Project Overview

My project is to utilize machine learning to predict how much to negotiate on a house you're looking to purchase. The idea came to me when a friend of mine asked me help her come up with a price for a house she wanted to bid on. Many sites such as Zillow already provide an estimate of a house is worth based on their own proprietary formula. However, these estimates can be sustainably different than the price the seller wants to sell at.

Problem Statement

What makes my project different is I want to attempt to predict the difference between the listing price and the sale price rather than the actual value of the house. The listing price is the price the seller is advertising to sell their home for. The sale price is what the home actually sells for after negotiating with the buyer.

Data set

The data set was compiled and prepared by my friend's realtor. The data contains recent sales of homes within a 5 mile radius of the home she was looking to purchase. It came directly from the multiple servicing list service (MLS). The MLS databases allows real estate brokers who share their listing with one another for the purpose of locating ready, willing, and able buyers more efficiently. This database is only accessible by professional real agents. I'm considering turning this project into an app in the future. Therefore, we only consider numerical features as they are easier for users to enter.

- # Bedrooms – number of bedrooms in the home
- # Baths – number of bathrooms in the home
- # Rooms – total number of rooms in the home
- Fin SF – total size of the house in square feet
- List Price – the price the seller wants to sell the home for
- Sales Price – the price the home actually sells for
- Days On Markets – number of days the home has listed for sale on the market before being sold

Data Preprocessing

In this section, we will preprocess the data by cleaning data and removing outliers. Preprocessing data is often times a critical step in assuring that results you obtain from your analysis are significant and meaningful. Finally, we will need to determine which feature and target columns.

Data Cleaning

We need to clean the data before we can do any analysis. The statistics above shows that the minimum value for Fin SF is 0. This is already a problem since a home cannot have Fin SF of zero for its size. All the values in each column must be greater than zero. Therefore, any row that contains values less than 1 will be removed. Rows with null values will be removed as well. Finally, we make sure that # Rooms is at least equal to the sum of # of Bedrooms and # Baths. 4 bad data points have been removed from the data set, because they were either null, less than 1, or are inconsistent.

Outlier Removal

Detecting outliers in the data is extremely important in the data preprocessing step of any analysis. The presence of outliers can often skew results which take into consideration these data points. There are many "rules of thumb" for what constitutes an outlier in a dataset. Here, we will use [Tukey's Method for identifying outliers](http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/): An *outlier step* is calculated as 1.5 times the interquartile range (IQR). A data point with a feature that is beyond an outlier step outside of the IQR for that feature is considered abnormal. The data points had more than one outlier per feature were removed. Those data points are more likely to be irregularities since there were outliers found in multiple features. They should be removed, because regression models are sensitive to outliers.

Identify feature and target column

The feature columns are the inputs we will use to predict our target column. A new buyer will have access to every column except for Sales Price when they're purchasing a new home. They cannot know what the sale prices without thoroughly negotiating with the seller. However, we trying to predict the

difference between the List Price and Sales Price rather than the Sales Price. Therefore, we need to take the difference between the List and Sale Price to create our target column.

Data Exploration

In this section, we will begin exploring the data through visualizations and code to understand how each feature is related to the others.

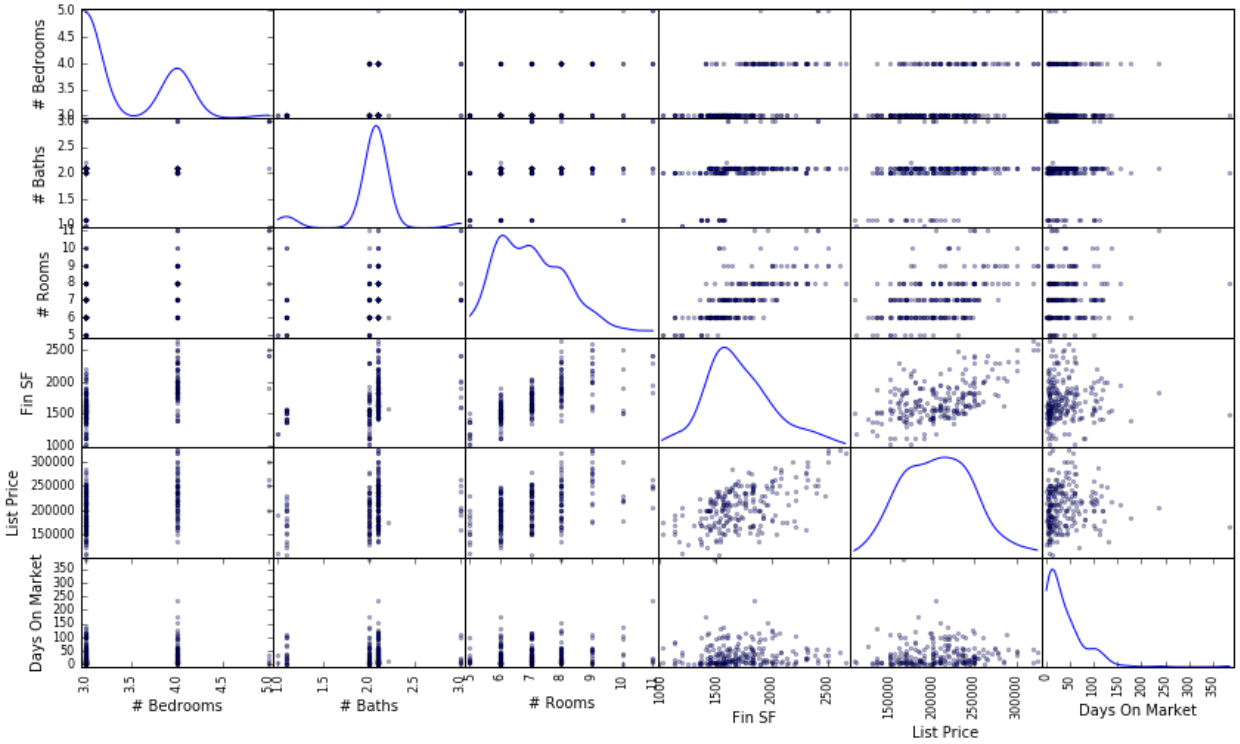
Feature Relevance

We need to consider is if any features are actually relevant. We can make this determination quite easily by training a supervised regression learner on a subset of the data with one feature removed, and then score how well that model can predict the removed feature.

- The feature # Bedrooms has a reported prediction score: -0.0732807215333
- The feature # Baths has a reported prediction score: -0.0178975289683
- The feature # Rooms has a reported prediction score: -0.0018943380341
- The feature Fin SF has a reported prediction score: 0.245262820137
- The feature List Price has a reported prediction score: -0.755191273519
- The feature Days On Market has a reported prediction score: -0.91701582358

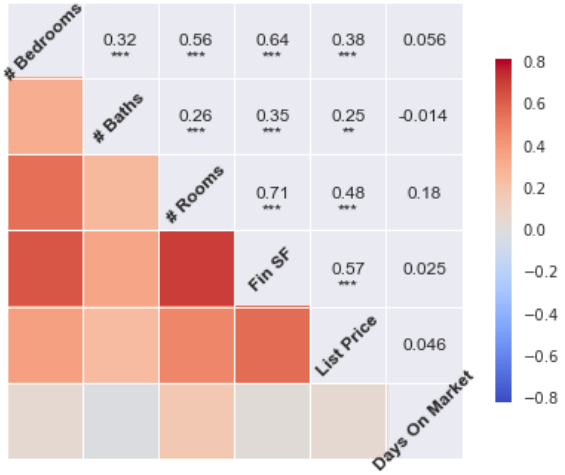
The coefficient of determination r^2 is telling you how much of the information contained in the selected feature is already contained in the remaining features. If the coefficient of determination is 1.0 then all the information contained in this feature is already contained in the renaming features (since we can predict the feature perfectly from the remaining ones). If the coefficient of determination is less than or equal to zero then the remaining features do not already contain the information found in the selected feature (i.e. the select feature is telling something new about the sample). Almost all the features have a negative r^2 score. The only feature with a positive r^2 was Fin Sf, but the r^2 was still very low. It appears every feature is relative.

Visualize Feature Distributions



A z-test was performed on each feature to determine whether or not the feature follows a normal distribution. It appears none of the features except for List Price follows a normal distribution. This means we will need to do feature scaling.

Correlation Matrix



All the features are positively correlated with one other except for # bathrooms and Days on Markets. It seems having more bathrooms helper a house seller quicker. The correlations make sense. An increase in rooms will increase the size (Fin SF) of the home. Bigger homes generally take longer to sell. # Room should be at least # Bedrooms + # Bathrooms. Therefore positive correlation between #Bedrooms, # Bathrooms, and # Rooms is expected.

Feature Scaling

The data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate to apply a non-linear scaling — particularly for financial data. One way to achieve this scaling is by using a Box-Cox test, which calculates the best power transformation of the data that reduces skewness. A simpler approach which can work in most cases would be applying the natural logarithm.

Training and Evaluating Models

In this section, we will tried various supervised learning models that are available in scikit-learn. We tested the models against one model, and select the best model. Then, we will fine tune our best model, and use it to make predictions.

Define a Performance Metric

Normally, r^2 is a good general metric to use in a situation like this. However, r^2 uses some assumptions about the linearity of effects in the model; explained variance score might be more appropriate if you don't expect the models to be linear in most cases (for example, SVMs, nearest neighbors, or neural nets). These model also penalize outliers.

Training and Testing Data Split

We need to split the data into training and testing sets to estimate how our model will perform on new data. This will also check to see if we are overfitting model to the data. An over fitted model will perform well on the training set but poorly on the testing set. We will use 200 data points for training our model. Then we will test our models on the remaining data points.

Choosing the Best Model

We will try various different regression models from scikit-learn. A regression model from each category was selected. Ensemble regressions were not considered due to the limited amount of data that we have. The regression models performed much better after implementing log feature scaling. The estimator with out of sample performance was the K Neighbors. It achieved an explained variance score of 0.2401.

Training set size: 200

Training LinearRegression...

Done!

Training time (secs): 0.000

Predicting labels using LinearRegression...

Done!

Prediction time (secs): 0.000

Performance score for training set: 0.0954831544834

Predicting labels using LinearRegression...

Done!

Prediction time (secs): 0.000

Performance score for test set: 0.181391879412

Training set size: 200

Training KernelRidge...

Done!

Training time (secs): 0.000

Predicting labels using KernelRidge...

Done!

Prediction time (secs): 0.002

Performance score for training set: 0.0595059934826

Predicting labels using KernelRidge...

Done!

Prediction time (secs): 0.000

Performance score for test set: 0.164951989561

Training set size: 200

Training SVR...

Done!

Training time (secs): 0.003

Predicting labels using SVR...

Done!

Prediction time (secs): 0.001

Performance score for training set: 0.000710364706033

Predicting labels using SVR...

Done!

Prediction time (secs): 0.001

Performance score for test set: 0.00244911471521

Training set size: 200

Training SGDRegressor...

Done!

Training time (secs): 0.000

Predicting labels using SGDRegressor...

Done!

Prediction time (secs): 0.000

Performance score for training set: 0.0534301129411

Predicting labels using SGDRegressor...

Done!

Prediction time (secs): 0.000

Performance score for test set: 0.210887698594

Training set size: 200

Training KNeighborsRegressor...

Done!

Training time (secs): 0.001

Predicting labels using KNeighborsRegressor...

Done!

Prediction time (secs): 0.001

Performance score for training set: 0.212154423466

Predicting labels using KNeighborsRegressor...

Done!

Prediction time (secs): 0.001

Performance score for test set: 0.240127177968

Training set size: 200

Training DecisionTreeRegressor...

Done!

Training time (secs): 0.002

Predicting labels using DecisionTreeRegressor...

Done!

Prediction time (secs): 0.000

Performance score for training set: 1.0

Predicting labels using DecisionTreeRegressor...

Done!

Prediction time (secs): 0.000

Performance score for test set: -1.6571797509

Model Tuning

We will use fine tune the best our model by using grid search to find the best hyper parameters. We will search for the optimal n_neighbors, and optimal weight settings. The optimal n_neighbors was 9 and the optimal weight settings was uniform. Unfortunately, our fine-tuned model performed slightly worse than the model with default parameters. It achieved an explained variance score of 0.2117 which is less than our original score of 0.2401. An optimal model is not necessarily a robust model. Sometimes, a model is either too complex or too simple to sufficiently generalize to new data.

Predicting Selling Prices

We're trying to use the fine-tuned model to predict how much room my friend has to negotiate the price. The home had 4 beds, 2.5 baths, 7 rooms, and is 2,088 fin sf in size. The home was listed for $195,000 and had been on the market for 43 days. We need to transform our input be applying the natural logarithm. The model predicts that my friend has about $3305 to work with. It suggests she should place an offer for that home at 191695.

Conclusion

My friend did end up getting this particular using the price suggested from the model. However, I don't think this model is very robust. The variance score for various estimators I tried were all very low. This means we need to gather more feature and more data points. This data set was very limited.