Udacity Machine Learning Nanodegree

Capstone Project

Catherine He
May 2017

I. Definition

A. Project Overview

In this problem, Sberbank (Russia's oldest and largest bank) aims to help their customers by making predictions about realty prices so renters, developers, and lenders are more confident when they sign a lease or purchase a building. Building an accurate forecasting model will allow Sberbank to provide more certainty to their customers in an uncertain economy; housing costs demand a significant investment from both consumers and developers. And when it comes to financial planning—whether personal or corporate—the last thing anyone needs is uncertainty when such huge sums are involved.

Personally, I am interested in taking on this project because I have always been interested in Real Estate and would like to understand what are the major drivers of Real Estate prices to make better decisions. Afterall, for most people, Real Estate is one of the biggest investment we will make in our lives.

B. Problem Statement

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 project, I will preprocess the data, pick out the important features, and run them through models (Random Forest, Gradient Boosting, XGBoost, Deep Neural Networks, including stacked models) to get predictions for the label (price of the unit) of the test set.

C. Metrics

To measure the effectiveness of my models and pick the best performing one, I used a combination of R-squared, explained variance score, Mean squared error (MSE), and Root Mean Square Error (RMSE). Each metric and the reasons for using them are highlighted below:

1. R-squared/explained variance score:

R-squared and explained variance scores are regression score functions, and they tell us how well the features of a model can predict the labels, with 1.0 being the best possible score. I used this because I wanted to know how relevant the features were in predicting the prices.

2. Mean squared error (MSE):

MSE measures the average of the squares of the errors or deviations. MSE is a measure of the quality of the estimator; the closer the values to zero, the better the quality. I wanted to see how well each model was in estimating the given data.

3. Root Mean Squared Error (RMSE):

RMSE takes the square root of the mean squared error. The use of RMSE is very common and it makes an excellent general purpose error metric for numerical predictions. Further, it is a measure of accuracy, as it compares forecasting errors of different models for the same features, which is what I am aiming to do here.

II. Analysis

A. Data Exploration

To explore the data, I first found out the size (rows x columns) of the datasets we were given. The train set had 30,471 rows and 291 features (excluding label), while the macro data set had 2,484 rows and 100 features. The macro set would not be sufficient on its own but could give crucial information on the Russian economy affecting the housing market, so I decided to combine both. This would also give the models more information to work with, giving me a resulting data set of 30, 471 rows and 391 features. Here is a sample of the first few rows and columns of the data as well as their descriptive statistics:

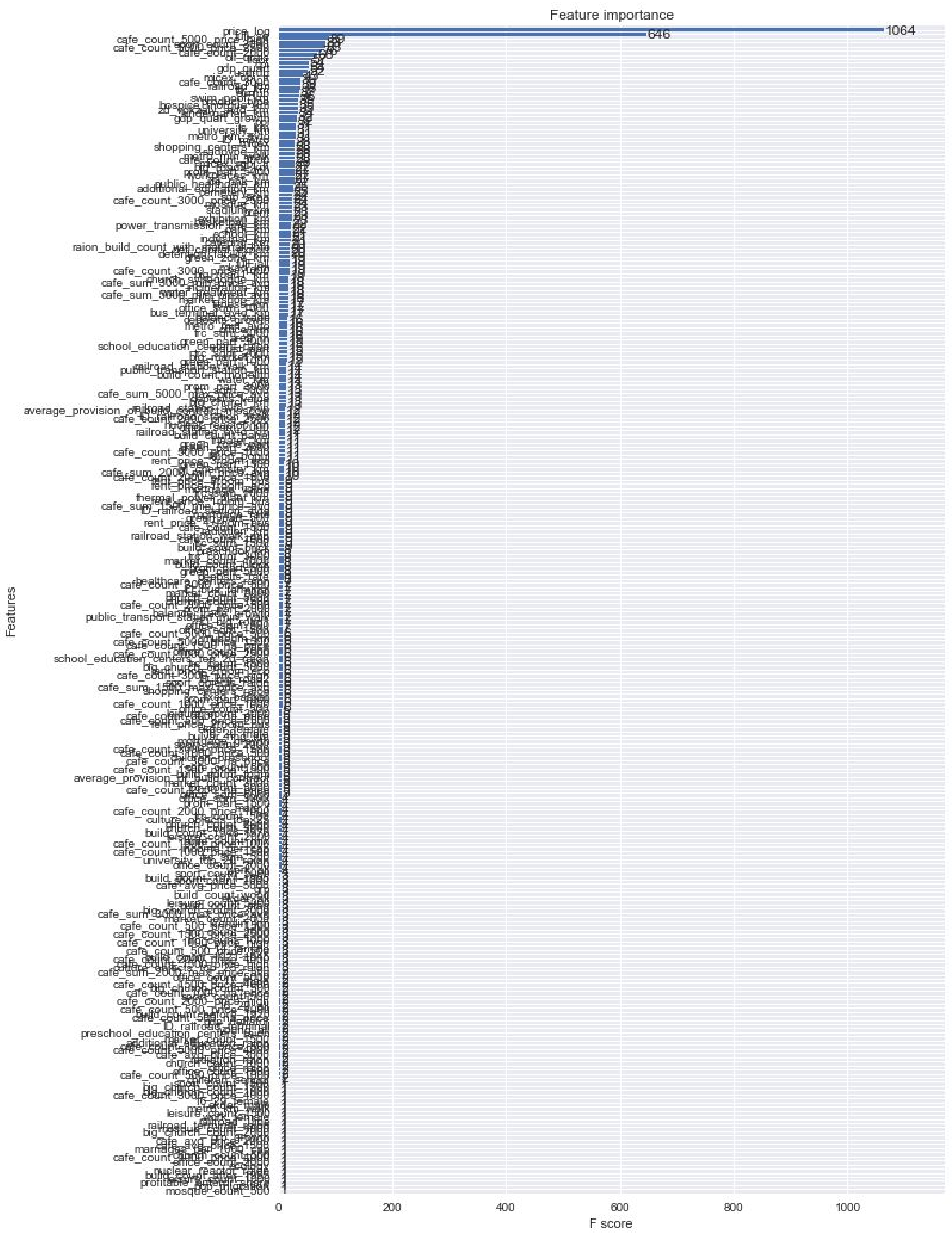
oil_urals	**gdp_quart**	**gdp_quart_growth**	**cpi**	**ppi**	**gdp_deflator**
count	30471	30471	30471	30471	30471
mean	97.704786	18216.99066	1.444098	411.66049	492.213019
std	18.996897	1668.958493	1.70222	32.436292	28.597338
min	46.3414	14313.7	-2.8	353	420.7
25%	93.74	17015.1	0.7	388.5	472.2
50%	107.1991	18410.7	0.9	408.3	487.9
75%	108.65	19566.5	2.1	428.6	510.1

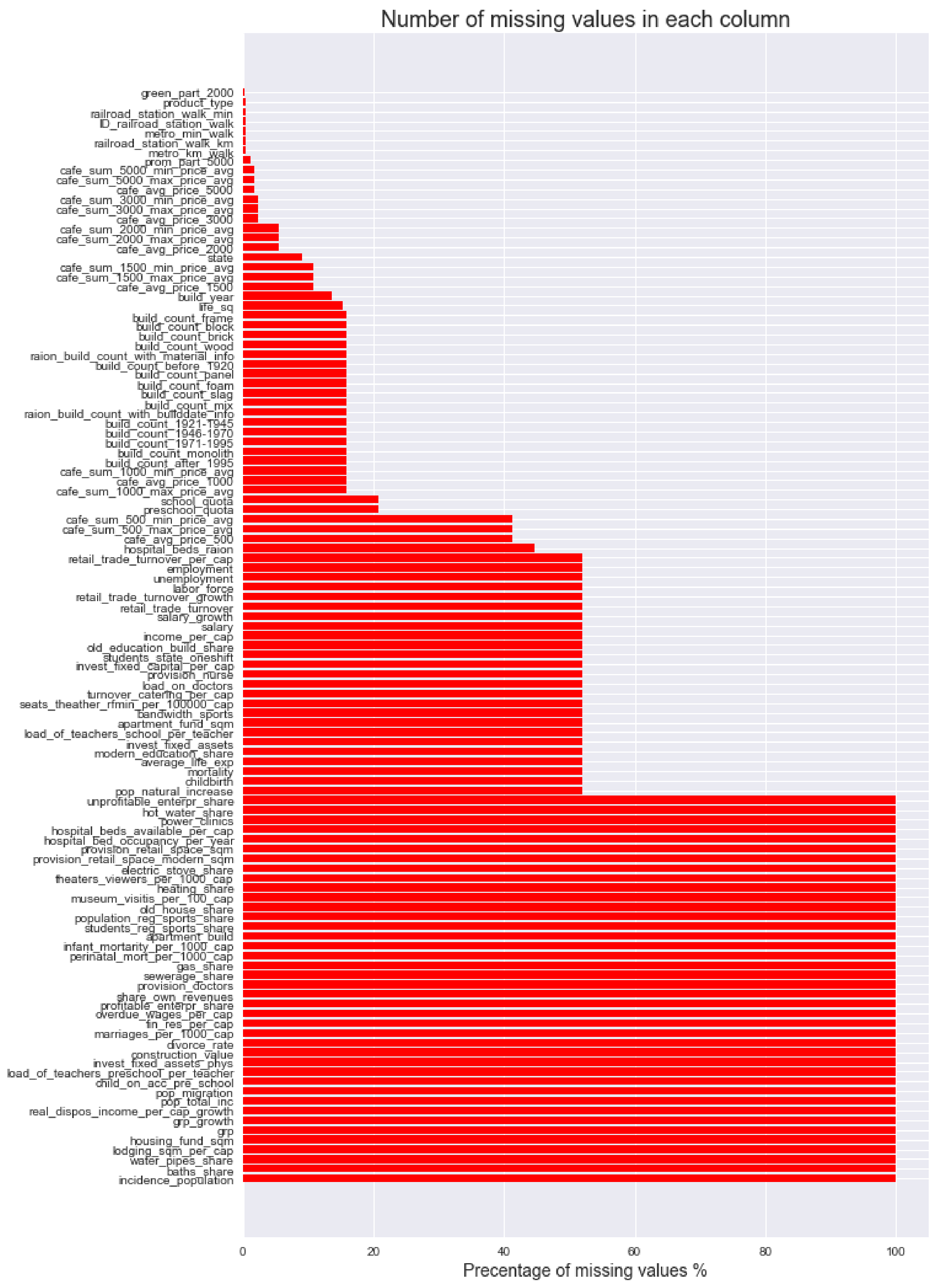
max	122.52	21514.7	5.2	489.5	575.9

Next, I tried understanding the features and why they were included. Then, I looked at which features had the most number of missing values and whether they were critical to the results. I dropped the features with more than 30% missing values and imputed other missing values with the median of the column.

B. Exploratory Visualization

To explore the data, I plotted the feature importances and percentage of missing values, graphs as below:


Feature importance
price_log
1064
646
cafe_count_5000_price[illegible]
Features
F score
0
200
400
600
800
1000
mosque_count_500


Number of missing values in each column
green_part_2000
product_type
railroad_station_walk_min
ID_railroad_station_walk
metro_min_walk
railroad_station_walk_km
metro_km_walk
prom_part_5000
cafe_sum_5000_min_price_avg
cafe_sum_5000_max_price_avg
cafe_avg_price_5000
cafe_sum_3000_min_price_avg
cafe_sum_3000_max_price_avg
cafe_avg_price_3000
cafe_sum_2000_min_price_avg
cafe_sum_2000_max_price_avg
cafe_avg_price_2000
state
cafe_sum_1500_min_price_avg
cafe_sum_1500_max_price_avg
cafe_avg_price_1500
build_year
life_sq
build_count_frame
build_count_block
build_count_brick
build_count_wood
raion_build_count_with_material_info
build_count_before_1920
build_count_panel
build_count_foam
build_count_slag
build_count_mix
raion_build_count_with_builddate_info
build_count_1921-1945
build_count_1946-1970
build_count_1971-1995
build_count_monolith
build_count_after_1995
cafe_sum_1000_min_price_avg
cafe_avg_price_1000
cafe_sum_1000_max_price_avg
school_quota
preschool_quota
cafe_sum_500_min_price_avg
cafe_sum_500_max_price_avg
cafe_avg_price_500
hospital_beds_raion
retail_trade_turnover_per_cap
employment
unemployment
labor_force
retail_trade_turnover_growth
retail_trade_turnover
salary_growth
salary
income_per_cap
old_education_build_share
students_state_oneshift
invest_fixed_capital_per_cap
provision_nurse
load_on_doctors
turnover_catering_per_cap
seats_theather_rfmin_per_100000_cap
bandwidth_sports
apartment_fund_sqm
load_of_teachers_school_per_teacher
invest_fixed_assets
modern_education_share
average_life_exp
mortality
childbirth
pop_natural_increase
unprofitable_enterpr_share
hot_water_share
power_clinics
hospital_beds_available_per_cap
hospital_bed_occupancy_per_year
provision_retail_space_sqm
provision_retail_space_modern_sqm
electric_stove_share
theaters_viewers_per_1000_cap
heating_share
museum_visitis_per_100_cap
old_house_share
population_reg_sports_share
students_reg_sports_share
apartment_build
infant_mortarity_per_1000_cap
perinatal_mort_per_1000_cap
gas_share
sewerage_share
provision_doctors
share_own_revenues
profitable_enterpr_share
overdue_wages_per_cap
fin_res_per_cap
marriages_per_1000_cap
divorce_rate
construction_value
invest_fixed_assets_phys
load_of_teachers_preschool_per_teacher
child_on_acc_pre_school
pop_migration
pop_total_inc
real_dispos_income_per_cap_growth
grp_growth
grp
housing_fund_sqm
lodging_sqm_per_cap
water_pipes_share
baths_share
incidence_population
0
20
40
60
80
100
Precentage of missing values %

C. Algorithms and Techniques

Firstly, I began by splitting the data into train and test sets. On this data, I tried some popular algorithms including Linear Regression, Random Forest, Gradient Boosting Regressor, XGBoost, and Deep Neural nets (Tensorflow) as explained below.

1. Linear Regression

Linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. Linear regression models are often fitted using the least squares approach, but they may also be fitted in other ways, such as by minimizing the "lack of fit" in some other norm (as with least absolute deviations regression), or by minimizing a penalized version of the least squares loss function.

2. Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

3. Gradient Boosting Regressor

Gradient Boosting produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

4. XGBoost

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. As compared to gradient boosting machines, it uses a more regularized model formalization to control overfitting, which gives it better speed and performance.

5. Deep Neural Nets (Tensorflow)

Tensorflow is an open source library to replicate the human brain by building and training neural networks to detect and decipher patterns and correlations. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

First I fitted the model to the training data, then used the fitted model to predict on the test set. Subsequently, I stacked selected models with the best results (i.e. RF, GBR, XGB and TF) on the test set.

D. Benchmark

I was aiming to be in the top 50% of results submitted in Kaggle, however the best result I got was only the top 69%.

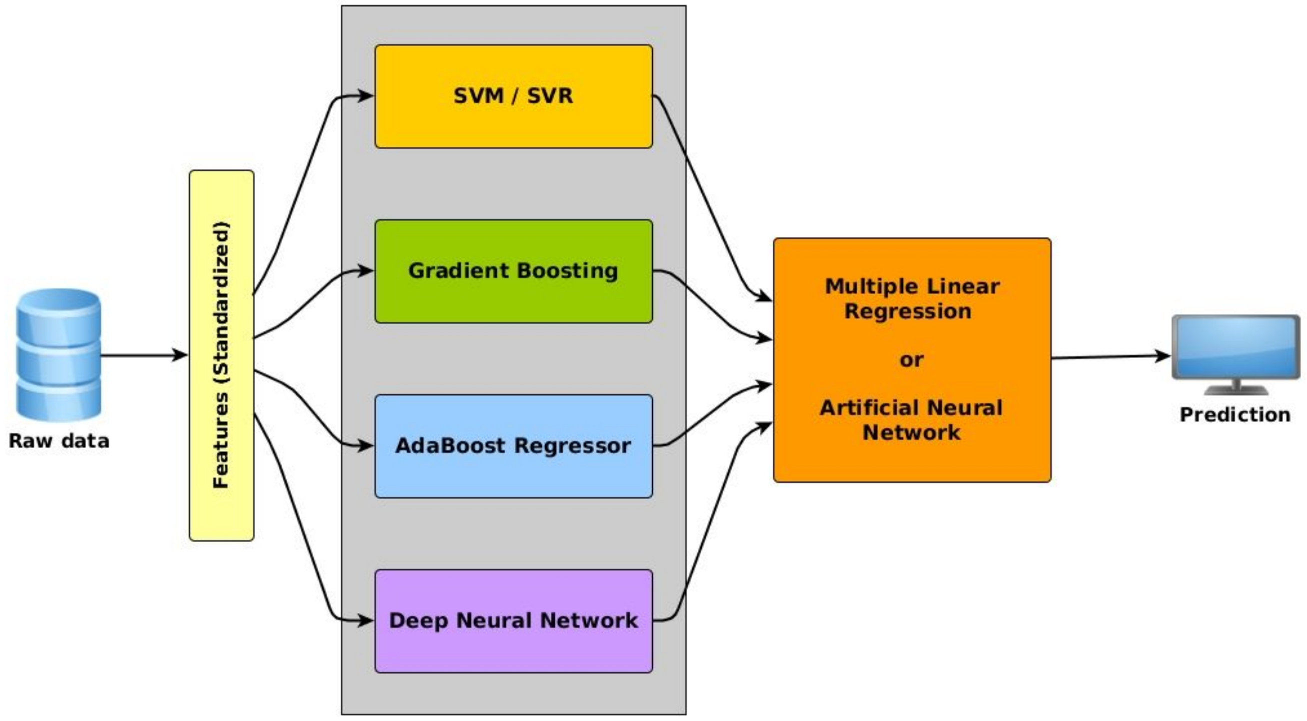
III. Methodology

A. Data Preprocessing

To preprocess the data, I merged the 2 given datasets (train and macro) together. Then I examined which features had the most number of missing values and dropped those with more than 30% missing values. I then imputed the rest of the missing values with the median of the column.

B. Implementation

I implemented the algorithms in Jupyter Notebooks and deployed them on Floydhub. For stacking, I ran each model separately and used their predictions as inputs for the stacked model. For the aggregator model, I used linear regression, ridge regression, the TheilSan regressor, as well as Tensorflow. The process follows the diagram below:



C. Refinement

I played around with the parameters manually to get the best results. Though this could be more efficiently done by writing loops (see improvements).

The hyperparameters I tuned for the various Algorithms are listed below:

1. Gradient Boosting Regressor

Max depth = [10,20,30]
N_estimators = [20,50,100]

Optimal:
Max depth =10, N_estimators = 100

2. XGBoost

eta = [0.01, 0.02, 0.05]
Max depth = [5,10,20]

Optimal:
Eta = 0.05, Max depth = 5

3. Tensorflow

Learning rate = [0.001, 0.005, 0.01]

Optimal:
Learning rate = 0.001

IV. Results

A. Model Evaluation and Validation

I evaluated the models based on Root mean square error (RMSE).

Model	**RMSE**
Random Forest Regression	0.776548
Gradient Boosted Regression	0.78121
Deep Neural Nets	0.786273

Ensemble	0.790231
XGBoost	0.799111
Stack	0.818542

Of these metrics, Random Forest regressor had the best scores among all the models. I split the data sets into train-test sets and used took out 100 rows of data to serve as validation set for the models. I believe more feature engineering has to be done with regards to what Russian buyers/sellers regard as important in influencing the prices in the housing market and whether the given data makes sense. However, it appears that the models produced decent results; these results could probably be used as a guide but further refinement has to be done before they are factored into real-world decisions.

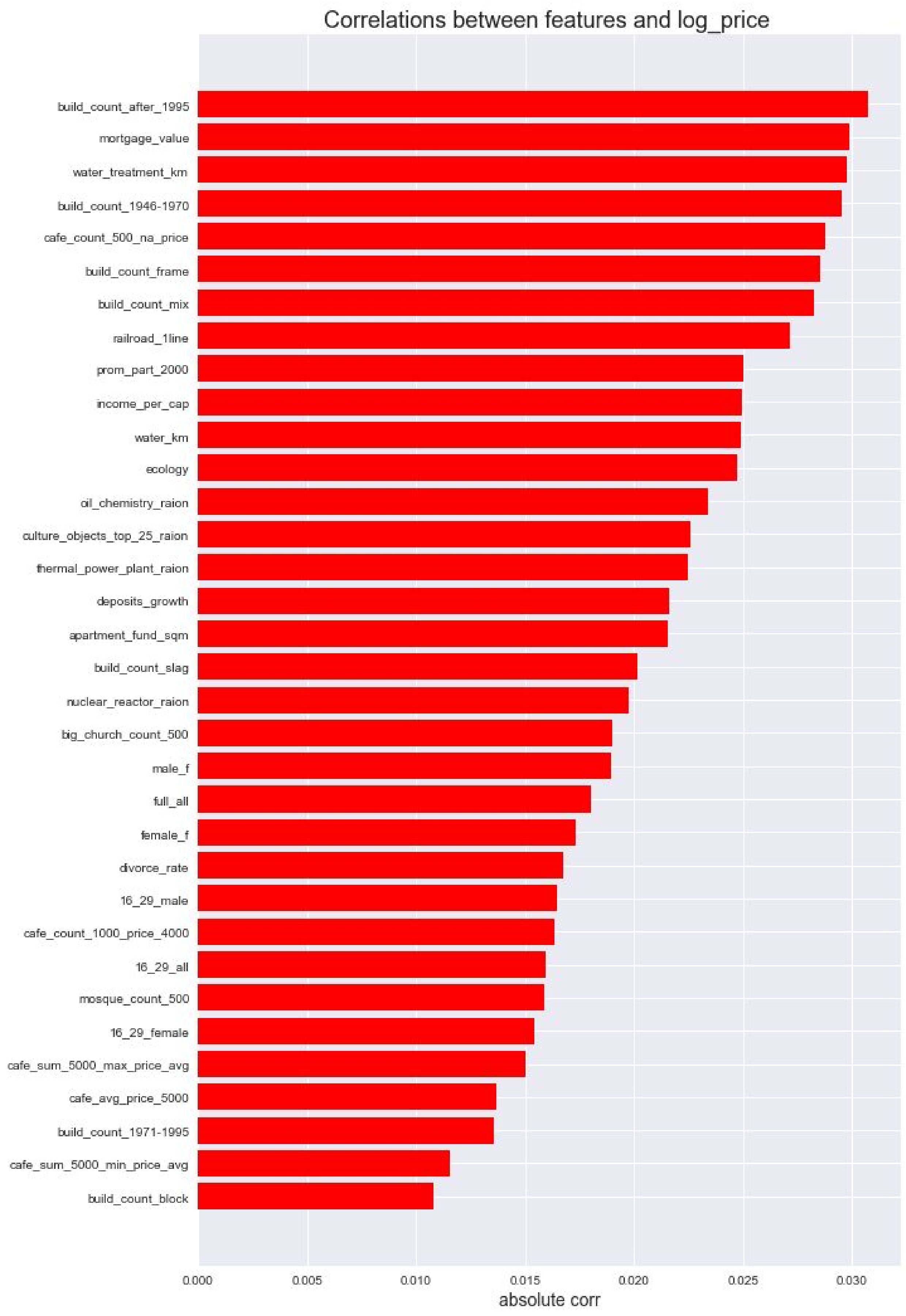
B. Justification

Getting 69% means the model wasn't robust enough, also the RMSEs are higher than what I usually see. As mentioned, there should be more feature engineering by really understanding the Russian housing market, and how each factor drives housing prices.

V. Conclusion

A. Free-form visualisation

From the correlation graph below, it was surprising that external features (macro, amenities) were more important than the hedonic features (floor, number of rooms, size) in determining the price. Therefore it was important that I included the macro dataset in the process.


Correlations between features and log_price
build_count_after_1995
mortgage_value
water_treatment_km
build_count_1946-1970
cafe_count_500_na_price
build_count_frame
build_count_mix
railroad_1line
prom_part_2000
income_per_cap
water_km
ecology
oil_chemistry_raion
culture_objects_top_25_raion
thermal_power_plant_raion
deposits_growth
apartment_fund_sqm
build_count_slag
nuclear_reactor_raion
big_church_count_500
male_f
full_all
female_f
divorce_rate
16_29_male
cafe_count_1000_price_4000
16_29_all
mosque_count_500
16_29_female
cafe_sum_5000_max_price_avg
cafe_avg_price_5000
build_count_1971-1995
cafe_sum_5000_min_price_avg
build_count_block
0.000
0.005
0.010
0.015
0.020
0.025
0.030
absolute corr

B. Reflection

In summary, I tried to understand the raw data given by Kaggle, preprocessed the data, picked out what was shown to be important, and ran them through different regression models to predict the labels on the test set (price of the property). Using different metrics, I selected the most effective models and stacked them to see if results could be further improved.

I was really glad to learn how stacking works and understand the difference between bagging, boosting and stacking through this project. In this case, it seems that stacking doesn’t improve the results, but is useful for certain stepped problems such as finding out how much donors would donate; your first model could predict whether a donor would donate (discrete) and the second model would predict how much they would donate (continuous).

C. Improvements

I would write loops to try different hyperparameters for the models (especially for the deep neural nets) and experiment with some feature engineering such as logging and scaling the data to improve the results further.