Machine Learning Engineer Nanodegree

Capstone Project

János Tamási

October 7th, 2018

I. Definition

Predicting Ames housing prices with the help of supervised learning algorithms and comparing the results of different approaches in data preparation, data selection and model architecture

https://www.kaggle.com/c/home-data-for-ml-course#description

Project Overview

Predicting housing prices based on the individual house's attributes is one of the classic examples of supervised learning tasks. It is a relevant task to solve because it has great practical and financial importance to be able to determine the real value of a house as precisly as possible based on its characteristics. The target variable is the price of the house in dollars, which we want to predict based on the predictors, which can be any characteristic of the property, both numerical and categorical. The task and the dataset is part of an ongoing [Kaggle competition](https://www.kaggle.com/c/home-data-for-ml-course#description) which has already more than 700 competitor. My goal is to try out as many possibilities in the data preparation and the model building process as possible to finally reach a model with the best possible predictive capability to rank as high on the competition's leaderboard as possible.

In the history of supervised learning it all started with linear regression and logistic regression, then decision tree algorithms became widespread. A more efficient version of the decision tree algorithm is the random forest model which is an ensemble type of model which means it incorporates many different decision trees in one single model to find a more efficient one that performs better than all of the elements. Lately an even more efficient version of the random forest algorithm was developed which is called gradient boosted decision trees and it iteratively adds new decision trees to the ensemble model to minimize the prediction errors. There are many other algorithms to use in supervised learning like support vector machines, but I plan to investigate and compare to each other the decision tree related algorithms only. Finally here is a link to an academic paper where the authors applied machine learning algorithms for housing price prediction: Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data

Another famous dataset that provides similar information about houses and their prices like the one we will use is the Boston Housing Prices dataset, but the Ames dataset has three times more data points and about five times more characteristics of the houses, so it provides much betters conditions for building a precise model.

Problem Statement

The problem that I chose to solve is to determine the value of a house in Ames, Iowa in a certain historical time based on 79 explanatory variables of the houses and the data of the previously sold houses and their characteristics in the area. The problem is not 100% objective because the value of a house can vary among individuals based on their subjective preferences, but at least can be approximated via actual transactions that took place in the past in the real estate market in the area. I plan to use supervised machine learning algorithms, namely decision tree related algorithms after proper data preparation. The predictive capability of the built model can be measured via mean absolue error values on a separated test data set and our predictions on those instances. To learn how good is our solution in the field we can compare our results with the results of the other competitors in the Kaggle competition.

Metrics

The mean absolute error is calculated via taking the mean of the absolute values of the differences between the predictions and the actual home prices of the certain dwelling. By minimizing this value we can assure that our predictions are more and more precise and closer to the actual values of the buildings. The reason why I chose this metric over other possibilities like RMSE or R^2 is because I think the idea of mean absolute error is the easiest to grasp among the alternatives and it was chosen as the main metric during the kaggle course on the subject.

II. Analysis

Data Exploration

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. I acquired the dataset free of charge in the context of the kaggle competition that I specified previously. The dataset is divided into training and testing datasets. The training dataset contains 1460 datapoints with 80 predictor variables (from which 37 is numerical and 43 is categorical), and a target variable (sale price). The testing dataset contains 1459 datapoints with the same predictor variables but without target variable data - this is used in the competition to check predictive performance of the competitors' models.

Here is an example data point, omitting some of the variables:

```
Id                        1
MSSubClass               60
MSZoning                 RL
LotFrontage              65
LotArea                8450
Street                 Pave
LotShape                Reg
LandContour             Lvl
Utilities            AllPub
```

```
LotConfig         Inside
LandSlope            Gtl
Neighborhood     CollgCr
Condition1          Norm
Condition2          Norm
BldgType            1Fam
HouseStyle        2Story
OverallQual            7
OverallCond            5
YearBuilt           2003
YearRemodAdd        2003
RoofStyle          Gable
RoofMatl         CompShg
Exterior1st      VinylSd
Exterior2nd      VinylSd
MasVnrType       BrkFace
MasVnrArea           196
ExterQual             Gd
ExterCond             TA
Foundation         PConc
BsmtQual              Gd
                  ...
BsmtFullBath           1
BsmtHalfBath           0
FullBath               2
HalfBath               1
BedroomAbvGr           3
KitchenAbvGr           1
KitchenQual           Gd
TotRmsAbvGrd           8
Functional           Typ
Fireplaces             0
GarageType        Attchd
GarageYrBlt         2003
GarageFinish         RFn
GarageCars             2
GarageArea           548
GarageQual            TA
GarageCond            TA
PavedDrive             Y
WoodDeckSF             0
OpenPorchSF           61
EnclosedPorch          0
3SsnPorch              0
ScreenPorch            0
PoolArea               0
MiscVal                0
MoSold                 2
YrSold              2008
SaleType              WD
SaleCondition     Normal
SalePrice         208500
Name: 0, Length: 76, dtype: object
```

During my analysis I will examine which variables to use, how to translate categorical data to numerical ones and how to deal with missing data by imputation or omission of the incomplete datapoints.

With the help of the .describe() pandas method we can examine some really important statistical aspects of all of the numerical data in our dataset, such as the count number (from which the number of missing values can be inferred), mean value, standard deviation, min, max, median and interquartile values.

We can find out two important aspects of the dataset from this analysis: the houses were built in the time period of 1872-2010, and all of the houses were sold between 2006 and 2010. These informations provide us the fact that our analysis is valid only around the year of 2010 and if we'd like to apply our model to today's houses in the area we should increment the prices with the inflation since then.

The range of the housing prices span between \$34 900 and \$755 000, with a standard deviation of \$180 921, to which we will be able to compare the deviations of our predictions from the actual values, thus defining how significant our prediction errors are.

The test data has the same time intervals for when the houses were built and sold.

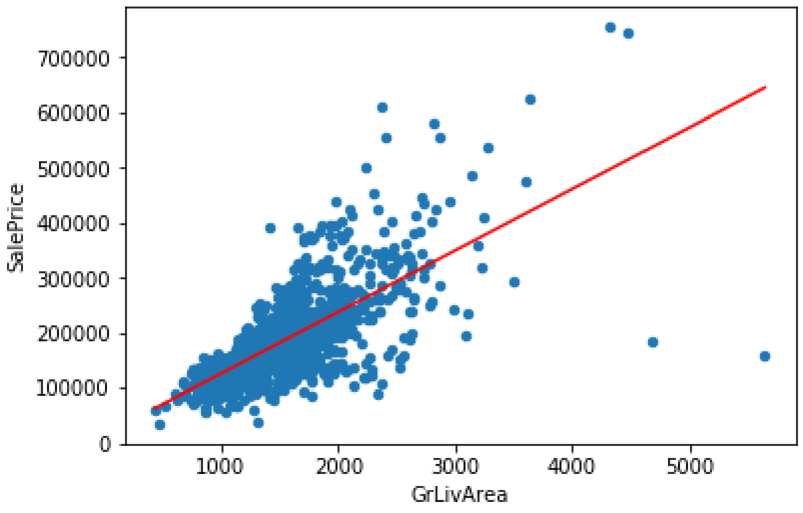
In the training data among the categorical variables there are 15 in which there is missing value, and among these there are 5 in which threre are more than a 500 missing value: Alley, PoolQC, FireplaceQu, Fence and MiscFeature.

The same analysis for test data gave 22 categorical variable columns with missing data. Among these, for the same 5 variable exceeds the number of missing values 500 as in the training data - these should be excluded from the analysis during data preparation.

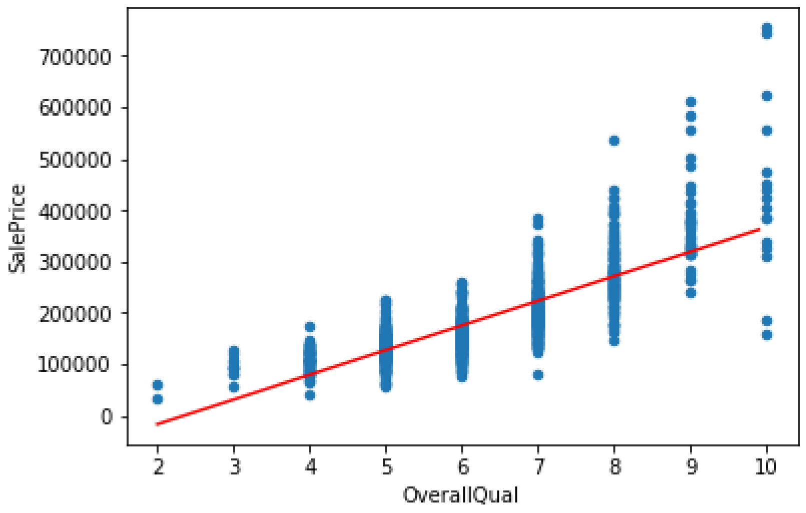
After doing the same analysis with the numerical data it is clear that there are much less missing values so non of the variables need to be excluded from further use on this basis.

Exploratory Visualization

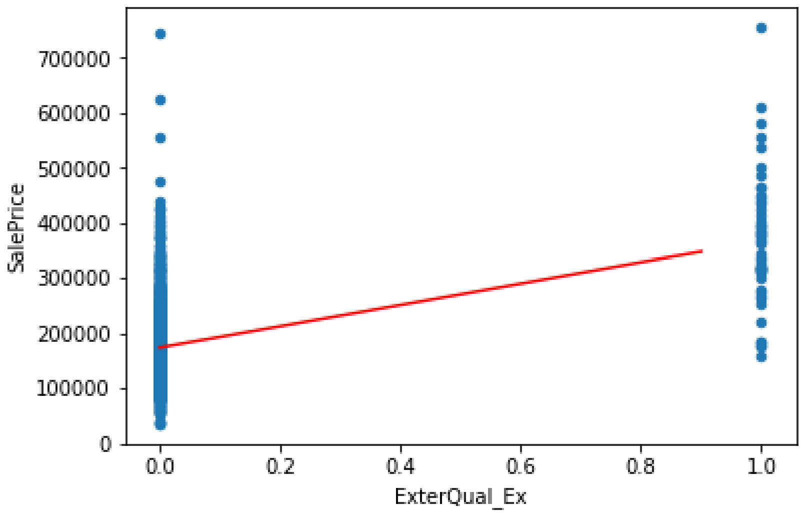
The following three visualizations will all show nice correlations between the chosen variable and the sale price. The difference between them is that the first one is a continuous numerical variable, 'GrLivArea', which defines the above grade (ground) living area square feet, the second is a discrete numerical variable, 'OverallQual', which defines the overall material and finish quality, while the third is a certain type of one-hot encoded categorical variable, ExterQual_Ex which defines the presence or the absence of excellent exterior material quality.



SalePrice
GrLivArea
0
100000
200000
300000
400000
500000
600000
700000
1000
2000
3000
4000
5000



SalePrice
OverallQual
0
100000
200000
300000
400000
500000
600000
700000
2
3
4
5
6
7
8
9
10



SalePrice
ExterQual_Ex
0
100000
200000
300000
400000
500000
600000
700000
0.0
0.2
0.4
0.6
0.8
1.0

Algorithms and Techniques

I chose to use decision tree related algorithms, namely simple decision tree, random forest, and xgboost regression models.

A decision tree model splits the datapoints during training based on different categorizing questions regarding the datapoints' characteristics and calculates the mean value for the target variable of the datapoints that falls into the certain category. During prediction the same questions serve for categorizing the datapoint which was learned during training and the model attributes the calculated mean value for the target variable that represents the leaf node (a leaf node is a group of datapoints that fall in the same category based on the questions).

Advantages of the decision tree model that it is easy to understand and interpret, can handle both numerical and categorical data and is able to handle large datasets (so it is scalable). Limitations of the model are the tendency to not being robust, the problem of finding the optimal decision tree is an np-complete problem (resulting in the need for using heuristics) and tendency to create over-complex models which overfit the data.

The random forest algorithm is an ensemble method which incorporates many simple decision trees at training time and outputting the mean prediction of the individual trees.

Its advantage is it can solve many limitations of the simple decision tree algorithm, but in turn it's a little bit less easy to interpret and harder to track down its functioning (so it's more like a black box).

The gradient boosting decision tree algorithm is also an ensemble method which incorporates decision trees but it is built iteratively to add a new decision tree to the ensemble in every cycle to minimize prediction errors. The optimal number of estimaters and the learning rate has to be found. Xgboost is a higly optimized implementation of this kind of algorithm.

Benchmark

A benchmark model could be a really simple one, that many people use in everyday practice when considering to buy a home, namely that how big is the home in square feets - this is basically a linear regression problem with only one predictor. But to have a benchmark model that is more difficult to supersede, I will use a linear regression model with the same predictors that I will use for the final xgboost model. A linear regression model can be implemented easily with the help of sklearn and I will compare my final results to this baseline model. The metric that I will use for comparison is the mean absolute error of the models.

III. Methodology

Data Preprocessing

- First I will read in the csv files with pandas' read_csv function.

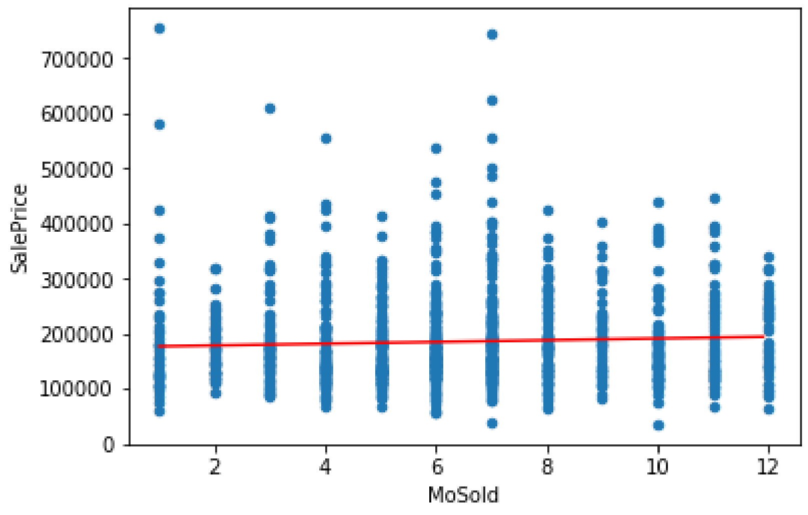
- Then I will separate the numeric data from the categorical data with the help of pandas' select_dtype function excluding object type to get the numerical data.

- Then I will examine the two datasets for missing values: I will use numpy's isnan function for numeric data and pandas isnull function for categorical data. After finding those columns which contain too many missing values (in a dataset with around 1500 datapoints I will drop every column with more than 500 missing values) I will drop them completely from both training and testing datasets.

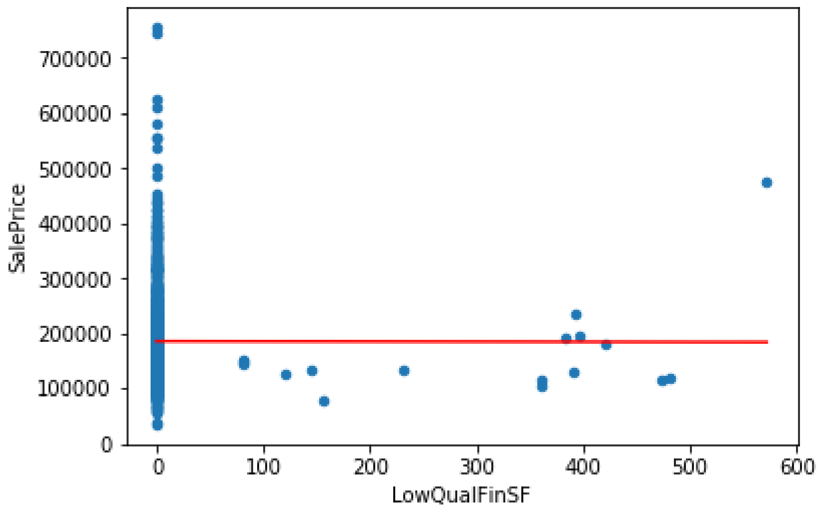
- Then I will make scatterplots (with .plot(kind='scatter')) from all the numerical data (after dropping the null values with .dropna()) having the chosen variable on the x axis and sale price on the y axis and I will also display the linear relationship that is fitted on the points with linear regression (I will use numpy's polyfit function with degree of 1 for this). I will examine all the scattterplots one by one and decide which features to exclude from further analysis because there is no apparent correlation between the variable and sale price or the correlation is negligible. I will also look for outliers.

Based on the following visual analysis (I made scatter plots and regression lines of the variables) I identified 12 numerical variable that I will exclude from further usage because they don't show strong enough or any linear correlation with sale price: Id, MSSubClass, OverallCond, BsmtFinSF2, LowQualFinSf, BsmtHalfBath, KitchenAbvGr, EnclosedPorch, 3SsnPorch, MiscVal, MoSold, YrSold.

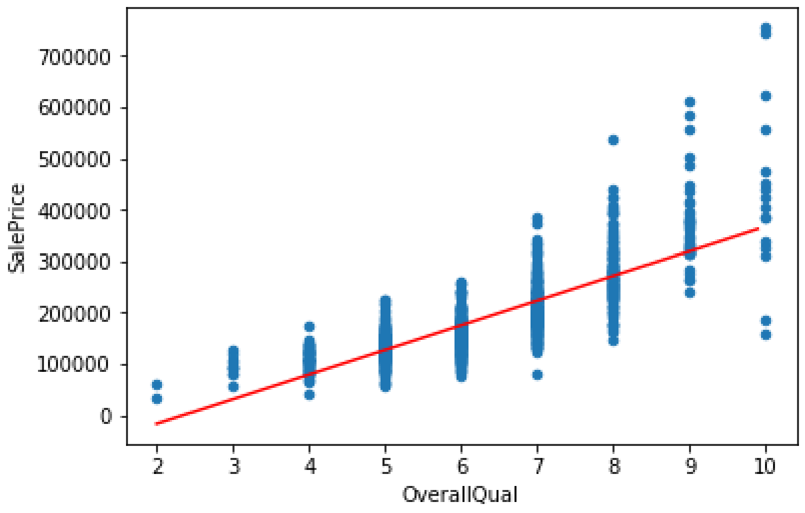
Example for no correlation with sale price (in which month the house was sold):



Example for negiligible correlation with sale price (Low quality finished square feet):



Example of strong correlation (Overall material and finish quality):



- Then for the categorical data, I will one-hot encode the dataset with pandas' .get_dummies() function. After having the one-hot encoded dataset I will make scatterplots with linear regression lines the exact same way as described in the previous point, but this time during the iteration through the columns I will also collect in a list the following four values in a tuple:

- the slope value (w1) of the linear regression,

- the number of datapoints that represent a True

- the number of datapoints that represent a False value at the certain one-hot encoded variable

- and the name of the certain variable.

- After having this list with the four valued tuples, my basic idea was to filter out those variables which doesn't include at least 50 values both with True and False values, because I didn't think the relationship would be robust enough to include them in the pridicting model. I also filtered out variables which one's slope didn't exceed $20 000 because I wanted to include only those variable which have large enough impact on sale prices. But then I decided that I will create 3 datasets and I will test each of them on the models that I built.

The first dataset contains those one-hot encoded variables from the total of 235, which fulfill the following criteria (these criteria were chosen subjectively):

- both groups' (which holds a True value and which holds a False value for the certain one-hot encoded variable) number of members exceeds 50 - this is important to choose robust enough variables only.

- the absolute value of the w1 score (the slope of the linear relationship) exceeds \$20 000 - this is important to only choose those variables that have large enough impact

The second dataset's variables fulfill only the first criterion.

The third dataset contains all the one-hot encoded variables.

This way the first dataset will contain 82, the second 101 and the third 235 one-hot encoded variables.

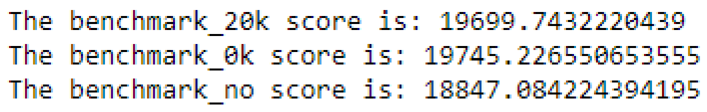
Imputation will happen as part of the pipelines that I created to avoid errors and conflicts. I used mean values to fill in nan values in numerical variables.

Implementation

-After this point we have all the datasets in the form how we want to use in our models.

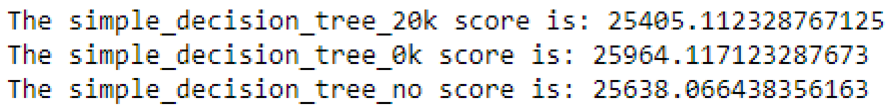
First we split our preprocessed training data into training and validating sets. It will be needed when we will search for optimal estimator number at the xgboost model. Otherwise I will compare the performance of the various models based on the mean of cross validation scores where mean_absolute_error is the metric function.

Then we create the benchmark model which is a multivariate linear regression model (sklearn’s version v0.20.0) with only default parameters and all the various predictor sets that we defined during the data preparation process.


```
The benchmark_20k score is: 19699.7432220439
The benchmark_0k score is: 19745.226550653555
The benchmark_no score is: 18847.084224394195
```

It is apparent that the model with all the one-hot encoded variables is the best performing so I will choose this as a benchmark model. So the mean_absolute_error score that I will try to beat with different decision tree related models is **18847**.

Then we create the simple decision tree model without spcifying any parameter.


```
The simple_decision_tree_20k score is: 25405.112328767125
The simple_decision_tree_0k score is: 25964.117123287673
The simple_decision_tree_no score is: 25638.066438356163
```

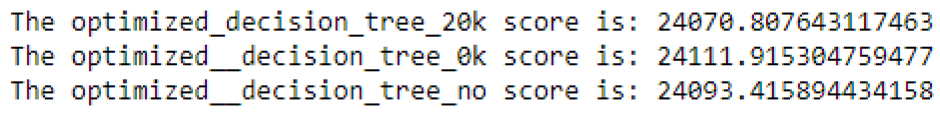
Interestingly the multivariate linear regression model perform far better than the simple decision tree model and another interesting aspect is that categorical predictor selection doesn't improve the results of the decision tree model.

Let's optimize some parameters of the decision tree model with GridSearchCV examining the following values:

'max_leaf_nodes': (10, 50, 100, 150, 200), 'min_samples_leaf': (1, 5, 10, 15, 20)

The best parameters of the search are: 100 for max_leaf_nodes and 10 for min_samples_leaf.

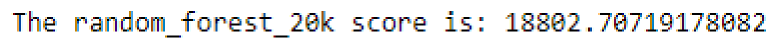
Let's see how the optimized tree model performs on the test set.



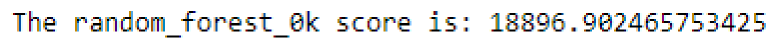
```
The optimized_decision_tree_20k score is: 24070.807643117463
The optimized__decision_tree_0k score is: 24111.915304759477
The optimized__decision_tree_no score is: 24093.415894434158
```

The score improved with about a 1000 points but this is still much worse than the benchmark score.

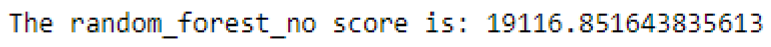
Let's examine what happens if we apply a random forest model without any parameter tuning.



```
The random_forest_20k score is: 18802.70719178082
```



```
The random_forest_0k score is: 18896.902465753425
```



```
The random_forest_no score is: 19116.851643835613
```

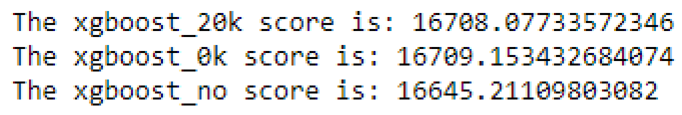
The random forest model performs a little bit better than the benchmark model. It is interesting to see that the categorical predictor selection that I used has beneficial effect on the predictive performance.

Let's examine what happens if we use xgboost model. I will use GridSearchCV to find optimal parameters from the following possibilities:

'n_estimators': (200, 250, 300, 350), 'learning_rate': (0.01, 0.03, 0.05, 0.07, 0.1)

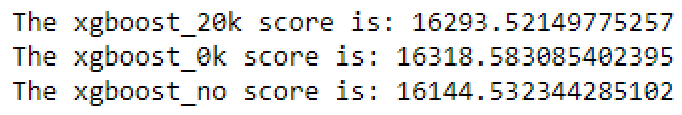
The best parameters are 300 for n_estimators and 0.05 for learning rate.

Let’s exmine the results of an xgboost model without any parameter tuning.



```
The xgboost_20k score is: 16708.07733572346
The xgboost_0k score is: 16709.153432684074
The xgboost_no score is: 16645.21109803082
```

Let's examine the results of an xgboost model with the found best parameters.



```
The xgboost_20k score is: 16293.52149775257
The xgboost_0k score is: 16318.583085402395
The xgboost_no score is: 16144.532344285102
```

Apparently the xgboost model is significantly better than the random forest model and the benchmark model. Parameter tuning helps around 500 points of improvement on average. Categorical predictor variable selection doesn't have significant effect on predictive performance.

There were no insurmountable difficulties during the coding process.

Refinement

In the beginning I tried a completely different categorical model selection technique in which I tested categorical variables one by one (not the one-hot encoded variables but the original ones) and I chose those variables to include in the model which improved the prediction score and omitted those which actually worsened it.

Previously I used train_test_split to divide the training data into training and testing set but later I realized that this approach results unreliable performance scores, so I started to use cross_val_scores instead with 5-fold partitioning which was far better. Regarding the fact that the dataset is relatively small cross validation has big beneficial effect.

Initially I didn't use GridSearchCV to find optimal parameters. Then I used grid search for find the best max_leaf_nodes (from 10, 50, 100, 150, 200) and min_samples_leaf (from 1, 5, 10, 15, 20) values. The best values were 100 for max_leaf_nodes and 10 for min_samples_leaf. Then I used grid search for finding the best number of estimators (from 200, 250, 300, 350) and learnig rate (0.01, 0.03, 0.05, 0.07, 0.1) for xgboost model. The best number of estimators were 300 and the best learning rate was 0.05.

The use of pipelines made everything much cleaner and concise - it helped the most for applying imputation because it can mess up many things if it isn't applied in the right order.

I tried the deletion of rows with missing values but I realized it is better to use imputation with mean values to avoid data loss.

Initially I tried only the plus_20k dataset on my models but I assumed that the inclusion of more of the categorical variables may produce better predictive performance and I decided to create three level of categorical predictor inclusion to test which model performs the best in which conditions.

IV. Results

Model Evaluation and Validation

The final model is an xgboost regression model with 300 estimators and a learning rate of 0.05. I used imputation for missing values. With this many estimators the model will surely build a reliable final model while the small value of the learning rate provide high precision which is also a desirable characteristic. I selected 25 numerical predictors and 82 one-hot encoded categorical predictors. The numerical predictors are the ones that surely follow a linear like (most of the time bigger is better) relationship with the target variable, while the selected categorical variables are the ones that are robust enough (at least 50 members in each group) and whose effect is high enough to be significant (at least 20000 absolute value for the slope of the fitted linear line).

I chose the final model after comparing 3 datasets with different predictors and 5 models from which the firs was the multivariate linear regression model that I chose as a benchmark model**.** I used 5-fold cross validation all along the selection process which makes the results mathematically robust enough to consider signifirant.

Justification

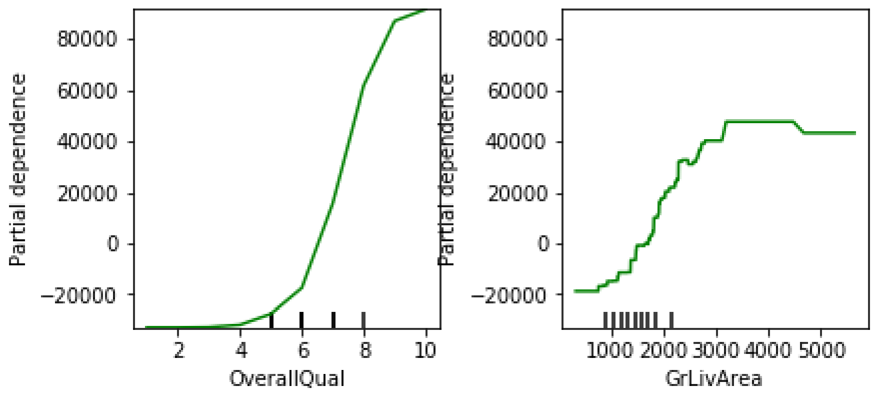
My final model produced significantly better prediction results (16293 mean absolute error points) than the benchmark model (18847 mean absolute error points) which was validated with cross validation scores and also on the test set of the kaggle competition so the results are robust and significant.

The best official score that I achieved on the real test_data during the competition with xgboost and categorical predictor selection was 15049.07939 with which I ranked 48th from 793 competitors.

V. Conclusion

Free-Form Visualization

In this section I will show some partial dependence plots that visualize the effect of the certain variable on the target variable.



From these graphs we get some useful insights about how much the investigated variable changes sale prices in different intervals. Thus we can decide more easily whether a difference between two houses' characteristics is a significant one or not.

Reflection

I think this dataset is a really well curated one with which it was quite easy to work with. The problem itself was really straightforward. After thorough data exploration the data preparation wasn't particularly difficult but definitely this part was the most time consuming. It was really interesting to see how different levels of categorical variable inclusion effected the performance of the investigated model. In summary I can say that all of the examined models were quite neutral which set of categorical variables we used.

The high performance of the benchmark model (multivariate linear regression) was impressive (18847) because it is a really basic and simple one which doesn't require any parameter tuning. It was nice to see how the different type of decision tree related models performed compared to each other and to the benchmark model. The simple decision tree algoritm didn’t perform particularly well (25405), whereas the decision tree model with optimized parameters performed a littlebit better (24070) but still far inferior to the benchmark model. The random forest model without any parameter tuning (18802) was the first that was able to beat the benchmark model. The xgboost model was the absolute winner and one of the major finding of this project that by optimizing its parameters the model’s predictive performance could be enhanced even more (the model with default parameters perfomed 16708, the one with tuned parameters 16293) . The derived results are consistent with the complexities of the certain models.

In summary we could find a highly optimized ensemble decision tree model, namely xgboost model, with tuned parameters of 300 for n_estimators and 0.05 for learning rate, which superseded the benchmark model significantly so I would say that the original aim of the project have been met.

Improvement

Further improvements could be made by trying to implement other supervised learning models which are possibly able to perform even better than the used ones or by optimizing certain parameters of the ones that I examined in this study

In general it is a good idea to apply unsupervised learning algorithms to select the most important predictor variables which I didn't leveraged in this study but it is surely worthy for further investigation.

The best score that was achieved in the competition by the time of the completion of this project was 12285 points which is significantly better than what we were able to reach so there is certainly room for improvement. Nevertheless I am happy to be in the 94th percentile.