***Sberbank Russian Housing Market***

*Baran Berkay Barakçin, Murat Baki Bozat, Ege Baştürk,*

*Ahmet Burak Çakmak, Emre Tunçay, Mustafa Yıldırım*

***Abstract***

*Machine Learning is very popular field of research and it is used for many subjects as well as real-estate market. Overall purpose of this study is making a meaningful machine learning model for predicting Russian Housing Market prices. The general problem is specifying the prices of houses in a fragile country like Russia is really overwhelming because there are a lot of variations in the variables. This was a Kaggle competition and we provided the data from there. There were 292 columns which shows the features of the houses and approximately 30.500 raws which represent the houses. We used decision tree algorithm for predicting the prices of the houses. After a proper cleaning of the data, we used XGBoost for feature selection to avoid overfitting. Then we selected appropriate features to build decision tree regressor. To conclude, our model had 0.39842 Root Mean Logarithmic Standard Error with 9 variables.*

**Introduction**

The housing market means that not only real estate because it brings together different stakeholders, which are the renters, the homeowners who are selling their properties, the real estate investors who buy and sell properties only for investment purpose, contractors and the real estate brokers who act as facilitators in the process of buying or selling a property. Thus, housing cost is affected too much except it affects too much consumer and developers.

Housing costs demand a significant investment from all stakeholders. But there is a one annoying thing that when people try to plan a budget, uncertainty about one of their biggest expenses, which is last thing is needed by whether personal or corporate.

Sberbank which is oldest and largest bank in Russia want to helps their customers in order to make predictions about uncertainty expenses. Therefore; renters, homeowner and lenders are more confident before the sign.

Volatile economy makes anticipating cost a unique challenge although the housing market is relatively stable in Russia. Housing features such as number of bedrooms and location interact complicatedly, which makes prediction complex. If Sberbank or their costumers face with unstable economy, means that they have to have more than simple regression models.

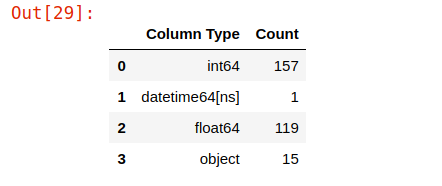
For that reason, in this competition, Sberbank tries to Kagglers to develop algorithms which predict realty prices by using a broad spectrum of features. And that algorithms provide more certainty to Sberbank customers in a volatile economy. Competitors will rely on a big dataset that includes housing data.

**Methods Review and Background**

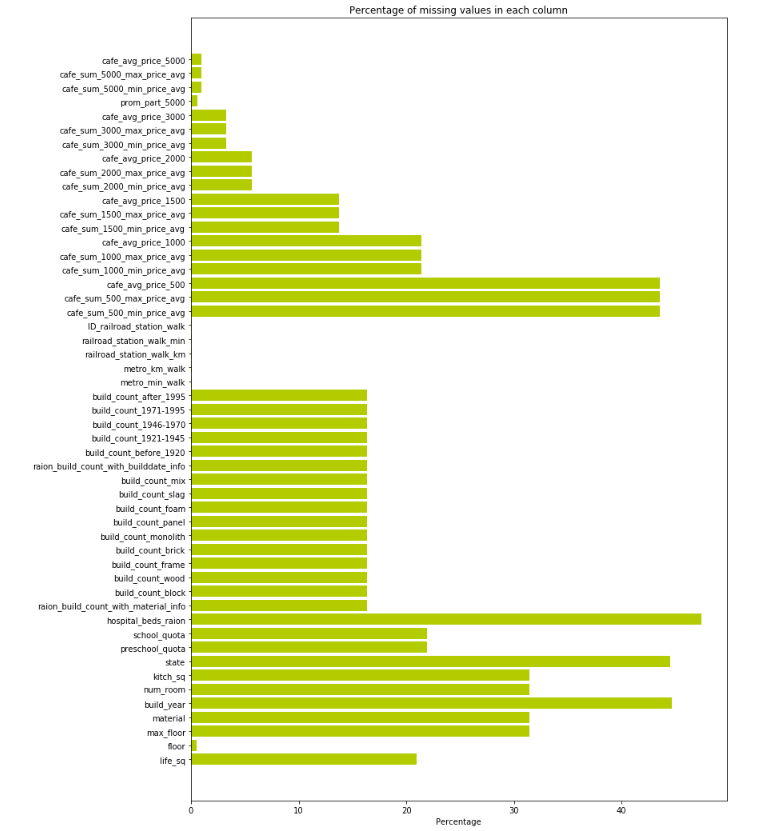
There are many Machine Learning algorithms and each of them have some advantages and disadvantages. Some of them are more accurate than others but not very good at how they reach the result and how independent variables affect the dependent variables like Neural Network algorithm. Some of them are good at explaining the model we are trying to build but less accurate then others like OLS Regression. Decision Tree Regressor is very popular and beneficial algorithm and it is really good at both explaining the how it reaches the result, in other words building the model, and it is really good at capturing a ratio of accuracy. This is why we choose to use it. Basically, decision tree regressor is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. One of the serious disadvantages of the decision tree regressor is it is really tending overfit the model. In other words, it is working really good with the training data set but when the subject is testing, it fails. The cause of overfitting is using lots of independent variables and model is getting more complex. One way to avoid overfitting is selecting the most important variables and using only them both in training and testing the model. Although, it reduces our accuracy, it helps us to build less complex and more explanatory model. First, we used XGBoost for selecting our most important variables and then we used decision tree regressor for our model. The result of the XGBoost showed that after nine independent variables, the effect rate of the variables is getting very low and we decided to train and test our model with nine variables.

Russian Federation is a transcontinental country in Eastern Europe and North Asia which has a really fragile economy like Turkey, Brazil and many other countries. In the countries which has fragile economies, it is hard to predict the values of immovable properties like houses, stores and lands because they have changing prices. The population of Russian Federation is 146,5 million and GDP per capita is 10.630$. The housing price change had an upward trend in the whole world and as well as Russia before 2008. After 2008, Russian housing prices balloon was burst and the housing market was collapsed. After 2010 until now, the housing market was in a recovery process and the prices are slightly increasing.

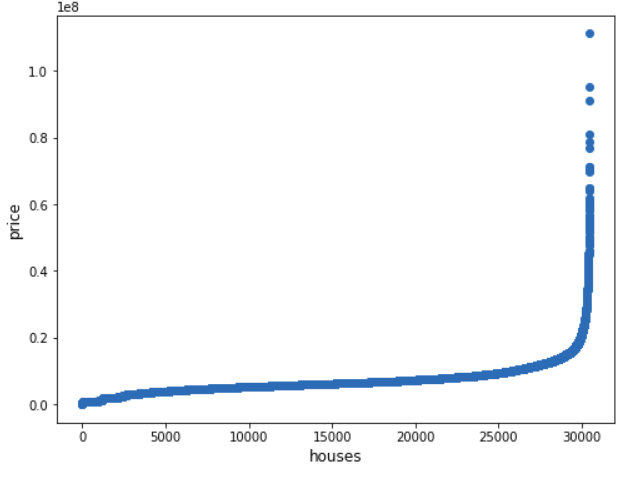
**Statistical Analysis and Data Exploration**



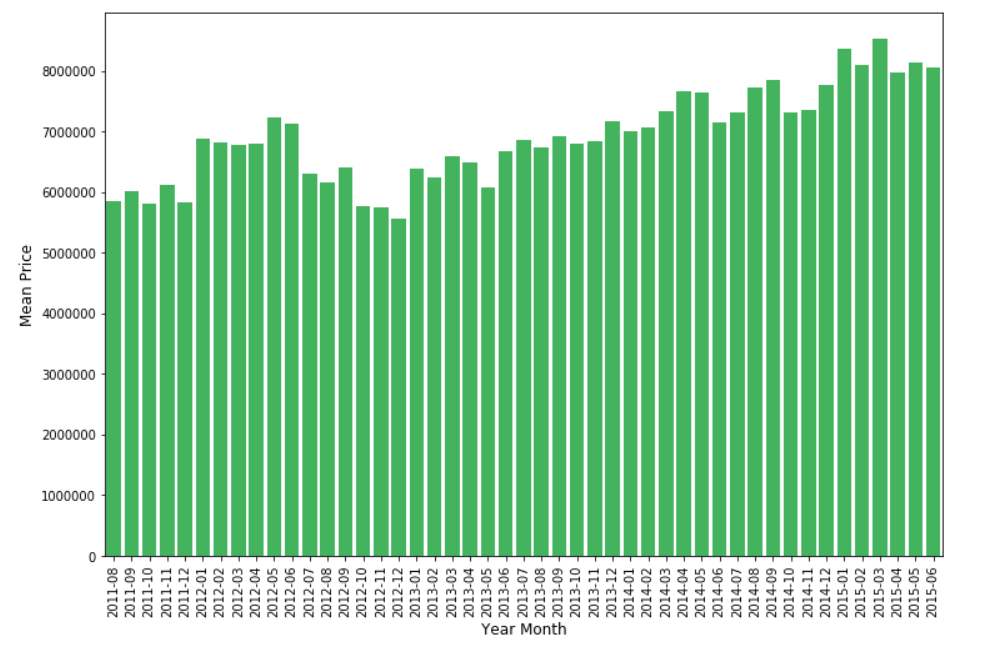
We started our data exploration and statistical analysis by analyzing the types of the variables. As you can see in the above table our data consist of 157 integer and 119 float variables so there are 276 numerical variables. The 15 variables are objects so these are the string variables in the data. The other variable shows the date. It is good for us that the big part of the data is numerical since it is harder to analyze strings.



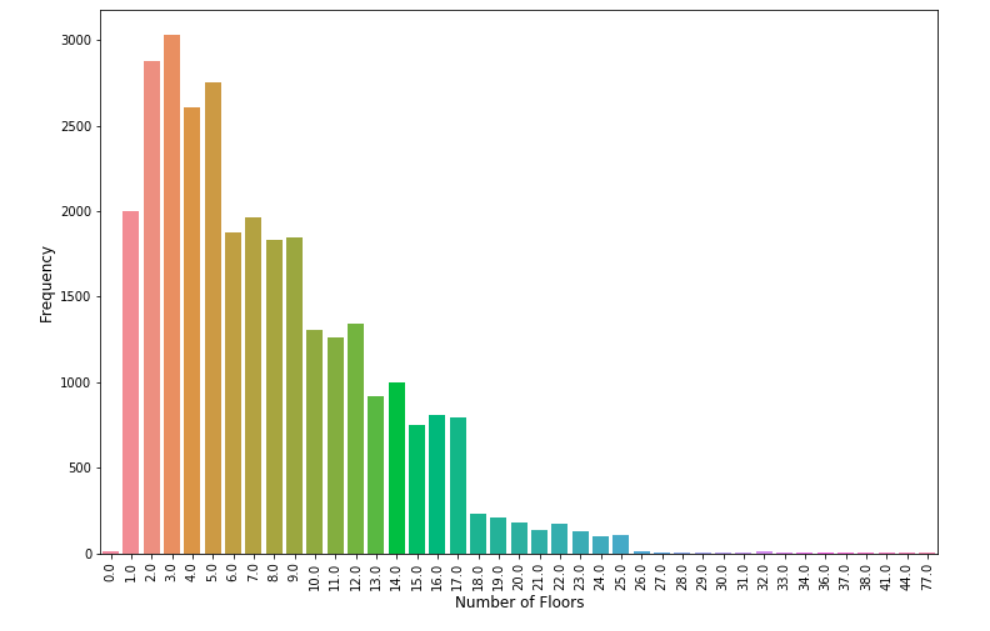
Second, we analyzed the missing values in each column. As you can see in the above graph, some of the variables have a lot of missing values. We analyzed these variables and see that most of the variables are not so significant determinants of housing prices.

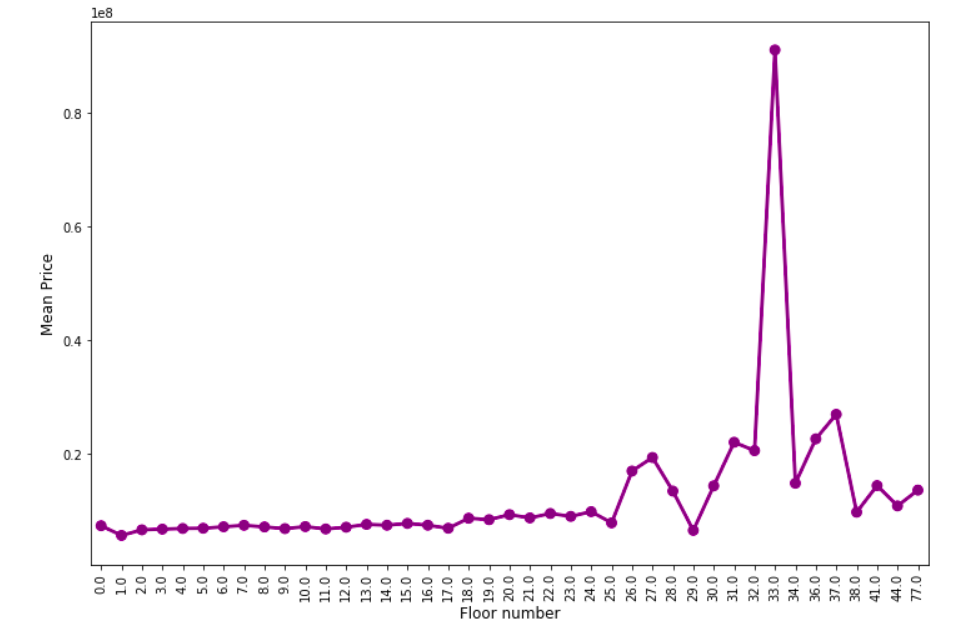


To detect outliers, we made a graph of prices and indexes of the houses. There are some outliers but they are not so much. We decided to continue with the outliers since they can’t affect our model much.



Then, we created a feature that shows monthly mean prices of houses. There is an increase in house prices by time. In the background of this study, we specified after 2008 crisis, the housing market of Russia is in a recovery process and its prices are increasing slightly. We can see this in the above table, there is slightly upward trend in the housing prices of Russia.





The pilot and the graph above show the counts of the floor numbers and the mean prices of the houses respectively. Most of the buildings are in the second third floor. The floor number is a determinant variable of our independent variable so analyzing this pilot is important. As expected, when the number of floors increase, the frequency of houses decreases. The price-floor graph shows an overall increasing trend. A sudden increase in the house price is also observed at floor 18. Also, the houses with 0 floor are slightly more expensive.

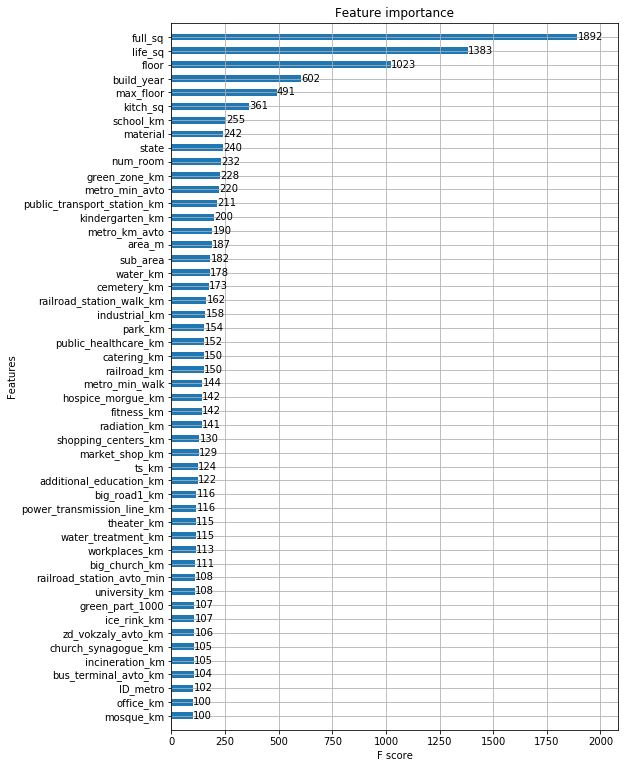
**Data Cleaning and Restructuring**

Cleaning and restructuring data are commonly done through ETL (extract, transform, and load) processes. There may also be data scrubbing processes involved. Sometimes this is performed through packaged tools and sometimes through in-house-developed scripts or programs. When packaged tools are used, the errors are not coming from them but rather from faulty specifications used for them. However, we take our data from Sberbank. There are many N/As. Thus, we have to cleaned and re-structed them.

We splitted the data to 30 percent to 70 percent as test and train data respectively. Then for the cleaning, first, we detected the outliers and dropped them. We used the price value to detect outliers and removed the 5 percent of the data from the both tails. Then we needed to deal with the NAs in the data. Our most determinant variable is full\_sq which shows the full size of the house. The other NAs replaced by the mean of each column. There were approximately 6.000 houses which doesn’t have the data of full\_sq so we dropped them. Lastly, there were some rows that have negative full\_sq value. Since a house can’t have a negative size, we dropped these values, too.

**Methods and Tools of Use**

To build our model, we used XGBoost, GridSearchCV and Decision Tree Regressor. In the XGBoost Feature Selection part, we tried some parameters and chose the best one. Learning Rate is a hyperparameter which determines to what extent newly acquired information overrides old information. Max Depth is maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. First, for XGBoost, we fixed the learning rate to 0.01 and tried 1, 3 and 5 for max depth. We analyzed the results and decided to use 5 for max depth. Then, we fixed the max depth to 5 and changed the learning rate. 0.01, 0.05 and 0.1 were the parameters we tried and we decided that 0.1 is the most optimal learning rate for our model. You can find all trials in the Appendix part. Below, there is a graph of feature importance model we used for building the algorithm. As you see, after nine variables, the importance of the features is getting very low to include the model so we chose seven variables to put in model. Logically, full square area of the house is the most important feature to determine the price of the house. People give importance to the building year of the house they live in or buy for an investment. Apartment floor is important for young and old members of household. School\_km is important feature for families with kids since it means the distance between the house and the nearest school. The material of the house is significant because some materials can keep the house in warm in cold days or some of them keep cool when it is hot while some of the materials can’t do it.



For the regression, we used decision tree regressor and for the hyper-parameter selection we used GridSearchCV. We used that because we want to know which variables affect the data, in other words we wanted an explanation, and it is a strong algorithm in the sense of accuracy. We did a cross-validation for max\_depth and found that most optimal value is 11. For the leaves, we didn’t want very little number of samples in each of them so we put min\_samples\_split value to 30. As we said, we divided the data 30 percent as test data and 70 percent as train data.

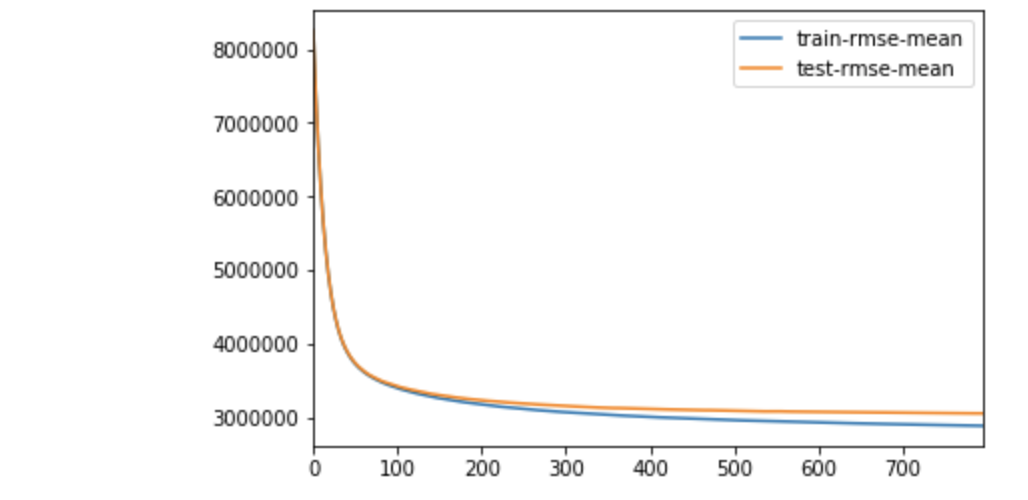
**Results**

TheR-Squared on training dataset was 0.6507. The accuracy ratio for the competition named Sberbank Russian Housing Market was RMLSE which means Root Mean Logarithmic Standard Error. Our test made 0.39842 Root Mean Logarithmic Standard Error which is better than Training Mean Benchmark (0.53347) in the Kaggle by 0.13505. Our ranking would be 2868th if we would submit this model on time. The winner of the competition had 0.30087 which is pretty lower than our standard errors. We think that we did a pretty good job since this was our first try but we know that we have a lot to go.

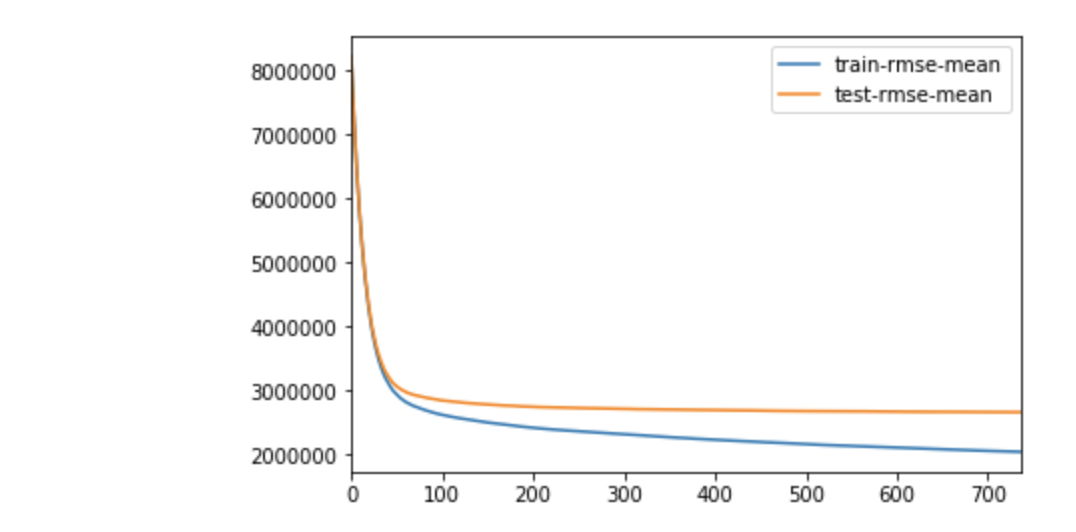
**References**

1. Sberbank Russian Housing Market | Kaggle. (2019). Retrieved from <https://www.kaggle.com/c/sberbank-russian-housing-market/kernels>
2. Scikit-learn: machine learning in Python — Scikit-learn 0.21.2 documentation. (2019). Retrieved from https://scikit-learn.org/stable/
3. Quinlan, J. R. (1987). "Simplifying decision trees". *International Journal of Man-Machine Studies*. **27** (3): 221–234. [CiteSeerX](https://en.wikipedia.org/wiki/CiteSeerX) [10.1.1.18.4267](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.4267). [doi](https://en.wikipedia.org/wiki/Digital_object_identifier):[10.1016/S0020-7373(87)80053-6](https://doi.org/10.1016%2FS0020-7373(87)80053-6).
4. Python Data Analysis Library — pandas: Python Data Analysis Library. (2019). Retrieved from <https://pandas.pydata.org/>
5. Hafidz Zulkifli (21 January 2018). "Understanding Learning Rates and How It Improves Performance in Deep Learning". Towards Data Science. Retrieved 15 February 2019. Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient.

**Appendix**

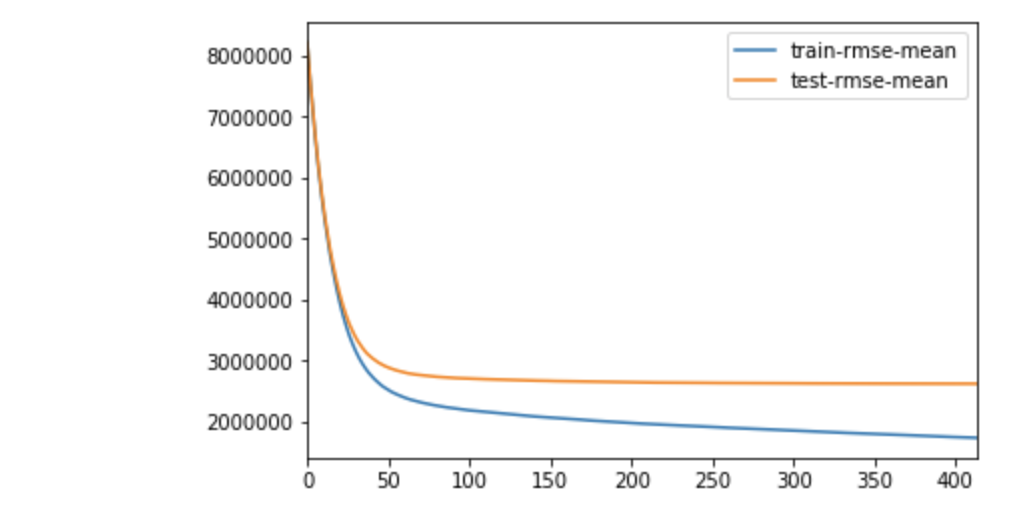
Learning rate: 0.05

Max depth: 1

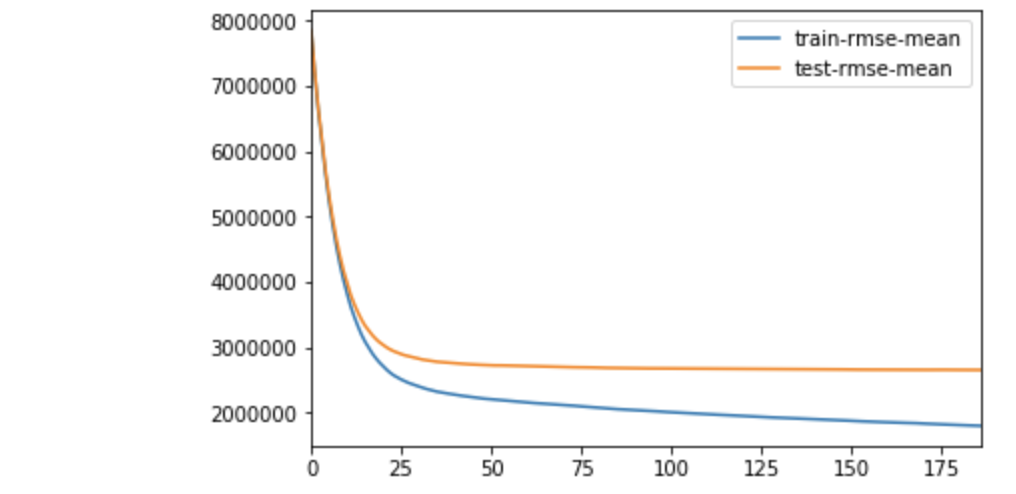


Learning Rate: 0.05

Max Depth: 3

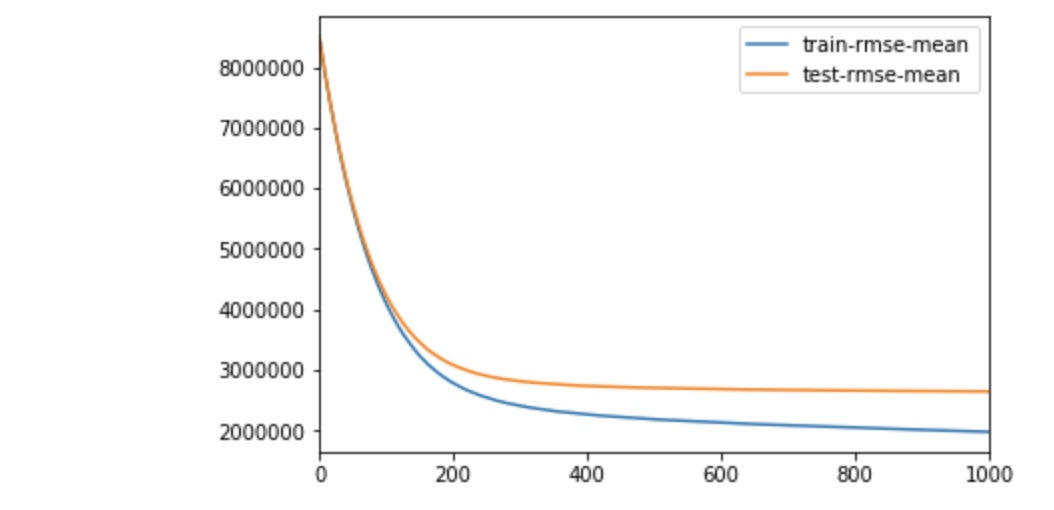


Learning Rate: 0.05

Max Depth: 5

Learning Rate: 0.1

Max Depth: 5

Learning Rate:0.01

Max Depth: 5