

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

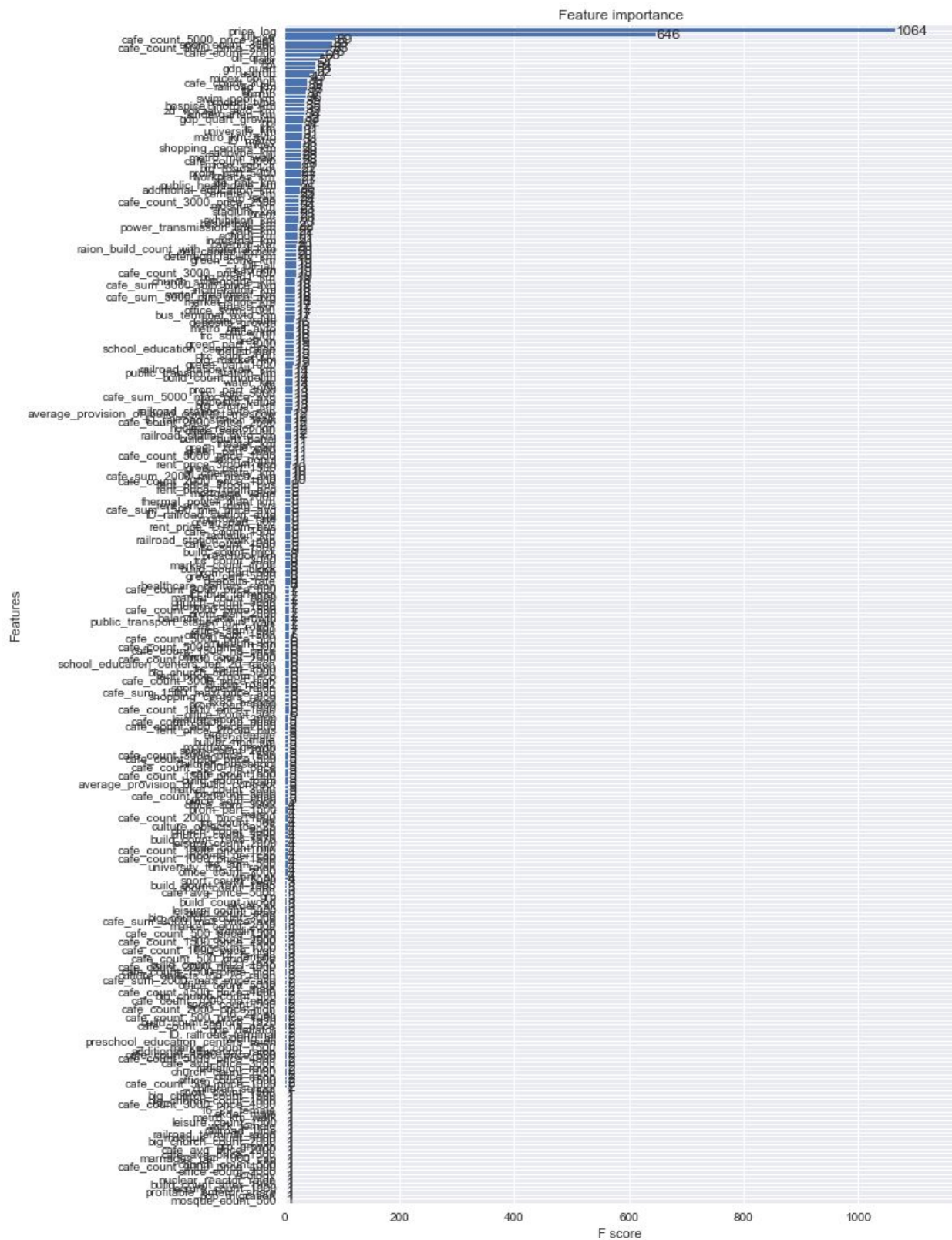
A. Data Exploration

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.

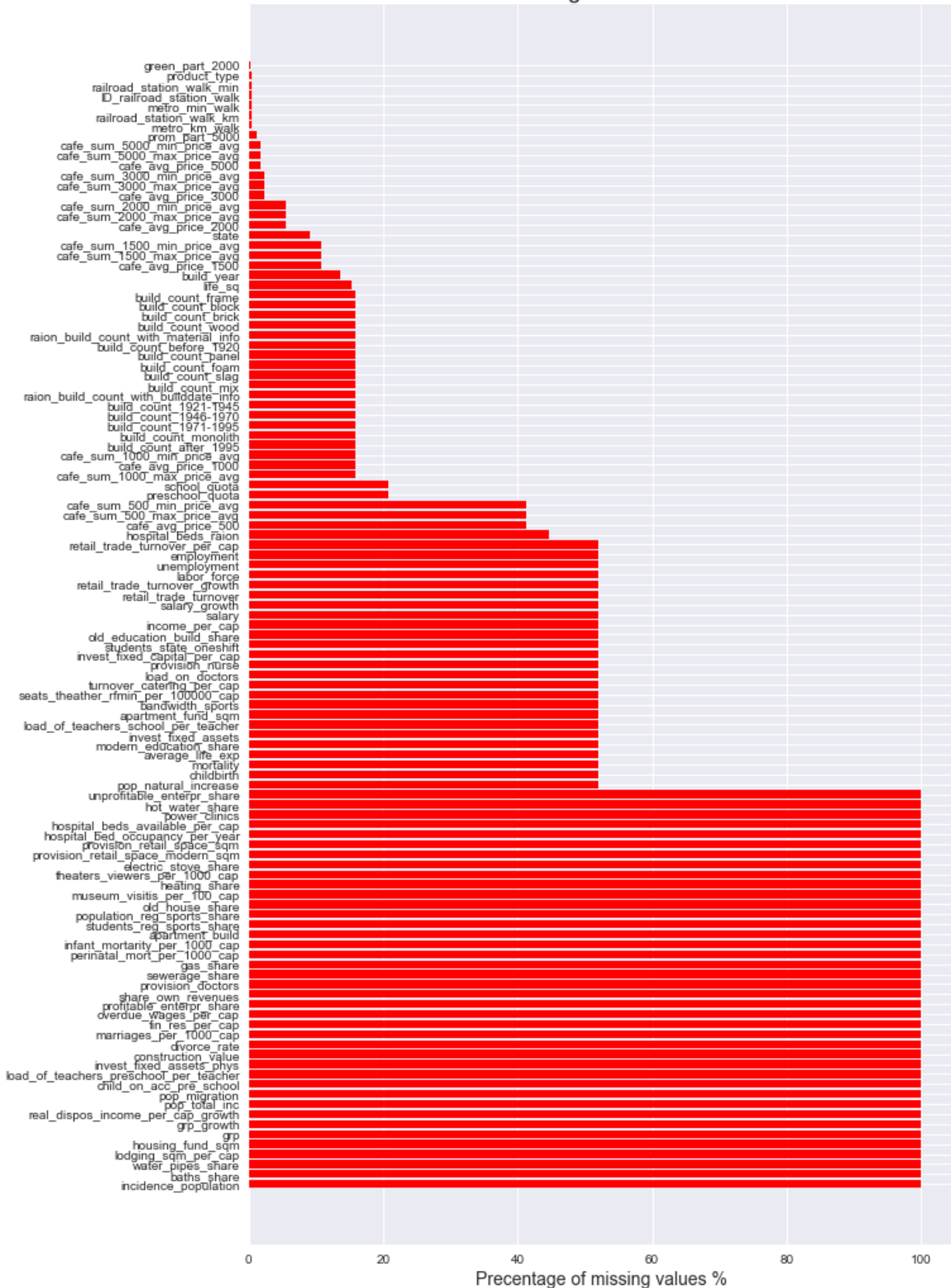
Next, I tried understanding the features and why they were included. Then, I looked at which features had the most number of missing values and whether they were critical to the results. I dropped the features with more than 30% missing values and imputed other missing values with the median of the column.

B. Exploratory Visualization

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



Number of missing values in each column



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 Tensorflow. 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%.

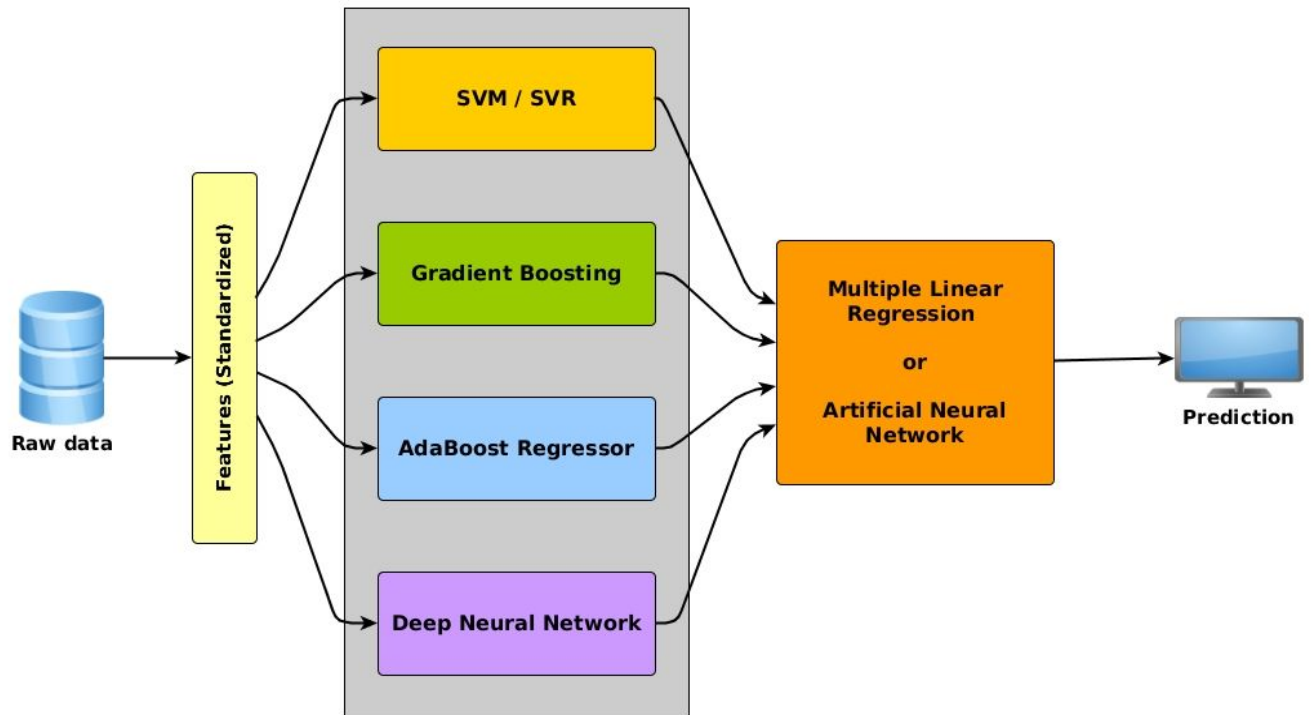
III. Methodology

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]

2. XGBoost

eta = [0.01, 0.02, 0.05]

Max depth = [5,10,20]

3. Tensorflow

Learning rate = [0.001, 0.005, 0.01]

IV. Results

A. Model Evaluation and Validation

I evaluated the models based on R-squared, explained variance score, and mean squared error. Of these metrics, XGBoost 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 think the model is not robust enough to solve the real-world problem. 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

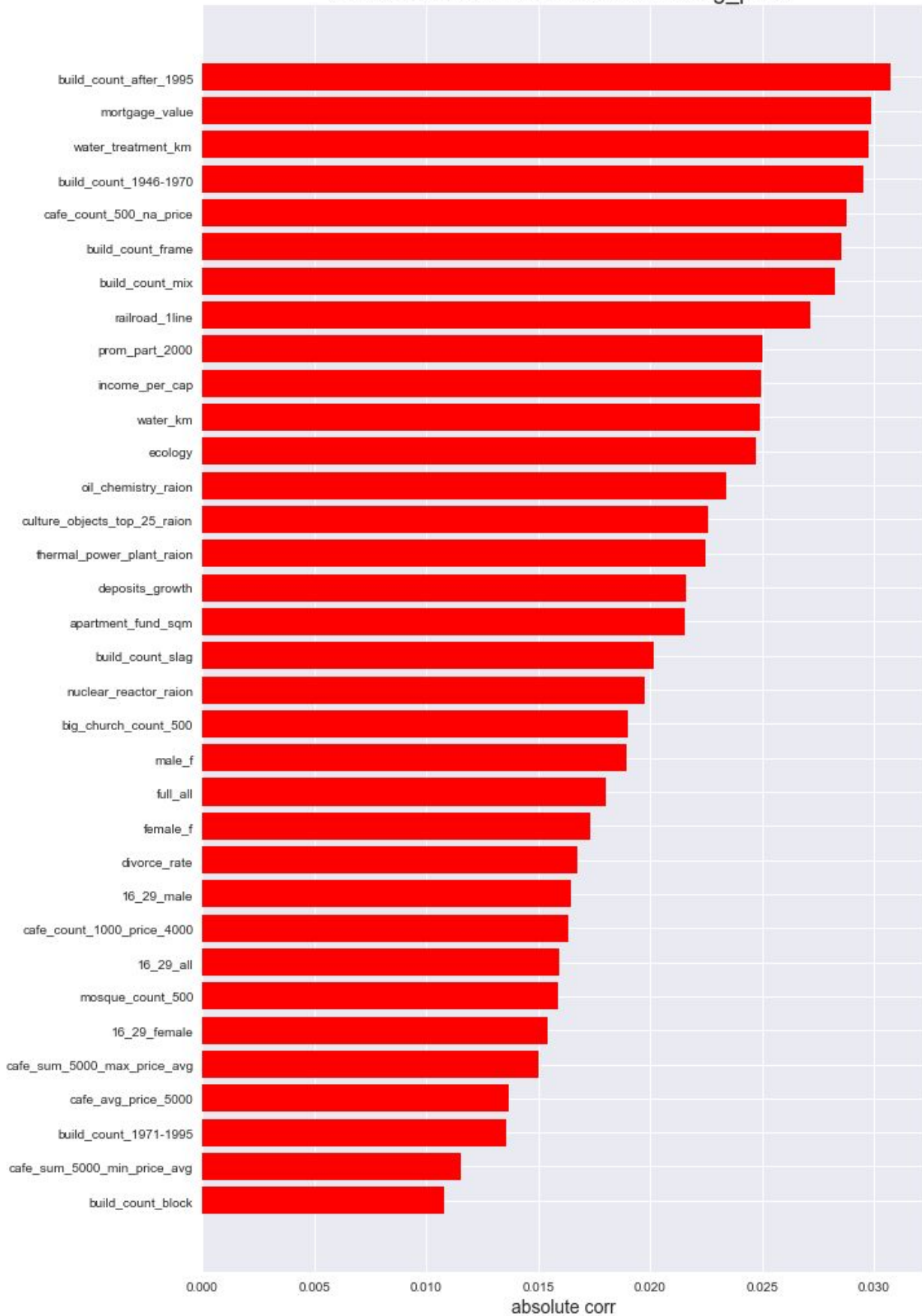
The reason why I took out the validation set from the train set is because this is a Kaggle competition and we were given a test set without the ability to know how each model would do on the score (unless we submit each model score as a separate entry).

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



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. It seems that stacking doesn't improve the results much 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.