**Data Scientist**

**NanoDegree**

**Capstone Project Report:**

**Sberbank Russian Housing Market**

İsmail Yıldız

Date: 27-05-2020

# Definition

## Project Overview

In the area of Data Science, Kaggle is one of the most popular platforms. Basicly Kaggle runs Data Science competitions where problems came from both industry and academia. In each competition, competition owner defines the problem, determine evaluation metrics and publishes train/test dataset to solve the problem. Competitor data scientists struggle to find the best solution to the problem and generally winners get a money prize.

This project is derived from one of Kaggle competitions that is brought by Sberbank, Russia’s oldest and largest bank. Sberbank want to find a way to help their customers by making predictions about real house prices, so renters, developers, and lenders will be more confident when they sign a lease or purchase a building.

There are various studies regarding house price prediction. One of these studies is article “Real estate value prediction using multivariate regression models” created by Manjula, R & Jain, Shubham & Srivastava, Sharad & Rajiv Kher, Pranav. and published in IOP Conference Series: Materials Science and Engineering (2017). In this paper, Machine Learning methods are applied to get better house price estimations.

## Problem Statement

This project is based on the Kaggle competition that is named as “Sberbank Russian Housing Market”. The main goal is to define/find a solution by using data science methods in order to predict real house prices. Competition details can be reached by the follwing link <https://www.kaggle.com/c/sberbank-russian-housing-market>.

Although the housing market is relatively stable in Russia, the country’s volatile economy makes forecasting prices as a function of apartment characteristics a unique challenge. Complex interactions between housing features such as number of bedrooms and location are enough to make pricing predictions complicated. Adding an unstable economy to the mix means Sberbank and their customers need more than simple regression models in their arsenal.

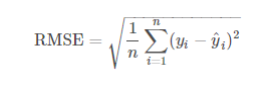
In this competition, Sberbank is challenging Kagglers to develop algorithms which use a broad spectrum of features to predict realty prices. Competitors will rely on a rich dataset that includes housing data and macroeconomic patterns. An accurate forecasting model will allow Sberbank to provide more certainty to their customers in an uncertain economy.

Basicly the problem is a well known type of regression problem in Machine Learning. Some features of individual house like room number, square feet will be used to predict the house prices.

## Metrics

Kaggle competition originally defined the evaluation metric as Root Mean Square Log Error (RMSLE). I think it a very good measure in regression type problems since log transform can decrease the effect of single extreme errors cases. Also after the project, I plan to submit results to Kaggle platform order to benchmark my solution with other solutions in the platform.

RMSE is defined by the formula below where y is the actual value and y hat is the predicted value.



RMSLE is, effectively, the RMSE of the log-transformed predicted and target values.

This measurement is useful when there is a wide range in the target variable, and you do not necessarily want to penalize large errors when the predicted and target values are themselves high. It is also effective when you care about percentage errors rather than the absolute value of errors.

# Analysis

## Data Exploration

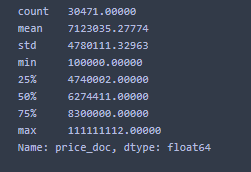
Provided datasets include train and test datasets. Train dataset include full features and the the target variable ‘price\_doc’. Whereas test dataset does not include the target variable (actual truth values) and test dataset is intended for measuring the Kaggle competitors success with untouched dataset. So throughout the project, I will use train dataset only for model training since target variable is not known in the test dataset.

Important findings from exploratory analysis are listed below:

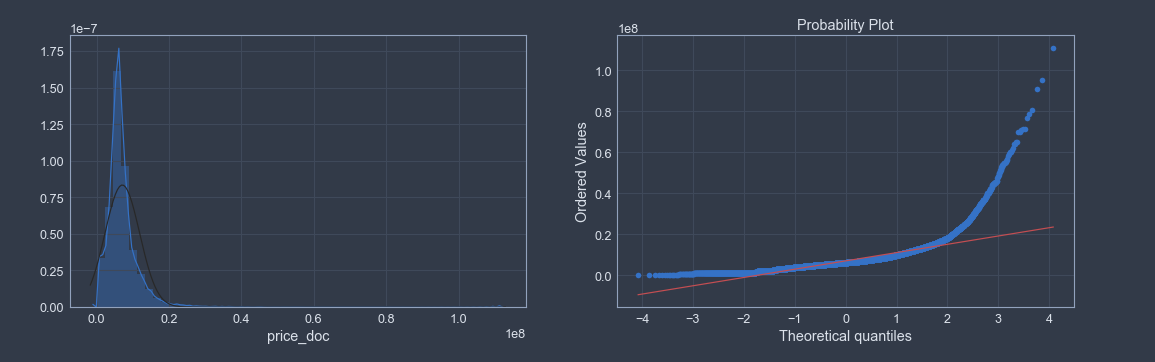
* Dataset include 30471 data points or records.
* It include 291 features and the target variable.
* Target variable is called ‘price\_doc’ which the the actual price of the house at the time of sale.
* ‘id’ column at the beginning of features is the unique identifier for house sales, it can be assumed as record id or transaction id. So this feature will not be used in the next phases.
* Among 291 features:
  + There is only one timestamp feature showing date of transaction.
  + There are 15 categorical variables among all features.
  + Remaning features are numeric.

## Exploratory Visualization

The most important column is the target variable which is ‘price\_doc’. As shown in below figures there are some outliers in the target variable and there is a skew in the distribution.

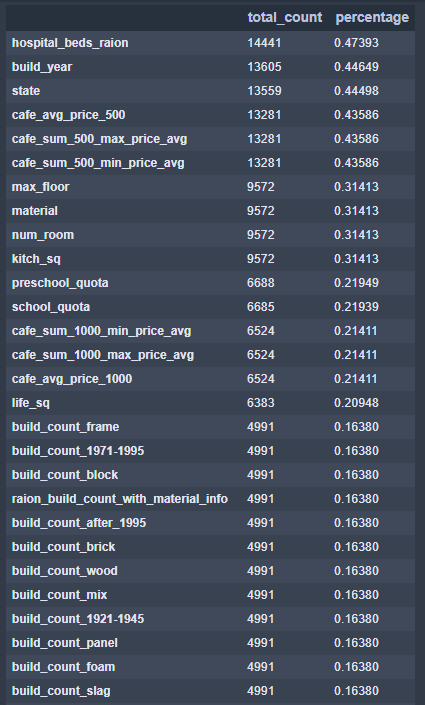


1. price\_doc quantiles



1. price\_doc distribution

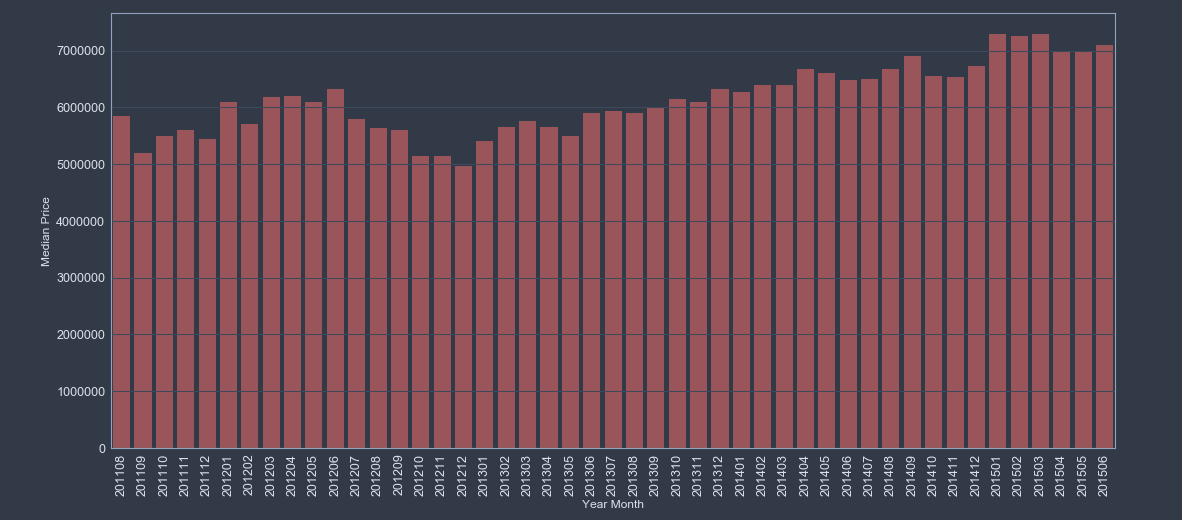
First thing to notice is that the data contains lots of NaN values, which need to be dealt with. Below figure shows top NaN including columns.



1. Top NaN columns and percentages

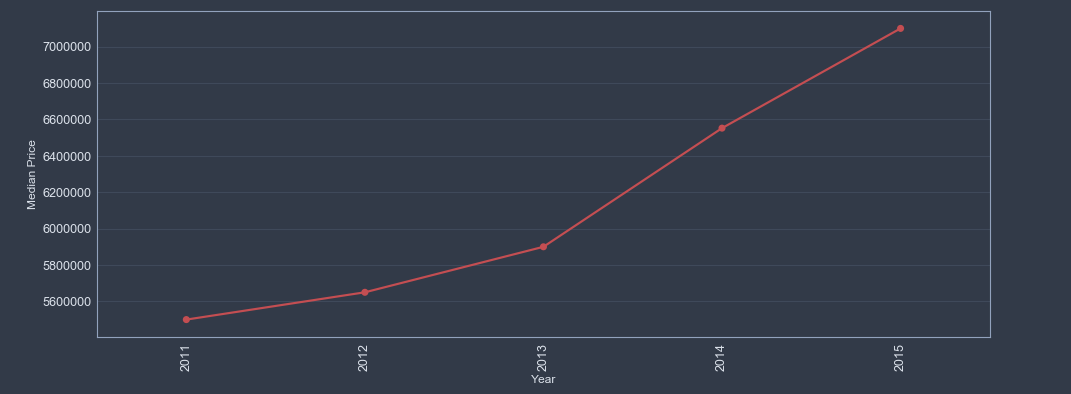
Sale price trend another important information to investigate. Figure 4 below show the trend.

There are some variations in the median price with respect to time. Towards the end, there seems to be some linear increase in the price values.



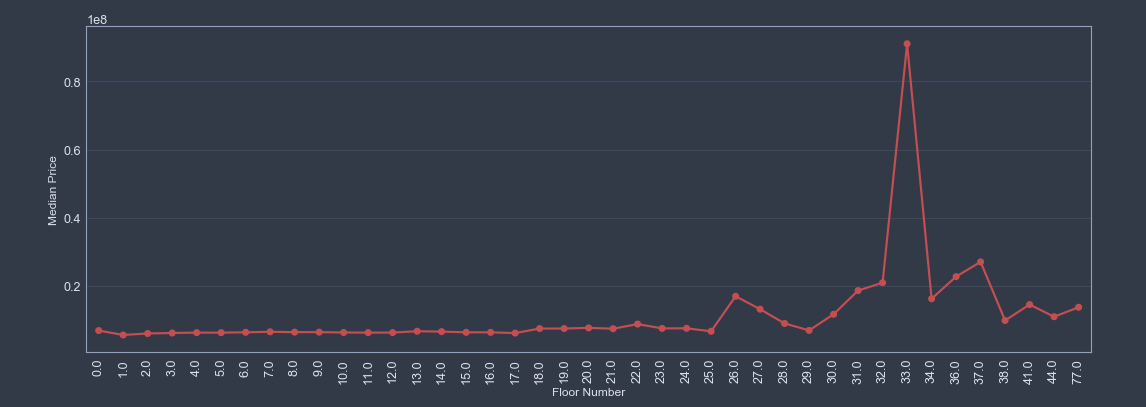
1. Year-Month based price trend

When we eliminate the month and just interested in year info, then linear increase is more obvious as shown below.

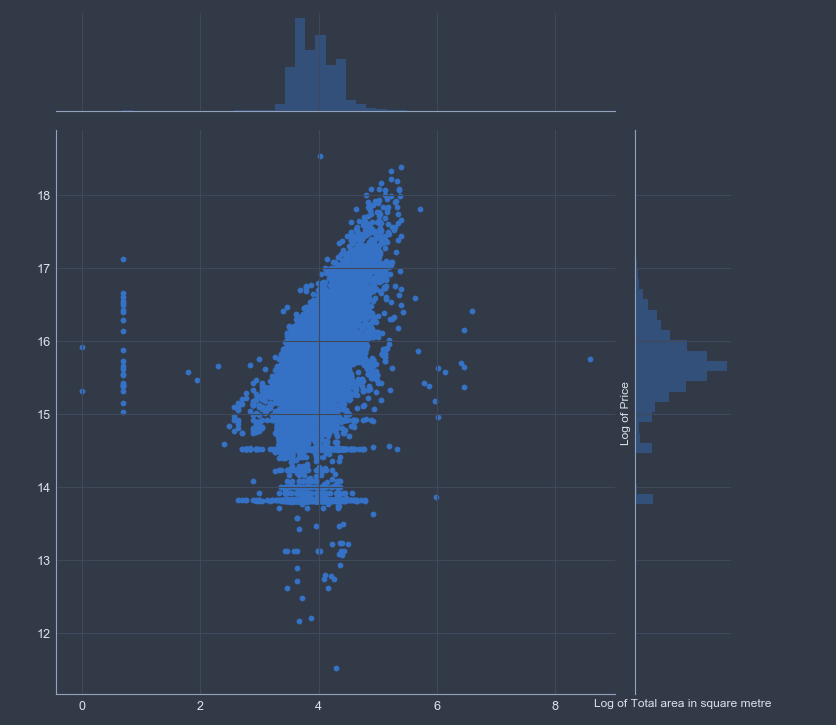


1. Year based price trend

Floor information may be another important feature for a house. Figure 6 shows an overall increasing trend with floor number. There is a sudden increase in the house price is observed at floor 18. Also note that individual houses seems to be costlier as well (refer to 0 floor houses).



1. Floor number price relation

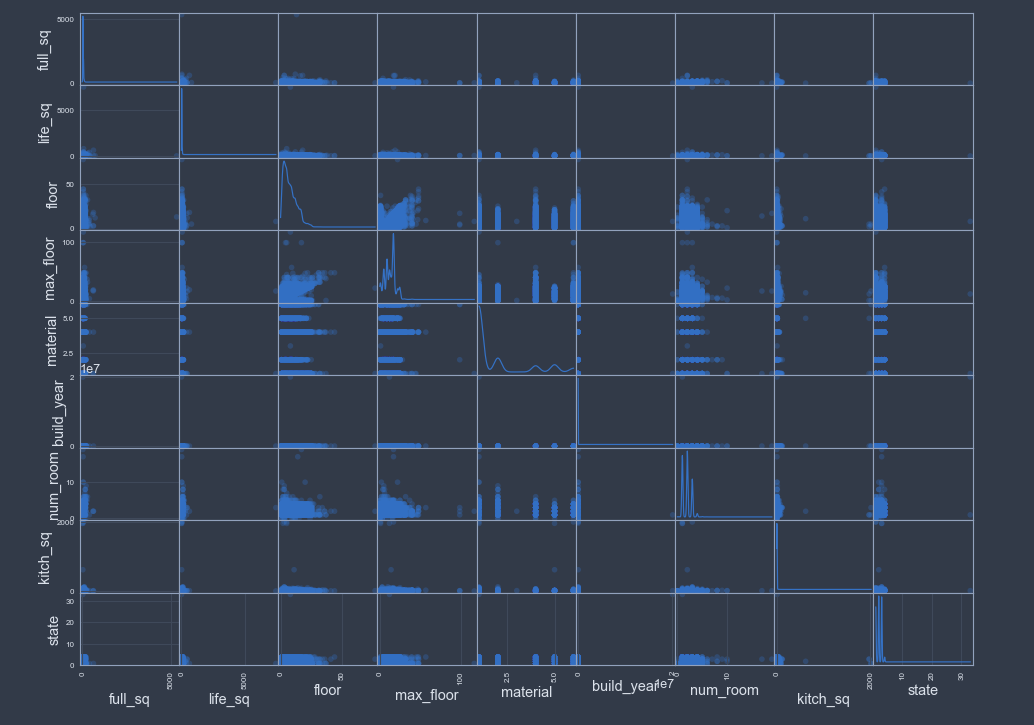
Square feet area of houses are one of the most important factors affecting the price, as shown below there is a strong correlation.  


1. Log Price vs Log Total Area

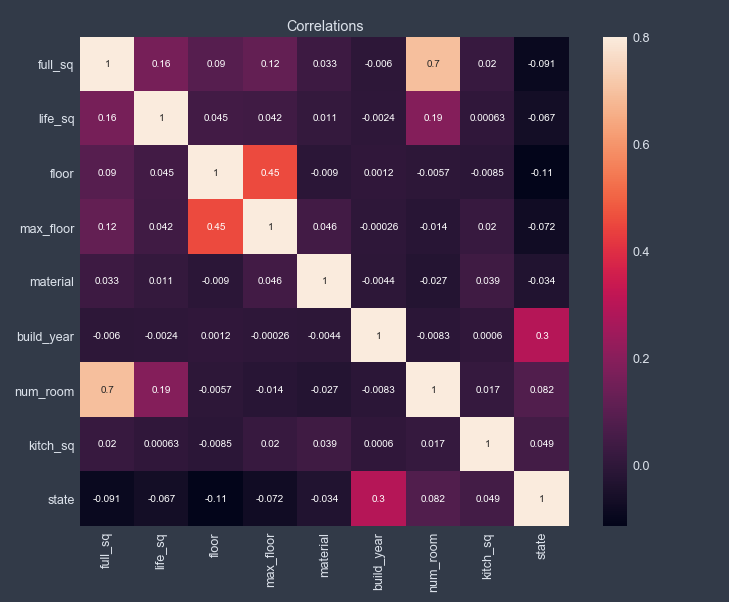
Another point is to discover correlations between features. But since there 291 feature it is not feasible to show all of them in the same figure. Instead, we try to group features into plausible groups. So I decided to generate correlation matrixes as following groups and generated correlation charts:

1. Housing Features:

* price\_doc: sale price (this is the target variable)
* id: transaction id
* full\_sq: total area in square meters, including loggias, balconies and other non-residential areas
* life\_sq: living area in square meters, excluding loggias, balconies and other non-residential areas
* floor: for apartments, floor of the building
* max\_floor: number of floors in the building
* material: wall material
* build\_year: year built
* num\_room: number of living rooms
* kitch\_sq: kitchen area
* state: apartment condition
* product\_type: owner-occupier purchase or investment

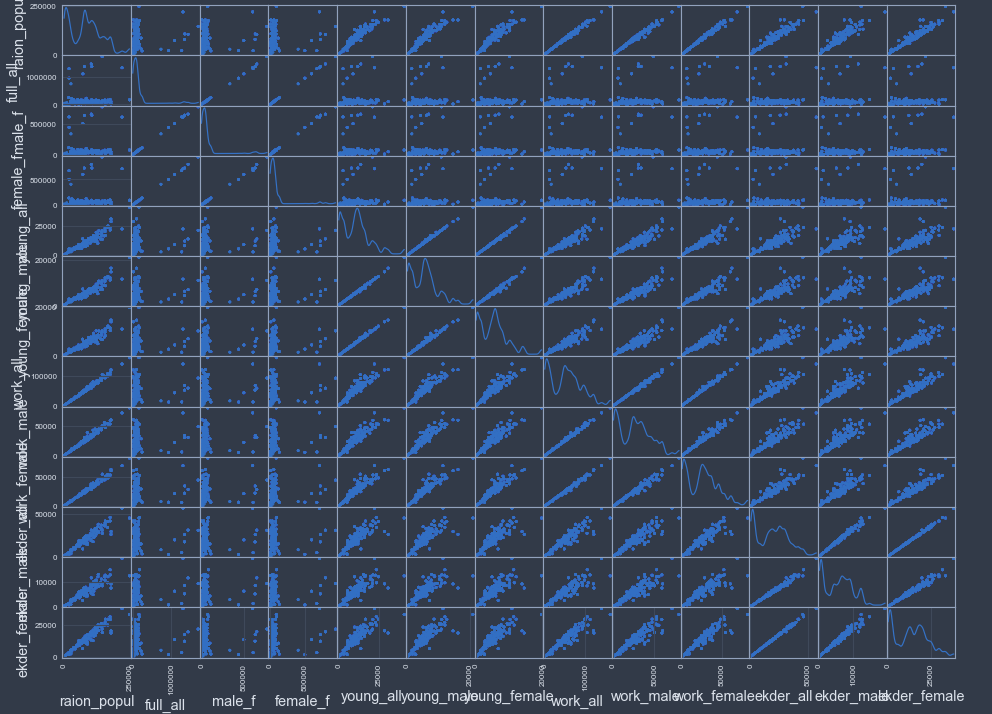


1. Scatter Matrix

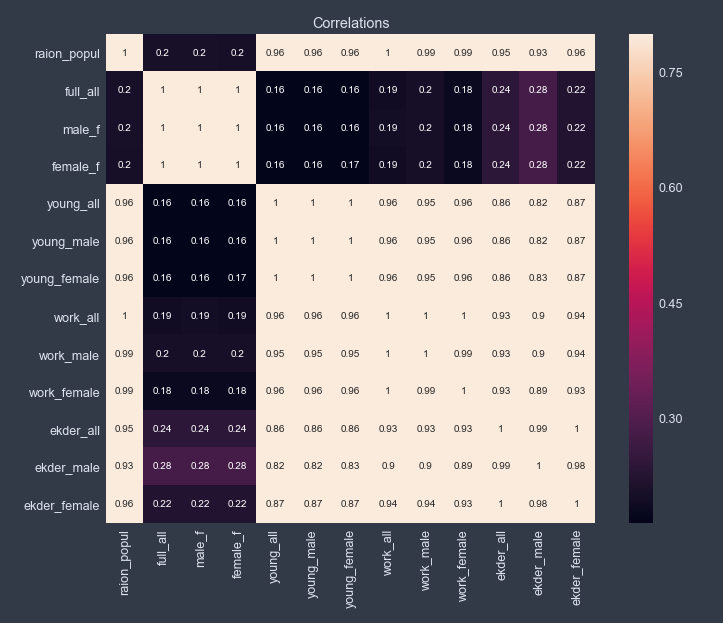


1. Correlation Matrix
2. Population Features:

* raion\_popul: Number of municipality population. district
* full\_all: Total number of population in the municipality
* male\_f: Male population
* female\_f: Female population
* young\_all: Population younger than working age
* young\_male: Male population younger than working age
* young\_female: Female population younger than working age
* work\_all: Working-age population
* work\_male: Male working-age population
* work\_female: Female working-age population
* ekder\_all: Population older than working age
* ekder\_male: Male population older than working age
* ekder\_female: Female population older than working age

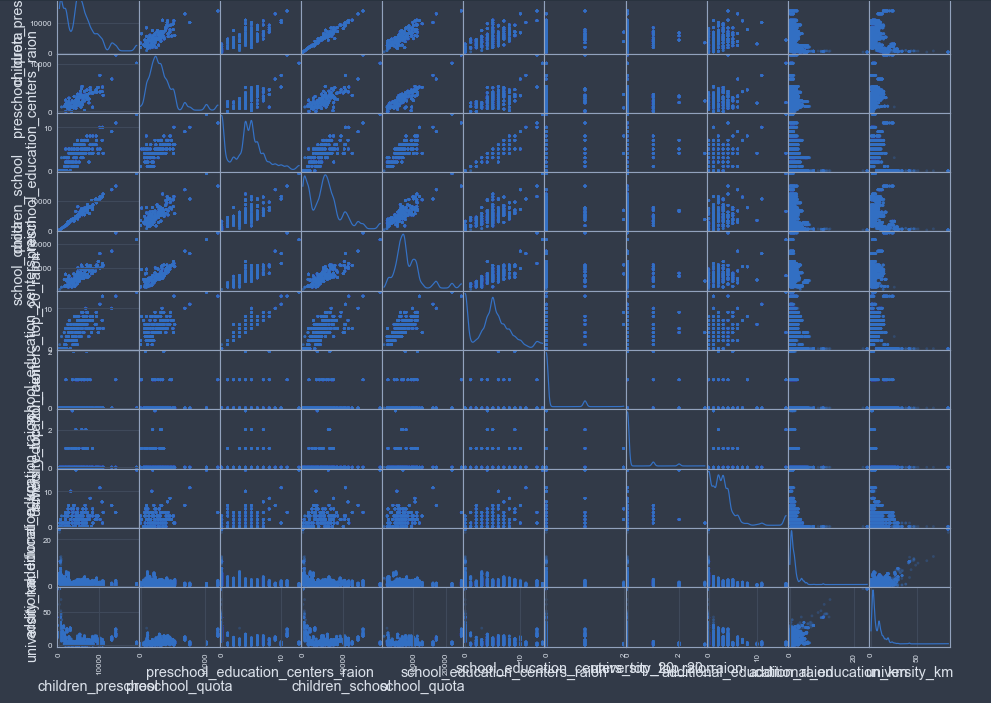


1. Scatter Matrix

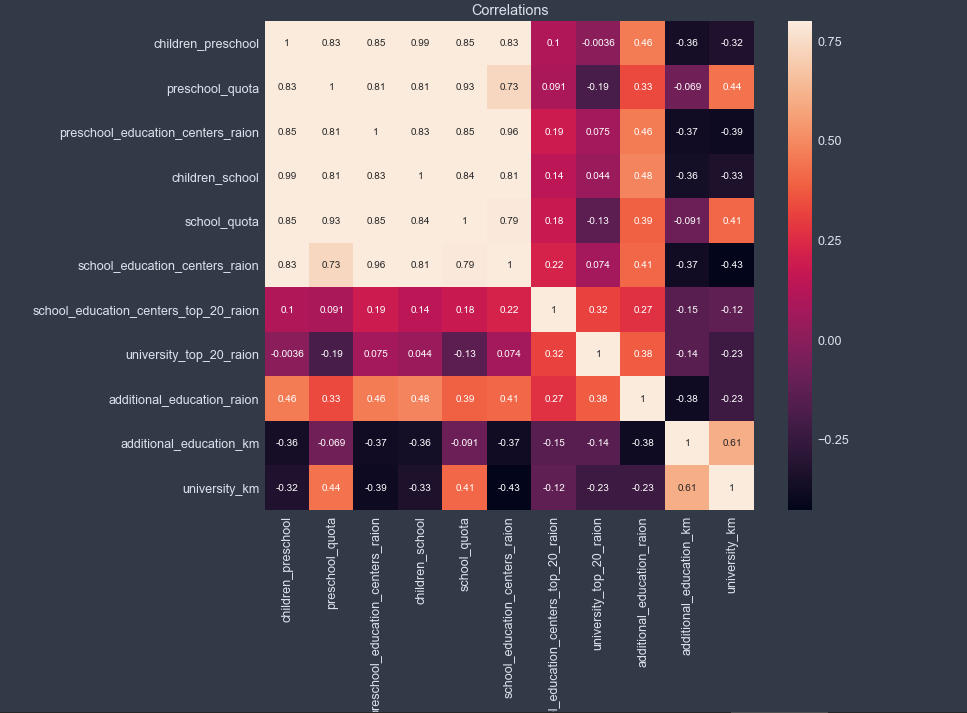


1. Correlation Matrix
2. School Features:

* children\_preschool
* preschool\_quota
* preschool\_education\_centers\_raion
* children\_school
* school\_quota
* school\_education\_centers\_raion
* school\_education\_centers\_top\_20\_raion
* university\_top\_20\_raion
* additional\_education\_raion
* additional\_education\_km
* university\_km

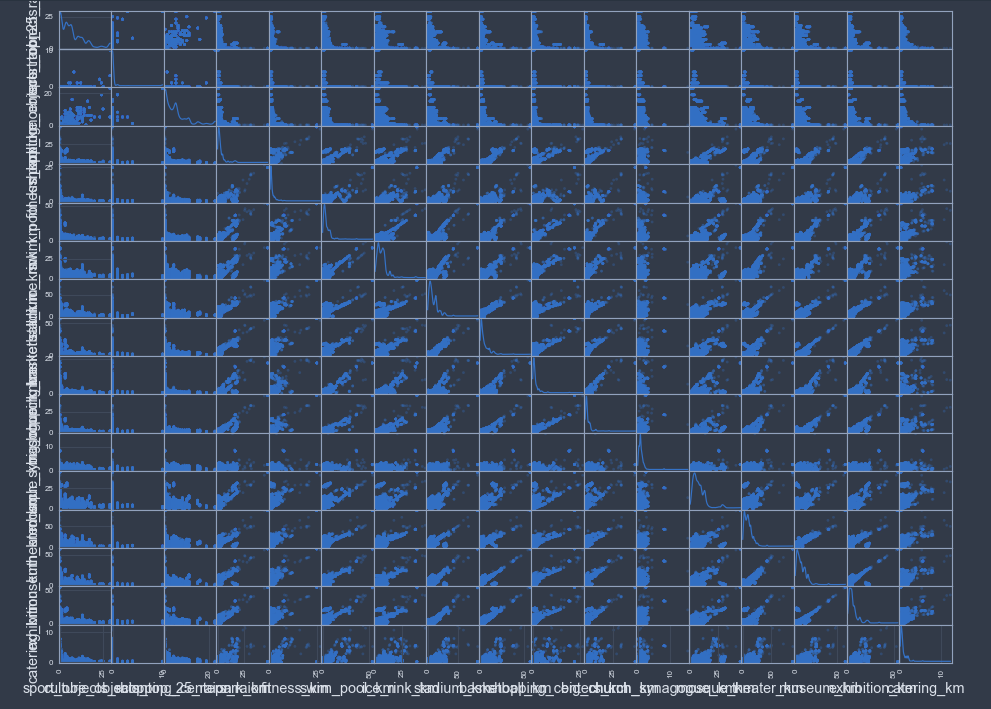


1. Scatter Matrix

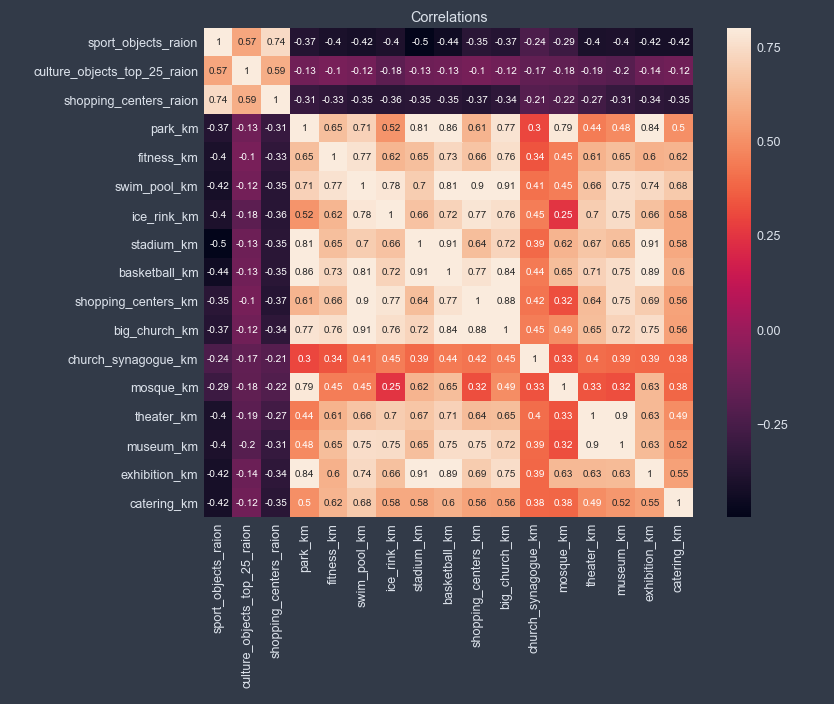


1. Correlation Matrix
2. Cultural Features:

* sport\_objects\_raion
* culture\_objects\_top\_25\_raion
* shopping\_centers\_raion
* park\_km
* fitness\_km
* swim\_pool\_km
* ice\_rink\_km
* stadium\_km
* basketball\_km
* shopping\_centers\_km
* big\_church\_km
* church\_synagogue\_km
* mosque\_km
* theater\_km
* museum\_km
* exhibition\_km
* catering\_km

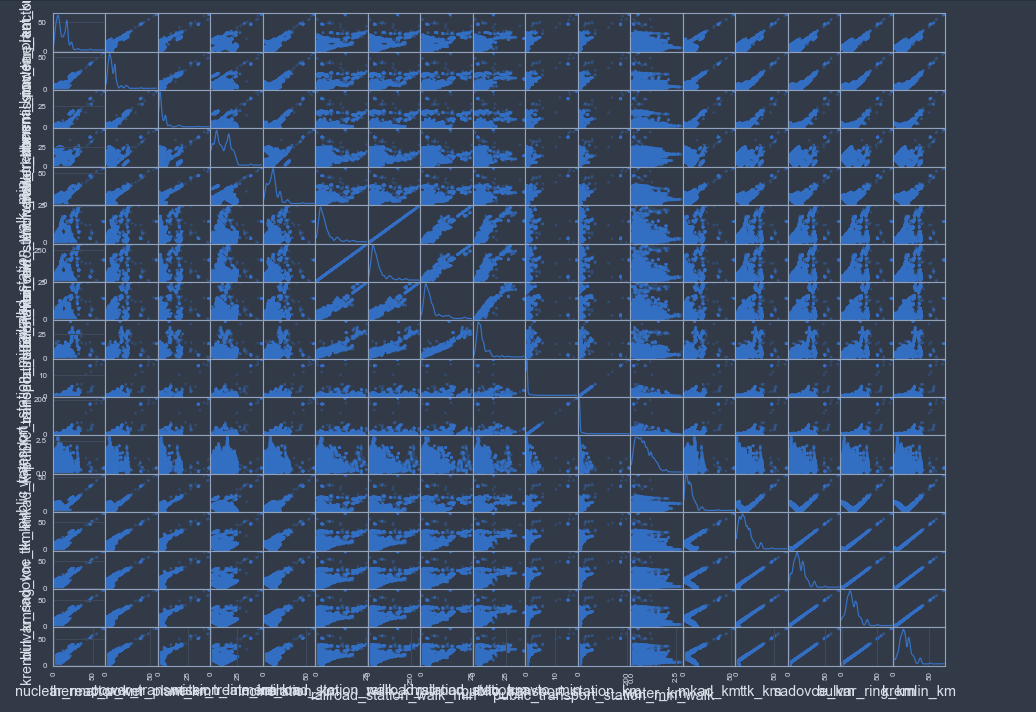


1. Scatter Matrix



1. Correlation Matrix
2. Urban Features:

* nuclear\_reactor\_km
* thermal\_power\_plant\_km
* power\_transmission\_line\_km
* incineration\_km
* water\_treatment\_km
* incineration\_km
* railroad\_station\_walk\_km
* railroad\_station\_walk\_min
* railroad\_station\_avto\_km
* railroad\_station\_avto\_min
* public\_transport\_station\_km
* public\_transport\_station\_min\_walk
* water\_km
* mkad\_km
* ttk\_km
* sadovoe\_km
* bulvar\_ring\_km
* kremlin\_km



1. Scatter Matrix



1. Correlation Matrix

## Algorithms and Techniques

In this section, all the algorithms and techniques used is discussed briefly. The choice of algorithms and techniques used are largely inspired by data properties and most used methods in tackling Machine Laerning competitions.

**XGBoost:**

The first algorithm of choice is XGBoost, this is a widely accepted and proved method most of the Kaggle competitions. It has a rich history of been successful in many recent competitions, this one of the major motivations to use XGBoost as the first algorithm.

Boosting is a sequential technique which works on the principle of ensemble. It combines a set of weak learners and delivers improved prediction accuracy. At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher. This technique is followed for a classification problem while a similar technique is used for regression.

Extreme Gradient Boosting or XGBoost is a variant of tree boosting method with deep considerations of system optimizations and fundamental principles machine learning. The main advantages of XGBoost are:

* Regularization: In machine learning, it is easy to build an algorithm that easily overfits the model. In XGBoost there is a more regularized model formalization to control overfitting. The parameters that aid this include: max\_depth(maximum depth of the tree), min\_child\_weight(the minimum sum of weights of all observations required in a child), gamma (minimum loss reduction required for a split), alpha(L1 regularization), lambda(L2 regualrization)
* Parallel Processing: Compared to most of the boosting algorithms, XGBoost is exceedingly fast as it implements parallel processing.
* Handling Missing Data: XGBoost has a built-in routine to handle missing values. Upon encountering a missing value, the algorithm ties different methods and learns which path to take for a missing value in future.
* Built-in Crossvalidation: XGBoost allows user to run a cross-validation at each iteration of the boosting process and thus it is easy to get the exact optimum number of boosting iterations in a single run.

**Multi Layer Perceptron:**

A multilayer perceptron is a network simple neuron called neurons. A perceptron uses computes a single output from a multiple inputs by forming a linear combination according to input weights and using a non-linear activation function. A single perceptron is not very useful because of its limited mapping ability. The perceptions can, however, be used as building blocks of a larger, much more practical structure. A typical multilayer perceptron (MLP) network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer.

On the downside, increasing layers build a good model can easily result in overfitting. Thus, to reduce overfitting regularization techniques such as: droupout(randomly drop units during training so that the doesn’t adapt too much), batch-noramlization(a method to reduce internal covariate shift in neural networks) early stopping(stopping the training when validation error increases)

**Random Forest:**

Random Forest is a supervised learning algorithm. As seen from its name, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result.

To say it in simple words, Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems.

Another great quality of the random forest algorithm is that it is very easy to measure the relative importance of each feature on the prediction. Sklearn provides a great tool for this, that measures a features importance by looking at how much the tree nodes, which use that feature, reduce impurity across all trees in the forest. It computes this score automatically for each feature after training and scales the results, so that the sum of all importance is equal to 1. Through looking at the feature importance, you can decide which features you may want to drop, because they don’t contribute enough or nothing to the prediction process. This is important, because a general rule in machine learning is that the more features you have, the more likely your model will suffer from overfitting.

**GridSearch:**

Gridsearch is a simple hyperparameter optimization method where, exhaustive searching / sweeping of manually specified subset of hyperparameter space is learning algorithm is performed. The performance of a gridsearch technique is mostly guided by the cross validation score of the training set.

In this project, gridsearch method is derived from the scikit-learn library and 5-fold cross validation method is used to find the best set of hyperprameters for Machine Learning algorithms.

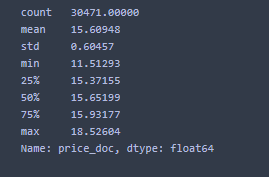
## Benchmark

Since the problem is a regression problem, I will use the output result of Linear Regression with 5-fold cross validation as a benchmark model. Then I will try to improve by using other algorithms and techniques.

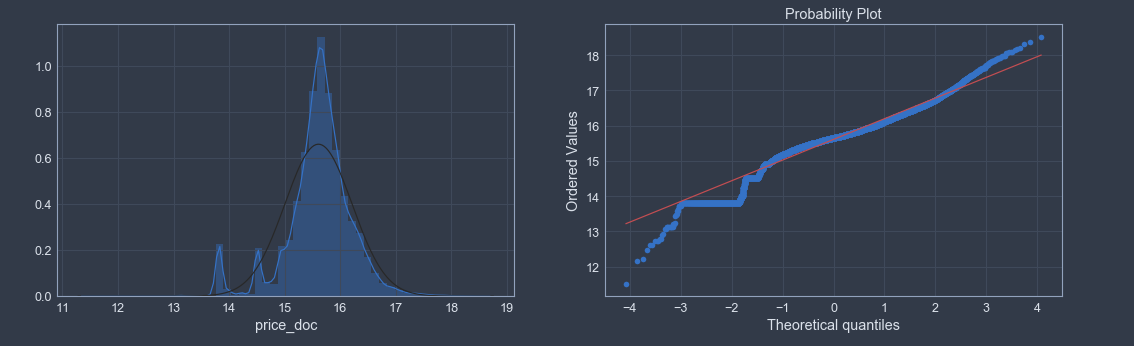
# Methodology

## Data Preprocessing

Since target variable is not uniformly distributed and have outliers, it is a good idea to log transform target variable. As shown in the below figures, after log transform distribution is more similar to uniform distribution.



1. log transformed price\_doc quantiles



1. log transformed price\_doc distribution

According to correlation analysis, we saw that there are very similar and highly correlated features. So I preferred to delete below features from the data:   


There are some erronous data points when we analyze data points in detail. I also prefer to correct them:

life\_sq >full\_sq:

It is impossible that living area size can not be greater than full area size of any house. So I prefer to set living area size to NULL for satisfying records.

When I applied the rule 37 records are updated.

kitch\_sq >full\_sq:

It is impossible that kitchen area size can not be greater than full area size of any house. So I prefer to set living area size to NULL for satisfying records.

When I applied the rule 12 records are updated.

full\_sq < 5:

Full area size smaller than 5 meter square is unrealistic for any house. So I prefer to set full area size to NULL for satisfying records.

When I applied the rule 27 records are updated.

life\_sq < 5:

Living area size smaller than 5 meter square is unrealistic for any house. So I prefer to set living area size to NULL for satisfying records.

When I applied the rule 435 records are updated.

kitch\_sq < 2:

Kitchen area size smaller than 2 meter square is unrealistic for any house. So I prefer to set kitchen area size to NULL for satisfying records.

When I applied the rule 6235 records are updated.

floor > train\_df.max\_floor:

When we look at floor data, there are records where floor number is greater than max\_floor number. But it is impossible. I prefer to set these records to max\_floor values.

When I applied the rule 1493 records are updated.

num\_room == 0:

When we look at number of rooms data, there are records equal to zero. But it is impossible. I prefer to set these records to NULL.

When I applied the rule 14 records are updated.

After data record corrections, remainin all NULL values are set to median values of each column.

One-hot-encoding method is used for categorical features. For every categorical variable, a new binary variable is added for each unique value. The output is a sparse. The output will be a sparse matrix where each column corresponds to one possible value of one feature. This is widely used method since in this transformation no ordinal relationships exist but this method leads to explosion in the feature size. This curse of dimensionality leads to large training time.

## Implementation

First of all in order to record all progress and see the results/improvements at the same time, I have designed the whole project as a single Jupyter notebook.

At the beginning of the notebook, I have performed basic Exploratory Data Analysis that I have mentioned in the previous sections.

Secondly, I have started data wrangling and feature engineering part in the Jupyter notebook. I have implemented data preprocessing and correcetions in this part as I mentioned in the previous sections. This part also includes data exploration, but the difference is that when I detect some problem immediately I fix it in this part.

Most important task in feature engineering was transforming the target variable. Log-transformation made the target variable more similar to normal distribution. Also helped the reach evaluation metric since it is the log transformed version of RMSE.

I splitted the timestamp info that is the date of house sale into year and month for better ML fitting.

Since some ML algoritms can not handle NULL values, I prefered to fill NULL values with the mean of each column.

Thirdly, I passed to the modelling part. In this part, I used one-hot encoding for categorical features and I tried to fit a model with my baseline algorithm Linear regression.

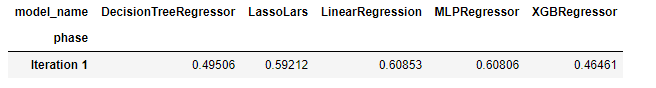
But it took so long that, I returned back to final dataframe. When I revisited the features in detail again, I realized that I did not drop correlated features and run the model with all features. So I dropped unnecessary columns and run the model again. The result and time of the model fit was reasonable.

I developed a custom function for my evaluation metric which calculates RMSE.

Then I developed another fucntion for model training and logging results into another result dataframe. Within this dataframe, I used 5-folded grid search for parameter optimizing.

When my functions are ready, I implemented trained models for Linear Regression, Lasso Lars, Decision Tree, Multi Layer Perceptron and XGBoost Regression algorithms.

Initial results were as follows:

As seen in the results, score of Linear Regresion (0.60853) is my baseline score for this project.

I wil mention about improvements and refinements in the next section of the document.

## Refinement

### Iteration 2:

In order to improve the results, I tried to find some clues within the data.

Firstly, I discovered that there were abnormal records in build\_year and state features. So fixed them as below in the second iteration:

1. state > 4:

When we look at state data, there are records other than 1,2,3,4 which are the possible conditions. But it is impossible. I prefer to set these records to NULL.

When I applied the rule 1 record is updated.

1. build\_year<1500:

When we look at build-year data, there are very abnormal values in the lower direction. There are many values as 0 or 1, which are impossible. I prefer to set build\_year<1500 records to NULL.

When I applied the rule 903 records are updated.

1. build\_year==20052009:

When we look at max value of build-year data, there is an impossible value as 20052009. It is possibly a typo error so the the actual value is either 2005 or 2009. I prefer to set this records to 2005.

1. build\_year>2017:

When we look at build-year data, there are also abnormal values in the higher direction. There are values higher than 2017 but it is impossible so that data belongs to year 2017. I prefer to set these records to 2017.

When I applied the rule 2 records are updated.

After data record corrections, remainin all NULL values are set to median values of each column. That should be done after almost every data manipulation because most of the algorithms can not deal with forgetten NULL values.

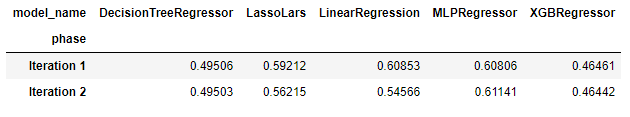
Label-Encoding: This is simple method where each unique category value is assigned an integer value. This is not preferred method the model might assume a natural ordering between categories may result in poor performance or unexpected results.

I detected boolean categorical features like ‘incineration\_raion’ and assign them 0 or 1 appropriately.

Additionaly, I label encoded ‘product\_type’ variable as 1 and 2.

These label encoding activity also, decreased number of final features after one-hot encoding process.

Iteration 2 results were as follows:



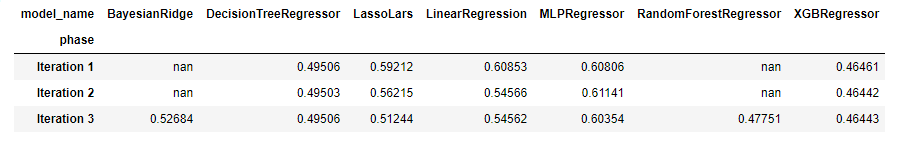
### Iteration 3:

In order to improve the results, I thought of a new variable age of building at the time of sale. So I calculated as year of build subtracted from year of sale. But I saw that there are minus aged records that means build year is higher than sale year which is impossible. So I updated these errornous records approriately and calculted the age of house again.

Another point was that, some features’ standart deviation was too high. I decided to standart scale some of features like ‘full\_sq’.

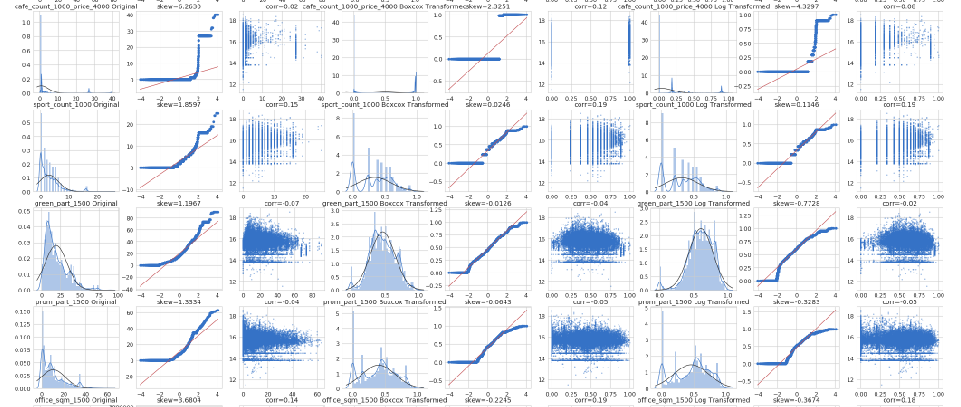
Lastly, I investigated from internet and added 2 more algorithms which are Random Forest and Bayesian Ridge Regressors.

Iteration 3 results were as follows:



### Iteration 4:

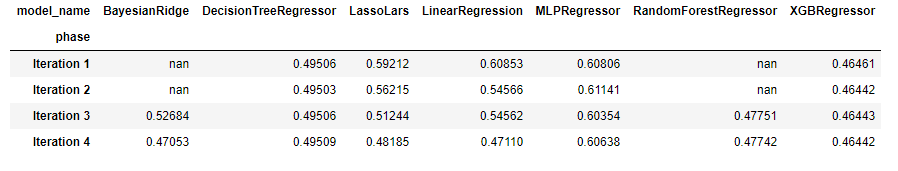
In order to improve the results, I focussed on transforming continous number features. 2 options to transform was log transform or boxcox transform. For this purpose, I prepared a script that calculates and visualizes original and transformed correlations with target sale price, skews graph and distribution graph.



1. Boxcox and log transform visualization for continuous features

As seen above when analysed the graphs in detail, I decided to boxcox trans most of the features based on skews and distributions.

Iteration 4 results were as follows:

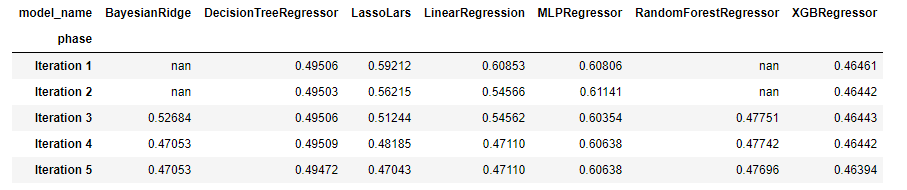


### Iteration 5:

In order to improve the results, I worked on parameter tuning in this iteration. Especially I focussed on XGBoost model parameter since it is the best scoring algorithm so far.

Regarding XGBoost, I added 2 more parameters in the grid search list that are ‘subsample’ and ‘colsample\_bytree’. Also, I have evaluted other parameters and tried to tune grid search list values of parameters.

Iteration 5 results were as follows:



# Results

## Model Evaluation and Validation

Among all models, I selected the XGBoost algorithm since it performs best in almost every iteration.

In order to test model robustness and stabilily, I rerun the final XG Boost model with different k-folds.

Results were stable and if number of folds increase the score gets betters with a small and steady percent. Table is given below:

|  |  |  |
| --- | --- | --- |
| Number of Folds / Score | Linear Regression | XGBoost |
| K=3 | 0.47654 | 0.47222 |
| K=5 | 0.47109 | 0.46394 |
| K=8 | 0.46669 | 0.45849 |
| K=10 | 0.46656 | 0.45764 |

Among all models, I selected the XGBoost of final iteration since it has the best score 0.45764.

The baseline Linear Regression model has a score of 0.60853. When I compare them, tyring other algorithms and feature engineering led to %24.80 increase in the score.

It think it is sufficient to use the model in real life.

## Justification

The baseline Linear Regression model has a score of 0.60853. Throughout the project I have performed 5 iterations and heavy feature engineering.

In each iteration, as expected ensemble methods XGBoost and RandomForest were best scoring models in general.

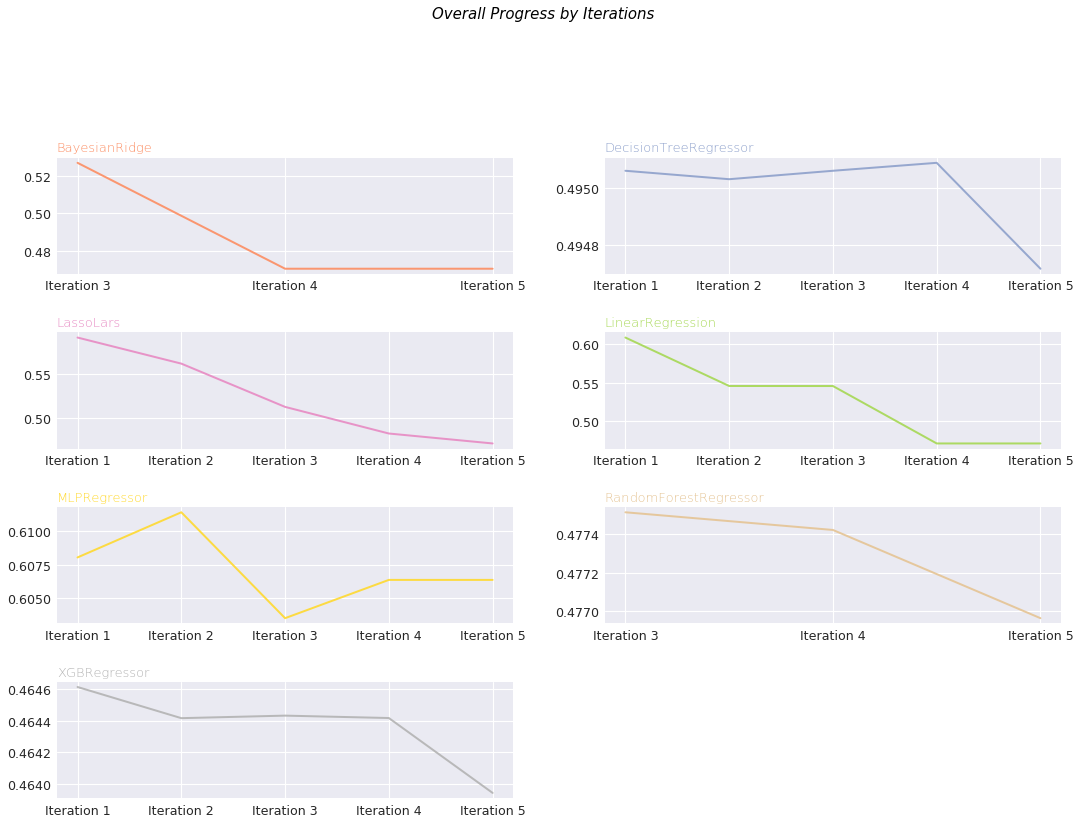
Also during iterations, Linear Regression and its similar methods LassoLars and Bayesian Ridge improve scores.

But among all models, not suprisingly final XGBoost model was the best performing with the score of 0.46394.

# Conclusion

## Free-Form Visualization

As a summary of the project, after basic EDA and feature enginering I have performed 5 basic iterations to reach the final model. In each iteration, we can say that results got better step by step as depicted below.



1. Iterations vs Algorithms vs Scores

## Reflection

The main scope of the project is to work on a real-world problem dataset. The project is a very good practical example of the importance of data pre-processing, algorithm and their optimization. Since it was a Kaggle competition, that helped me to investigate what type of solutions that others applied and get inspired from them.

Interesting Aspects:

The working experience with such a feature rich real estate data itself is an exciting task and this was one of the major motivation for selection of the project. Dealing with so mant features and evaluating one by one is was a good experience.

Challenges

The biggest challenge I had was that there was approximately 300 features so understanding them one by one took so much time. After that, I had determine correct transformation for each feature or deleting the feature from modelling set one by one That task also took so much time.

Another serious challenge posed by the project is the requirement of a good computational infrastructure. Since I had performed many iterations to improve results, making some changes and running the models again again took so much time.

Also I used 5-fold cross validation. That lead to 5 times longer time or computation power needed to implement the project.

## Improvement

In the competition, another dataset called macro.csv was also provided in which macro economic and some financial figures are included. Throuhout the project, I did not prefer to use this extra data set but these new features may be useful and can be integrated into the mplementation in the future.

Another potential improvement may be using 10-fold cross validation instead of 5-fold. In my implementation, beacuse of time constraint and running times getting higher I used 5-fold cross validation.

Xgboost implementation and algorithm can be studied in more detail. Related model could have been more complex by adding more estimators and using early stopping to prevent it from overfitting.

In the project, I used MLP implementation from scikit-learn library. Instead of this, a custom deep learning model may be implemented using tensorflow and keras. I am not sure that will lead to guarantied success since data set record size is not very big and sufficient for a deep learning model.