A comparison of machine learning approaches to predict Russian housing market prices

Hubert Jenq

Abstract

Sberbank, a Russian bank is interested in predicting the house price of homes in Russia. They have provided a total of 38,113 records with 395 covariates which include housing characteristics, housing area characteristics, and macroeconomic data. The goal is to predict the housing price most accurately measured by the root mean log squared error. In tackling this problem, the data is first looked at and issues such as missing data and feature selection are addressed. Then three types of models are used: regularized linear models, random forest, and gradient boosted trees. The model parameters are optimized by random search. Gradient boosted trees performed the best with a validation error of 0.312. A linear stacking model is also implemented combining the three types of models but performs poorly with a validation error of 0.348.

Introduction

Housing is one of the largest expenses for individuals. Thus, the ability to accurately assess the value of a home would be extremely beneficial. Sberbank, a Russian bank has posted their housing data on the data science competition website www.kaggle.com in hopes of obtaining an accurate housing price forecasting solution to help its customers have more confidence when they lease or purchase a building. The data provided includes both housing data and macroeconomic patterns of Russia during the time period where the houses were sold. The

metric for evaluation is root mean log squared error (RMLSE) for the predicted vs. the actual price. The formula is as follows:

$$\epsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i + 1) - \log(a_i + 1))^2}$$

where n is the number of test samples, i indexes record, p_i is the predicted price, and a_i is the actual price, and ε is the error. In this report, I will attempt to create an optimal model by the following steps. 1. Data overview which involves looking at housing characteristics, missing values, missing value imputation, data errors, and feature generation and selection. 2. Model selection with various methods and an attempt of model stacking, 3. Results, and 4. Discussion.

Housing Characteristics

Before working with the data, I would like to get a better overview of how the variables are related with each other. The data provided is broken into a training and test set. The training set has 30,471 records from August 2011 to June 2015. The test set has 7,662 records from July 2015 to May 2016. There is accompanying Russian macroeconomic data for each of the records in that time period. The housing data contains 291 covariates which are broken into two categories: housing characteristics and the housing area characteristics. There is a total of 12 housing characteristics and 279 housing area characteristics. The macroeconomic data includes 100 covariates that include various inflation rates, exchange rates, mortgage rates, and more. Below is a Table 1. which gives the definitions of the housing characteristics.

House Characteristics	**Definition**
timestamp	Time of transaction
Full_sq	Total living area in square meters

Life_sq	Living area in square meters
floor	Floor of the building
Max_floor	Number of floors in the building
material	Wall material
Build_year	Year built
Num_room	Number of rooms
Kitch_sq	Kitchen area
state	Apartment condition
Product_type	Owner occupied purchase or investment
Sub_area	Area of district

Table 1. Housing Characteristics

To look at the distribution of housing prices, a histogram is plotted below in figure 1. The house price looks like it has a little bit of right skewness so a log transform is applied to help normalize the data.

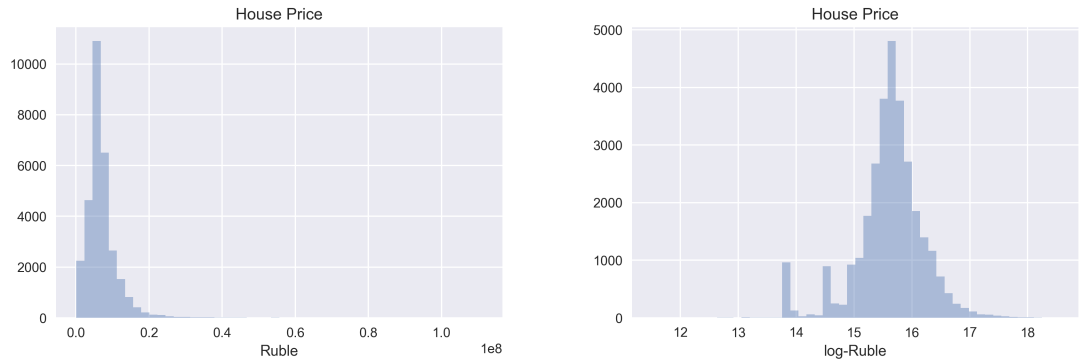

Figure 1. Histogram of house price in Rubles and log-Rubles.

The log transform looks to be effective in making the house price to appear more normal. There are two anomalies in the house price around 13.8 and 14.5 log-rubles where there are nearly 1000 instances. Upon further investigation, all of these housing prices correspond to exactly 1 million and 2 million rubles. The housing sale types are investment and not owner-occupied purchase which leads me to believe that they were under-reported prices in order to evade taxes.

The next step is to examine how the housing features are linearly correlated with the housing price. A correlation plot is shown below in figure 2. to look at the correlation coefficients.

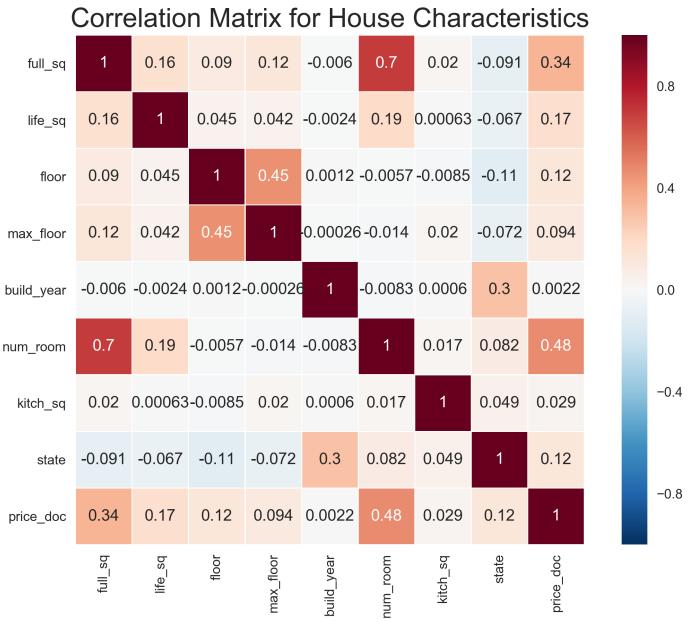

Figure 2. Correlation matrix of Housing Characteristics

Here we see that the highest correlation with price of the house is with the number of rooms at 0.48, and then full square footage at 0.34. The correlation between the housing characteristics themselves seem relatively uncorrelated except for full square footage and number of rooms. From looking at the correlation, it appears that a linear model using few of the variables is unlikely to predict the price of the home accurately.

To examine relationship between housing price full_sq and life_sq we plot them against the log price_doc with a LOWESS curve in figure 3. below.

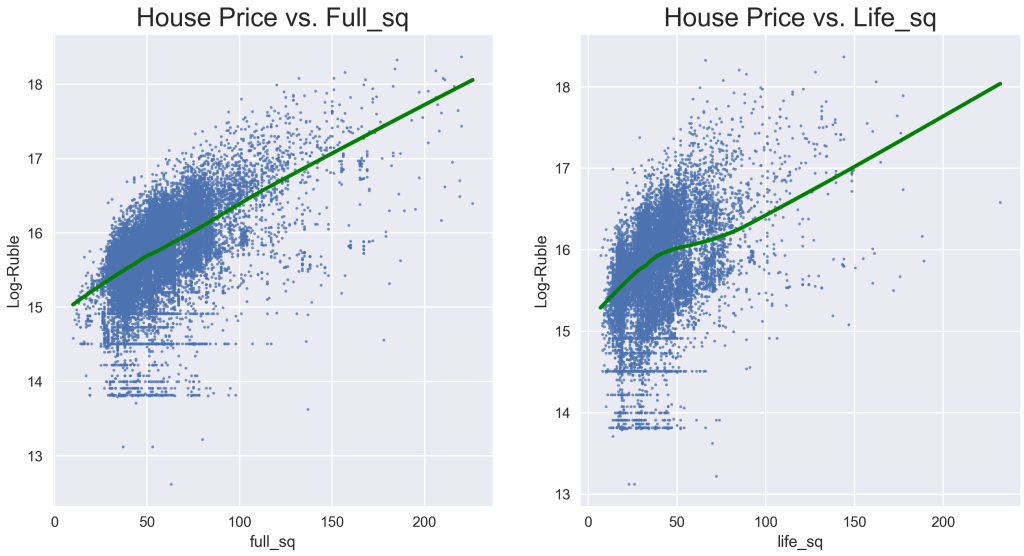

Figure 3. Scatterplot of house price in log-rubles vs. full_sq and life_sq

We see that the life_sq LOWESS curve has a hump in it around 40 square feet whereas the full_sq LOWESS curve is linear throughout. This could account for the 0.34 correlation vs. 0.17 between the house price and full_sq, life_sq. Despite a linear LOWESS curve, the housing price is still quite variable for the full_sq and a simple linear model would perform quite poorly.

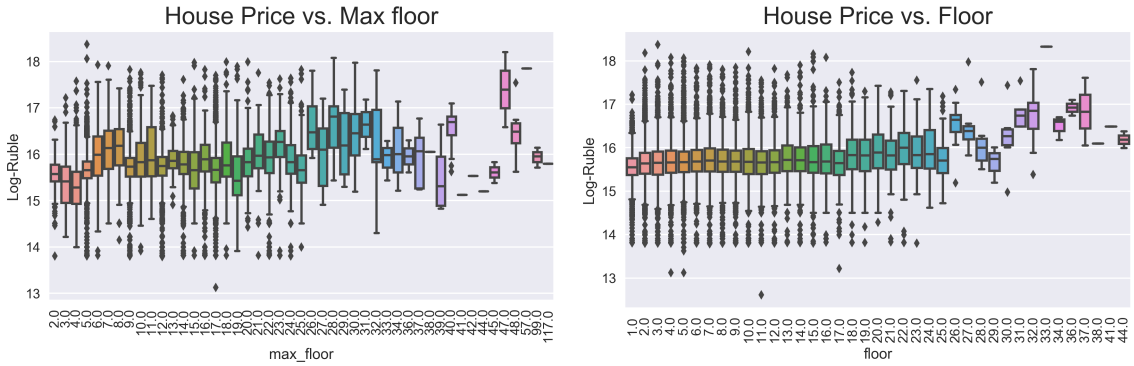

Figure 4. Boxplot of house price vs. max floor and floor

Looking at the housing price vs max floor and floor number in the figure 4. above we do not see a significant linear trend which is consistent with the low correlation numbers of 0.09 and 0.12 respectively. However, there are nonlinear fluctuations in the median housing price with the floor number. We see that the median housing price is higher for those higher than floor 26. This points towards using a model that can account for nonlinear relationships.

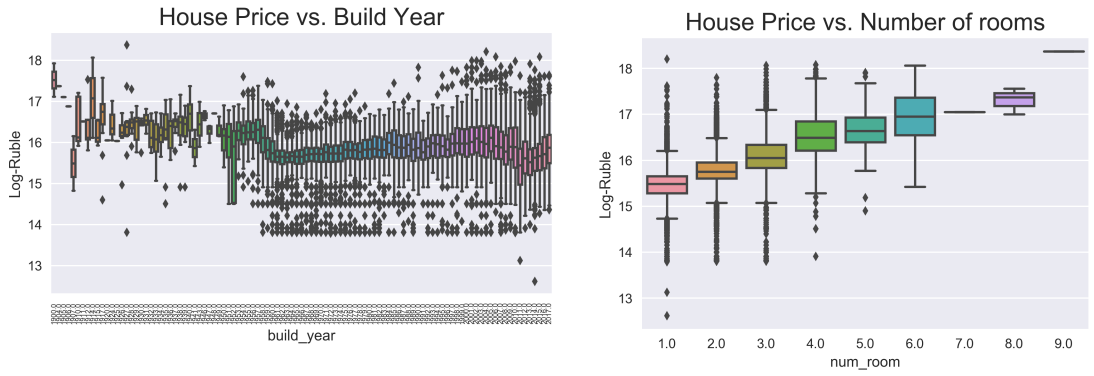

Figure 5. Boxplot of house price vs. build year and number of rooms

In figure 5. we see that housing price by build year also shows that it has a non-linear relationship with older buildings having a higher median price. The number of rooms seems to have a linear relationship with housing price but there are a large amount of outliers in the lower number of rooms from 1-4 as seen in figure 5. This was counter intuitive but looking into the older build year buildings showed that they had higher number of rooms indicating they may be entire buildings rather than individual homes or apartments.

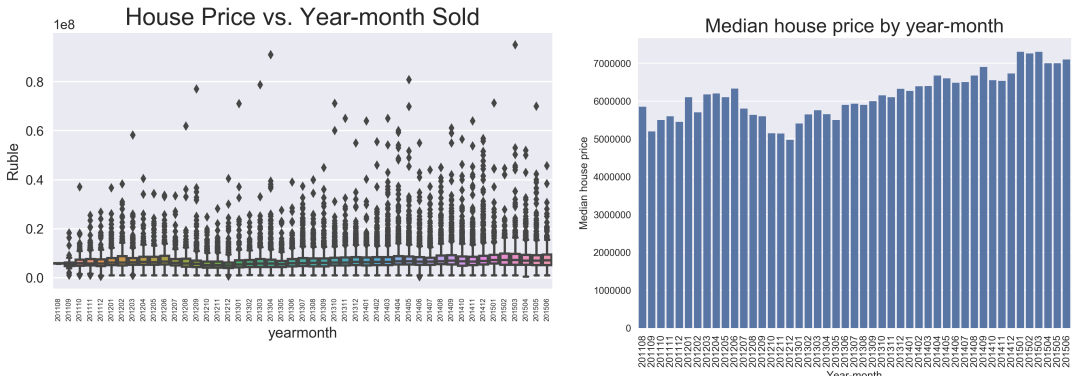

Figure 6. House price vs. time

Finally the housing price compared to the year-month sold is plotted with a box plot, and a bar grpah using only the median house price in figure 6. We see that there is an increasing trend for higher median housing prices for the later years which could be due to inflation or an increased demand on housing.

From examining how the housing characteristics relate with each other, we see that the housing characteristics are mostly non-linearly related to the housing price so using a linear model may be difficult.

Missing Values.

The dataset given by Sberbank involved many missing values and determining how to deal with the missing data will important for the final model. The amount of missing values are plotted below in figure 7.

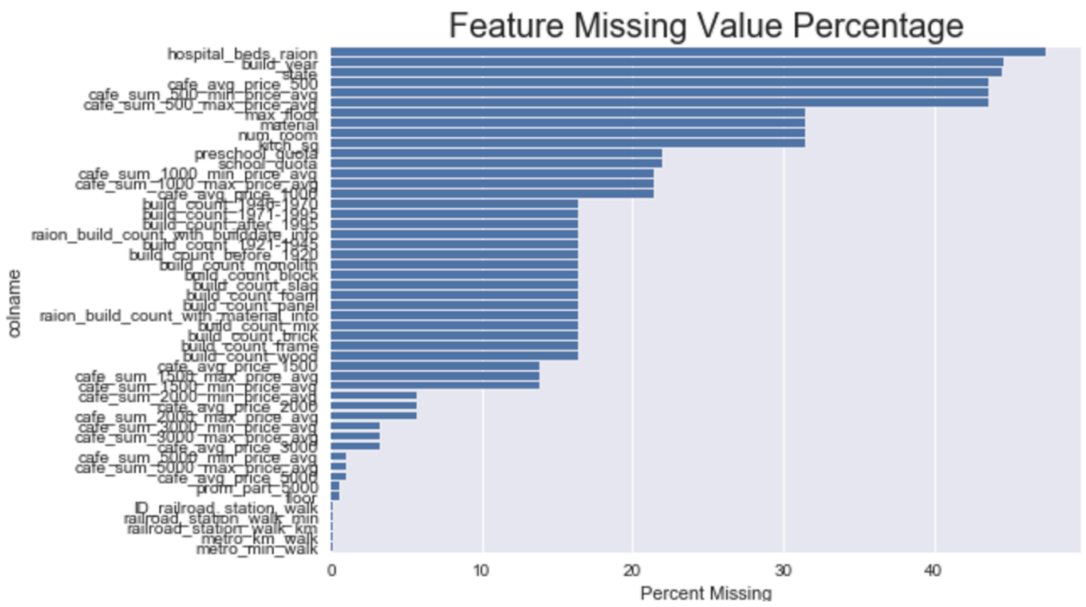

Figure 7. Feature Missing value percentage

Several important housing characteristics like build year, max floor, material, number of room, and kitchen square footage have over 20% missing values. To further examine the missing value trend the number of missing characteristics is plotted against time in year-month in figure 8. We see that there is a sharp drop in missing house characteristics in October 2013, and there is a large increase in missing housing area characteristics past September 2012. The opposing trends for missing characteristics is bizzare and should be noted for future analyses.

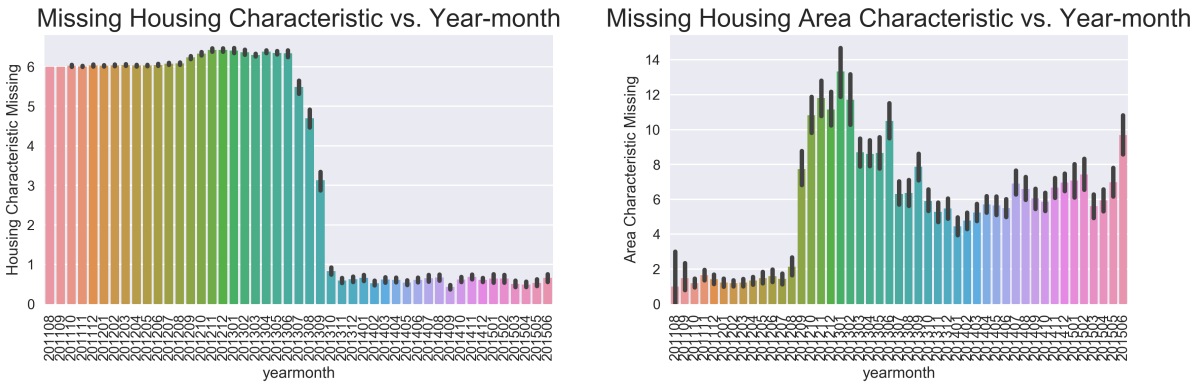

Figure 8. Missing value temporal trends

Missing Data Imputation

Due to the abundance of missing values, the missing values will be imputed instead of dropping all cases with missing values which would remove a large percentage of the data. Two methods of data imputation will be applied to create three datasets total: data with missing values, data with median imputation, and data with K-NN imputation. A total of 1,070,908 missing values will be imputed. The first method of median imputation will be applied across all covariate filling in the missing values with median of the covariate. This is done separately for the training and test set to prevent any information leakage. The second method of imputation is done by K nearest neighbors clustering. The covariates are split into housing characteristics, and area characteristics. For each missing value, the value is filled in by the average of the three closest records based on normalized Euclidean distance. The missing values for a single record are filled in before moving onto the next record. The records are filled in from the past towards the future. Shuffling the records will result in a different K-NN imputation since it will change the order of values that are imputed.

Feature Generation

Additional features are generated from the covariates based on intuition in hopes to improve predictive performance. The building age will be converted to house age by taking 2017-build_age. The number of properties sold in the month will be counted with the belief that the price for a house will decrease the more available houses are. Floors from the top is generated from max floors minus the floor number which may be more meaningful than absolute floor number since being on the highest floor of a building is desirable. Similarly, relative floor

is calculated from the floor divided by the maximum floor number. The average room size is calculated from life square footage divided by number of room since more expensive homes are more likely to have large rooms compared to many rooms. The percentage of kitchen is calculated from the kitchen square footage divided by the full square footage. Finally the number of null values for the record is summed since it may be related to the quality of the listing. The variables created are house age, properties sold in the month, floors from top, relative floor, average room size, kitchen percentage, and null values. A summary of the features generated are shown in the table 2. below.

Feature	Definition
House Age	2017 – build age
Number of properties sold in month	Count of houses sold in the month
Floors from top	Max floor - floor
Relative floor	Max_floor/floor
Average room size	Life_sq/number of rooms
Kitchen Percentage	Kitchen square footage/full square footage
Null values	Number of NA's

Table 2. Generated features

Feature Selection

The combination of the housing characteristics, housing area characteristics, macroeconomic data, and features generated yield a total of 398 features. Many of these features are correlated with each other and would be redundant to input the combination of them into the model. To determine which ones are multicollinear, a method of feature selection using the variance inflation factor is employed. The variance inflation factor(VIF) for each continuous covariate is calculated in the covariate matrix. The covariate with the highest VIF is removed and then the VIF is calculated again for all of the covariates. This is iteratively done until all covariates are dropped and have a corresponding VIF of when they were removed. This method was performed on macroeconomic data only since there may be less of a relationship between

housing price and macroeconomic data compared to the housing characteristics. Using a VIF cutoff of 10 for the macroeconomic data yielded 11 covariates that are shown in the table 3. below.

Macroeconomic Variable	Definition
Balance_trade	Trade surplus
Balance_trade_growth	Trade balance (% of previous year)
Eurrub	Ruble/EUR exchange rate
Average_provision_of_build_contract	Provision of orders in Russia
Micex_rgbi_tr	MICEX index for government bonds (MICEX RGBI TR)
Deposits_rate	Average interest rate on deposits
Mortgage_value	Volume of mortgage loans
Mortgage_rate	Weighted average rate of mortgage loans
Income_per_cap	Average income per capita
Apartment_build	City residential apartment construction
Museum_visitis_per_100_cap	Number of visits to museums per 1000 of population

Table 3. Macroeconomic covariates with VIF<10

Data errors

The data received from Sberbank is ridden with errors. The following errors that were caught from looking at summary statistics were fixed but many errors may still exist. For example, there are many instances where the life_sq which is the square footage of living space, is greater than total_sq which is the total square footage of the area. For the records where life_sq>total_sq, the life_sq is replaced with the value of full_sq. If the kitchen square footage took up more than 90% of the full_sq, the kitchen square footage was set as zero. Similarly, if the kitchen was less than 3 square meters, it was set as zero since it is unlikely that the home has a kitchen. In the buildings where the floor number was greater than the max floor number, the values were set as NA for both the floor and max_floor. Max floor and floor numbers of 0 were set as 1.

Model selection

From the data overview above, we see that many of the variables have nonlinear relationships between the covariates and outcome variable. Three modeling methods are chosen: linear models, random forest, and gradient boosted trees.

Model evaluation

To evaluate the model, the training set containing 30,471 records is further partitioned into a model training set with 24,377 records and a validation set with 6,094. This is a random 80/20 split of training/validation set of the original training data. The models will be trained on the 80% of the data, and validated on the last 20%. If 5-fold cross-validation is used to optimize model parameters, the model is 5-fold cross validated on the 80% of the data (24,377) records. The 20% set is used as the validation set to prevent overfitting of the model to the 80% of the data.

Linear Models

Ordinary least squares regression, ridge regression and LASSO models are trained on the median imputed and K-NN imputed datasets. These models are trained on the 24,377 records. The categorical variables are changed to dummy variables. The covariates are also normalized for the ridge regression and LASSO model to correctly penalize the coefficients. For the regularized models a 5-fold cross validation on the 24,377 records was done to obtain the optimal penalization parameter alpha. A grid of 1000 alphas were chosen to find the optimal value. The results of the optimal alpha, training error and validation error for the six models are shown in the table 4. below.

Regression Method	Imputation Method	Alpha	Training Error	Validation Error	Train-Val Diff
OLS	Median	-	0.324	0.329	-0.05
	K-NN	-	0.324	0.328	-0.04
Ridge	Median	0.004	0.325	0.328	-0.03
	K-NN	0.003	0.324	0.327	-0.03
LASSO	Median	1.85e-6	0.327	0.328	-0.01
	K-NN	1.86e-6	0.326	0.327	-0.01

Table 4. Training errors for linear models

We see that the OLS has the lowest training error and the highest validation error which is consistent with the overfitting of the training dataset due to the lack of regularization. LASSO and Ridge regression both perform equally well on the validation set however the discrepancy between the training and validation error is greater in the ridge regression indicating that it is likely more overfit than the LASSO model. This may be because LASSO is capable of zeroing coefficients since it the penalty is L1 instead of L2. We see a slight improvement of 0.01 across all validation and training errors for the K-NN imputed data compared to the median imputed data pointing towards K-NN being the superior imputation method.

Random Forest

The next model implemented is random forest which should be able to deal with the nonlinearities and possible interactions among the covariates in the dataset. Parameters that are important for tuning the random forest algorithm are the number of trees grown, the number of features to select from at each split, maximum tree depth, and minimum samples per leaf node. To first determine the number of trees necessary, starting parameters of 33% of features used per split, 5 minimum samples per leaf node, max tree depth of 8 is used and the cross-validation error is calculated based on the 24,377 records as a function of number of trees. The error is shown as a function of number of trees is shown below.

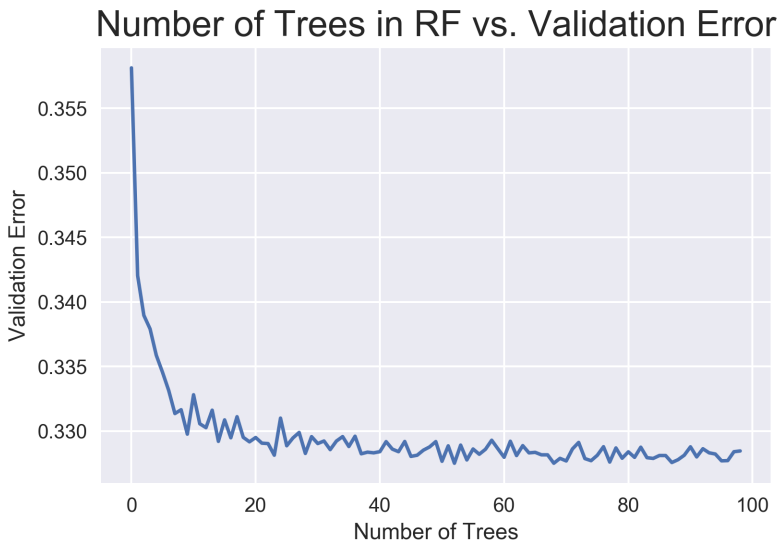

Figure 9. Line plot of Number of trees vs. validation error

Here we see that the validation error stabilizes around 60-80 trees. To ensure the forest validation error has stabilized, we will use 100 trees. With the initial parameters and 100 trees, the forest has a validation error of 0.315. The K-NN imputed dataset is chosen over the K-median imputed dataset since it offers an decrease of 0.002 in the initial model in its 5-fold cross validation error. To optimize the rest of the parameter optimization, 5-fold cross validation on the training set of 24,377 records is done to obtain the best set of random forest parameters. The validation error is calculated on the remaining validation set of 6,094 records. The parameter search space is specified in the table 5 below. Parameters were searched randomly.

Variable	Definition	Search Space	Optimal value
N_estimators	Number of trees	100	-
Max_features	% of features at each split	Unif(.2,.45)	0.341
Min_samples_leaf	Minimum leaf size	1,2,5,10	5
Max_depth	Maximum tree depth	5,8,15,30	30

Table 5. Random forest parameters

The resulting validation and training error after the random forest hyperparameter optimization is 0.314 and 0.224. This resulted in a 0.001 reduction in validation error from the initial parameter

set. The max_depth parameter was the only one to significantly increase from 5 to 30 after optimization. For the random forest, the optimization of parameters did not seem to improve the validation score much. There is also a large discrepancy between the validation and training error which indicates that the initial model may be overfit. This is consistent with the large max tree depth parameter of 30. The forest would benefit from an increase in number of trees to reduce the variance for an increase in bias since the initial number of trees was determined by a max depth of 5. Feature importance from the random forest is shown in figure 10 below.

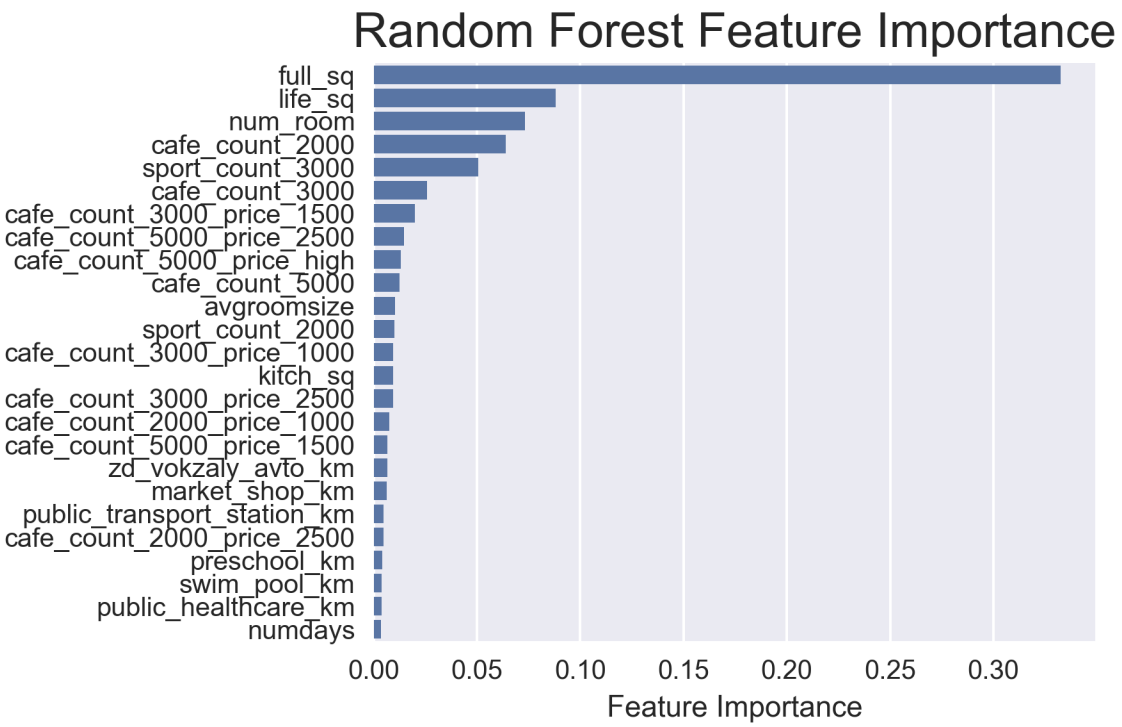

Figure 10. Random forest feature importance

Higher values of feature importance correspond to a larger out of bag error when the values of the feature are permuted. Here we see that full square footage is the most important followed by life square footage. The feature average room size which was generated is also one of the more important features. Numdays corresponds to the number of days since 2011 which also indicates

that the time the property was sold is important in determining its price. Café_count_3000_price_1000 can be interpreted as the number of cafes within d-meters that have an average bill under price p. The feature importance list is dominated by the café_count variable which seems to heavily tied to the housing price of the region.

Gradient Boosted Trees

The final method used is gradient boosted trees which is also an ensemble method of classification and regression trees. The variant of gradient boosted tree used is XGBoost which includes regularization in the tree growing process. The unimputed data is used for the XGBoost model since it is able to handle NA's by assigning all NA values to the split that minimizes error. There are three categories of XGBoost parameters that are important to optimize. The data parameters involve the subsample which is the percentage of original sample to use per tree, and the colsample which is the number of features for each split. The regularization parameters include alpha and lambda which are the L1 and L2 penalization parameters, Eta which is the learning rate for each additional tree and gamma which is the minimum loss reduction for the next tree split. The tree parameters are the same as in random forest which are the maximum tree depth, the number of trees, and the minimum number of samples in a leaf node. The XGBoost parameters are summarized in table 6 below.

Type of Parameter	**Parameter**		**Search space**	**Optimum values**
Data	Subsample	% of samples for each tree	0.6, 0.7, 0.8, 0.9, 1	1
	Colsample	% of features at each split	0.5, 0.6, 0.7, 0.8, 0.9, 1	0.5
Regularization	Eta	Tree learning rate	0.01,0.015,0.025,0.05,.1	0.05

	Alpha/Lambda	L1/L2 penalization parameter	0,0.01,0.05,.1,1	Alpha=0, lambda=0.01
	Gamma	Minimum loss reduction for next partition	0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1	0.7
Tree	Max_depth	Depth of tree	3,5,7,9,12,15,25	5
	N_estimators	Number of total trees	244	-
	Min_child_weight	Minimum number in leaf node	1,2,3,4,5,6,7	7

Table 6. XGBoost parameters

Similar to the random forest, the number of trees, n_estimators will be first determined based on minimizing the cross-validation error. The number of trees is determined when there has not been an improvement in the cross-validation error in 20 iterations. Below in figure 11 the number of trees is graphed against the validation error.

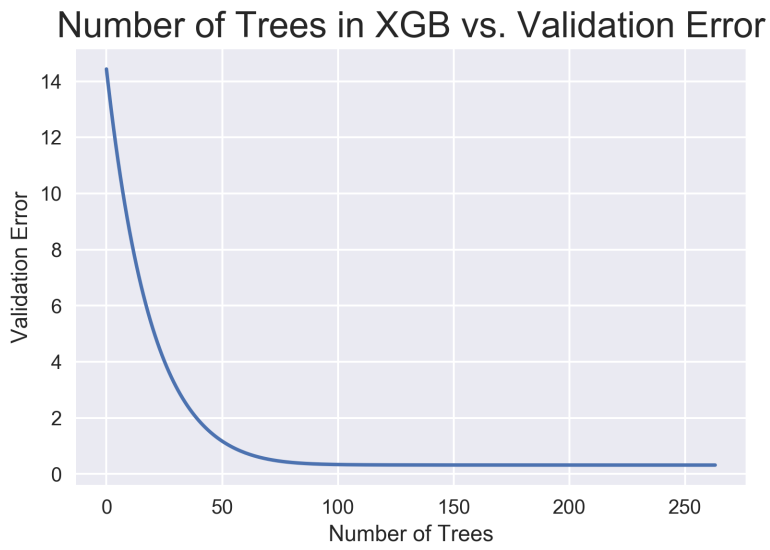

Figure 11. XGB number of trees vs. validation error

We see that the error stops dramatically dropping at about 100 trees, but is still making small improvements until the optimal number of trees n_estimators was determined to be 244 with a validation error of 0.316. The initial parameter settings of eta=0.05, max_depth = 5, subsample=1, colsample = 0.7, gamma=0, min_child_weight = 1, alpha=0, lambda=1. A random search in the XGB parameter space shown in TABLE above is done to optimize the xgb model parameters. 1000 models were generated, each with a random sample of parameters. The optimal model parameters are shown in the TABLE above and had a corresponding validation error of 0.312 and a training error of 0.271. The discrepancy in the training and validation error still indicates that there is some overfitting happening but not as much as the previous random forest models.

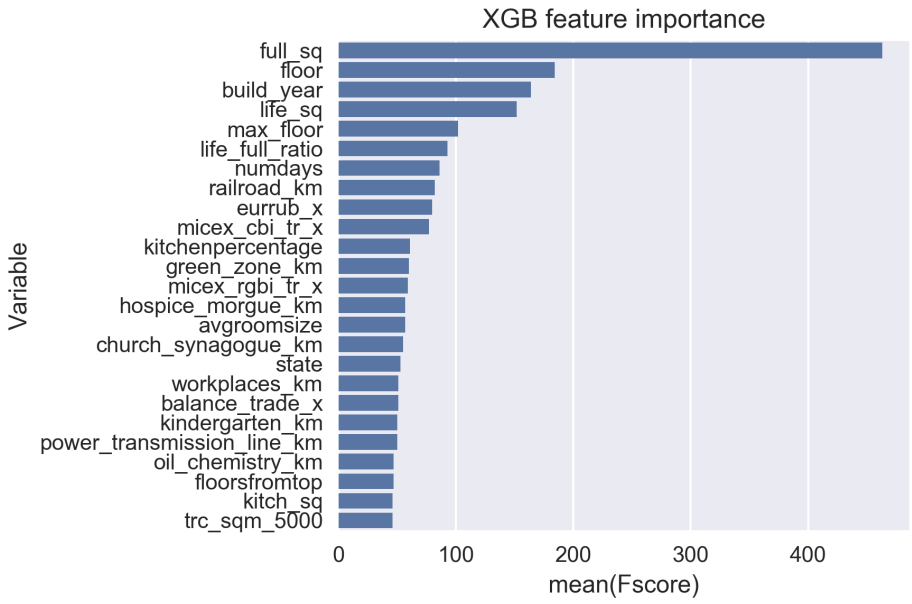

Figure 12. XGB Feature importance

Comparing the feature importance of the random forest and XGB model we see some overlap as both full_Sq and life_Sq are at the top. However, there are quite a few differences such as floor and max_floor are important in XGB but not random forest. The café_count variables are not to

be found in the XGB model. Here we see the generated features kitchenpercentage, floorsfromtop, and avgroomsize are in the top 25 most important variables.

Model Stacking

In an attempt to improve and combine the strengths of individual models, the outputs (predicted house price values) of the Ridge Regression, XGB, and random forest models are used as features in a linear model. The three models are 5-fold cross-validated on the training set spanning 24,377 records with the optimal parameters that were achieved from the previous optimizations. The predicted values for the 5-fold validation step are stored so we have 24,377 predicted values based on models that did not include the record it is predicting. These predicted values are used as the new feature set for the linear stacking model. The linear stacking model is trained on the three models predictions of the 24,377 records. The resulting coefficients of the model are 1.38 for ridge regression, 0.27 for xgb, and -0.61 for the random forest model. We can interpret these coefficients as the weighted average of the models. The training error for the linear stacking model is 0.207 which is much lower than any of the three individual model training errors. However, the validation error is 0.348 which is larger than any of the three individual models.

Results

The results of the models run is summarized in the table 7 below. XGBoost model performed with the best validation error of 0.312. Although XGboost model is regularized, it seems to be more overfit relative to the LASSO model since the training-validation error difference is larger. The random forest was the most overfit model with a difference of -0.09. The linear stack of the

ridge regression, random forest, and XGboost did not perform well with a validation error of 0.348.

Regression Method	Imputation Method	Training Error	Validation Error	Train-Val Diff
OLS	Median	0.324	0.329	-0.05
	K-NN	0.324	0.328	-0.04
Ridge	Median	0.325	0.328	-0.03
	K-NN	0.324	0.327	-0.03
LASSO	Median	0.327	0.328	-0.01
	K-NN	0.326	0.327	-0.01
Random Forest	K-NN	0.224	0.314	-0.09
XGBoost	None	0.271	0.312	-0.041
Stacked Linear model	n/a	0.207	0.348	-0.141

Table 7. Model Error Summary

Discussion:

The dataset provided by Sberbank involved a large amount of missing values. The K-NN imputation seemed to have performed better than the median imputation based on the validation error across all models. Due to the large amount of missing values, future efforts should be placed into imputation. Methods to try can involve using geographical location to impute missing features and to determine unique building ID's to impute the rest of the missing buildings features. We also observed a temporal pattern in the missingness of the features. It may be more effective to use the data of the later years to train the model since they are missing fewer housing characteristics which fall on the top of the variable importance list.

Feature selection in this case was done by iteratively removing features with high VIF. The large amount of covariates in this dataset may benefit from other methods such as PCA, weighted least squares or selection by feature importance from random forest.

The random forest and XGBoost models cannot account for any values that are not in the training dataset. From the data overview, we see that the median house price is increasing with time. As a result, the tree-based models are incapable of accounting for this trend. The linear models such as LASSO and Ridge regression have the opposite problem, they are incapable of accounting for any variable interactions or nonlinear trends. With these two shortcomings, we see that the tree-based models have a better validation error compared to the linear models. The linear stacking model combining the ridge regression, XGBoost, and random forest performed worse than any of the three individual models. The discrepancy between the training and validation error suggests that the solution generalizes poorly and needs to be more robust. The individual models may be trained with a larger regularization penalty which may improve the stacking model. Alternatively, a nonlinear stacking model may be implemented with better success.