Machine Learning Engineer Nanodegree

Capstone Project

RABIA KHAN

November 2017

Definition

Project Overview:

This document covers a predictive analytics report on data exploration, transformation, regression analysis and validation completed on housing data provided on Kaggle.com. I have chosen this dataset as it is similar to the Boston housing market dataset and that was quite an interesting project for me. The feature that is being predicted is the house prices given other correlated variables like condition of house, no. of rooms etc.

The dataset I am working with comprises of 79 explanatory variables of residential homes in Ames, Iowa. Some of the features include to name just a few:

- SalePrice - the property's sale price in dollars. This is the target variable for prediction.
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access

> **"The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset" Kaggle**

In Supervised Learning, we can use algorithms that predict continuous numerical outcomes namely regression models, or we can use classification models which classify what class a data point belongs to (discrete values). Prediction models that use regression algorithms are most suitable for this kind of problem as we are predicting house prices from a continuous set of values. Our predictive model is trained on features of houses we know the prices of (training data) and used to predict unknown house prices.

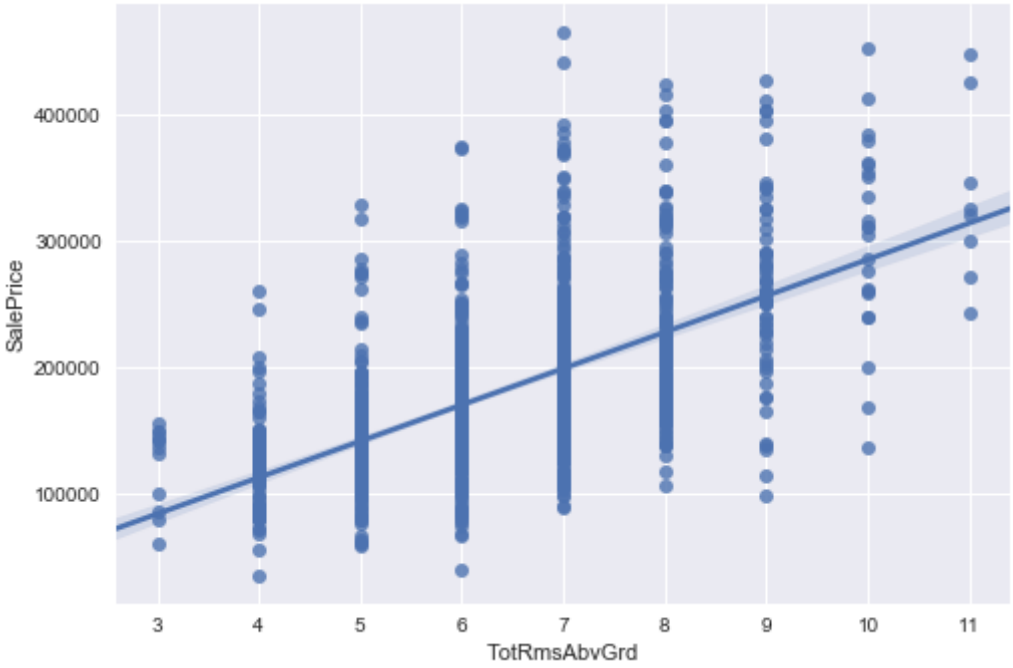
Problem Statement

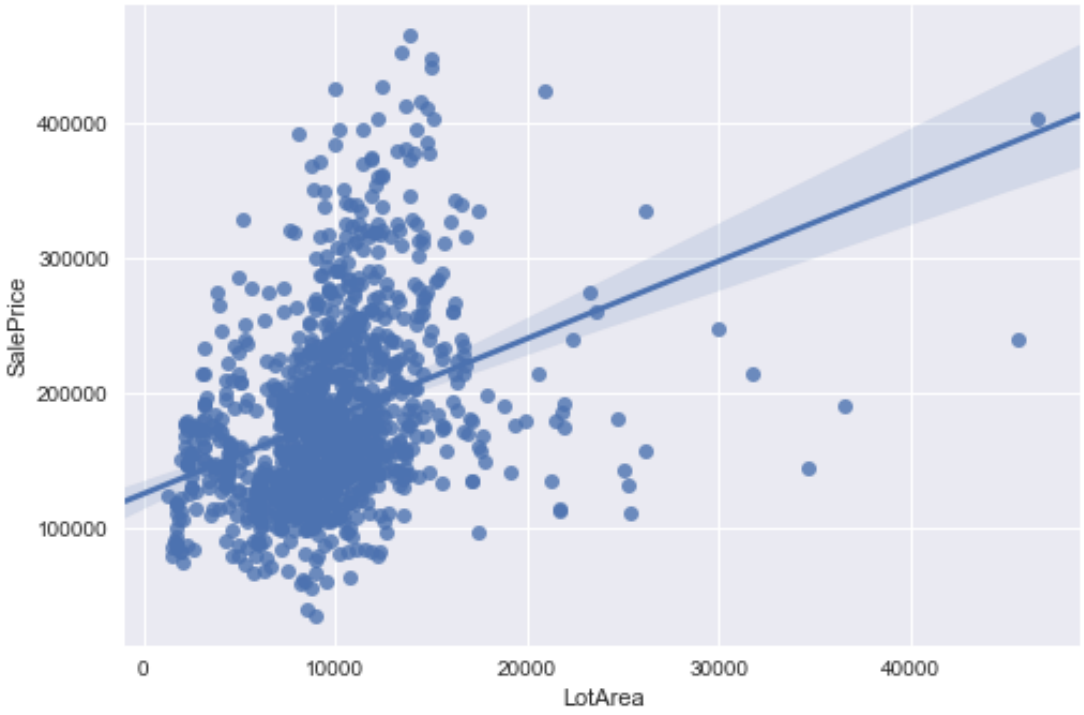
As we will summarize in this report, we can assert with confidence that with the right set of feature values available, one can predict the target variable which is SalePrice of houses. Regression analysis is a way of sorthing which variables have an impact of the independent variable. It is a good fit for the problem as we are trying to predict a continuous independent variable (SalePrice of houses) from independent variables such as lot size, number of rooms etc.

While there were a number of features or data points considered in the predictive experiment, the following features had a direct correlation when calculating the Housing Prices:

- OverallQual,
- GrLivArea,
- TotalBsmtSF,
- 2ndFlrSF,
- BsmtFinSF1,
- 1stFlrSF,
- GarageCars,
- GarageArea
- YearBuilt
- MasVnrArea

We can visualize the impact of some of these variables on SalePrice:





The machine learning process to determine predictive values followed a set of key processes or phases:

Data Exploration: In this phase the data relationships were identified using statistical methods as well as visualization techniques.

Feature Selection: During feature selection various techniques were used to pick the best features for model training, this requires a fine balance between over engineering a model and thereby overfitting it and selecting the right features that lead to better accuracy for predicted values.

Data Transformation: This involves cleaning up the datasets by removing duplicates, outliers, null values and other data cleaning routines to ensure that the model is being trained on suitable dataset.

Model Development: During model development, various algorithms are used to check for best performing estimator and other techniques are utilized to further tune the hyperparameters being used while training the model.

Model Evaluation: Finally the trained model is tested with a separate test dataset for validating its results and to measure the model's accuracy.

Metrics

I used r2 score and mean absolute error in my benchmark model. R2 score measures the goodness of fit of line between predicted values and actual values with 1 being the highest value. Mean_absolute_error

is a measure of difference between two continuous values which is our case were predicted and actual house prices.

I used r2 score because it is a good measure of how close the data points are to the predicted regression line. I did not use Mean Squared Error because it makes larger values of deviations from actual price even larger. I opted to use Mean_absolute_Error as it is more robust to outliers.

I used Root Mean Squared logarithmic error to get my final score as this was being used by the Kaggle competition. Taking the log does not penalize the huge differences between actual and predicted prices as in the case of House Prices which are valued at hundreds of thousands of dollars and taking the actual differences would make our results less informative.

Analysis:

Data Exploration

Numeric Data Analysis

The dataset has been downloaded from the competition based site of Kaggle.com.

"The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an incredible alternative for data scientists looking for a modernized and expanded version of the often cited Boston Housing dataset" Kaggle

The data set describes the sale of individual residential property in Ames, Iowa from 2006 to 2010. The data set contains 2930 observations and a large number of explanatory features (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. There are 81 features in the training dataset and 1460 rows. Here's a brief version of what you'll find in the data description file.

- SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

Here is a sample set of the train data file

	Id	MSSubClass	MSZoning	LotFrontage	...	YrSold	SaleType	SaleCondition	SalePrice
0	1	60	RL	65.0	...	2008	WD	Normal	208500
1	2	20	RL	80.0	...	2007	WD	Normal	181500
2	3	60	RL	68.0	...	2008	WD	Normal	223500
3	4	70	RL	60.0	...	2006	WD	Abnorml	140000
4	5	60	RL	84.0	...	2008	WD	Normal	250000

5 rows × 81 columns

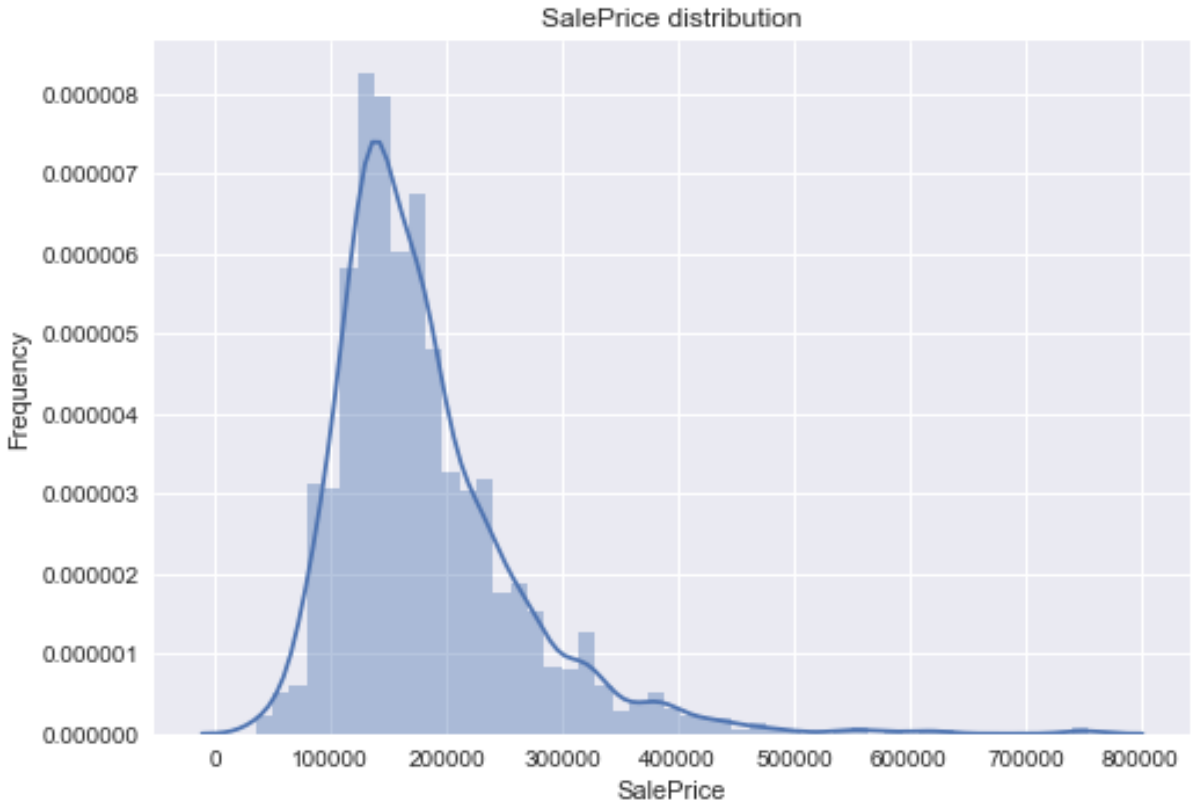
Given a set of relevant features of houses we are trying to predict the house prices. With features like the lot area and house square footage and number of bedroom, bathrooms etc. we hope to make useful predictions about the house price by making sense of the data using machine learning models and feature scaling.

Exploratory Visualization

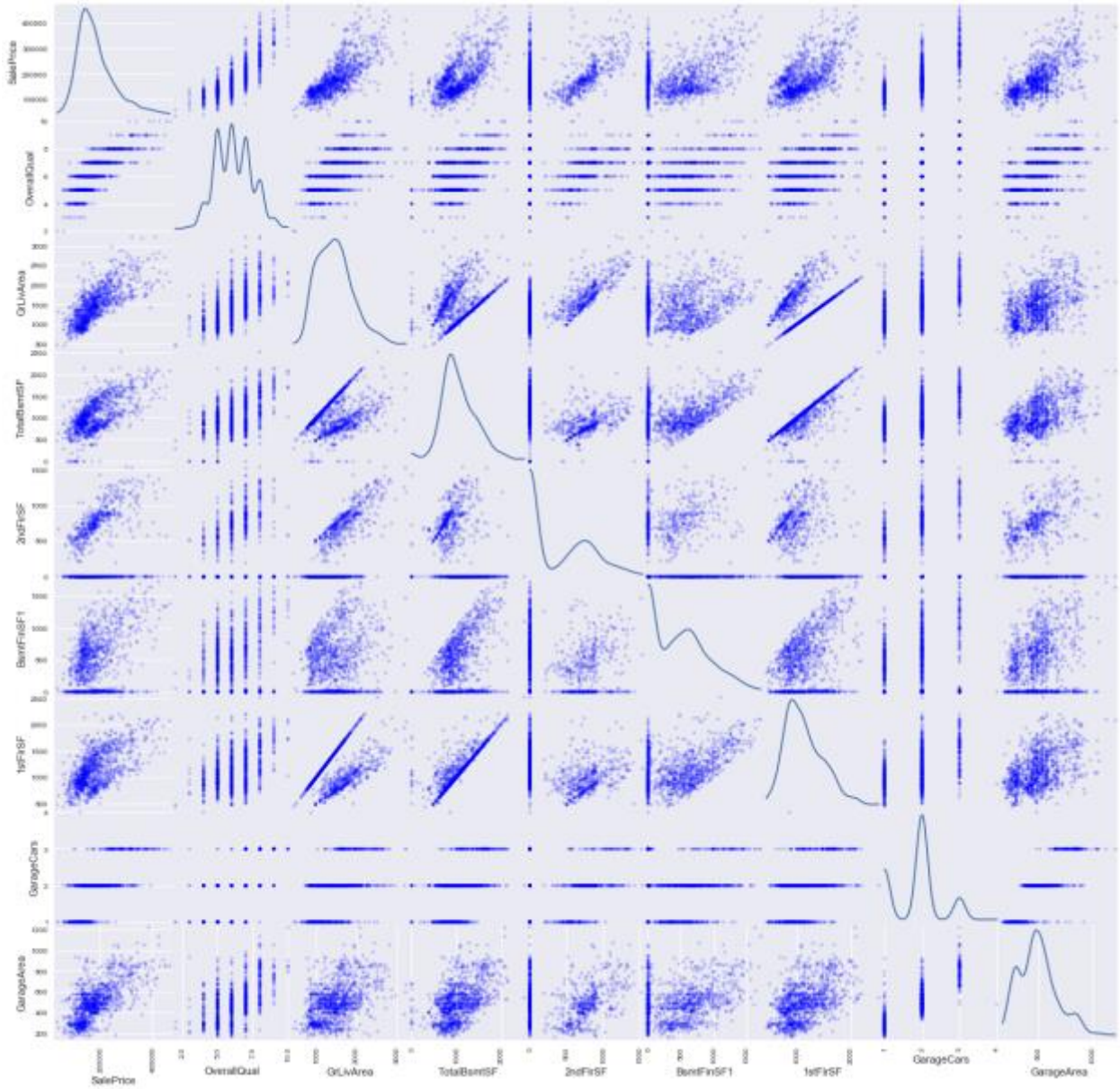
During this phase of the experiment, various techniques were utilized to understand the data, its statistics and how various features related to target variable income. To start with, we looked at summary statistics for the SalePrice to understand the shape of the dataset along with its minimum, maximum, standard deviation, median and mean.

Column	Min	Max	Mean	Median	Std	Count
SalePrice	34900	755000	180921	163000	79442	1460

From the above summary statistics on income, we can tell that the data has some higher valued outliers as the maximum is much larger in difference from the mean than the minimum value. This will be important aspect to consider during data transformation when we are dealing with removal of outliers.



From our scatterplot visulalization we can see there is a linear correlation between some of our features and SalePrice.



Algorithms And Techniques:

As we have defined this kind of problem to be a regression analysis problem, there are a few algorithms that are suited for this type of problem. In its broadest terms regression analysis is fitting a line/curve to the given data points such that the differences between the data points and fitted line/curve are minimized. One of the algorithms I used was Linear Regression which is used to determine if there exists a linear relationship between a dependent variable and one or more independent variables. The dependent variable is continuous while independent variables can be continuous or discrete. In our case we are trying to predict the value of the dependent variable SalePrice based on the features given to us. We used training data on the Linear Regression model to predict future outcomes. First we downloaded the necessary model from Skicit library and then trained our model by fitting it to the training data

clf = LinearRegression()

clf = clf.fit(X_train, y_train)

One of the advantages of using multiple linear regression is that it finds correlations between variables. For example it can find correlations between size of lot, number of rooms with the house price.

Some of the disadvantages of using Linear Regression are that it assumes a straight line relationship between dependent and independent variables and this might not always hold true. Also Linear regression looks at the mean of the dependent variable and is sensitive to outliers. For example there might be an outlier with 25 bedrooms which is a rare occasion and that will be taken into consideration when predicting the dependant variable price. However that would not be a good predictor as that is a rare example.

I used Random Forest Regressor which is an ensemble learner that uses weak learners to form a strong learner. It uses fully grown decision trees, to narrow down some of the features by using it's feature_importances_ methods. The important features are chosen based on which features reduce the impurity of samples the most to the least.

***Trees and Forests*. The random forest starts with a standard machine learning technique called a "decision tree" which, in ensemble terms, corresponds to our weak learner. In a decision tree, an input is entered at the top and as it traverses down the tree the data gets bucketed into smaller and smaller sets.**
http://blog.citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics

Random Forests are prone to low bias and high variance. The errors in this model are reduced by controlling the variance. Some of the advantages of using decision trees are that they do not assume a linear relationship between the dependent variable and independent variables and they can implicitly perform feature selection. This is done automatically when fitting a decision tree to a training dataset as it chooses the most important features in the top few nodes on which the tree is split.

For my final method I used Gradient Boosting which is also an ensemble method that learns from weak learners to form a strong learner in an iterative fashion. A simple way of explaining it is in the setting of least squares regression where the goal is to teach a model F to predict y_pred = F(x) by minimizing the least square error in the form of (y_pred-y)^2.

At each stage, the model tries to improve the predecessor algorithm where there was an imperfect model that just predicts the mean as y from the training set. Gradient Boosting provides a better model by adding an estimator h that would improve on the previous model : F_new(x) =y+ h(x) where y is actual outcome. Thus we can fit the model to find h(x) that improves the model :h(x) = F_new(x) – y.
Next we want to minimize the errors using gradient descent and we can achieve it by the following formula

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) = \arg\min_{\gamma} \sum_{i=1}^{n} (\gamma - y_i)^2 = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

The only difference between the above formula and the method we actually used in the project is that for loss function we used logarithmic error. Gradient boosting follows these iterations until stopped by the number of estimators specified or other regularizers or when the model can no longer be improved.
This model also uses decision trees but they are much shallow and more prone to bias individually than variance. The sequential combination of these trees where the next tree in line learns from the errors of the previous tree produced better results in predicting house prices than the previous methods. The reason I used Gradient Boosting for my final model was that it extracts the most important features in the top levels of the tree. The previous methods discussed for Linear Regression for fitting the model were applied to the train and test set to get final predictions. Grid Search was used to choose the best parameters that produced the best result.

Benchmark

The features given to us are the most prevailing features in predicting house prices. We can expect houses with high Overall Condition Rating , more lot area and more rooms to have higher sale prices then their counterparts. For my Benchmark model, which I have attached as a separate ipynb file by the name of benchmark_model, I have first done some data cleaning to make it easy to work it. I filled the nan values and used pd.dummies to convert categorical values to numerical ones. I also reduced the data to 271 features instead of 289. I did this by dropping some of the features that had low correlation with SalePrice.

I used Standardscaler to normalize the data and then used decisiontree model as my benchmark regression model. I used r2 score and mean_absolute_error and got scores of 28630.57 and 0.70 respectively.

Methodology :

Data Preprocessing:

Feature Selection

The first technique I used to remove some redundant features was to remove those that did not have more than one unique value as they would not provide any value to our model used later to predict price.

When doing data exploration using various visual plots, we identified a few relationships that correlate with the target SalePrice variable, some with high degree of linearity. The two additional techniques to further narrow down the list of features were creation of a correlation matrix (which does the same result as scatter plots but allows you to get those in numerical values) and using decision forest model for identifying feature importance. I also experimented with number of features and tried 25, 100, 120,130 and got best results with 130 features.

Correlation Matrix:

A correlation matrix is a table that shows correlation coefficients between various sets of features. Panda dataframes provides a built-in function to calculate this matrix which computes pairwise correlation of columns, excluding nan values.

OverallQual	0.808349
GrLivArea	0.751582
GarageCars	0.695571
GarageArea	0.669124
TotalBsmtSF	0.640912
1stFlrSF	0.618424
FullBath	0.596747
TotRmsAbvGrd	0.596193

Decision Trees for feature selection

Another technique for feature selection is to run all features through a decision tree model (after some transformations) and check the feature importance variable of the model itself, feature important determination in decision trees is based on sum of the improvements the model evaluates in all nodes in which the attribute appears as a splitter. The author used decision trees to determine feature importance and merged that knowledge with the rest of the data exploration exercise (and model testing) to determine the final set of features.

I also tried PCA, NMF and SelectKBest in a pipeline to reduce features but got better results with decision trees.

Implementation

Data transformation is a critical step if you want your model to be able to perform effectively on new datasets and to ensure that the observations being used in the various features are cleaned and represent valid range of values. Data transformation is different from feature selection because it does not change the number of features, it changes the values in them or changes the number of observations. When building the income estimator model, various data transformation steps were implemented. These include:

- Handling Nans effectively is critical to having robust variables that are later transformed by the prediction model. In this project we have used the method fill.na(0) to fill all missing values with zeros.

- PCA Analysis, NMF and SelectKBest(chi2) was also conducted on the data which is a process for reducing the dimensionality of the data, but the results did not prove very positive for the overall scoring and was later taken out of the process.

- Outlier removal is another critical step, one must remove any significant outliers to ensure that the model can generalized well with new datasets that are in nominal range. A technique of zscores was used to measure outliers and I applied the method where anything above 4 standard deviations from the mean was removed.

- Dealing with categorical data can be considered another data transformation step, although it does lead to new features and perhaps not all of those features are relevant to the model. I applied pd.getdummies to convert the categorical data.

- Finally, the issue with scale of the data was addressed (we can see from the scatter plot that most of the data is not normally distributed), the experimentation involved munging data against varies scalers and StandardScaler method was finally selected to ensure that all features are within the same scale for model training.

Model Development

Model development is about how the final “transformed” dataset is then used to train the estimator with provisions for testing and validation. Key to model development is about making sure that training data and testing data is kept separate. Given the way this competition was organized, we did not have labels for the test data as it was only known the administrators of the competition, so we had to create our own test data out of the training set available. This was done using sklearn’s model_selection library function train_test_split.

X_train, X_test, y_train, y_test = train_test_split(Data, y, test_size = 0.2, random_state=0)

Estimator Selection

Once the training and testing sets were separated, we used the training set to train various estimators (algorithms) and further explored the one that provided the best results. In the process of training estimators the experiment involved training Linear Regression, XGBoostRegressor and Random Forest Regressor. Linear Regression scores of RMSLE were the lowest when tested on the competition site with score of 0.235. XGBoost RMSLE score was better at 0.18. Eventually the experiment led to use of an ensemble method called Gradient Boosting Regressor which provided the best results of 0.14.

One thing to be noted was that although my RMSLE scores were better on my local test set (0.115) when compared to features I manually extracted after using decision tree feature importance method (0.09), the scores I obtained with decision trees on the Kaggle site was better.

I had some difficulty figuring out what number of features out of 281 features would result in best scores and I tried feature numbers between 100 to 150 and got best scores with 130 features.

```
GradientBoostingRegressor(alpha=0.95, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='huber', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_split=1e-07, min_samples_leaf=1,
              min_samples_split=50, min_weight_fraction_leaf=0.0,
             n_estimators=250, presort='auto', random_state=None,
             subsample=1.0, verbose=0, warm_start=False)
```

Tuning Hyperparameters

Gradient Boosting Regressor offers various hyperparameters that can be adjusted to validate further optimization however trying out each combination of hyperparameters one at a time can be time consuming, one method we utilized for hyper tuning parameters is called Grid Search. Grid Search allows validating against a range of hyperparameters and provides the best estimator from that range. After running Grid Search for alpha, max_depth and alpha values, the final parameters were alpha 0.1 and max_depth = 3

```
params = {'max_depth': [1,3,4,8],


          'alpha':[0.1,0.4,0.9]}
clf = GridSearchCV(model_gbr, params, cv=4 )


clf = clf.fit(X_train, y_train)


print clf.best_estimator_
```

I tried my model with GridSearch and without it. My Root Mean Square Log Error score with tuning hyperparameters using Grid Search were slightly better than the ones without Grid Search (0.14369, 0.14474 respectively) on the Kaggle board.

Transforming Test Data

One critical step that is often neglected and this case in particular where the test data is provided separately from the training data, that step is about making sure that all the transformations that have

been done to training data prior to model training (or fitting the model against the training data), the same transformations are completed on the test data set. This includes any feature adjustments like dealing with categorical data through one-hot encoding and scaling it against the same Scaler that was used to transform the training dataset. In this case, we did not filter the data based on outliers as that would have removed certain rows making it an invalid result set for the competition (as number of observations would change).

Refinement

I tried several different methods to improve my model by using feature scaling, tuning hyper parameters with Grid Search and trying out different feature reduction models like decision trees, PCA, SelectKBest etc. Removing outliers was also used. Other improvements can be made by visualizing all the features relationships with each other and removing some features that are very closely correlated because they don't provide much information and can result in overfitting. However overfitting problem is addressed by using ensemble method which use average of trees that will be overfitting and underfitting so in the end the result is reduced overfitting. I have also reduced features from 281 to 130 manually by trying different number of "important features" derived from decision trees. Furthermore we have used parameters like "max_depth" to address the problem of overfitting.

Results

Model Evaluation

The final model chosen was as mentioned above Gradient Boosting which uses sequential ensemble learning from shallow trees. Gradient Boosting has become quite popular for solving regression tasks, sometimes second only to XGBoost. This regression model works well with our data as each separate tree is basically a weak learner (that is required to do just better than random guessing) which become better and better sequentially as it learns from its past mistakes. The higher the number of estimators (number of trees) the better it is supposed to perform. I used 250 estimators which gave us a good final RMSLE score of 0.14006. The parameters chosen were max_depth=3, learning_rate = 0.1 and alpha = 0.95. Also min_sample_split parameter which controls overfitting was set at 50 for best score.

Generally this model is robust to overfitting so we would expect a lower max_depth number to achieve a better result as higher depth would make the model prone to overfitting. Similarly a lower learning rate would make the model robust to the specific characteristics of the tree and thus allow it to generalize better.

For model evaluation, scoring method selected by the competition administrators is Root Mean Square Logarithmic Error (RMLSE) which is calculated by taking square root of the mean square error of the logs of predicted and actual values . RMSLE is usually used when you don't want to penalize huge differences in the predicted and true values when both predicted and true values are huge numbers. Since the House prices were in hundreds of thousands of dollars it made sense to use this method. My score for the test data was 0.095

The model test RMSLE score was 0.0994 compared to a score of 0.14006 on the competition site which shows that there was some overfitting happening as I got a better result locally and it the model did not generalize that well on unknown data.

In Grid Search I tried cross validation folds equal to 4 and 3 (the default). The RMSLE score was 0.0994 with my local test split data and 0.0954 with the default value. However the score on the competition site was better for cv=4 (0.14006 compared to 0.14435).

Final RMLSE for the competition 0.14006

Justification

When comparing our simple benchmark model which was built on a simple decision tree regression model with regularization in the form of max_depth = 4 , we have done a good job of improving our model by implementing Grid Search to choose the best parameters from a given set and using GradientBoosting Regression model that uses an ensemble of decision trees. We regularized our final model with max_depth, alpha, learning_rate and min_samples_split in order to be more proactive in dealing with the issue of overfitting. We also used feature_importances_ method to extract and reduce the number of features available to us in order to tackle overfitting.

I believe we have solved the problem as our final model RMSLE score is 0.0994 from my test split data compared to benchmark of 0.199. My score on the competition Kaggle site also improved from 0.24 to 0.14 and we can safely say that we solved the problem of predicting house prices with reasonable accuracy.

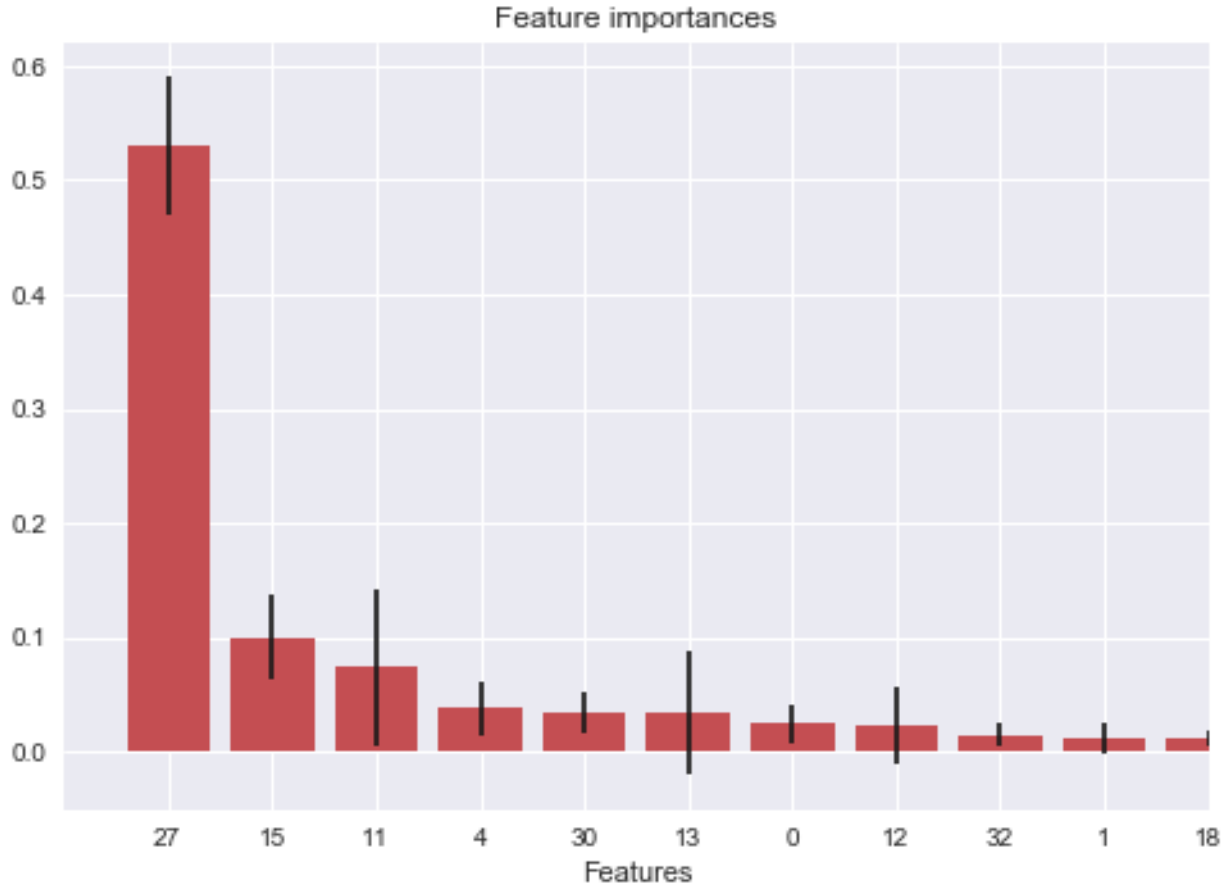
Conclusion

Free Form Visualization

We can visualize from the chart below some of the rankings of features in the order of their importances from left to right

Feature ranking:
1: 27 - 27 - 0.529545811365
2: 15 - 15 - 0.0998439617286
3: 11 - 11 - 0.073133077872
4: 4 - 4 - 0.0371768034241
5: 30 - 30 - 0.0333756173273
6: 13 - 13 - 0.0333141192137
7: 0 - 0 - 0.0243322884028
8: 12 - 12 - 0.0223990219187
9: 32 - 32 - 0.0145827411094
10: 1 - 1 - 0.0110120942019
11: 18 - 18 - 0.010544594244
12: 68 - 68 - 0.00726129765755
13: 33 - 33 - 0.00650820454557
14: 26 - 26 - 0.00544879946257
15: 8 - 8 - 0.00541184432357

16: 25 - 25 - 0.00499330190874
17: 22 - 22 - 0.00436915347722
18: 19 - 19 - 0.00413222400671



From this we could conclude that overall Quality of the house is the single most important feature of house prices with value of 0.52 while all the rest were below 0.099. This seems to apply that we can get house prices mainly based on the overall condition. While Overall Quality is an important feature we cannot ignore others that are almost as important as we can see from our correlation matrix

SalePrice 1.000000
OverallQual 0.808349
GrLivArea 0.751582
GarageCars 0.695571
GarageArea 0.669124
TotalBsmtSF 0.640912
1stFlrSF 0.618424
FullBath 0.596747
TotRmsAbvGrd 0.596193
YearBuilt 0.590412
Foundation_PConc 0.548230
YearRemodAdd 0.543443
GarageYrBlt 0.542859
BsmtQual_Ex 0.538249

```
ExterQual_Gd            0.524756
KitchenQual_Ex          0.482427
BsmtFinType1_GLQ        0.467832
HeatingQC_Ex            0.454975
MasVnrArea              0.449504
Fireplaces              0.429834
Neighborhood_NridgHt    0.423664
OpenPorchSF             0.407691
GarageFinish_Fin        0.404695
BsmtFinSF1              0.377308
ExterQual_Ex            0.376956
Exterior1st_VinylSd     0.373483
Exterior2nd_VinylSd     0.371399
SaleType_New            0.359831
KitchenQual_Gd          0.354577
SaleCondition_Partial   0.353077
LotArea                 0.349494
MasVnrType_Stone        0.341781
WoodDeckSF              0.331011
FireplaceQu_Gd          0.324611
GarageType_Attchd       0.323511
2ndFlrSF                0.299174
BsmtQual_Gd             0.289709
Neighborhood_NoRidge    0.283436
HalfBath                0.281374
BsmtExposure_Gd         0.256982
HouseStyle_2Story       0.256625
```

We can see that GR_Living_Area, Garage_cars, Garage_Area, TotalBsmtSF and others are almost as important as OverAll_Qual as they have strong correlation values with SalePrice. These results are not much different from our intuition of what factors play a part in determining house prices. We would expect Living_area, overall quality, number of garages , year house was built etc. to have an impact on house price.

Reflection

The conclusion from this experiment was that one can successfully estimate or predict the price of homes if selected features are available to do the prediction. I have tried different techniques in this project to come to this conclusion. In my quest to fit the training data in a good model, I transformed categorical data to numerical and normalized skewed data. In trying to reduce features to avoid the problem of overfitting I tried several feature reduction models such as PCA but was surprised to find that the model scored best results when I selected the number of features from the feature_importances_ method from decision tree model.

I had to try different numbers of those important features and settled on 130 features which were giving better scores. I used grid search to help pick the best parameters for my model including max_depth and used GradientBoostingRegressor which is an ensemble learning method and predicts by combining outputs from individual trees. This model takes longer time to train than Random Forests because they build one tree at a time and the next tree helps to correct errors made by previously trained tree. They are however considered better learners than Random Forests from benchmark results.

I was able to improve my competition score from 0.24 to 0.14 by feature scaling and feature reduction and using GBRT model.

One of the things I was not expecting and had to work on was the fact that feature_importance_ was not an absolute measure of the variables that had an impact on house prices. I learnt we should not just rely on one method to reduce features but try out at least 2 or 3 methods and experiment by error and trial o n the final number of features that produce the best results. With gradient boosting I found out that havi ng more features than less resulted in better RMSLE score. However, the improvements in score lasted till approx. 130 features and then tapered off.

Improvement

I believe there is always room for improvement when working with machine learning algorithms. One area of improvement could be to treat all features(columns) with Nan values separately and replace Nans with mode, median or zeros depending on which is most suitable statistically. For example, GarageCars with Nan makes sense to use "none" or zero value but for LotFrontage Nans we could have used mode of the neighborhood (most occurring value for that neighborhood). These improvements would be potentially better as they would represent the data better and thus would generalize better.

Another area for improvement could be using Box Cox Transformation to transform highly skewed values instead of Standard Scaler. This kind of treatment is found to be slightly better with highly skewed data as in our case.

For our final model specifically, we could try Stochastic gradient boosting. In this method, instead of using the full training sample we use a sub-sample of it.

"at each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner".

— Stochastic Gradient Boosting [PDF], 1999

The subsample could be in terms of rows or columns of the training dataset:

"Generally, aggressive sub-sampling such as selecting only 50% of the data has shown to be beneficial."

https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Research results indicate that column subsampling is more effective in resolving overfitting issues when compared with row sub-sampling.