House Price Estimation In Piraeus Using Machine Learning

Dimitrios Patiniotis Spyropoulos

February 2022

1. Introduction

A house/shelter is widely accepted to be an essential human need. As living standards rise, the demand for houses has followed suit. This demand is either due to the fact that houses are a great investment, as they have a proven intrinsic value, or simply because they provide shelter. Home ownership is a great indicator of a countries economy and has also been shown to be closely related to monetary stability [1].

The housing market, like the economy, is ever changing with new trends showing up regularly. Usually, to get an estimation of one's property value one would have to ask for the advice of a domain expert (most likely a real estate agent). The expert would have to take into account many factors, such as location, property area, current market state etc.

In the context of Machine Learning, this price estimation can be approached in a similar way, with small modifications. Firstly, the data has to be fetched instead of being recalled by one's memory and, secondly, the estimator has to be a program instead of a human. In the following report I will present the process of creating a web crawler to fetch usable data on a daily basis; how to automatically process that data so that it is as clean and usable as possible for the machine learning algorithms; and how different algorithms cope with the task at hand. The goal of this project was to explore different areas of the vast field that is machine learning whilst also creating a useful tool that produces the latest house prices estimations for the city of Pireaus.

2. Methodology

2.1 Fetching Data

Getting the latest house data, although not an impossible task, it has its own challenges. Firstly, there are no free or open source datasets that are up to date for the location that we will be focusing on. Furthermore, companies that do have access to said data most likely are not willing to give away one of their competitive advantages for free, so they block all bots via captcha. After a few failed attempts with other real estate websites, I made a bot that fetches data from xe.gr, a major player in the industry.

2.2 Evaluating and Modifying the Data

As mentioned above, fetching real time data when dealing with house prices in Piraeus is hard enough — let alone getting high quality data. In the context of a streamlined process, as the one we will be creating in this project, our automated data engineering can go so far. Our models will inevitably have to deal with some amount of noise. Before we discuss where most of that noise is located, let us first have a look at our features and our target variable. The dataset that will feed our models will have 7 features:

1) Type. For example, house can be a one-storey family house or an apartment building.
2) Location. The sample that is fetched has usually at least 15 sub-regions, allowing for the model to identify the ones that are in high demand.
3) Surface Area. The area is measured in square meters.
4) Level. This corresponds to the level of the apartment.
5) Number of bedrooms in the house.
6) Number of bathrooms in the house.
7) Year of construction. Note that this will be transformed to the age of the building measured (current year minus year of construction).

The target variable is the asking price for a given house in Euros.

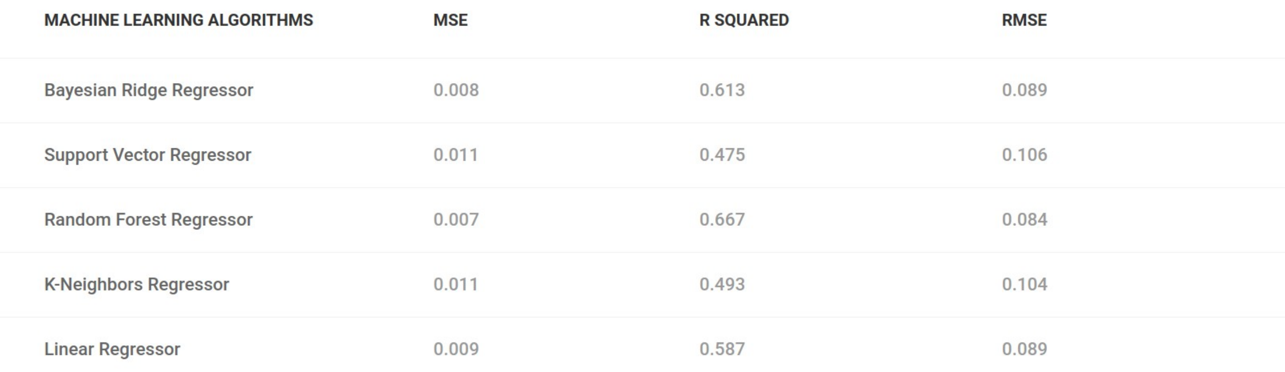
After going through some of the postings we can discover three main sources of noise. The first is missing data. Not all postings have the above characteristics. A second source of noise is that many people renovate their apartment and claim that the house was built the year it was renovated. This adds major noise to the year feature, although when prediction is based on current market sentiment it can also be ignored, as it is a current trend. Thirdly, although more rare than the above, there are postings that have erroneous values, which proved to be, out the three noise sources, the hardest one to combat.

To deal with missing data, the web scrapper acts preemptively and only adds rows that have all of their columns filled with valid values. This in a sense also helps with the third problem. We will not take any measures to combat the second noise type. This is done to better reflect current market trend. For example, if someone wants to sell his / her newly renovated house it would be foolish of him to price it at the same price range as if there was no renovation at all. Lastly, the third type of noise is really hard to be dealt, especially when we have to schedule a periodic fetching and processing of new up-to-date data. Having said that, we will be removing outlier cases by setting a lower and upper limit on the price of a house.

2.3 Categorical Encoding

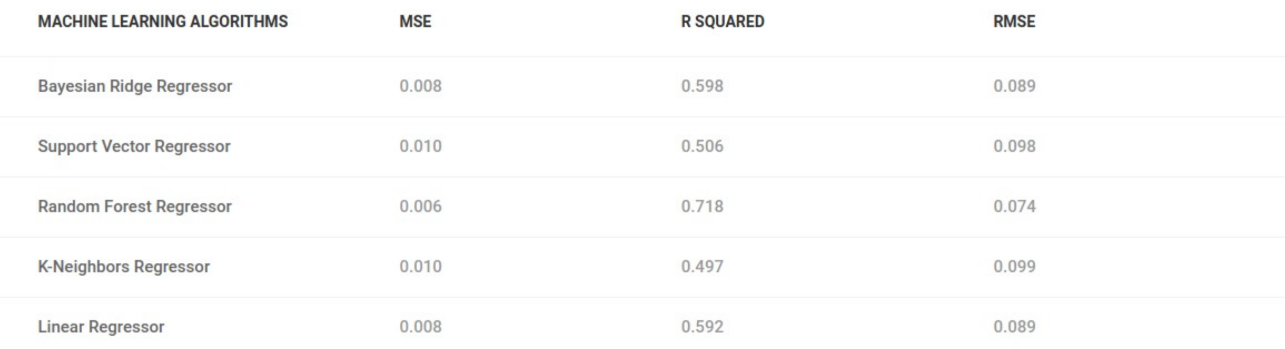
From the feature set we can observe that Type and Location have categorical values. These values have to be represented as numbers. This can be achieved by using categorical encoding. In this project I tested One-Hot encoding and Label-Encoding on normalized data and One-Hot proved to be better, ceteris paribus (One hot displays both higher Coefficient of Determination and lower Mean Squared Error). When we normalize the Lebel-Encoded columns, the gap becomes non existent with the Random Forest Regressor perfoming noticeably better with this configuration.

Table 1 – One Hot Encoding with Min-Max Normalization - MSE, R Squared and RMSE



MACHINE LEARNING ALGORITHMS	MSE	R SQUARED	RMSE
Bayesian Ridge Regressor	0.008	0.613	0.089
Support Vector Regressor	0.011	0.475	0.106
Random Forest Regressor	0.007	0.667	0.084
K-Neighbors Regressor	0.011	0.493	0.104
Linear Regressor	0.009	0.587	0.089

Table 2 – Label Encoding with Min-Max Normalization - MSE, R Squared and RMSE



MACHINE LEARNING ALGORITHMS	MSE	R SQUARED	RMSE
Bayesian Ridge Regressor	0.008	0.598	0.089
Support Vector Regressor	0.010	0.506	0.098
Random Forest Regressor	0.006	0.718	0.074
K-Neighbors Regressor	0.010	0.497	0.099
Linear Regressor	0.008	0.592	0.089

2.4 Min-Max Normalization vs Standard Scaling vs Log Transformation

For the experiments I mainly used the Min-Max normalization, but there is definitely a case to be made for the use of Standard Scaling. Min-Max produces a lower MSE and Roote MSE, something that is to be expected since Min-Max produces values in the 0-1 range, and thus yields smaller error. We cannot draw a conclusion by those two metrics, and the R-squared seems to be somewhat indifferent. Since the data seems to be skewed (see Figure 1), I also tried log transformation, which proved to be by far the worst of the three methods tested. Table 3 and 4 show the average (over 100 iterations) MSE, R-squared and RMSE of each of the algorithms tested when we apply min-max normalization and standard scaling, respectively. Based on Tables 1 and 3 I chose the Min-Max normalization but, Standard Scaling also seems like a viable option, as mentioned above.

Figure 1 – House Price Distribution of 1000 samples (26-1-2022)

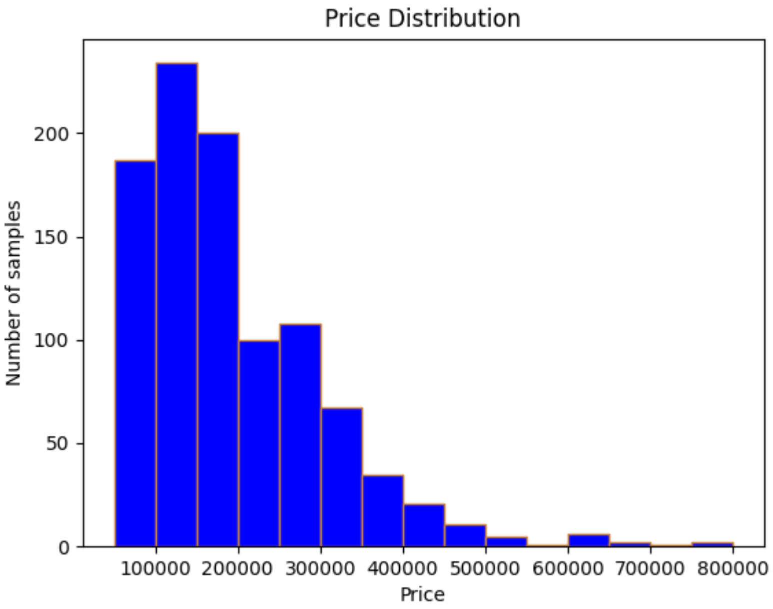
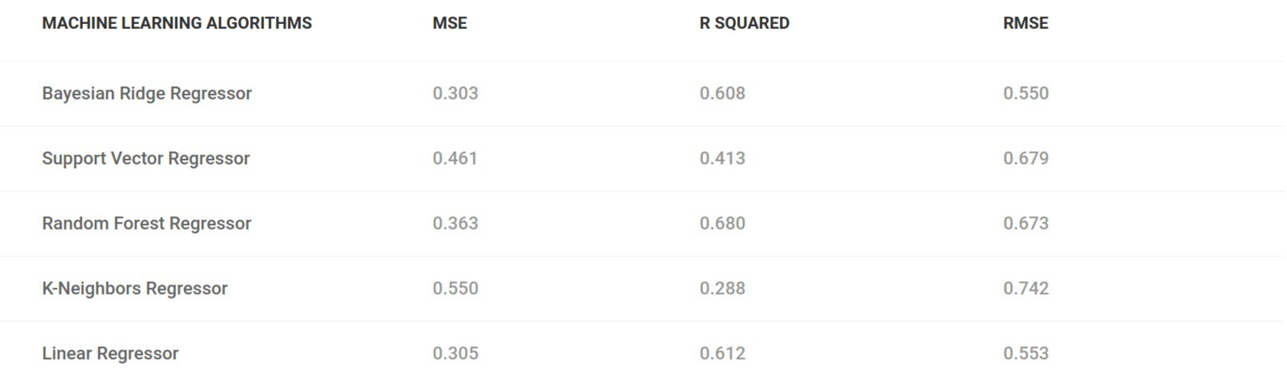


Table 3 – Standard Scaling - MSE, R Squared and RMSE



MACHINE LEARNING ALGORITHMS	MSE	R SQUARED	RMSE
Bayesian Ridge Regressor	0.303	0.608	0.550
Support Vector Regressor	0.461	0.413	0.679
Random Forest Regressor	0.363	0.680	0.673
K-Neighbors Regressor	0.550	0.288	0.742
Linear Regressor	0.305	0.612	0.553

3. Algorithms and Results

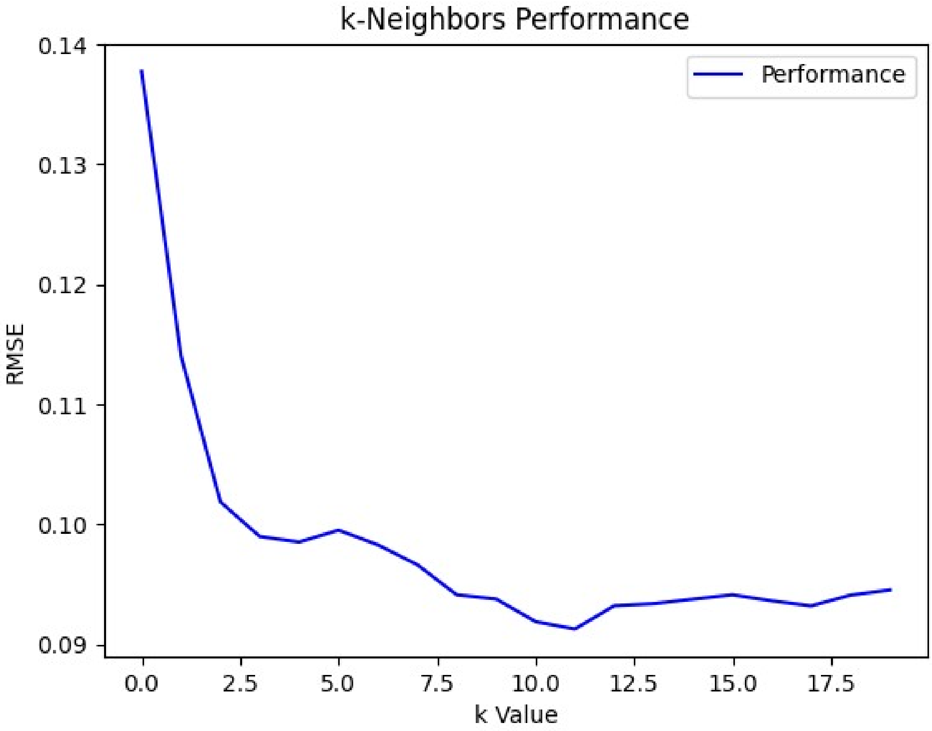
Five different ML algorithms have been tested on this dataset. We will discuss each one of them and how their modifications affect their performance leaving the best for last. Whilst label encoding and min-max scaling is not clearly the best combination, it provides the best performance with Random Forest Regression, which ultimately models the data best. Thus, for all the following experiments label encoding and min-max scaling have been applied.

3.1 K-Neighbors Regression

The k-nearest neighbors algorithm (*k*-NN) was first proposed as a non-parametric classification method developed in the early 50s [2]. The classification is the result of a plurality vote of its neighbors. For example, if *k* is set to 1, the object is classified the class of its closest neighbor. To turn this classifier to a regressor, the output value is the average of its *k* nearest values.

Out of the 5 algorithms tested, *k*-NR seems to has very mediocre performance, no matter the data prepossessing configuration, or the *k* value. Figure 2 shows the RMSE achieved when changing *k* when feeding the *k*-Neighbor Regressor with our dataset.

Figure 2 – *k*-Neighbors Performance when changing *k*



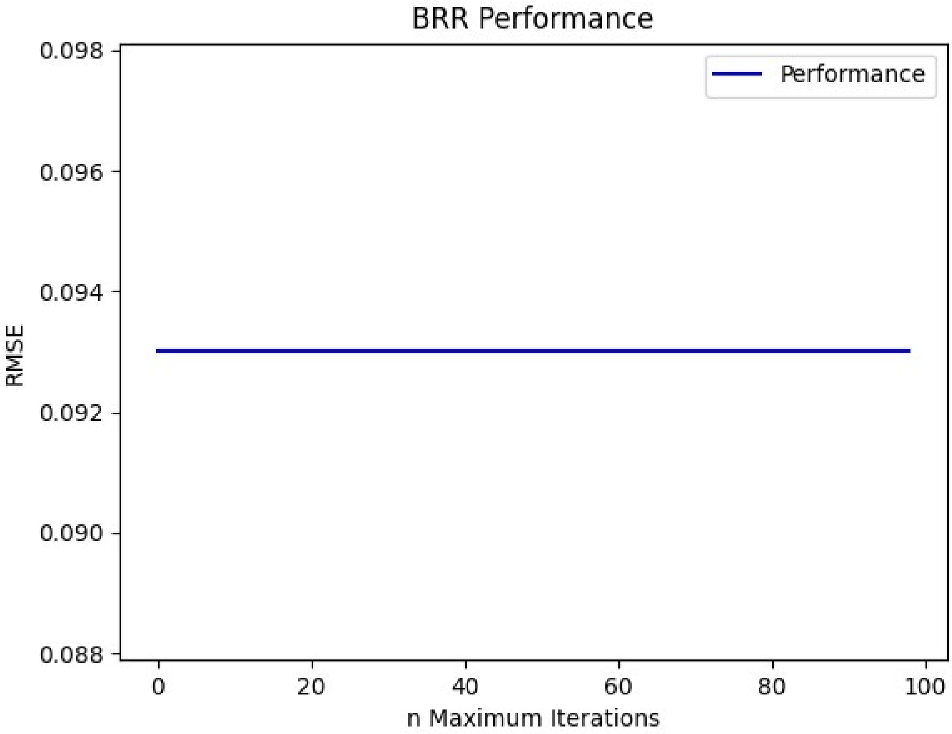
3.2 Multiple Linear Regression

Linear Regression is one of the simplest linear approaches for modeling the relationship between a independent variable and a dependent one. When our feature set has more than one independent variables, then this process is called Multiple Linear Regression [3]. As we can see from Tables 1, 2, and 3, MLR had a solid performance across the board, but could not match the performance of a Random Forrest Regressor in any metric.

3.3 Bayesian Ridge Regression

Bayesian Ridge Regression is a Bayesian Regression technique. The main idea is that we formulate a linear regression, but this time using probability distributions and not point estimates. The aim is not to find a "best" value for our models parameters, but to determine their distribution[4]. When tested with our dataset, it also had solid performance as shown in Tables 1, 2, and 3. When tweaking the maximum Regressor iterations, performance does not change (Figure 3).

Figure 3 – Tweaking the maximum iterations parameter of BRR

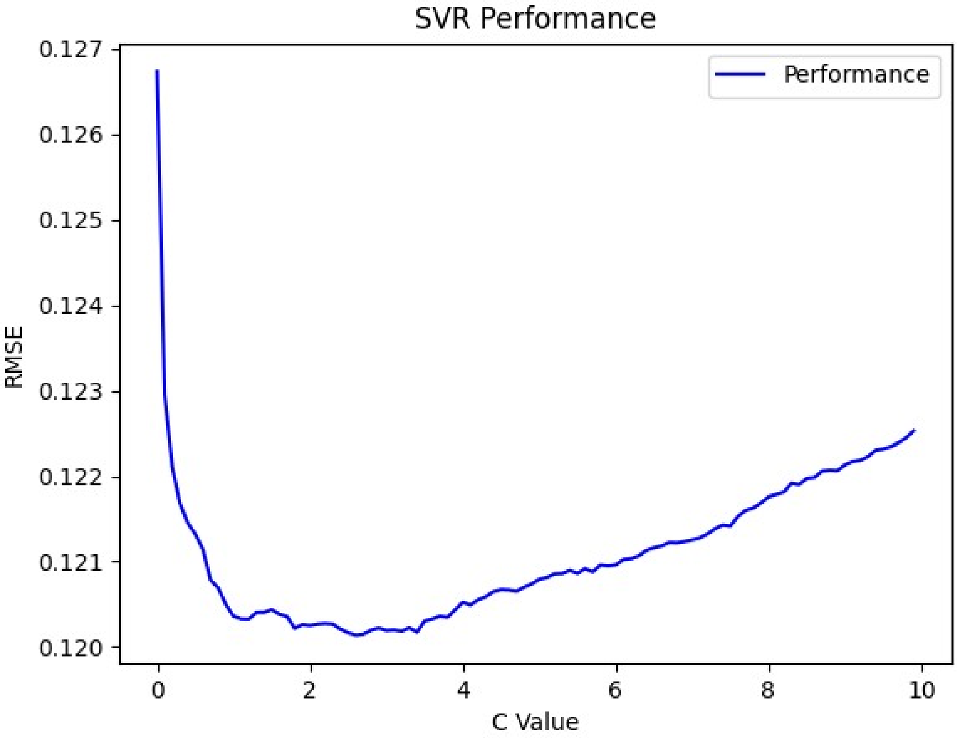


3.4 Support Vector Regression

Support Vector Regression (SVR) is a modification of the well-known Support Vector Machines [5,6] used for classification. In short, SVM constructs a hyperplane, the dimensionality of which is usually higher than the number of the initial dataset attributes, and uses it to classify test cases into different classes. That hyperplane is created such that it has the maximum distance from the nearest training cases, which are also called support vectors. For each side of the hyperplane, and based on the support vectors, a sub-hyperplane is constructed. By having maximum margin between these sub-hyperplanes we reduce the number of misclassifications. In the case of SVRs, the depentent variable is calculated by calculating the disctance from different hyperplanes. When compared with the regression methods mentioned above, SVR is more flexible in the sense that we can define how much error is acceptable, and based on that create the appropriate hyperplane to fit our data.

It becomes apparent that SVM (and SVRs for that matter), when given noisy attribute data, can create inaccurate hyperplanes and sub-hyperplanes because they are based on erroneous support vectors. This is a probable reason for its mediocre performance, as shown in Tables 1, 2, and 3. Figure 4 displays the relationship of the regularization parameter C and RMSE of the SVR model.

Figure 4 – Tweaking Regularization Parameter of SVR



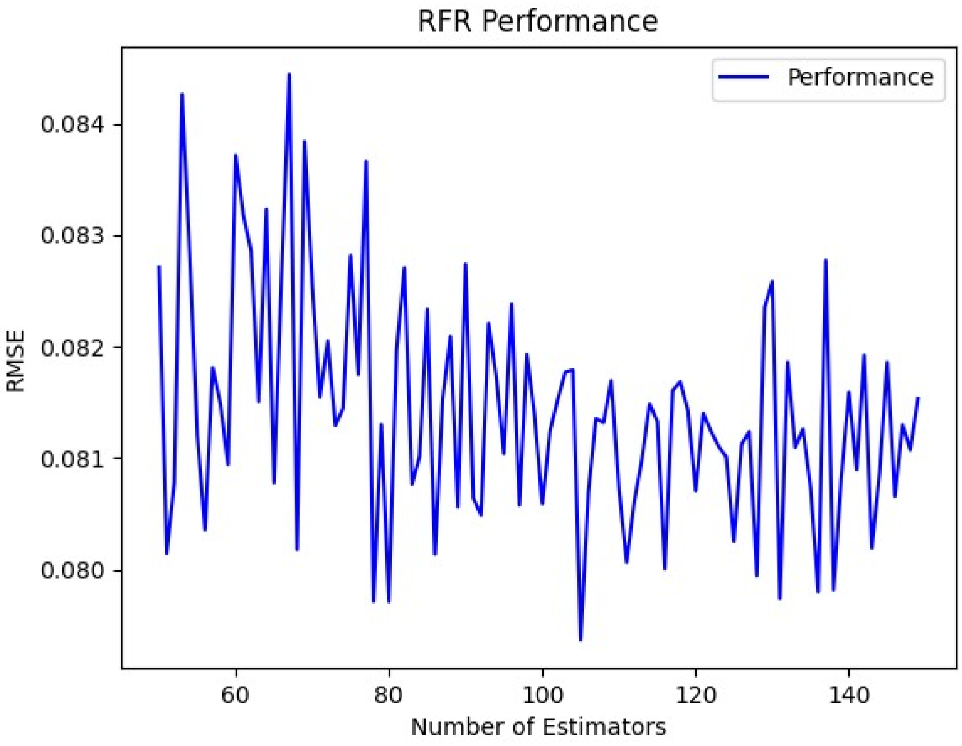
3.5 Random Forest Regression

Random Decision Forests work by constructing many decision trees and, in the case of Regression, they output the average, or mean prediction of these trees[7]. As shown in Tables 1, 2, and 3, the Random Forest Regressor (RFR) achieves the best performance in all metrics compared to the other 4 algorithms. Figure 5 shows how RFR performs with different number of estimators. Although performance varies, we can see that, in general, our model achieves slightly better results when using more estimators.

4. Deployment

Alongside the code for the experiments and this report I have put together an API which processes input data and estimates the value of a property located in Pireaus. It fetches new data from www.xe.gr and processes the data using Label Encoding and Min-Max normalization. I have chosen the RFR as a Regressor, since it displayed the best performance in testing.

Figure 5 – Number of Estimators of an RFR and RMSE



5. Further Thoughts & Conclusion

After dealing with the complexities of noisy, real world data and the nuances of the main data engineering and machine learning methods, the end result is a relatively realistic estimator. When we try to evaluate the end result we come to the realization that such a task is daunting. Estimating the value of a property is complex, and besides a basic categorization of properties (e.g. cheap housing, luxury apartments etc) estimating the exact value is hard, since no one really knows what the correct number really is.

Having said that, we should acknowledge a few blind spots of the models produced by this dataset. Firstly, although the concept is that this model is reconstructed every day with new data, it still does not fully take into account market trends. It can provide the latest estimation but if, let us say, there is an upward trend, pricing your house at the current evaluation (and not a bit higher) may not be optimal. Secondly, we estimate property value by asking price and not selling price. Although these tend to be closely related, if for example houses are sold at lower prices than the asking price, this could indicate a recession-like trend. This also cannot be detected by this model.

Improvements and further work could also be made. A step forward is implementing this pipeline to different cities and neighborhoods, and making it available for anyone who wants an estimation of his/her own property. As for improving accuracy, acquiring more features could also be greatly beneficial. The sole reason it was not done for this project was to avoid sending too many automated queries to the site that data was crawled from. For reference, to get a day’s worth of data, the crawler makes about 40 requests. If we tried to add 1 more feature, let us say energy class, we would have to make 2000 more requests (a 5000% increase) on that data, since 2 more requests are needed for each instance to crawl the needed information.

References

[1] A. S. Temür, M. Akgün, and G. Temür. *Predicting Housing Sales in Turkey Using Arima, Lstm and Hybrid Models*; J. Bus. Econ. Manag., vol. 20, no. 5, pp. 920–938, 2019.

[2] E. Fix, J. L. Hodges. *Nonparametric Discrimination: Consistency Properties*; USAF School of Aviation Medicine, Randolph Field, Texas, 1951.

[3] K. Unanik, G. Nese. *A Study on Multiple Linear regression Analysis*; Procedia – Social and Behavioral Sciences. vol. 106, no. 10, pp 234-240, 2013.

[4] L. Pasanen, L. Holmstrom and M. J. Sillanpaa. *Supporting Information for ’Bayesian LASSO, scale space and decision making in association genetics’*; Biocenter Oulu, Finland, 2018.

[5] Vapnik, V.N. *The Nature of Statistical Learning Theory*; Springer-Verlag: New York, NY, USA, 1995.

[6] Vapnik, V.N. *Statistical Learning Theory*; Wiley: New York, NY, USA, 1998.

[7] T.K. Ho. *The Random Subspace Method for Constructing Decision Forests*; IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 8, pp. 832-844, 1998.