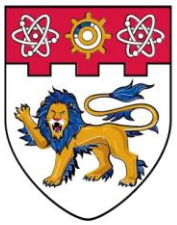


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC4000/CZ4041/CE4041 - Machine Learning

Group 33

Sberbank Housing Prices Prediction

Name	Contributions	Matriculation No.
Amanda Ling Zhi Qi	Data Cleaning, Model Building of Old Model (Catboost), Ensemble Learning	U2022213G
Wong Da You	Building LGBM model, Exploratory Data Analysis, Data Cleaning, building CatBoost model, building ensemble, Model evaluation	U2021409A
Wong Jing Yen	Fine Tuning LightGBM Model, Data Cleaning and Feature engineering for Model 1	U2020809H
Jensen Lim Chang Sheng	Exploratory Data Analysis, Data Cleaning, Video Editing	U2021364D
Tio Guo Yong	Problem Formulation, Challenges, Proposed Solutions, Video Publishing	U2123181B

Video link: https://youtu.be/nGB6pFlW12M

Content

1 Introduction

1.1 Problem Formulation

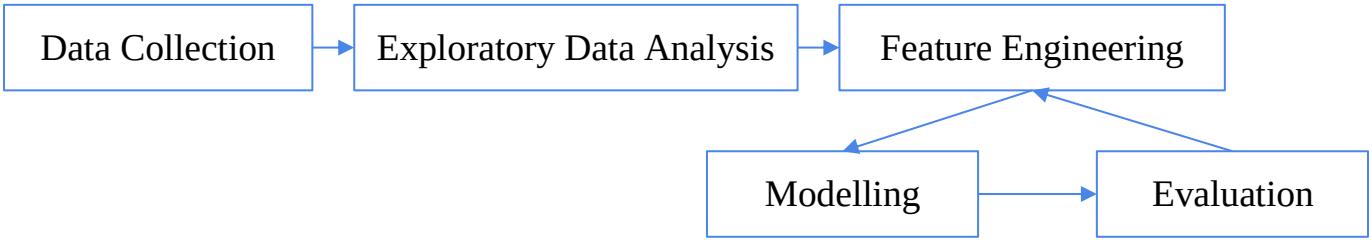
The real estate industry has become more significant in a world of rising population and increased livelihoods. In particular, the pricing of residential houses has become a key factor in this industry. However, such pricing can vary according to the characteristics of the houses and can fluctuate according to the status of the economy. With the prediction of the changes in the pricing in advance, consumers can make a more decisive choice on the purchases and the sales of the houses. The multi-directional interactions of the housing features and the economic indicators have become a challenge in predicting realistic and reasonable prices. Hence, a holistic, inclusive and effective prediction model needs to be able to consider such variables.

1.2 Dataset Description

To model such a housing market in the real world, the dataset we use is the Sberbank Russian Housing Market dataset obtained from Kaggle. The dataset consists of information on the properties of the houses in Russia such as the number of rooms and their sizes. It also provides features on the neighbourhoods and the surroundings of the houses. As the housing prices are influenced by the economic status from time to time, the country's macroeconomic data are also provided.

1.3 Pipeline and Challenges

We implement the general data science pipeline as follows:



With more than 200 different housing features and more than 2000 economic features in the dataset, it has been a significant challenge to study and utilise the features to predict the housing prices. After the exploratory data analysis, we iteratively perform the steps in feature engineering, modelling and evaluation to achieve the best balance in the performance of the models. We also explore the use of gradient boosted decision tree algorithms from the LightGBM framework. Numerous self-learning and explorations on the nature and performances of algorithms have been done especially on the tuning of optimal hyperparameters.

1.4 Proposed Solution

We focus on data cleaning to remove unrealistic values and feature engineering to derive new relationships that have weightage on the housing prices. We propose the implementation of the models with gradient boosted decision trees. Multiple models with different feature engineering steps performed on each model will be used to contribute to the final prediction (ensemble methods). This will allow the models to deal with high variance of input data. By using the LightGBM framework, we can speed up the hyperparameter tuning and training steps with parallel and distributed processing.

2 Exploratory Data Analysis on Train and Test Set

To help us with the data cleaning feature engineering for the models, we first had to understand the train test and test set respectively.

2.1 Train Set

At one glance of the train set, the first thing that we understand is that a significant proportion of data is missing from different columns. Hence to understand the extent of how much data is missing, we plotted a graph as shown in the figure below on the percentage of missing values in the train set.

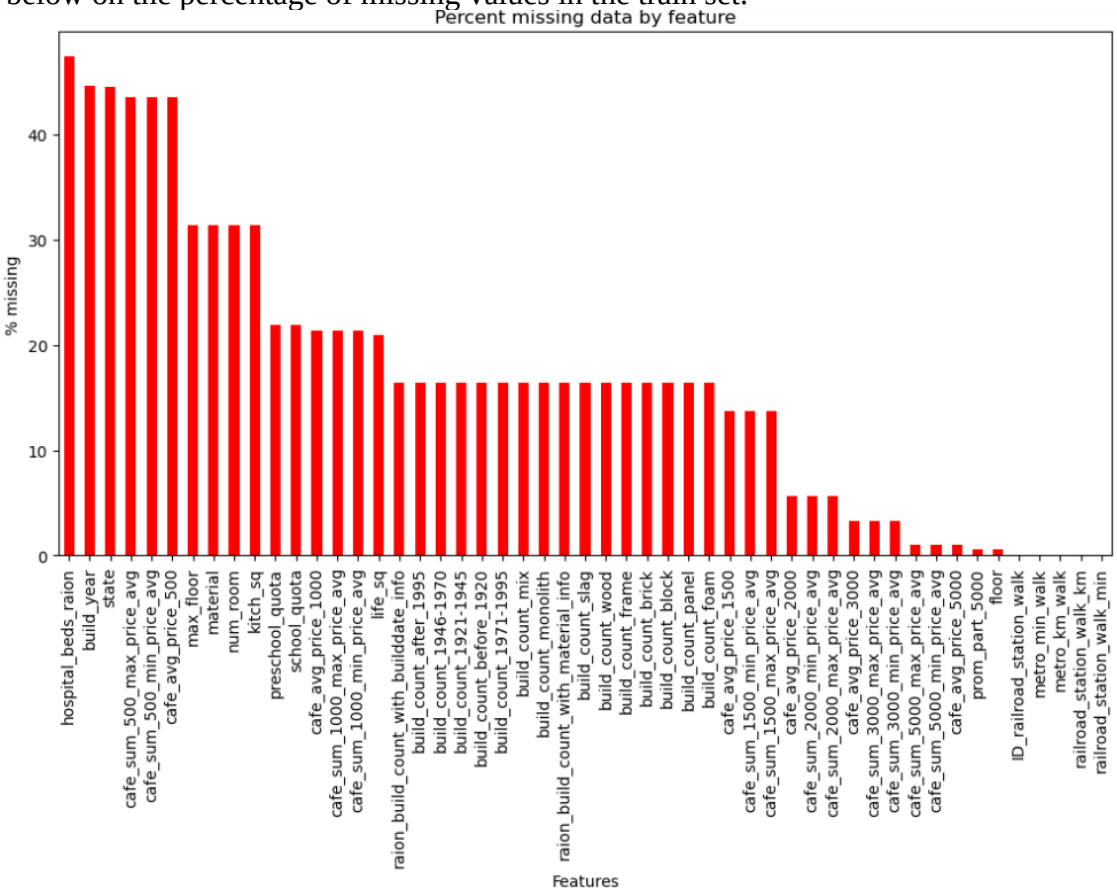


Figure 2.1.1: Percentage of missing data by feature

From the plot above we can understand that some features have a huge proportion of data that is missing. Hence when choosing our model, it was essential for our group to choose a model that could handle missing data in the dataset. Further on we also experimented in filling up these missing data to achieve better model results.

Acknowledging the potential existence of substantial erroneous data in our dataset, our focus shifted to identifying areas prone to incorrect information, crucial for refining our machine learning model.

Firstly, we found that there were a total of 37 observations where the total area in square meters which included loggias balconies and other non-residential areas (full_sq) was more than the living area in square meters which excluded the loggias, balconies and other non-residential areas (life_sq).

Upon closer examination at the feature that represented the year in which the properties were built in (build_year), we also realised that there were a few abnormal years such as -0,1,3,20,71,215,4965 and 20052009.

In relation to the property details itself, we also noticed that some of the data properties had labelled their number of rooms as 0 (num_rooms) an inconsistency we deemed incorrect as minimally a property should have at least one room. Furthermore, there were instances of some data points where the current floor (floor) where the property resided was larger than the maximum floor (max_floor) the property had. In some instances, we also found out that some of these properties have also been labelled their maximum or the floor they reside in to be 0, a contradiction given the minimum floor should be 1. Regarding the kitchen area of the property (kitch_sq), we also observed multiple instances where it exceeded or equalled the living area of the property itself (life_sq). Additionally, there were also some weird instances of data points which stated that the property where their kitchen area equaled 0 or 1 which technically should be impossible.

Upon identifying all potential erroneous data in the dataset itself, my team then directed our attention to exploring different features and the relation it has with the property price itself so that it may potentially help us improve our feature engineering when building our machine learning model.

We first decided to explore if there was any correlation between the internal home characteristics in our dataset and our target, price (price_doc) shown in the figure below:

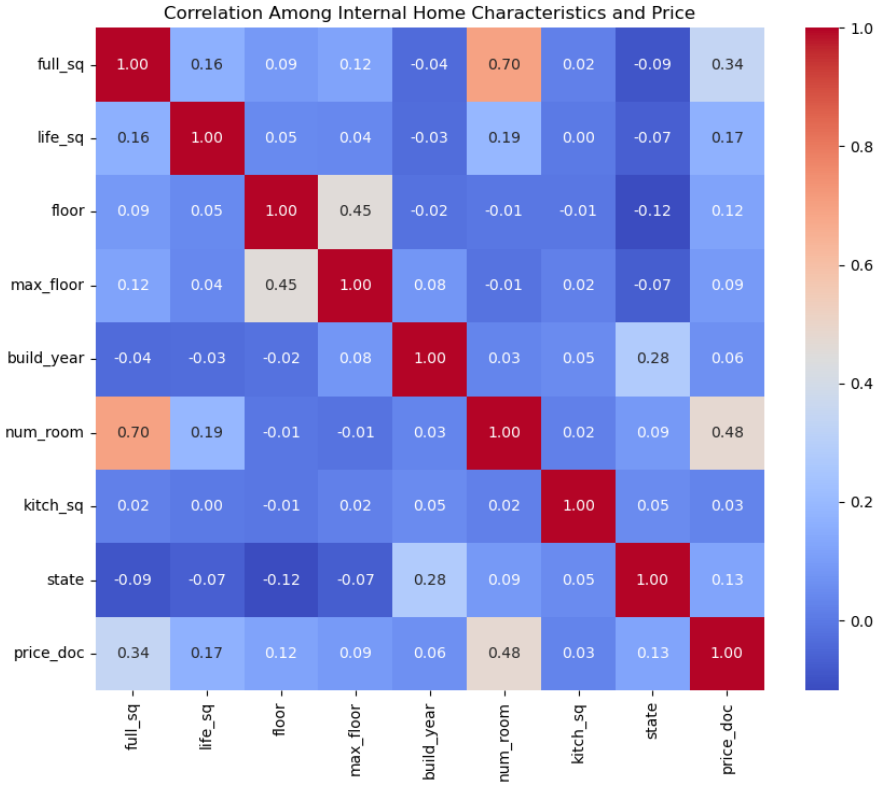


Figure 2.1.2: Correlation among internal home characteristics and price

From the above, we understand that there is still a substantial relationship between the respective features and the price of the property itself and hence most of the features mentioned above should be used later in our models itself and should not be omitted. It can also be observed that num_room has quite a high correlation with full_sq which we might be able to leverage on.

Another area we decided to look into would be the price with respect to the area of the property itself as it is logical to assert that a property's price should be directly correlated to the price of the property itself.

Figure 2.1.3: Price by Area in square meters

Upon plotting this graph we realise that there are data points that can be considered as outliers or even potentially some more data points in our train dataset that are erroneous which should not be considered in our machine learning model as it would most likely skew our results.

Similarly, we also looked into the state of the home as it should have a positive relation with the price of the property itself.

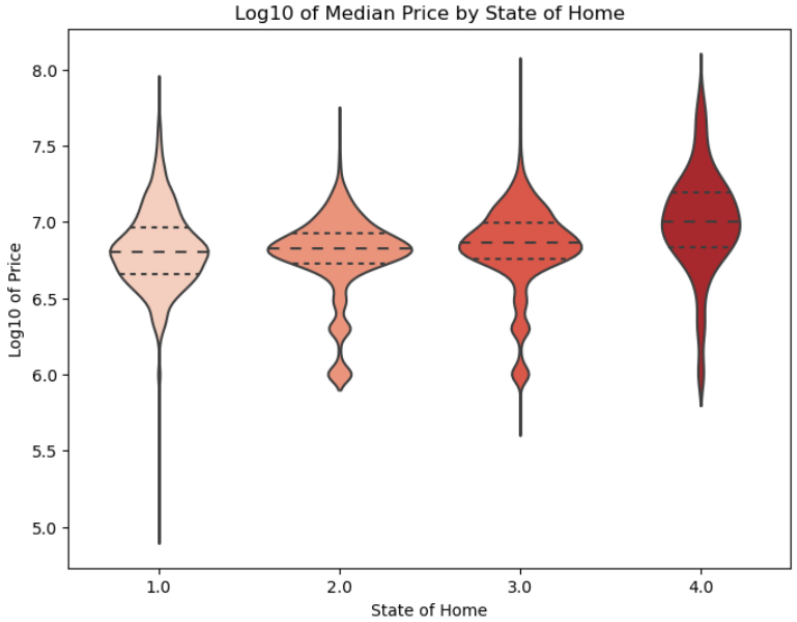


Figure 2.1.4: Median price by State of home

However in this instance, the data points in our train tend to align with our hypothesis and seem to indicate a positive relationship with the state of the property and the price of it.

Another hypothesis that we wanted to test was whether there is a significant price difference between homes bought by an owner-occupier and homes bought for investment.

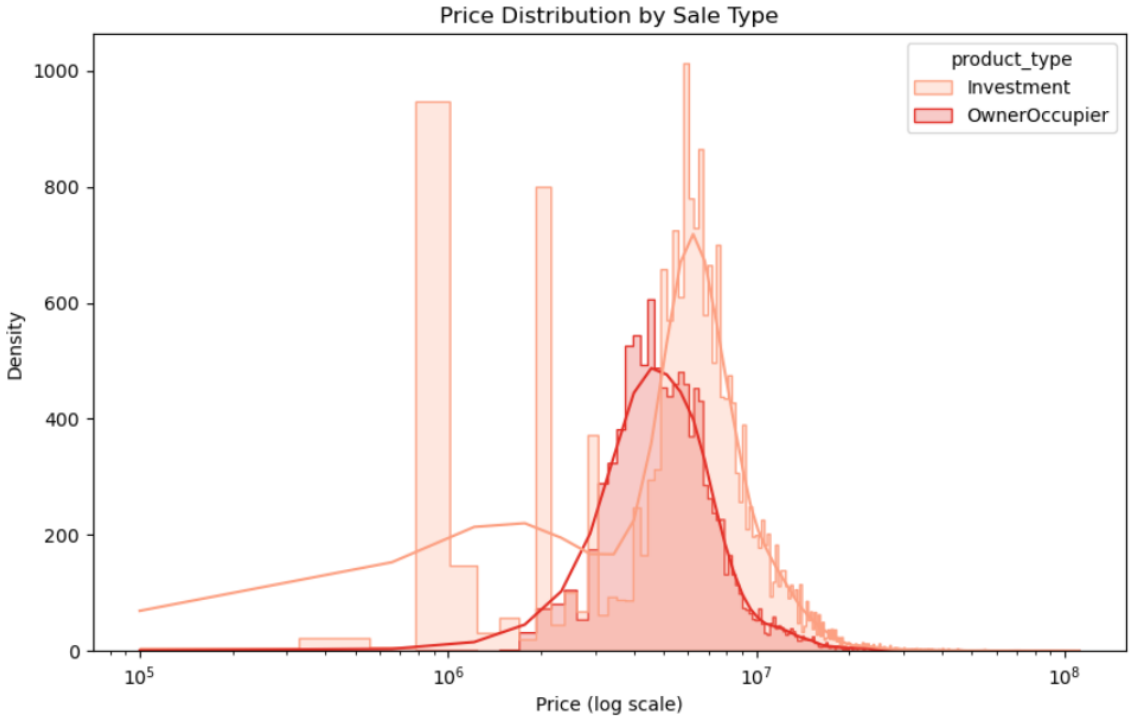


Figure 2.1.5: Price Distribution by Sale Type

Plotting their price distribution by the different sale types with respect to the price as shown above, we were able to understand that distinct price distributions exist based on the property's sale category. Hence with that observation, one thing we decided to experiment on in the later stages was to split our model into two, training a model individually for investment and owner-occupier property types in the hope of improving our overall model accuracy.

We also decided to plot the property's build in relation to the average price to see if there was any potential peak period that we had to potentially look out for when it came to building our model.

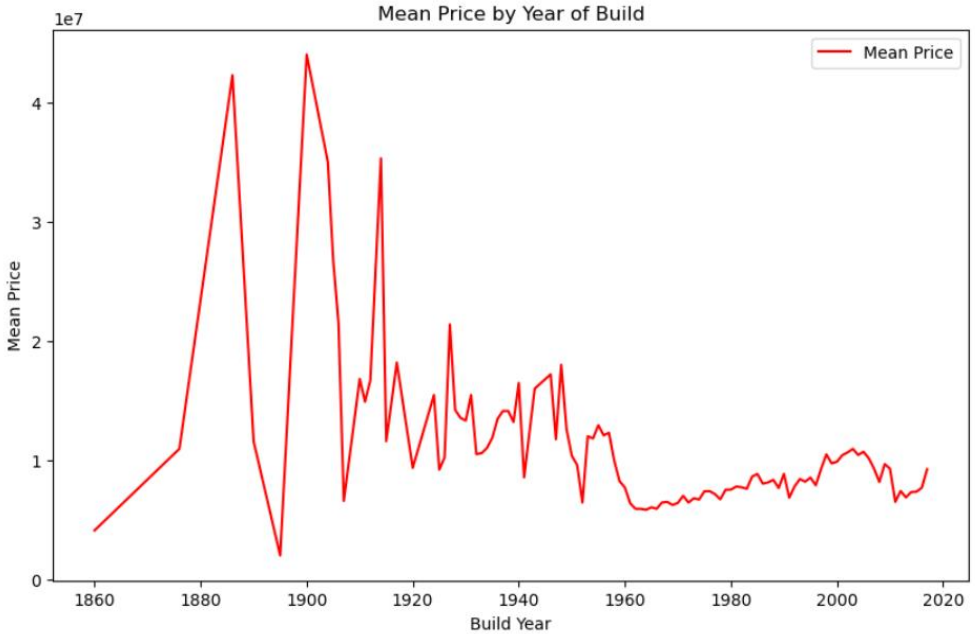


Figure 2.1.6: Mean Price by Year of Build

From the graph, we understood that the relationship is somewhat steady over time. There is some volatility in the earlier years of the data but however, we realise that it was most likely due to the sparseness of observations up till the year of 1950. To substantiate our findings we also plotted the daily median price over time,sales volume over time and distribution of build year shown in the appendix of our report.

To understand the possibility of a seasonal component within the year itself, we also decided to plot the median price by month of the year as shown below.

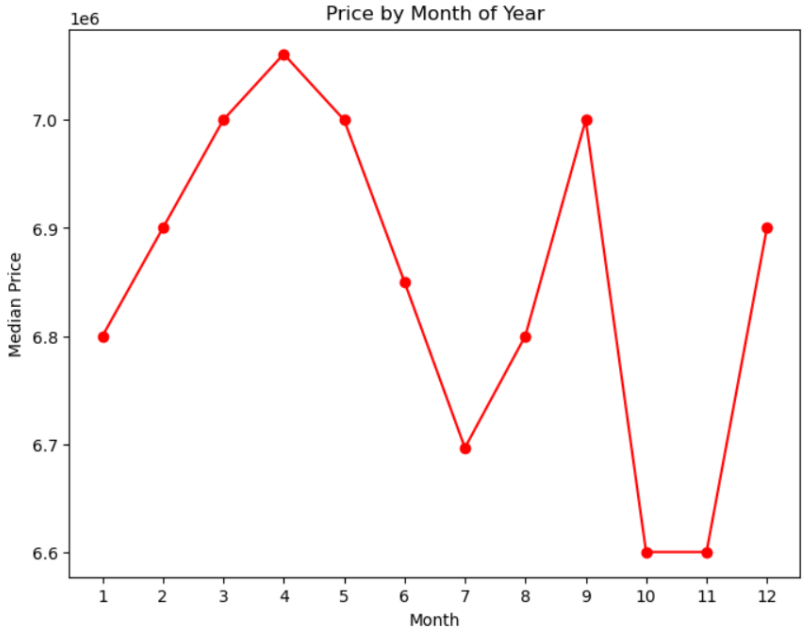


Figure 2.1.7: Median Price by Month of Year

From this we deduced that there is a noticeable seasonal period within the year or across the year as a whole that we should look into when building the machine learning model.

In addition to all these observations, we also look at potential features such as build material, population density, the share of working age population, school characteristics, number of top 20 universities in raion, cultural and recreational characteristics, number of sports objects, number of culture objects, distance to the nearest park, infrastructure features and distance to Kremlin and their relation to the price of the property itself but did not identify any potential areas that we had to look out for in our machine learning model. For detailed reference and evidence, the corresponding analyses and plots are meticulously documented in the appendix of our research.

2.2 Test Data

Similarly to the train data, we also have similar percentages in missing data and erroneous data. Hence we decided that we would most likely clean the test data set very similarly to our train data set in our machine learning models to ensure that we are able to get higher accuracy when submitting our test results.

In terms of the distribution of the dataset itself, we have identified that most of the features have very similar distribution and hence there should not be any issue in transferring our model from our train set to our test set itself.

3 Machine Learning models

3.1 LightGBM Model

3.1.1 Introduction of LightGBM

LightGBM is a gradient boosting algorithm that uses tree based learning algorithms. LightGBM has faster training speed and uses less memory space as compared to other tree based algorithms. This is because many boosting tools use pre-sort-based algorithms for decision tree learning which is not easy to optimize. On the other hand, LightGBM uses histogram-based algorithms which bucket continuous feature values into discrete bins, reducing the cost for calculating the gain for each split.

LightGBM also provides more accurate results as compared to other tree based algorithms. LightGBM grows trees leaf-wise, while other decision trees grow level wise. If the number of leaves remains constant, leaf-wise algorithms have a lower error rate than level-wise algorithms. Furthermore, LightGBM chooses the optimal split for categorical features. Categorical features are usually represented by one-hot encoding, but this method is suboptimal for tree based algorithms because the tree tends to be unbalanced and needs to grow very deep to achieve good accuracy. LightGBM splits on a categorical feature by partitioning its categories into 2 subsets, and it finds the optimal partition according to the training objective at each split. It sorts the histogram for a categorical feature according to its accumulated values, and then finds the best split on the sorted histogram.

3.1.2 Implementation of LightGBM

In our original model, we performed data cleaning and feature engineering from observations collected in the initial data analysis. As there are several erroneous values, we had to replace them with a suitable estimate or with NaN. Given that many rows have NaN values, we figured it would be a better idea to give them an estimated figure rather than removing them. The proposed methods are listed below.

<u>Data Cleaning</u>

1. state should be discrete valued between 1 and 4. There are data entry errors where they are outside of this range, we replace them with the mode of state.
2. build_year has an erroneous value 20052009. Replace it with 2007
3. There are observations where floor is greater than max_floor. Set all to NaN.
4. Subsequently, replace all NaN floor values with the median floor in the sub_area (Assuming the max floor limits in the sub area is the same, they would have similar floor values)
5. Replace all NaN num_room values with the num_room value from the closest full_sq. (Assume that we can get a good estimate of num room from full sq given that corr score=0.7 between full_sq and num room)
6. Replace nonsensical build years with NaN , 1691<= build_year or build_year>=2018
7. Replace all NaN build_year with median build_year in the same sub_area (Assuming the build year in the sub area is roughly the same)

<u>Feature Engineering</u>

1. There seems to be a trend with seasons. Added months column
2. Group data according to num_room, build_year, floor, sub_area and calculate the full_sq mean for the groups. Create a column full_sq_mean. Create column normalised_full_sq full_sq/full_sq_mean.
    a. This is to reduce the fluctuations in the full_sq, there are many bad values in full sq.

Remove the bad prices

- Remove price lesser than 1 million, price == 2 million, price == 3million
- After studying the Housing market prices in Russia, it was found that recent (past 10 years) housing prices rarely fell below 1 million. Hence prices below 1 million might not provide useful features for predicting recent housing prices. Housing prices at exactly 2 million and 3 million were removed as it is unusual.

Price adjustments

- Adjust investment price for build_year <2000 with 0.9 multiplier
    - down scale only the "old" investment properties, as the new ones are supported by the mortgage subsidy program

Remove columns with rare features

- Perform 1 hot encoding for Categorical columns
- Find the sum of proper values (not NaN) for the Category. Remove categories with lesser than 200 values

Hyperparameter tuning

We perform 3 fold validation to tune our hyperparameters. There were 6 parameters in total that were considered important for the LightGBM model. Here are the results of the hyperparameter tuning.

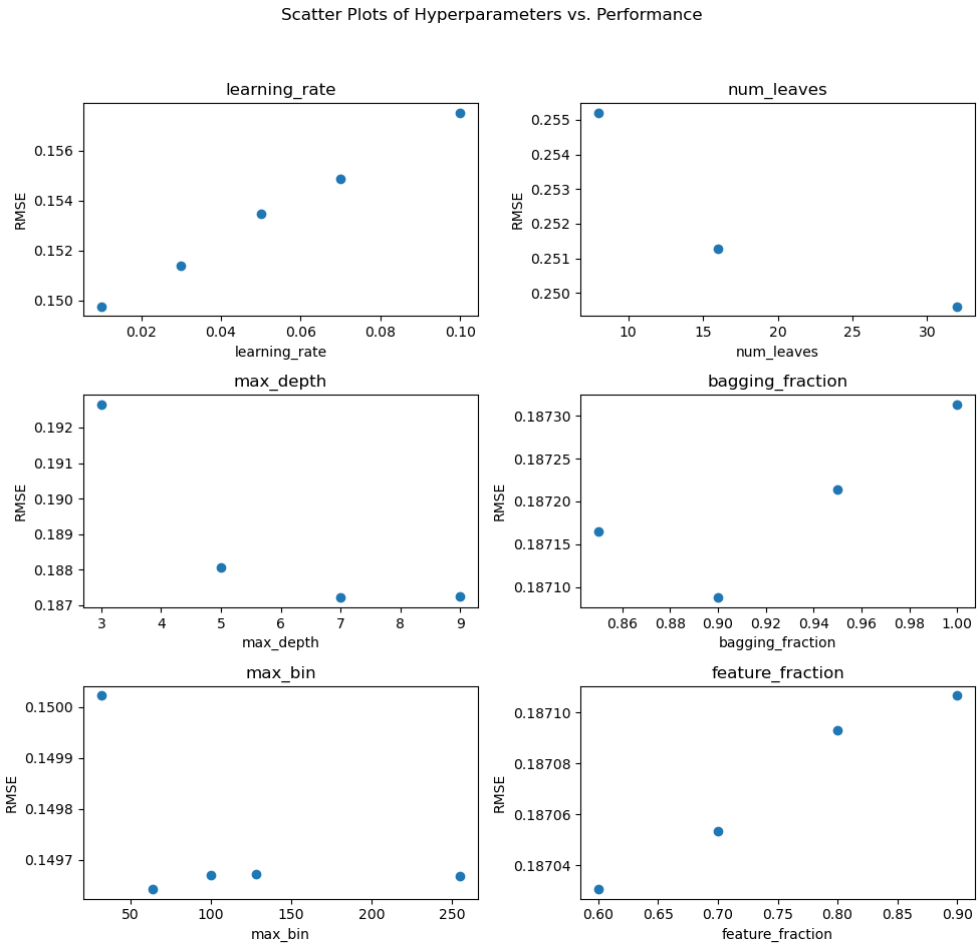


Figure 3.1.1: Respective Scatter plots of hyperparameter vs performance

The optimal hyperparameters:

- Learning_rate : 0.01
- Num_leaves : 32
- Max_depth : 7
- Bagging_fraction : 0.9
- Max_bin : 64
- Feature_fraction : 0.6

These will be the parameters for our subsequent models.

This model(model 1) placed us on the leaderboard position 653 with a public score of 0.31069.

Submission and Description	Private Score	Public Score	Selected
submissions.csv Complete (after deadline) · 11h ago	**0.3138**	**0.31069**	☐

However, in a SHAP analysis, it was found that our model relies heavily on full_sq to predict prices

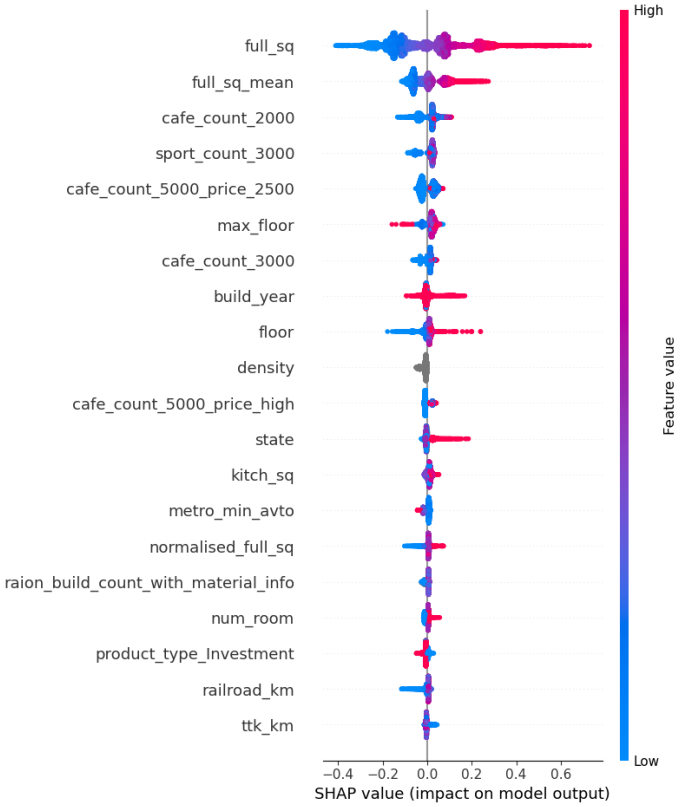


Figure 3.1.2: SHAP analysis on feature dependence

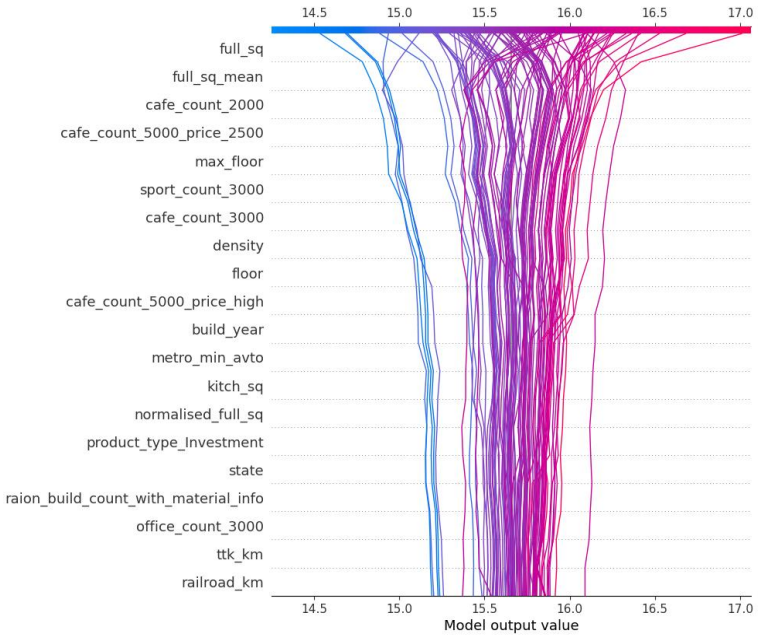


Figure 3.1.3: Score influence by various features

This can be problematic as full_sq has many outliers or erroneous values that can be difficult to pinpoint and remove (Figure 11). This should make it, in theory, unreliable as a feature to predict price.

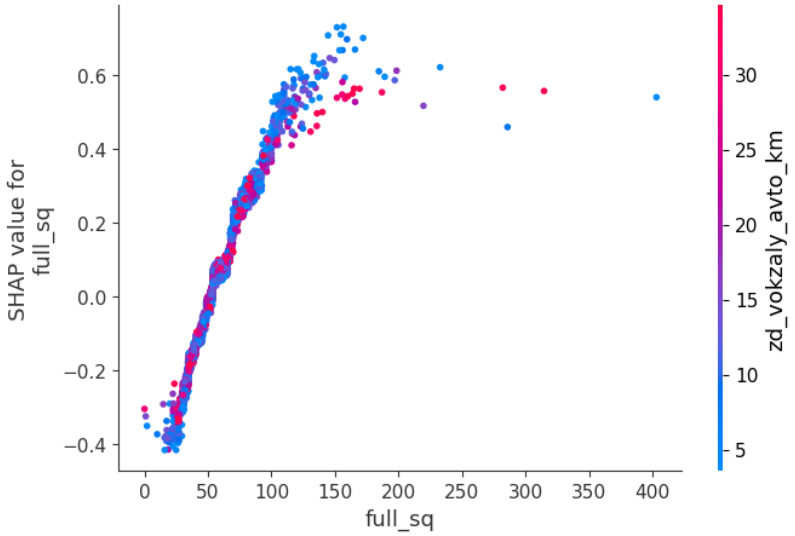


Figure 3.1.4: SHAP dependence for full_sq

The addition of full_sq_mean improved the accuracy significantly by stabilising such fluctuations in full_sq. However, it does not eliminate the influence of said bad values in full_sq. As such, we decided to train new models to cover up this particular weakness of our model. We will look into models that rely on different features to improve the robustness of our overall model predictions, and combine the results with this model.

3.2 Ensemble

To improve our overall accuracy, we looked toward ensemble learning and combining 3 different LightGB models to create a more accurate and robust decision than any of the individual models could achieve on their

own. By learning different aspects of the data and combining the models' predictions, we can capture multiple perspectives and lower the error rate of the overall combined model.

For ensembles to work, we need to ensure that the base classifiers are not perfectly correlated with each other. We have done so by ensuring that the 3 models use different training datasets, different features and different prediction methods.

Firstly, we changed the training datasets to be different from the dataset we used to train model 1. Fortunately for us we have found an updated post where the competition hosts released new data to correct some of the training and test data. The new data focuses on correcting the coordinates for some properties, which in turn affected the values for some features, such as the distance to the city centre [3]. Unfortunately, the potential erroneous data that was found in our data was still present. Hence in addition to a corrected dataset released by the competition host, we also decided that we would handle the erroneous data differently for models 2 and 3. Thus, all 3 models are uses slightly different training sets.

As compared to model 1, these are the different data cleaning steps that we used for models 2 and 3

1. Changed living areas in square meters(life_sq) to NaN for train and test data with life_sq more than total are in square meters (full_sq) as we explored in the EDA that these data points should be incorrect
2. Changed life_sq to NaN for train and test data with life_sq less than 5 as we felt that it was almost impossible for an apartment in Russia to have a living area less than 5 square meters
3. Changed full_sq to NaN for train and test data with full_sq less than 5 for reasonings similar to the previous change
4. Changed life_sq and full_sq to NaN for train data with life_sq more than 300 as during the time period of which the train set was taken it seemed impossible for a house to have more than 300 square meters
5. Changed life_sq and full_sq to NaN for test data with life_sq more than 200 for reasoning similar to the previous change but because due to the difference in the time period, we felt that it would be impossible for an area more than 200 square meters
6. Changed kitchen area (kitch_sq) to NaN for train and test data with kitch_sq more than or equal to life_sq as shown in EDA should be impossible for the kitchen to be bigger than the living area itself
7. Changed kitch_sq to NaN for train and test data with kitchen_sq equal to 0 or 1 as we felt that it would be impossible there to be no kitchen area or just a kitchen area that equalled 1 square meter
8. Changed year of building built (build_year) to NaN for train and test data with build_year less than 1500 as it was clarified by the hosts that any properties with year indicated to be before 1500 were erroneous.
9. Changed number of rooms (num_room) to NaN for train and test data with num_room that equal to 0 as minimally a house should have at least one room itself
10. Changed maximum floor (max_floor) to NaN for train and test data with max_floor equal to 0 as minimally the floor a building should have should be 1
11. Changed floor to NaN train data with floor equal to 0 for similar reasons as the previous
12. Changed max_floor to NaN for train and test data with floor more than max_floor as it would be absurd for a property to reside in a floor higher than the maximum floor
13. Changed state to NaN for the training data property that has state 33 as all the other properties only had states from 1 to 4.
14. Changed life_sq to NaN for data where full_sq == life_sq as it seems extreme that properties have no non-residential areas, such as balconies and loggias.
15. Changed full_sq to 50 for data with full_sq is null, as it is impossible for a house to have no space.
16. Dropped ecology features as it does not seem to be a significant factor in affecting the price.
17. Dropped sub_area feature as it does not seem to be a significant factor in affecting the price.
18. Changed product_type to 1 (investment) for data with product_type equal to null, as we assume that properties will at least have one type of use.

Then, we moved on to changing the features for the different models. For the feature engineering in models 2 and 3, we applied these changes to both the train and test data

1. Added relative floor (floor / max_floor), as it seems reasonable that people will consider the relative floor when deciding on a price.
2. Added relative kitchen sq (kitchen_sq / full_sq), as the kitchen size seems to be an important consideration for the Russians when deciding on a property.
3. Added room size (life_sq / num_room), as it seems reasonable that the size of a room affects the property prices.
4. Added month from the timestamp attribute, as the month when the property was listed could affect its selling price.
5. Added day of week, as the day when the property was listed could affect its selling price.
6. Added "bought minus built" (year bought - year built), as the age of the property could affect its selling price.

For both models 2 and 3, we also modified the prices with regards to housing price changes between each quarter. The price for each quarter is calculated as such:

price of quarter n = price of quarter (n + 1) / Rate of price changes

The rate of price changes is detailed in the table below:

Year	Quarter	Rate of price changes
2015	2	1
2015	1	0.9932
2014	4	1.0112
2014	3	1.0169
2014	2	1.0086
2014	1	1.0126
2013	4	0.9902
2013	3	1.0041
2013	2	1.0044
2013	1	1.0104
2012	4	0.9832
2012	3	1.0277
2012	2	1.0279
2012	1	1.0279

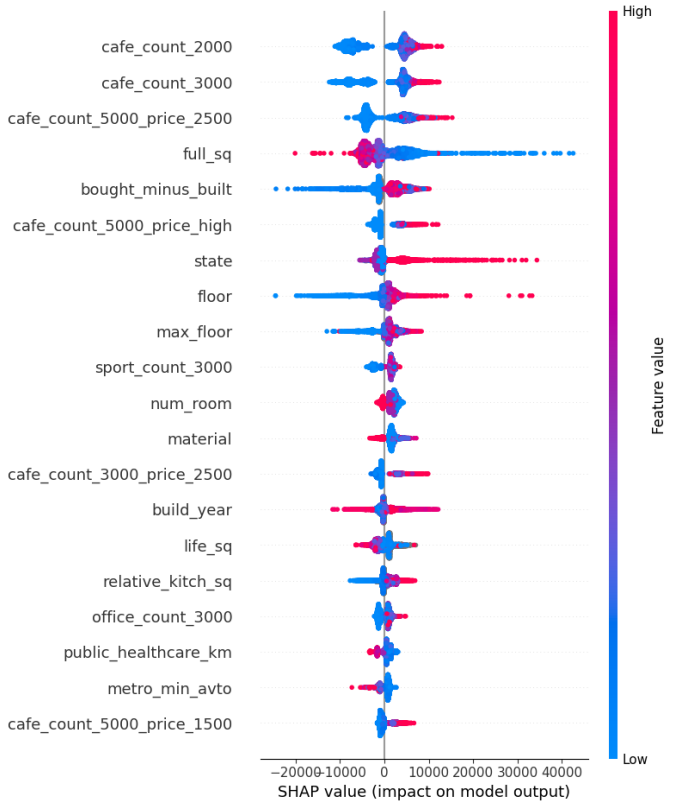
2011	4	1.076
2011	3	1.0236
2011	2	1
2011	1	1.011

These price factors were determined from a detailed analysis on the price conditions at that time that was done by other competitors and published on the competition's discussion board. The link to the discussion is shown in the references section [4].

In addition to data cleaning and feature engineering changes above, models 2 and 3 also use a different approach in predicting the price. Firstly, model 1 is trained on all housing types, be it for Owners or for Investment. On the other hand, models 2 and 3 train one LGB model each for homes used for Investment and homes used for owner occupation. The training data is split into homes for investment and homes for occupation respectively, using the values indicated in the "product_type" column. The data are then fed into their respective LGB models for training. The same splitting process is done on the test data, before the respective LGB models take in the test data and perform their predictions. The results from both LGB models are then combined to form the final result.

Models 2 and 3 are also different with regards to the predictions they make. Model 2 computes the predicted price per full_sq while model 3 computes the unmodified predicted price. To compute the predicted price per full_sq, we first divide the target price for the training data by their respective full_sq values before fitting the LGB model with it. After prediction, we then multiplied the results with the full_sq of the corresponding data to get the final results for submission.

3.2.1 SHAP Analysis for model 2



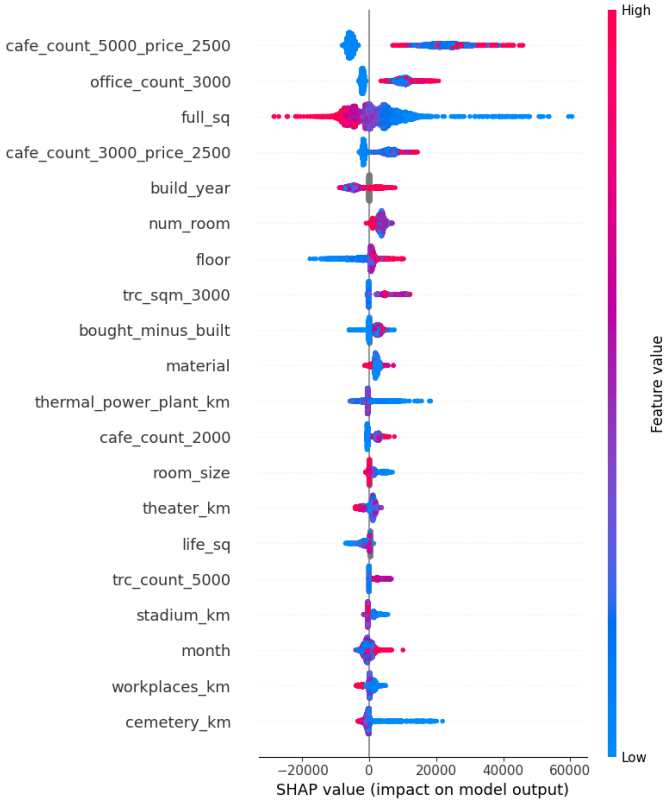
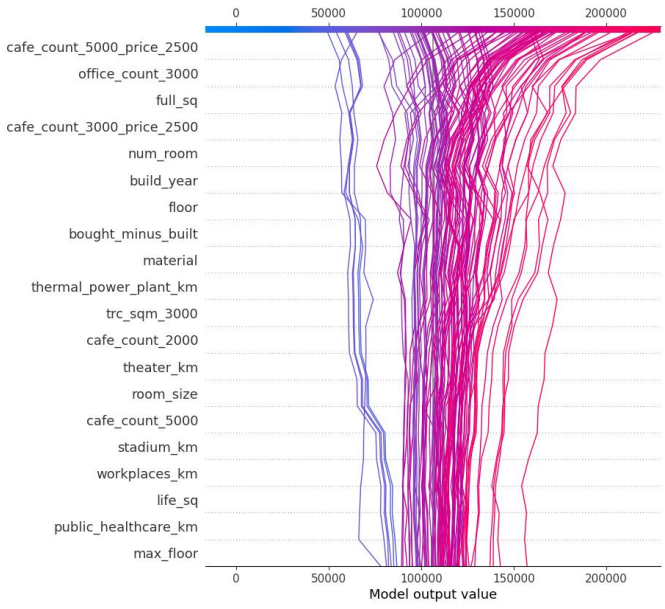


Figure 3.2.1 (Left): SHAP Analysis for Model 2's investment LGB model
Figure 3.2.2 (Right): SHAP Analysis for Model 2's owner LGB model

From the SHAP analysis in Figures 3.2.1 and 3.2.2, the models now rely less on the full_sq feature. The effects on the model's prediction is now roughly spread out among many features, such as the number of cafes and offices around. This is good as it shows that we were able to take away the heavy reliance on the full_sq feature to predict the price, and the price would be less likely to be skewed by the full_sq feature.



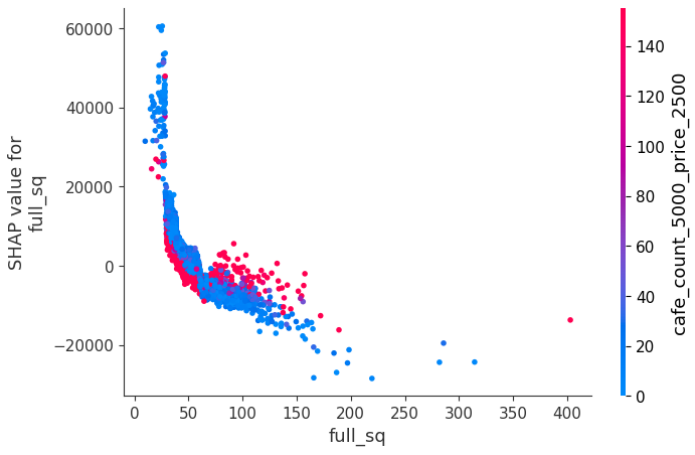
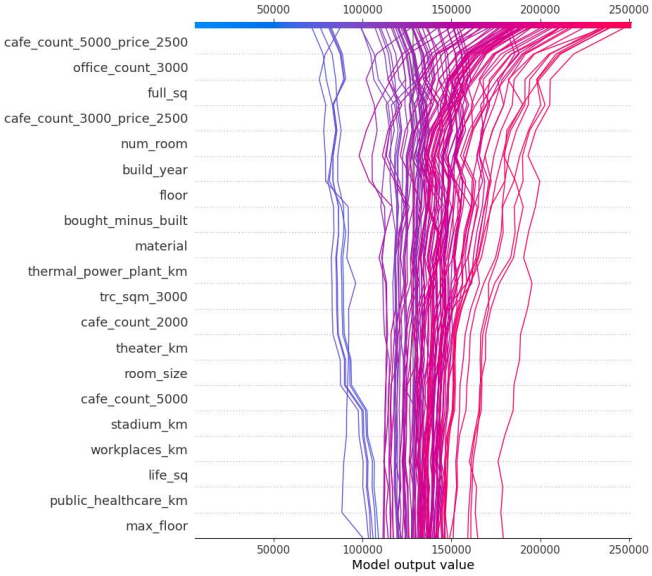


Figure 3.2.3 (Left): Score influence by various features for model 2's owner LGB model

Figure 3.2.4 (Right): Influence of full_sq on score for model 2's owner LGB model



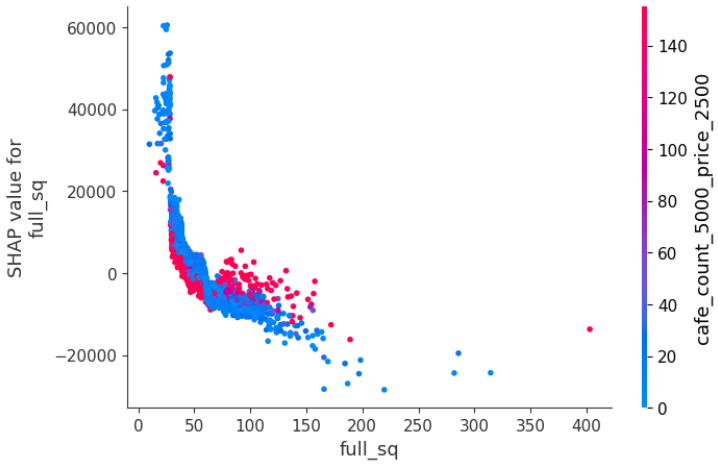
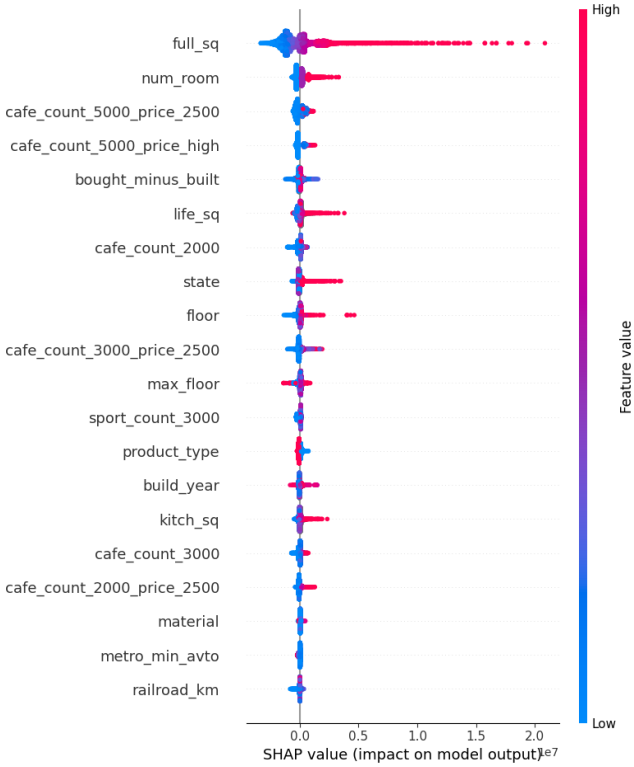


Figure 3.2.5 (Left): Score influence by various features for model 2's investment LGB model

Figure 3.2.6 (Right): Influence of full_sq on score for model 2's investment LGB model

Figures 3.2.3 to 3.2.6 also confirm that model 2 has less reliance on the full_sq feature now, and SHAP value for full_sq is now centered within a small range from -20000 to 20000. This is in comparison with model one, where the SHAP value for full_sq was widely spread out.

3.2.2 SHAP Analysis for model 3



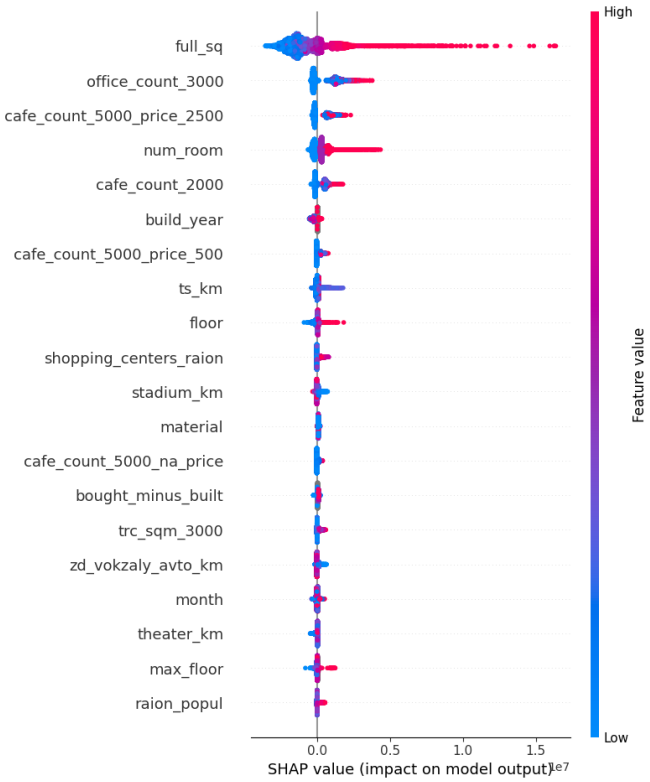
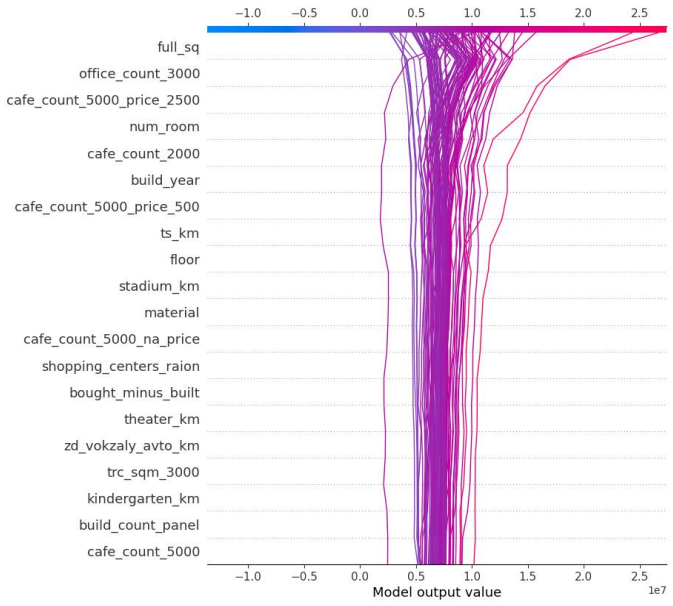


Figure 3.2.7: (Left): SHAP Analysis for Model 3's investment LGB model
Figure 3.2.8: (Right): SHAP Analysis for Model 3's owner LGB model



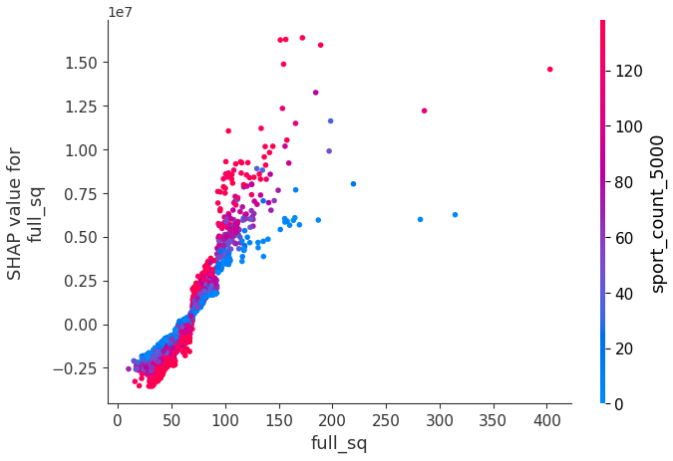
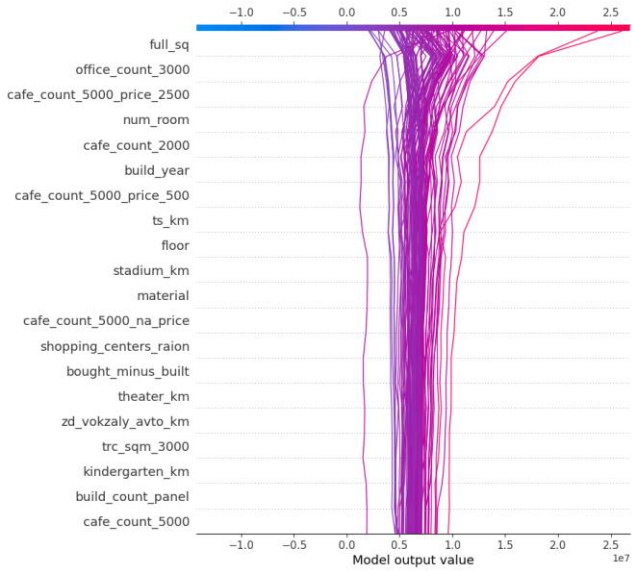


Figure 3.2.9 (Left): Score influence by various features for model 3's investment LGB model
Figure 3.2.10 (Right): Influence of full_sq on score for model 3's investment LGB model



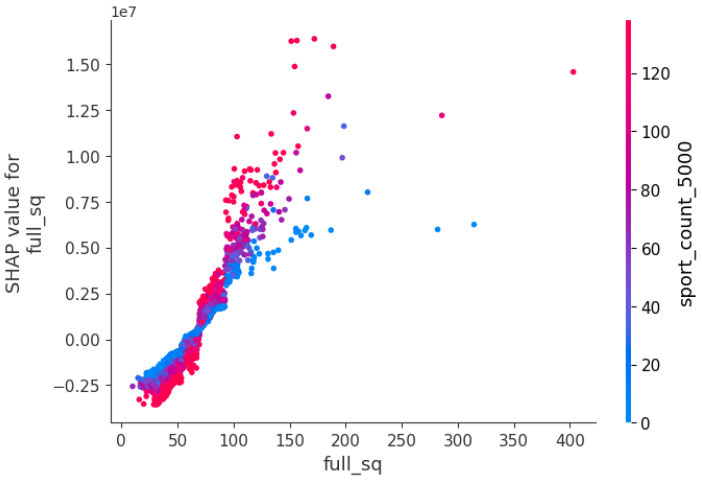


Figure 3.2.11 Influence of full_sq on score for model 3's owner LGB model

Figure 3.2.12: Score influence by various features for model 3's owner LGB model

Interestingly, from Figures 3.2.7 to 3.2.12, full_sq is the most dominant feature again when affecting the price. However, the SHAP value for full_sq is not as spread out compared to model 1. Each feature now also has a smaller range of SHAP values compared to model 1. This could indicate that each feature is now more accurate and consistent in predicting the price, and the price would be less likely affected by extreme values in the features.

3.2.2 Results

After getting the results for model 1, we calculated the RMSLE using the validation set and got an RMSLE of 0.2 and the LB score of 0.31069.

After getting the results for models 2 and 3, we first submitted their predictions individually to the competition to observe the results from both models. Model 2 was able to achieve a score of 0.3109 whereas Model 3 was able to achieve a score of 0.31157. This was around the same score as Model 1.

model_1_output_lgb.csv Complete (after deadline) · 1d ago	0.31449	0.31104
model_3_output_lgb.csv Complete (after deadline) · 1d ago	0.3152	0.31157
model_2_output_lgb.csv Complete (after deadline) · 1d ago	0.3138	0.3109

We then combined the results from all 3 models by computing the weighted average, with each model having a weightage of 33%. The final result was able to achieve a score of 0.30775, which was significantly better than each of the 3 models individually. The score allows us to be placed at around 12th position in the leaderboard.

submission.csv **0.31043** **0.30775**

Complete (after deadline) · 2m ago · final lgb version

4 Conclusion

In conclusion, our machine learning model comprises various steps. We first performed exploratory data analysis to find out trends and outliers in our dataset, which is where we found out that we should be using all of the features in the dataset to train our model and that the prices for investment homes fetch a higher price than homes by owner occupiers. We then moved on to conduct data cleaning to remove outliers and adjust the wrong data, improving data quality. Then, we moved on to model building.

Our model consists of an ensemble of 3 models, all of which uses LightGBM. LightGBM is well-suited for tasks involving regression, including predicting housing prices. LightGBM is a powerful gradient boosting algorithm with high accuracy. However, it is important to be aware of the limitations of our model 1, such as its reliance on the full_sq feature, which is prone to outliers and erroneous values. To address this issue, ensemble learning can be used to combine predictions from multiple LightGBM models, each trained on a different dataset or using different features. Furthermore, the 3 models have different algorithms. Due to the price difference in investment homes and homes by owner occupier, for models 2 and 3, we reflected this difference by training different models for different owner types, before combining both results. Model 1 has only one model trained on all owner types. This approach can help to reduce the overall error rate and improve the robustness of the predictions.

Finally, we evaluated our model by computing the RMSLE score using a validation set and finding out the final LB score. Through ensemble, we managed to achieve our goal which is to get a higher accuracy through the ensemble of models as compared to the individual models. We achieved a LB score of 0.30775 which places us at the 12th position on the leaderboard.

References

[1] Sberbank. (2017). *Sberbank Russian Housing Market*. Kaggle. https://www.kaggle.com/c/sberbank-russian-housing-market

[2] LightGBM. (2023). *Welcome to LightGBM's documentation!*. https://lightgbm.readthedocs.io/en/stable/

[3] Sidorova, A. (2017). *Additional data - Tverskoe issue*. Kaggle. https://www.kaggle.com/c/sberbank-russian-housing-market/discussion/34364

[4] Global Property Guide. (n.d.). *House Price Trends by Country*. https://www.globalpropertyguide.com/home-price-trends#russia

Appendix

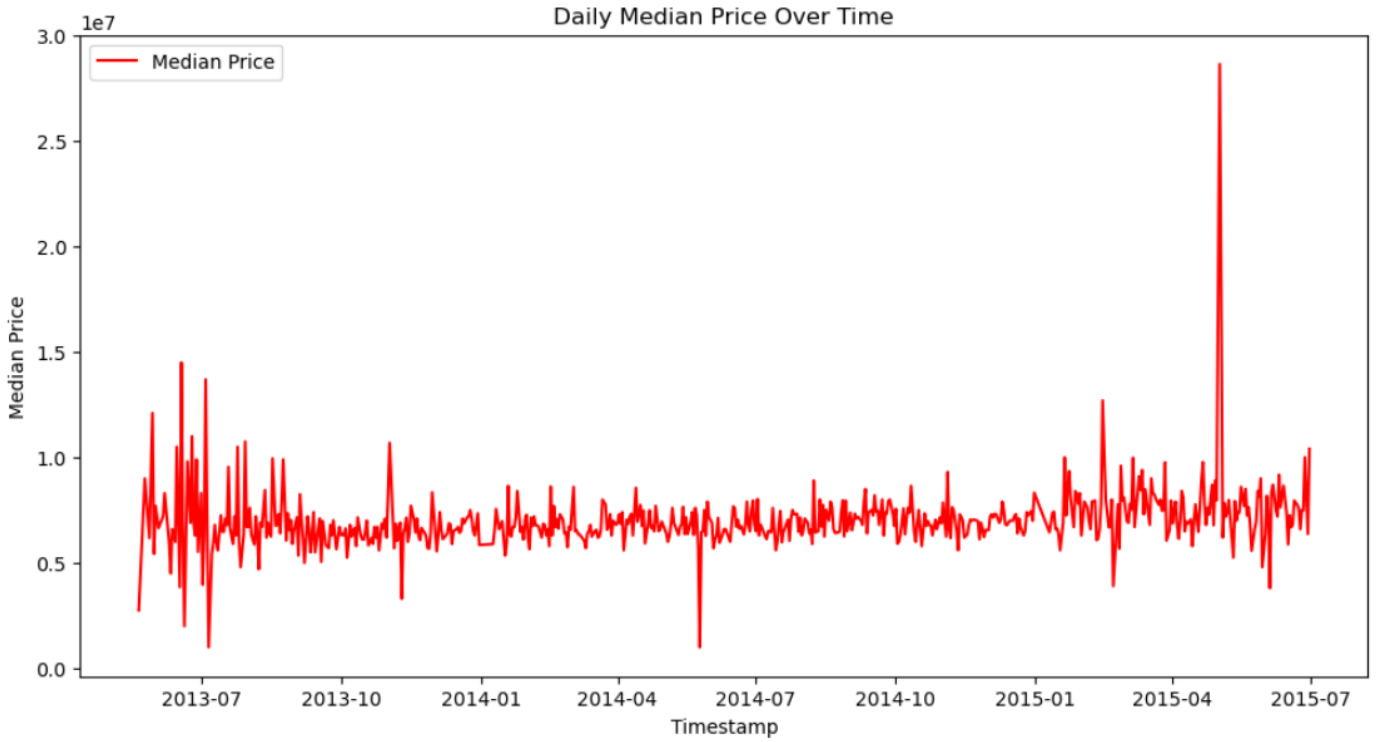


Figure 1: Daily mean price of property over time

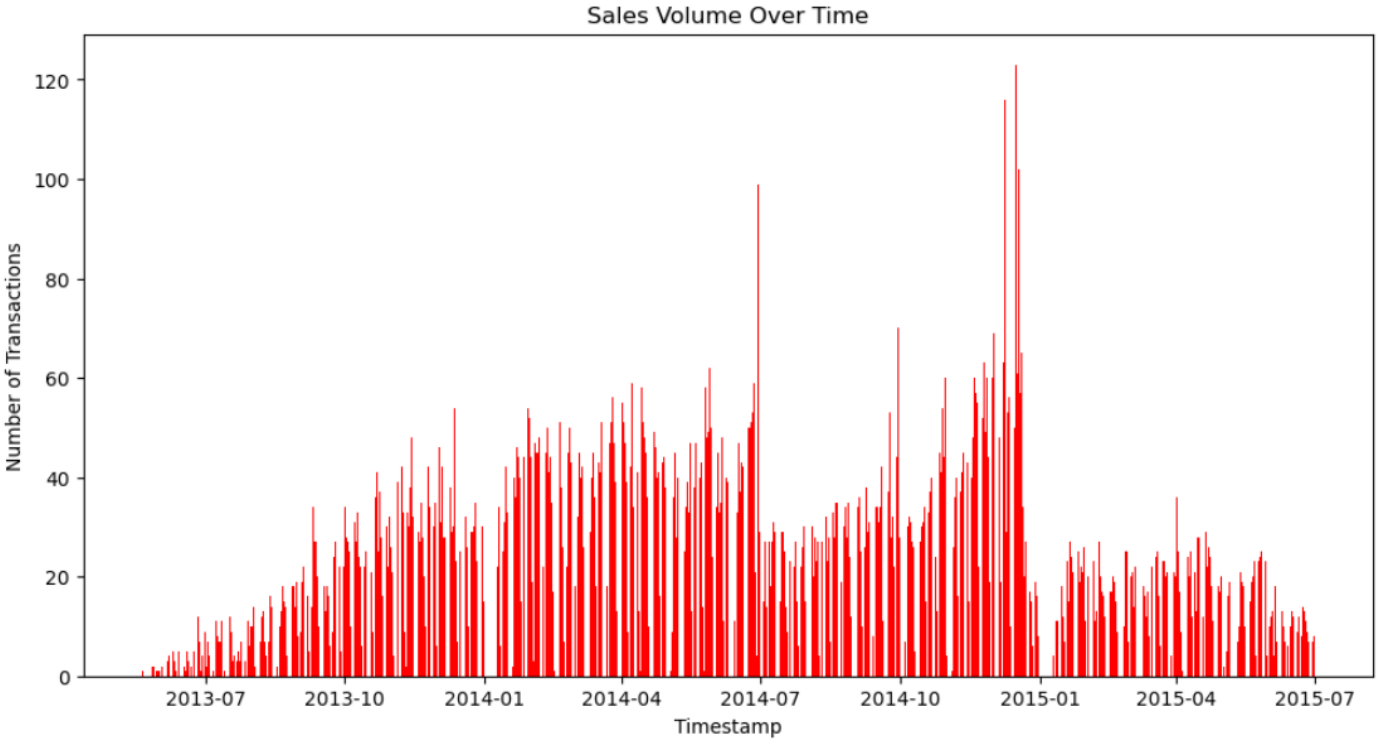


Figure 2: Seasonal Volume over time

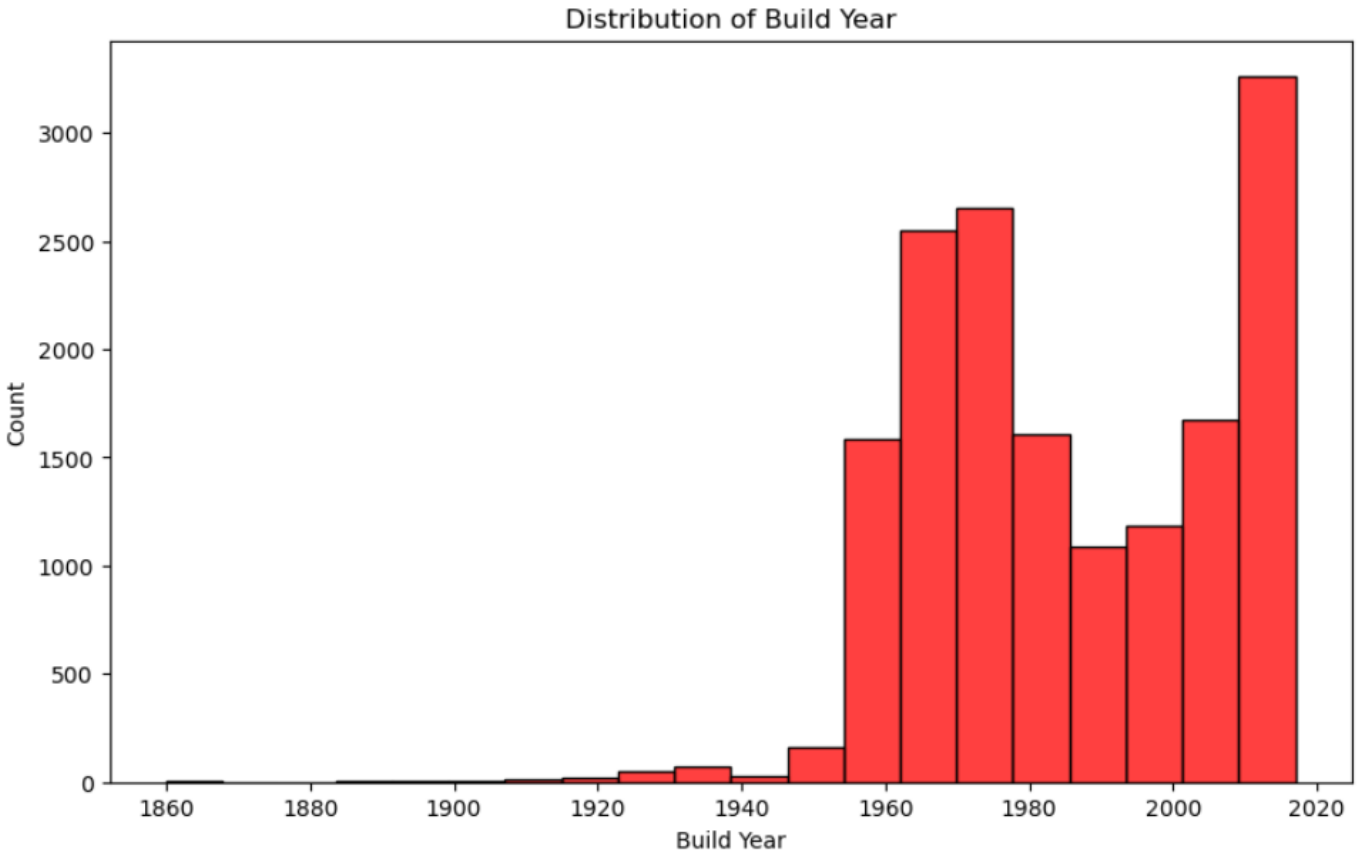


Figure 3: Distribution of Build Year

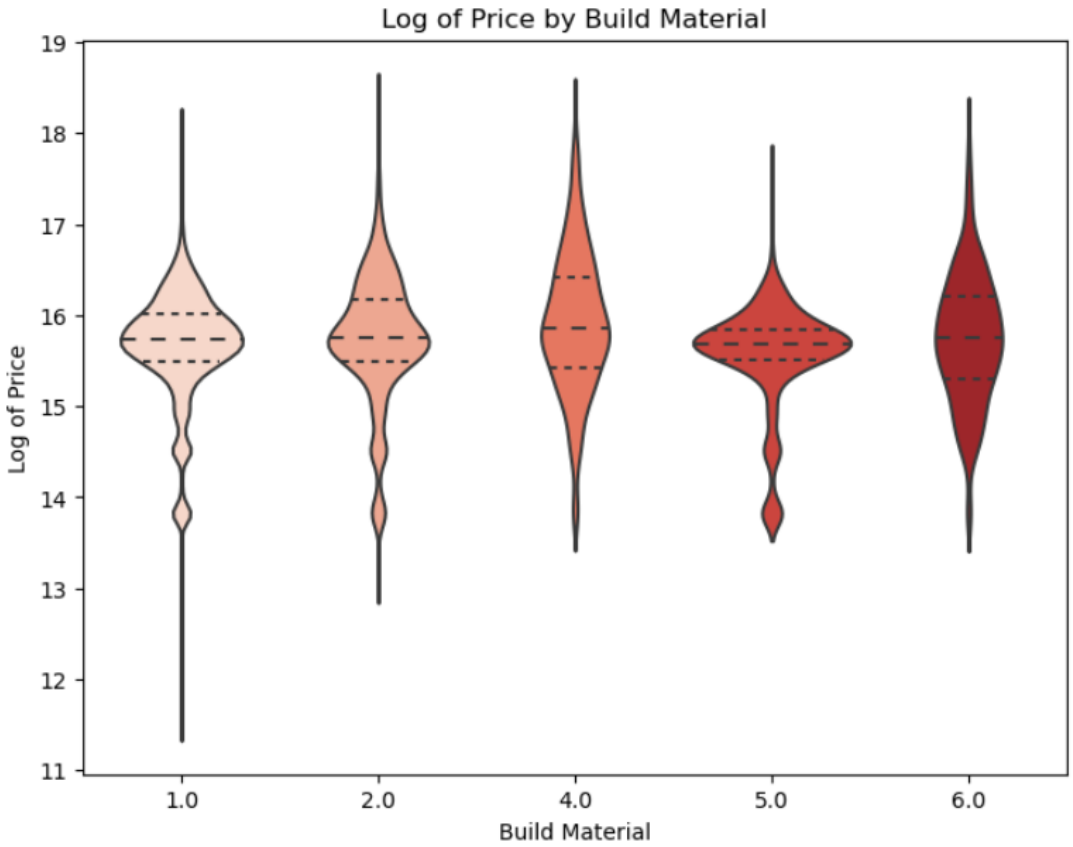


Figure 4: Price by Build Material

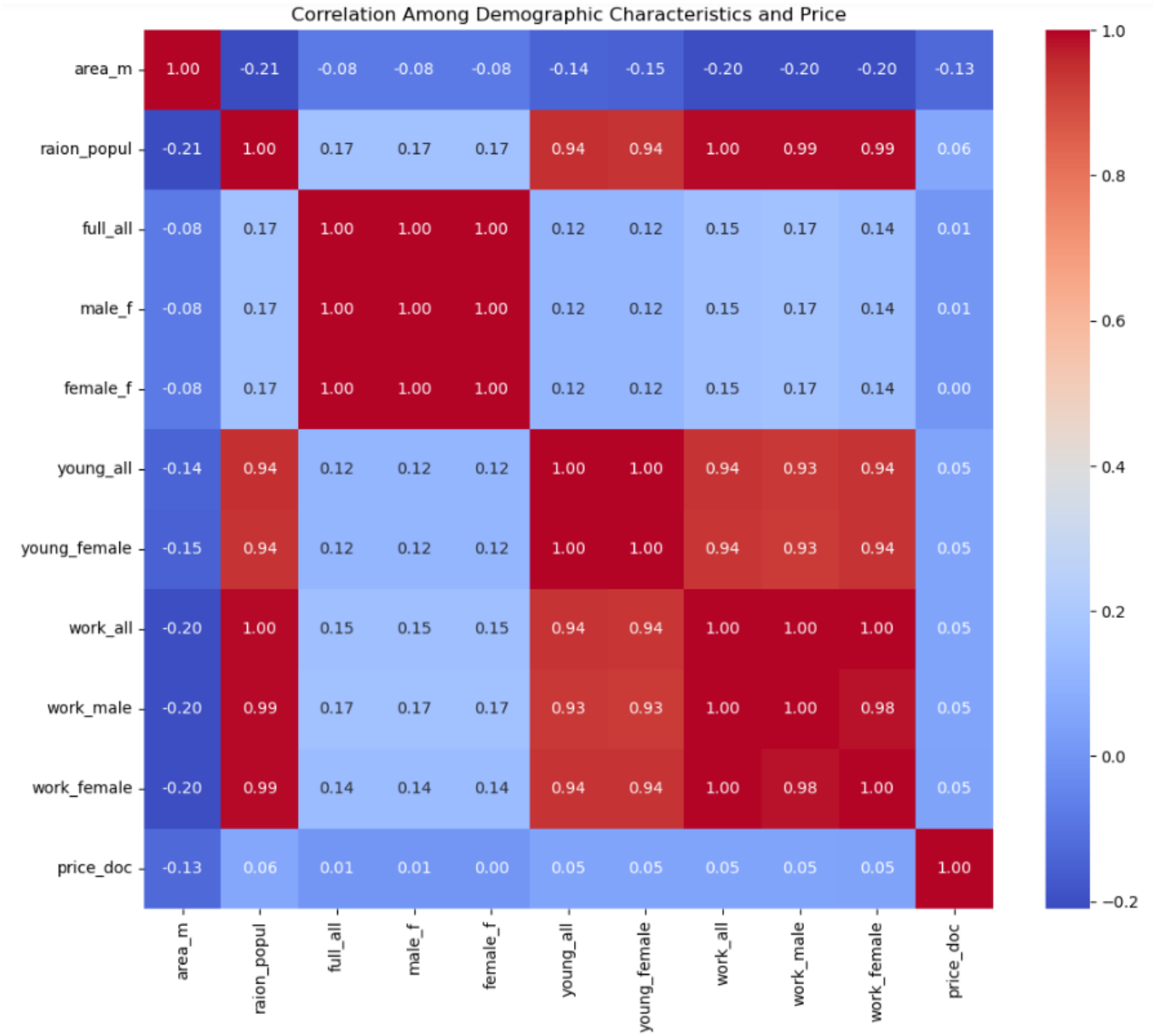


Figure 5: Correlation between Demographic characteristics and Price

Figure 6: Median Home Price by Raion Population Density

Figure 7: District mean home price by Share of working age population

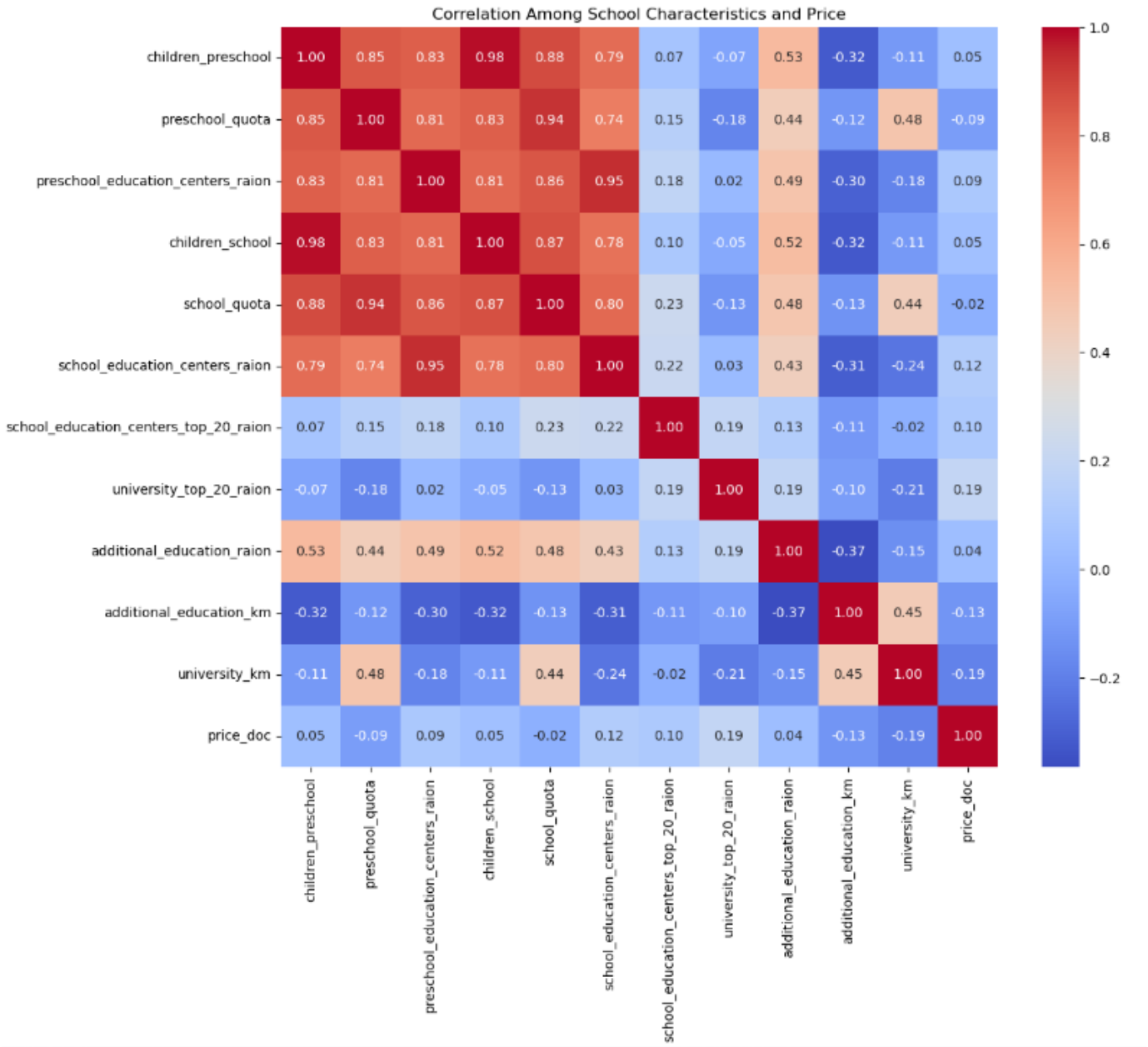


Figure 8: Correlation between School Characteristics and Price

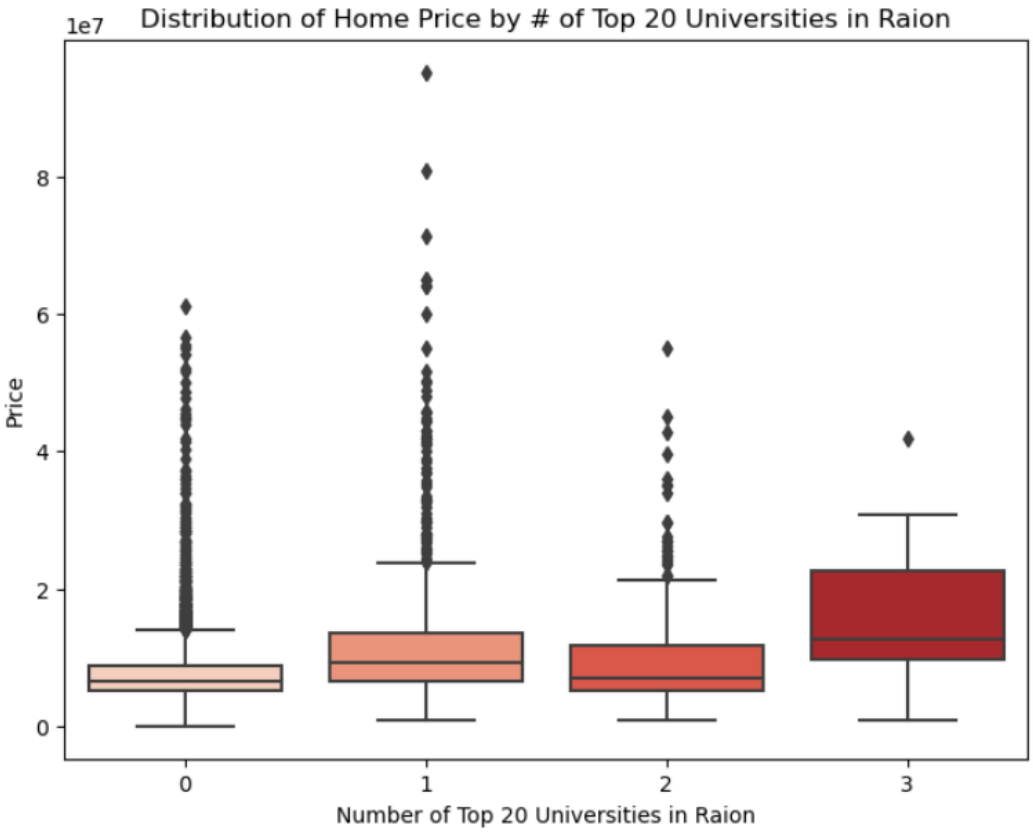


Figure 9: Distribution of Home Price by # of Top 20 University in Raion

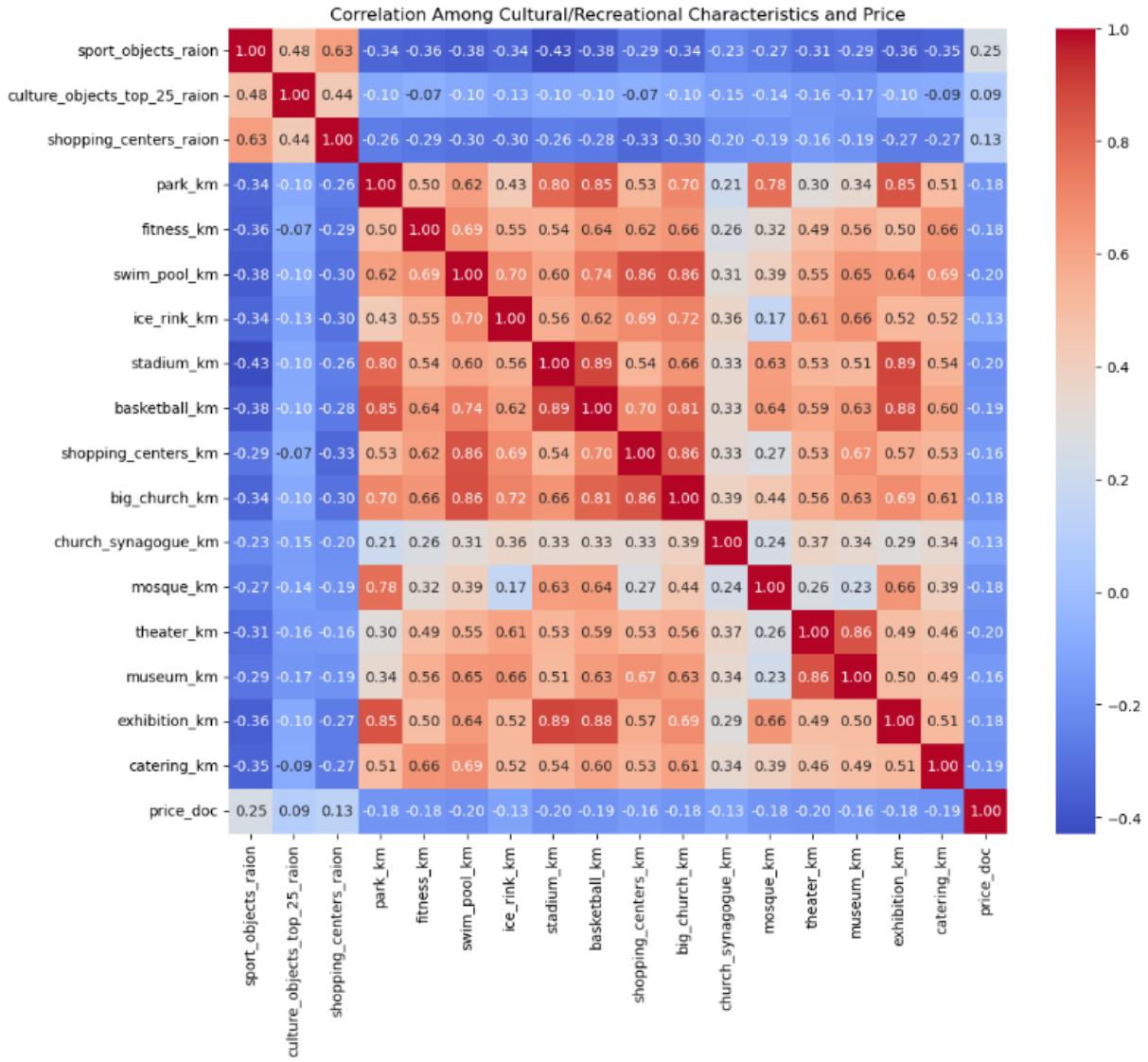


Figure 10: Correlation Among Cultural/Recreational Characteristics and Price

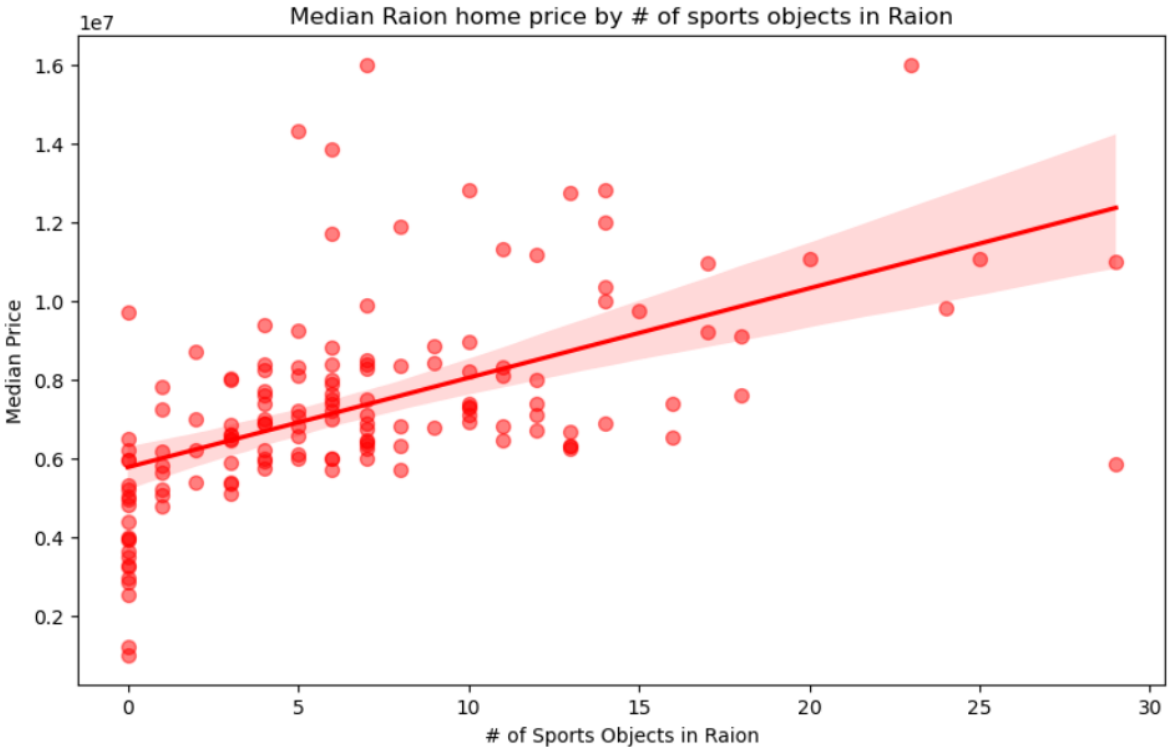


Figure 11: Median Raion home price by # of sports objects in Raion

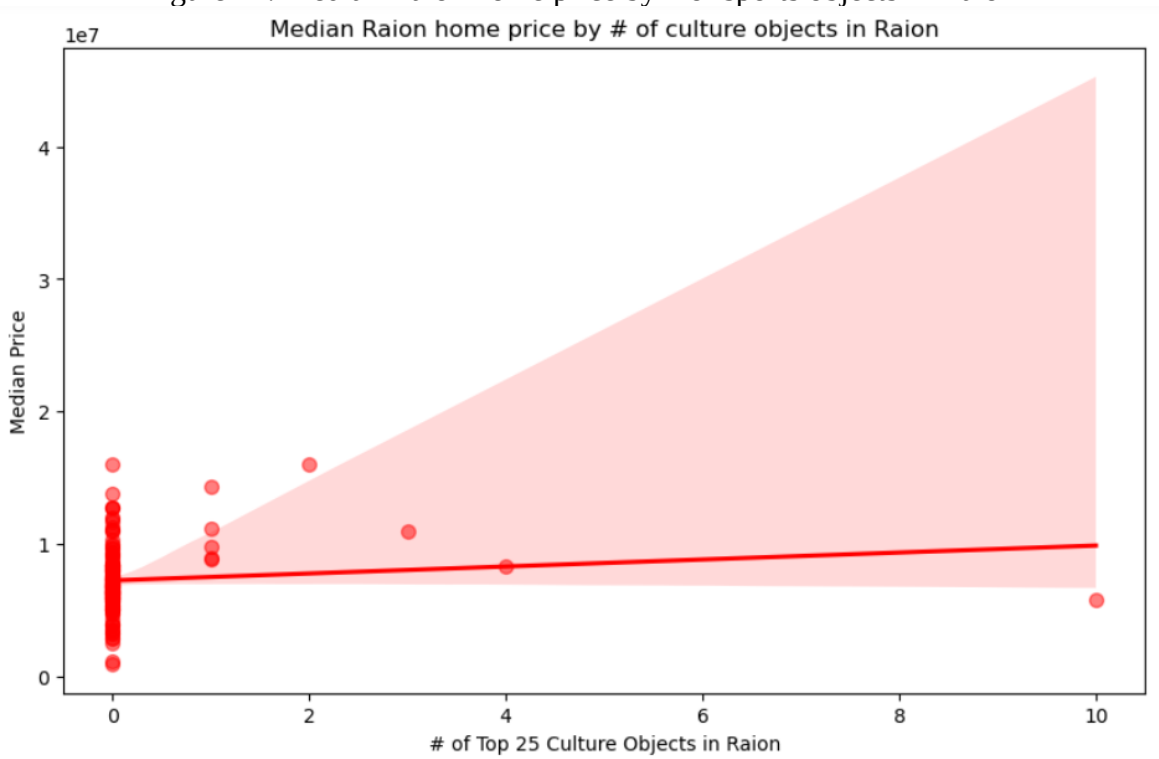


Figure 12: Median Raion home prices by # of culture objects in Raion

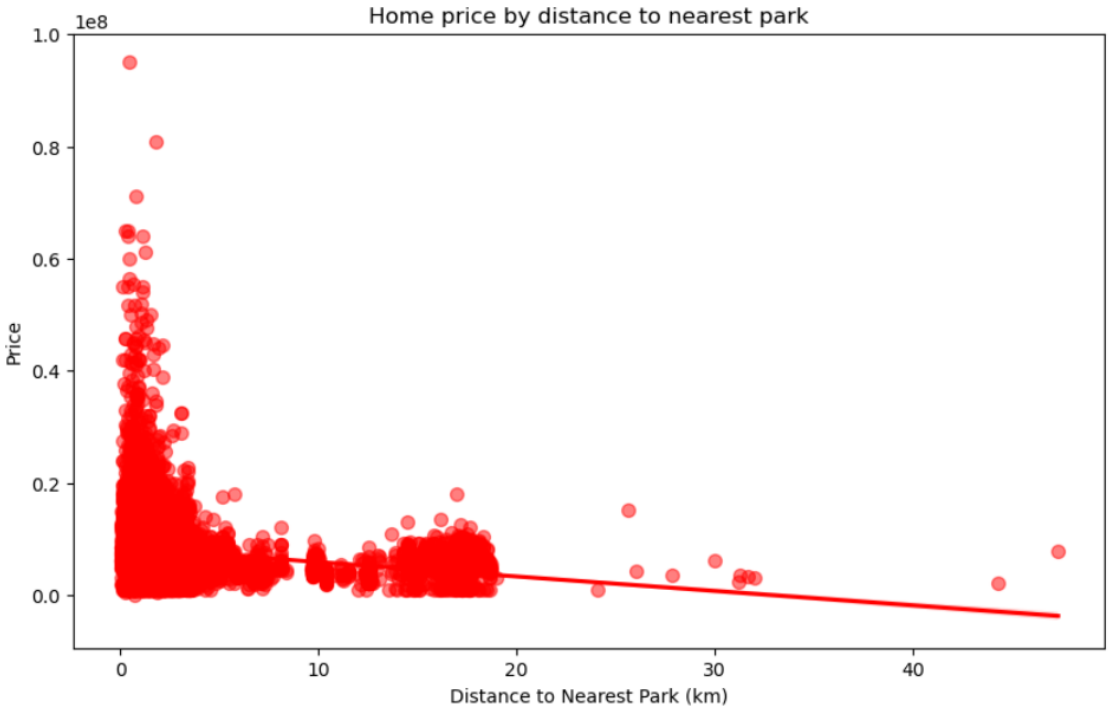


Figure 13: Price by Distance to nearest park

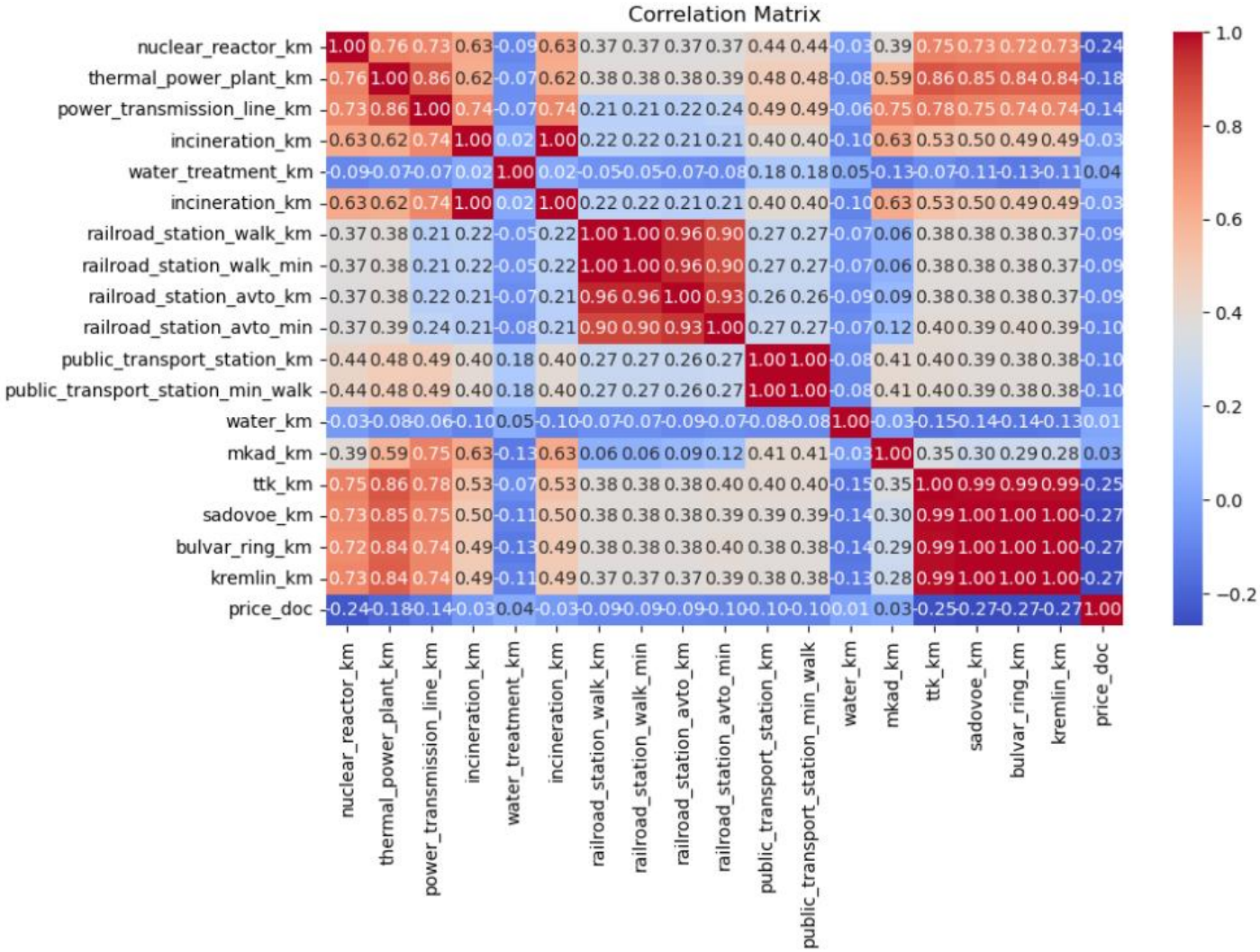


Figure 14: Correlation between infrastructure features with respect to price

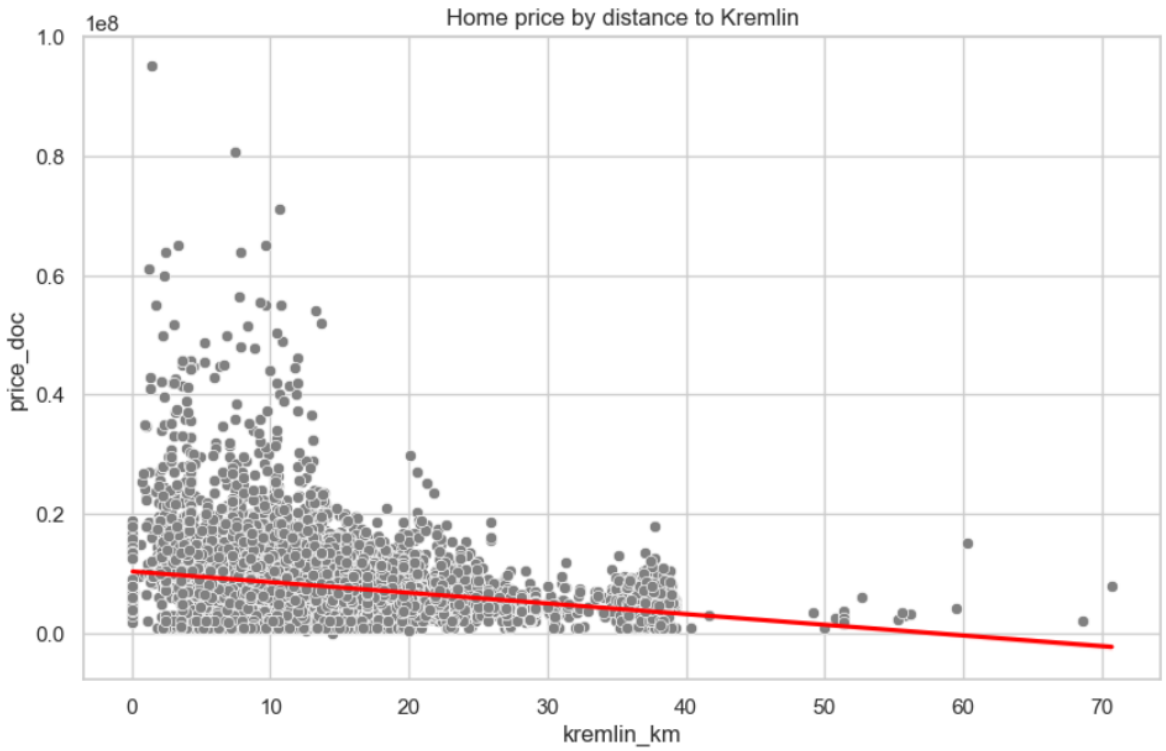


Figure 15: Price by distance to Kremlin