PREDICTING HOUSING PRICES IN IOWA

1. Introduction

The aim of the project is to find the algorithm that can best predict housing prices in Ames, a city in the State of Iowa. Ames is a growing city in Iowa that has experienced some recent real estate developments. This project intends to provide insight that can inform decision making in real estate companies operating in this market.

In the following, I will describe the different steps that were taken to arrive at a functional algorithm and its possible business applications. Some of them are:

- Better price their housing assets, providing a benchmark that encapsulates the information in the market.
- Provide insight on the type of housing, relevant features, that will be more profitable to build in future real estate development in the area.

2. Data

The data is the Ames housing database provided by kaggle.com. The data consists of 1460 observations and 79 features describing houses in the area.d. In the following, I will describe the different steps that were taken to arrive at an algorithm that performed well and its possible business applications.

3. Data Pre-Processing

The data will need some treatment before we can start thinking of implementing machine learning models.

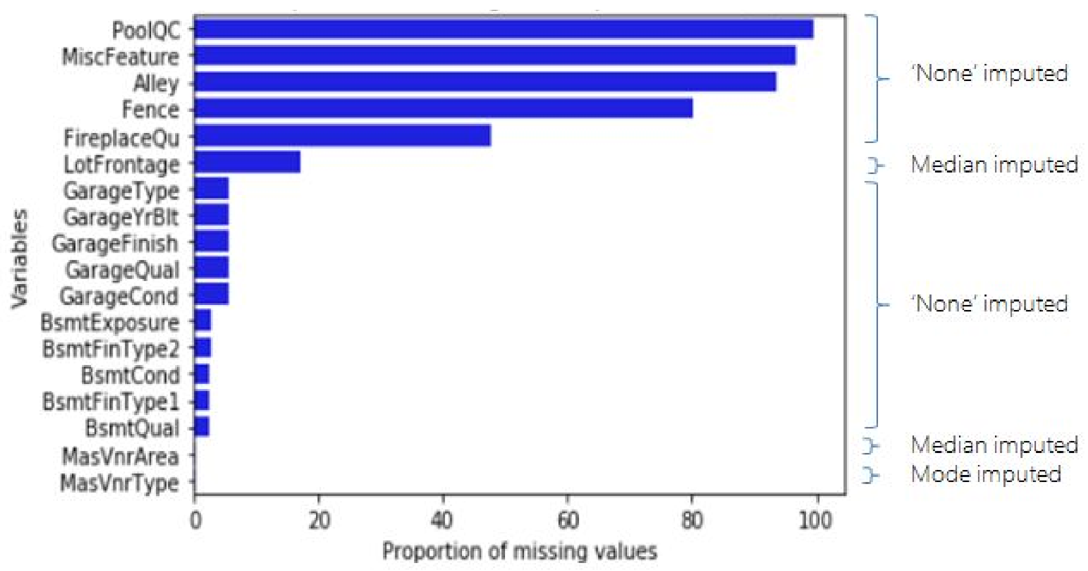
3.1 Splitting the data frame

First, we split the data frame into training and testing sets. Every step in the project will be applied separately to each of them. This way, we make sure that the performance we get from our models can be better generalized to work with data the model has not been trained in.

3.2 Missing data

Now we can have a look at whether there is missing data in any of our datasets. We find that there are 5568 missing values in the training set concentrated in 1168 different rows. In order to check how evenly distributed missing values are across features, we plot the proportion of missing values per variable:

Figure 1: Proportion of missing values in training data set.



We can see that though there is a high number of missing values, they are spread over a few variables that have a high proportion of missing values. When imputing, we followed these steps:

1. Analyze the variables individually and think if some of the missing values have a meaning of their own. For example, the missing values in the feature PoolQC probably indicate the fact that those houses have no pool rather than any other possible issue with the data. The same can be said of the garage and basement variables, given that all of these features have missing values in the same observations, probably indicating that those houses have no basement or garage. This was later confirmed by checking into the variables description provided available in Kaggle. In all these variables, 'None' gets imputed.
2. For numeric variables the median was imputed. In the case of LotFrontage, the data was first grouped by neighborhoods, as we expect the linear feet of street connected to property to be more similar within neighborhood.
3. For the remaining categorical variable (MasVnrType), we impute the mode of the variable.

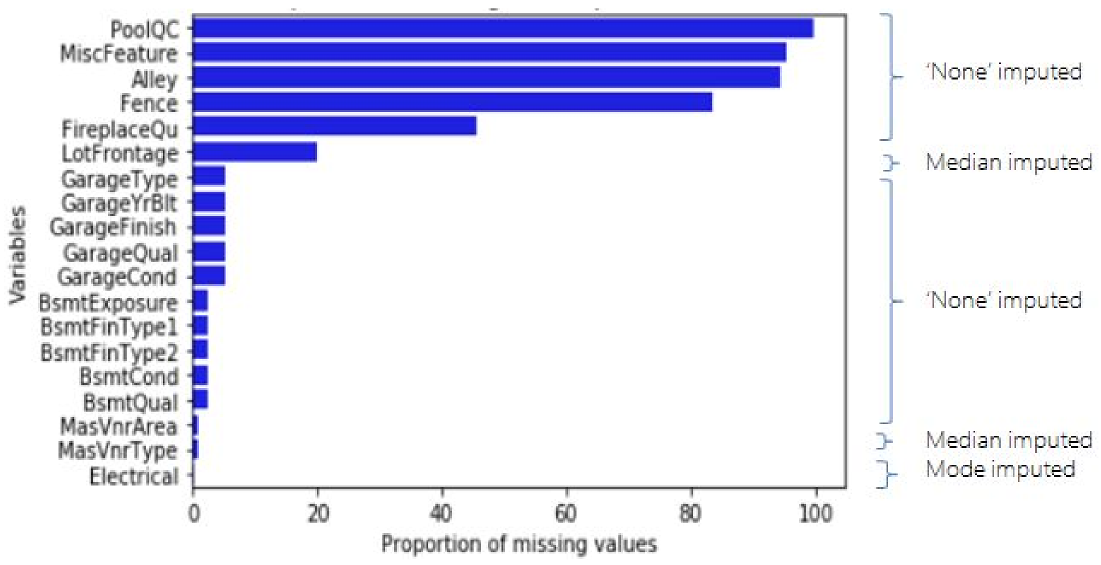
Imputation with k-nearest-neighbors (knn) was considered but finally discarded. The reason for this is that this method of imputation is dependent on the feature space that is formed by our data. Because of this reason, having numeric variables is important, as the distance between the points reflect the internal logic of the data. Given the fact that we have categorical variables, we have two options:

- Impute using only the numeric features: the problem with this is that this can lead to an unrealistic imputation if categorical features play an important role.

- Impute using all variables after dummifying the categorical ones: the problem is that this way forward greatly increases the extension of the feature space, again jeopardizing an effective knn imputation.

Now, we check for the testing set. There are 1397 missing values concentrated in 292 rows. Again, we plot the proportion of missing values in each variable:

Figure 2: Proportion of missing values in testing data set.



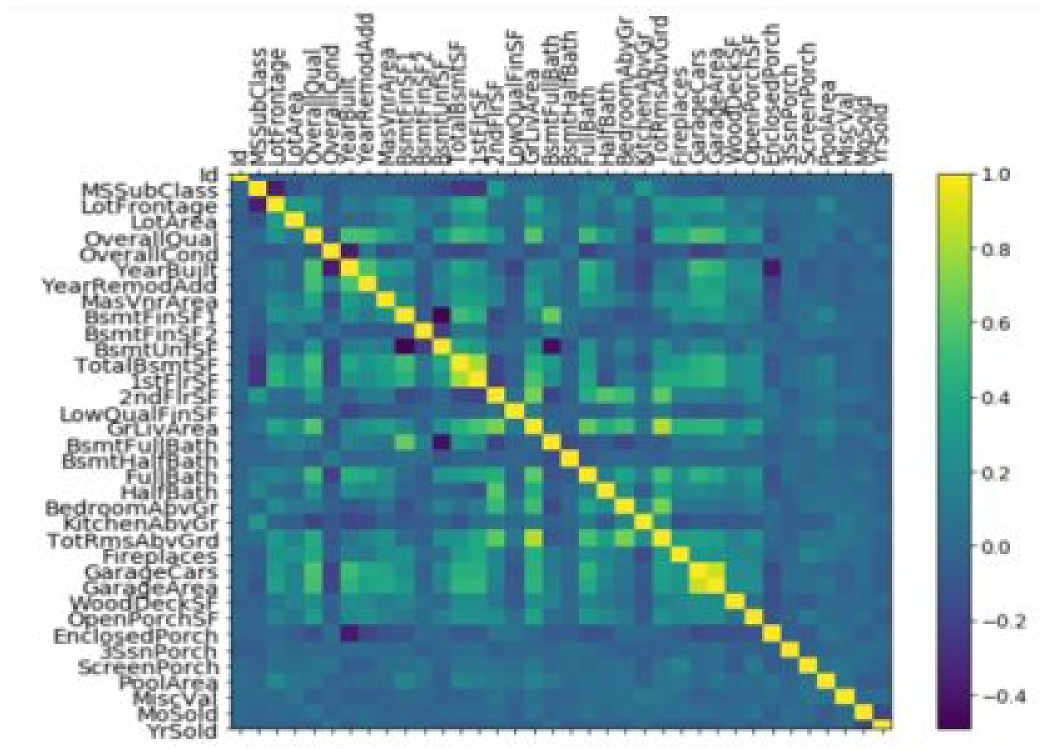
As we could expect, the distribution of missing values is almost equal to the training set. The imputation rules followed here were mostly the same as in the training test. However, some differences apply:

- The training was done in the training set and then applied to the testing one. This implies that the median or mode imputed on the testing set were calculated on the training one.
- Electrical is a feature with only one missing value. By chance, it fell in the testing set when the split was performed. The mode from the training set is imputed.

3.3 Feature engineering

A correlation matrix was used to try to identify multicollinearity between the variables (see Figure 3).

Figure 3: Correlation matrix before feature engineering.



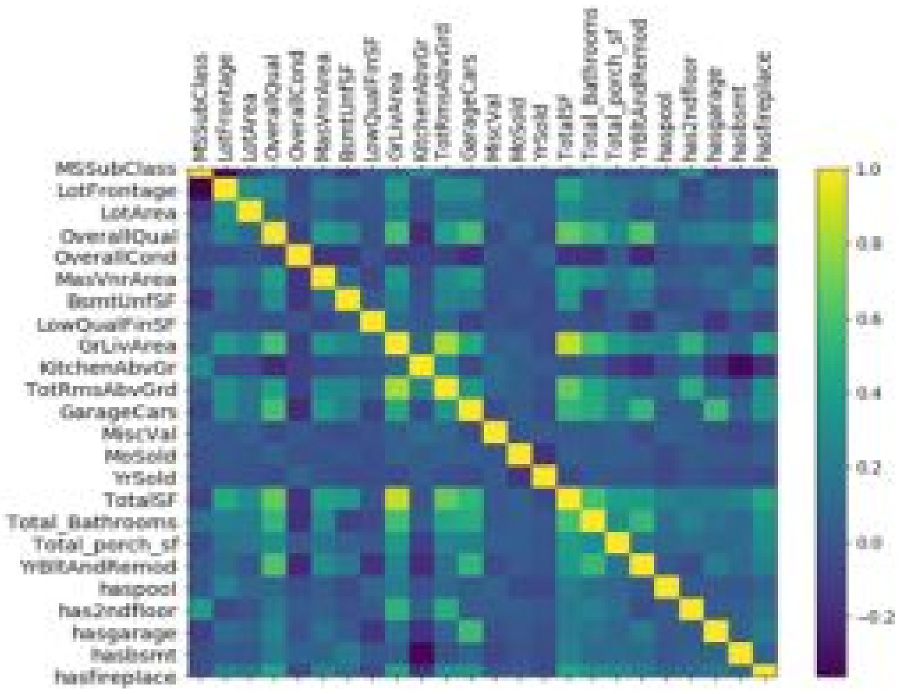
In an attempt to reduce this multicollinearity, we created new variables that encapsulated the information of several other ones. They consisted of the following:

1. Id: this feature is just an index so any correlations with the dependent or any other variables is random.
2. TotalSF: this new feature is created by adding up other variables that add up to the total square footage of each house/observation.
3. Total_Bathrooms: this new feature includes the number of bathrooms in the house/observation.
4. Total_porch_sf: new feature that includes the total square footage of the porch in each house/observation.
5. YrBltAndRemod: this variable takes the number of years since the house was built or remodeled.
6. haspool: dummy variable reflecting whether there is a pool.
7. hasgarage: dummy variable reflecting whether there is a garage.
8. haspool: dummy variable reflecting whether there is a pool.
9. has2ndfloor: dummy variable reflecting whether there is a second floor.
10. hasbsmt: dummy variable reflecting whether there is a basement.
11. hasfireplace: dummy variable reflecting whether there is a fireplace.

This feature engineering reduces multicollinearity between the explanatory variables. Given the large number of features, reducing multicollinearity to zero is not realistic.

The current levels of multicollinearity are visualized in the correlation matrix in Figure 4.

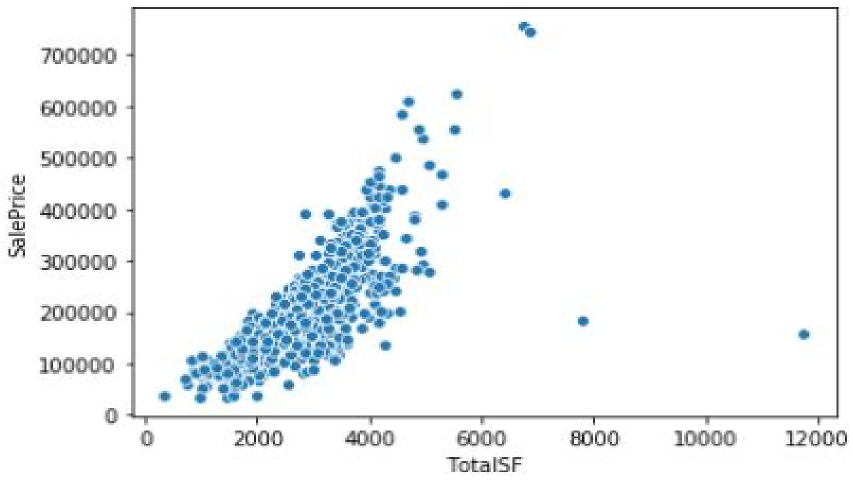
Figure 4: Correlation matrix after feature engineering.



3.4 Outlier detection

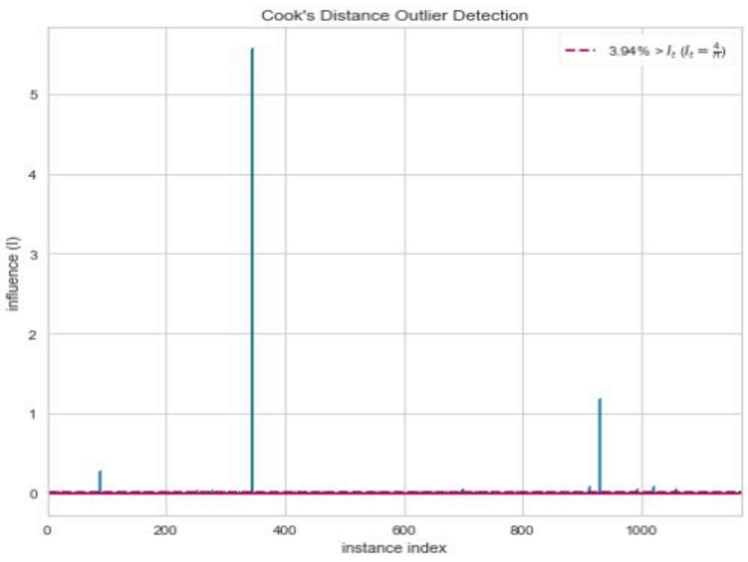
We can start by plotting the dependent variable, the price of the houses ('SalePrice'), and the total square footage ('TotalSF'). As we can see in figure 5, we can visually identify observations that may be outliers. Adopting a more systematic approach, we implement Cook´s distance and also check the observations that fall outside the interquartile range (IQR).

Figure 5: Scatter plot of total square footage and housing prices.



First, Cook´s distance is implemented. The output can be seen in figure 6. The observations that are signaled out are appended to a list.

Figure 6: Cook´s distance plot.

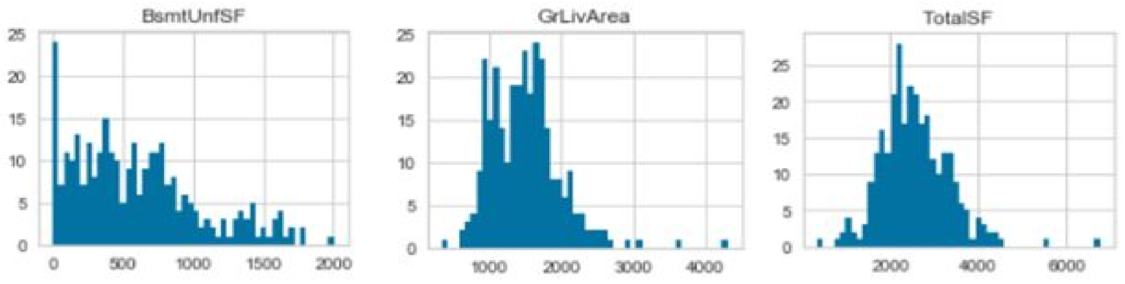


We also try to identify outliers by checking whether they fall out of the inter quantile range (IQR). These outliers are also appended to a different list. Afterwards, we decide to drop the outliers that appear in both lists. The reason we follow this approach is that the Cook´s distance assumes an ordinary least square regression in order to identify outliers. As we also plan to implement different linear models (lasso and ridge in particular) we decided to also use the IQR. This approach also allows for a more conservative treatment of outliers. This is important because we don't always want to remove outliers; they only are eliminated when they reduce the predictive performance of the model. Following this double evaluation we limit the observations we drop to the most clear outliers, the houses that are more set apart and thus less likely to be reproduced in later developments in Ames.

3.5 Distribution of numerical variables

We checked the distribution of our continuous variables (you can see a sample of them in figure 7).

Figure 7: Distribution of a sample of numeric variables before transformation.



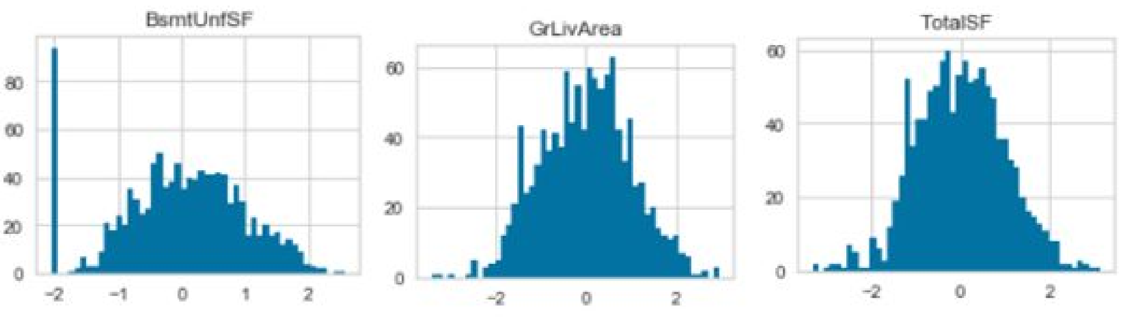
6

Numeric variables are right skewed, which may affect the performance of the linear models. In order to avoid this, we apply a Yeo-Johnson transformation. There are three reasons why the Yeo-Johnson transformation is chosen:

1. Many of our features have zero values and the Box-Cox transformation demands strictly positive values.
2. Yeo-johnson transformation allows us to train in the training data and then apply to the testing data. Therefore, it keeps the rigour we need when later realistically evaluating our model.
3. Yeo-Johnson transformation also scales the data and centers it around zero in our numerical variables.

We can see the results of applying this transformation in figure 8 below:

Figure 8: Distribution of a sample of numeric variables after transformation.



3.6 Dummifying the categorical variables

The categorical variables are dummified in order to be able to implement the linear models.

3.7 Target variable´s transformation

The distribution of the dependent variables is right skewed, which will bias the implemented linear models.. That’s why we perform a logarithmic transformation on the dependent variable. We then check that it is now normally distributed in both the training and testing datasets.

Figure 9: Logarithmic transformation of training set´s target variable.

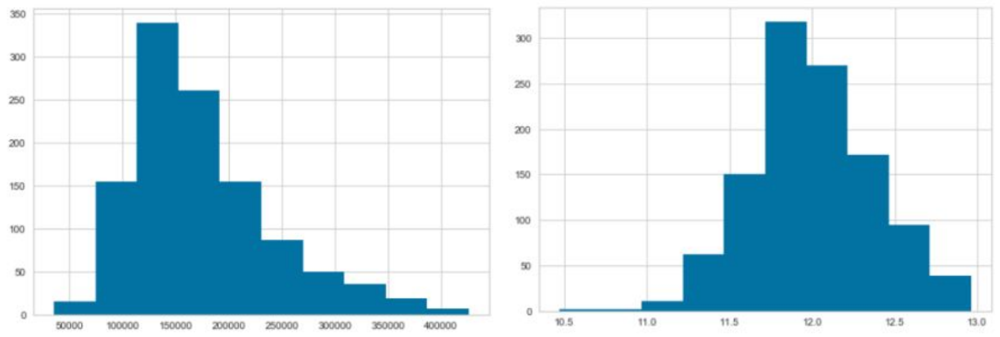
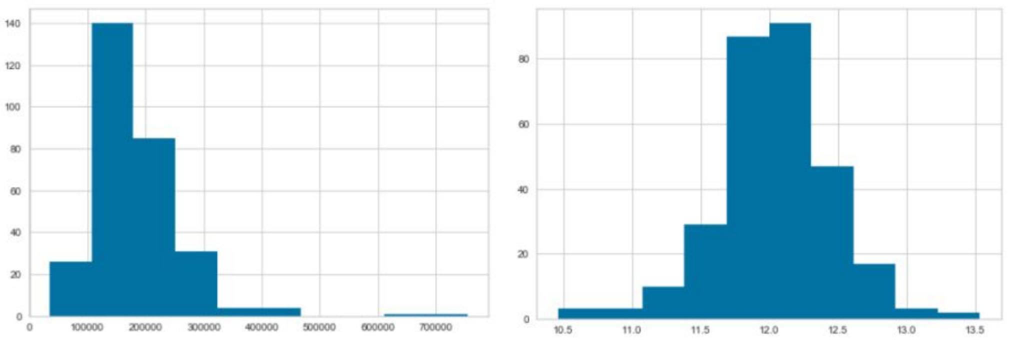


Figure 10: Logarithmic transformation of testing set´s target variable.



4. Machine learning models

4.1 Multiple linear regression model

We start by trying a multiple linear regression model. R-squared for the training set is 0.9569 and 0.9112 for the testing set. The divergence between these scores may be due to overfitting or to the model performing better on the data it has been trained in instead of the unseen data in the testing set.

The assumptions of the linear model are checked below:

Figure 11: Residuals plot (multiple linear regression).

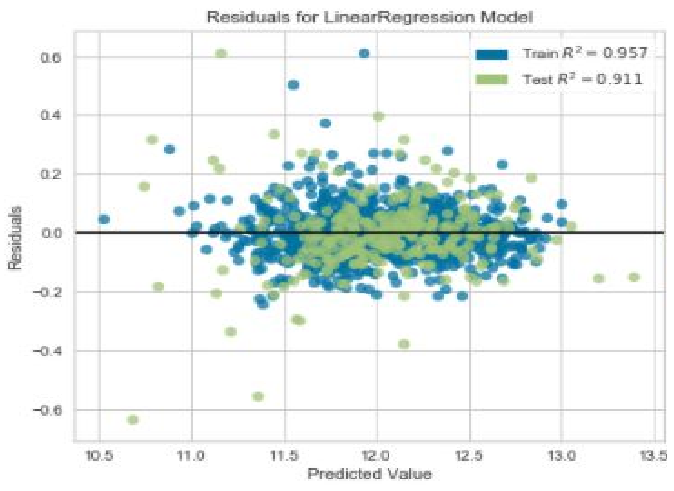
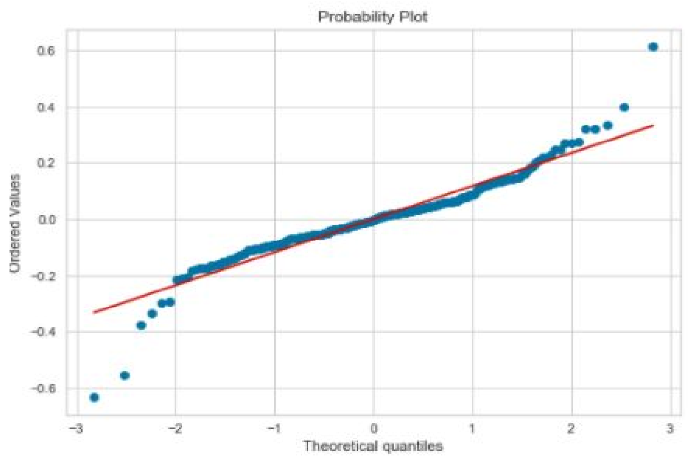


Figure 12: Q-Q plot (multiple linear regression).



As we can see, the model suffers from a certain degree of heteroskedasticity, and the residuals are not strictly normally distributed. The errors however appear to be independent. This is confirmed by implementing a Durbin-Watson test[1]. We also know from the correlation matrices that our model suffers from a certain degree of multicollinearity. The fact that the normality assumptions do not hold should make us skeptical of the reliability of the model. Part of the reason this is happening, and will also happen in different linear models, is the number of zeros that we had in our variables that were not strictly normalized by the Yeo-Johnson transformation. On account of that, we will compare the predictions of the model against the prices in our testing dataset we are trying to predict. This allows an empirical comprobation. We try three different measures:

1. Mean squared error (MSE) = 0.01493
2. Mean absolute error (MAE) = 0.08406
3. Root mean squared error (RMSE) = 0.12222

[1] 1.8999

As explained before, our multiple linear regression model suffers from multicollinearity. In order to solve this problem, we move on to try regression models with regularization properties. We will implement cross-validated lasso and ridge models in order to also correct for possible overfitting in the models.

4.2 Lasso regression

The cross-validated lasso regression has a training R-squared of 0.92946 and a testing set score of 0.92196. The divergence between both scores has decreased in comparison to the multiple linear regression, suggesting that the latter may have suffered from overfitting.

Again, the model suffers from some heteroskedasticity, and the residuals are not exactly normally distributed. The Durbin-Watson[2] test suggests again that the residuals are independent.

Figure 13: Residuals plot (lasso regression).

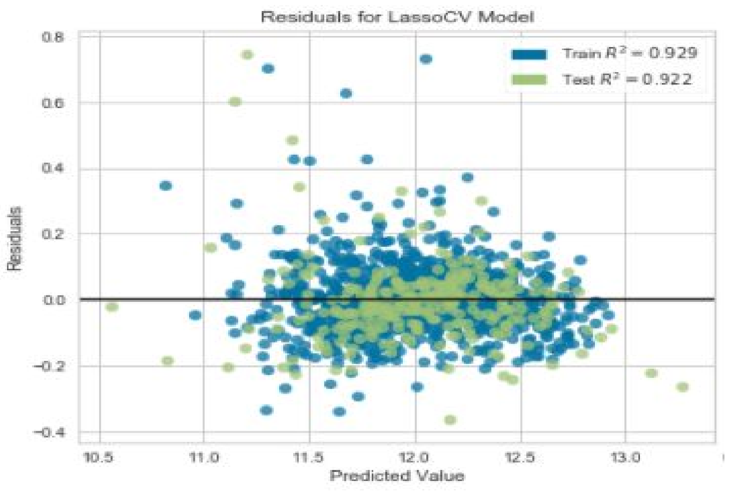
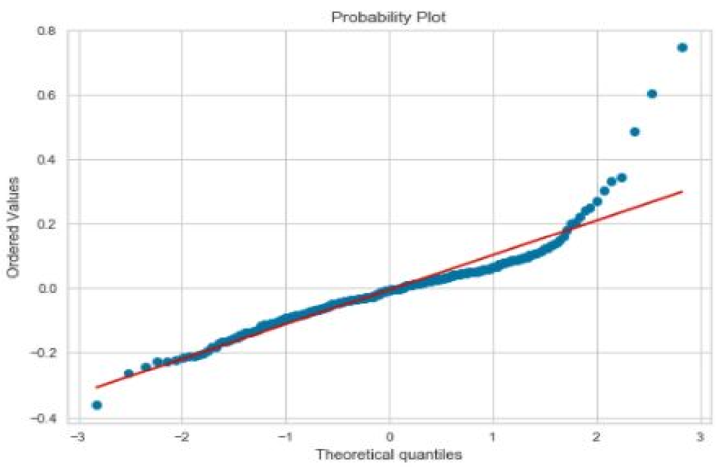


Figure 14: Q - Q plot (lasso regression).



[2] 1.924

We now evaluate the accuracy of the predictions:

1. Mean squared error (MSE) = 0.01313
2. Mean absolute error (MAE) = 0.07634
3. Root mean squared error (RMSE) = 0.11460

4.3 Ridge regression

The cross-validated ridge regression has a training R-squared of 0.92946 and a testing set score of 0.92196.

As with lasso, the model suffers from some heteroskedasticity, and the residuals are not exactly normally distributed. The Durbin-Watson[3] test suggests again that the residuals are independent.

Figure 15: Residuals plot (ridge regression).

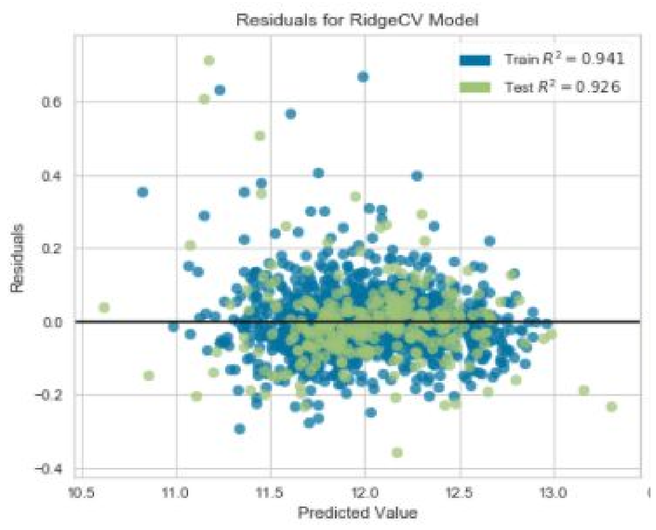
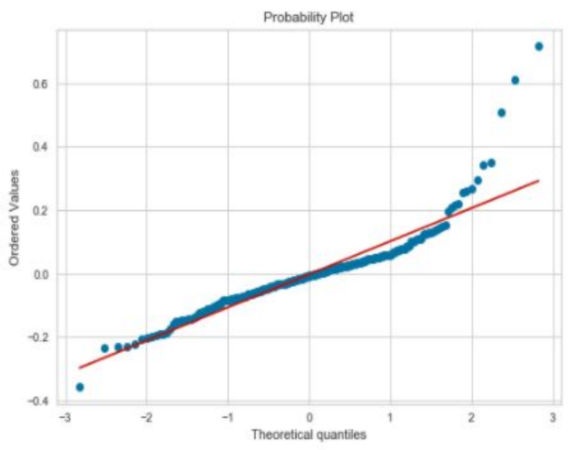


Figure 16: Q - Q plot (ridge regression).



[3] 1.883

The evaluation of the predictions is as follows:

1. Mean squared error (MSE)   = 0.01252
2. Mean absolute error (MAE) = 0.07323
3. Root mean squared error (RMSE) = 0.11190

4.4 Gradient boosting

In addition to the linear models implemented above, we also try non-linear, tree-based, models. There may be non-linear relationships in our data that our tree-models are able to capture, perhaps offering better performance that the linear models tried above. We will start by implementing gradient boosting.

We also cross-validate our gradient boosting model in order to avoid overfitting. The R-squared in the training set is 0.93337 and the one from the testing set is 0.88656. The evaluation of the predictions are:

1. Mean squared error (MSE) = 0.01909
2. Mean absolute error (MAE) = 0.08601
3. Root mean squared error (RMSE) = 0.13817

4.5 Random forest

Random forest in the other non-linear model that we try. The R-squared in the training set is 0.98281 and 0.86796 in the testing set. The evaluation from the predictions are:

1. The mean squared error (MSE) is 0.02151
2. The root mean squared error (RMSE) is 0.14666
3. The mean absolute error (MAE) is 0.09125

5. Conclusions

The model that performs best here is the ridge regression. The R-squared in the testing score is the highest, which implies that this model is the one that explains the most variance in the model. Also, the predictions are the best ones according to the evaluation metrics that we have picked. They suggest that our model can predict housing prices with a margin error of 10%, which is the industry's standard.

This information can help inform decision-making at the business level. As stated above, it can provide insight on the pricing of real estate assets just by plugging the house's characteristics in and letting the model return a price. In addition, it can provide information on which features of a new house are more valuable for potential customers.

We found the following insights about the correlation between features and housing prices that can be useful for planning purchases or renovations for buyers, sellers, and developers :

1. The variable that has the biggest causal effect on prices is total square footage.
2. The second most influential feature is OverallQual, which reflects overall material and finish quality. These are expected results.
3. Proximity to the main road or railroad is also within the top 5 features. This can be important in case of new developments.
4. There are 3 features reflecting different neighborhoods between the top and bottom 5 features that have an effect on price. This speaks to the importance of where the new developments take place. Crawfor seems to be the neighborhood that positively affects the prices most. This does not mean that it is the neighborhoods with the most expensive housing, just that building in that location has the biggest positive causal effect.
5. A garage condition of 'fair' has little impact on the price of the house (fifth from the bottom). On the other hand, garages that are ranked as 'Average,' which is just slightly higher quality than 'fair' have a significantly higher effect on the price of the house. This suggests that trying to upgrade low-quality garages to average may be a worthwhile investment in light of the return in a higher sale price.
6. Another insight comes from the fact that an unfinished basement has little effect on the price of houses. When remodeling houses, bear in mind that there is not much ROI for finished basements.