Explaining Real Estate Market

with Decision Tree based Models

Jernej Brlek   
 *Business Mathematics and Economics*   
*Zagreb school of Economics and Management*Zagreb, Croatia  
jbrlek@student.zsem.hr

*Abstract*—This paper gives insight into the usefulness of decision

tree-based algorithms in the space of investing decision making in real estate. The issues covered include gathering the data, preprocessing, exploratory data analysis, different tree algorithms and their predictive power, as well as comments on the usefulness of the predictions.

Keywords—decision tree, real estate, modelling, features,

# Introduction

# Real estate is an often overlooked option for investing due high requirements for entry. However, it is an option that everyone is considering, at least at some point in life. Moving out or sticking with parents? Renting or buying? House or a flat? City or suburbs? What is the right price? Options are vast. Which one is the right one?

Everyone dislikes the feeling of buyer's remorse. That is only partially why many of those decisions are difficult to make. The other reasons include lack of awareness of one's true preferences and the sheer number of possibilities as a result of the market size.

Since the focus of this paper is on machine learning algorithms and their predictive power in a real estate setting the questions we would like to answer are as follows: Is there a place in real estate pricing for machine learning? What data is available online for the real estate listings in Slovenia through the advertising agencies websites and is it sufficient for the use of ML? How well can ML predict the price of a flat in Slovenia?

# Gathering the Data

Two main advertising platforms for real estate in Slovenia are nepremicnine.net and bolha.com. For the ease of aggregating the information from individual advertisements, the latter has been chosen, with possibility of extending the research onto the former.

Dataset was generated using web scraping algorithms with the use of Python packages Selenium and Beautiful soup. Given the size of the aforementioned site and its status as one of the two main Slovenian real estate advertising platforms, the goal of the research was to create a ‘useful’ model, which can be used to predict prices of the flat given information about its features and potentially identify investing opportunities.

# The goal

The goal of this project is to learn about ML and expand the knowledge of more advanced algorithms and processes. That already sets up the first category of goals, which are related to my personal progress in the field. That includes going more in depth on every stage of the process, mainly in data acquisition, pre-processing, exploratory data analysis (EDA), model optimisation and interpretation.

The second part of the goals relates to the content of the project itself. It consists of designing multiple models and selecting for the best predictor of the price of properties in a regression setting. Based on the price estimates, a tool will be designed that can effectively and reliably provide aid in judgement for whether to rent or buy based on some input parameters and detect whether newly published properties are under or overpriced. However, this is, unfortunately, not a part of this paper.

# What Has Been Done

## Web Scraping

In the Github repository [1] from Luka Tašler, an implementation of such a method for the purpose of tracking the activity is implemented. It provides the user with information about newly published listings with an automated scanning feature. I would like to take it a step further and provide insight on whether the new listing is actually a ‘good deal’ or ‘not worth the bill’.

## Real Estate Analysis

A renowned ongoing Kaggle competition with the name House Prices - Advanced Regression Techniques [2], intended for advancing ML knowledge and sharing it deals with the problem of predicting real estate prices based on various features (81 to be exact). The dataset consists of roughly 3000 samples split equally between train and test. With more than 30 thousand submissions of various skill levels, different approaches are displayed, plenty of them based on tree models with varying degrees of success.

Predicting Housing Prices: Simple Approach [3] uses multiple decision tree based algorithms, making it a great resource for this project. However, due to the vastly different dataset, we cannot expect the models to transfer without some additional optimization and oversight.

A paper Identifying Real Estate Opportunities Using Machine Learning [4], written by A. Baldominos et al. details a search for the best performing regression model for predicting real estate prices in the section of Madrid, Spain. It considers 4 categories of models, namely k-nearest neighbors regression, support vector machines, ensembles of decision trees and multi-layered perception. Out of those, the decision tree based model is performing the best, based on several metrics. Authors explain feature selection and transformation using one-hot encoding for non-continuous variables.

The paper is a great example of a project proposal with very in-depth explanations of the procedure. Decision tree based models will be further examined and compared among themselves.

The paper ‘Housing Price Prediction via Improved Machine Learning Techniques’ [5] by Q. Truong et al. deals with the Kaggle dataset ‘Housing prices in Beijing’ and explores Random Forest, LightGBM, XGBoost, Hybrid regression and Stacked Generalization. The metric used for evaluation or the models is root mean squared logarithmic error (RMSLE), which offers an advantage over the RMSE as the effect of the difference between predicted and actual price is the same regardless of the price level. Performance of the methods is very closely matched on the test dataset, the best being Stacked Generalization regression with a score of 0.16350 and the worst LightGBM with 0.16944. The paper states that although Random Forest does well on the train set, it is due to overfitting and it does not perform as well on the test.

## Interpretation

Scott Lundberg presents numerous examples of implementation of SHAP package for explanation and understanding of the model on the individual level of a sample.

It comes as an ideal tool for this project as identifying drivers of the price of a property is directly one of the goals, as well as being a very useful insight into the fabric of the market.

# Methodology

## Collecting the Data

For the purpose of this project, a dataset was collected from the site bolha.com/nepremicnine, through the process known as web scraping, performed in Python using Selenium and BeautifulSoup4 libraries. It was then manipulated and stored as Pandas data frame and .csv format for the ease of use in the following phases. Given the nature of user-generated ads and the inconsistent information available, extensive preprocessing was necessary to prepare a useful dataset. It consisted of manipulating values for each feature to obtain desired values and formats, removing the outliers and imputing to prepare numeric and categorical features.

## Exloratory Data Analysis

Importance of intuition cannot be overlooked when designing and interpreting a machine learning model. Using Matplotlib and Seaborn, relationships between features were visualized to gain the insights. The following Figure 1. shows the heatmap of all numerical features and the Figure 2. shows a combination of histograms and scatter plots of *price*, *sizeM2*, *year\_built*.

## The Model

The purpose of this paper was to develop the best decision tree-based model possible given the data scraped. The following models were considered: RandomForest, Gradient Boosting Regressor, XGBoost and LightGBM. Each of those models has been evaluated without specifying parameters and then compared with the rest.

The models were optimized using grid search GridSearchCV method from Sci-kit learn.

Lastly, the models were stacked and the stacked model evaluated in comparison to individual ones.

## Testing and Evaluation

Models was evaluated through root mean squared logarithmic error (RMSLE) [6] method. This method is selected to avoid bias towards the more expensive properties.

### Unoptimized model performance

Table 1 shows how the models stand before optimization. XGBoost shows the best performance on the train dataset, with a very small error of 0.0086. However this outstanding performance is very likely due to overfitting, as the scores for test dataset indicate. The lowest score of 0.562 is achieved by Gradient Boosting Regressor, followed closely by LightGBM and then XGBoost. RandomForest regressor performs the worst judging by the RMSLE score on test data.

| Scores of unoptimized models | | | |
| --- | --- | --- | --- |
|  | Name of the model | RMSLE\_traina | RMLSE\_testa |
| 0 | Random Forest | 0.131437 | 0.616475 |
| 1 | GBoost Regressor | 0.192764 | 0.562449 |
| 2 | XGBoost | 0.008577 | 0.581207 |
| 3 | LightGBM | 0.121851 | 0.576637 |

1. RMLSE stands for a performance metric root mean squared error of logarithms of price.

### Optimized model performance

Table 2 shows how the models compare after grid search optimization [7] was performed. In comparison with previous scores before optimization, the scores of all models on the train dataset worsened, while the scores for the test set improved.

| Scores of optimized models | | | |
| --- | --- | --- | --- |
|  | Name of the model | RMSLE\_traina | RMLSE\_testa |
| 0 | Random Forest | 0.204033 | 0.590414 |
| 1 | GBoost Regressor | 0.027424 | 0.555606 |
| 2 | XGBoost | 0.122582 | 0.517286 |
| 3 | LightGBM | 0.245088 | 0.558541 |

1. RMLSE stands for a performance metric root mean squared error of logarithms of price.

The best performance on the test is achieved by XGBoost, the worst by RandomForest regressor.

The difference between the best and worst model is increased after optimization.

### Stacked model performance

Figure 3 shows the performance of each model before optimization together with stacked model with secondary model being LinearRegression. The method of performance evaluation in this case is mean squared error as it is a built-in function.

Surprisingly, the stacked model does not show any notable improvement over the best performing Gradient Boosting Regressor.

## Interpretation

Scores for all models improved with the grid search optimization, despite train scores worsening (except in the case of GradientBoostingRegressor). I believe that this is the result of overfitting, at the default parameter settings, which is held back with selection of parameters for optimization.

The biggest improvement from the optimization is seen on the XGBoost model, which performs better after the max\_depth is set to 4 instead of default 6.

Feature importance hierarchy is very similar among all models (optimized models only considered, however it mostly applies to unoptimized too). Table 3 shows the three most and three least important features.

The three most important are always the size of the flat (sizeM2), region, where it is located (l-region) and the year the building it sits in was constructed (year\_built).

The three least important features are willingness to allow the possibility to switch the property for another (switch), whether it has a garden or a shed (garden) and the type of building the flat is located within (sits\_in).

Figure 4 shows shap plots for each model. Plots show the impact of features on the final prediction of the model.

| Feature importance | | | | |
| --- | --- | --- | --- | --- |
| **Model** | Random  Forest | GBoost regressor | XGBoost | LightGBM |
| Most important features | | | | |
| 1 | sizeM2 | sizeM2 | l-region | sizeM2 |
| 2 | l-region | l-region | sizeM2 | l-region |
| 3 | year\_built | year\_built | year\_built | year\_built |
| Least important features | | | | |
| 1 | switch | switch | switch | switch |
| 2 | garden | garden | garden | sits\_in |
| 3 | sits\_in | sits\_in | sits\_in | balcony |

1. The numbers on the left indicate the order of feature importance, where 1 means both most and least important, depending on the cas

# Insights and Conclusion

Web scraping for data, preparation and the issues related to it consumed the majority of the time spent on the project.

More feature selection should be done to obtain a better model. In this case, it was done manually with filtering, however an automated solution should be added to improve.

The most important features are obvious; however, it is valuable information that the data in this case supports our understanding and intuition in the real estate market. It is interesting that the size itself is such a clear indicator of price.

Counter-intuitively, the stacked model does not show improvement over the unoptimized models. In my opinion, it is due to two factors, the first being the overlap of the individual models, in the way that they pick up on the same features and miss the others, and second being my lack of   
  
  
experience and expertise when it comes to designing such models.

The model shows some promise; however, it lacks data points, proper feature selection and model hyperparameter tuning. It will be improved further as my knowledge base has grown.

Through the process of creation of the paper I have learned a great deal of things, but if I had to choose one, it would be to appreciate a well-designed and prepared dataset.

##### Acknowledgment

I thank my two mentors, Andrija Štajdohar and Tomislav Lipić, for help, guidance, patience and encouragement throughout the course of the project.

# References

1. L. Tašler, ‘Nepremičnine.net Scraper’, Github: <https://github.com/ltasler/nepremicnine.net-scraper>
2. ‘House prices-Advanced regression techniques’ Kaggle dataset <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview>
3. Abhinand, ‘Predicting Housing prices: Simple approach’ Kaggle: <https://www.kaggle.com/abhinand05/predicting-housingprices-simple-approach>
4. <https://www.mdpi.com/2076-3417/8/11/2321/htm>
5. Truong et al. ‘Housing Price Prediction via Improved Machine Learning Techniques’ Elsevier: <https://www.sciencedirect.com/science/article/pii/S1877050920316318>
6. J. Moody,’What does RMSE really mean’ Towards data science: <https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e>
7. J. Brlek, Capstone project; Fitting models, Github: <https://github.com/jbrlek/Capstone_webScrape_Real_Estate_Regression/blob/main/Fitting%20models.ipynb>
8. ‘Machine Learning Crash Course’ Google: <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>
9. S. Lundberg,’Shap’ Github: <https://github.com/slundberg/shap>
10. ES, Shahul, ‘Hyperparameter Tuning in Python: a Complete Guide 2021’ Neptune: <https://neptune.ai/blog/hyperparameter-tuning-in-python-a-complete-guide-2020>
11. W. Koehrsen, ‘Hyperparameter Tuning the Random Forest in Python’ Towards Data Science: <https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74>
12. M. Restrepo, ‘Doing XGBoost hyper-parameter tuning the smart way — Part 1 of 2’ Towards Data Science: <https://towardsdatascience.com/doing-xgboost-hyper-parameter-tuning-the-smart-way-part-1-of-2-f6d255a45dde>
13. K. Kaczmarek, ‘Understanding LightGBM Parameters (and How to Tune Them)’ Towards Data Science: <https://towardsdatascience.com/understanding-lightgbm-parameters-and-how-to-tune-them-6764e20c6e5b>