**Machine Learning Engineer Nanodegree**

**Capstone Project**

Jiri Manak  
August 8th, 2017

**I. Definition**

**Project Overview**

A purchase of a house is usually one of the largest investments in life of most people. For a proper personal finance planning a knowledge of property price is a first step towards this goal.

The price of a house is determined by number of features. The number of square meters, number of rooms and house location are the common attributes. Anyhow, price of the house can be influenced by some specific features. For a buyer, it would be interesting to know not only an average price of the property with required attributes but also which features influence the price most. Comparing features, which have the strongest impact on the price with the personal preferences, allows buyer to make a tradeoff between the dream house and budget.

**Problem Statement**

We have a collection of data about houses which were already sold. Base on this data, can we predict how much will cost my dream house? Or a property owner may ask: „How much is my house worth? “

This problem can be solved by a predictive regression model which will be able predict the price of an unsold house . The task to do is to create such a model.

Second goal is to create list of the features sorted by its importance to influence the price of the house.

Third goal is to compare results from different regression models.

**Datasets and Inputs**

Collect data from usual sources as web portals, advertisements, real estate agencies offerings will be an enormous task. Therefore, for purpose of this project I used data publicly available on Kaggle: House Prices: Advanced Regression Techniques.

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

This collection consists of 1462 data points for training and 1461 data points for testing. Data points for testing are delivered without labels and are supposed to be submitted to the Kaggle for competition with other data analysts.

**Metrics**

To stay compatible with Kaggle metrics I will evaluate the model on RMSE.

* **Root-Mean-Squared-Error (RMSE)** - the logarithm of the predicted value and the logarithm of the observed sales price.

where is predicted value of observation *i* of regression dependable variable computed for different predictions

For evaluation and comparison of different models I will also use

* **R^2 score – coefficient of determination –** which isa proportion of the variance in the dependent variable that is predictable from the independent variable. An R2 of 0 means that the dependent variable cannot be predicted from the independent variable. An R2 of 1 means the dependent variable can be predicted without error from the independent variable.

where is predicted value of observation *i* of regression dependable variable computed for different predictions

<https://en.wikipedia.org/wiki/Coefficient_of_determination>

<http://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination>

The disadvantage of the metric based on ‘mean error’ is that the score also depends on the absolute size of predicted values. For example, if I recalculate the house prices from US dollars to Japanese yen – which is approximately 1:109 and therefore the house prices in yen will be nominally more than hundred times higher, I will get also bigger mean error score for the same model.

Advantage of R2 score is that the value is always between 0 and 1, therefore it always delivers consistent view how good is the model, which is independent from the absolute value of the predicted variable. R2 score also delivers the same results if values in data set were transformed (scaled, unskewed) without the need of inverse transformation. This is the reason why I also use R2 score metrics in model evaluation.

**II. Analysis**

**Data Exploration**

#### Features

There are almost 80 features for each record which describe each house from more aspects. Description of the features is delivered with the data in separate file. Partial information are visible on following web page:

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

#### Sale Price

‘SalePrice’ as a sale price of the house is the target variable. I will finde and train a model which will be able to predict this value based on specified features.

The basic statistic of the sale price:

*Maximum price: $755,000.00*

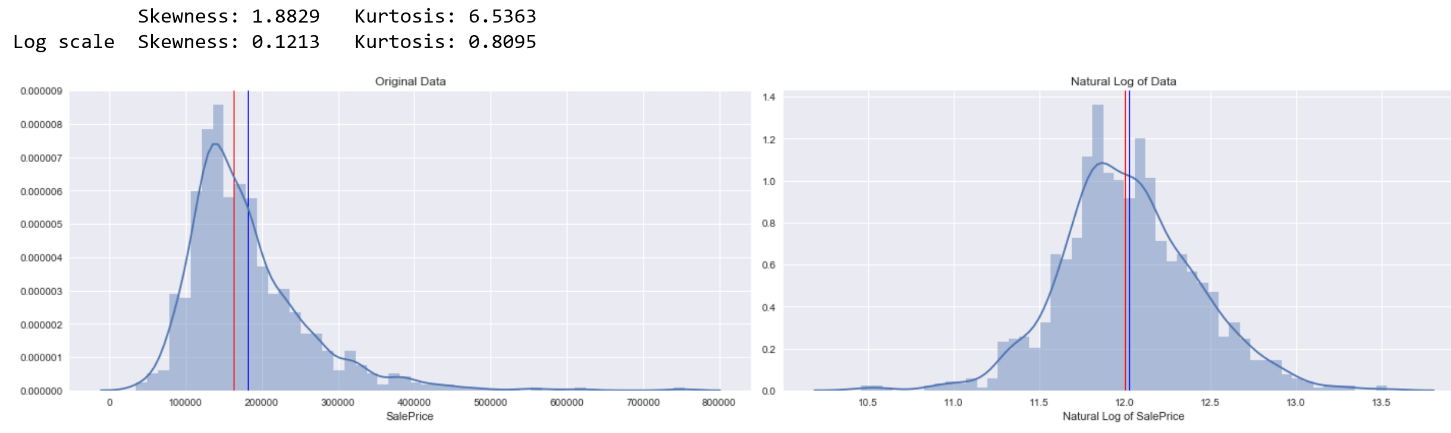
*Minimum price: $34,900.00*

*Mean price: $180,921.20*

*Median price $163,000.00*

*Standard deviation of prices: $79,415.29*

The distribution of ‘SalePrice’ deviates from normal distribution, it shows skewness to the left (towards lower prices) which is not a surprise, because maximal price of a house has no limits, but something like ‘smallest’ usable house does exist.



#### Categorical and Continuous values

Simple request on data type stated, that thera are 27 features which are numerical and 43 are categorical.

The features describing measurable attributes as for example square footage are described by continues values, or it can hold number specifying of a number of objects.

Typical continues variable features are for example:

* GrLivArea – Above grade (ground) living area square feet
* LotArea: Lot size in square feet
* BsmtFinSF1: Type 1 finished square feet
* TotalBsmtSF: Total square feet of basement area
* PoolArea: Pool area in square feet
* Fireplaces: Number of fireplaces

There is also a number of categorical values which can hold one value from specified set of values.

Typical categorical varibales are Overal Quality

* KitchenQual: Kitchen quality
* FireplaceQu: Fireplace quality
* Functional: Home functionality rating

#### Missing Data

Exploring the data by looking for missing values shows, that there is a quite a large number of missing data. Anyhow, in data description it is stated, that values are usually missing if the object doesn’t exist. For example, a pool. If property has a pool, then feature ‘PoolSF’(pool square footage) is filled out with a number of pools area in square feet. In case there is no pool, then the feature value isn’t filled out. In this case I can simply replace missing value by zero. For categorical features I can use value *‘None’*.

There are really only few features in dataset with missing values. These values were replaced by the most frequent values for particular feature in the dataset.

#### Outliers

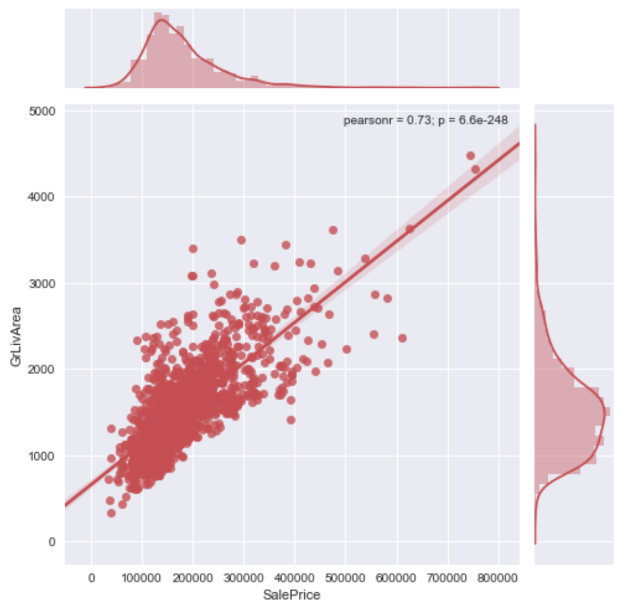
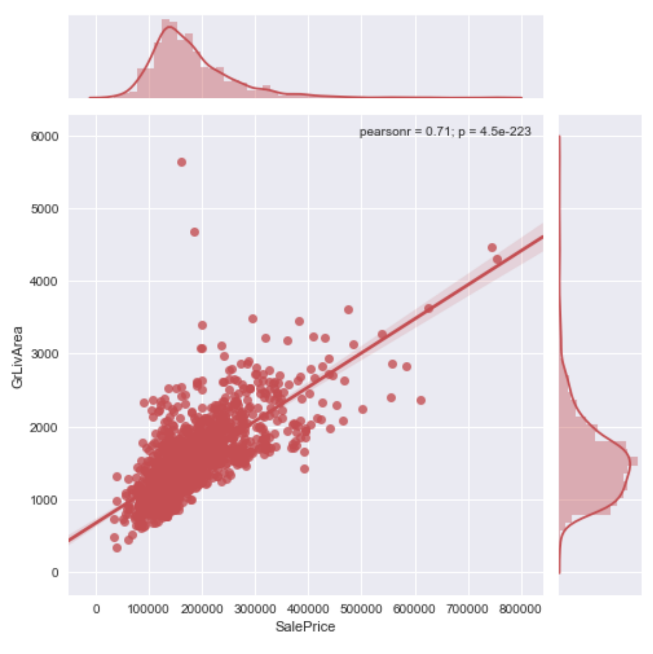
There are more possible ways how to identify outliers. Removing them might by risky because it can influence the model in negative way. Therefore, I will take a conservative approach. Correlation graph of ‘GrLiveArea’ with respect to ‘SalePrice’ shows two data points in upper left corner indicating large living area with unusual low price. I have identified these data points as outliers.

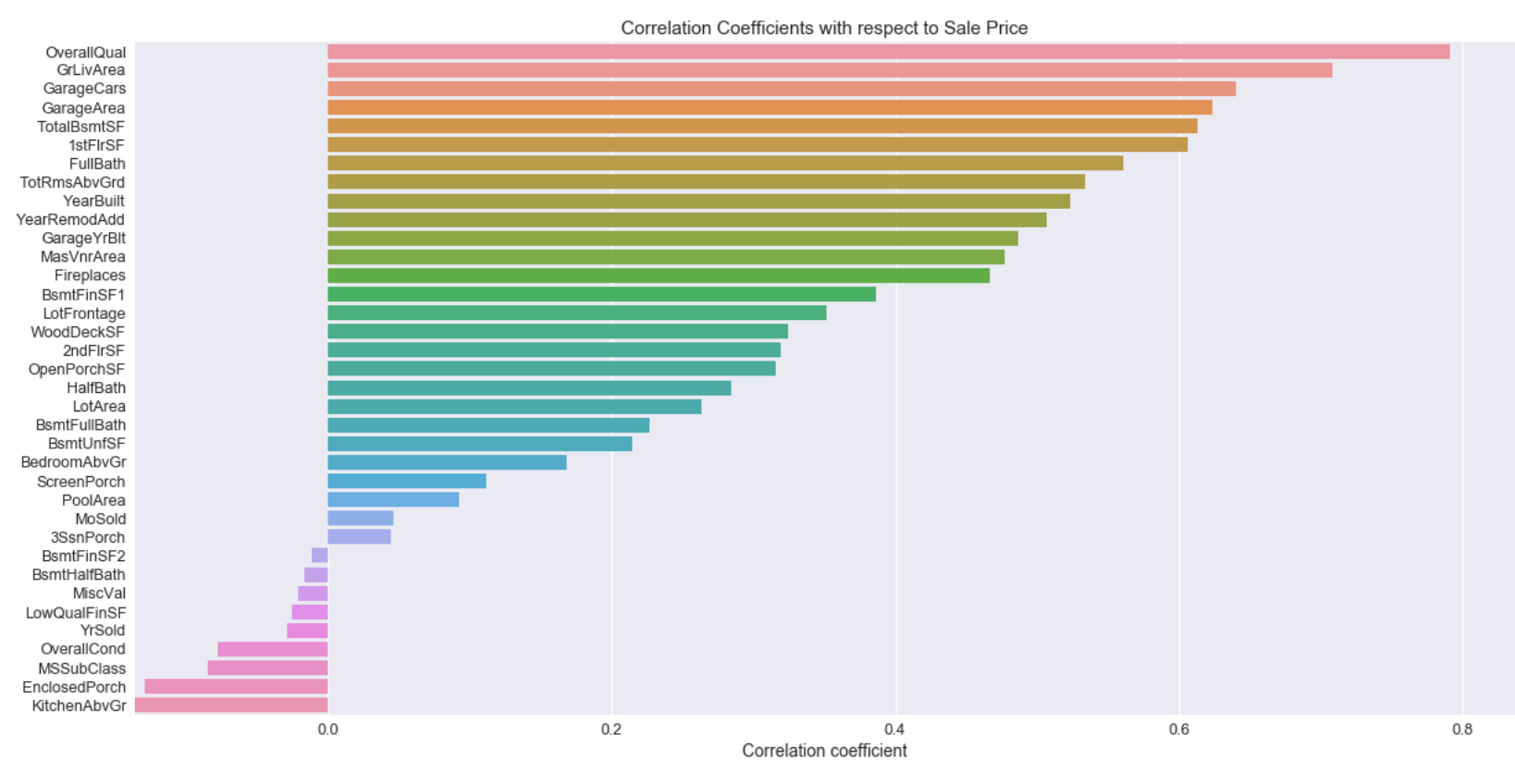
Figure 1. Without outliers

Figure 2. With outliers



**Exploratory Visualization**

Basic data investigation shows features correlation with respect to SalePrice. Graph shows that there is a number of features which correlates with SalePrice.

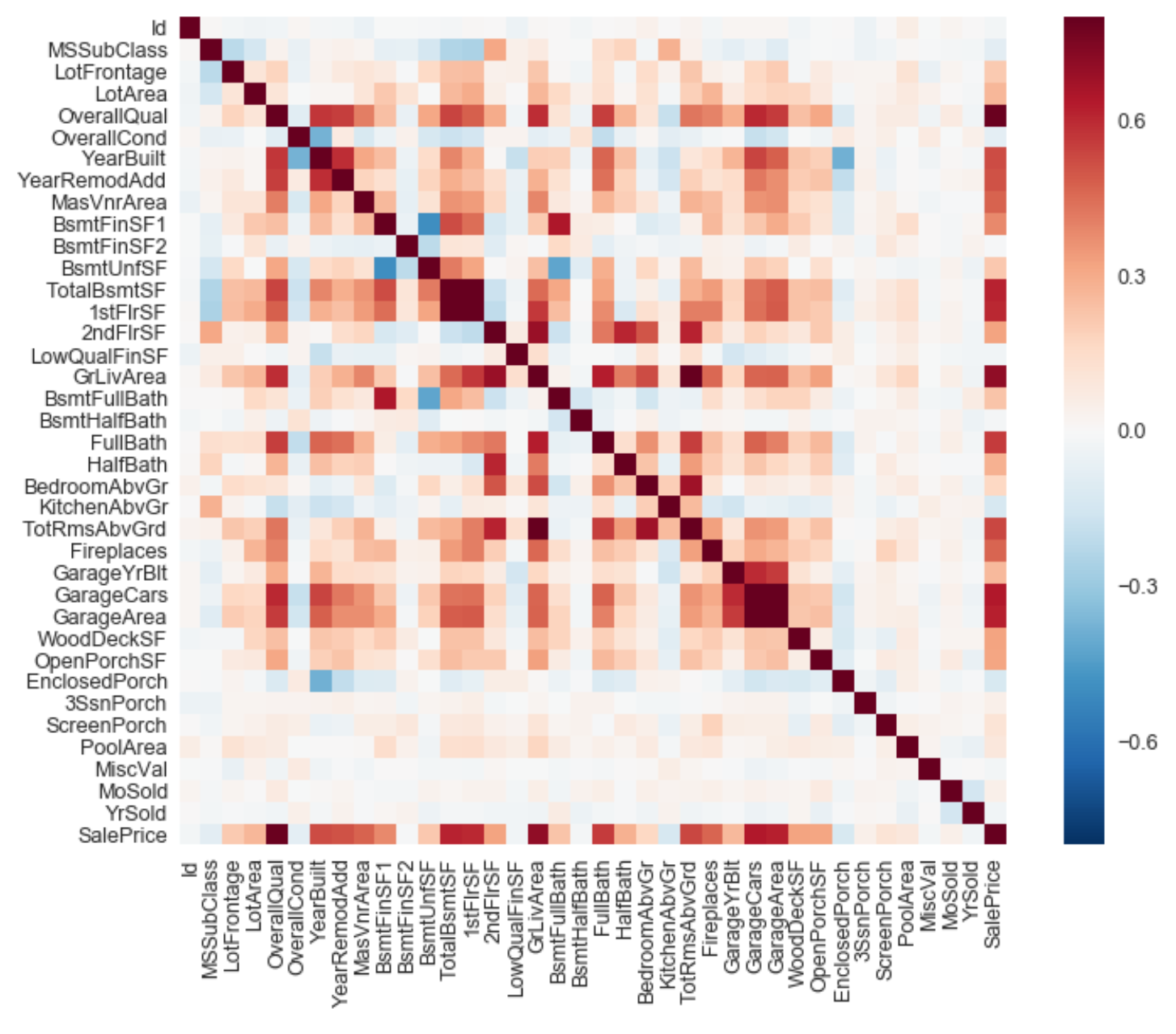


Top 5 features

* OverallQual Overall material and finish quality
* GrLivArea: Above grade (ground) living area square feet
* GarageCars: Size of garage in car capacity
* GarageArea: Size of garage in square feet
* TotalBsmtSF: Total square feet of basement area

Overall quality is a categorical value. Next four features describing a size of a property part. This is an evidence that size of house usually has the highest influence on the price.

Next chart shows correlation heatmap. Investigation shows that high correlated features usually describe the same thing. As for example ‘GarageArea’ and ‘GarageCars’.



**Algorithms and Techniques**

For this project I chose three ensemble models based on decision tree regression techniques.

To describe how decision tree model works I will reuse my explanation from the project ‘CharityML’ were the goal was to identify possible donors:

*Decision Tree model mirrors human decision making very closely. In this case we can imagine the learner is asking data questions about the donors with their income over 50K similar to this:*

*Q1: Do they have family? Model remembers the answer yes or no*

*Q2: Do they have positive capital gain?*

*Q3: Do they have the bachelor education level?*

*Learner can also ask question like this:*

*Q4: If they don't have family is their work class 'self-employed'?*

*Q5: If they don't have family and their work class is 'self-employed' is their education level 'HS-grad'?*

*By asking questions this way we create a tree of questions. We can create different trees which have different sequence of questions asked. We are looking for the best tree which delivers best predictions on our training and test data. During the training we can decide maximum depth of the tree.*

*Gradient Boosting works this way: To create a complex questionnaire which is should help to decide if person has income over 50K is huge effort for one guy. Therefore, we engage number of other guys. Each of them will get only subset of data. We also restrict guys to create limited number of questions. For example only 8. To be fair, we put all the records (aka data points) in a hat. First guy will draw his subset randomly from the hat. He will note down the data from these records and put them back into the hat. Based on the data he has, he creates his own questionnaire. This questionnaire will be tested on all records in the hat. Records which were false predicted will be multiplied in the hat. The size of the multiplication will determine the learning rate. The reason for multiplication is, that we want the next guy to have bigger chance to draw records from the hat, where previous guy was unsuccessful. To give him a chance to make questionnaire which can evaluate the data better. The same way we will continue with third guy, with forth guy and so on. We will evaluate the prediction success of each guy. We will stop adding new guys if the average prediction success will stop growing. We can also trust more to the guys which individual predictions are more successful than of the others. Predictions of these guys will be taken more into account before the final result of the group will be delivered.*

Decision Tree technique builds models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets. The model creates for this subset trees. These small trees we call weak learners because the can ‘answer’ only simple questions and their overall prediction success isn’t great. Ensemble models using the collective power of large number of the week learners, which together can make more accurate predictions.

Strengths of decision trees are:

* they are easy to interpret
* are able to handle both numerical and categorical data
* performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

and the weaknesses:

* prone to overfitting
* small changes in the dataset can lead to different trees

**Random Forest** **Regressor** is an ensemble classifier made using many decision tree models. Creates multiple trees with randomly selected data.

**Gradient Boosting Regressor** takes many week learners and combine results of them.

**XGBoost Regressor** is an optimized implementation of Gradient Boosting technique. More details are to found at <https://github.com/dmlc/xgboost>

Other possible techniques to use for this problem is

Logistic Regression

Strengths:

* Agnostic to correlations
* Fast and works well with small number of features
* Provides class probability

Weaknesses:

* Presumes a linear relationship between the features and the log-odds of the response
* Does not perform well with many features
* No more than two classes
* non-linear class separations are not well handled

Neural Networks

Neural network is good candidate for the problem solution. It will follow the idea that price of the house is influenced by the features. Some features have large influence on the price some are weaker. Some features have positive influence on the price some negative. This may work like weights in neural network. The architecture will be simple: number of input neurons equal to the number of features, next layer with 100 neurons and output layer with one neuron.

**Benchmark**

The property price strongly correlates with square footage, the simple model will be linear regression of square footage with respect to sale price.

Prediction of linear regression of ‘SalePrice’ with respect to ‘GrLivArea’ (above ground living area square feet) deliver following results:

RMSE : 56034.3039

RMSE of logarithms : 0.2756

R2 score : 0.502149

Summing square footage of overall basement area with first and second floor creates new feature ‚TotalSF‘. ‘TotalSF’ is a sum of basement, 1st floor, and 2nd floor square footage.

Prediction of linear regression of ‘SalePrice’ with respect to ‘TotalSF‘ deliver even better results:

RMSE : 49471.9159

RMSE of logarithms : 0.2406

R2 score : 0.611931

**III. Methodology**

**Data Preprocessing**

#### Basic preprocessing

Data from Keggle’s come in two sets. One part are data to be used for training and the second part are the test data to be used for prediction and is to be submitted to Kaggle competition.

Both sets consists of ‘Id’ column, which is only sequence number and doesn’t hold any information about the house.

Basic preprocessing will include

* separation of ‘SalePrice’
* droping of the ‘Id’ column
* joining of the training and testing sets together

#### Missing Data

Missing data will be proceed following way:

* numeric data – if missing, imputed by zero
* categorical data – if missing, imputed by ‘None’
* features 'Electrical', 'Functional', 'KitchenQual', 'MSZoning', 'Exterior1st', 'Exterior2nd' and 'SaleType' were imputed by most frequent values in the data set

#### Removing Outliers

I will use simple rule for an elimination of outliers

'GrLivArea' > 4000 SF

and

'SalePrice' <300000

#### Processing categorical data

It will be necessary to transform categorical values to numbers. I will use label encoders and one hot encoding.

#### Dealing with skewness

Investigating of data shows number of features with skewed distribution.

**Implementation**

My goal was to create procedures for data preprocessing which I can freely ‘turn off and on’ without the need to change a code. This has consequences, that I created long *jupyter notebooks.* This is the reason why I split notebook into two notebooks. One for data analysis, next for model creation and tuning. Disadvantage of these notebooks is, that I need to run all the fields with the code which is necessary to proceed. Moving some procedures into separate .py files solved this.

Data are processed and model is created in Python 2.7 programming language using libraries

* pandas - data analysis toolkit v0.20.3
* scikit learn – machine learning library in Python
* matplotlib
* seaborn
* XGBoost Library

Data are analyzed using jupyter notebook: data\_analysis.ipynb

and for model tuning I created another notebook: model\_tuning.ipynb

Some procedures are implemented in separate files

* usefull\_methods.py
* do\_actions.py
* stopwatch.py

**Refinement**

I used two techniques how to tune the model. I will implement each preprocessing procedure in separate function, that I can simply include or exclude it in data preprocessing. I will observe which combination of preprocessing steps deliver best results. The possible actions which can be turned on and of:

* elimination of outliers
* eliminate skew
* remove non-significant features

Second technique is tuning hyperparameters for chosen regressors. I will use grid search technique to tune *max\_depth* and *n\_estimators*.

*max\_depth* is maximal depth of decision tree  
*n\_estimators* is number of estimators used by the ensemble regressor

Looking for best hyperparameters also with GridSearch is time consuming process. Therefore I proceeded this action in three basic steps.

* for max\_depth define 5 values from the range of 2 to 100
* for n\_estimators define 5 values from the range of 100 to 3000
* run GridSearch
* evaluate results and create new ranges of 5 values for the next run

new ranges

* + extend the range toward smallest or largest value if the first or the last value has been reached
  + try to precise parameters if best parameter is within the range
* run GridSearch with new ranges
* repeat this till best parameters found

A visualization of the performance with different settings help me also to learn in which direction to move

Excerpt from the code showing parameter ranges for the last GridSearch run. Bold values are values with the best model performance.

RandomForestRegressor   
param\_grid = {'max\_depth': [ 20, **25**, 30, 32 ], 'n\_estimators':[375, 400, **500**, 600, 700 ]}

GradientBoostingRegressor  
param\_grid = {'max\_depth': [ **2**, 5, 25, 30 ], 'n\_estimators':[ 800, 900, 1000, 1100, 1200 ]}

XGBRegressor  
param\_grid = {'max\_depth': [ **2**, 3, 4], 'n\_estimators':[ 1200, 1400, 1600, **1800**, 2000 ]}

Best performing hyperparameters will be used for final model

**IV. Results**

Tuning model by changing actions and searching for best hyperparameters delivered slightly different results. Best results delivered a combination of:

* elimination of outliers ON
* eliminate skew OFF
* remove non-significant features OFF (all features active)

**Best results on training data**

XGBoost

RMSE : 21546.7702

RMSE of logarithms : 0.1181

R2 score : 0.926484

Gradient Boost

RMSE : 22004.9835

RMSE of logarithms : 0.1224

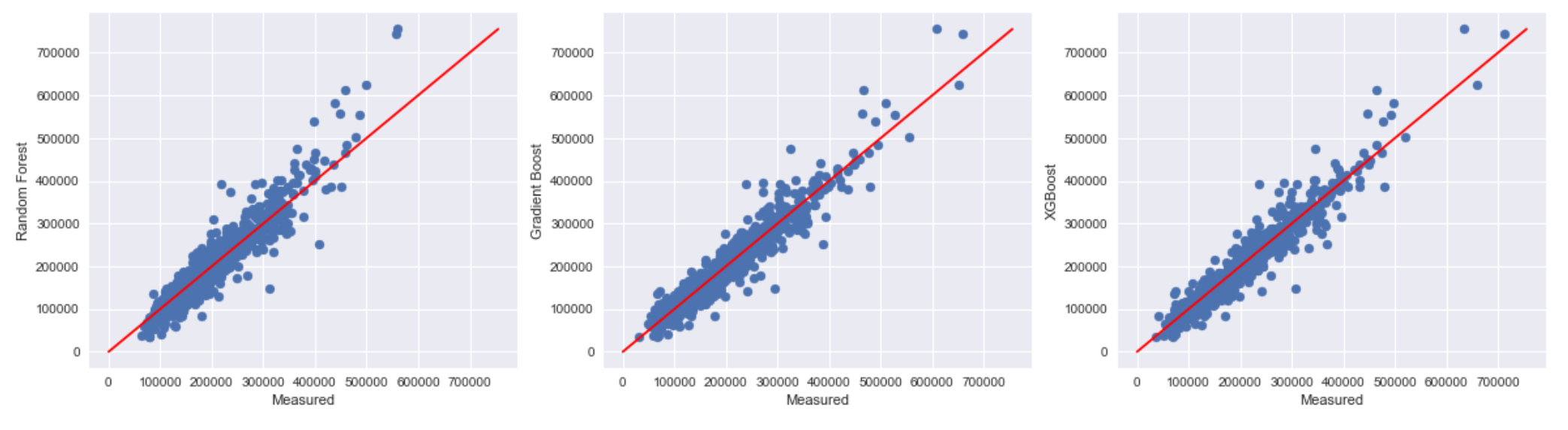
R2 score : 0.923324

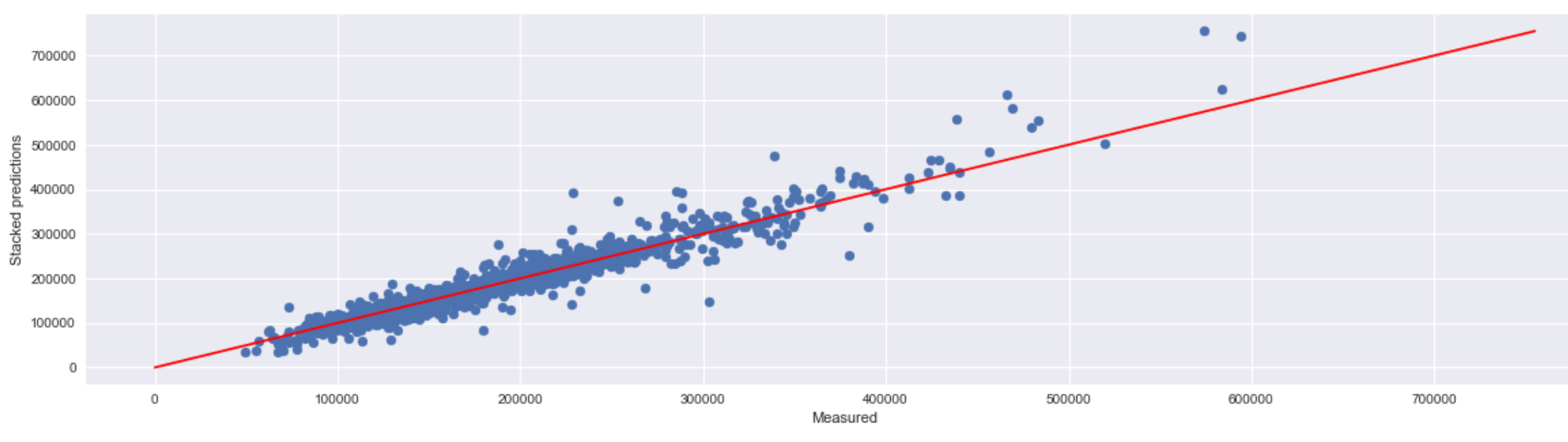
Random Forest

RMSE : 25372.8165

RMSE of logarithms : 0.1352

R2 score : 0.898058





Weights calculation for stacked regressions is calculated as relation of particular r2 score to sum of all r2 scores.

sum\_r2 = xgb\_r2 + gbr\_r2 + rfr\_r2

xgb\_w = xgb\_r2/sum\_r2

gbr\_w = gbr\_r2/sum\_r2

rfr\_w = rfr\_r2/sum\_r2

**Best results on Kaggle test data**

Result of stacked regressions gives better result in Kaggle competition: RMSE

* 0.06629 this best result so far on Kaggle (valid August 12th 2017)
* 0.10567 2nd best result on Kaggle
* ….
* ….
* **0.12723** my best result which I was not able reproduce later
* **0.12900** my result for stacked Decision Tree regressions
* **0.13289** for XGBoost, the best individual model

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **RMSE** | **RMSE log** | **R2 score** |
| Linear Regression SalePrice/GrLivArea | 56034.3039 | 0.2756 | 0.502149 |
| Linear Regression SalePrice/TotalSF | 49471.9159 | 0.2406 | 0.611931 |
| Random Forest Regression | 25899.3463 | 0.1325 | 0.893783 |
| Gradient Boost Regression | 22624.9584 | 0.1199 | 0.918942 |
| XGBoost Regression | 21935.7116 | 0.1144 | 0.923806 |
| Stacked RFR/GBR/XGB | 22240.0829 | 0.1153 | 0.921677 |
| SupportVectorMachine Regression |  |  |  |

**Robustness**

Robustness is a metric which evaluates the model from the perspective of prediction stability if some changes in occur. To test robustness I chose randomly selected 80% of the training set to train the model. I chose one row from the test set which represents the features of the house I want to predict price for. For the next round I chose another 80% of the training set to train the model and calculated predicted price again. I repeated the prediction 10 times. It is expected, if model is stable, that the prediction will be equal in the best case. Here is the result:

Predicted price of the TEST HOUSE: 311344.151018

Predicted price of the TEST HOUSE: 312353.180924

Predicted price of the TEST HOUSE: 310742.888764

Predicted price of the TEST HOUSE: 305297.402991

Predicted price of the TEST HOUSE: 309272.883843

Predicted price of the TEST HOUSE: 310039.075738

Predicted price of the TEST HOUSE: 315665.998018

Predicted price of the TEST HOUSE: 312224.561249

Predicted price of the TEST HOUSE: 312308.370025

Predicted price of the TEST HOUSE: 316477.858605

Statistics:

* + mean 311572.637117
  + std 3164.951906
  + min 305297.402991
  + 25% 310215.028994
  + 50% 311784.356133
  + 75% 312341.978199
  + max 316477.858605

It shows that standard deviation of the predicted prices is around 10%. I thing this is quite good prediction from practical point of view and shows that model is so stable that it can be used.

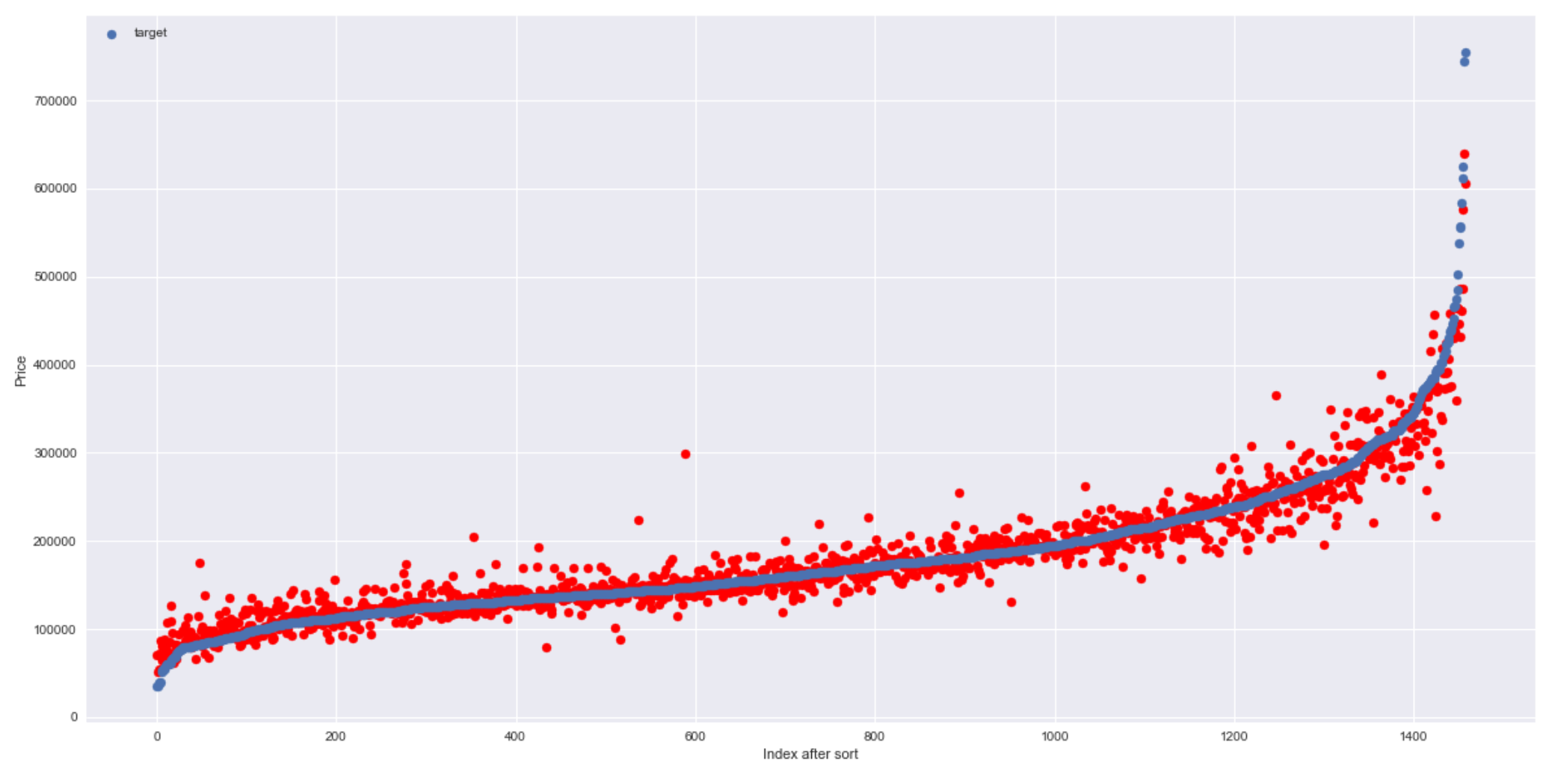
**V. Conclusion**

**Free-Form Visualization**

Following visualization shows the differences between predicted values and target values. Basically the predicted values follow the target variable, anyway there are some data points which are quite distant from the target curve. In case we would like to improve the model, I would suggest to investigate these points.

Set is sorted in respect to target values

* red dots: predicted values
* blue dots: target values
* x-axis: position in sorted dataset
* y-axise: price of the house



**Reflection**

This was very interesting project. On the beginning, it looked like a simple task. But it is possible to spend weeks trying all ideas one can have.

I am satisfied with the results, and I believe that model is usable for estimation of house prices for given area and time period. I have tried to achieve better results on the Kaggle competition and I spent lot of time with tuning but wasn’t able really to make a significant step forward. I have learned, that one skill of the good machine learning engineer is also to know when to stop twisting the current model.

**Improvement**

There is a number of possible ways how to improve the model. One way is to use other methods of machine learning as for example:

* use other regressors
* tune more or other hyperparameters
* support vector regression (I tried to use this method, but it delivers strange results, therefore I skipped it. Probably because I didn’t transform, normalize the data? Needs more time to investigate.
* Neural Networks Regressor – this method I had in the scope, but I read that this method might not work well because not sufficient data for NN training available. Anyhow would be nice to try, but it will costs me another month ☹.

There are still open possibilities how to tune current model (based on ensamble decision trees). Some ideas are:

* use another function to unskew data
* precise selection of data to be dropped
* tuning of weights of models stacking
* data transformation – data normalization
* use other numeric data as categorical (yearBuild, yearSold)