Patrick Oline

Capstone Final Report:
Predicting Housing Prices

Problem Statement
The goal of this project is to create a machine learning model that will help predict housing prices based on the most relevant features of a house.

The data set, which is pulled from kaggle.com, seems to focus on housing data from Ames, Iowa. The focus of the analysis will be on using the provided information to create a machine learning model that will predict housing prices based on the most relevant variables within the data set. I will try a number of different ML methods, varying the parameters/hyperparameters for two different machine learning models to see if I can optimize my results.

I look to inspect and clean the data first, keeping relevant columns and removing ones that do not seem to have any correlation with pricing. Then I will train a ML model using the training data, and finally, apply the model to make price predictions on the test data.

In the end I hope to have a model that can have an accuracy for about 10-20% difference from the actual prices of the houses.

Data Wrangling
I will be using data from the kaggle.com website. This data has already been split into a training and test set, but I will focus on using only the train set for my work in this project.

The data has 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa. While there were a lot of features, there was also a lot of missing data. First, I decided to drop any numeric columns with more than 5% missing values. Second, I did the same for any text columns. Finally, the remaining columns with missing information were only ordinal columns. For these, I replaced the missing value with the most common value in that feature.

Before moving into exploring the data, I created two columns. The first column was a calculation of ‘Built_to_Sale’, which is the difference between the sale year of the house and the year the house was constructed. The second feature I created was ‘Years_Since_Remod’ , which gives us the number of years it has been since the house has been remodeled.

Finally I dropped the following columns because they were either no longer needed, irrelevant, or leaked data regarding the sale price:
["Id", "YearBuilt", "YearRemodAdd", "MoSold", "SaleCondition", "SaleType", "YrSold"]

Exploratory Data Analysis

When first investigating the data I noticed that it has a slight skew to the right with a few sale prices acting as outliers. The histogram below shows the layout of the data spread over 15 bins, and has the mean (red dotted line) and standard deviation (black dotted line) graphed on top of the histogram.

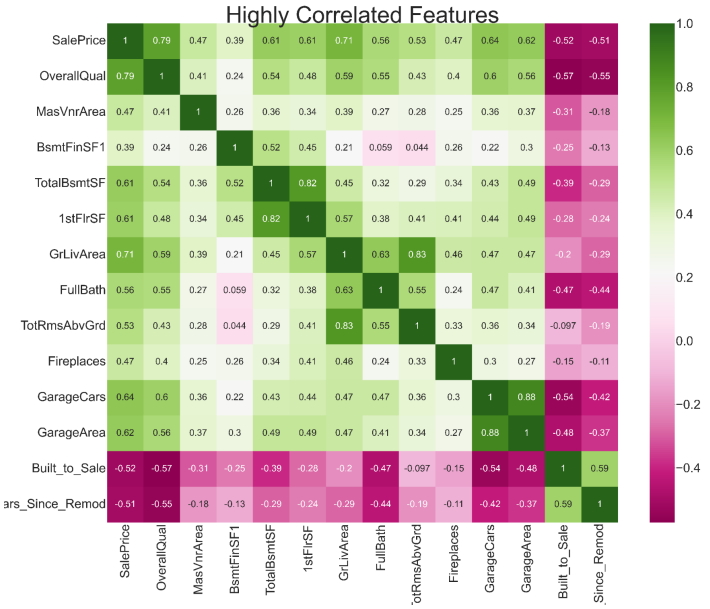
mean = $180,921.20 *STD = $79,442.50* *STD Boundaries = ($101,478.69, $260,363.70)*

There were so many columns I found it difficult at first to look at them all. I decided to break the columns into groups and observe the relationship to 'SalePrice'.

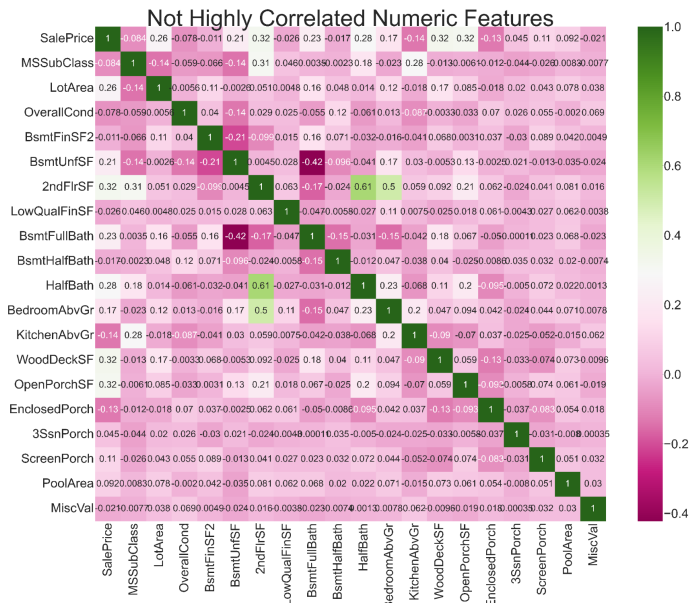
Here is the group breakdown:
1) *house_corr* = Columns with a correlation to '*SalePrice*' > 0.39 or < -0.5 (all numeric)
2) *house_int* = Remaining int64 columns not in *house_corr* (all numeric)
3) *house_obj* = Remaining object columns not in *house_corr* (all categorical)

In-Depth Analysis

Below is a breakdown of the correlation for each numeric column with the *'SalePrice'* column.



house_corr

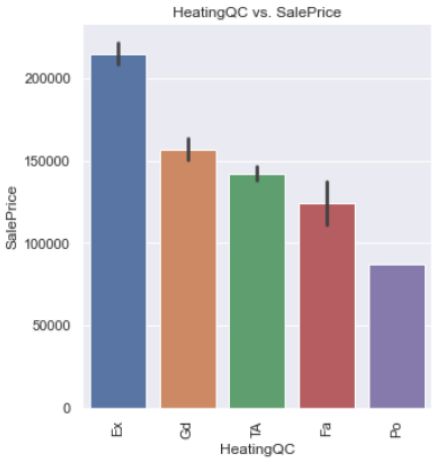


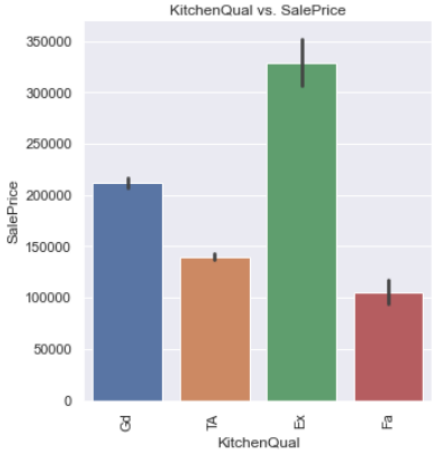
house_int

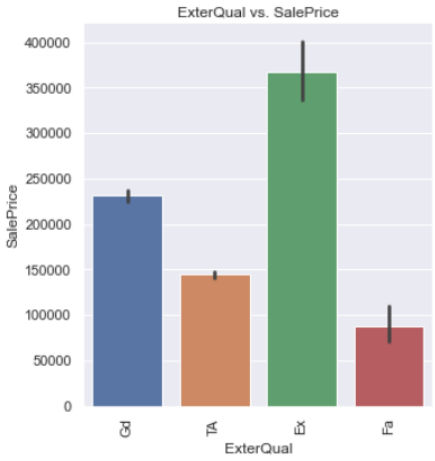
Looking at the correlations as two separate charts is much more palatable. While we will focus on only the top row of each chart, is it important to see if any of the columns have a high correlation to each other because that could create collinearity which would affect our machine learning model. It appears that the only *'GarageCars'* and the *'GarageArea'* might fall into the collinearity category having a correlation of 0.88, which is very high. (This makes sense, having more Garage space means you can have more cars in the Garage). I have decided to keep both categories, and as an idea for future improvement I will consider removing one or both of those features.

Moving into the categorical columns, *house_obj* dataframe, I decided to create bar charts for each of the features, comparing the mean of the *'SalePrice'* for each category within that feature. By comparing the items within each feature I was looking to see if certain features tended to have higher or lower sale price values. Some of the more interesting bar charts are pictured below.

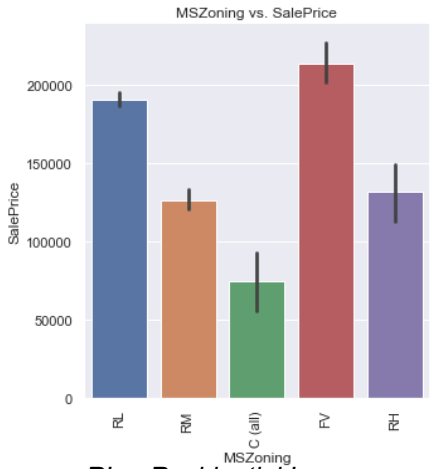
Features vs Mean Sale Price Charts

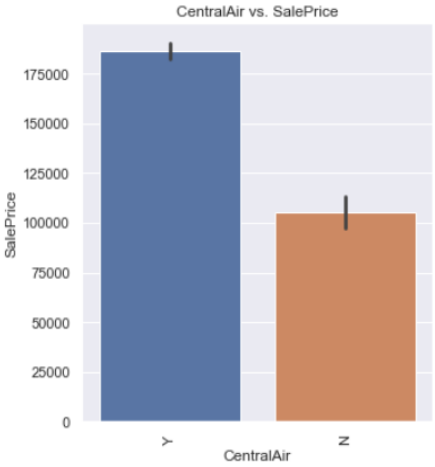


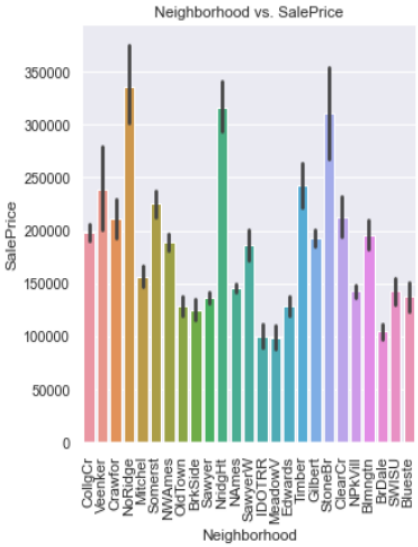




Ex = Excellent *Gd = Good* *TA = Average/Typical* *Fa = Fair* *Po = Poor*







RL = Residential Low
RM = Residential Medium Density
C(all) = Commercial
FV = Floating Village Residential
RH = Residential High Density

Y= Yes *N=No*

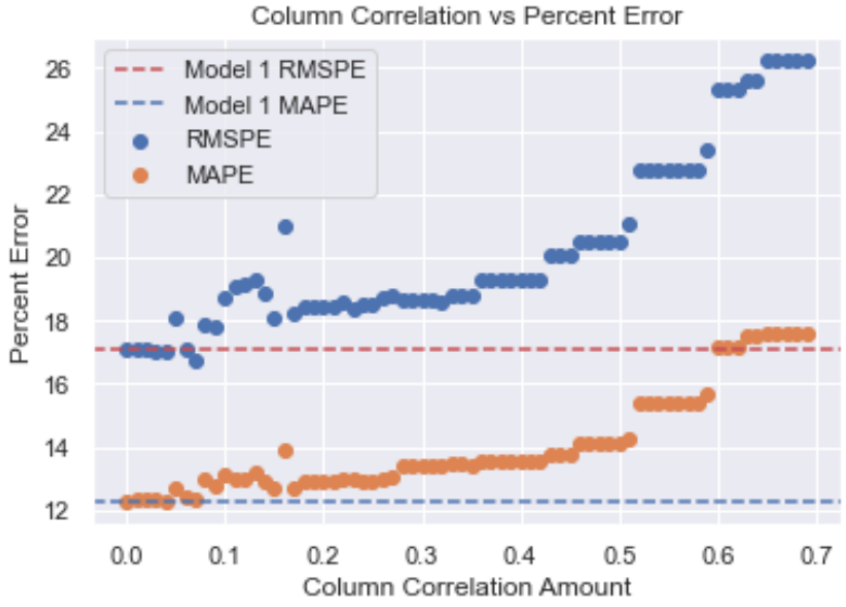
Looking at these charts, it is clear that the sale price is much higher as the quality of the heating, kitchen, and exterior increases. Also, the sale prices increase quite a bit as population density lowers, while medium and high density zones are very similar in pricing. The addition of having central air increases the average price of a house by around $75,000. The final chart is a breakdown of specific neighborhoods, and it is interesting to see the varying prices across these neighborhoods.

Model Selection
For my model selection, I decided to try two different machine learning models, Linear Regression and Random Forests. For each of these I tried a couple different methods to try and optimize the parameters/hyperparameters and attain the smallest percent error. I used two error measurement methods, the root mean square error and the mean absolute error. At the end of this section there is a table with the listed errors and percent error.

Linear Regression Models
First, I ran a straight forward Linear Regression Model without adjusting any hyperparameters. The results came back with a **root mean square percent error of 17.09%** and a **mean absolute error of 12.30%**.

Then I tried to run a cross validation model using the same parameters, but for some reason the results came back as astronomical numbers. I decided to skip this model and move on to the next.

The third linear regression model I tried involved adjusting the number of features to train and run the model on. I found the correlation each feature had with 'SalePrice' and ran linear regression models with different correlation cut offs. I started at a correlation threshold of 0, and each time through the loop I increased the correlation threshold by 0.01. As the threshold increased, more and more features were dropped from the training/test sets. The chart below displays the results as the correlation threshold went from 0 to 0.7. It is clear that as features were removed the percent error (both RMSPE and MAPE) increased, making the model more unreliable.



min(RMSPE) = 16.72% *min(MAPE) = 12.25%*

Random Forest Model
First, I ran a straight forward random forest model without adjusting any hyperparameters. The results came back with a **root mean square percent error of 18.73%** and a **mean absolute error of 10.67%**.

In my second random forest model I used RandomSearchCV and looped through a variety of different hyperparameters to try and find the optimal set.

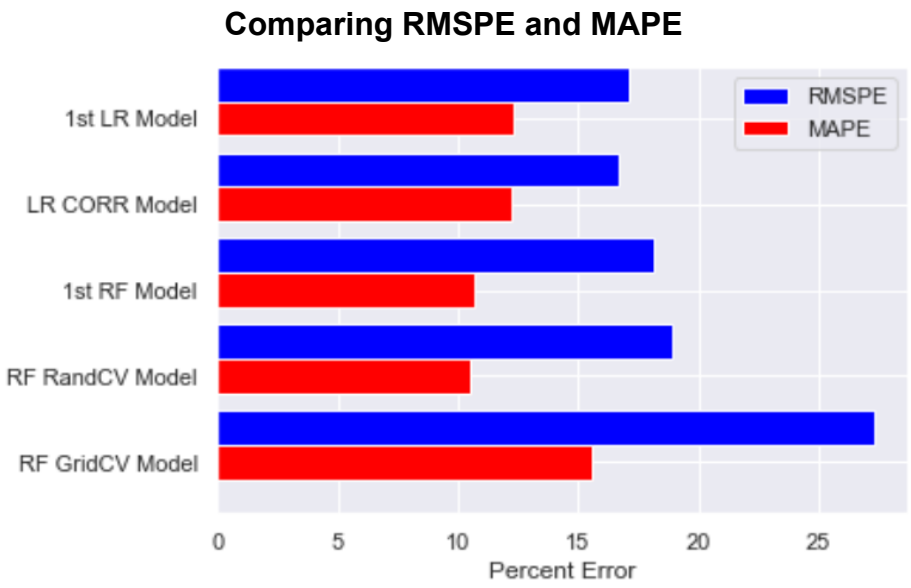
In my third random forest model I used GridSearchCV to try and find the best hyperparameters, similar to my method in the previous step.

The best parameters and percent errors for each model can be seen in the table below.

RandomSearchCV	GridSearchCV
n_estimators: 400 min_samples_split: 2 min_samples_leaf: 1 max_features: 'sqrt' max_depth: None bootstrap: False	n_estimators: 200 min_samples_split: 8 min_samples_leaf: 3 max_features: 3 max_depth: 90 bootstrap: True
RMSPE = 18.93%	**RMSPE =** 27.32%
MAPE = 10.50%	**MAPE =** 15.58%

Model Results

Name	RMSE	RMSPE	MAE	MAPE
1st LR Model	$34044.04	17.09%	$21260.01	12.30%
LR CORR Model	$34015.87	16.73%	$21254.74	12.25%
1st RF Model	$27763.13	18.18%	$17505.46	10.67%
RF RandCV Model	$28930.04	18.93%	$16803.02	10.50%
RF GridCV Model	$42705.14	27.32%	$24293.05	15.58%



Takeaways

In the linear regression models it appeared to be fine to remove a few of the less correlated features but if too many features were removed then the error steadily increased. It was clear that the mean absolute error in the random forest models was lower than the linear regression models. The random forest also produced a decent RMSE in 2 of the 3 models.

Using the optimal random forest model, I collected random data from the training set for each of the columns and ran a prediction for three different house sale prices. Some of the features and the predicted prices are below.

MSSubClass	LotArea	OverallQual	KitchenQual	Remodeled	Built to Sale	SalePrice
20	14000	4	Good	3 years	47 years	159,153.19
50	9135	6	Average	1 years	31 years	161,670.08
20	27650	5	Average	0 years	5 years	167,280.12

MSSubClass
20 = 1-STORY 1946 & NEWER ALL STYLES
50 = 1-1/2 STORY FINISHED ALL AGES

Future Research

For future research I could continue to collect data from housing in this area. It would be interesting to see how these models hold up over time, or if they would need to be adjusted and retrained.

Further investigating into local locations like neighborhoods, and how the model holds up when it is covering a smaller location with more similar housing features. It would also be interesting to see how accurate this model is in other housing markets around the country. This would require a lot of data collection in different regions, but would provide an interesting test to see how the different locations compare.

Another potential way forward would be to increase the accuracy of the model. We can continue experimenting with the hyperparameters of the functions, try other ML algorithms rather than the linear regression model and random forest, and apply some other approaches to feature engineering and selection.

As mentioned earlier, the *'GarageCars'* and the *'GarageArea'* fall into the collinearity category having a correlation of 0.88, which is very high. Both of these columns were kept in this project, but in future work it may be worth finding and removing the features with high collinearity.