Final Project: House Price Predictor Using Machine Learning

By Kokil Dhakal and Dan Hoogasian

1. Description of Problem

Our project was based on the house price predictor dataset. The goal was to be able to predict the price of houses based on their attributes using machine learning. This service would be used by people looking to know the price of a house for a variety of reasons.

Real estate business has a great impact on the national economy. That is why predicting house prices has great significance for buyers, sellers, real estate agents and other related economic professionals. A good house price prediction model is also important for builders, investors, and tax assessors. To do this we used machine learning techniques that are widely used for price prediction, which have remarkable advantages over traditional methods.

2. Description of Data and Pre-processing

The dataset used has 2919 instances of 81 columns. One column contains the ID numbers, and another has the sale prices. The other 79 columns contain attributes. The attributes are both qualitative and quantitative, and are things such as year built, number of half-baths, kitchen quality, and type of lot. The data included a file briefly describing the attributes and what the possible values of each attribute could be. The houses were in the small city of Ames, IA.

The data we received was raw data, which is not ideal for analysis, so it required significant cleaning. Some attributes like length of house street, type of alley, fireplace quality, pool quality, fence quality, and miscellaneous features, had large portions of the data missing. For convenience, we dropped these attributes before analysis since they also had little determination on the sale price. After dropping these columns there are still columns that have NaN values which needed to be fixed before doing analysis. For convenience, we dropped all NaN values instead of imputing null values.

The data needed a good amount of preprocessing before it could be analyzed. Some of the attributes were not applicable to a large portion of houses, so they had zero or no values. The qualitative attributes were converted to a numerical representation for the analysis. Outliers were determined as being more than three standard deviations from the mean of each attribute and were removed for analysis. This included removing the nineteen houses that sold for more than $426,000 and one that sold for under $35,000. Once the data was ready for processing, we were left with a seven by 1,350 cleaned data set.

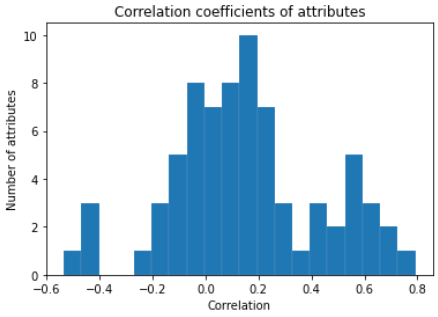
Commented [DH1]: Might not be due to what I said

3. Basic Methods and Techniques

Many of the data attributes didn't have a significant determination on the sale price, so for our first extension idea, we used a linear regression model function from the scipy module to determine the correlation each attribute had on the sale price. In this model, it is called the R-value or the Pearson correlation coefficient. There are many ways to calculate the correlation between variables, but we used one of the most standard techniques here. It has a range of -1 to 1, with zero being no correlation, 1 being a perfectly positive correlation, and -1 being a perfectly negative correlation. We set the thresholds for significance at -.6 and .6. This gave us the following six attributes to use in designing our machine learning model: overall quality, exterior quality, above ground living area square footage, kitchen quality, size of garage (in car capacity), and garage square footage. Some of these features might not be significant in most parts of the county, but due to the mid-west location where land is in abundance and there is little public transportation, these features were more important.

We used the matplotlib module to get basic plots of the data. Figure 1 shows a histogram of the correlation coefficients.

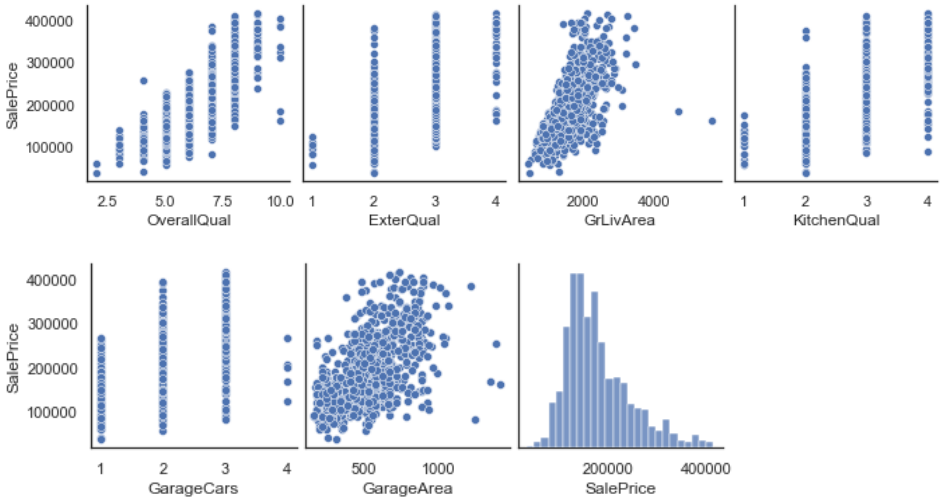
Figure 1



Histograms are a effective way to visualize continues data, which is what our predicted values were. When looking at a histogram of the R-values, it holds a mostly normal distribution, with more values being slightly more positive than negative.

Figures 2 through 8 show the scatter plots for each attribute compared with the sale price, which are all sloping upward, as we would expect. Scatter plots are an effective way to visualize the data when you initially look at it.

Figures 2 through 8



The sklearn module was used for the machine learning. We used train_test_split from the model_selection function package to divide our data into two-folded sets. We used a 30 percent versus 70 percent test and train split, respectively. It is important to split the data into appropriately sized test and train sets because you want to make sure you have sufficient data for both parts. It is a bad idea to test on the same data you used for training, which is called data leakage. This causes your model to be unrealistically accurate. We looked at the accuracy scores of this and it proved to be immensely true.

Many of our attributes had values that were of different types, which meant their nominal values were not comparable. We used some statistical functions from sklearn to calculate a scaled score of all the attributes, which normalized the values to comparable numbers.

4. Extensions, and Remaining Methods and Techniques

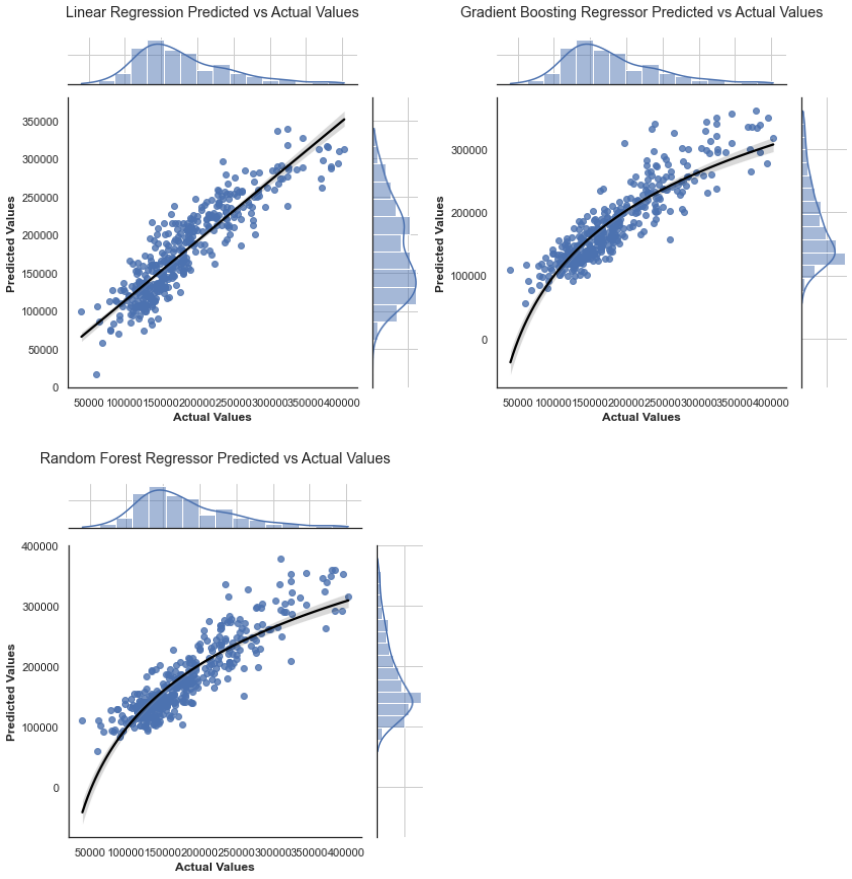
The first algorithm we used for predictions was the linear regression algorithm. The accuracy of our algorithm on the testing data was 80.1% with a mean squared error of 891,000,000. We used a graph from the seaborn module to visualize the predicted versus actual outcomes, showcasing our model's accuracy and lack of outliers.

Secondly, we used the gradient boosting algorithm from the ensemble package of sklearn. The accuracy was slightly better that the linear regression's, usually at 81.3% with a mean squared area usually between 833,000,000 and 835,000,000. We used the same plot as we did with the linear regression model to visualize the data, but this time the data looked

more accurate in the lower range of values than the higher range. It also held a curved line of best fit, as we would expect from a non-linear model.

Lastly, we used the Random Forest Regressor algorithm. The accuracy was around 81.5% with a mean square error of 829,000,000. The last two algorithms had versions for clustering with unsupervised learning, but since our data was discrete and we were using supervised ML, we used the regression versions.

Figures 9 through 11



We used bar plots from the seaborn module in conjunction with the standard matplotlib module to visually compare the mean squared errors and accuracies. Linear regression models typically work better with less complicated or less detailed datasets, so we expected this model

to do worse than it did. This was likely due to our small selection of attributes. Though, it did have a slightly lower accuracy.

Researching the different algorithms for this extension idea proved how many different algorithms there were. We were also considering implementing KNN and neural network algorithms, as they also looked applicable to our data, but we chose only these three as they seemed to be the best fit. It was interesting to see the varieties of all the algorithms.

Although we might have had better luck with different algorithms, what we used appeared fairly accurate. Although it was surprising to see such similarity in results from all three.

5. Conclusion and Concepts Learned

One thing we learned is that there is much work needed to preprocess the data. For example, converting the data from categorical to quantitative data is complicated and time consuming. Randomly assigning numbers to categorical elements that have order is time intensive, so this was only partially done. If this was done to the data in its entirety, we would have had a better selection of attributes and our models would have been more accurate. We also learned dealing with missing values is time consuming, but also essential.

When looking at the many attributes the data has, initially it is hard to understand and somewhat overwhelming. Getting good visualizations of the data using matplotlib and seaborn was helpful, but there was still more work that could be done to explore correlations and patterns.