Housing Prediction King County – Zillow Take-Home Assignment
Usman Gohar
10-24-2020

1) Overview

In this project, the aim is to predict sale prices of houses in King County. I will be doing a comprehensive data analysis, feature engineering & model selection.

Note: All the steps for data pre-processing were repeated for testing set (Refer to notebook)

2) Data Preparation

In this section, I will be preparing the data by removing nulls & outliers since linear regression is highly sensitive to outliers.

Dataset

The dataset includes 24 variables that range from numerical, data and categorical data. We have a total of 11588 training rows of data for King County in the training set. Testing set has a total of 4402 testing rows. The figure 1 below is a small snapshot of the features in the database (refer to notebook)

	PropertyID	SaleDollarCnt	TransDate	censusblockgroup	ZoneCodeCounty	Usecode	BedroomCnt	BathroomCnt
0	48648941	285000.0	5/23/2015	5.300000e+11	R7	9	4.0	2.00
1	48648982	309950.0	8/22/2015	5.300000e+11	R8P	9	3.0	2.00
2	48649024	476000.0	8/27/2015	5.300000e+11	SF 7200	9	4.0	1.00
3	48649040	324950.0	7/1/2015	5.300000e+11	R1	9	4.0	2.25
4	48649057	325000.0	6/20/2015	5.300000e+11	LDR	9	4.0	1.75

Figure 1: Snapshot of Data

Missing Values:

It is very important to determine the missing values in the data so we can either remove those or replace those with suitable values. Removing data can lead to loss of information but replacing nulls with suitable values (mean, median) can also introduce noise. Figure 2 shows us the number of missing values for each feature in the training set.

	Total	Percent
ViewType	8956	77.29
GarageSquareFeet	2841	24.52
BGMedRent	2631	22.70
BGMedYearBuilt	247	2.13
BGMedHomeValue	6	0.05

Figure 2: Missing values & Percentage

Initial Observations:

- Generally, most of the variables have few or no missing values except the ones shown in table 2.
- Most of the variables in the data are continuous. However, there are categorical variables as well (**ViewType**, **ZoneCodeCounty** etc.)
- **ViewType** & **GarageSquareFeet** are actually not missing values. They simply indicate the absence of a View & Garage respectively.

Pre-Processing (Missing Values):

- **GarageSquareFeet**: The null values in this column are there for a purpose; they indicate that there is no protected garage in the house. We replace these null values with 0
- **ViewType**: The null values here indicate absence of view so we replace them with 0
- The rest of the null values are simply replaced with the mean of the columns.

Now there are no missing values in the dataset. The same process was repeated for the test case.

3) Exploratory Data Analysis

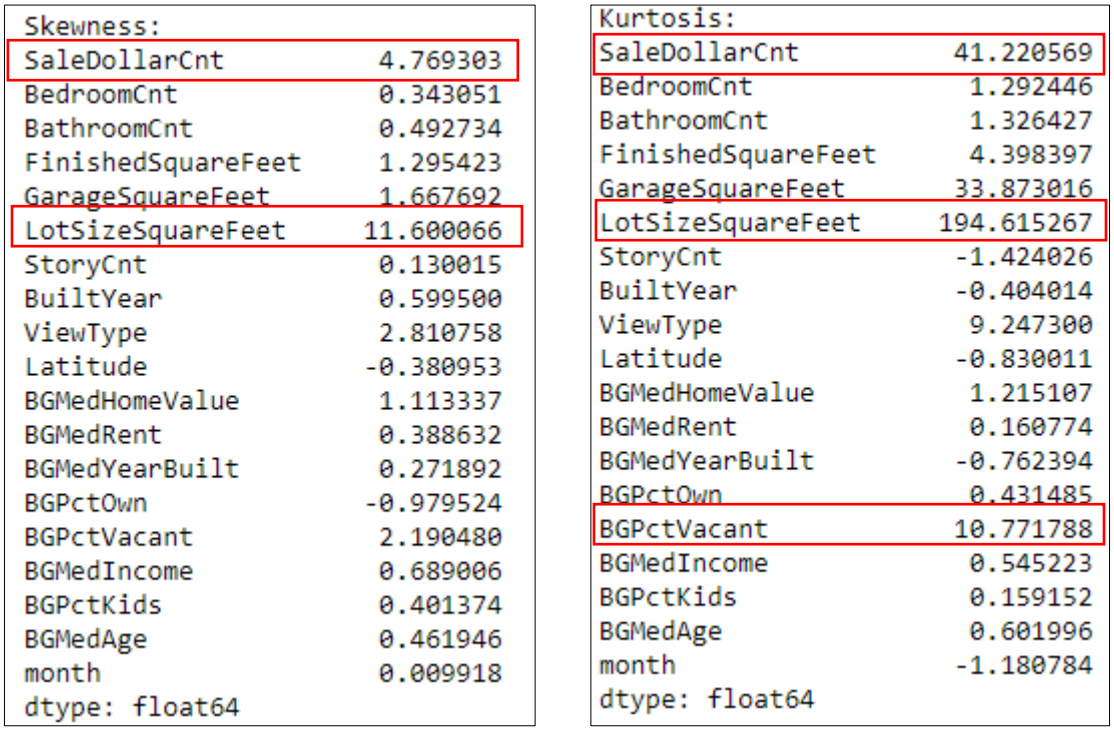
In this section, I will explore the data in more detail to extract important information and visualize the data.

Since we will be applying a multiple linear regression model as one of our solutions, it is important to first check whether the data is normally distributed. That is one of the assumptions of linear regression.

Normal Distribution Assumption/Data Skewness

One of the most important pre-processing tasks is to ensure our data is normal. More reliable predictions can be made if the data follows a Gaussian Distribution. I use two measures to test the distribution of the data i.e. Kurtosis & Skewness. Kurtosis is simply a measure to determine how the tails of a distribution vary from the tails of a normal distribution. On the other hand, skewness tells us how asymmetric the probability distribution is for a set of data.

In the figure below, I show the results of these statistics on the complete dataset.



Skewness:	
SaleDollarCnt	4.769303
BedroomCnt	0.343051
BathroomCnt	0.492734
FinishedSquareFeet	1.295423
GarageSquareFeet	1.667692
LotSizeSquareFeet	11.600066
StoryCnt	0.130015
BuiltYear	0.599500
ViewType	2.810758
Latitude	-0.380953
BGMedHomeValue	1.113337
BGMedRent	0.388632
BGMedYearBuilt	0.271892
BGPctOwn	-0.979524
BGPctVacant	2.190480
BGMedIncome	0.689006
BGPctKids	0.401374
BGMedAge	0.461946
month	0.009918
dtype: float64	

Kurtosis:	
SaleDollarCnt	41.220569
BedroomCnt	1.292446
BathroomCnt	1.326427
FinishedSquareFeet	4.398397
GarageSquareFeet	33.873016
LotSizeSquareFeet	194.615267
StoryCnt	-1.424026
BuiltYear	-0.404014
ViewType	9.247300
Latitude	-0.830011
BGMedHomeValue	1.215107
BGMedRent	0.160774
BGMedYearBuilt	-0.762394
BGPctOwn	0.431485
BGPctVacant	10.771788
BGMedIncome	0.545223
BGPctKids	0.159152
BGMedAge	0.601996
month	-1.180784
dtype: float64	

Figure 3: Skewness & Kurtosis Measure

A Kurtosis value of greater than or less than 3 shows that the data is not normally distributed. Secondly, a skewness value of greater than or less than 0 shows skewness in the data. Based on these findings, I will attempt to fix this using transformation.

First, let's visualize the data for "**SaleDollarCnt"**

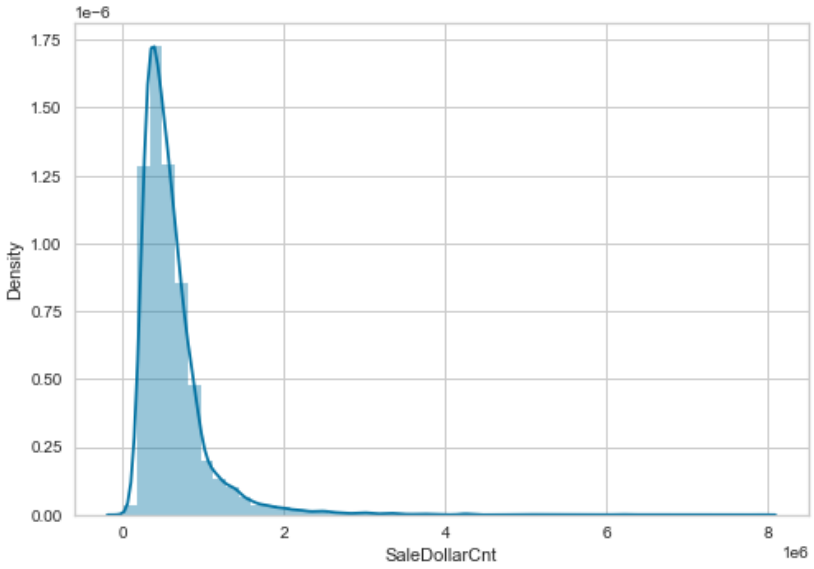


Figure 4: Probability Distribution of SaleDollarCnt

The figure 4 reaffirms our earlier test results. More importantly, the plot tells us that the "**SaleDollarCnt"** is positively skewed (Longer tail to the right). To fix that, we use the log transformation. We use a Log transformation when the data has few large values as in the case of **SaleDollarCnt**

Figure 5 shows the result of the transformation. The distribution for **SaleDollarCnt** looks more like a normal distribution now. Figure 6 also compares the probability plots before and after the transformation

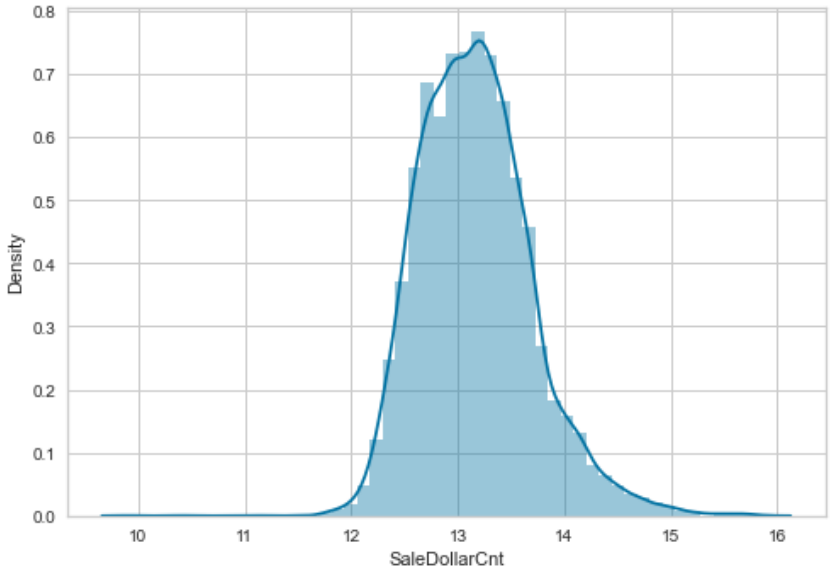


Figure 5: After Log Transformation

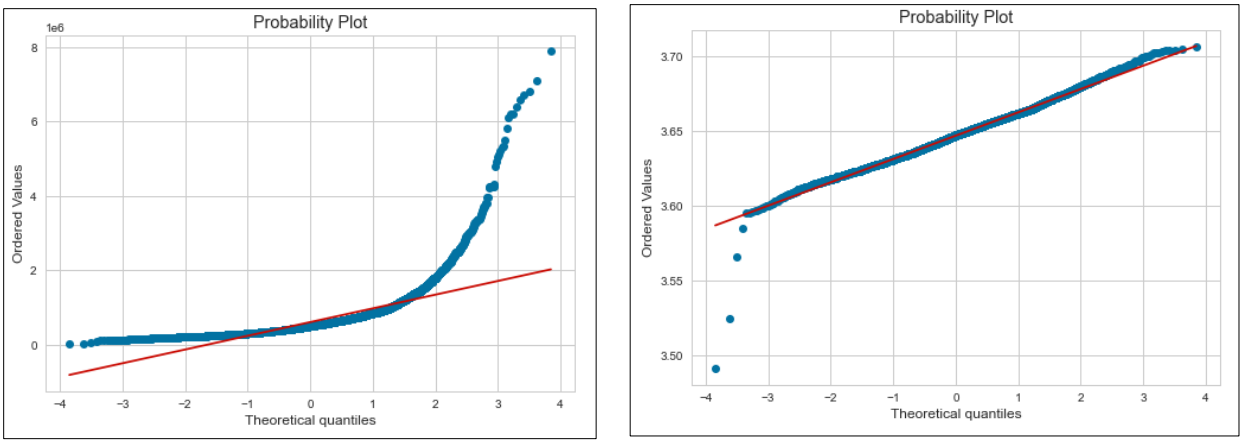


Figure 6: Comparisons of Probability plot (Before vs After)

The same process was now repeated for all the independent variables. Below are sample probility distribution graphs for LotSizeSquareFeet before & after Box-Cox Transformation. Log Transformation is actually a type of Box-Cox transformation. For the rest of the data we use a box-cox transformation to fix the skewness in data

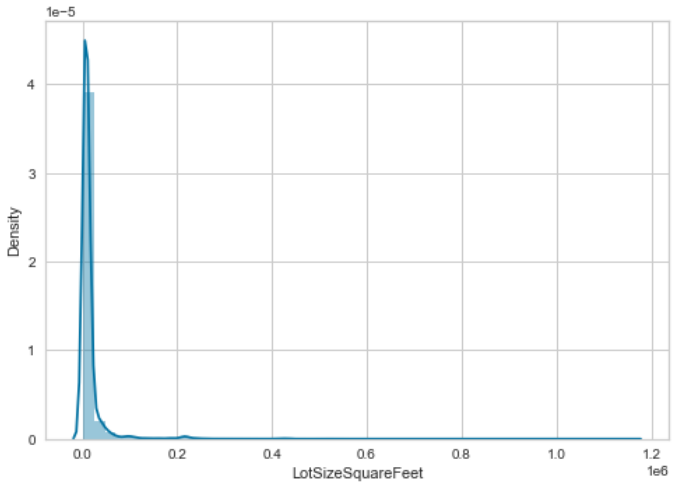


Figure 7: Before Transformation

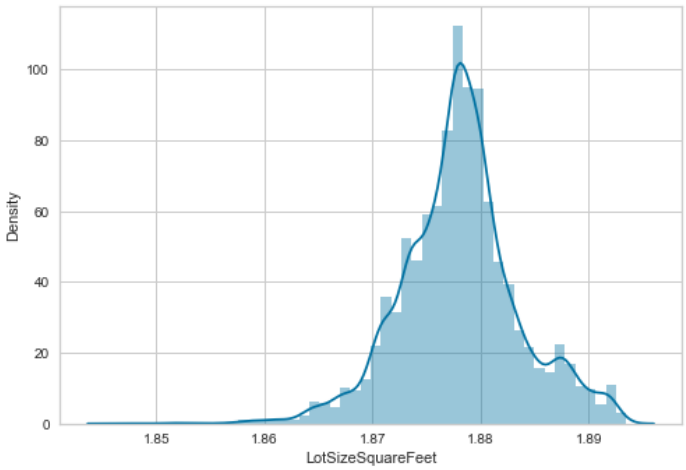


Figure 8: After Transformation

Correlation between features:

Now that the data is clean & ready, next step is to find the features that can be useful for us to predict the target variable. I do this both visually & statistically.

The following image shows us the correlation with the predictor variables in a descending order

```
BGMedHomeValue        0.681865
FinishedSquareFeet    0.678446
BathroomCnt           0.506672
BGMedIncome           0.427578
Latitude              0.317772
BedroomCnt            0.310897
StoryCnt              0.267300
ViewType              0.249966
GarageSquareFeet      0.245984
BGMedRent             0.229267
BGMedAge              0.173956
BGMedYearBuilt        0.121045
BGPctOwn              0.094028
LotSizeSquareFeet     0.067874
BGPctVacant           0.010543
month                -0.010586
BGPctKids            -0.028768
BuiltYear            -0.139941
Name: SaleDollarCnt, dtype: float64
```

Figure 9: Correlation values

From figure 8, we can see that **MedianHome** value, **FinishedSquareFeet** & **BathroomCount** are highly correlated with our target variable. **Median Income**, **Latitude**, **Bedroom Count**, **Garage Square feet**, **Story Count** & **Median** rent are slightly correlated.

Next, we confirm these results by visualizing a couple of these features.

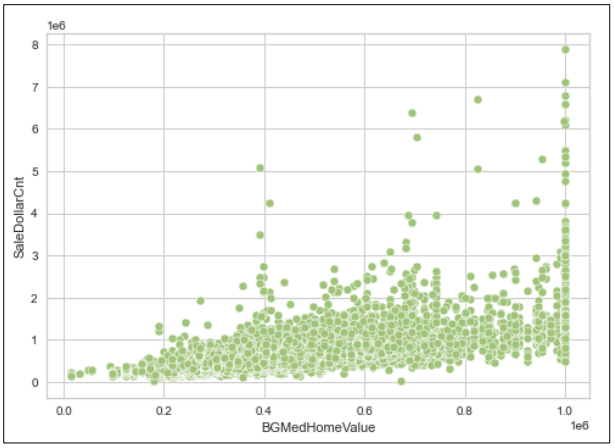


Figure 10: ScatterPlot MedHomeValue

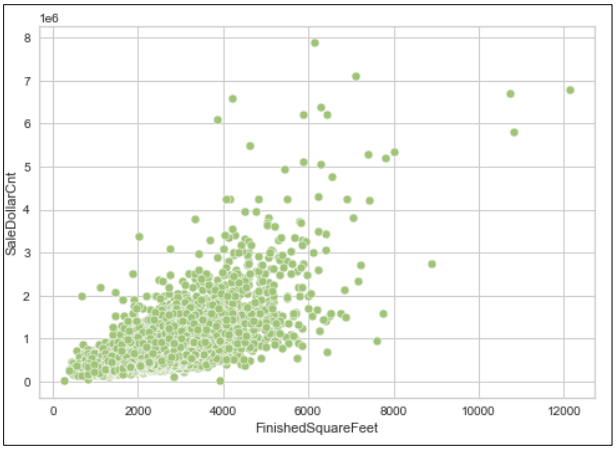


Figure 11: Scatterplot FinishedSquareFeet

In Figure 10, we can see that there are a couple of outliers in the top right corners. However, they seem to follow the trend, so we will keep them. It probably corresponds to the data row with 9 bathroom counts (figure 11)

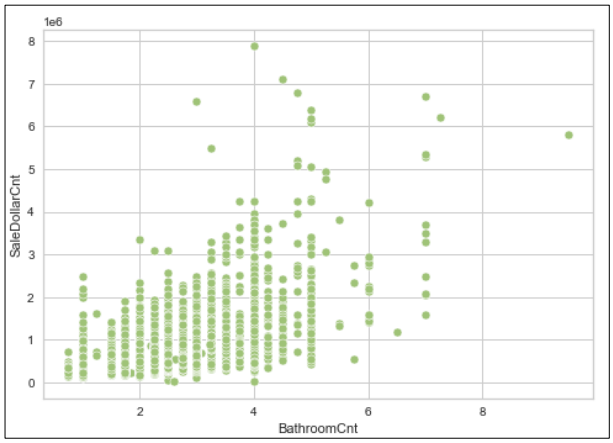


Figure 12: Scatterplot BathroomCnt

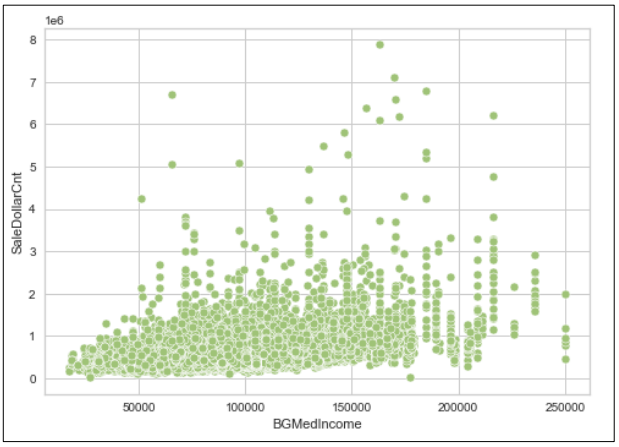


Figure 13: Scatterplot BGMedIncome

Probably not the best way to visualize bathroom count as it is a categorical variable but it seems to support the linear relationship as can be seen in the plot

Multicollinearity:

Next, I try to explore whether there is correlation between the explanatory variables i.e. Multicollinearity. This is one of another assumptions of linear regression where correlation between features needs to be removed. One way to find multicollinearity is through a heat map which shows correlation matrix between all the features

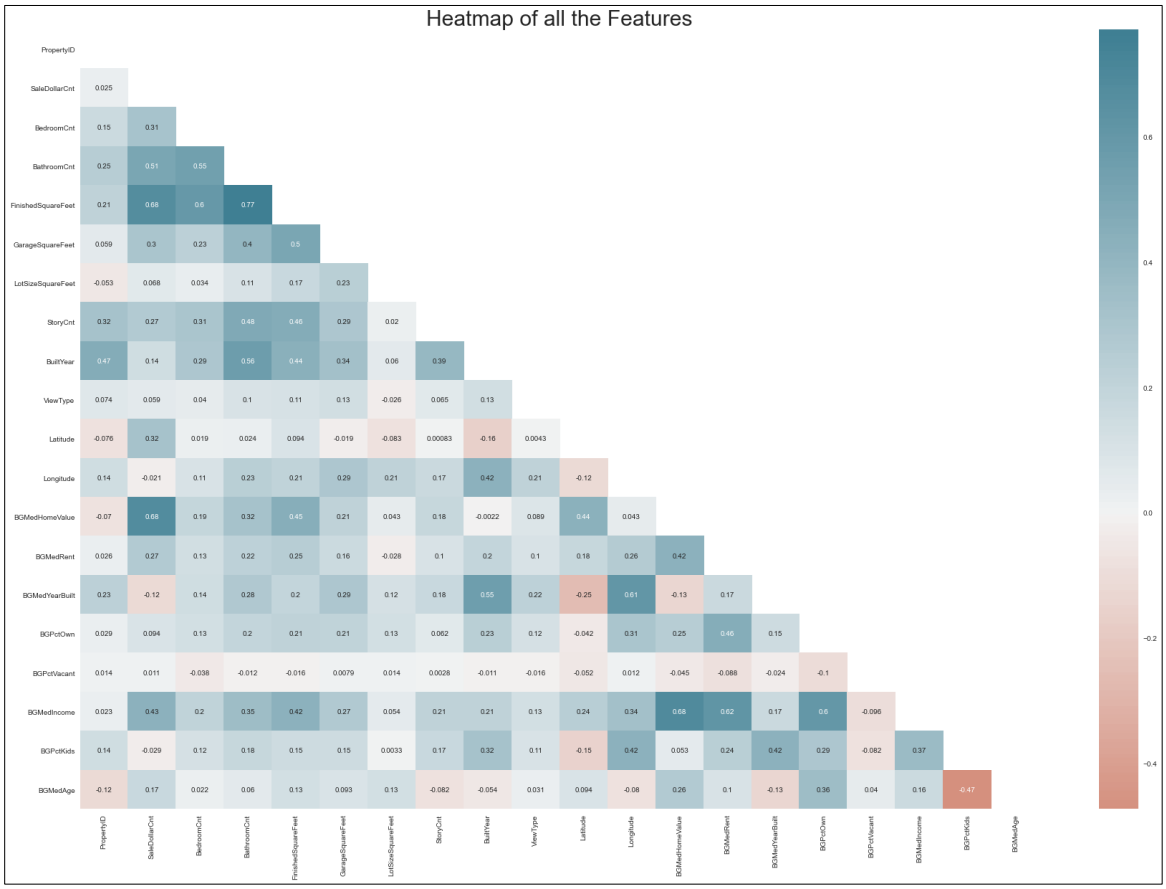


Figure 14: Heatmap representing Correlations between independent variables

Observations:

- There is some collinearity between some variables like **BathroomCnt** & **FinishedSquareFeet, BathroomCnt** & **BedroomCnt** which is understandable as more Finished Square feet implies more of these two.

  Since I will be exploring models other than multiple linear regression, I will keep these features for now. Though in my notebook, I would only pick the highest correlated features and drop the rest for Linear Regression. I picked these not only based on the correlation but on multiple experiments using cross-validation

4) Feature Engineering:

I added a few new features and changed they way some were represented to see if that helped. Some of these were used to experiment with model but not necessarily used.

- **Proportion of FinishedSquareFeet**

  The idea behind this feature was that a lot of homes had greater LotSquare Footage but little finished squarefootage. This was especially evident in SF categories (Single Family). Since the ZoneCodeCounty encoding (see below) didn't do well, I used this to capture some of the information. It did manage to be **high up in the correlation** and was the **4th most important** feature in XGBoost

- **BGMedYearBuilt & YearBuilt**

  Instead of using the actual year, I thought it would be more prudent to calculate the "Age" of the house etc, as that numeric value might have a greater affect on the target variable than year

- **Month**

  The transaction date is in datetime format so cannot be fed to the model as is. Both the training & the testing data cover transaction dates for the year 2015. Since the year is a constant, it doesn't help the model. Instead, we use it to create a column "**Month"** to see if the month can possibly explain the Sale Price of a house

- **ViewType**

  I transformed the Viewtype into a one-hot encoding vector. The idea behind this was that since ViewType is a categorical data type, higher ViewType values such as 78, 79 (I am assuming they correspond to a type) might skew the models and results in inaccurate results. Instead of label-encoding it, I decided to go with one-hot vector

  Finally, I converted ZoneCodeCounty using one-hot encoding. However, since there were a lt of types of codes, I did not use them. One of my future approaches include trying to use this data more effectively

- **High Season**

  Naturally, housing prices are higher in the summer compared to the "off-season". I decided to incorporate this in the data. However, the issue was that the training set had no purchases outside these months and vice-versa for testing dataset. Hence, I couldn't use this. However, it should be a good indicator otherwise.

5) Model Fitting

I tested the performance of the models using MAPE score & RSME values on the split of the train set only

1) Linear Regression

For multiple linear Regression, initially I simply picked the top features that were highly correlated. After multiple experiments using cross-validation, I dropped **Longitude, TransData, CensusBlockGroup, Usecode & ZoneCodeCounty.** I reached this conclusion after iterating through all the features

To prevent overfitting on the data, I followed this by using **Lasso Regression**. In addition to penalizing the complexity models, it also helps in feature selection to minimize loss. This helped improve the performance of the model based on my initial train/test split data. I used cross-validation to find the best alpha (1e^-15)

Error Analysis

My attached notebook has more details about the RMSE scores & MAPE scores I achieved with the split train/test data. But since the actual problem is to test the models on the testing data, I am obviously unable to share that information.

One way to visualize the performance of the model is using a regression plot. If the residuals are randomly dispersed around the horizontal axis, the model is well-suited for the data. This is because if we project those points on the y-axis, we can see a normally distributed histogram. I have written a blog explaining residual graphs here https://towardsdatascience.com/how-to-use-residual-plots-for-regression-model-validation-c3c70e8ab378

My linear model seems to be well-suited to the data, given Figure 15's results

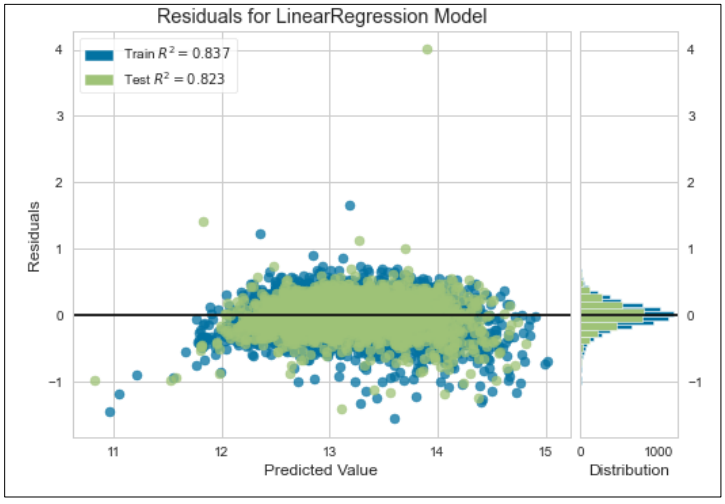


Figure 15: Residual Plot

2) XGBoost

The second model I selected was XGBoost Regressor. This is a tree-based model, different than our linear regression model.

For this model, I still dropped the features discussed above. However, I did include my ViewType One-hot encoding vectors. XGBoost seemed to perform better than the regression model.

HyperParameter Tuning

I used GridSearch to tune the hyperparameters for XGBoost. The parameter values are shown in my notebook and as follows:

n_estimators=500, **learning_rate**=0.03, **gamma**=0, **subsample**=0.75, **colsample_bytree**=0.7, loss='huber', **max_depth**=7, **min_child_weight** = 4, **objective** = 'reg:linear', **max_features**='sqrt'

I also used huber loss function which is less sensitive to outliers in data.

Feature Importance

One way to visualize which features are useful to predict the target variable is Feature Importance. It is calculated using node impurity in the tree. I plot the histograms to see which features are useful for predicting the Sale Price. Figure 16 (next page) shows the feature importance without the feature engineering I did earlier.

After adding the features, I introduced, you can see that **BuiltYear & Month,** especially the former, is further high up the importance list, which means it is a very useful feature for prediction. This result can be seen in Figure 17 (next page) (And notebook)

For my models, the best results I achieved, had 83% of the predicted sale price within 20% of the actual price. (Training set)

3) Stacking

I stacked the models to improve the average performance of the models on new data. However, I did not submit the results since I felt the Regression Model (Lasso) did not perform as well as XGBoost.

6) Final Model

I ended up picking the XGBoost model since it gave me the lowest MAPE scores. I achieved a best MAPE score of 11.43% with XGBoost i.e. the final predictions were within approximately 10% of the actual sale price. This can be recreated in the attached notebook.

7) Future Direction & Conclusion

I really enjoyed exploring this data and predicting the sale prices using different models & iteration of features. One of the challenging parts of the task was using Zone Code County. Also, one of the surprising results for me was the year the home was constructed did not have a significant linear relationship with the Sale Price, given usual circumstances. Since this data was only limited to 2015, I am curious to see how the results would be different if we had data from more years.
Lastly, I felt the testing data had a lot more of Single-Family Homes then the training set and as usual the model didn't replicate the performance of the training data

Based on my experience, the following are the future next steps I would like to explore;

1) Using external data such as economic data (mortgage rates etc), Zip codes, Crime information etc. to improve prediction. Latitude was one of the top features in the XGBoost model so I suspect zip code will likely help
2) Exploring the performance of Neural Network models on this data. That would help us incorporate other data like images of the house etc.
3) Visualize the actual location of the homes on the map to see whether clusters/ patterns exist (zip code divided)
4) Using **Stacking** approach to predict final results. I have an implementation for it already in the attached notebook. However, per my results, I wasn't very statisfied with the linear models so decided to only use XGBoost for the final csv. I definitely want to

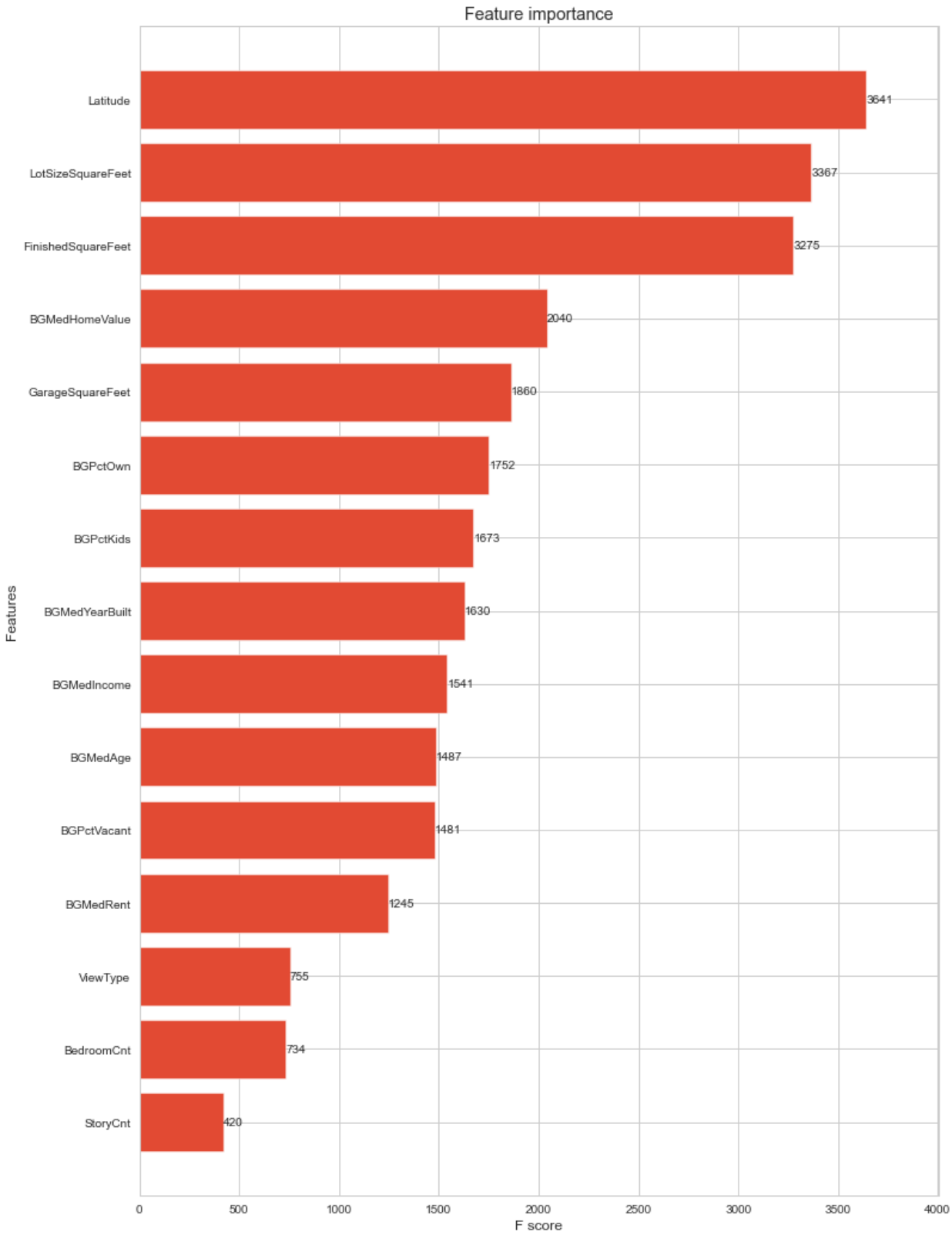


Figure 16: Feature Importance without Feature Engineering

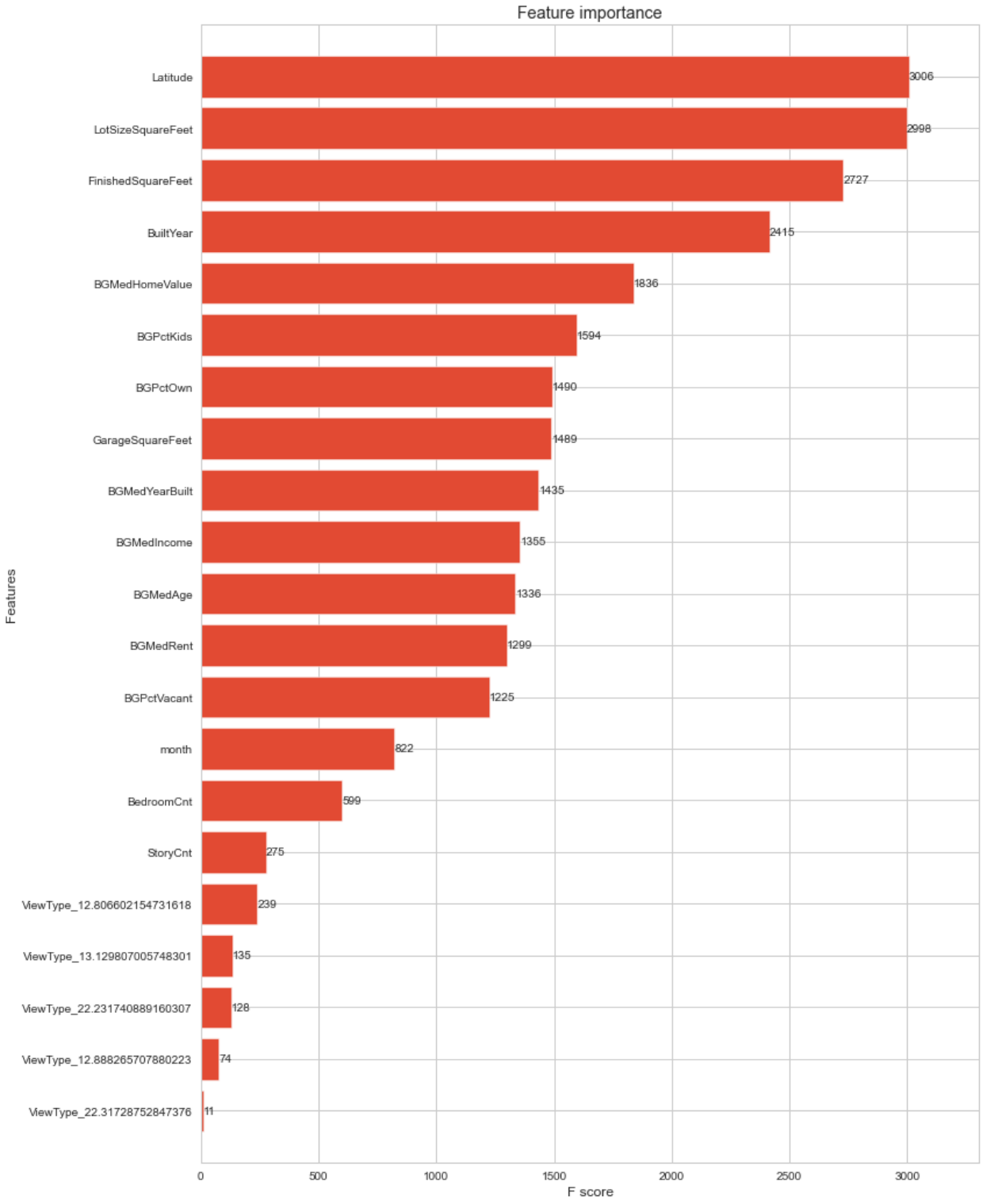


Figure 17: Feature Importance with Feature Engineering