Definition

Project Overview

My project is to utilize machine learning to predict how much to negotiate on a house you're considering purchasing. The idea came to me when a friend of mine asked me to 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's worth based on their own proprietary formula. However, these estimates can be substantially different than the price the seller wants to ask for.

Dataset

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 listing service (MLS). The MLS database allows real estate brokers to share their listings with one another for the purpose of locating ready, willing, and able buyers more efficiently. This database is only accessible by professional real estate agents. I'm considering turning this project into an app in the future. Therefore, we only consider numerical features as they are easiest 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 Market – number of days the home has been listed for sale on the market before being sold

Problem Statement

What makes my project different is that I want to attempt to predict the difference between the listing price and the selling 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 selling price is what the home actually sells for after negotiating with the buyer. Therefore, we can compute our target variable with the following formula:

$Target = List_{price} - Sales_{price}$

My strategy for solving the problem is to first clean the data and then remove any data outliers. I will then analyze the numerical distribution of each feature variable to see how they impact one another. This analysis will tell me which features to include or exclude. I may also need to do feature scaling if the data is not normally distributed. After that, I will try out various supervised learning models to see which model has the highest out-of-sample accuracy. The model with the highest accuracy will be further fine-tuned and optimized. Finally, I will use that model to make predictions to answer my friend's original question which was how much flexibility on price does she have to work with?

Metrics

This is a regression problem rather than a classification because the values we are trying to predict are continuous rather than discrete. Therefore, we should only consider performance metrics for regression problems. 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). This metric is more fitting than r^2 because I will be using nonlinear models such as SVMs and nearest neighbors to solve my problem.

Explained Variance Score

$explained\ variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$

$\hat{y}$ is the predicted target value.

y is the actual target value.

Explained Variance Score compares the residual variance against the total variance of your dataset. The residual variance will be less than or equal to the total variance of the dataset. No residual variance means your model explains the dataset perfectly since there is no difference between your predicted target values and the actual target values. A residual variance equal to the total variance means your model does not explain the data at all. The explained variance score is 1 minus the ratio of the residual variance and total variance which is between zero and one. Therefore, the higher the score, the more the model explains the data.

Data Preprocessing

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

Data Cleaning

We need to clean the data before we can do any analysis. 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. Four 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 that 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 will be without thoroughly negotiating with the seller. However, we are trying to predict the difference between the List Price and the 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.

Analysis

In this section, we will begin exploring the data through visualizations and code. We will first compute some descriptive statistics to gain some general insights on the data. Then, each feature will be examined to determine whether or not they are relevant or redundant. The features will be also tested for normality. Finally, we will study the correlation between features.

Feature Relevance

We need to consider whether 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

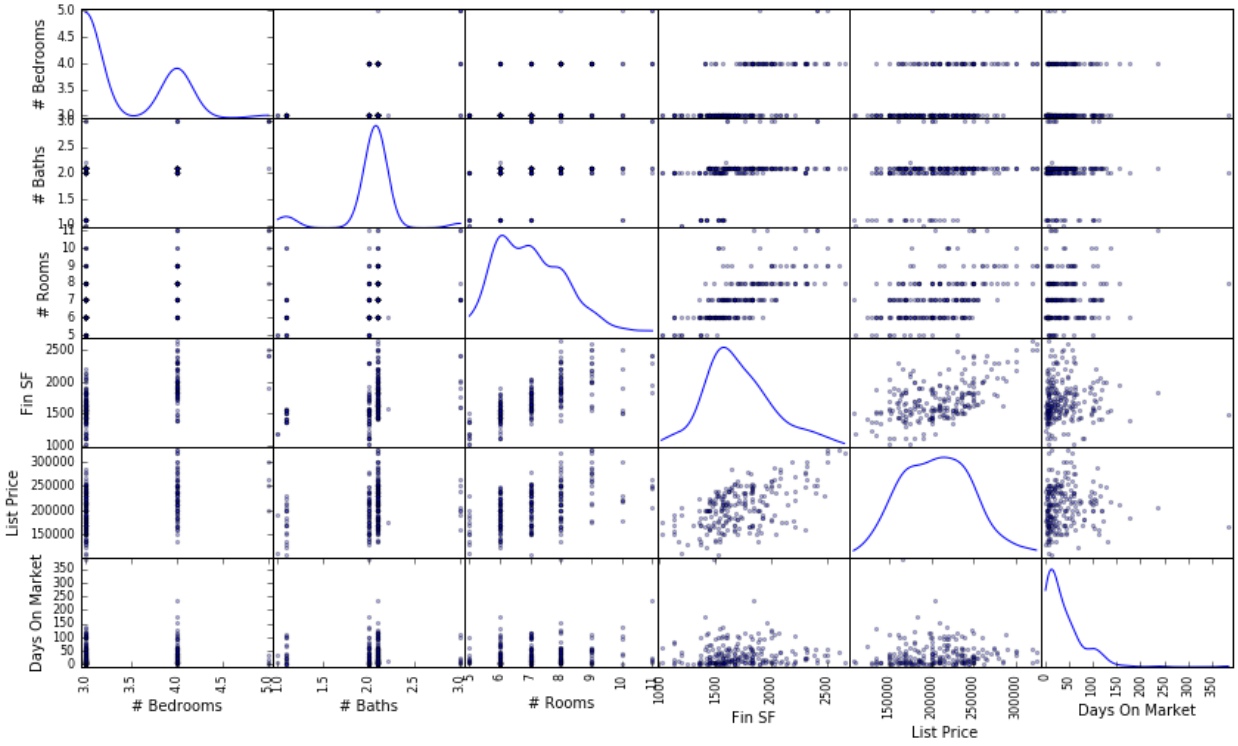
The feature Fin SF has a reported prediction score: 0.245262820137

The feature List Price has a reported prediction score: -0.755191273519

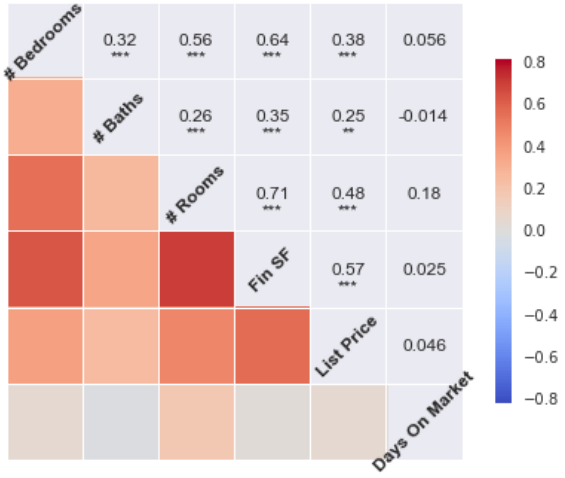
The feature Days On Market has a reported prediction score: -0.91701582358

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. Each feature contains enough information that is not found in other features to include in our estimator. It appears that every feature is relevant.

Exploratory Visualization



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



All the features are positively correlated with one another except for # Baths and Days on Market. # Baths and Days on Market has a small negative correlation value of -0.014. The correlation value is so small that one can conclude there is not a significant relationship between # Baths and Days on Market. The other correlation make a lot of sense. We should expect # Rooms to be highly correlated with # Bedrooms and # Baths since # Rooms is the sum of number of bedrooms, bathrooms, and other rooms a home has. Rooms take up space so logically more rooms means more Fin SF. Bedrooms are usually larger than bathrooms. You can see the relationship between FIN SF and # Bedrooms is stronger than the relationship between FIN SF and # Bedrooms. There does not appear to be any significant relationship between Days on Market and the other features except for # Rooms. Homes with more rooms take slightly longer to sell.

Algorithms and Techniques

I decided to take a brute force approach and try out as many different regression models from scikit-learn. I wanted to try everything that I can think of to help my friend with her home purchase. The data set is small enough for me to do something like this. However, in the real world such an approach might be not feasible due to time and computational constraints. If I had to pick one, I would use K-NN regression

because the model is nonparametric. Nonparametric model do not make any assumptions about the probability distribution of the dataset being assessed. It addresses the issue that housing data from different regions most likely have different probability distributions as well. Another advantage is it can pick up on weird systematic but localized features of data set. The disadvantage of K-NN regression is a lazy learner. There is no computational cost in the learning process. However, you must query the all the necessary data points each time you wish to make a prediction. This is not a major issue in our case since we are already getting the recent sales data that near the home each time we're make a prediction. The table below shows the advantages and disadvantages of other regression models that I will try.

Model	**Strengths**	**Weaknesses**
Linear	• Fast and easy to use	• Data needs to be independent and linear
Kernel Ridge	• Faster training time than SVR • Faster than SVR for datasets less than 1000 samples	• Slower prediction time than SVR • Does not scale as well as SVR and is slower for greater datasets
SVR	• Work well in complicated domains with a clear margin of separation • Data does not need to be linearly separated due to the kernel transformations and several functional forms • Effective in high dimensional spaces.- • Still effective in cases where number of dimensions is greater than the number of samples	• Very slow • Do not work well with large datasets • Might over fit to the noise in the data • Results depend on model selection
SGD	• Fast enough to work on very large datasets • Easy to code and implement	• Requires feature scaling • Has many hyper parameters
Decision Tree	• Very fast • Requires very little data preparation • Can be used for feature selection • Easy to interpret and explain results	• Prone to overfitting • Works with poorly with independent features • Need to monitor complexity and prevent the tree from growing too complex

Benchmark

I was not able to find any discussion on what constitutes a good score or threshold for explained variance score besides higher is better. There is lots of discussion on what consists of a good r^2 score. We can use the guidelines for r^2 as an approximation for an good explained variance score since both metrics are between 0 and 1. A r^2 of 0.70 to 0.90 is considered good for engineering and physical sciences. A r^2

that is only 0.05 might be acceptable in social sciences such as psychology because human beings are very difficult to predict. However, r^2 greater than 0.40 is viewed with suspicion in social sciences for the same reason. My expectations are not high, because the data set we're working with is really limited. We have a small number of data points and a small number of features. Given what we're working with, I expect a good reasonable explained variance score would be greater than 0.10.

Methodology

In this section, we will try various supervised learning models that are available in scikit-learn. The data has been preprocessed. The data will be set into two sets; one for training our models and the other one for testing. We will test the models against one another, and select the model with best performance in the testing set. Then, we will fine tune our best model, and use it to make predictions.

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 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 will work in most cases would be applying the natural logarithm. Feature scaling needs to be perform, because some of the regression models we're using such as linear regression require the data follows a normal distribution to work properly.

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 over fitting the 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.

Selecting the Best Model

We will use a variety of 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. Each model was tested with the default parameter first. We will select the model that has the highest explained variance score.

Results

In this section, we analyzed relative performance of each supervised learning model to another one. We will also examined the performance of the final model.

Model Evaluation and Validation

Model	Training Time	Prediction Time	Score(Train)	Score(Test)
Linear	0.000	0.000	0.0954831544834	0.181391879412
Kernel Ridge	0.000	0.002	0.0595059934826	0.164951989561
SVR	0.003	0.001	0.000710364706033	0.00244911471521
SGD Regressor	0.000	0.000	0.0534301129411	0.210887698594
K-Neighbors Regressor	0.001	0.001	0.212154423466	0.240127177968
Decision Tree	0.002	0.001	1.0	-1.6571797509

All the regression model other than the Decision Tree Regressor achieved a positive explained variance score. The Decision Tree has a perfect score on training data set but it failed on the training set. This is definitely a case of overfitting. Other regression models such as Linear, Kernel Ridge, and SGD did significantly better on the testing data vs. on the training model. Training performance should be better than testing performance, because the model is optimized and fitted to the training data. The models are suffering from high bias as the models are not really learning from the data. The K-NN Regressor achieved the highest explained variance score among all these estimators. It achieved an explained

variance of 0.2121 in the training set and 0.2401 in the testing set. These two scores are almost identical which is really good. The model is able to explain both sets of data equally well.

Model Tuning

We will fine tune our best model by using grid search to find the best hyper parameters. We will search for the optimal n_neighbors, and optimal weight settings.

- n_neighbors: number of clusters to break the data set into
- weights can be set to uniform or distance
    - Uniform weights. All points in each neighborhood are weighted equally.
    - Distance: weight points by the inverse of their distance. Closer data point will have a greater influence than data points which are further away.

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 are trying to use the fine-tuned model to predict how much room my friend has to negotiate the sales price of the home. The house had 4 bedrooms, 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 by applying the natural logarithm. The model predicts that my friend has about $3305 to work with. It suggests that she should place an offer for that home at $191,695.

Conclusion

Reflection

We started out by preprocessing the data before we did any type analysis. Invalid data points and outliers were removed from the data set. Our analysis showed us that each feature was relevant and not exclude any of them from our model. We also discovered that most of the features were not normally so we had to log feature scaling to normalize them. Afterwards, we tried many different regression model and found out none of them worked really well. The variance score for an assortment of estimators I tried were all very low.

My friend did follow the recommendation from the model. There is a happy ending here as she end up getting this particular house. However, I don't think the model contributed a lot to her success due to the many shortcomings that I have mentioned.

Improvement

The biggest improvement would be getting more and better data. The fact that many different regression failed means the data set is not very good. I would like to try this approach on more homes to see the results are any better. I need to find a free API that has the data I'm looking for. I need to be able to systemically get the recent sales data around x mile radius of the home I'm looking at. The data right now is gathered and compiled by manually by hand. This makes such a task infeasible.