Capstone Project Report – Sberbank Russian Housing Market

1. Definition

a. Overview

The scope of this capstone project is based on a Kaggle competition launched by Sberbank, Russia's largest banking and financial services company, with the goal of predicting real estate price fluctuations in Russia's volatile economy.

The provided training data files include a training data set, a test data set, macroeconomic patterns data set, and a sample submission file.

b. Problem Statement

As stated in the competition's guidelines, Sberbank helps their customers by making predictions about realty prices so they are more confident when they sign a lease or purchase a building.

Furthermore, Russia's volatile economy makes forecasting prices as a function of apartment characteristics a unique challenge. Internal property characteristics and external factors considerably influence the sale price of each property.

Thus, the objective is to build and assess different learner models that predict realty prices by relying on a dataset that includes housing data and macroeconomic patterns.

To solve this problem, we will begin by exploring the data provided, relationship between its various features, then we will process and wrangle the data to prepare it for analysis, and finally we will conduct our analysis on the data with a combination of learner methods. The learners will include a series of supervised learning models such as ensemble methods, generalized linear models, and neural networks. Upon the completion of the analysis, the resulting test set will be submitted to the Kaggle competition for evaluation.

c. Metrics

According Sberbank's Kaggle competition rules, solutions are evaluated on the RMSLE (Root Mean Squared Logarithmic Error) between their predicted prices and the actual data. The target variable, *price_doc*, in the training set, is the sale price of each property.

The RMSLE is defined as $\epsilon=sqrt((1/n)\sum(i=1n)(log(pi+1)-log(ai+1))\hat{}2)$. The only difference between RMSLE and the traditional RMSE (Root Mean Squared Error) is that it applies the logarithmic function to both the predicted and actual values before calculating the root mean squared error. Thus, It is worth noting that RMSLE penalizes an under-predicted estimate greater than an over-predicted estimate.

Additionally, the ranking on the competition's leadership board will serve as a benchmark for how various learners and approaches measure up against one another.

2. Analysis

a. Data Exploration

The following libraries were leveraged to explore and analyze the provided data set: *pandas* (data manipulation & analysis), *numpy* (linear algebra), *scikit-learn* (learning & mining modules), *matplotlib* (figures), *seaborn* (visualizations), *geopandas* (map analysis), and *xgboost* (distributed gradient boosting).

The training data set contains 30,470 transactions with 291 features (internal property characteristics and external factors), as shown by using *dftrain.shape*.

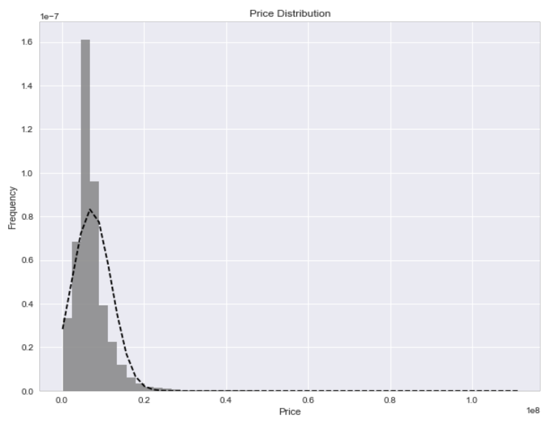
Given the large amount of features in this data set, a series of dimensionality reduction and preprocessing methods will be required to reduce noise and obtain accurate results. These are discussed in the *Methodology* section.

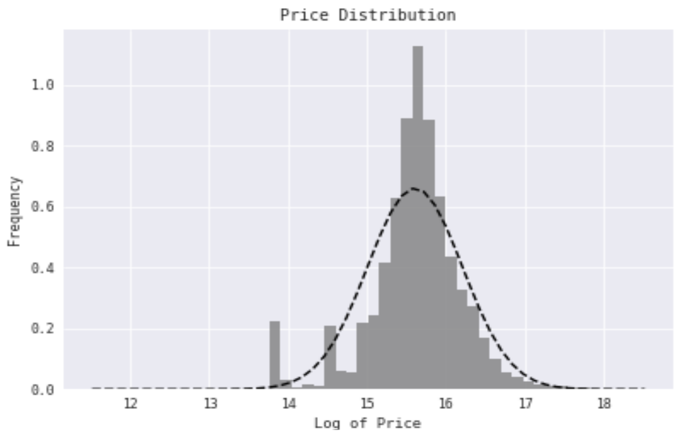
b. Exploratory Visualization

The large number of features and information types allows for exploring the data set from several perspectives such as distribution, time, and location. Furthermore, the relationship between different features and the target variables are discussed in the *Methodology* section.

i. Transaction Price Distribution

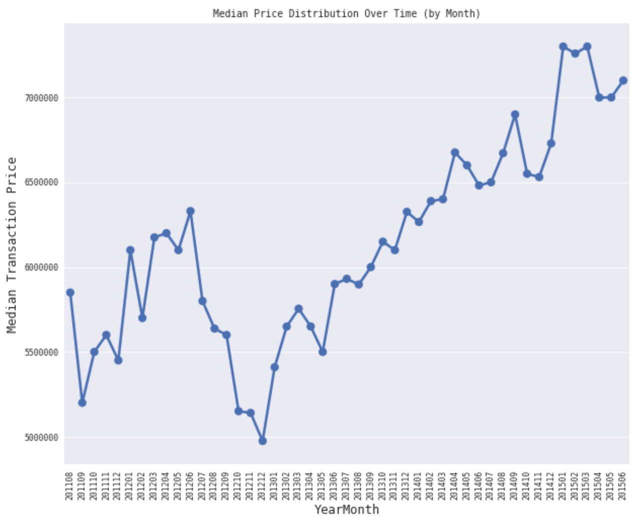
Plotting the transaction price distribution histogram (shown on left) reveals that the data distribution is skewed and has a fairly long tail. As previously mentioned in the *Metrics* section, the results are evaluated using RMSLE. This prompts plotting the price distribution histogram using the log of the price data (shown on right). This provides a more centered distribution curve.



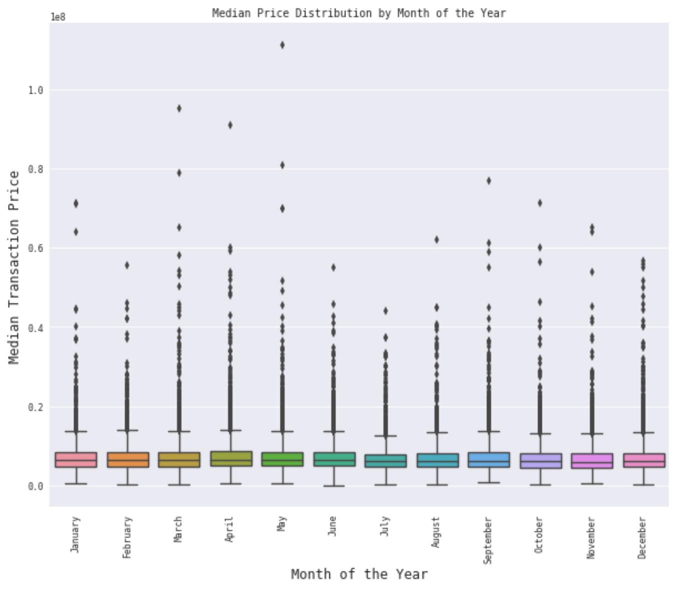


ii. Effect of Transaction Time on Price

Timing also plays an important role in determining the transaction sale price of realty. The following pointplot explores the median transaction price from 2011 to 2015. Prices appear to rise over this period from 6e7 RUB to 7e7 RUB with the exception of a dip between mid-2012 to early 2013.



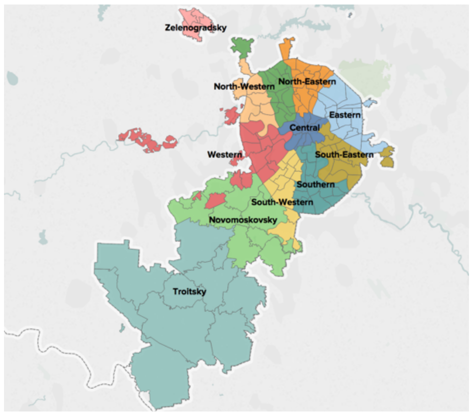
Seasonality might also play a role in sale price fluctuation, which prompts investigating the median transaction price by month of year. The following boxplot shows that the month of year does not have a significant impact on the realty sale price.

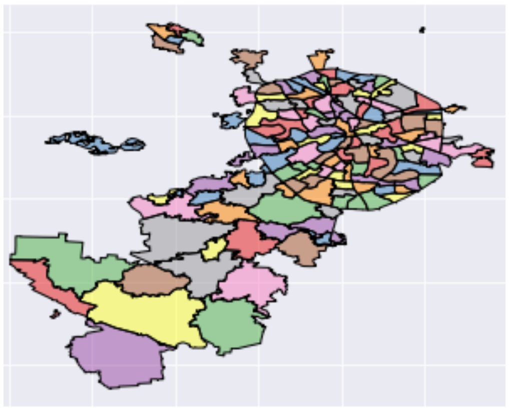


iii. **Effect of Location on Different Metrics**

Location, location, location. Location is arguably the biggest factor when considering the sale price of any real estate asset. With the use of the *geopandas* library and Tableau, to read a shapefile of Moscow plotted and translated by Kaggle user, JTremoureux, one can query the data to visualize and explore relationships between various features and asset location.

Each transaction in the data set is located in one of 12 administrative okrugs (aka “district” in the data set), as show on the left, which in turn are comprised of the 146 raions (aka “sub-area” in the data set), as shown on the right.

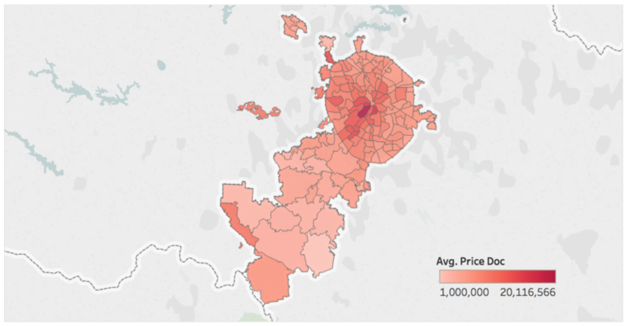




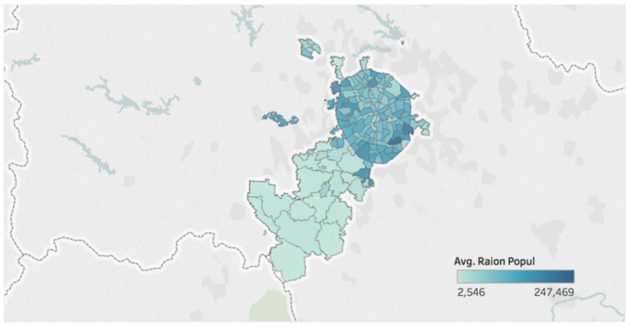
The following four map figures (from top left, clockwise) demonstrate:

- The average transaction price by sub-area, which instantly reveals that prices tend to be highest in sub-area located within the *Central* and *Western* districts.
- The population size by sub-area, which demonstrates that residents tend to be concentrated in 6 (*Central, Eastern, North-Eastern, South-Eastern, Western, North-Western, South-Western*) of the 12 districts.
    - Relatively higher average transaction prices are reflected in these 6 districts within the first chart.
- Industrial Parts by sub-area, not surprisingly, illustrates that these six districts are in fact more “industrial” than all others.
- Green Zone by sub-area, reveals that grasslands and forests, tend to be concentrated in the most-southern and largest district, *Troitsky*, where population size and average transaction price tend to be lowest.

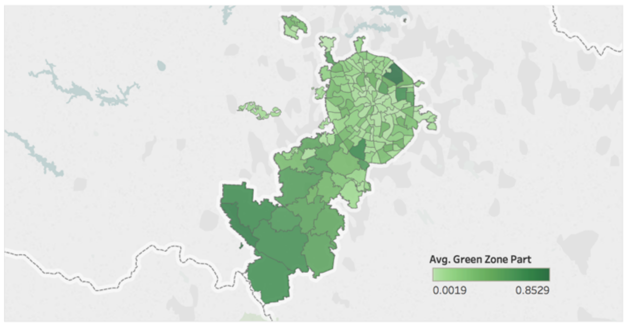
Mean Transaction Price by Sub-Area



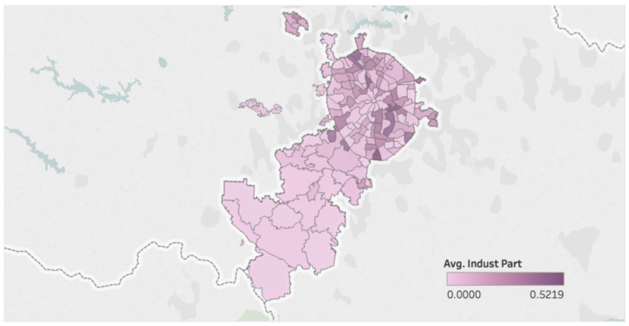
Population Size by Sub-Area



Green Zone Index by Sub-Area



Industrial Parts by Sub-Area



c. **Algorithms and Techniques**
Based on the nature of the problem and the input data, a regression method might seem the most suitable to obtain the numerical target variable (price_doc). Regression is concerned with modeling the relationship between variables that is iteratively refined using a measure of error in the predictions made by the model.

d. **Benchmark**
As this is a Kaggle competition, the employed algorithms will be evaluated based on the best (lowest) RMSLE achieved on the training data set when evaluating and predicting the price_doc.

Furthermore, the models will be used to infer the price_doc variable in the test data set and entered to the competition to benchmark their performance on the Kaggle leaderboard relative to other participants.

Furthermore, an Ensemble method might also prove useful for our purpose. Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction. More specifically, an implementation of gradient boosting machines, called XGboost (Extreme Gradient Boosting), which focuses on scalability and performance, will also be used on its own. Details on these methods are as follows:

i. **LASSO**
LASSO (least absolute shrinkage and selection operator) is a regularization technique for performing linear regression. It includes a penalty term that constrains the size of the estimated coefficients. Its objective function is (1 / (2 * n_samples)) * ||y - Xw||^2_2 + alpha * ||w||_1. Using the aforementioned features, a LASSO model was first implemented with an alpha value options, which is a constant that multiplies the L1 in the optimization objective function, of 0.1, 10, and 50. Furthermore, RMSLE scoring model and 10-fit cross-validation were used.

ii. **ElasticNet**
ElasticNet is a combination of LASSO and ridge regression. It is the same as LASSO when alpha is 1, and as alpha shrink towards 0, it approaches ridge regression. It minimizes the objective function 1 / (2 * n_samples) * ||y - Xw||^2_2 + alpha * l1_ratio * ||w||_1 + 0.5 * alpha * (1 - l1_ratio) * ||w||^2_2 .

In our case, we used 10-fit cross-validation and tried alpha constants, which are multiplied by the penalty terms, of 0.1, 1, 10, and l1 ratios, which are ElasticNet mixing parameters, of 0, 0.5, 1, thus alternating between a penalty of L1 and L2, or a combination of both.

iii. **XGBoost**
As introduced above, XGBoost is a library designed and optimized for boosted tree algorithms. In comparison to traditional gradient boosting, XGBoost uses a more regularized model formalization to control over-

fitting, which usually gives it better performance. Overall, its objective is to push the limit of computations resources for boosted tree algorithms. This makes it suitable for our case for both defining important features and inferring our target variable, price_doc.

In our case, after trial and error, we used XGBoost with a 10-fit cross-validation, 500 boosted rounds, and the following parameters: a maximum depth of 8, a linear regression objective, learning rate (eta) of 0.05, and a Root mean squared error evaluation metric (note: XGBoost does not support RMSLE).

iv. Ridge

Ridge Regression is useful for analyzing multiple regression data that suffer from multicollinearity, such as our Sberbank realty data set. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value. Ridge regression was leveraged with a 10-fit cross-validation, with alpha options of 0.1, 1, 10 and a RMSLE scoring method.

v. Stacked Regressor

Stacking regression is an ensemble learning technique to combine multiple regression models via a meta-regressor. In this case, LASSO, Ridge and ElasticNet models are trained on the complete training set; then, the meta-regressor, XGBoost, is fitted based on the outputs - meta-features - of the individual regression models in the ensemble.

In this model, a RMSLE scoring model was leveraged to evaluate the outputs, a 10-fold cross-validation and the meta learner (XGBoost) had a maximum depth of 7.

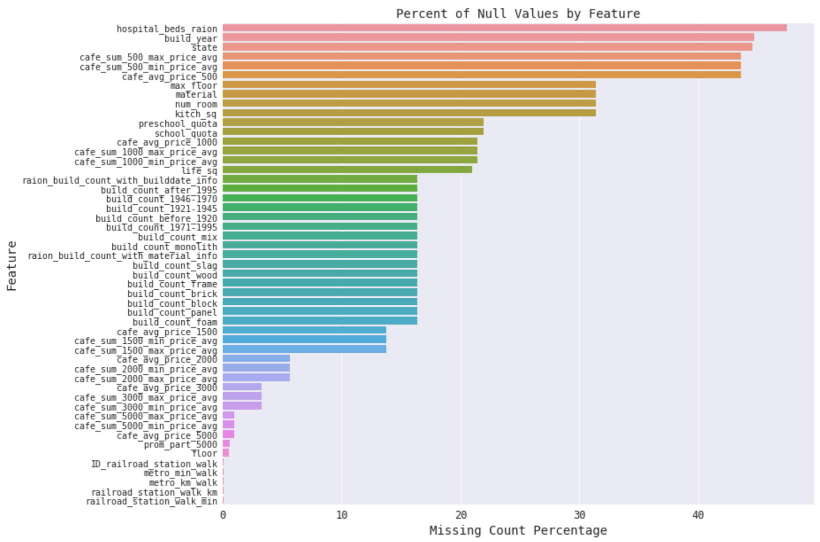
3. Methodology

Prior to analyzing the data using learning models, it is critical to uncover any anomalies or characteristics within the data or input that need to be addressed and corrected.

a. Preprocessing

i. Null Values and Missing Data

At first glance of the data set, there appears to be a significant number of missing values across various features. Plotting this helps us understand which features might negatively impact our modelling if included. The top 5 features, in terms of missing values, are missing for more than 40% of transactions. In our case, we decide to replace the null value cells with a number outside of the range of considered values, such as -99.

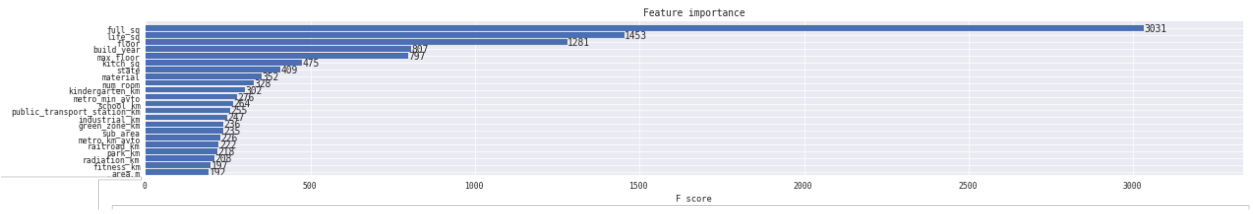


ii. Label Encoding (Categorical to Numerical Conversion)

Within our data, there are many features for which their values are considered categories as opposed to numbers. We leverage sklearn's *preprocessing LabelEncoder* library to convert those to numerical values and discover that there is a handful of these features: product_type, sub_area, culture_objects_top_25, thermal_power_plant_raion, incineration_raion, oil_chemistry_raion, radiation_raion, railroad_terminal_raion, big_market_raion, nuclear_reactor_raion, detention_facility_raion, water_1line, big_road1_1line, railroad_1line, and ecology, that required conversion.

iii. Feature Importance

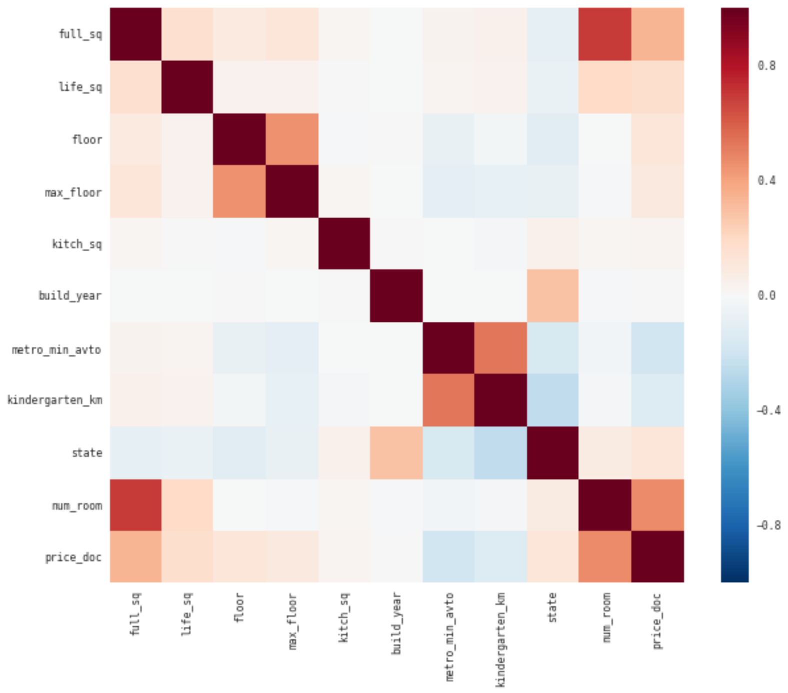
Using the XGBoost library, we can generate importance scores for each attribute after the boosted trees are created. The importance provides a score that indicates how useful or valuable each feature was in the construction of the boosted decision trees within the model. This is based on the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for.



One observes that the most important features, based on F score, are full square foot size, living area square foot size, floor, build year, kitchen square foot size, state, material and number of rooms. As, the remainder of the features are considered less important, we can use these most important features in our training model, as opposed to using all 294 features, and increasing our model's complexity, and in turn, degrading its performance.

iv. Correlation Detection

After extracting some of the most important features, it is useful to plot a correlation heatmap matrix to visualize which of our top features are (1) correlated to one another and (2) to the target variable, pride_doc.

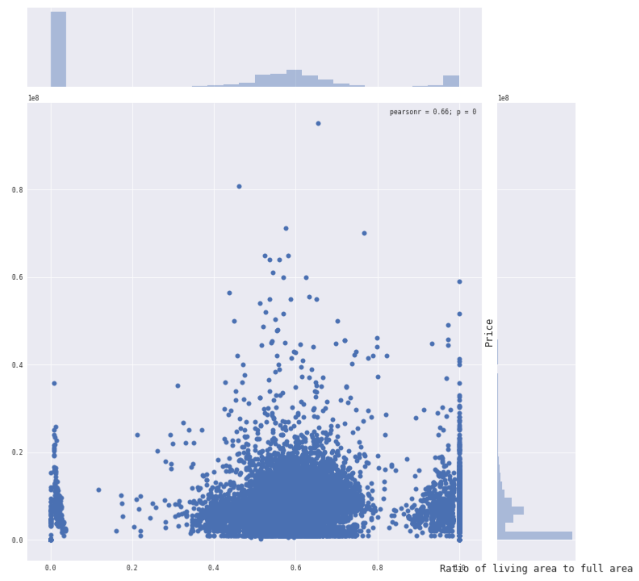


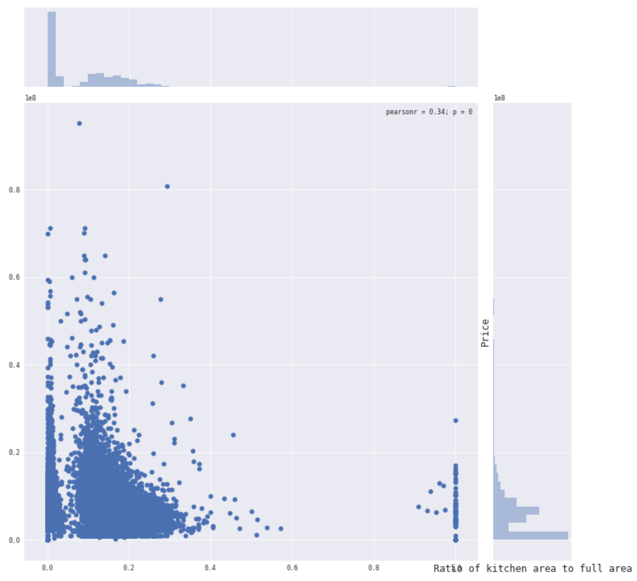
Based on the above heatmap, it is evident that (1) there is a strong correlation between the number of rooms and full area square foot size (2) above average correlation between num_room and price_doc (3)

v. New Variable Creation

As some of these values are correlated to one another, we decide to generate to combine some of them into a new variables or ratios. Below is an example of two of those and their relationship with the price_doc:

- *Ratio of living square foot area/full square foot area vs. price_doc* (show on left): demonstrates that the majority of units have a living area that's around 60% of the full realty size, with a few outliers where the living area value was either 0 or exceeding the full realty size in the provided data. However, there's no strong correlation between this ratio and price_doc.
- *Kitchen square foot area/ full square foot area vs. price_doc* (shown on right) demonstrates that a smaller kitchen relative to full property size is more desirable, as the bulk of high prices is around the 0-0.15 ratio. Note: the values around 1 are outliers, for which the data set provided kitchen size values larger than the actual property size.





b. Implementation

Since we determined, through XGBoost, that only a handful of features are most important and to avoid overfitting, we can leverage a Regularization algorithm. Regularization is an extension made to another method (typically regression methods) that penalizes models based on their complexity, favoring simpler models that are also better at generalizing. We will leverage three regularization algorithms: (1) Least Absolute Shrinkage and Selection Operator (LASSO), (2) Ridge, and (3) ElasticNet.

After evaluating each method separately, we will use a StackingRegressor, an ensemble-learning meta-regressor for stacking regression, to combine the aforementioned model with a meta-learner and assess whether this improves our results. In terms of features, we will be using full_sq, life_sq, floor, max_floor, kitch_sq, num_room, material, state, kitch_full_ratio, life_full_ratio.

c. Refinement

Within each model, trial and error was conducted to determine the appropriate parameters needed for each algorithm in order to achieve improved results. These refinements include:

- Alternating between various cross-validation fits such as 5, 10, and 15 and comparing RMSLE scores.
- Using various alpha values for the regularization models (LASSO, Ridge, ElasticNet) to provide varying weights to the L1 and L2 penalties and comparing the results.
- Using different combination of features of analysis based on XGBoost's feature importance results and assessing whether each degraded or improved the model's performance.
- Alternating the learned used as the meta-learner in the StackedRegressor mechanism and reaching a conclusion that having XGBoost as the meta-learner performed best.
- Replacing null values with a value outside the range to improve performance.

4. Results

a. Model Evaluation

Using the training set, the following RMSLE error scores were achieved using the four models described above:

Model	RMSLE Score
LASSO	0.0355559851955
ElasticNet	0.0355630000202
XGBoost	0.0316809316027
Ridge	0.0355630000202
Stacked Regressor	0.0328858196671

Based on the results above, we can conclude that these models are fairly robust for the problem, as the RMSLE never exceeds 3.6% RMSLE for any of the models. This is a considerably accurate prediction score for an application such as determining the price of a realty sale. Granted, these scores are merely based on the RMSLE scores of the training set and can perform extremely differently in real-world scenarios due to factors such as market shifts (regulatory changes, macroeconomic events), transaction costs, optimization bias (overfitting), supply/demand fluctuations, and a plethora of market-impacting factors.

b. Justification

Based on the above results from evaluating the models using the RMSLE score, one can observe that the XGBoost model performed best (lowest RMSLE) – even outperforming the StackedRegressor. The StackdRegressor outperformed every regularization method but did not outperform XGBoost. Thus, XGBoost was used to evaluate and predict the price_doc in our testing data set.

Using the XGBoost model and predicting the price_doc in the data set, the XGBoost achieved a ranking of 1365/1730 participants (as of May 18th, 2017) in the Sberbank Housing competition.

5. Conclusion

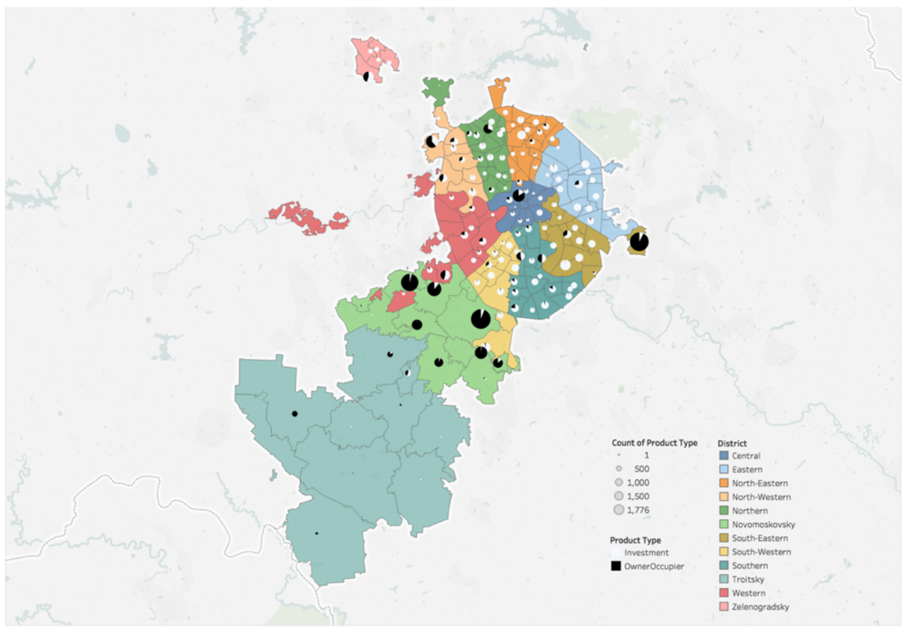
a. Free-form Visualization

An important aspect about this project is understanding the intersection of location, time, internal, and external factors' impact on the sale of a realty in a volatile economy such as Russia. The following map figure combines sales volume and transaction type by sub-area to uncover additional information about realty sales across the aforementioned 12 districts:

- The map is segmented by colour across the 12 districts
- Circle size correlates to the volume of transactions in each of the 146 sub-areas. This demonstrates:
    - The majority of transactions/sales are concentrated in 10 of the 12 districts, with ~8 sub-areas dominating sales volume.

    - Sales volume is evenly split amongst sub-areas in the 6 most heavily populated districts
    - Very low sales volume in *Troitsky* and *Zelengoradsky*
- Each circle is segmented by transaction type, which is categorized by “Investment” (in white) and “Owner/Occupancy” (in black)
    - Most sales in the popular 6 districts are purchased with the intent of investment with the exception of a few sub-areas in the *Central* and *South-Eastern* districts.
    - Almost all transactions in *Novomoskovksy* are purchased with intent of occupancy by buyers

Ownership Type and Number of Transactions by Sub-Area



Thus, to summarize our solution, we began by exploring the data provided, to determine relationships and draw correlations between a subset of the 294 features in our data set. We proceeded with evaluating the data from the perspectives of time, location, internal properties, and external factors. Based on these findings, we processed and modified the data to better suit our analysis needs. Some of these modifications included replacing null values, creating new variables, converting categorical values to numerical ones and more. We then proceeded to analyze the data through various regularization and ensemble learner methods. After refining and tuning the models’ parameters, we selected the best-performing model from an RMSLE scoring perspective and submitted the solution to the Sberbank Kaggle competition for evaluation.

b. Reflection

It is particularly interesting that the provided macroeconomic data only captures a few years, which is not specifically useful when attempting to model the overall economic and social climate of a country/city. It would have been very useful to get macro data over a longer period of time, in order to increase its usefulness, since variables beyond exchange rate and stock prices, rarely move on a daily or monthly basis.

The second fascinating aspect is the performance of the various models and how the XGBoost model outperformed the StackedRegression model, which included multiple learning methods. This demonstrates that adding multiple models does not necessarily equate to superior results.

c. Improvement

To improve this model, several steps can be taken to achieve a more accurate prediction. These include further data processing to remove outlier values, creating new strongly correlated variables, leveraging macroeconomic data to overlay it on internal property characteristics, and perhaps relying on domain knowledge to incorporate factors that might not be captured by the data sets provided. Furthermore, different learning models within deep learning, such as Deep Boltzmann Machine (DBM), Deep Belief Networks (DBN), Convolutional Neural Network (CNN), and Stacked Auto-Encoders may perhaps help improve the performance.