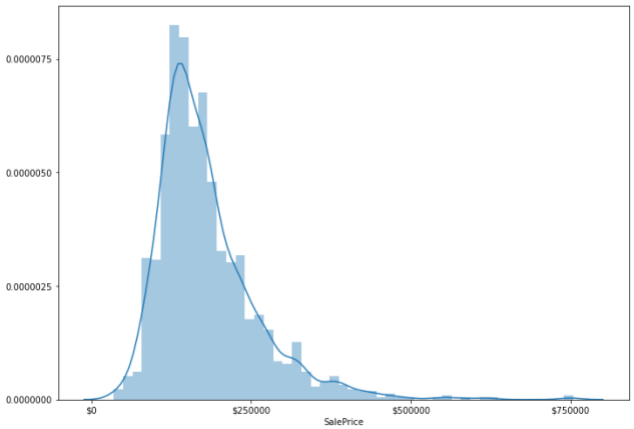
Wyatt Carter
BDA 411
Final Project Report

Data Preparation

First we needed to be able to read our data. To do this we import numpy, pandas and seaborn from sklearn in python. We also import matplotlib.pyplot from matplotlib. Doing this allows us to read our data and perform the necessary regression analysis. Once both the training and testing data sets are imported through our code, we can analyze the different types of variables each parameter is. All in all there are 79 descriptive parameters that could be used to estimate sale price. 79 is more parameters than necessary to build a regression curve. Because of this we can trim some parameters away, we do this by moving all variables that are not numeric. We can do this in python by separating parameters based on variable type. We only want numeric non-descriptive data so we pull only float64 and int64 parameters. This leaves us with 38 parameters, cutting our parameters in half but leaving plenty for regression and these variable types will be easier to code and understand.

Looking at Sale Price

Now that we have trimmed our parameters we begin to look at the target variable, sale price. We want to get a look at how sale price is distributed. To achieve this we build functions to give sample size, mean, median, standard deviation, skew and kurtosis. Sample size merely gives a look into how large our data set is, another reason trimming so many variables is acceptable. Our sample size is 1,460. Our mean sale price is $180,921.20, this gives a raw estimator of sale price. Median is $163,000.00. Median being less than mean, pretty significantly in this case with a 10% difference, means there could be outliers in our dataset with some sale prices being grossly larger than the rest. Our standard deviation is $79,442.50 showing just using the mean as a predictor would yield a large error. A skew of 1.8829 confirms the suspicion we have outliers of the data set, which we will deal with later on. Kurtosis of 6.5363 shows a sharp peak of the sales price data set. Here is a look at the distribution.

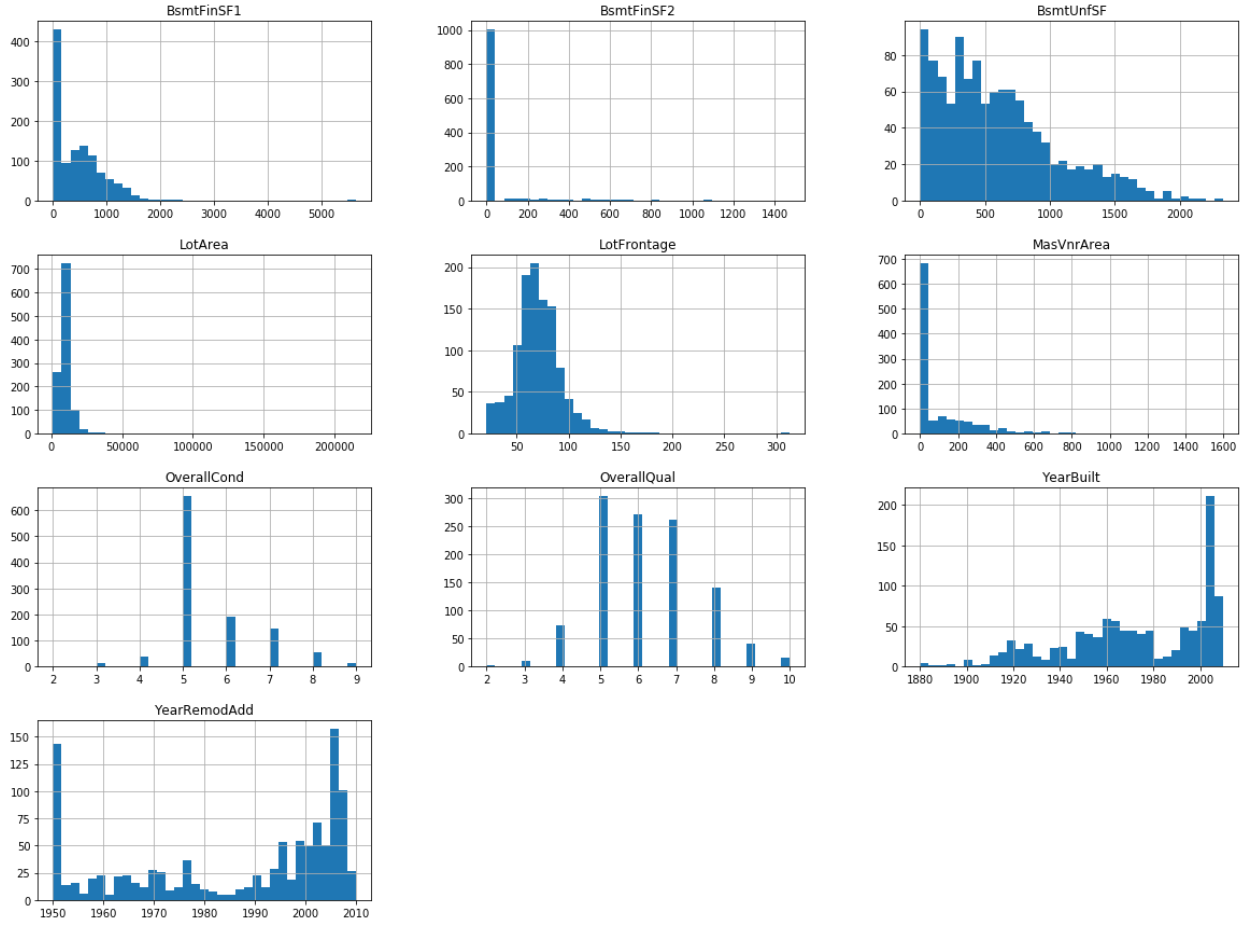


Data Cleaning

In order to clean our data we perform the steps discussed above, removing variable types that are not float64 or int64. In order to do this we create a new dataset from the old one. We achieve this with the code `df_train_num = df_train.select_dtypes(include=['float64', 'int64'])`. This creates a new array of the name 'df_train_num' with only the float64 and int64 datatypes from the previous array 'df_train'. Lastly to clean our data we remove any samples with `NaN` values. We update our data and from here on this is the only data we deal with until the testing data.

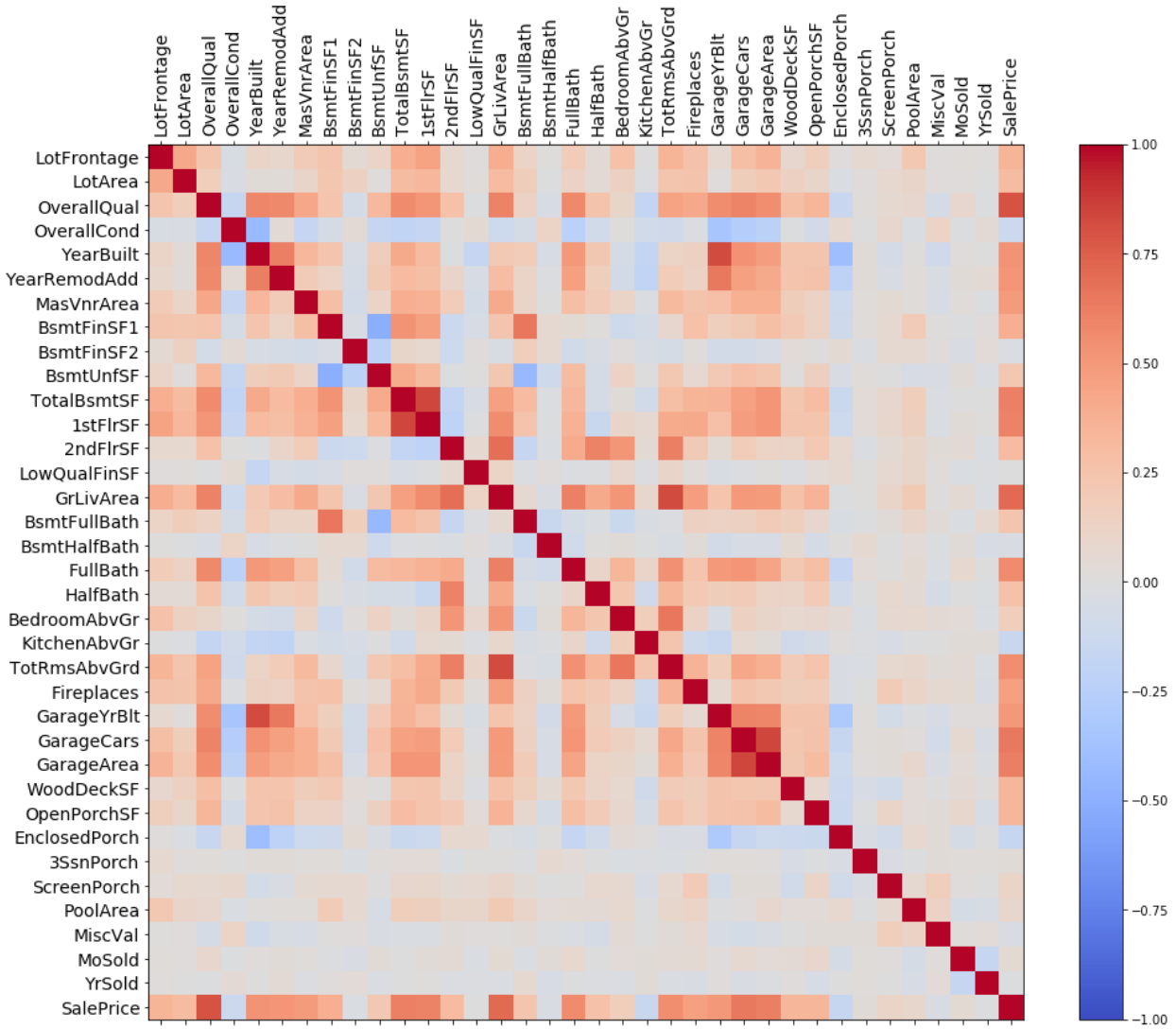
Data Exploration

Now we take a look as remaining parameters. First we look at the distribution of each parameter. Here are a few examples



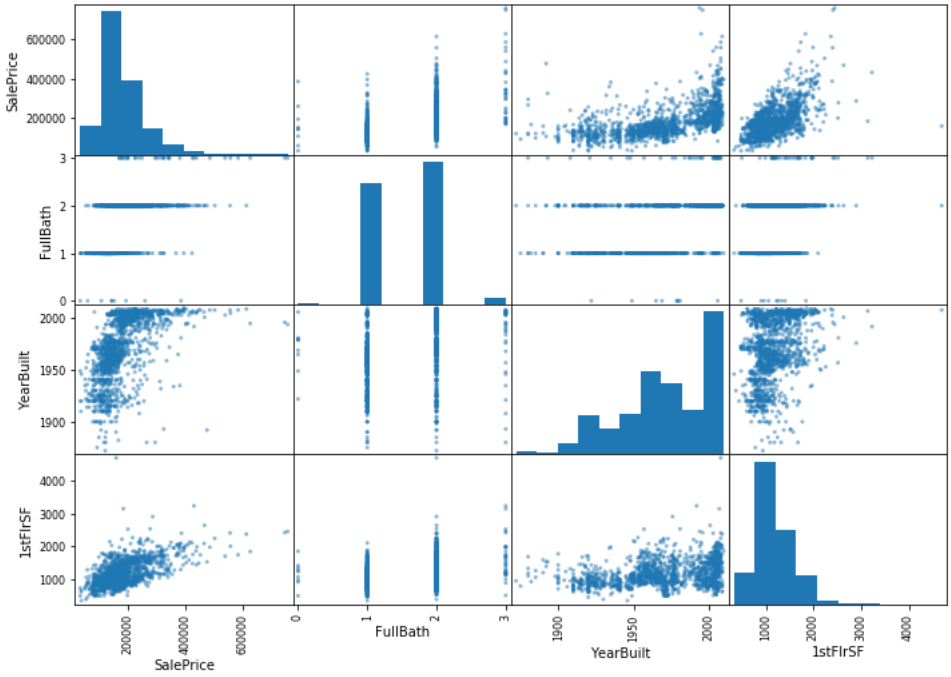
Most of the parameters are normally distributed. Year built is skewed but this is to be expected, with population growth. We also want to take a look at correlation between variables. Including variables that are correlated with each other leads to overparameterizing regression curves, basically giving too much weight to certain variables if they both effect the response (sale price) in the same way. We create a coolwarm map to show correlation

between variables.



Perfect correlation across the diagonal is to be expected. The correlation map shows not only correlation with each parameter but also sale price. Anything highly correlated with sale price would be a good parameter for our model. To get a closer look at a few of these parameters

with sale price we graph them with respect to each other.



Focusing on the top row each of the parameters has a relationship with sale price. Full bath can be seen with a slightly linear build, focusing on the middle of each set of points, the graph is spaced because full bath is a discrete variable type. 1st floor square footage shows a strong linear relationship at low square footage then the relationship weakens as the house size is increased. Year built seems to have an exponential relationship with sale price. This follows the 'new car' philosophy, meaning a car loses a large chunk of its worth when it is driven off the lot and the same philosophy holds true for houses.

First Shot at Regression

Our first regression curve we choose full bath, year built and 1st floor square footage. We choose these variables by inspecting the correlation with sale price based on the chart above. Running these using LinearRegression() we get an intercept of -1,417,463.30. The influence of full bath is 38,800.61, essentially meaning each full bathroom adds about $40,000 to your sale price. Year build and 1st floor square footage have estimates of 728.49 and 87.4 respectively. This model gives a root mean square estimate (RMSE) of $53,073.49.

Smarter Model = Gridsearch

Gridsearch offers a more streamlined approach compared to LinearRegression. First we import GridSeachCV from sklearn.model_selection. Running our previous model through gridseach we achieve a RMSE of $45,909.59. Gridsearch did not perform as expected but since it does all the heavy lifting for us and eases coding we continue to use it, but adjust our model. For our next model we choose the top five features from 'YearRemodAdd', 'YearBuilt', 'TotRmsAbvGrd', 'FullBath', '1stFlrSF', 'TotalBsmtSF', 'GarageArea', 'GarageCars', 'GrLivArea'

correlated with sale price. Gridsearch chooses the top 5 for us and gives a RMSE of $46,375.48. This is good but still has not beat our best model we manually selected.

Dropping Outliers

Now that we have a few linear models to choose from we go back to trimming our data set and rerunning our regression curves. Recalling our standard deviation is roughly $180,000 we choose to eliminate any samples with sales prices greater than three standard deviations from the mean. Running our model again with garage living space, year built and basement full bath as our parameters, these were chosen as the best fit based on the new data set, we achieve a RMSE of $39,029.86. This beats any regression we made before taking away outliers. We try again removing any sale prices greater than two standard deviations from the mean. The updated model gives a RMSE of $34,062.69, creating an even better model. Now we remove any outlying data using bisection. After all the removals, 39 in total, we achieve a RMSE of $27,737.95.

The Test

Now we put our model against the test data. Running our last model against test data we achieve a RMSE of $39,662.89. Removing outliers from the test data we obtain an updated RMSE of $22,052.16.

References

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.".