages. Those variables have to be encoded into numeric values.

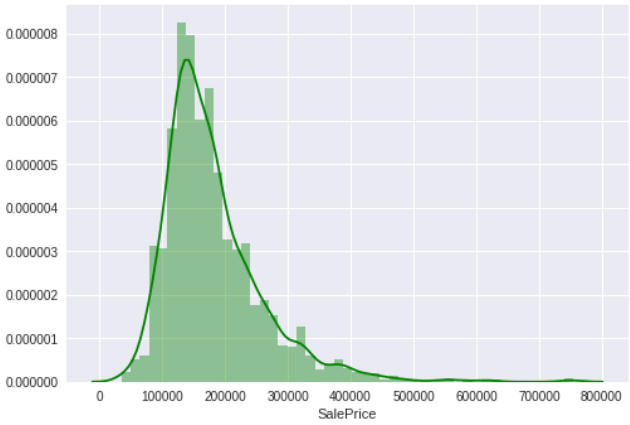
The overall dataset has two types of categorical variables. There are those that have meaning to their order and those that don't. A variable such as 'BsmtCond,' which measures the condition of the basement, has values ranging from 'Ex' or excellent to 'Po' or poor. A variable such as 'SaleType' indicates what type of sale it is; categorical variable options range from 'Warranty Deed – VA Loan' to 'New home just constructed and sold.'

For those values with meaning to their order, the encoding scheme used captured that. The higher the numeric value for 'BsmtCond,' the better condition the basement is in. The other type of categorical variable was encoded starting from 1 on the basis on how they were ordered in the data dictionary.

Mapping was used to accomplish the task. Label encoding was the preferred method, but the same changes made to the training set had to be made in the same exact manner to the testing set, for the purpose of modeling accuracy. Label encoding will order categorical or numerical values based on frequency. The frequencies may be different across sets. More broadly, the changes made to the training set discussed were made to the testing set.

### 3.3 HANDLING OF SKEWED DISTRIBUTIONS

One thing that is noticeable about the data upon exploration is the skew of the distribution for many features, including the response. Only select variables have a gaussian shape. To treat the skewness issue, the selected technique was to take the logarithm of the values to make it be more normal. Two approaches were used when it came to this. Initially, the logarithm of every feature was taken so as to be consistent throughout the data set. Error decreased relative to a model incorporating all features unskewed, with those features scaled or not. As discussed in the related work section, broad log transformation can lead to the overcorrection of skewness and result in a negative impact on model performance (Feng, Wang and Lu). In lieu of skewing for all features, only those with a positive skew above .75 were selected. Model performance showed noticeable improvement across the board using this filter.

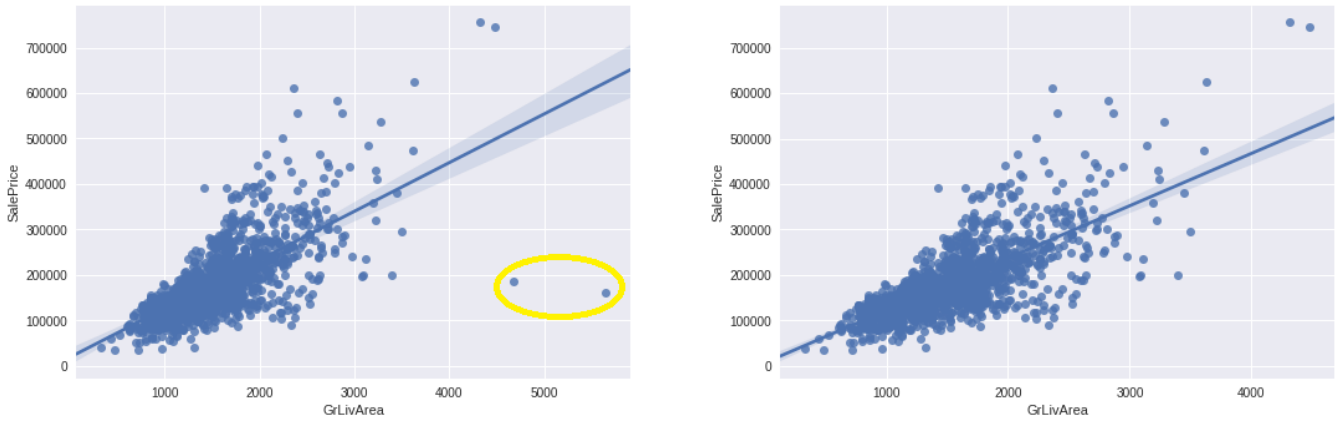



'SalePrice' has a skew of 1.88

### 3.4 HANDLING OF OUTLIERS

Another aspect of this dataset is outliers. Outliers are not unexpected for a topic such as house prices. It's not uncommon for a house with a similar profile to hold greater value than another for reasons unrelated to a given relationship. The trend realized in the training set is the strong, positive correlation between the sale price and the area of the home above the basement. A regression plot of the two variables showcases significant variability for houses with a living area above 4000 square feet. Upon further inspection of the data, what differentiated the data points in question was 'SaleType.' Two large homes were sold at a discount because they were transferred to family members. In a normal circumstance, they would have sold for the higher prices comparable homes went for.

The initial approach was to ignore outliers across the dataset to be consistent. Going through every relationship, trying to manually evaluate every outlier would have been a time-consuming activity possibly without a positive result. As different iterations of the approach were tried, it was found that treating some outliers resulted in better scores. Some machine learning algorithms handle outliers well and others don't (Abedin, Rahman and El-Baz). An increase in accuracy for linear regression was noticed immediately treating for those outliers that had a large living area above 4000 square feet, but a low sale price under $300,000. Performance did not improve for those outliers that involved an uncharacteristically high sale price above $700,000 for that size of living area. The compromise made was to treat outliers for that specific relationship and leave the others in the training set be. Without a sale price to compare to, the testing set was left as is.




Before and after outliers (yellow oval) are treated

## 4 METHOD

After cleaning the data, the training set was split into a sub training and testing set. The default scikit-learn split of 75% training and 25% testing was used. The sub-training set trains the model and the sub-testing measures the accuracy. For methods that allow for meaningful parameter tuning, 10-fold cross validation was used to measure the accuracy of each iteration to come up with the best parameters.

Most of scikit-learn's built-in regression models were used. In addition, a XGBoost module was imported. Keras was used in the TensorFlow plugin to round out the candidate models.

For all techniques with the exception of neural networks, the model was run with and without feature scaling. Further, a 4-principal component PCA analysis was run for KNN regression; accuracy suffered significantly with other modeling techniques when PCA was applied.

There were 3 different iterations of the dataset. The first was the data with all of the features remaining after cleaning. The second included features with high correlation to the sale price and limited variability in their distributions. The third was data adjusted for a restricted attempt at feature engineering.

Save for the TensorFlow implementation, performance was evaluated on accuracy and the RSME of the sub-testing set. There is a strong, negative correlation between those two metrics; increasing accuracy will result in decreasing RSME and vice versa. The RSME of the sub-training set was calculated to explore the fit of the of the model relative to the sub-testing set.

## 5. RESULTS

### 5.1 LINEAR REGRESSION

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.927 | 0.927 | 0.911 | 0.911 | 0.928 | 0.928 |
| Test RSME | 0.114 | 0.114 | 0.126 | 0.126 | 0.113 | 0.113 |
| Train RSME | 0.116 | 0.116 | 0.13 | 0.13 | 0.117 | 0.117 |
| Difference | -0.002 | -0.002 | -0.004 | -0.004 | -0.004 | -0 |

### 5.2 KNN REGRESSION

| | All Feat. | | | Feat. Sel. | | | Feat. Engin. | | |
|---|---|---|---|---|---|---|---|---|---|
| | R | FS | PCA | R | FS | PCA | R | FS | PCA |
| Accuracy | 0.68 | 0.815 | 0.845 | 0.68 | 0.848 | 0.868 | 0.679 | 0.82 | 0.846 |
| Test RSME | 0.239 | 0.181 | 0.166 | 0.238 | 0.164 | 0.153 | 0.239 | 0.179 | 0.166 |
| Train RSME | 0.211 | 0.167 | 0.15 | 0.211 | 0.15 | 0.145 | 0.211 | 0.16 | 0.155 |
| Difference | 0.028 | 0.014 | 0.016 | 0.027 | 0.014 | 0.008 | 0.028 | 0.019 | 0.011 |
| | | | | | | | | | |
| Best K | 8 | 15 | 9 | 8 | 10 | 13 | 8 | 10 | 13 |

### 5.3 RIDGE REGRESSION

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.927 | 0.925 | 0.911 | 0.91 | 0.928 | 0.926 |
| Test RSME | 0.114 | 0.116 | 0.126 | 0.126 | 0.11 | 0.115 |
| Train RSME | 0.116 | 0.117 | 0.13 | 0.131 | 0.117 | 0.118 |
| Difference | -0.002 | -0.001 | -0.004 | -0.005 | -0.007 | -0 |
| | | | | | | |
| Best Alpha | 1 | 100 | 1 | 100 | 1 | 100 |

### 5.4 LASSO REGRESSION

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.931 | 0.929 | 0.913 | 0.912 | 0.931 | 0.929 |
| Test RSME | 0.111 | 0.112 | 0.125 | 0.125 | 0.111 | 0.112 |
| Train RSME | 0.119 | 0.116 | 0.131 | 0.131 | 0.12 | 0.117 |
| Difference | -0.008 | -0.004 | -0.006 | -0.006 | -0.009 | -0.01 |
| | | | | | | |
| Best Alpha | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 | 0.001 |

## 5.5 DECISION TREES

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.83 | 0.83 | 0.816 | 0.816 | 0.829 | 0.828 |
| Test RSME | 0.174 | 0.174 | 0.1807 | 0.181 | 0.175 | 0.175 |
| Train RSME | 0.104 | 0.104 | 0.158 | 0.158 | 0.105 | 0.105 |
| Difference | 0.07 | 0.07 | 0.0227 | 0.023 | 0.07 | 0.07 |
| | | | | | | |
| Best Max_Depth | 7 | 7 | 5 | 5 | 7 | 7 |

## 5.6 RANDOM FOREST

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.891 | 0.88 | 0.896 | 0.883 | 0.89 | 0.892 |
| Test RSME | 0.139 | 0.146 | 0.136 | 0.144 | 0.14 | 0.139 |
| Train RSME | 0.077 | 0.084 | 0.083 | 0.082 | 0.079 | 0.08 |
| Difference | 0.062 | 0.062 | 0.053 | 0.062 | 0.061 | 0.059 |
| | | | | | | |
| Best Max_Depth | 10 | 10 | 10 | 9 | 10 | 10 |
| Best N_Estimators | 10 | 9 | 8 | 9 | 10 | 9 |
| Best Max_Features | 10 | 9 | 8 | 8 | 10 | 10 |

## 5.7 ADABOOST

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.903 | 0.901 | 0.89 | 0.889 | 0.902 | 0.904 |
| Test RSME | 0.131 | 0.136 | 0.14 | 0.141 | 0.132 | 0.131 |
| Train RSME | 0.088 | 0.087 | 0.096 | 0.096 | 0.088 | 0.087 |
| Difference | 0.043 | 0.049 | 0.044 | 0.045 | 0.044 | 0.044 |
| | | | | | | |
| Best Max_Depth | 5 | 5 | 5 | 5 | 5 | 5 |
| Best N_Estimators | 400 | 1600 | 1600 | 1600 | 800 | 1600 |

## 5.8 GRADIENT BOOST

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.933 | 0.933 | 0.912 | 0.912 | 0.929 | 0.929 |
| Test RSME | 0.109 | 0.109 | 0.125 | 0.125 | 0.112 | 0.112 |
| Train RSME | 0.089 | 0.089 | 0.092 | 0.092 | 0.09 | 0.09 |
| Difference | 0.02 | 0.02 | 0.033 | 0.033 | 0.022 | 0.022 |
| | | | | | | |
| Best Max_Depth | 1 | 1 | 3 | 3 | 1 | 1 |
| Best N_Estimators | 1600 | 1600 | 100 | 100 | 1600 | 1600 |

## 5.9 SUPPORT VECTOR REGRESSION

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.826 | 0.835 | 0.771 | -6.427 | 0.801 | 0.922 |
| Test RSME | 0.176 | 0.171 | 0.202 | 1.149 | 0.185 | 0.118 |
| Train RSME | 0.188 | 0.175 | 0.212 | 1.136 | 0.201 | 0.128 |
| Difference | -0.012 | -0.004 | -0.01 | 0.013 | -0.016 | -0.01 |
| | | | | | | |
| Best C | 0.1 | 10 | 1 | 0.01 | 10 | 1 |

## 5.10 XGBOOST

| | All Feat. | | Feat. Sel. | | Feat. Engin. | |
|---|---|---|---|---|---|---|
| | R | FS | R | FS | R | FS |
| Accuracy | 0.933 | 0.933 | 0.914 | 0.914 | 0.93 | 0.93 |
| Test RSME | 0.11 | 0.11 | 0.124 | 0.124 | 0.111 | 0.111 |
| Train RSME | 0.09 | 0.09 | 0.091 | 0.091 | 0.091 | 0.091 |
| Difference | 0.02 | 0.02 | 0.033 | 0.033 | 0.02 | 0.02 |
| | | | | | | |
| Best Max_Depth | 1 | 1 | 2 | 2 | 1 | 1 |
| Best N_Estimators | 1600 | 1600 | 400 | 400 | 1600 | 1600 |

## 5.11 NEURAL NETWORK

### 5.11.1 SCIKIT-LEARN

| | All Feat. | Feat. Sel. | Feat. Engin. |
|---|---|---|---|
| | FS | FS | FS |
| Accuracy | 0.904 | 0.885 | 0.909 |
| Test RSME | 0.13 | 0.143 | 0.127 |
| Train RSME | 0.13 | 0.144 | 0.132 |
| Difference | 0 | -0.001 | -0.005 |

### 5.11.2 TENSORFLOW

| | All Feat. | Feat. Sel. | Feat. Engin. |
|---|---|---|---|
| | FS | FS | FS |
| Test RSME | 0.127 | 0.128 | 0.123 |

## 6. DISCUSSION & CONCLUSIONS

### 6.1 BASIC REGRESSION MODELS

After basic cleaning, linear regression performed poorly, with accuracy hovering around .70. Feature scaling did not help in improving the score. It was only after the log transformation of the variables did the score begin to increase. Linear regression performed better with certain features, including the response, being more normally distributed. The removal of outliers added accuracy. Linear regression using the cleaned data offered excellent performance that more advanced models struggled to improve on or failed to

do so, including linear regression with ridge or lasso regularization. The restrained log transformation worked well.

KNN regression offered some of the poorest accuracy of all methods. This is not surprising. KNN is an algorithm that is sensitive to the number of features. It is a victim of the curse of dimensionality. The number of observations should grow exponentially with the number of features. The training set, after cleaning, included just 72 meaningful features to train the model, but just 1460 observations. That's just 20.27 observations per feature.

There are a wide array of dimensionality reduction approaches that can be used. They range from a simple technique like manual feature selection to a complex technique in PCA. Principal Component Analysis (PCA) is a means of reducing dimensionality of the data by using features that explain the variability in the data. Applying a 4-component principal analysis to the data reduced the error significantly.

### 6.2 BOOSTING

The performance drop from basic linear models to decision trees was noticeable. This was hardly surprising. Decision trees are not the most accurate modeling technique. They do serve as a basis for more advanced, accurate regression models. In the random forest model, an average of decision trees of sub samples is taken. The process of selecting the features to split decision tree nodes at random improved performance by a decent margin.

Decision tree boosting provided the expected performance improvement on bagging techniques. The general framework for boosting models was the same across the different implementations used. The idea is that decision trees are grown sequentially, training weak performers in each iteration, eventually culminating in a model that averages to a high score. Interestingly, there was a difference in score between Adaboost and the other two variants. Gradient boost updates each iteration based on gradients, as opposed to weights in Adaboost (Dangeti). XGBoost is gradient boosting with an extra layer of protection from overfitting in the form of a regularization term (xgboost developers). The performance difference between the latter two datasets was marginal.

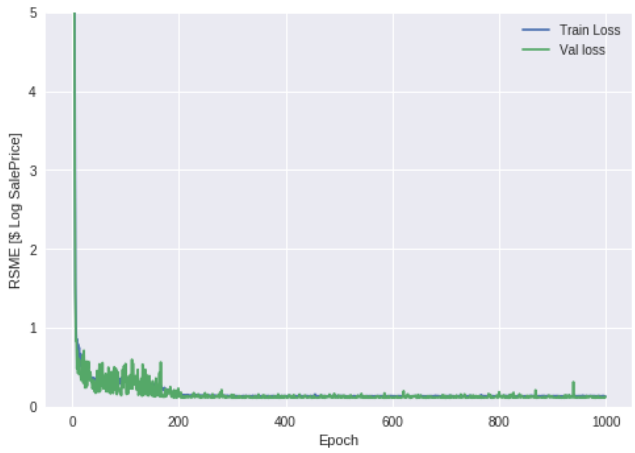
While gradient boosting offered the best score overall, the difference between boosting and a basic regression models was relatively insignificant. The reason for this is likely the data cleaning that was accomplished. The score for boosting didn't get worse after cleaning. Treating outliers and addressing improved performance to the extent that the linear regression model became optimized for the dataset.

One aspect that was consistent across modeling techniques was the limited effect that feature scaling had on performance. This is deceiving, however. When applied to new data, the unscaled gradient boost model trailed the scaled iteration by nearly .05, which is a significant number in this competition.

### 6.3 NEURAL NETWORK

Neural network performance with the built-in scikit-learn package ranged from poor to good. Performance was primarily dependent on the activation function used. Through a process of trial-and-error, it was discovered that the use of the relu function resulted in porous performance. This may be a result of the nature of the data. Reflecting on the difference between KNN and linear regression performance, the shape of the function of the data can explain the difference. Linear regression performed better because the function is linear. The same logic applied to the neural network. The relu function is usable for linear data, but works better when the data is non-linear.

The initial scikit-learn neural network score with the relu activation function motivated the use of an additional deep learning technique. There are numerous deep learning packages available for use. For this project, TensorFlow was chosen given its compatibility with Google Colaboratory. Google Colab has the package built-in to their collaborative Python editor. TensorFlow also interacts well with commonly used Python packages, Pandas and Numpy.




TensorFlow model performance over epochs

In the process of building the candidate models, it was discovered that both neural network packages were incredibly sensitive to unscaled features. The models offered scores with negative accuracy, indicating an issue with underlying data.

Initial attempts at using TensorFlow involved porting the parameters from the best Scikit-learn neural network score and applying them in the TensorFlow model. The TensorFlow model was built using 3 sequential Keras layers. Like the neural network built in the scikit-learn model, the 2 hidden layers were of size 64. An output layer of size 1 provided for the continuous variable output. The model was fitted using the same 75%-25% test-train split and the number of epochs, or iterations, used was 1000. An alpha parameter, or l2 regularizer, of .1 was applied.

The large number of epochs used can result in overfitting (Chollet). L2 regularization helps prevent that, but an extra layer of protection was supplied in the form a stopping parameter. If no performance improvement was noticed through any given range of 200 observations, the fitting would stop.

Performance was maximized using the aforementioned parameters. Adding layers can result in overfitting if the training data isn't sufficient (Chollet). Reducing a layer didn't help performance either. Changing the node size in the hidden layers didn't have a noticeable impact. Changing the activation function did significantly hurt performance for the reasons mentioned earlier.

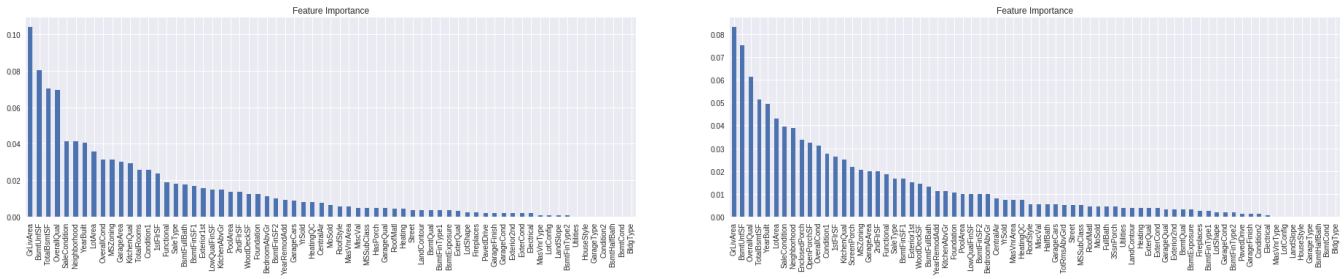
### 6.4 THE IMPACT OF FEATURE SELECTION AND FEATURE ENGINEERING

In searching for improvement on the best score, dimensionality reduction was explored in the form of feature selection. If the number of unimportant features were to be stripped from the model, an improvement might be possible.

The approach used to filter the features was to utilize correlation and variance. Correlation was chosen given how important the highly correlated variables turned out to be when feature importance was calculated for applicable techniques. Variance was chosen so as to explore the minimization of features with outliers in their distributions.

Interestingly, performance degradation was noticed across the board, with the exception of KNN. KNN is not surprising given that reducing the dimensionality of the dataset can increase performance. As discussed in the related work, removing features, even if they're not correlated, involves the risk that relevant features are lost (Chandrashekar and Sahin).

Feature engineering was attempted on a small scale. It was limited to just a few features that had generally been unimportant in the modeling of all features. In lieu of having several different porch features, the variables were combined into a catch all, 'HasPorch' feature. Not every home has a porch. If a value is present for any of the porch related variables, a 1 was filled in for the 'HasProch' variable and 0 if not. One other created variables was that of 'TotalRooms.' Bathrooms are not counted toward 'TotalRmsAbvGrd.' A variable to measure the total number of bathrooms was created, as they're split into two separate variables in the broader set to measure full and half rooms in the original set. The temporary variable was added to create 'TotalRooms.'




Feature importance for gradient boost with feature scaling before (left) and after (right) feature engineering.

Performance with the engineered features improved marginally with linear regression, but not for other variables. The process is very much trial-and-error.

For gradient boost with all features, 'EnclosedPorch' and 'OpenPorchSF' were important features for the model. The engineered feature 'HasPorch' was not.

'TotalRooms' became more important, but that didn't improve the score. Feature engineering has value when a tenth of a point is important, but the extent to which

when evaluated for time is questionable, at least for this dataset.

## 7. BEST MODEL

The best model was gradient boosting using all features. The prediction RSME was .12864, good for position 1572 in the rankings at the time of submission.

## 8. FUTURE WORK

This work attempted not to incorporate the Kaggle kernels provided by other competition participants. It was only after the score maxed out were those user generated approaches explored. In doing so, overlap in certain aspects of our approach was noticed. What was not attempted in this project that others had success with was feature stacking. That involves taking the prediction of many different modeling techniques and averaging them for the final prediction.

Applying these techniques to our contributions, regarding more micro data clearing and TensorFlow neural network optimization could result in improved performance.

## INDIVIDUAL CONTRIBUTIONS

Both team members worked collaboratively and equally. Jay did the data cleaning, developed the modeling functions and explored TensorFlow. Diyva addressed skewing, feature selection and feature engineering. Both participants contributed equally in the literature review.

## ACKNOWLEDGEMENTS

The code and underlying theory learned in Machine Learning taught by Dr. Abdelhakim was essential in developing this approach. The tutorial provided in the TensorFlow documentation cited in a prior section was instrumental in our understanding of how to use the package in the regression context and provided the framework for our implementation.

## REFERENCES


Abedin, Menhazul M.D., et al. "Accurate Diabetes Risk Stratification Using Machine Learning: Role of Missing Value and Outliers." *Journal of Medical Systems* (2018). <Accurate Diabetes Risk Stratification Using Machine Learning: Role of Missing Value and Outliers>.

Chandrashekar, Girish and Ferat Sahin. "A survey on feature selection methods." *Computers & Electrical Engineering* 40.1 (2014): 16-28. <https://www-sciencedirect-com.pitt.idm.oclc.org/science/article/pii/S0045790613003066>.

Chollet, François. *Predict house prices: regression* . 20 November 2018. <https://www.tensorflow.org/tutorials/keras/basic_regression>.

Dangeti, Pratap. *Statistics for Machine Learning*. Packt Publishing Limited, 2017. <https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781788295758/4/ch04lvl1sec34/comparison-between-adaboosting-versus-gradient-boosting>.

Feng, Changyong, et al. "Log-transformation and its implications for data analysis." *Shanghai Arch Psychiatry* 26.2 (2014). <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4120293/>.

Kaneko, Yuta and Katsutoshi Yada. "A Deep Learning Approach for the Prediction of Retail Store Sales." *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (2016). <https://ieeexplore-ieee-org.pitt.idm.oclc.org/document/7836713>.

Mu, Jingyi, Fang Wu and Aihua Zhang. "Housing value forecasting based on machine learning methods." *Abstract and Applied Analysis* (2014). <http://go.galegroup.com.pitt.idm.oclc.org/ps/i.do?p=AONE&u=upitt_main&id=GALE|A426543755&v=2.1&it=r&sid=summon>.

xgboost developers. *Introduction to Boosted Trees — xgboost 0.81 documentation*. 1 December 2018. <https://xgboost.readthedocs.io/en/latest/tutorials/model.html>.

Yu, Li, et al. "Prediction on Housing Price Based on Deep Learning." *World Academy of Science, Engineering and Technology - International Journal of Computer and Information Engineering* 12.2 (2018): 10. <https://waset.org/publications/10008599/prediction-on-housing-price-based-on-deep-learning>.

Zurada, Jozef, Alan S. Levitan and Jian Guan. "A Comparison of Regression and Artificial Intelligence Methods in a Mass Appraisal Context." *Journal of Real Estate Research* 33.3 (2011): 38. <https://www.researchgate.net/publication/254425990_A_Comparison_of_Regression_and_Artificial_Intelligence_Methods_in_a_Mass_Appraisal_Context>.