Ames Housing Prices: Advanced Regression

Capstone Project Report

Davin Buczek

Definition

Housing markets are incredibly competitive. I am experiencing this first hand as I am trying to become a first-time home owner. I am interested in the application of machine learning techniques in context of housing markets to gain a competitive edge and use the abundance of data associated with housing market to my advantage. Using a dataset of 79 characteristics (variables) of nearly 1500 homes in the Ames, Iowa area a potential home seller or buyer could possibly influence price negotiations of a house by examining more than just the number of bedrooms, number of bathrooms and square footage of a house. The Ames Housing dataset was compiled by Dean De Cock for use in data science education. The Kaggle competition and dataset details can be found at https://www.kaggle.com/c/house-prices-advanced-regression-techniques

Problem Statement

The problem set forth is utilizing this housing dataset to its maximum potential by building a model to accurately predict house prices. This strategy has many critical components to be successful including feature selection, model selection and model tuning. Being able to identify the optimal features using data exploration to identify patterns, correlations and feature engineering to determine the appropriate inputs for the model is a crucial aspect in concert with selecting an appropriate model and tuning it to be robust and reliable. My initial approach to the problem was the scenario given a real estate ad with the most commonly listed features and attributes and no price, could a machine learning algorithm accurately predict a price or competitive counter offer given data from the surrounding geographic area. The expected solution will be a robust supervised learning model that is capable of predicting housing prices given a dataset of features.

Metrics

The performance metric best suited for this type of advanced regression model is Root-Mean-Squared-Error (RMSE). The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Taking the RMSE between the logarithm of the predicted value and the logarithm of the observed sales price. Taking logs means that errors in predicting expensive houses and cheap houses will affect the result equally.

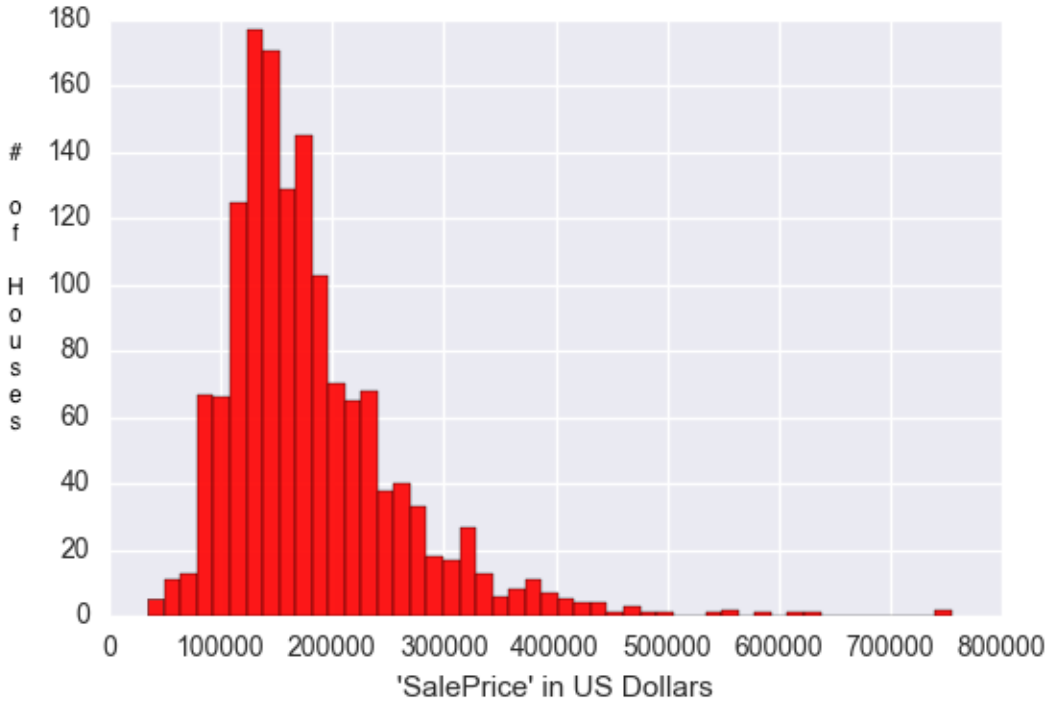
Analysis

Data Exploration

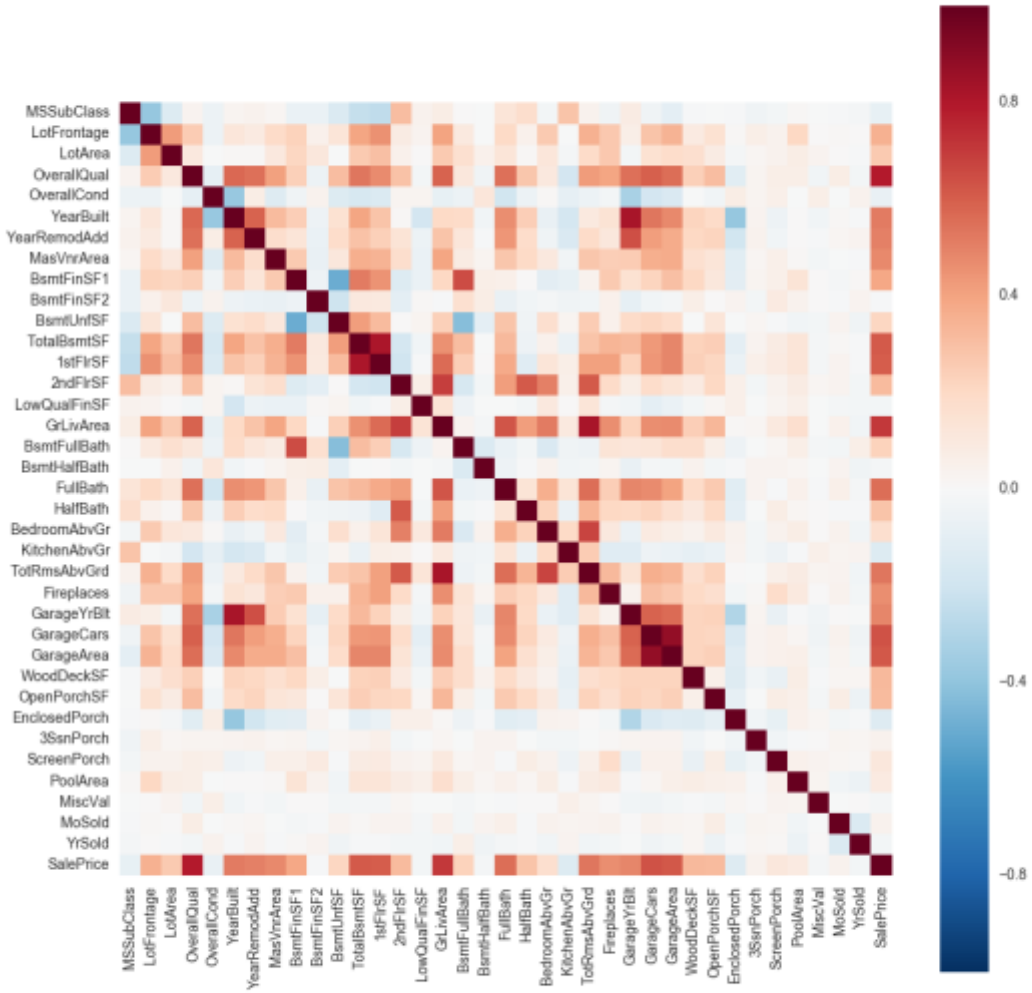
Given the problem domain is relative sale prices of homes based on a calculated subset of features, elementary statistics of the housing data will be useful to analyze various prediction results from the implemented model. Statistics to be calculated include minimum price, maximum price, mean price, median price and the standard deviation

Exploratory Visualization

To better understand the target output value 'SalePrice', I have included some visualizations including a simple bar graph visualization of the values from the column labeled 'SalePrice' in the training dataset. As well as a heat map to get a high-level overview of which features in the dataset correlate highly with 'SalePrice' to guide further feature observation and engineering.



The average 'SalePrice' from the dataset is $180,921.20, while the min and max are $34,900.00 and $755,00.00 respectively. The are a few outliers but most the data points are within one standard deviation of $79,442.50.



To better understand and use the heat map effectively for feature observation. A sorted dictionary was created that list the numerical features in descending order by their correlation with 'SalePrice'.

Feature Observation

- 'OverallQual' is the Rating of the overall material and finish of the house (1-10, Very Poor to Very Excellent)
- 'GrLiveArea' is above grade (ground) living area square feet
- 'GarageCars' is size of garage in car capacity
- 'GarageArea' is size of garage in square feet
- 'TotalBsmtSF' is total square feet of basement area
- '1stFlrSF' is first Floor square feet
- 'FullBath' is full bathrooms above grade
- 'TotRmsAbvGrd' is total rooms above grade (does not include bathrooms)
- 'YearBuilt' is the original construction date
- 'YearRemodAdd' is remodel date (same as construction date if no remodeling or additions)

To most effetively select features to inform the model it is a reasonable approach to look at factors that realtors use to advertise properties for sale. By identifying key features that are most often found by looking at local newspaper ads, open house fliers and online listings we can gain insight into what features most likely influence 'SalePrice'. Looking at the top 10 features that correlate most (>0.50) highly with 'SalePrice' several features are highly correlated amongst certain descriptive categories such as Garage space wich contains attributes 'GarageCars' and 'GarageArea'. The correlation graphs and data above along with my intuition and sampling of various physical and online real estate ads should allow for a initial reduction in the feature set for ease of analysis and model building to ensure changes and optimizations are generating reasonable outputs. It is ideal to start with a simpler model with less features initially to make sure the data pipeline is functional and reliable. After the initial model is proven robust it could then be revisited and made more complex by adding additional features to the model. The features chosen for an initial analysis include OverallQual', 'GrLiveArea', 'GarageCars', 'FullBath' and 'TotRmsAbvGrd'.

- 'OverallQual' was chosen because it is a numerical value that is a good overall indicator of quality. At a quick glance a realtor could give a score from (1-10) and the buyer would have a general impression of the house. This feature is unique because it is subjective in nature and not normally an aspect that is advertised in a real estate ad.
- 'GrLiveArea' was chosen because it is a numerical physical feature that is most commonly found when describing a house for sale in a real estate ad.
- 'GarageCars' was chosen for similar reasons because it is a numerical physical feature that is commonly found when describing a house for sale in a real estate ad. It was chosen over 'GarageArea' due to garage space most commonly referred to by number of cars and not square footage when referring to size.
- Both 'FullBath' and 'TotRmsAbvGrd' were chosen because these two features are the two biggest factors that dictate the sale price of a house. All five features chosen represent important features of a house that would most commonly be found in a real estate ad that do not correlate too highly amongst each other.

Description of the features and their respective value ranges can be found in the data_description.txt file included with the project files.

Algorithms and Techniques

Decision trees

For this regression type problem, the algorithm most suited for the characteristics of this problem are decision trees. The goal of a decision tree is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. There are many advantages to using a decision tree for this problem because they are visual, simple to understand and easy to interpret. Additionally, to implement a decision tree there is little data preparation to do which allows for faster iterations and more nimble and adaptable analysis. Decision trees are also easy to refine by augmenting the minimum number of samples required at a leaf node or setting the maximum depth of the tree to avoid overfitting. There are some disadvantages to using decision trees including creating overly complex trees that do not generalize well. Decision trees are sensitive to small variations in data that may cause an entirely new tree to be created unnecessarily.

Random Forest Regressor

Another technique that was also considered for this project was a random forest regressor. A random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement. Additionally, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. Because of this randomness, the bias of the forest usually slightly increases. However, using a random forest regressor is not as easily interpretable by a human compared to the visual nature of a decision tree where you can follow the logic through the tree. Additionally, the training speed is not as fast as a decision tree although it may yield more accurate predictive results. Lastly given a high signal to noise ratio a random forest regressor will most often underperform when compared to a decision tree.

Support Vector Machines

Support Vector Machine models because they excel in data sets where there are a lot of features compared to the overall size of the data set. One weakness of SVM is data sets with noise. If there is noise present it would be hard to margin and may require computationally intensive kernels to classify all the data points correctly. SVM models are a good candidate for this data set because it does not contain a significant amount of noise. SVM are also very versatile with the capability to implement different kernel functions which can be specified for the decision function. However, one major disadvantage to consider for SVM is if the number of features is significantly larger than the number of samples it is likely to provide poor performing predictions.

Grid Search

Grid search is a way of automatically testing multiple combinations of parameter tunes while cross-validating for best performance. The grid search trains and evaluates the model on each grid, then systematically searches the best performing grid.

Cross-Validation

k-fold cross validation takes a large amount of data and splits it into 'k' equal buckets. It then chooses 1 bucket to use for testing data and the rest for training. After you run 'k' testing sets, you average the

results from those 'k' experiments. k-fold cross-validation although requires more compute time provides much more accurate test results which makes a grid search more efficient when optimizing a model.

Algorithm Selection

Taking into consideration all the advantages and disadvantages as well as the unique dataset I have chosen to implement a decision tree model in combination with grid search and k-fold cross-validation. Decision are simple to understand and easy to interpret and can be enhanced using grid search and cross-validation techniques.

To ensure that an enhanced model is produced it will be trained using the grid search technique to optimize the ‘max_depth’ parameter for the decision tree. The 'max_depth' parameter can be thought of as how many questions the decision tree algorithm can ask about the data before making a prediction.

Once a model has been trained on a given set of data, it can now be used to make predictions on new sets of input data. In the case of a decision tree regressor, the model has learned what the best questions to ask about the input data are, and can respond with a prediction for the target variable, ‘SalePrice’. These predictions can be used to gain information about data where the value of the target variable is unknown — such as data the model was not trained on.

Benchmark

Additionally, to appropriately benchmark the output prediction values from the model using the testing data set a reasonable metric for assessing performance would be to look at the statistical attributes of the dataset. For this specific type of problem domain of predicting housing prices a good attribute to determine the robustness of the model would be the percent difference in the average 'SalePrice' from the training data compared to the predictions from the model using the test data.

To appropriately compare the performance of the model a benchmark result will be calculated using the Root-Mean-Squared-Error (RMSE). The benchmark results will be a score that indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. The benchmark will be comparing the average housing price from the subset of training data ('benchmark_avg') and comparing it against the actual housing prices from the subset of the training data ('y_train'). The average housing price is chosen as a benchmark because housing prices a historically grouped by geographic areas and pricing can be generally associated to regional trends in the market. For the purposes of this project's scope assessing performance against a benchmark scored against the average housing cost should be more than adequate. As for the model predicted results ('preds') that performance will be assessed directly from the known labels of the subset of the training data set that was split for testing ('y_test') the model performance. A percent difference < 10% could be classified as robust in terms of initial performance before any model tuning is applied.

Methodology

Data Preprocessing

In order to implement the decision tree model the training data has to be processed. The first step in processing the data is creating a new data structure only containing the features selected from the feature observation section above. There are no obvious abnormalities or characteristics about the training dataset that needs to be addressed.

	OverallQual	GrLivArea	GarageCars	FullBath	TotRmsAbvGrd
count	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	6.099315	1515.463699	1.767123	1.565068	6.517808
std	1.382997	525.480383	0.747315	0.550916	1.625393
min	1.000000	334.000000	0.000000	0.000000	2.000000
25%	5.000000	1129.500000	1.000000	1.000000	5.000000
50%	6.000000	1464.000000	2.000000	2.000000	6.000000
75%	7.000000	1776.750000	2.000000	2.000000	7.000000
max	10.000000	5642.000000	4.000000	3.000000	14.000000

All five features are described with numeric values and are present for all 1460 data points in the training dataset.

Implementation

To implement and train a model using the decision tree algorithm I followed several steps listed below:

- Imported 'GridSearchCV' from 'sklearn.metrics' to find the optimal 'max_depth' parameter for the decision tree. Grid search is a way of automatically testing multiple combinations of parameter tunes.
- Imported 'DecisionTreeRegressor' from 'sklearn.tree' to create a decision tree regressor object.
- Imported 'make_scorer' from 'sklearn.metrics' to create a scoring function object.
- Created a dictionary for 'max_depth' containing the values from 1 to 10.

Train/test Split

Performing a train/test split on a data set is important for many reasons. Firstly, it is necessary for a supervised learning model to train on a dataset with both features and correct labels. Secondly, it is necessary to test a supervised learning model on a portion of the dataset with features absent the correct labels. A train/test split allows the model to be trained on a portion of a dataset (80%) and then tested on the remaining (20%). Lastly, with a train/test dataset split you can go back and verify the model's performance by comparing the predicted label given a feature with the actual label. A train/test dataset split can be easily achieved by using cross_validation available the sklearn.

Complications

One complication I experienced during the coding of this project was the constant incompatibility of sklearn modules due to impending deprecation. Working on this project over an extended period would cause some modules to become out of date and stop functioning properly. Modules would either produce errors or no output at all. The lesson I learned is to not update any packages or modules over the course of a single project or there will be headaches.

Refinement

When the model is trained with a maximum depth of 1 it appears to suffer from high bias where it is underfit due to the model being too simple and unable to represent the complexity. When the model is trained with a maximum depth of 10 it appears to suffer from high variance where it is overfit because the decision tree is too large. At the maximum depth of 1 both the training and testing scores are low and at the maximum depth of 10 the training score is high while the testing score is low.

Parameter 'max_depth' is 5 for the optimal model.

When the model is trained with a maximum depth of 1 it appears to suffer from high bias where it is underfit due to the model being too simple and unable to represent the complexity. When the model is trained with a maximum depth of 10 it appears to suffer from high variance where it is overfit because the decision tree is too large. At the maximum depth of 1 both the training and testing scores are low and at the maximum depth of 10 the training score is high while the testing score is low. I think a 'max_depth' of 5 results in a model that best generalizes the unseen data because it appears to be the sweet spot where it minimizes bias and variance due to model complexity. There is sufficient data to reduce bias and the error between the training and validation score is reasonable.

The final parameters used in the implemented decision tree model are as follows:

- 'max_depth' = 5
- 'min_samples_split' = 2
- 'random_state' = 0

Results

Model Evaluation and Validation

I think a maximum depth of 5 results in a model that best generalizes the unseen data because that appears to be the sweet spot that minimizes bias and variance due to model complexity. There is sufficient data to reduce bias and the error between the training and validation score is reasonable. At a maximum depth of 5 the training and validation scores are trending close enough together to predict a reasonable 'SalePrice' while also being able to provide a generalized model that is not to biased or over-fit to the training data and therefore I believe predicts useful results.

At this point, there are diminishing returns from trying to replace a good learning algorithm with an exceptional one for this specific problem. Instead most of the improvements will come from feature engineering that will enable the learning system to extract more patterns and generalize more accurately.

Justification

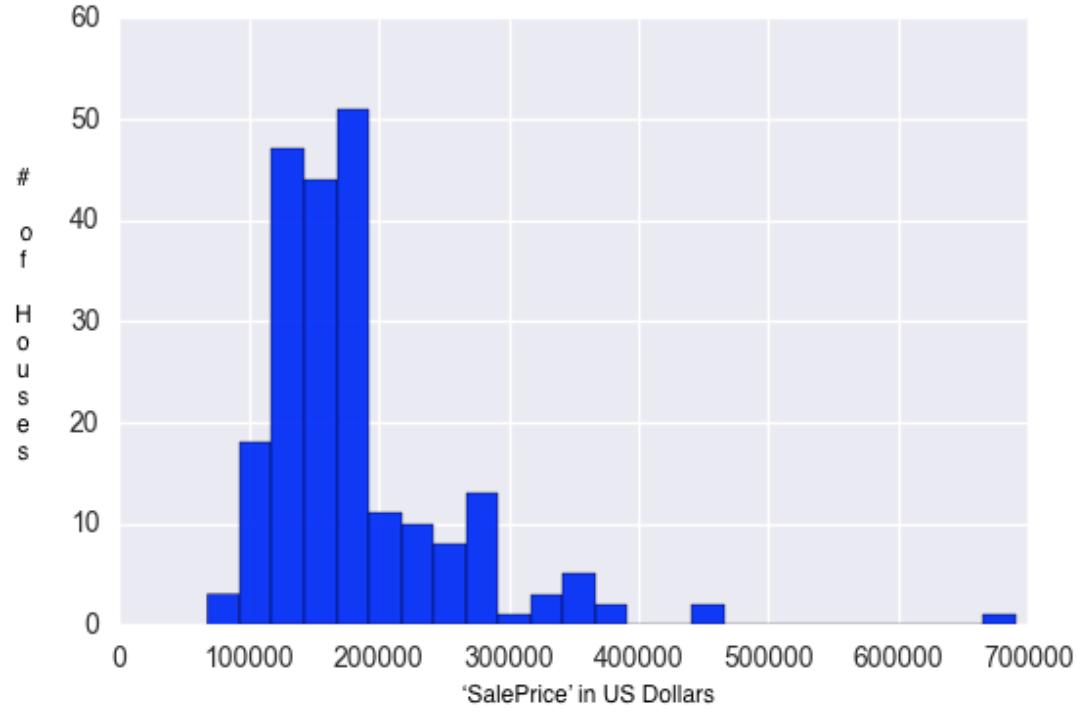
Looking at the percent difference between the RMSE of the benchmark (training data and average house price values) and comparing them to the RSME of the prediction (actual house price and predicted values). The percent difference is small.

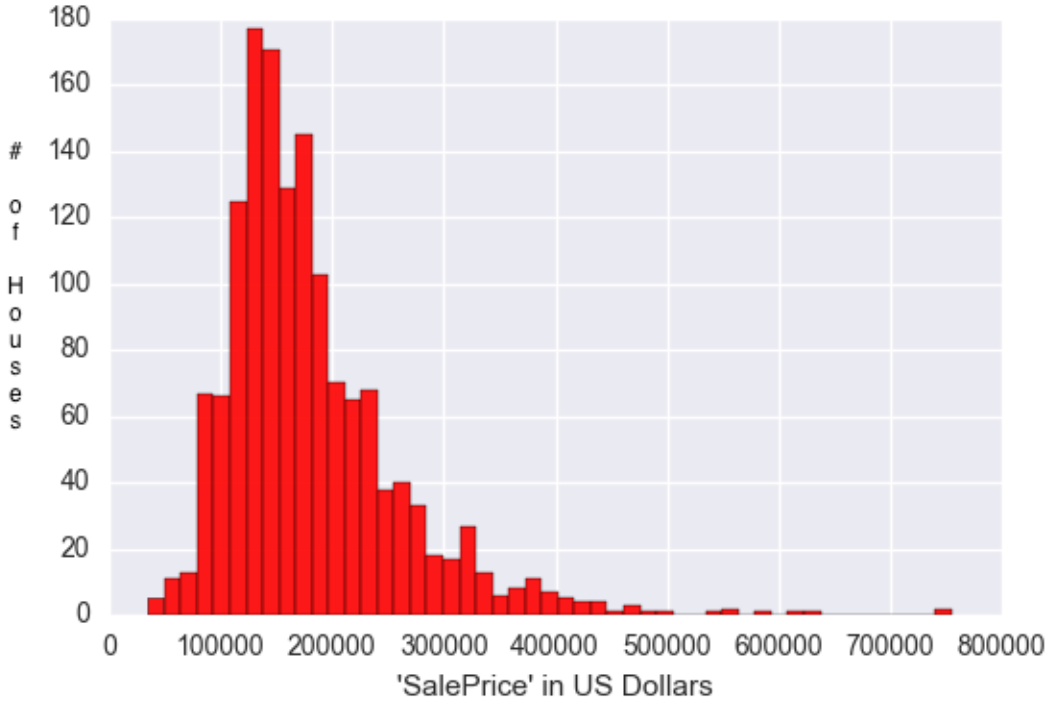
RSME – Benchmark	RSME – Prediction	Percent Difference
22.57	20.80	-7.842%

Visualization

The table below represents the percent difference of actual house price and model predicted price in USD of 5 data points. Given such a small data sample it is hard to say conclusively that the model is perfect but the output is reasonable in terms of the features values for each case.

Actual Price (USD)	Model Prediction (USD)	Percent Difference (%)
$200,624	$185,660	7.74
$133,000	$139,033	4.43
$110,000	$120,937	9.47
$192,000	$233,164	19.36
$88,000	$120,937	31.52





At a high level, both 'SalePrice' graphs have a similar shape and trend. Overall I think the comparison of the two graphs represents a rough although robust model for predicting housing prices given a set of engineered features for analysis. The predictions graph in blue could have a bit more fidelity but as an initial survey of the data it well represents the high-level trends of the original red 'SalePrice' graph.

Reflection

Initially given a wide range of features that describe a given house it was relatively straight forward to identify and filter out the most relevant features that most highly correlated with the 'SalePrice' of the house. Determining a benchmark to assess and engineer a robust model was one of the more difficult tasks of this project. The implemented solution involves a comparison between a benchmark score and a prediction score. The benchmark score is calculated by doing an RSME between the average 'SalePrice' of a portion of the training set and comparing it to the actual 'SalePrice' data. I took this approach for this problem due to the clustering behavior of real estate markets in each geographical area. By comparing the average price to the actual price over a given geographical area you can get a baseline housing price range to gauge the robustness of the model's predictions. The score that is compared against the benchmark is calculated by looking at the output prediction from the model and comparing it to the actual 'SalePrice' of the home from the other portion of the train_test data split. The final analysis was done by comparing the raw scores from the benchmark results and the prediction results and assessing performance based on percent difference. Initially a 10% threshold was used a metric of a successful and robust model given most housing prices are negotiable within that range.

After completion of this project an analysis of the results it became clear that the most important features to determine a given houses' price are more physical in nature and can be easily referenced on a realtors' flyer posted outside the house for sale. Although there are many other factors a homebuyer may take into consideration such as location, neighborhood or school districts it is most beneficial to consider features such as overall quality, square footage, number of bedrooms and number bathrooms when trying to determine the 'SalePrice' of a house.

Improvement

One aspect of the implementation of the model that could be improved for further analysis would be the use of something like a RandomForestRegressor in place of a heat map to determine optimal correlation of features to determine 'SalePrice'. A random forest is a pre-process estimator that fits many of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. I chose the heat map to determine correlation due to its visual nature and how easy it is to interpret. Going beyond the scope of this project a random forest could provide a richer analysis but lead to a similar outcome. A random forest could help with over-fitting and provide a wider range of prediction values given a more diverse feature set. I believe the current implementation of the model and feature set is robust enough to provide an initial set of reliable predictions.