Springboard--DSC Program
Capstone Project 2: Milestone Report 2
An Exploration of Housing Sales in Washington D.C. and King County

By: Garrett Yamane

March 21st, 2020

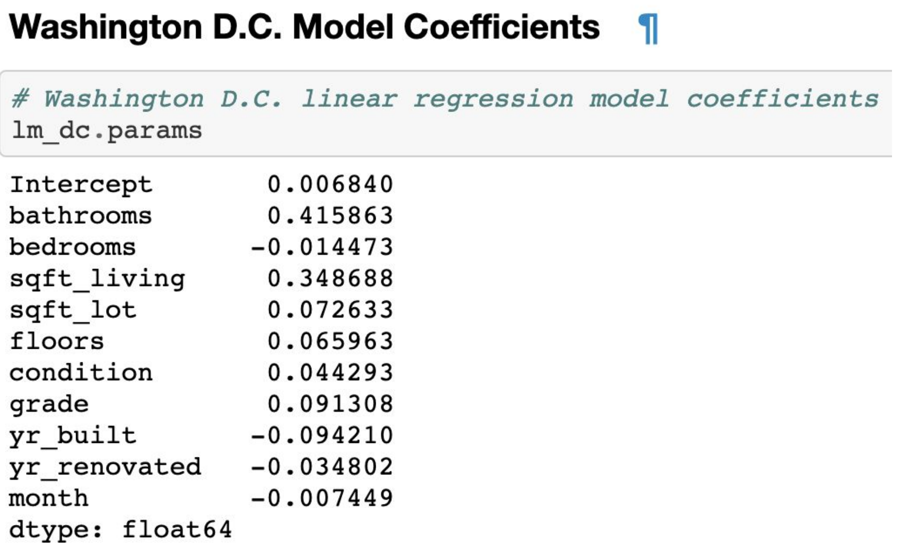
1. Baseline Model Analysis

First, I defined the data frames with a couple of modifications made. Rather than having sale dates by day, I am creating another column labeled 'month.' The 'condition' and 'grade' columns need to be represented with categorical codes to be used with the rest of the data, which all are continuous variables. Additionally, I scaled the data to normalize each feature to have a consistent range of values with the rest of the columns.

a) Linear Regression Models: No Modifications

Before selecting the best features to build my models with, I was curious to see how well a linear regression model will perform without any modifications made to the scaled data. By including all features in the initial baseline model, I could see how well a generic model could represent the data for both Washington D.C. and King County. I wanted to explore how each feature could potentially affect (both positively and negatively) the overall fit of the models. The following was the resulting coefficients for each linear regression model:

i) Model Coefficients

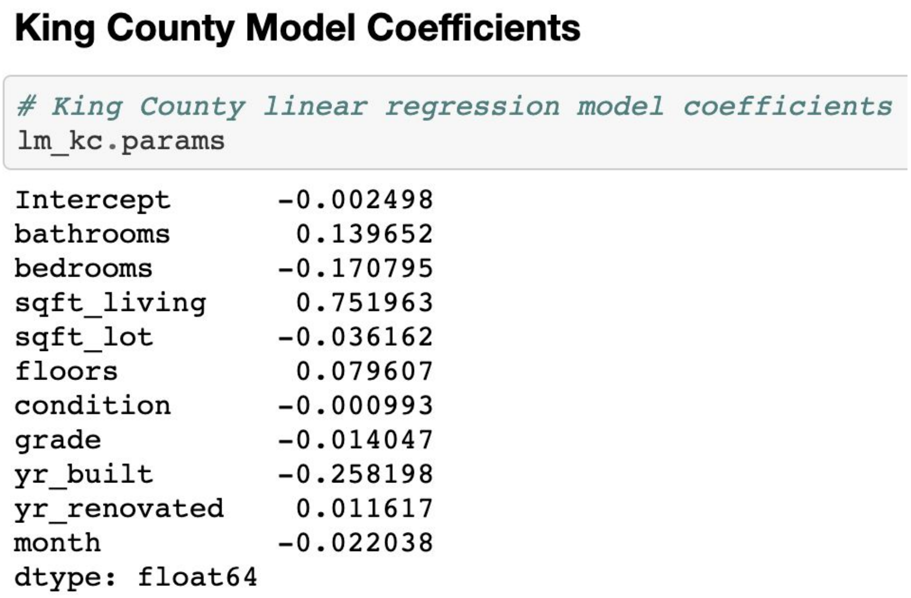


Washington D.C. Model Coefficients ¶

```
# Washington D.C. linear regression model coefficients
lm_dc.params
```

```
Intercept       0.006840
bathrooms       0.415863
bedrooms       -0.014473
sqft_living     0.348688
sqft_lot        0.072633
floors          0.065963
condition       0.044293
grade           0.091308
yr_built       -0.094210
yr_renovated   -0.034802
month          -0.007449
dtype: float64
```

Figure 23: Washington D.C. linear regression model coefficients with no features excluded



King County Model Coefficients

```
# King County linear regression model coefficients
lm_kc.params
```

```
Intercept      -0.002498
bathrooms       0.139652
bedrooms       -0.170795
sqft_living     0.751963
sqft_lot       -0.036162
floors          0.079607
condition      -0.000993
grade          -0.014047
yr_built       -0.258198
yr_renovated    0.011617
month          -0.022038
dtype: float64
```

Figure 24: King County linear regression model coefficients with no features excluded

Looking at the initial coefficient values, there are some distinctions I would like to point out. Living square feet is by far the largest positive coefficient for both models, which makes sense due to the fact that more square footage tends to positively increase the price of a house. However, the magnitude of the coefficient for the King County model is significantly higher

(0.75 compared to 0.34 in the Washington D.C. model). The coefficient values can range from -1 to 1, so 0.75 is very high. Another interesting difference is the "yr_built" coefficient. Although I expected newer buildings to increase the price, in both models it appears the more recent a house was built, the lower the price would be. Lastly, another interesting distinction is the "grade" coefficient. In Washington D.C., higher grades would positively impact the price of a house. However, in King County, this was not the case. One potential explanation for this could be due to King County's wider range of houses being sold (rural vs. urban). King County contains cities where a lot of the land is farmland where houses may have a lower grade compared to their city counterparts, and this may impact how the price is affected.

ii) Goodness of fit on Training Data

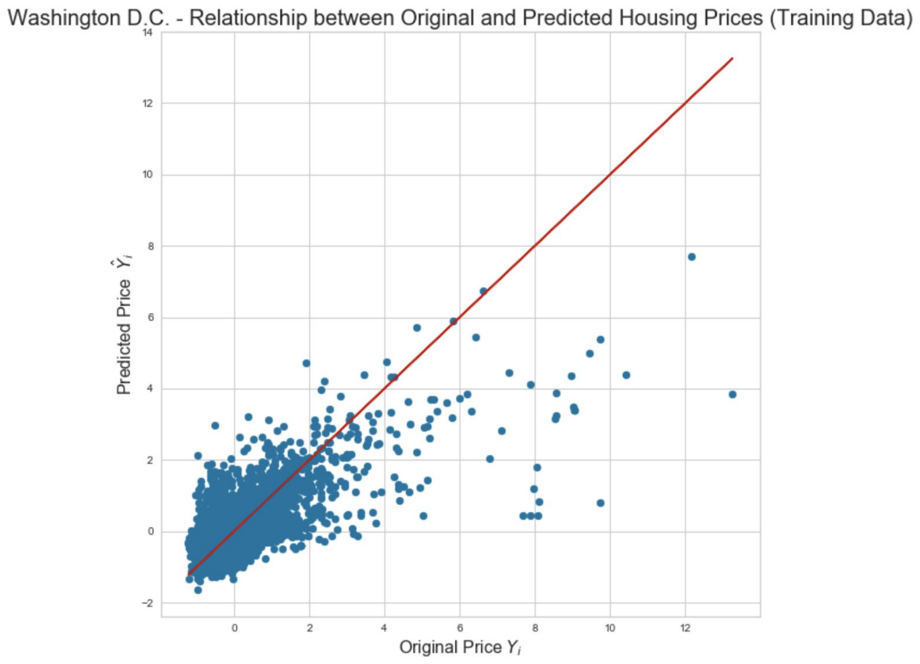


Figure 25: Washington D.C. Original vs. Predicted Housing Price on Training Data

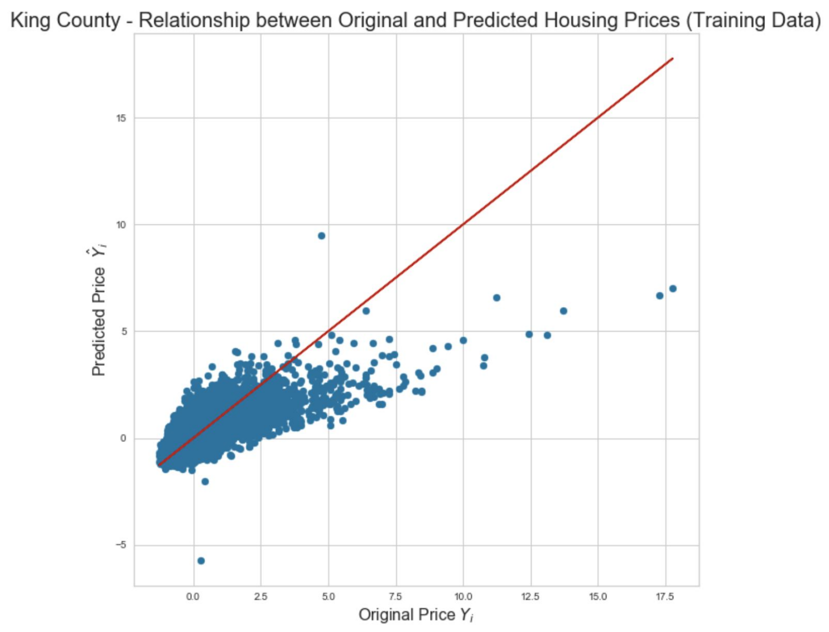
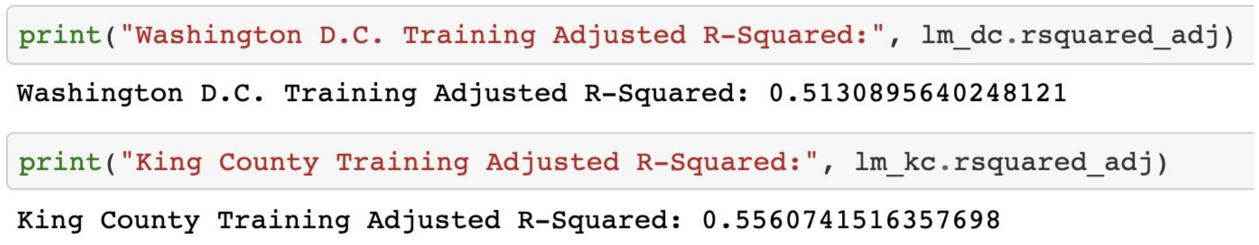


Figure 26: King County Original vs. Predicted Housing Price on Training Data

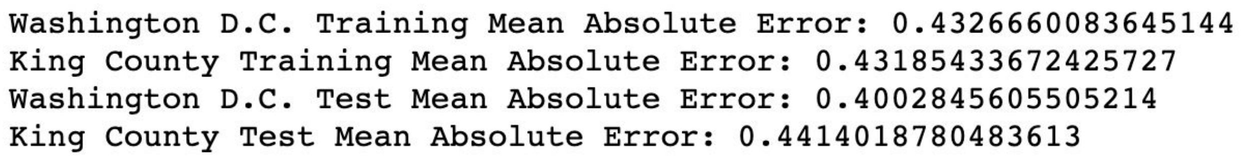
The first visualization I created for the linear regression model is illustrated in Figure 25 and 26 above. The line $y = x$ drawn through the graph represents perfect predictions, so points that are closer to the line represent better predictions made by the model. As we can see, there are a few points that are plotted for higher housing sales. The model appears to have had more difficulty making accurate predictions for these points.



```
print("Washington D.C. Training Adjusted R-Squared:", lm_dc.rsquared_adj)
Washington D.C. Training Adjusted R-Squared: 0.5130895640248121
print("King County Training Adjusted R-Squared:", lm_kc.rsquared_adj)
King County Training Adjusted R-Squared: 0.5560741516357698
```

Figure 27: Adjusted R-Squared values for Washington D.C. and King County Models

The coefficient of determination, R^2, signifies the percentage of the variance in the housing sale prices that can be explained by each linear regression model. Higher R^2 values means that a higher percentage of the variance is explained by the model. By comparing the R^2 values of each model on the training and test data, I can see how the explained variance changes for predictions by the model. The reason I am using the adjusted R^2 value is because this value can increase *and* decrease based on additional features being used in the linear regression model. If necessary features are added to the model, the adjusted R^2 will decrease, thus implying that the additional features are unnecessary. The R^2 value alone can only increase, which means that adding more and more features will inevitably increase the R^2 value. The King County model had a slightly higher adjusted R^2 value, meaning that it accounted for a higher variance percentage than the Washington D.C. model.



```
Washington D.C. Training Mean Absolute Error: 0.4326660083645144
King County Training Mean Absolute Error: 0.43185433672425727
Washington D.C. Test Mean Absolute Error: 0.4002845605505214
King County Test Mean Absolute Error: 0.4414018780483613
```

Figure 28: Mean Absolute Error for Washington D.C. and King County Models

Another metric that I am using to evaluate my model is the mean absolute error. The mean absolute error measures the average magnitude of the errors in a set of predictions, without considering their direction. It helps me determine the average difference between my predicted sale prices and the observed sale prices. I first looked at the MAE for my training data, and then compared this value to my test data to determine if there is a significant flaw in my model. On just initial observation, it appears that there isn't a large difference between the two. However, one interesting note was that the MAE went down from the Washington D.C. training data to the test data. This means that the D.C. model was more accurate for data that it had not seen.

iii) Goodness of fit on Test Data

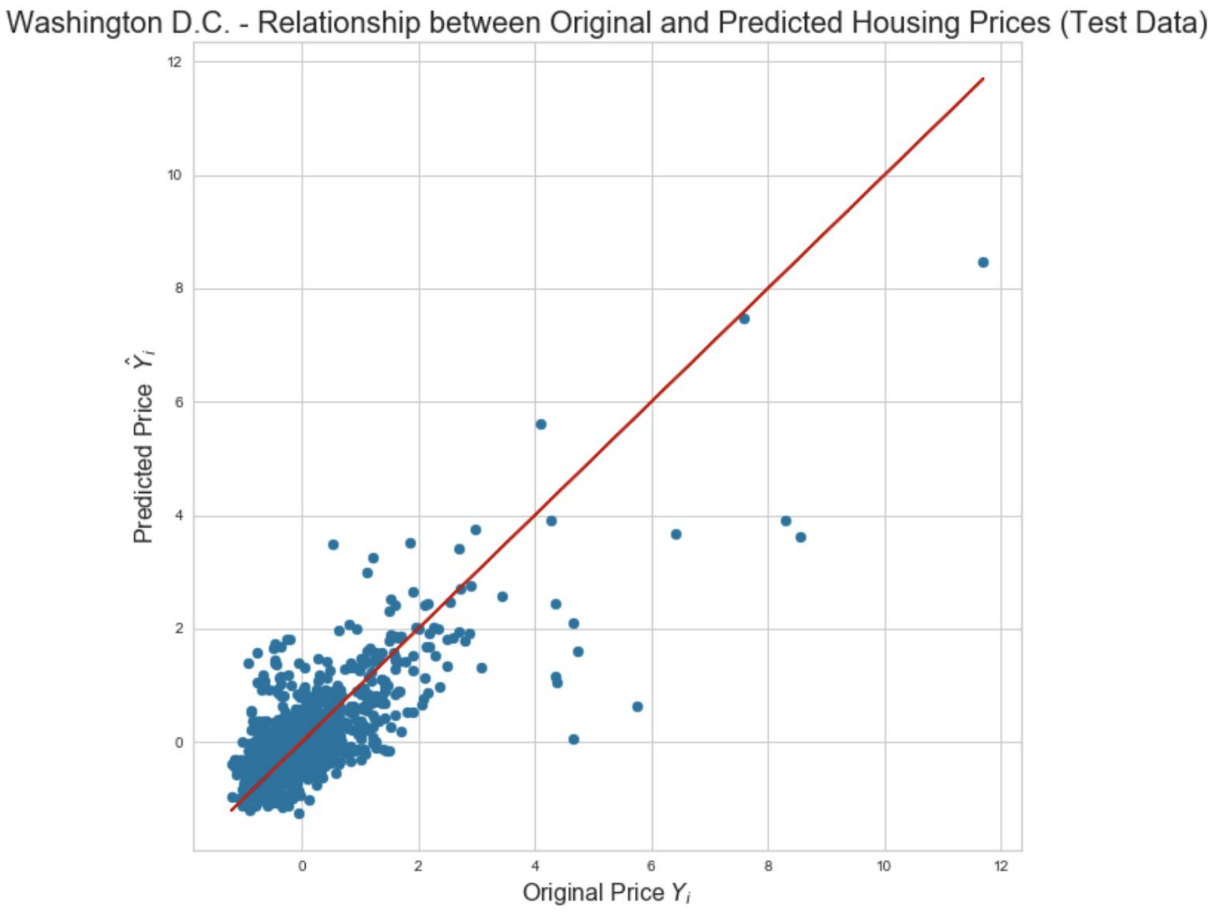


Figure 29: Washington D.C. Original vs. Predicted Housing Price on Test Data

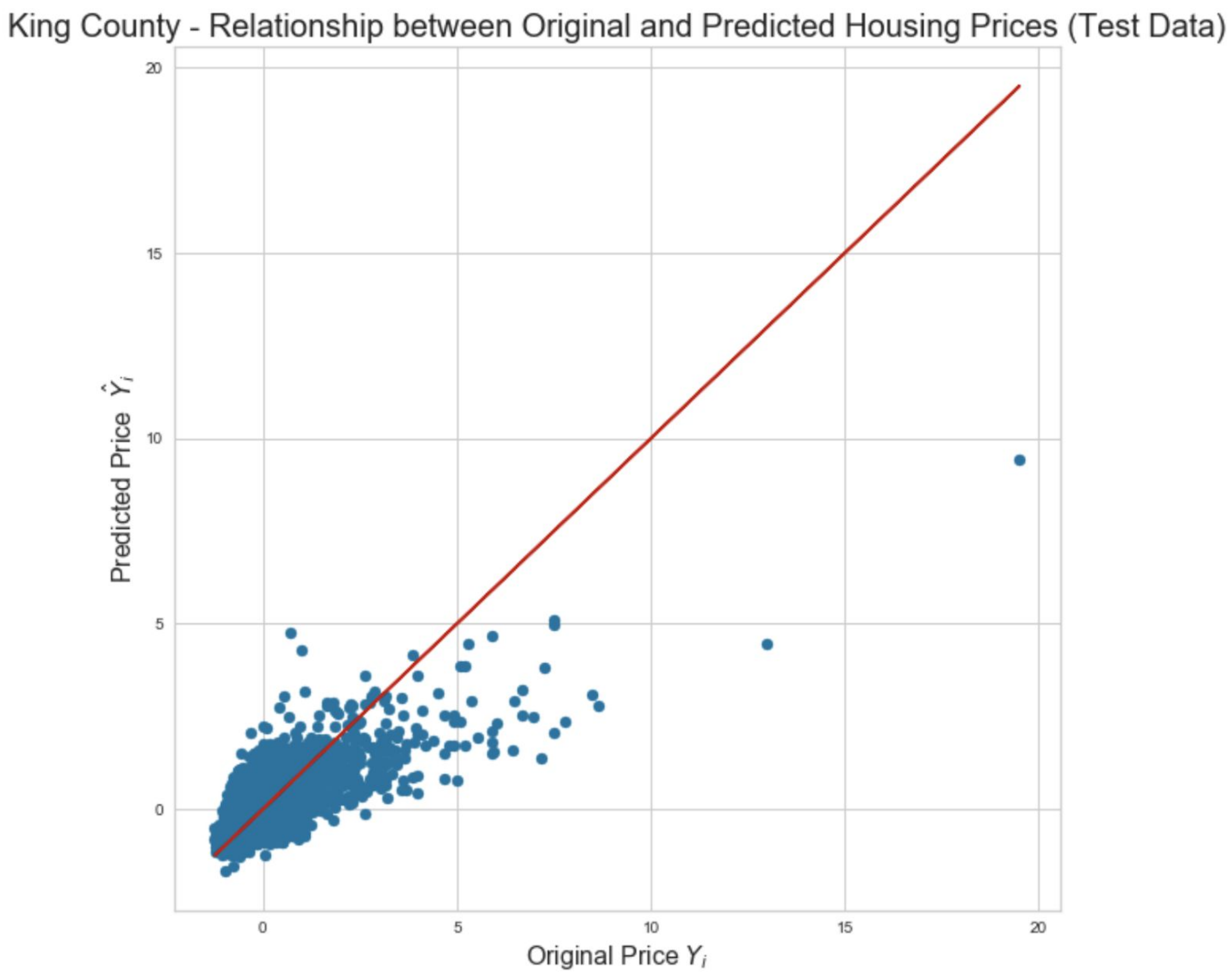


Figure 30: King County Original vs. Predicted Housing Price on Test Data

There are a few assumptions that Linear Regression models make:

- The standard deviation of y = 'price' should be constant for different values of X
- Normal distribution of errors (test for skew of model)
- Independence between errors (observations are obtained independently)

iv) Test for Constant Standard Deviation: Fitted vs. Residual Plot

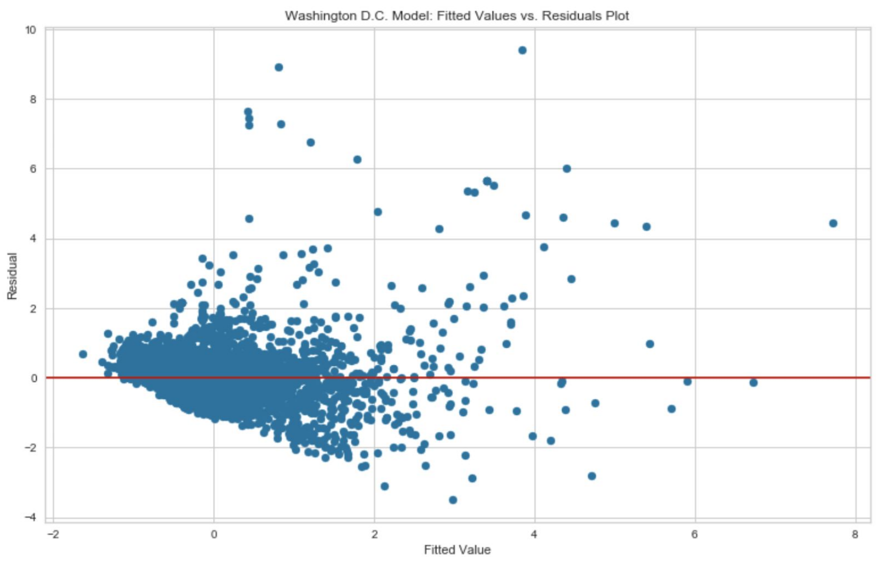


Figure 31: Washington D.C. Fitted Values vs. Residual Plot

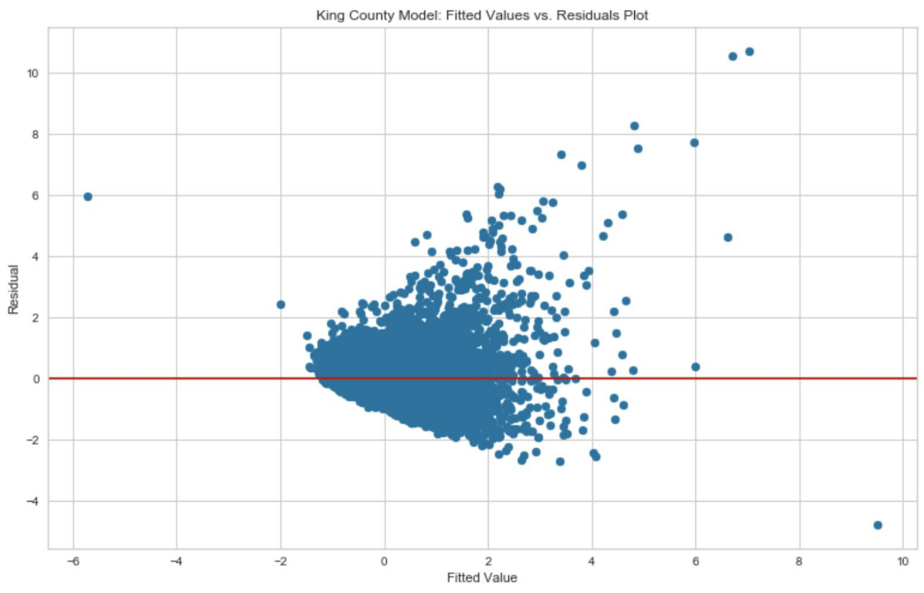


Figure 32: King County Fitted Values vs. Residual Plot

Because there appears to be a 'fanning' effect in the plot, this implies that the standard deviation is not constant for different values of *X*. For both models, it seems that my models will need some tuning for the features that I choose.

v) Test for Normal Distribution of Errors: Quantile Plots

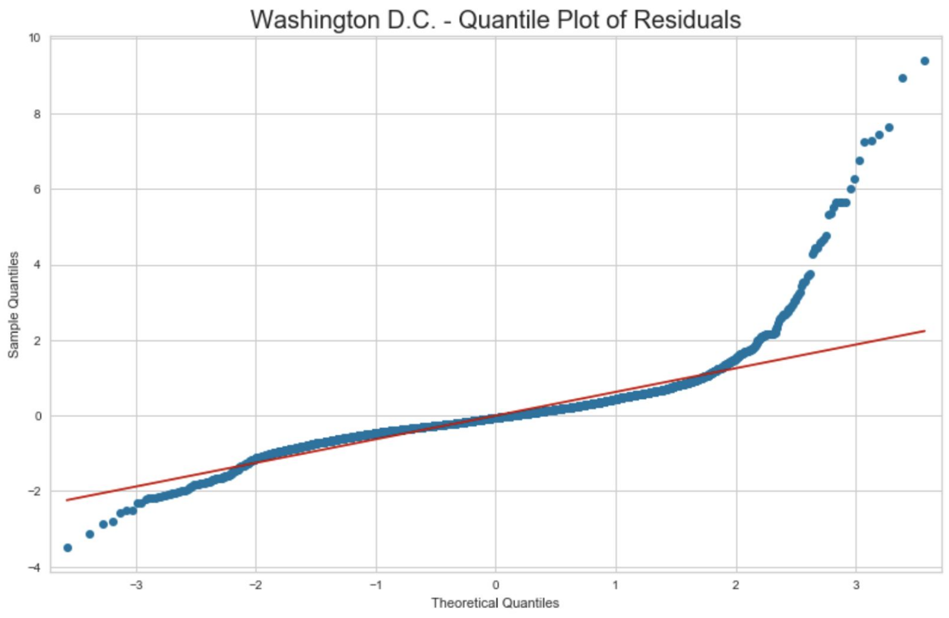


Figure 32: Washington D.C. Quantile Plot of Residuals

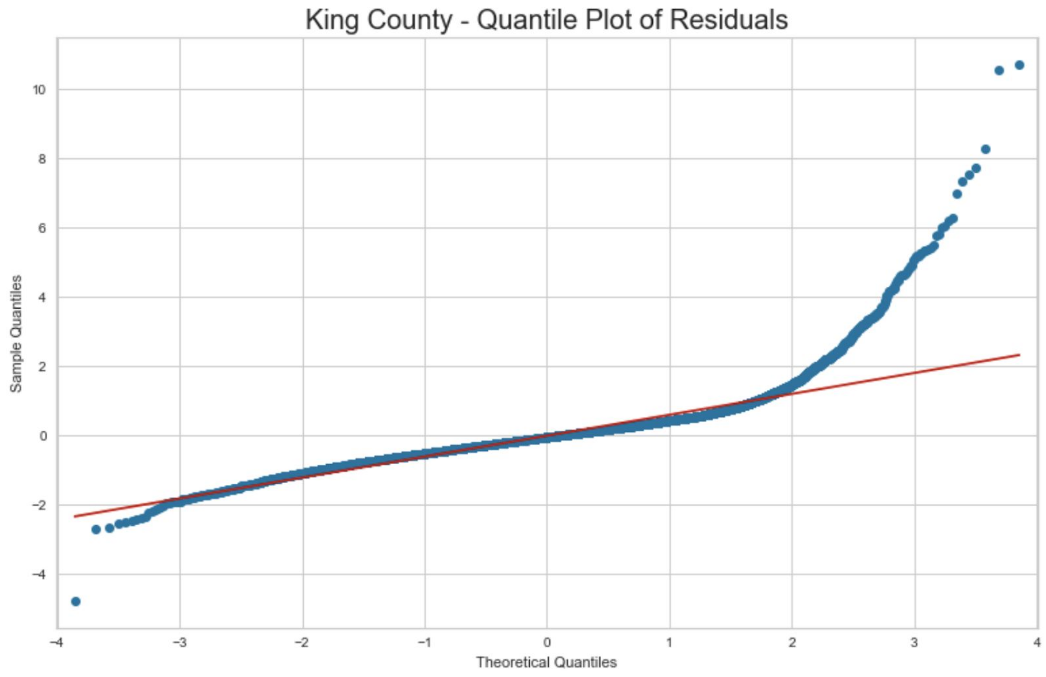
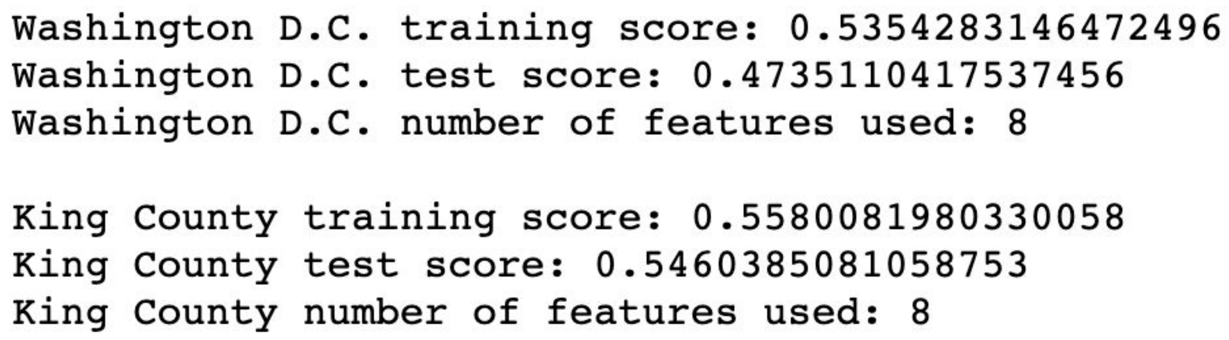


Figure 33: King County Quantile Plot of Residuals

The red line on the plot above signifies a normal distribution of errors. If the errors of each model were normally distributed, they should follow the red line. However, it appears that for both Washington D.C.'s model and King County's model there are clusters of values that deviate from the line, meaning that my model can improve. The points that deviate from the line could be considered outliers and may affect the overall accuracy of each model.

b) Feature Selection: Lasso Regression

One benefit of using Lasso Regression is to help induce sparsity into the model. Although my previous linear regression models from above do not drastically overfit the data (the adjusted R-squared values and Mean Absolute Values are not extreme between the test and data set to imply overfitting of the model), I would still like to introduce some form of feature selection to the models' predictor variables. Lasso Regression zeros out variables that have a high penalty for the model, thus eliminating poor predictor variables.

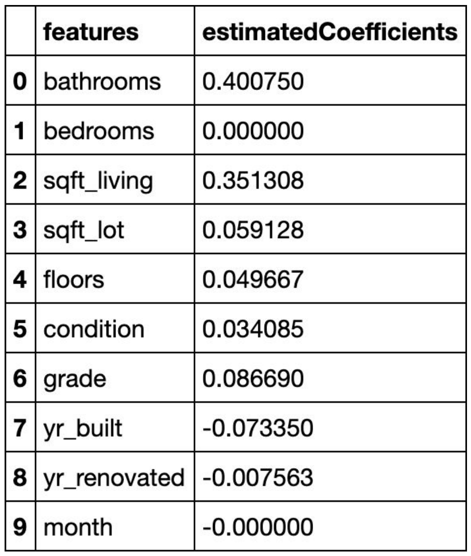


```
Washington D.C. training score: 0.5354283146472496
Washington D.C. test score: 0.4735110417537456
Washington D.C. number of features used: 8

King County training score: 0.5580081980330058
King County test score: 0.5460385081058753
King County number of features used: 8
```

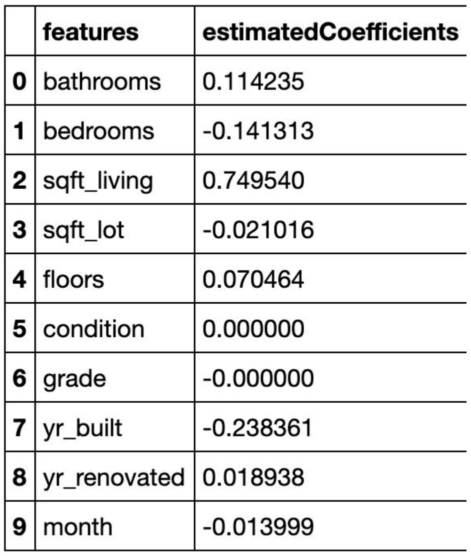
Figure 34: Lasso Regression model scores and coefficients used

These scores are based on the *R-squared* value. Compared to my previous models built with no modifications, these scores are similar, with the Washington D.C. lasso regression model score being slightly lower.



	features	estimatedCoefficients
0	bathrooms	0.400750
1	bedrooms	0.000000
2	sqft_living	0.351308
3	sqft_lot	0.059128
4	floors	0.049667
5	condition	0.034085
6	grade	0.086690
7	yr_built	-0.073350
8	yr_renovated	-0.007563
9	month	-0.000000

Figure 35: Washington D.C. Lasso Regression Model Coefficients

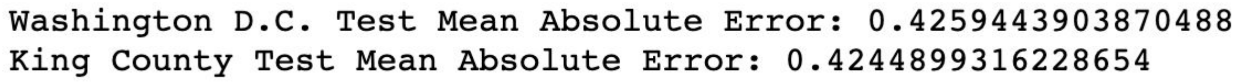


	features	estimatedCoefficients
0	bathrooms	0.114235
1	bedrooms	-0.141313
2	sqft_living	0.749540
3	sqft_lot	-0.021016
4	floors	0.070464
5	condition	0.000000
6	grade	-0.000000
7	yr_built	-0.238361
8	yr_renovated	0.018938
9	month	-0.013999

Figure 36: King County Lasso Regression Model Coefficients

Running the Lasso Regression on the training data helped eliminate 2 features for each model that had a high prediction penalty. In the Washington D.C. lasso regression model, *bedrooms* and *month* were zeroed out. However, in the King County model *grade* and *condition* were zeroed out. This reveals the idea that certain housing features have more of a significant impact depending on the location that a house is being sold.

i) Evaluation of the Model: Mean Absolute Error



```
Washington D.C. Test Mean Absolute Error: 0.4259443903870488
King County Test Mean Absolute Error: 0.4244899316228654
```

Figure 37: Mean absolute error for Lasso Regression models

ii) Goodness of fit on Test Data

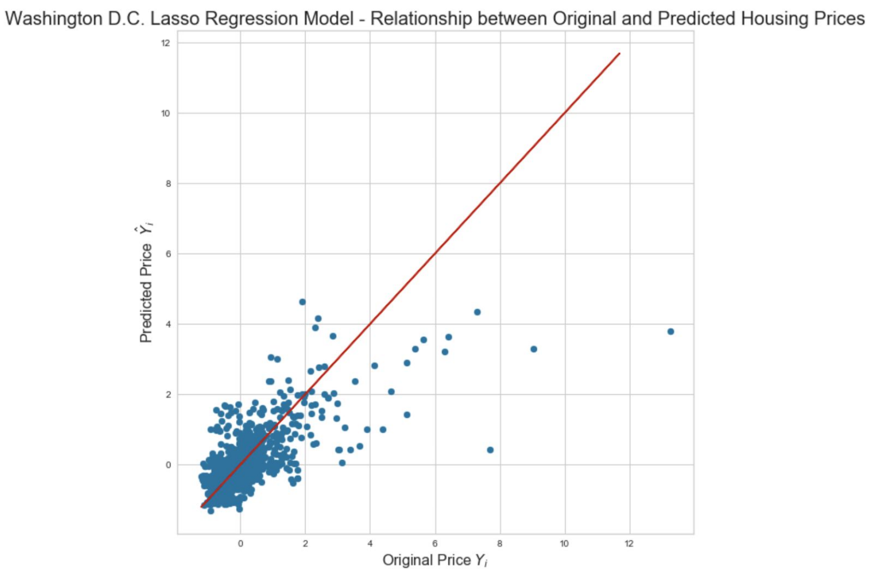


Figure 38: Washington D.C. Lasso Regression - Original vs. Predicted Model Housing Price

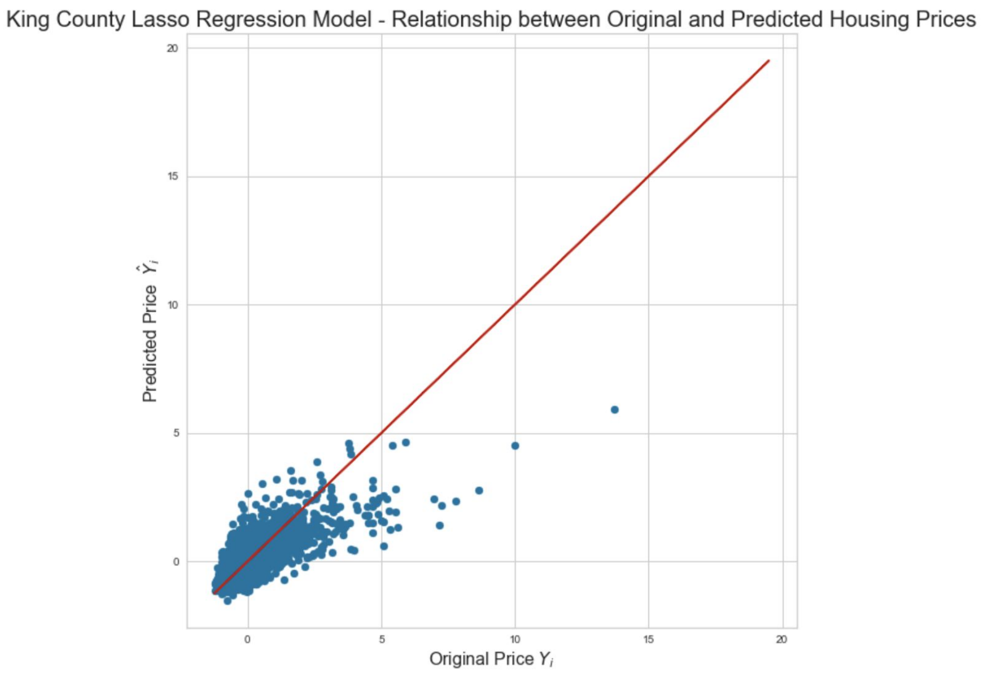


Figure 39: King County Lasso Regression - Original vs. Predicted Model Housing Price

iii) Test for Constant Standard Deviation: Predictions vs. Residual Plot

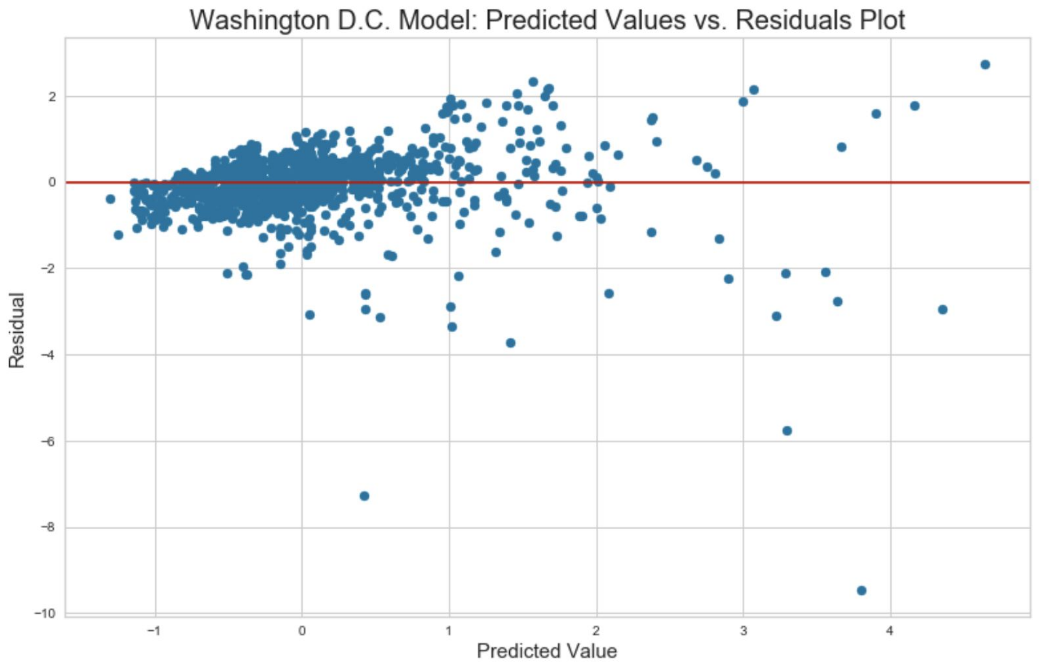


Figure 40: Washington D.C. Lasso Regression Model - Fitted Values vs. Residual Plot

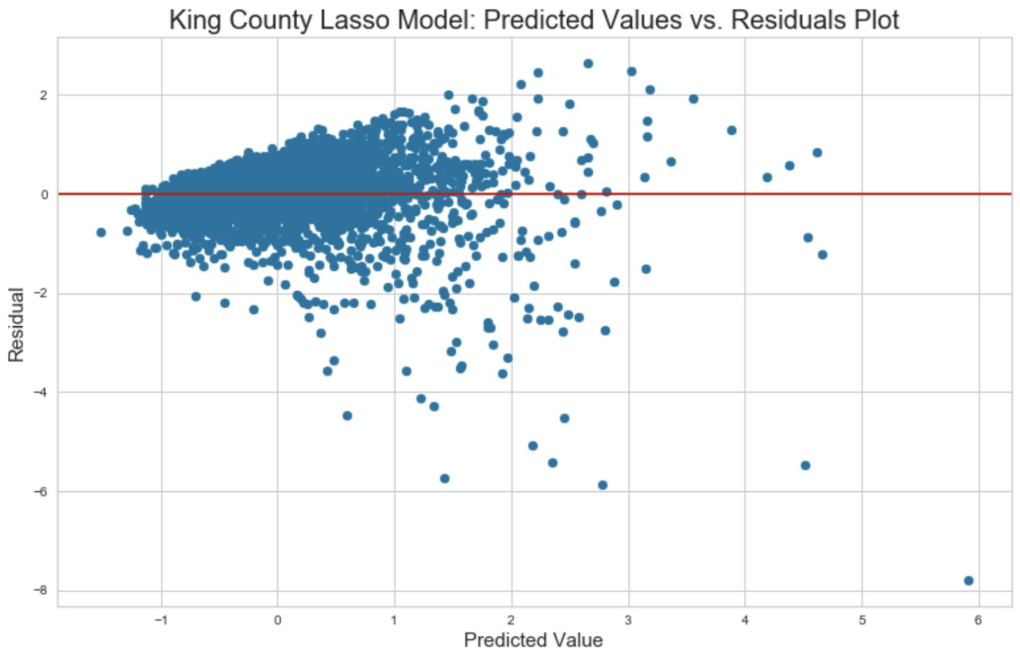


Figure 41: King County Lasso Regression Model - Fitted Values vs. Residual Plot

In Figure 40 and Figure 41, this is where it becomes much more clear that the Lasso Regression Model has been more powerful for predicting housing prices. There is not as much of a 'fanning' effect for the predicted values' residuals, meaning that they are closer to the true price values of the original data. They are more random about the Residual = 0 line, which indicates that the standard deviation is constant.

iv) Test for Normal Distribution of Errors: Quantile Plots

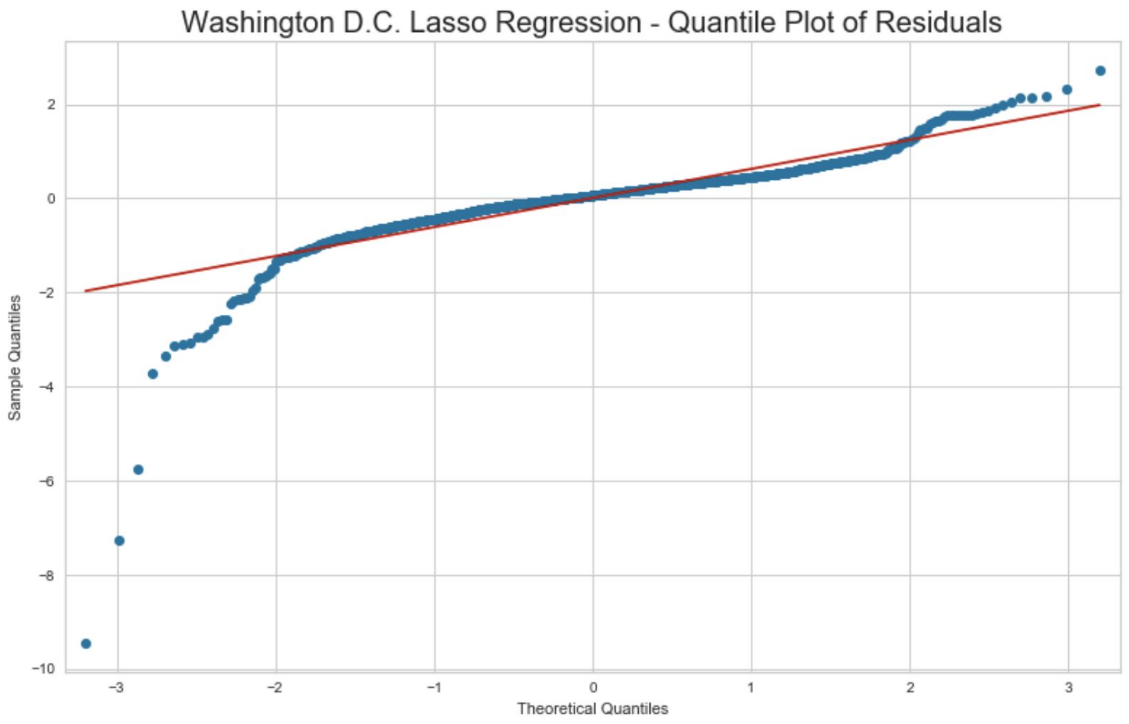


Figure 42: Washington D.C. Lasso Regression Model - Quantile Plot of Residuals

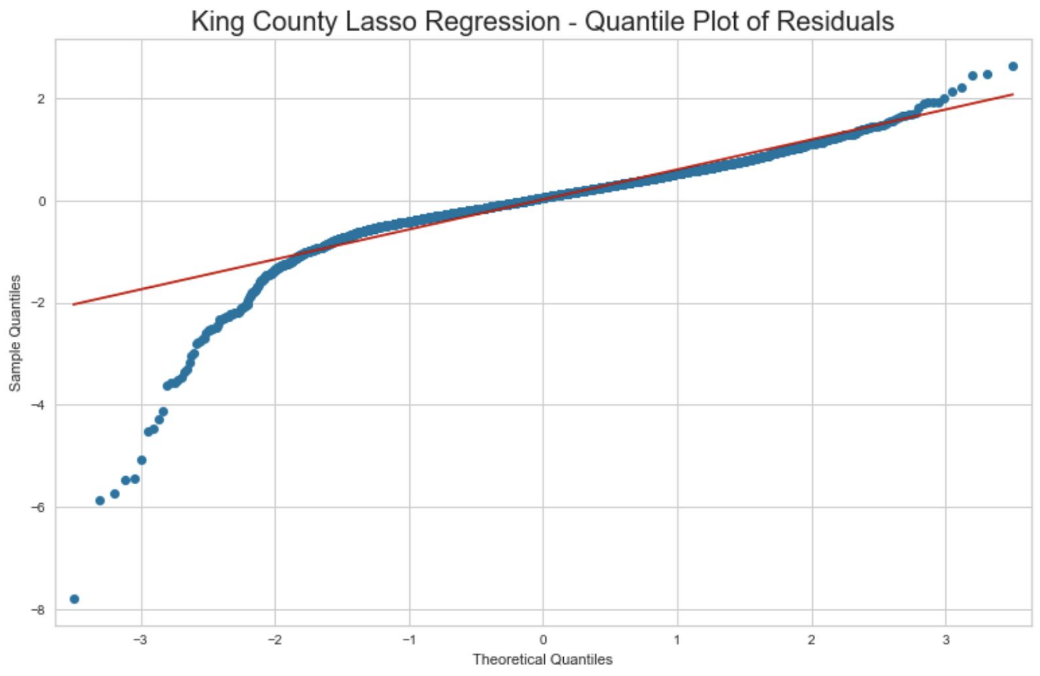


Figure 43: King County Lasso Regression Model - Quantile Plot of Residuals

As the theoretical quantile values get larger, the sample quantiles tend to the red line much closer than my original models. However, the Lasso Regression models for both the Washington D.C. and King County data struggle for lower theoretical quantiles. This implies that there is not a normal distribution of errors for each model, which highlights a weakness for the predictive power of them.

<u>2. Baseline Model Extension: Random Forest</u>

In my baseline model, I began with a linear regression model where I did not modify any of the features. I built the model solely with the original features to see how a model would score when considering all of the housing features. I then moved on to apply a lasso regression model to the data set, which in turn eliminated certain features from each data set that gave the models a high penalty. As an extension to the baseline models I defined, I applied an ensemble learning method: Random Forest.

Random forest regression is an ensemble of randomized decision trees. Decision trees fall victim to overfitting as a result of being trained and built on specific data. Different data can result with vastly different decision trees, which is an issue when we are trying to build a model that can generalize to data beyond the training subset. Random forest takes these decision trees and combines them together to form one cohesive model.

Random forest is also a bagging technique. Bagging is a combination of bootstrapping and aggregation. By sampling from the population with replacement (bootstrap) and aggregating these models built on these samples together, we can avoid overfitting a model to a single subset of the data. These individual models are run independently and in parallel, and do not interact with each other. This is good to keep training completely uninfluenced by each other.

Each model I built was on a random tree containing a bootstrapped sample of my housing data. The predictions for the housing price of each tree will be used to aggregate the models into a single general model.

a) Build Initial Random Forest Model and Hyperparameter Tuning

Since random forest regressors do not fall victim to overfitting, it didn't make sense to look at how well the model does based on the training data. This is where the OOB (out-of-bag) error comes in. The OOB error is what is used to measure the prediction errors of random forest models, since this error is based on how well the model predicts from the bootstrapped samples.

I first built a base set of random forest models with various settings for the *max_features* parameter. This *max_features* parameter represents the number of features to consider when deciding which feature at a split is the best one to use. For example, when given a node in my decision tree, the *max_features* parameter tells me how many random different features to consider in order to decide how to proceed down the tree. This process of selecting a certain number or random features helps prevent overfitting.

Once the initial base set of models were built, I built a visualization to illustrate the OOB errors of the models on the training data. This helped with determining how the models could be improved and led into the next step of hyperparameter tuning. By examining how manipulating the *max_features* and *n_estimators* (*n_estimators* is the number of trees that are made in the forest), I examined the model that returned me the best OOB error to use to compare to my base model:

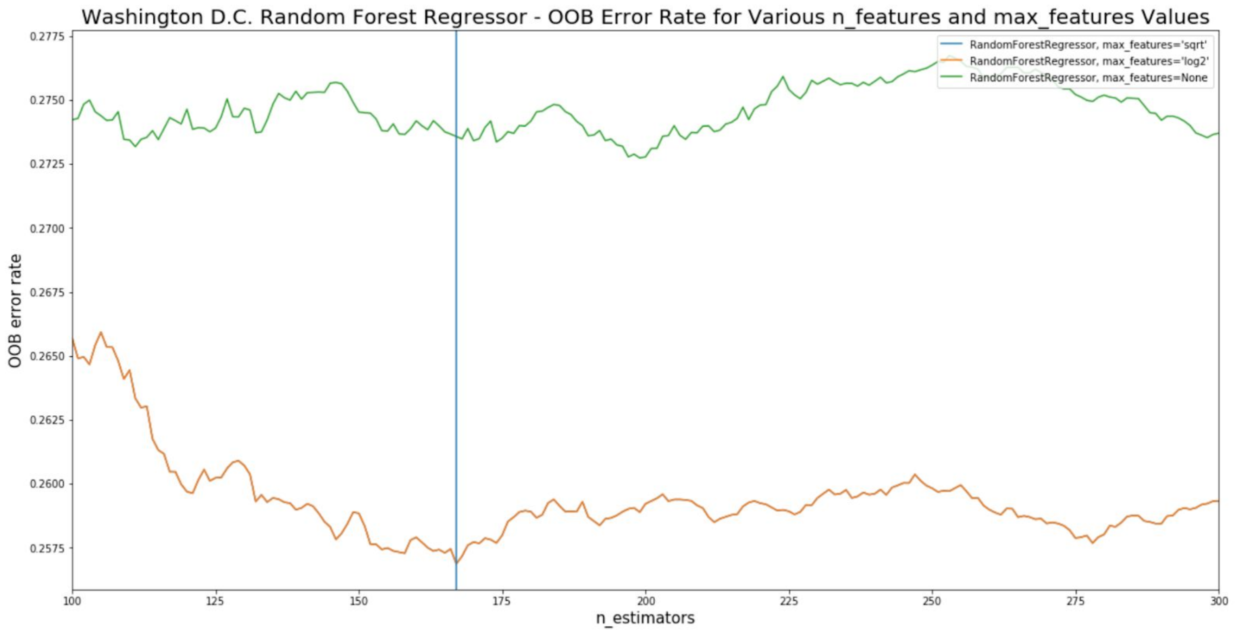


Figure 44: Washington D.C. Random Forest Regressors with OOB error rate

In the above graph, it appeared that when *max_feautures* = 'sqrt', the model's OOB error rate didn't even appear. This is because after looking deeper into the error rate, the error rate was identical to when the *max_features* = None. This highlights the fact that there appears to be a certain threshold for the *max_features* before the error rate does not change. However, when I set the *max_features* = 'log2', the error rate decreased significantly. The vertical line that I drew in the above plot is the *n_features* value where the error rate for the model was the lowest. This model (*max_features* = 'log2', *n_estimators* = 167) appeared to be the best model for the Washington D.C. training data set, and I used this same process to find the best two parameters to build out the Random Forest regressor for the King County data set:

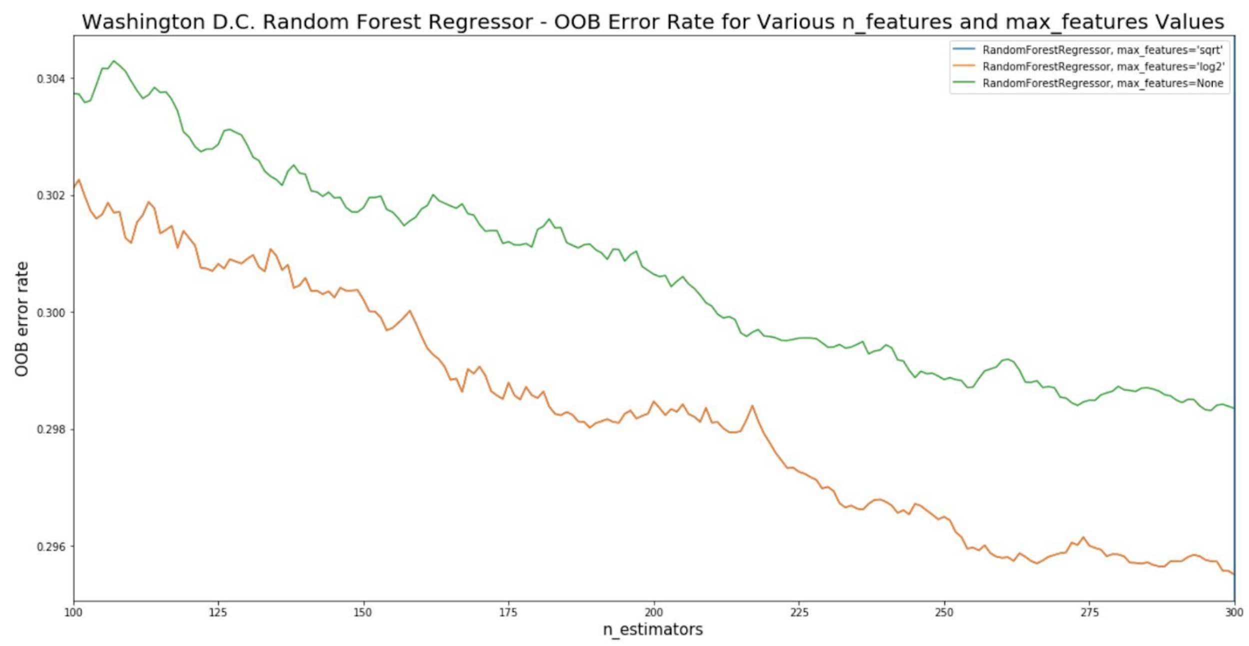
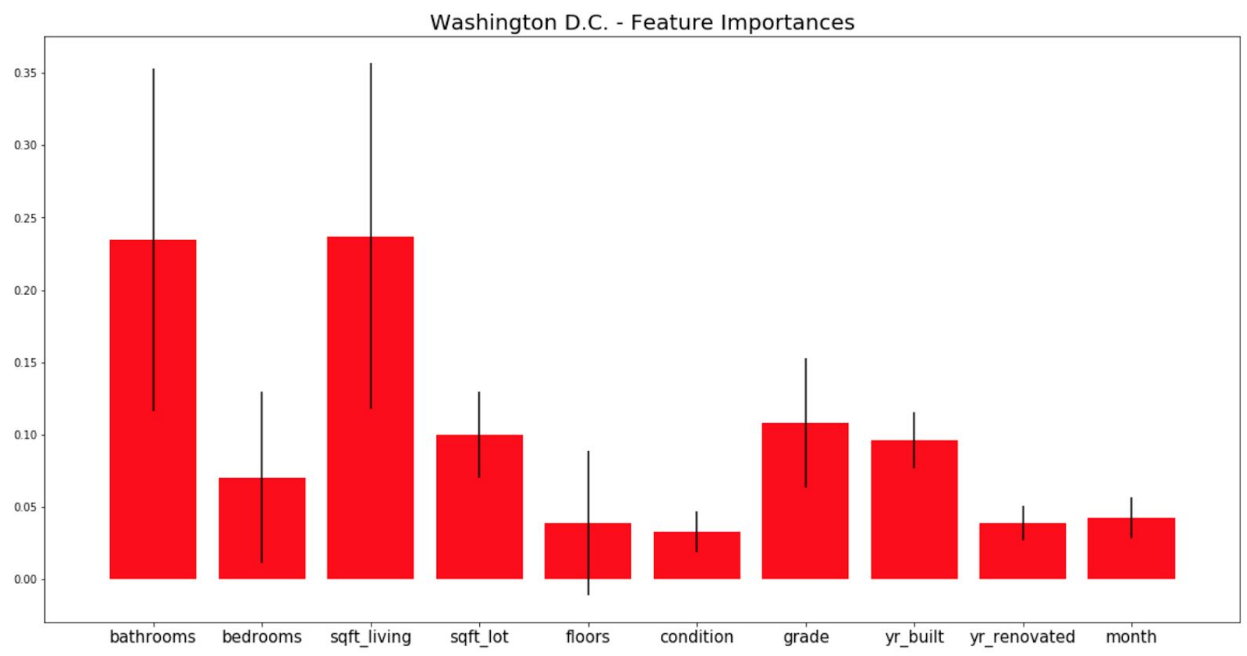


Figure 45: King County Random Forest Regressors with OOB error rate

Similar to the Washington D.C. random forest regressor, the King County random forest regressor for *max_features* = 'sqrt' doesn't show on the graph. However, the trend for the OOB errors for *max_features* = 'log2' appeared to decrease as *n_estimators* increases. The best n_estimator for the King County random forest model is 300 when I tested the estimator values ranging from 100 to 300, so I used this parameter as the best model for the King County data. Now that I have the parameters for the random forest models, I will compare them to their baseline counterparts and see how well they perform on the test data.

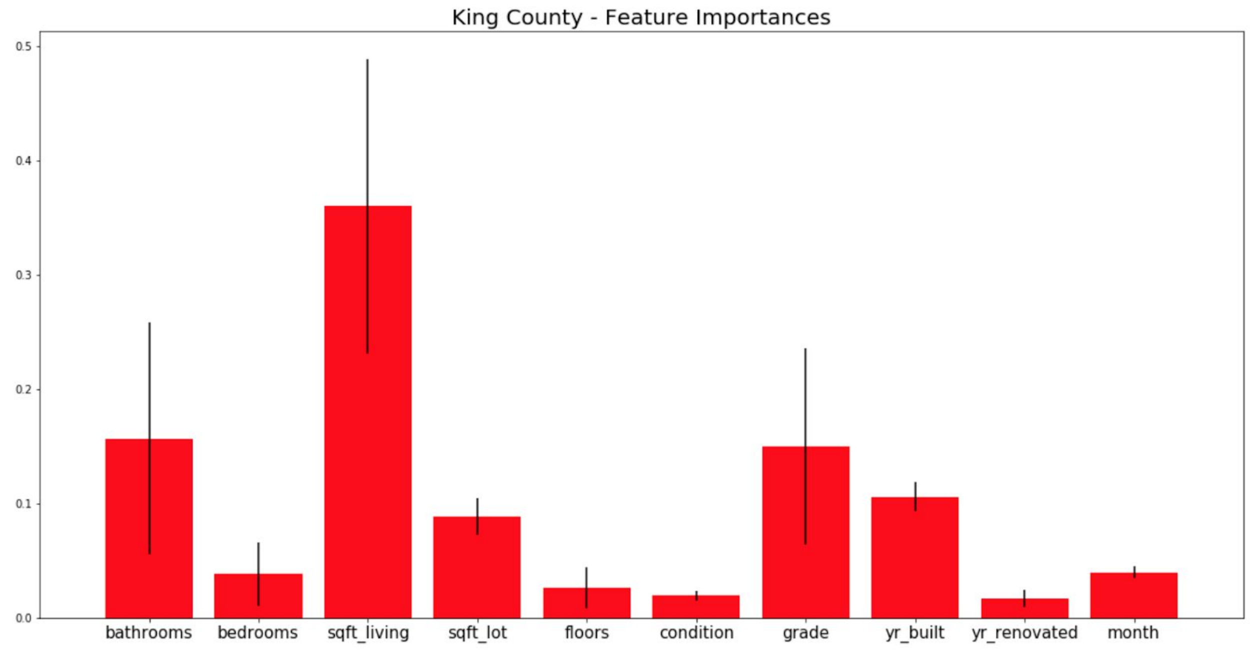
b) Feature Importance

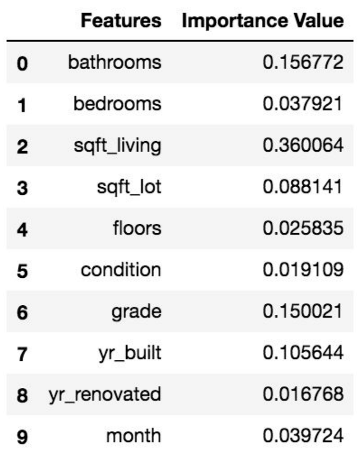
Random Forest Regressors (RFR) belong to the family of non-parametric models, and do not possess a known closed form. What this means is that I couldn't directly look at the model coefficients like I did for the Linear Regression baseline models that took the base form *y = a + b1x1 + b2x2 +* .... Instead, what I did was look at the feature importance of each housing feature for the random forest models. Scikit-learn's *feature_importances_* model attribute allowed me to examine the importance values of each feature. Higher values equates to higher importance. Below is a bar chart of the importances for each random forest model:



	Features	Importance Value
0	bathrooms	0.234591
1	bedrooms	0.070417
2	sqft_living	0.237226
3	sqft_lot	0.099992
4	floors	0.039018
5	condition	0.032830
6	grade	0.108064
7	yr_built	0.095940
8	yr_renovated	0.039032
9	month	0.042889

Figure 46: Washington D.C. Random Forest Regressor Feature Importance Values





	Features	Importance Value
0	bathrooms	0.156772
1	bedrooms	0.037921
2	sqft_living	0.360064
3	sqft_lot	0.088141
4	floors	0.025835
5	condition	0.019109
6	grade	0.150021
7	yr_built	0.105644
8	yr_renovated	0.016768
9	month	0.039724

Figure 47: King County Random Forest Regressor Feature Importance Values

From Figures 46 and 47, it is clear that the *sqft_living* had the highest importance for both the Washington D.C. random forest model and the King County random forest model. However, one distinction that I immediately noticed was the *bathrooms* feature. In the D.C. Random Forest model, the *bathrooms* and *sqft_living* had similar importance. However, the *bathrooms* in the King County model was significantly lower. This highlights a major difference in how the random forest regression algorithm was being affected by different features. Washington D.C. appeared to place higher importance on the number of bathrooms to determine the price of a house, whereas King County placed significantly more importance on the living square feet than any other feature.

c) Evaluation of the Random Forest Models

Similar to my baseline model, I evaluated the Random Forest models using both the R-squared values and Mean Absolute Errors and compare them to their baseline counterparts.

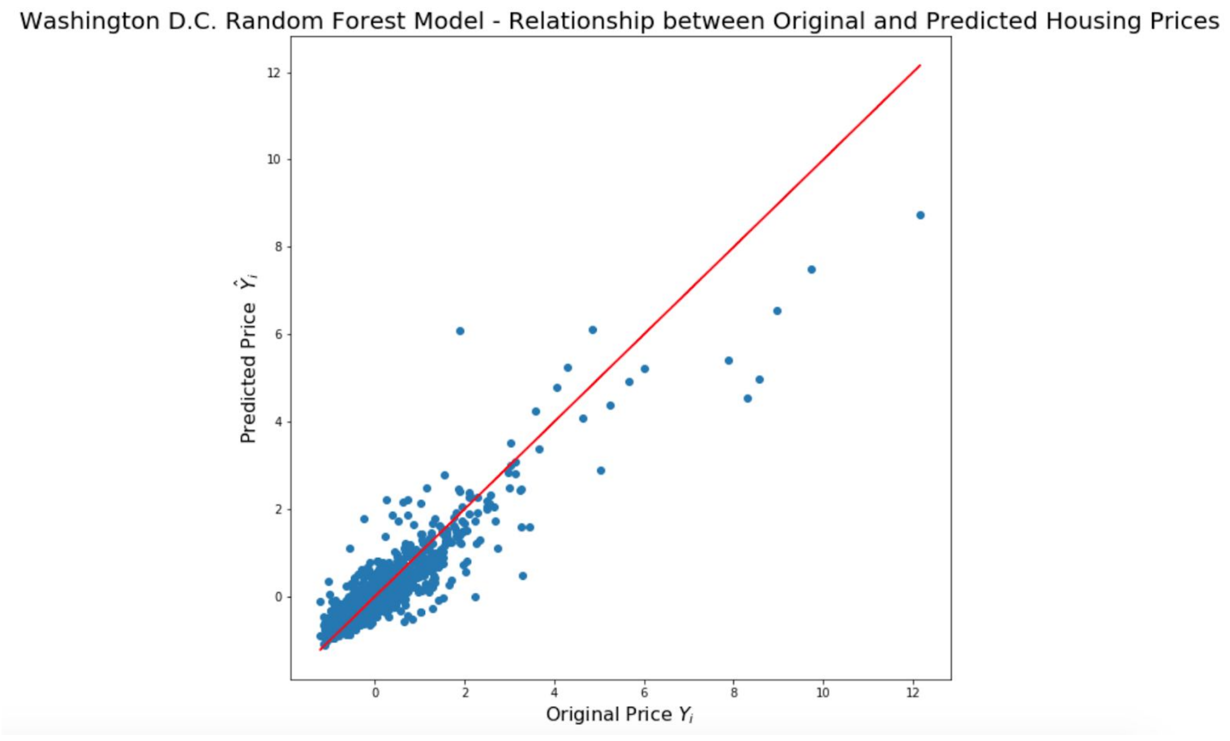


Figure 48: Washington D.C. Random Forest Original vs. Predicted Housing Price on Test Data

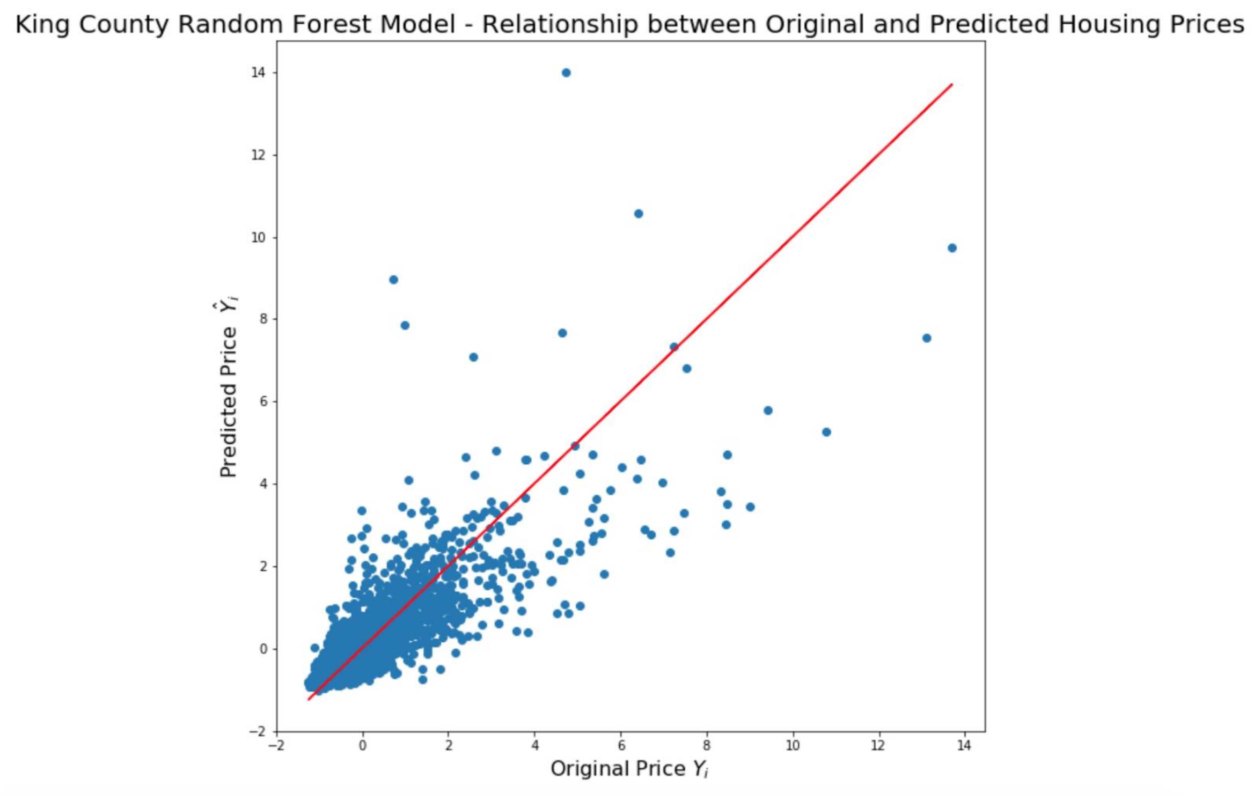
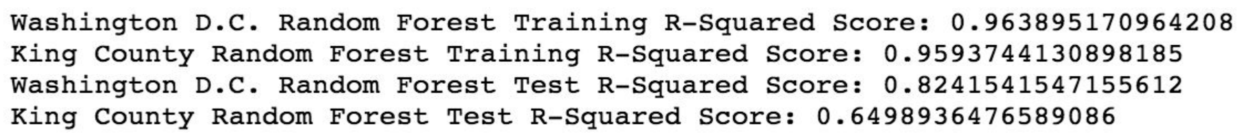


Figure 49: King County Random Forest Original vs. Predicted Housing Price on Test Data

i) R-Squared Values



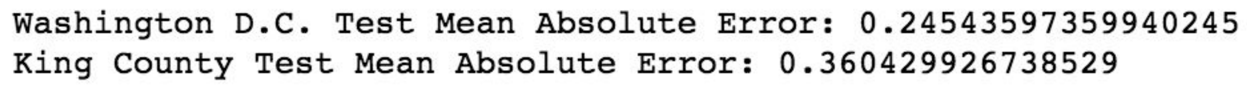
```
Washington D.C. Random Forest Training R-Squared Score: 0.963895170964208
King County Random Forest Training R-Squared Score: 0.9593744130898185
Washington D.C. Random Forest Test R-Squared Score: 0.8241541547155612
King County Random Forest Test R-Squared Score: 0.6498936476589086
```

Figure 50: Random Forest Regressor R-Squared values

For the training data, the R-squared values were high. Both are above .95 out of 1, which means that 95% or more of the variance of the prices are accounted for in both models. It is important, however, to also look at how they did for the test data. The Washington D.C. R-squared value went down, but not as significantly as the King County R-squared value. This means that the Washington D.C. Random Forest model generalized better than the King County model.

Additionally, I compared the R-squared values to the Linear Regression models. The baseline R-squared values for the Linear Regression models were 0.5 for Washington D.C. and 0.56 for King County (for the training data). Already, the R-squared values were lower for the Linear Regression models, which supports the fact that the Random Forest models are stronger. Additionally, the Random Forest models generalized better on the test data.

ii) Mean Absolute Error



```
Washington D.C. Test Mean Absolute Error: 0.24543597359940245
King County Test Mean Absolute Error: 0.360429926738529
```

Figure 51: Random Forest Regressor Mean Absolute Error values

Similar to the R-squared values for the Random Forest models, the mean absolute errors were also significantly lower for the test data when using the Random Forest models. For the Washington D.C. Linear Regression model, the MAE was 0.42, and for the King County model the MAE was 0.45. For both data sets, the MAE dropped when using the Random Forest Regressor, thus making the predictions more accurate and reliable.