Real estate prices prediction

Made by Ihar Shulhan & Kevin Khanda

Data description

Data is representing 291 different parameters, which may somehow affect the final price of the real estate in Moscow. The variety of parameters is very high, which leads to data noisiness and good amount of junk values, for example like in the first row on train data.

Property with area of 1 square meter costs 15 million rubles. That's nonsense.

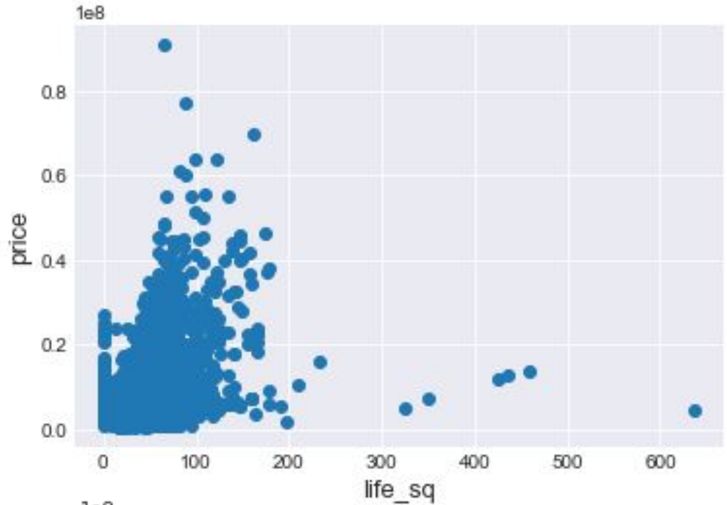
	full_sq	life_sq	floor	max_floor	material	build_year	num_room	kitch_sq	price
0	1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	15318960

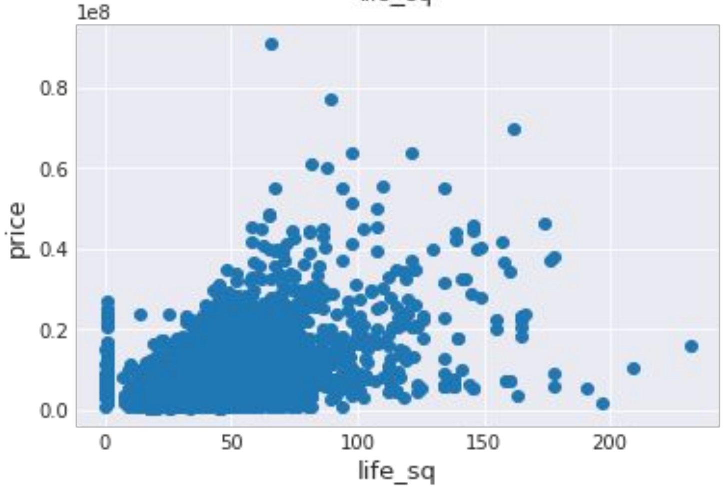
Outliers

The first thing, that we should do is to deal with outliers.

Both graphs shows the correlation between price and life area of a property. There are some outliers with life area greater than 300 square meters on the first graph.

We will just drop them.



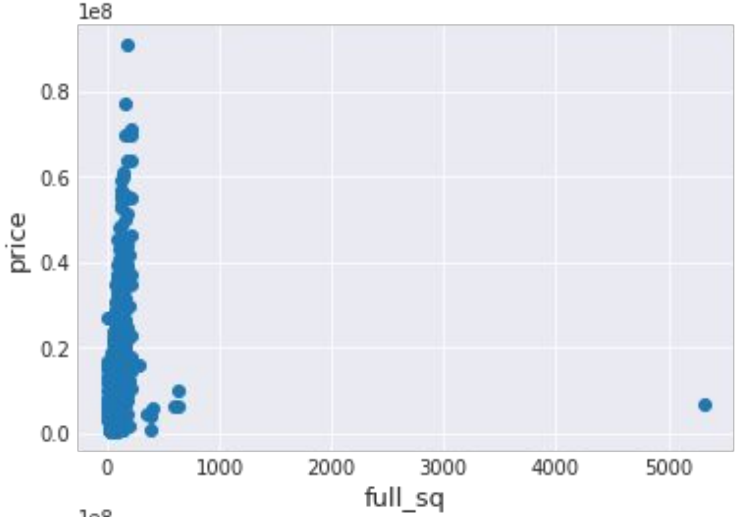


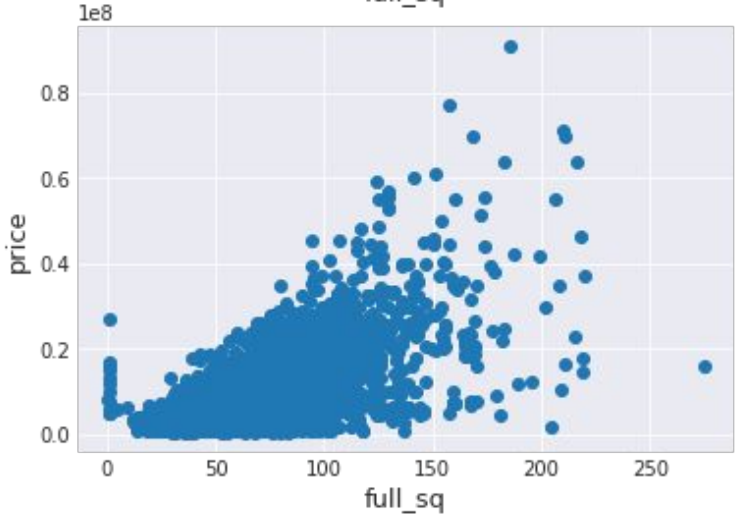
Outliers

These two graphs shows the correlation between full area and property price.

On the first graph there are some outliers where the full area is greater than 300 square meters.

We will do the same, drop them from the training set.





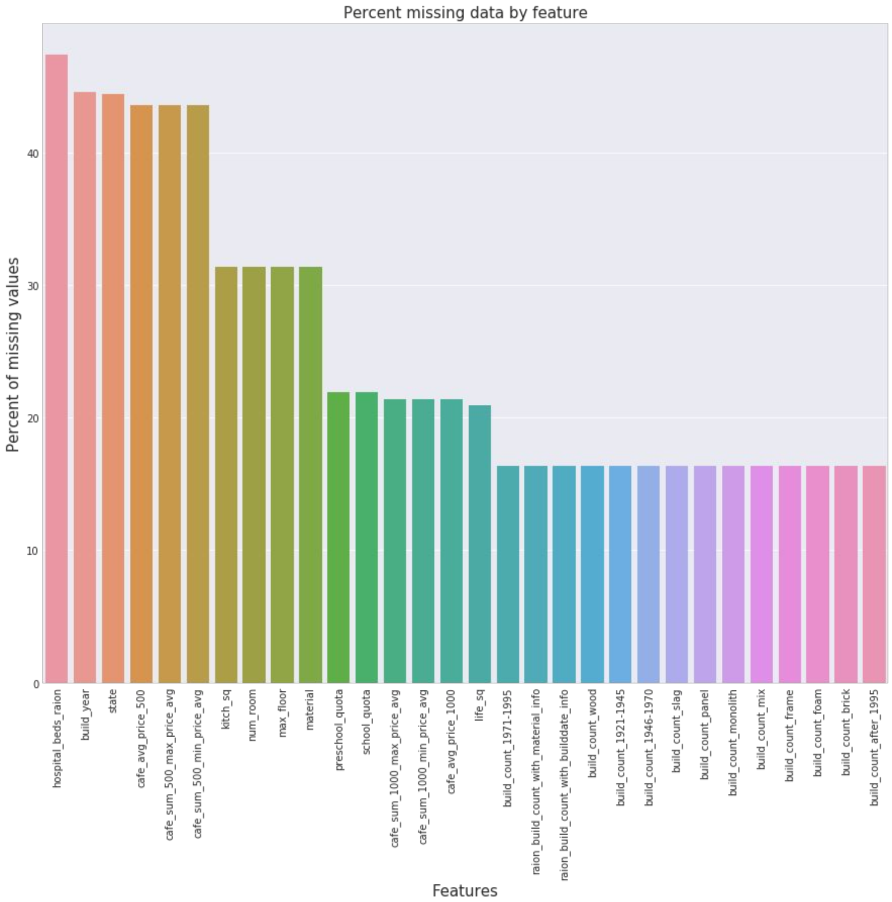
Feature preparation

Missing data

As we mentioned before, there are a lot of junk or missing data in a dataset.

On the graph percentage of missing values by columns in a whole dataset (both train and test) is shown.

We replaced empty values with mean and median of other points grouped by geographical distance.

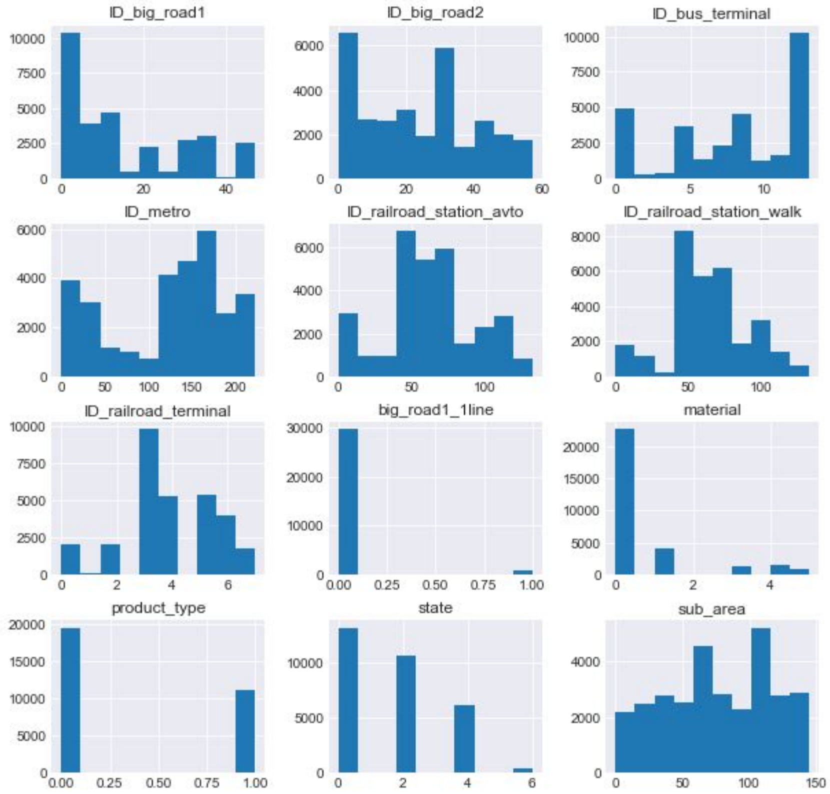


Feature preparation

Discrete data

There are 12 categorical features presented in dataset.

We have used Label Encoder in order to preprocess this data without dimensionality increase.

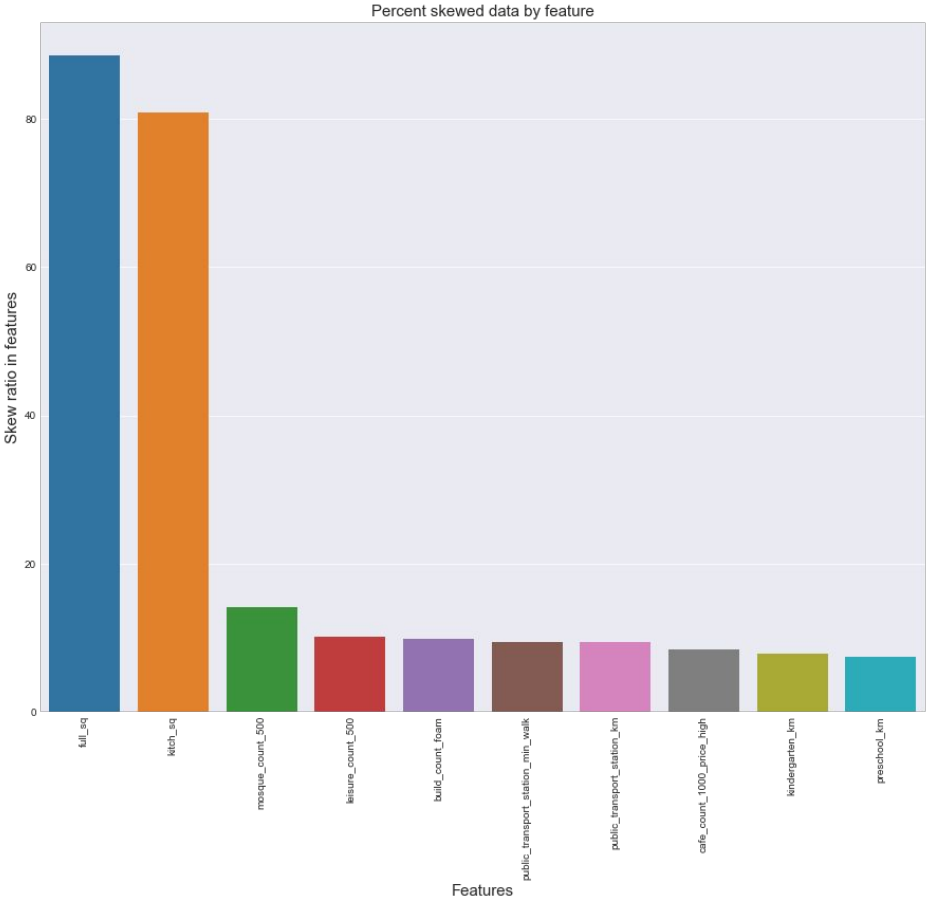


Feature preparation

Skewed data

Data in some features are slightly skewed. However, algorithms usually assume Gaussian distribution. That's why we also must deal with skewed data.

We have done it using the Box Cox transformations.



Models

Model	Error Score	Standard Deviation
KNN	2864443.4758	129349.4387
Lasso	1798894.4912	39704.0032
Elastic Net	1788020.9399	33958.0071
Kernel Ridge Regression	1665800.6071	45606.9925
XGBoost	1394365.2318	35655.3379
Neural Network (9 activation layers)	1367997.3056	37969.3186
Gradient Boosting Regression	1273835.5522	33283.3347
LightGBM	1236379.6835	31888.2631
Ensemble model (Averaging between NN, GBR & LightGBM)	1234194.5062	34013.4975

Models parameters

Model	**Best score parameters**
KNN	n_neighbors=15, weights='distance'
Lasso	alpha=0.005, max_iter=2000
Elastic Net	alpha=0.005, l1_ratio=0.5
Kernel Ridge Regression	alpha=0.4, kernel='polynomial', degree=2, coef0=2
XGBoost	colsample_bytree=0.7, gamma=0.01, learning_rate=0.05, max_depth=10, n_estimators=500, reg_alpha=0.7, reg_lambda=0, subsample=0.8
Neural Network (9 activation layers)	’relu’ on each activation layer, loss=’mean-absolute_error’, optimizer=Adam(), 481 neurons, batch_size=512, verbose=1, validation_split=0.1, epochs-600
Gradient Boosting Regression	n_estimators=112, max_depth=9, min_samples_leaf=25, max_features=47, loss='lad'
LightGBM	objective='mean_absolute_error', num_leaves=10, learning_rate=0.05, n_estimators=1400, bagging_fraction = 0.8, bagging_freq = 3, feature_fraction = 0.8, min_data_in_leaf=40
Ensemble model	Equal averaging between NN, GBR & LightGBM with their best score parameters

Best Model

LGBM was the final model we chose, as it provided the best results both on CV (1.234 * 10^6) and on LB (1.279 * 10^6). The important parameter of the model is to set objective to mean absolute error, which decreases error almost by 2 * 10^5.