Automated valuation of commercial property

Project report for the module “Interdisciplinary project”

Universität Potsdam

Computational Science (Master of Sc.)

Author: Stepanov, Mikhail

Student ID: 811991

Reviewer: Prof. Dr. Anna-Lena Lamprecht

Date of submission: 28.10.2023

Table of contents

1. Introduction

The valuation of rent-oriented investment properties such as multi-family houses with many residential units and commercial buildings is a great challenge due to their uniqueness and lack of good comparable properties. In contrast to the classic own-use properties, such as individual flats, single-family houses, semi-detached houses, whose acquisition is more emotional and personal, purely economic factors are in the foreground for investment properties. The most important factors influencing the value of investment properties include, for example, the current annual net cold rent (ANCR) of all units in the property and their difference to the market rents in the region, the condition of the property, the amount of maintenance reserves, the micro and macro location of the property and the current economic situation on the real estate market. As an expert in the valuation of investment properties with many years of experience, I have already been able to value several thousand properties throughout Germany. The valuations of the properties were carried out by means of a prepared Excel tool using the discounted cash flow (DCF) method. Depending on the complexity of the property, this process can be very time-consuming and take several hours in individual cases. With the development of machine learning techniques several attempts have been made to automate the valuation process of the houses [1]–[3]. However, most of the projects were limited to small owner-occupied flats or single-family houses, because it is much easier to find out a certain pattern for such properties, which makes it possible to assign all properties to a common cluster according to a specific criterion and train an accurate machine learning model. Moreover, the total database for this kind of properties is much larger compared to the investment properties due to a very high demand. Abigail Bola Adetunji et al. investigated in their work the effectiveness of the Random Forest model for the task of predicting prices of residential houses by using the Boston housing dataset and were convinced of the effectiveness of this algorithm [3]. The prices predicted by the model were within a 5% interval of the actual house price. Raul-Tomas Mora-Garcia et al. compared several ensemble models such as Gradient Boosting Regressor, Extreme Gradient Boosting, Light Gradient Boosting Machine, Random Forest and Extra-trees regressor with linear regression models on the task of predicting of residential houses prices by trying to identify the best and also to analyse the influence of Covid-19 on the house prices in Spain [1]. They concluded that ensemble models perform better than simple linear models and that the Covid-19 pandemic did not affect prices as much as the 2008 crisis. The only paper, that could be found during this project, where the study also included commercial property was the paper of Walter Coleman et al., where they added information from dynamic mobile location data and analysed the impact of the availability of this data on the quality of the model [4]. They found that the model with this information performed slightly better and showed a 3% lower MSE. However, this work did not focus exclusively on commercial property and did not fully consider the differences between commercial and residential property.

Despite the complexity of the task of commercial property valuation and modest database in comparison with residential property market, the amount of data is also increasing year by year. Also, during my work in the field of commercial real-estate, I found that despite many differences in each individual property, a certain pattern of characteristics can be still identified that allows to group commercial objects into similar price categories and thus analyse how these characteristics influence the price of the property. This fact and the significant grow of

our intern reference portfolio and database were the main reasons to start searching for solutions in automatization of the valuation of commercial properties by using the machine learning methods. In this project such machine learning algorithms as Linear, Lasso, Ridge regression, Decision Tree, as well as group of ensemble methods including Random Forest, AdaBoost and LightGBM were applied for this purpose. After intensive data preprocessing process, tuning of hyperparameters and validation of models the median absolute percentage error on the test set for the best model of about 8% was achieved, which is a very good result, that even outperforms the average error spread in the manual evaluation of objects by different experts.

2. Data preprocessing

For the purposes of this project the valuation data from the period between March 2018 and June 2023 carried out by our commercial department were taken. The total number of valuations carried out across Germany in this period is around 15.000. The only source of price-influencing information for each individual property is the Excel tool used for the valuation. This tool includes not only the property-specific characteristics such as for example the net annual cold rent or residential and commercial area, but also the information about the current market situation, such as the market rent and the vacancy rate in the location. Therefore, the first phase of the project was to extract the information from the Excel tools and create a dataset for all properties. It had to be considered that there were at least six different tool versions during these five years. Also, since the valuations were caried out manually by experts with different experience all possible human errors in the data had to be taken into account and handled accordingly. For the data extraction from the Excel files, a Python script was written that first collected all paths to the Excel files in a dictionary, with the city names used as keys and with a list of all paths to the Excel files with the valuations in this city.

This project was limited to the investment properties from 100 largest cities only. The main reason for this limitation was the lack of references from other smaller cities. Also, it was considered that the values from the large cities are representative for the entire region. And can be then extended for other regions. The following property characteristics were taken as object features: city name (City), address, living area (LivingArea), commercial area (CommercialArea), annual net cold rent (ANCR), DCF-price, average market rent (AMR), maintenance reserve, Capex, the date of valuation as time code and in the D-M-Y format. This resulted in a dataset with a total of 6.486 properties.

Initially, several additional features that were calculated from those already existing in the dataset were added, in order to facilitate the filtering process. One such feature is the total area of the property, calculated as the sum of commercial and residential area. Information on commercial share in relation to the total area has also been added. Another important feature of the investment properties is the so cold multiplicator (or factor), that can be calculated as DCF-price divided by ANCR and can be interpreted as the investment payback period. The factor is the most meaningful criterion for quickly assessing the quality of the collected data and identifying outliers of any kind.

The first part of preprocessing was the filtering of outliers. This filtering was done on three main characteristics of the property and market condition: living area, DCF price and average market rent. The objects with the living area of smaller than 250 m² and larger than 2.000 m² were filtered out, as this group of houses represent extremely rare outliers that can negatively impact the generalization ability of the model and thus significantly reduce the quality of the model prediction for the new unseen data. For the same reason, properties whose AMR was less than 15.000 € and more than 250.000 € were filtered out. Finally, only houses with the DCF price between 140.000 € and 3.500.000 € were left in the dataset. In addition, the properties were filtered out for which only the default values were entered as the area and rent data, since it meant that the valuation of this object had not yet been started. Depending

on the tool version, the default values for living area and ANCR were 1.000 m² and 60.000 € or 600 m² and 48.000 € respectively. After this filtering, the number of objects decreased to 3.212.

During the data preparation, all objects with missing information such as for example missing area data were removed from the dataset. There were only 37 of such objects in the whole dataset, which was approximately 1% of all data and had not significant impact on the model performance. This was a reason, why these objects could be just dropped from the dataset, without further analysis.

Based on the experience as well as reliable market reports, the usual market factors, regardless of location class, are in the range of 8-40 [5]. Any values outside this range in most cases represent either an unprofessional valuation or very property-specific features such as for example lucrative amenities or very large plot with additional development potential or for example extremely bad condition of the house. These outliers should also be removed from the dataset. Figure 4 in the Appendix of this report shows the distribution of factors in our dataset before and after removing of the outliers outside of these range.

Subsequently, each property was assigned a location class according to the city. According to this classification, the locations are divided into the following four categories: A-location: metropolis of international importance with very high real estate demand and very high purchase and rental prices, B-location: major city with great national and regional importance with high real estate demand and high purchase and rental prices, C-location: city with limited national but great regional importance with varying real estate demand, D-location: cities that are only important for their immediate surroundings. Afterwards, the factor ranges for different city classes were also further adjusted based on the experience and the information from the usual property market reports such as E&V market report [5]. The objects which factors were outside of this region were also considered as outliers. Thus, only properties in the factor range between 8 and 21 were considered for city class D, between 11 and 24 for city class C, between 13 and 28 for city class B and between 17 and 35 for city class A, which reduced the total number of objects in the dataset to 2.495.

Although the investment properties include several different types of real estate, such as office buildings, supermarkets, residential and commercial buildings with a high proportion of commercial area, it was decided in this project to limit only to multi-family houses and residential and commercial buildings with a low proportion (less than 15%) of commercial area. All other kinds of properties represent a very special class with the lack of references and with very specific individual features and thus are not directly comparable with multi-family houses. So, one of the steps in preparing the data was to filter out the special investment properties and houses with more than 15% of commercial area. After this filtering the total number of objects was reduced further to 1.929.

At the last stage of the general preprocessing all duplicates were also removed from the dataset, whereby only those properties where all information is identical were considered as duplicates. Another class of duplicates are valuations that belong to one property but were made at different times and probably by different experts and thus differ in valuation price. These price differences can also have several other reasons. Some of them are subjective

difference of opinion between the valuation expert and the real-estate agents or also constantly changing market situation for example in times of crisis. Since the goal of this project was to build the generalized model that able to simulate a valuation expert, this kind of duplicates could even improve the quality of the model. Also, some features, that were calculated from other existing features were dropped to avoid duplication of linear-dependent features. After this final step, the sample size was reduced to 1.895.

Since all linear models used in this project and the sklearn implementation of the most ensemble methods require numerical representation of features, the machine learning technic called one-hot encoding was applied for all categorical features. In the one-hot encoding process, categorical features are represented as 0 or 1, answering the question whether a given object belongs to this category or not. For example, Berlin in the attribute “City” can be represented as a separate feature and get values 1 if the object is in Berlin and 0 otherwise. In the dataset used in this project the city name, province name and the city class are the only categorical features. While the province and city name were one-hot encoded, a more representative numerical representation was found for the city class. Thus, the city classes were encoded through the corresponded average factors. It has been widely discussed that a significant increase in the number of features due to one-hot encoding can have a negative impact the quality of the model [6]. The LightGBM ensemble model also used in this project is implemented so that categorical features can be used as they are and do not require numerical representation. So, the original categorical representation of the city name and province was used for this model.

In the later stages of the project in the process of model analysis a more comprehensive representations of the already existing features were tried out to increase the model performances. As it is already mentioned above, the date of valuation is originally represented as a timecode. Thus, this feature is unique for each object and in such a form the linear model cannot reveal certain trends in the price changes of objects for certain time periods. However, based on the experience and knowledge of very heterogeneous market dynamics in recent years caused by several crises, it was required for the models to be able to clearly track the impact of the valuation period on price. For this purpose, the feature with the valuation date of the property was first represented in quarters starting from the current moment, where the second quarter of 2023 (the most current valuations) was represented by 0, the first quarter of the current year by 1 and so on for all past quarters ending with 21 for the earliest valuations and subsequently one-hot encoded. So, the linear model could fit a specific coefficient for each quarter separately and all ensemble models could use them separately for splits. The impact of one-hot encoding of this feature on the model performance is shown on the Table 10 in the Appendix of this report.

Also, in the process of searching for a better representation of features, the values of such features as DCF-price, ANCR and AMR were divided by the total area and represented for a square meter, and not for the total area in some experiments. It was assumed that by getting rid of multiplicative features, the data would become more standardized, what could also increase the model performances. By making this transformation, we get rid of additional noise imposed by the area, making the dependence more linear. The real impact on the performance after this transformation will be shown in Part 5.

As mentioned above, since the valuations were done manually by humans, the data collected contained poor quality valuations, with various anomalies such as unrealistically low/high valuation price due to low professionalism of the valuation expert, erroneous values due to typo and so on. To detect such anomalies the objects with the largest prediction mistake have been investigated more thoroughly. The list of such properties included all houses for which the price predicted by the model was outside the 30-% interval of the actual DCF-price. All these objects were analysed manually by checking of the corresponding excel files. All objects with obvious anomalies that could not be accurately represented by available features were removed from the dataset. The results of this filtering will be discussed in Part 5.

Later in the project such additional features about market situation and based on market reports as vacancy rate and average m^2-prices for different locations were added to give the models more local information. The impact of adding these features on the model performances can be seen on the Table 11 in Appendix.

3. Models

The task of predicting the houses prices is a supervised regression problem. In supervised learning there are objects with their descriptive characteristics (features), as well as the targets corresponding to each object. The goal of supervised machine learning algorithms is to train the model to predict the certain target variable [7]. A regression task in machine learning is called a problem in which the target variable is continuous [8].

In this project different regression models are considered that are suitable for this task. This part will describe the basic mathematical foundations of the models used, as well as their advantages and disadvantages, relative to each other. All models were carefully evaluated with the problem-specific formulated metrics described in Part 4.

3.1 Baseline model

A baseline model was used in this work in order to quickly assess the quality increase of more complex models, when certain improvements are added. For this purpose, the simple linear regression model, trained only with one feature (AMR) was chosen, since its quality was already relatively noticeable and representative.

3.2 Linear models

In this part basic mathematical foundations of linear models, such as simple linear regression and linear regression with L1 (Lasso) or L2 (Ridge) regularisations will be discussed.

The linear regression is the simplest linear model, that describes the relationship between one or more independent variables x_1 ... x_{k-1} and the dependent variable y. For n measurements this relationship can be represented by the following formula [9]:

$$y_i = b_0 + b_1 x_{i1} + \cdots + b_{k-1} x_{ik-1} + \epsilon_i \quad (1),$$

where ϵ_i is a random observational error (residuals) and $i = \overline{1,n}$ [9].

It can be represented in matrix form that combines all n samples:

$$\boldsymbol{Y} = \boldsymbol{XB} + \epsilon \quad (2),$$

where $\mathbf{Y} = [y_i]_n$, $\mathbf{X} = [x_{ij}]_{n \times k}$, $\mathbf{B} = [b_j]_k$, $\epsilon = [\epsilon_i]_n$.

The goal of the linear regression is to estimate the unknown coefficients $\widehat{\boldsymbol{B}} = [\widehat{b_j}]_k$ so, that the following ordinary least squares expression (3) is minimized [9]:

$$\boldsymbol{min}_{\boldsymbol{b}} \sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{k-1} b_j x_{ij}\right)^2 \quad (3).$$

The coefficients $\widehat{\boldsymbol{B}}$ can be then find by using of following formula, given $\det \boldsymbol{X^t X} > 0$:

$$\widehat{\boldsymbol{B}} = (\boldsymbol{X^t X})^{-1} \boldsymbol{X^t Y}$$

The predictions $\widehat{\boldsymbol{Y}}$ that correspond to the features $\boldsymbol{X}$ can be defined as follows:

$$\widehat{\boldsymbol{Y}} := \boldsymbol{X}\widehat{\boldsymbol{B}}$$

The closer the predictions vector $\widehat{\boldsymbol{Y}}$ to the real targets $\boldsymbol{Y}$ the better the regression model fits the data $\boldsymbol{X}$ [9]**.**

Although simple linear regression can be effectively applied to a wide range of problems, it has several limitations. Firstly, linear regression, by its nature, attempts to linearly approximate the data, and therefore assumes a linear relationship between the feature and the target. However, in many cases, the relationship between the data is more complex and non-linear [11]. Secondly, simple linear regression works effectively when there is no correlation between the dependent variables, which is also not always the case. In this case, the solution for the coefficients becomes very unstable. This effect is called multicollinearity and to solve this problem, the regularisation can be used. In this case, an additional shrinkage parameter is introduced in the search for optimal coefficients, which controls their size, makes solution more stable and shrink or even nullify the most correlated or the most unimportant of them. Some of the most applied linear models with regularisation are Lasso and Ridge regression [9].

The coefficients determined by Lasso regression tend to be sparser. So, it performs a so-called feature selection reducing the coefficients of the less important features that have the smallest effect on the target variable to zero and leaving only the important features, thus also increasing the interpretability of the model [9], [10].

In Lasso regression, an L1 regulariser is used and the equation (3) takes the following form [10]:

$$\boldsymbol{min}_{\boldsymbol{b}} \ \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \left(y_i - \sum_{\boldsymbol{j=0}}^{\boldsymbol{k-1}} b_j x_{ij}\right)^{\boldsymbol{2}} + \boldsymbol{\lambda}\|\boldsymbol{B}\| \quad (4).$$

Here the OLS (3) is penalized by additional term $\lambda\|\boldsymbol{B}\|$, where λ is a constant regularization parameter and $\|\boldsymbol{B}\|$ is L1-norm of the coefficient vector [9], [10].

Unlike the Lasso regression, the Ridge Regression model is only capable of compressing coefficients, but not completely nullifying them [9].

In the Ridge regression, equation (3) is augmented as follows [10]:

$$\boldsymbol{min}_{\boldsymbol{b}} \ \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \left(y_i - \sum_{\boldsymbol{j=0}}^{\boldsymbol{k-1}} b_j x_{ij}\right)^{\boldsymbol{2}} + \boldsymbol{\lambda}\|\boldsymbol{B}\|^2 \quad (5),$$

where $\|\boldsymbol{B}\|^2$ is a squared L2-norm.

The method used to select the optimal shrinkage parameter $\boldsymbol{\lambda}$ will be discussed in the Part 4.

3.3 Decision tree

Although linear models can be quite efficient, they have several limitations. As it was already mentioned in the previous part, one of the main concerns of such models is that they are effective only if the relationship between features and target variable is linear, which is not always the case [11] (see Figure 5 in Appendix as an example). Linear models assume that the relationship between features and target can be described by an analytical function and solve the problem of finding an approximation of this function that best describes the data on the entire dataset. However, such an analytical solution to the problem is not always possible [11]. There are many non-linear approaches in machine learning that can efficiently solve these problems. One of the basic non-linear models is the decision tree.

The decision tree algorithm relies on a method called divide-and-conquer. It is algorithm relies on a method called divide-and-conquer, in which one complex mathematical problem with a large dataset is split into many smaller subtasks, each with its own subset of data. There are two types of decision tree nodes: internal and terminal. An internal node evaluates a decision function and chouse the next child node that should be visited. A terminal node is associated with one of the partitions of the input space and has no childes. In regression trees the terminal node contains a model that maps a continuous value to the given input object [11].

In simple words, the decision tree algorithm splits the training dataset into subgroups, based on some condition for one of the features. It investigates all features in the dataset and the goal of the algorithm is at each level to select the best possible split that most strongly reduce uncertainty in the data. There are several algorithms that can be used in decision tress. In this project, the default sklearn implementation of decision trees were used, that bases on a CART (Classification and Regression Trees) algorithm, where the decision tree is first grown as deep as possible and then pruned back on the principle of minimum cost complexity [10]. In the CART algorithm, the training data are also sequentially split into subsamples by the split that maximises the empirical risk reduction for the selected loss function (in this project, the basic splitting criterion from sklearn was used - the mean squared error) [11].

The main problem of all decision tree algorithms is that they are highly susceptible to overfitting – the effect when the model is too strongly fitted to the training data but works poorly for the new unseen data [12]. Thus, for example if the tree is not limited in depth it will grow until all objects in the training sample are perfectly predicted, however, since the intrinsic pattern underlying the data was not identified, this model will be most probably ineffective for new data. The tuning of hyperparameters of Decision Tree may help to reduce overfitting. The main hyperparameters of Decision Trees are the maximum depth to which a tree can be grown, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, the number of features to consider when looking for the best split, etc.

One of the main advantages of a decision tree is the high interpretability of the results. A small decision tree can be visualised, and the conditions on which the splits were conducted can be analysed. The way a decision tree works is very similar to the way a human makes decisions. Regardless, even with careful selection of hyperparameters, the decision tree can still overfit greatly. Also, the decision tree can be sensitive to outliers in the data [11], [13]. However, in

cases where a single decision tree does not perform well, ensemble models that combine multiple trees can be an effective tool.

3.4 Ensemble methods

Ensemble method is one of the machine learning techniques in which a large number of weak estimators (for example non-pruned decision trees) are combined into one large model. This combination of weak estimators is often more generalised and robust than a single complex model and may be less susceptible to overfitting [10], [11]. This part will cover commonly used ensemble methods such as Random Forest, Adaptive Boosting Regression (AdaBoost), as well as a more modern and advanced method, LightGBM.

3.4.1 Random Forest

In the Random Forest method, many of non-pruned decision trees are combined to solve a single problem. An important condition for a Random Forest is that all trees in the forest should be as independent of each other as possible. This is achieved by training each tree on its own randomly selected subset of data, and each object in this subset is formed from a randomly selected subset of features. This decorrelation of trees from each other should reduce the overall variance of the model [11], [14].

So, the stages of Random Forest algorithm can be briefly described as follows [11]:

1) Generate randomly sampled subset of training data B.
2) Fit to each set B a decision tree, by using a random subset of size n' of features for splits.
3) At each split store the empirical risk reduction for the corresponding split variable.
4) Return the final prediction as the average of the predictions of all the trees in the forest.

The main advantages of Random Forest include the ability of reducing overfitting in decision trees, efficient dealing with non-linear parameters, often high performance on large and complex data, ability to work with both numerical and categorical types of variables and the possibility to evaluate the feature performance based on the analysis of selected splits [11]. The main disadvantages include a longer training process and limited interpretability of this algorithm compared to the single decision tree [14], [15].

Only new hyperparameter in Random Forest compared to simple Decision Tree is the number of trees in the forest.

3.4.2 AdaBoost

Another group of effective ensemble methods are boosting algorithms which also combines many weak estimators. One of the most popular boosting methods is Adaptive Boosting (AdaBoost). Unlike Random Forest, in AdaBoost the weak estimators depend on each other and are trained sequentially based on the quality of predictions of the previous estimator [11], [16]. At each training step, different weights are assigned to the objects, while the objects with

the highest prediction error in the previous step are assigned a higher weight. Thus, objects that are more difficult to predict during training receive more attention from the estimators. The first estimator is trained on original data, since at the first step all objects are weighted equally. The main steps of the algorithm can be described as follows [17]:

1) Initialize all object weights w_i with 1.
2) Pick N objects from the training set with the probability $p_i = w_i / \sum_{i=1}^{N} w_i$.
3) Construct a weak estimator for this training set (in this project Decision tree with depth of 3).
4) For every object in the training set make a prediction and calculate the individual loss function L_i.
5) Calculate an average loss $\bar{L}$ of all objects.
6) Calculate a measure of confidence of this estimator by the following formula: $\beta = \bar{L} / (1 - \bar{L})$.
7) Update weights of the objects by $w_i = w_i \cdot \beta^{(1-L_i)}$.
8) Repeat it for T estimators in sequence.
9) Return the final prediction of an ensemble as the weighted median prediction of all estimators.

Here the weight of an estimator characterises the degree of contribution of an individual estimator in the final prediction. This is a one of the tunable hyperparameters of the AdaBoost model. Another important tunable hyperparameter is the number of weak estimators T.

AdaBoost algorithm has been shown to be effective in many tasks. One of its main advantages is a relatively small number of hyperparameters, which facilitates the process of its optimisation. However, it also has several disadvantages. One of them is a high sensitivity to outliers and noise in the data [14]. Also, at least in this project, the training process of this algorithm was longer compared to Random Forest.

3.4.3 LightGBM

Another effective type of boosting is a method called gradient-boosted decision trees (GBDT). The goal of the algorithm is to reduce the prediction error in the stagewise manner by propagating the error through many weak estimators. The first tree should be formed so that the overall loss is minimised as much as possible. The second tree is formed based on the residuals of the objects in dataset from the first tree, but the splits of the second tree may be different from the first tree. These residuals are calculated by using the gradient of the chosen loss function on all objects. This process continues until a certain stopping criterion (such as for example the number of trees) is reached or until the model perfectly predicts the targets [18]**.** The final prediction of the model is then a weighted linear combination of all these trees [11].

In this work, a state-of-the-art improved method relying on GBDT, called LightGBM, was used.

Unlike many other algorithms, it uses a leaf-wise tree growing process and not a depth-wise one. In the leaf-wise approach the next split occurs on a leaf on which the loss function will be reduced as much as possible. This technique can speed up the learning process and result in less loss compared to depth-wise grown trees but may cause overfitting [19]. Another

special property of this algorithm is that it uses a feature bundling technique called Exclusive Feature Bundling (EFB) that bundle sparce features together to speed up the algorithm. This mechanism can be very effective when the feature space is very large, for example when there are many one-hot encoded features in the dataset [20]. However, in this project this additional feature of this approach was not used, since in the standard implementation of LightGBM the categorical features can be used as they are and the feature space of our data without one-hot encoding was relatively small.

LightGBM has a very large number of tunable hyperparameters, which makes this method flexible and gives a large space for tuning the model but makes the parameter fitting process more difficult. The main parameters of this algorithm are number of estimators, number of leaves in each tree, maximal depth of each individual tree, minimal amount of data in each leaf, bagging fraction (randomly select a fraction of all objects) and bagging frequency (tells how often the algorithm needs to re-sample the data) [19]. This algorithm is also relatively fast compared to the many other ensemble models. The big advantage of this algorithm over the others is that in its basic implementation it is possible to use categorical features, thus keeping the dimensionality of the sample quite low. Its weak point, however, is its noticeable tendency to overfitting, which can however be controlled, with careful selection of hyperparameters.

4. Evaluation of models

Model evaluation is a big and highly important part of machine learning to create a high-quality model. This part will discuss the techniques used to overcome the overfitting problem when training a model, as well as describe metrics for comparing the quality of models against each other. It will also describe in detail the process of tuning of hyperparameters used in this project.

4.1 Train-test split

One of the main indicators of a high-quality machine learning model is its ability to identify a generalized trend in training data and the ability to work effectively on new data. When training a model, it needs to be considered that the training data often contains various errors, noise and outliers and testing the model performance on the same data that were already used for the training is often not the best practice. To check the quality on the new data the entire dataset is often divided into training and test samples, using the training data to train the model and the test data to evaluate the model. This approach allows to detect and control possible overfitting of the model [21].

In this project, the splitting of the data into train and test occurred randomly each time the experiment was run, wherein 80% of the data were used for training and 20% for testing the models. To assess the quality of the model's predictions, independent on particular sampling each individual model was trained five times and the models' performance measures were averaged.

4.2 Cross-validation

However, it is often not enough to split the sample into train and test just once to ensure that the model is highly generalised and will perform well on unseen data. One good way to improve the effect of such data splitting is a technique called cross-validation, in which the data is partitioned many times. The original dataset is split into n separate chunks of equal size, and model training is repeated n times, with each iteration using n-1 chunks of data for training and then testing the model on the remaining data. Thus, after n iterations, all n chunks should be used both for training and for testing. The final prediction of the model is then an average prediction after all n runs [22]. Also, cross-validation is often used to tune the hyperparameters of a model. For this purpose, the data are first split into train and test set, then n-fold cross-validation with a particular set of hyperparameters is performed on the train data, and after all hyperparameters of interest have been tested on cross-validation in this way, the hyperparameter on which the model has shown the best result is selected. The model trained with this hyperparameter is considered to be the best and the final testing is performed on the initial test set, which was never used in the training process [10]. This process is illustrated on the Figure 1 from sklearn documentation. There is a tool that allows us to test all combinations of hyperparameters on cross-validation and select the best combination. This tool calls GreadSearchCV and it was also used in this project. It goes through the entire set of

passed hyperparameters, tests all possible combinations of them using n-fold cross-validation and returns the best model trained on the best combination of hyperparameters [10].

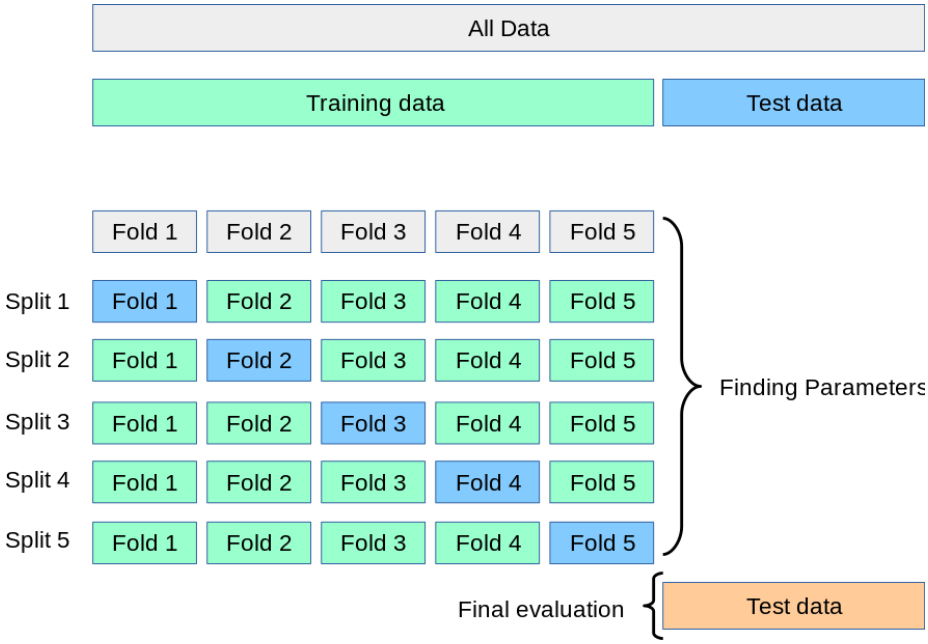

Figure 1. *Vsualization of the process of hyperparameter tuning using 5-fold cross-validation* [10].

For each model considered in this project, a large number of different hyperparameter combinations were tested with GreadSearchCV. Gradually, a smaller range of hyperparameters was found in which the effect of overfitting is least noticeable. For the Decision Tree algorithm when selecting the best hyperparameters, trees with depths from 3 to 5 were considered. As the minimum number of samples required to be at a leaf node, 10, 15, 20 were considered, and as the minimum number of samples required to split an internal node, 25, 35, 50. For Random Forest algorithm, the following values were tried for the number of trees in ensemble: 60, 150, 300. Also, the values 3, 4, 5 were tried out as maximum depth of each tree, as the minimum number of samples required to split an internal node: 20, 30, 40 and as the minimum number of samples required to be at a leaf node: 10, 15, 20. For AdaBoost algorithm 300, 600 and 700 were tested for the number of estimators and 0.01, 0.03, 0.06, 0.1 as the weights of an estimator. For the LightGBM algorithm 100, 200 and 300 estimators were tried out, as a number of leaves in each tree 2, 3 and 5, as maximal depth of each individual tree 2, 3 and 4, as minimal amount of data in each leaf 20, 25 and 30, as bagging fraction 0.1, 0.2, 0.3 and as bagging frequency 2 and 4 were investigated.

To ensure additional reliability of the models' quality metrics, this project also used a slightly modified cross-validation approach described in Part 4.1, where the models were trained and tested five times on differently sampled data. In each run the best combination of hyperparameters found by GreadSearchCV algorithm was used. The difference of this

approach compared to the original cross-validation was that objects in the test data could be partially repeated at each iteration.

4.3 Evaluation metrics

The evaluation metric allows us to assess the quality of the model's performance and compare different model to each other. There are many different metrics to assess the quality of a model from different perspectives. Along with classic metrics for regression problems such as mean squared error (MSE), mean absolute error (MAE) and coefficient of determination (R^2), a group of other more task specific metrics were chosen, that are more representative for understanding of the model performance specifically for the price prediction problems. Such metrics as median absolute error (MDAE), median absolute percentage error (MDAPE), mean absolute percentage error (MAPE) were also analysed.

For calculation of MAE the following formula was used [23]:

$$\boldsymbol{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}(x_i)| \quad (6),$$

where $\widehat{y_i}$ is the model prediction for i-th object.

The median absolute error can be calculated as follows:

$$\boldsymbol{MDAE} = \boldsymbol{med}\,|y_i - \widehat{y_i}(x_i)| \quad (7),$$

The MSE was calculated by using of the following formula:

$$\boldsymbol{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i}(x_i))^2 \quad (8),$$

The main motivation for using median-based metrics is that they are less influenced by outliers than metrics that use means. But metrics that rely on mean values are in turn good in problems where large computational errors need to be reduced [23].

In another approach in error estimation, errors were treated as ratios to actual targets. In a price estimation task, such metrics can be much more informative due to the fact that the error on the same amount for a very expensive object and a very cheap object differ significantly [23].

Both mean-based metrics and median-based metrics have also been considered here.

The symmetric version of mean absolute percentage error was calculated by using of the following formula:

$$\boldsymbol{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|1 - \frac{\widehat{y_i}(x_i)}{y_i}\right| \quad (9).$$

And symmetric version of median absolute percentage error was calculated as:

$$\boldsymbol{MDAPE} = \boldsymbol{med}\,\left|1 - \frac{\widehat{y_i}(x_i)}{y_i}\right| \quad (10).$$

The smaller the value of the metrics listed above, the closer the model predictions are to the actual targets and thus the better is the model [21].

Another widely used metric is coefficient of determination R^2. This metric shows how much variance is explained by the model [21], [23]:

$$\boldsymbol{R^2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i(x_i))^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (11),$$

where $\bar{y}$ is the arithmetic mean of the targets.

In addition to the metrics described above a 5-10-20 % error intervals were chosen as a performance metrics that are most important for our business. These metrics show the number of objects (in %) whose predicted price was within a 5, 10 or 20 % interval of the actual valuation price (DCF price). The reason why these metrics are most interesting for the business is that the valuation price of an object cannot be considered as the only one unique discrete value. As it was mentioned above, the valuation price depends on several subjective factors such as for example the evaluator's own opinion and experience. Thus, a high-quality and accurate evaluation is always represented by a certain price range. We assume that the valuation price taken from the excel is a compromise of the valuer's and sales manager's opinions and reflects the market well, but that prices in a relatively small range around it also represent a good quality valuation.

4.4 Feature importance

Also, when analysing the model, it is important to understand that in general, the features used for the training have unequal influence on the target variable. Often only a small fraction of the features is most important for prediction. In machine learning, there is a measure called feature importance to assess how significant a feature is for prediction. There are different ways to measure the feature importance also depending on the model [24].

In this project, feature importance has been analysed for Decision Tree, Random Forest and LightGBM models, since the implementations of these models already have built-in methods that return feature importances. For the decision tree and for the random forest model the importance of a feature is computed as the normalized total reduction of the mean squared error brought by that feature after all splits [10]. In the LightGBM model the feature importance was calculated as the total frequency of use of this feature for the split [19].

5. Results and discussion

In this part the main results of this project will be presented and analysed. The quality of the models used was evaluated using the metrics described in Part 4. For the purpose of more clarity, the results of two metrics - MDAPE and the 20-% error interval - will be shown here for different experiments, since these metrics are the most informative in the context of property valuations, as well as the most important for businesses. The results on both train and test data will be shown here in order to assess the degree of overfitting of the models and to check the quality of performance on unseen data. The results of all metrics discussed in Part 4 will be shown on the Table 12 in the Appendix of this report for one experiment only.

To better demonstrate the impact of different preprocessing phases on model performance, five different types of data preprocessing were considered.

In the first experiment (see Table 1), objects with anomalies were not filtered out from the dataset, also the house price for the whole area of the house was used as a target and not the price per square metre, as it will be done in the next experiments. Also in this experiment, objects from all cities were used, regardless of the number of objects in this particular city.

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	20.49	21.18	0.49	0.47
LinReg	9.83	10.73	0.8	0.77
Lasso	9.9	11.08	0.8	0.76
Ridge	10.05	10.28	0.8	0.78
DecTree	11.64	13.07	0.74	0.7
Random Forest	10.64	11.51	0.79	0.75
AdaBoost	13.47	14.21	0.69	0.66
LightGBM	8.65	9.52	0.86	0.82

Table 1. *MDAPE and 20% on both train and test data for the dataset without filtering of objects with anomalies, for the whole house price and with all cities independent on the number of objects. Although the LightGBM ensemble model is the most powerful for these datasets, all linear models show almost comparable performance and outperform other ensemble methods as well as the decision tree model.*

As it can be seen on the Table 1, all models outperformed the baseline model and showed significant increase in predictions quality. In this experiment, regularization in linear models didn't show a noticeable effect. As can be also noticed in almost all models the level of overfitting of models is quite low and all of them perform relatively well on objects not seen before.

In the second experiment (see Table 2), objects with anomalies were again left as they were in the entire dataset without any filtering. Also, all the cities independent on the number of

objects were used, but in this case the sqm-price was used as a target, also such features as ANCR and AMR were divided by the total area and used in the price for sqm.

As can be seen from the results on Table 2, this data transformation improves the quality of almost all models, but for the linear models this improvement is more significant. This may be because the relationship between price and area is less linear and has a rather strong variation (see Figure 6 in Appendix). Thus, using multiplicative features, the relationship between target and attributes (such as ANCR and AMR) becomes less linear or more spread (see Figure 7 and 8 in Appendix). The sqm-features and sqm-target will be used for all future experiments.

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.78	15.18	0.59	0.61
LinReg	8.51	9.53	0.86	0.83
Lasso	8.56	8.75	0.86	0.85
Ridge	8.64	8.79	0.86	0.84
DecTree	10.54	11.78	0.78	0.75
Random Forest	9.69	11.08	0.82	0.78
AdaBoost	11.98	12.44	0.73	0.71
LightGBM	8.33	9.17	0.87	0.84

Table 2. *MDAPE and 20% on both train and test data for the dataset without filtering of objects with anomalies and with all cities independent on the number of objects, but for the sqm-house price. While the ensemble models show only a slight improvement in quality, the linear models have become noticeably more powerful.*

In the third experiment shown on the Table 3 all objects with anomalies were filtered out, but all other modifications were left as they were in the second experiment. As can be seen, even though the number of objects with anomalies in our data was relatively small, after filtering the data, the performance of all models slightly increased. So, it was considered to filter out objects with anomalies in the next experiments.

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.42	15.76	0.61	0.58
LinReg	7.82	8.73	0.9	0.85
Lasso	8.03	8.52	0.89	0.87
Ridge	8.02	8.3	0.89	0.87
DecTree	10.36	11.37	0.81	0.76
Random Forest	9.4	11.2	0.84	0.8
AdaBoost	11.59	11.53	0.76	0.74
LightGBM	7.97	8.72	0.89	0.85

Table 3. *MDAPE and 20% on both train and test data for the dataset. Here the objects with anomalies were filtered out. All cities independent on the number of objects were left in the dataset, ANCR, AMR and DCF-price are represented for sqm. After this filtering, the performance of all models slightly increased.*

In the next experiment shown on the Table 4, unlike the third one, all cities with less than three objects were filtered out. Such filtering did not bring a significant gain in the performance of the models but influenced them even slightly negatively. Thus, this filter was not used in further analyses.

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.7	14.82	0.6	0.63
LinReg	8.09	8.43	0.89	0.87
Lasso	8.14	8.48	0.89	0.87
Ridge	7.95	8.64	0.89	0.86
DecTree	10.24	11.36	0.8	0.75
Random Forest	9.6	10.21	0.84	0.8
AdaBoost	11.65	11.98	0.75	0.74
LightGBM	7.95	8.84	0.89	0.86

Table 4. *MDAPE and 20% on both train and test data for the dataset. Here the objects with anomalies were filtered out. Only the cities with more than three objects were left in the dataset, ANCR, AMR and DCF-price are represented for sqm. This filtering didn't bring a significant improvement.*

As discussed in Part 4, not all features have the same impact on the target variable. Selecting the most important features and discarding the insignificant ones can not only speed up the learning process of models, but also improve their efficiency. Feature importances of six most important features for Decision Tree, Random Forest and LightGBM are shown on the Table 5.

As can be seen from Table 5, feature importances can also be slightly depended on the chosen model. However, in most cases the set of the most important features is approximately the

same. As it was already known from the practical experience in commercial property valuation, the most important price-forming factor is the real annual rent (ANCR). This is well reflected in the feature importances of different models. Knowing the real rent solves the main uncertainty in this problem. Also, the fact that the relationship between ANCR and the price is linear could explain the dominance of linear models over non-linear models.

Features DecTree	Feat_Imp DecTree	Features RandFor	Feat_Imp RandFor	Features LGBM	Feat_Imp LGBM
ANCR	0.814	ANCR	0.644	ANCR	123
Market_sqm	0.151	AMR	0.228	AMR	64
AMR	0.026	Market_sqm	0.111	LM_Timecode	45
Capex	0.005	Vacancy rate	0.010	Market_sqm	37
Vacancy rate	0.003	Capex	0.003	City	28
LivingArea	0.000	LivingArea	0.002	LivingArea	24

Table 5. *Feature importances of six most important features for Decision Tree, Random Forest and LightGBM models. Features importances in LightGBM model are shown as the number of times they were used for the split. And for Decision Tree and Random Forest models the importance of a feature is shown as the normalized total reduction of the MSE brought by that feature after all splits.*

As was seen in the previous experiments, there is one feature (ANCR) that is much more important than all other features when estimating the price of commercial property. This reflects the reality, however, as already mentioned, it is not the only criterion affecting the price of each particular property and although in most cases, based on ANCR alone, it is possible to make a relatively good prediction for the price, the more specific characteristics of the property and location are also extremely important. To test the predictive ability of models without ANCR, this feature was removed in the next experiment (see Table 6) and training was conducted only by using of the remaining features. The purpose of this experiment was also to assess whether these models can be used for a simplified version of property valuation, which is also an important application for the real-estate business. In the evaluation of commercial properties, there are cases when it is necessary to value a property based only on general knowledge about the characteristics of the house, such as area and about the current market situation in the region, what could be also one possible application of this project.

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.56	15.46	0.61	0.61
LinReg	11.9	12.78	0.74	0.7
Lasso	11.76	12.71	0.74	0.71
Ridge	11.61	12.93	0.74	0.68
DecTree	12.8	14.34	0.69	0.64
Random Forest	12.1	13.4	0.72	0.68
AdaBoost	14.72	15.34	0.64	0.61
LightGBM	11.24	12.98	0.75	0.71

Table 6. *MDAPE and 20%-error interval on both train and test data for the dataset. Here the objects with anomalies were filtered out. All cities independent on the number of objects were left in the dataset, AMR and DCF-price are represented for sqm, but ANCR is removed.*

Although the performance of the models has decreased significantly, surprisingly the results are still good for the linear models as well as for LightGBM ensemble model. These models still considerably outperform a simple baseline model. Thus, it can be assumed that these models are also well suited for the task of simplified quick price estimation, since the expectations to this kind of evaluations are also not so strict and the acceptable price ranges and evaluation uncertainties may be much larger. This also further supports the suggestion that other features, beyond the most relevant ones, can also have an important impact on the quality of predictions.

After all these experiments it can be concluded that the most efficient models for this problem are linear models with regularization and the LightGBM ensemble model. The relatively weak performance of the other ensemble models, as well as of the Decision Tree, may be due to the possible presence of some difficult-to-identify outliers in the data (for the Decision Tree, as well as for AdaBoost), as well as to the need to significantly expand the feature space, with the help of one-hot encoding (for Decision Tree, AdaBoost and Random Forest). The best among all models were the linear models with regularization, but in some experiments the simple linear regression outperformed them. The advantage of linear models over ensemble models may be because the price dependence on the most important features is strictly linear and the general dependence of the target variable on attributes can indeed be well approximated by an analytical linear function. So, linear models were examined further in more details.

When analysing the predictive ability of a model, it is also important to look at the trends in model errors and their distribution. This allows to better assess the nature of the data and to identify the possible presence of outliers or other anomalies not detected at the preprocessing stage. For this analysis the predictions of the simple linear regression model were compared with real targets. It was done for those objects where the model predictions were most wrong. As most wrong predictions were chosen those where the difference between the target and the predicted value exceeded 25% one way or the other. This was done to identify possible biases in the trend of errors, towards constant overestimation or vice versa underestimation of the price. Such bias could mean the presence of outliers in the training set towards too

expensive objects, in case the prediction errors tended to overestimate the price and towards too cheap, in case the predictions tended to underestimated prices. This experiment was run several times, each time with the data randomly split into a train and a test. Figure 2 shows one example of comparison between the predicted values and the targets. In this case the model tends to overestimate the price. However, this trend did not persist throughout all experiments. It is likely that both types of outliers are still present in our data. However, there were only about 20 objects on which the model made an error of more than 25% in each experiment, which is only about 5% of the entire test set. Thus, it can be assumed that in general the dataset is good cleaned from outliers. This is also because not all large errors are caused by outliers. One possible reason for the largest errors could be also that these objects have some very specific characteristics (e.g. particularly bad or particularly good house condition), information about which was not available to the model during training, but could be added in the future stages of this project.

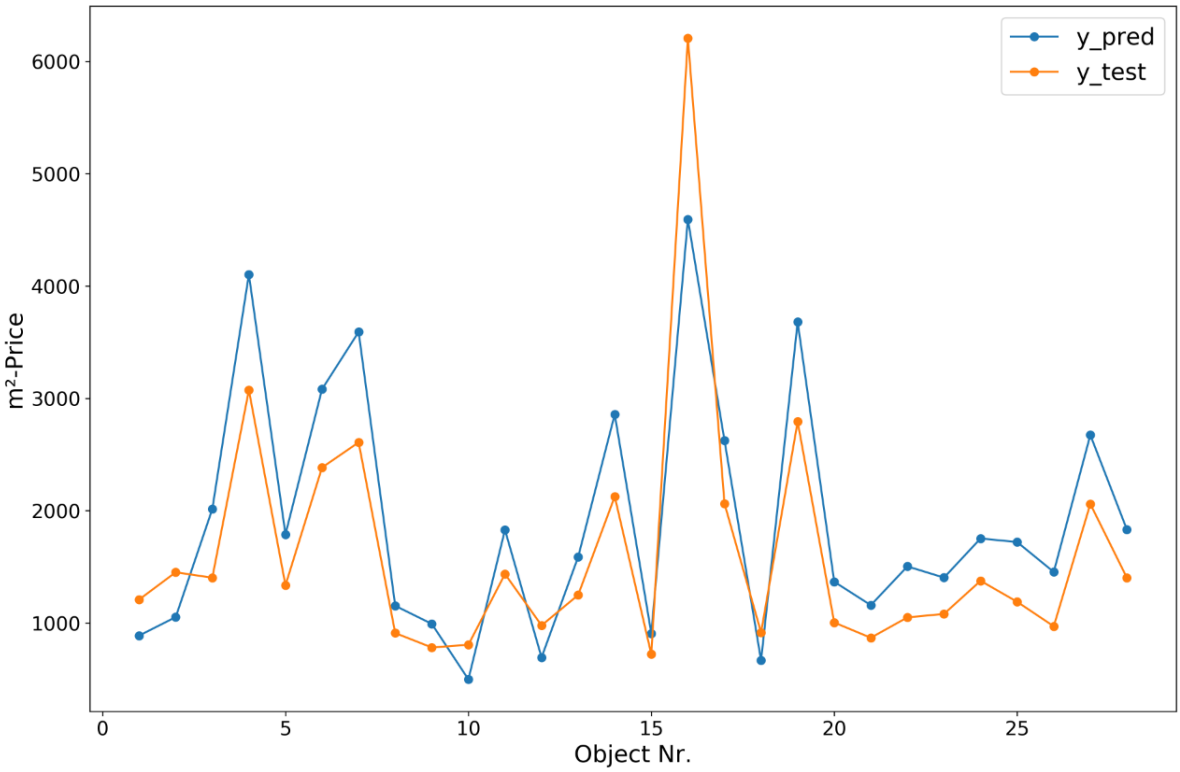

Figure 2. *Comparison of prediction prices of linear regression model to real targets for the objects with the largest prediction error of more than 25%. In most of the cases the model tended to overestimate the price.*

Another way to analyse the quality of the model based on the residuals is the checking of the distribution of residuals. For linear regression, the expected error distribution is a normal distribution with mean around 0 [25]. As can be seen in Figure 3, the linear model errors in this project do follow a normal distribution around 0, except for some outliers.

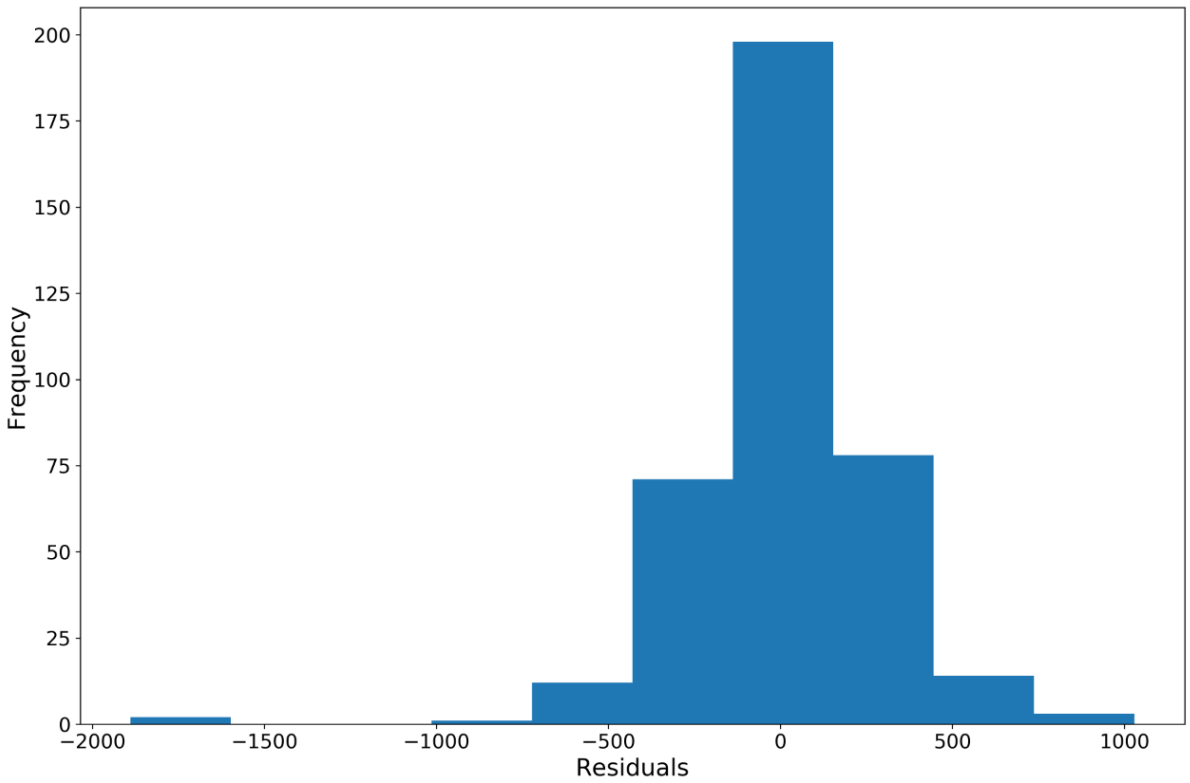

Figure 3. *Residuals distribution for the linear regression model. Except for a few outliers the residuals are as expected for this model normally distributed around 0.*

One alternative way of assessing the quality of linear regression models is analysis of the coefficients. The coefficients of linear regression should directly reflect the actual "physical" dependence of the corresponding attribute on the target [21]. So, for example, we expect to have a positive coefficient for the rent feature, but a negative coefficient for the feature characterises the amount of required investment, sine by holding all other coefficients fixed, we expect the increase of the final house price by increasing of a rent by one unit, but we also expect the decrease in price with increasing investment costs. Table 7 shows some of the coefficients of linear and Lasso regression models. As can be seen from the table, the coefficients for most of the features reflect their physical (natural) meaning very well. Negative coefficients were obtained for required investment and vacancy rate, but positive for rent. It is also well seen how the coefficients reflect the two crisis periods observed on the property market in the last three years. Table 7 also shows the effect of shrinking and nullifying the coefficients in the Lasso regression. This may be the reason for the slightly higher performance of the Lasso model in the most of experiments.

Features	Coefficients LassoReg	Coefficients LinReg
LivingArea	-0.03	-0.03
CommercialArea	-0.37	-0.53
ANCR	19.21	18.75
AMR	6.46	7.78
MaintenanceReserve	-42.69	-39.82
Capex	-21.9	-21.71
Market_sqm	0.27	0.23
Vacancy rate	-9.53	-12.33
2_2023	-35.1	-32.11
1_2023	-45.27	-97.39
4_2022	59.85	43.4
3_2022	71.86	98.26
2_2022	202.33	210.55
1_2022	254.86	247.58
4_2021	183.65	159.24
3_2021	197.14	179.86
2_2021	151.93	151.15
1_2021	125.43	135.09
4_2020	59.6	15.77
3_2020	48.29	-6.54
2_2020	16.57	17.46
1_2020	-0.0	2.01
4_2019	-27.28	-38.6

Table 7. *Comparison of coefficients of Lasso and simple linear regression. The coefficients reflect well the expected "natural" dependencies between the features and target.*

At the end of this project, another final quality test of the models was conducted. Since the models should be used in the future on completely new and unseen data, an additional dataset was extracted from our database. As it was mentioned in the Part 2 only objects from the period till June 2023 were used for training and testing of the models. However, a certain amount of new data has been collected over the period between July and August 2023. These data were used for the last quality test. During this period, 24 new objects were collected, matching all criteria, to the data on which the models were initially trained and tested. The prices for these objects were predicted by using of the best model from all experiments, namely Lasso regression model from the third experiment. Table 8 compares the results of predictions for initial test data and for the new test data from the period between July and August 2023 based on MDAPE and 20%-error-interval metric. As it can be seen, the model shows comparable high quality even on completely new data that has never been used in training and testing process. Table 9 shows several objects from this dataset along with some of their main characteristics, as well as the real price and the predicted price. Having analysed these estimates, we can say that, given the limited data that we could use to train the models,

the predictions reflect the current market situation very well. All the predicted prices can be considered as accurate estimates given the limited set of information that the model had.

Data	MDAPE	20%_Interval
Test Original	8.52	0.87
Test July-August	9.43	0.88

Table 8. *Comparison of the results for predictions for initial test data and for the new test data based on MDAPE and 20%-error-interval metrics.*

City	LivingArea	CommercialArea	ANCR	y_true	y_pred
Frankfurt am Main	494.15	0.0	10.94	3056.0	3379.84
Köln	354.0	0.0	9.32	2260.0	2727.07
Frankfurt am Main	700.0	35.0	8.65	2871.0	2836.52
Duisburg	416.0	0.0	4.86	913.0	746.84
Dresden	252.0	0.0	8.26	1786.0	2080.35
Essen	288.9	0.0	9.73	1765.0	2042.7
Duisburg	393.0	0.0	8.4	1425.0	1560.34
Nürnberg	430.0	0.0	5.87	1700.0	1682.39
Dortmund	335.0	0.0	5.28	1045.0	1016.48
Hannover	572.0	0.0	7.01	1713.0	1739.85
Leipzig	842.2	0.0	7.38	1876.0	1933.56

Table 9. *Several examples from the new test dataset collected in the period between July and August 2023. The table provides basic information about the property's characteristics, location, as well as the information about predicted and real prices. Note that for the purpose of better visualisation and understandability, the ANCR is presented here in sqm and per month, rather than per year as was used during the training.*

6. Conclusions and Outlook

The results of this project have shown that machine learning methods can be successfully applied not only for valuation of small residential properties, but also for large commercial objects. For most properties, the prices predicted by the models were within a range that perfectly reflects the commercial property market in the regions considered. At this stage of the project, with the currently available dataset, the best results were obtained by linear models, which do not require a complex parameter selection process, are less prone to overfitting, and are fast to train. This can be explained by the unequal feature importance for the price estimation. Thus, the most significant for predictions were attributes, the price dependence on which is almost strictly linear and the analytical solution of the linear model in this case outperformed the randomised ensemble algorithms, as well as the decision tree model. Also, one of the possible negative factors that affected the quality of some ensemble models was the impossibility of using categorical features in unchanged form in the sklearn implementation. Even though the algorithm itself allows to work with them directly. The problem of having too many features is called the “curse of dimensionality” and often a disproportionate number of features relative to the total sample size can lead to lower model performance [6]. However, this was not the case for LightGBM model, that outperformed linear models in some experiments and presumably could be more effective overall, with further work on tuning hyperparameters and adding new features. The main limitations that prevented us from achieving higher efficiency especially in predicting prices of some special objects that can be considered as exceptions were relatively small size of the training data, as well as the lack of features that allow for a more specific description of each individual object. So, for example the information about the real condition of the object was not available in this project and could not be qualitatively expressed in numerical form as a separate feature. Whereas this information is available during the manual evaluation because it can be obtained from the object description text. Regardless, as can be seen by analysing the importance of the features, even in the rough form in which it is presented at this stage of the project, information about the condition of the house is quite important and has been considered by many models in this project. Also, information on the location of the property could only be considered at the city level, but based on experience, the actual price can vary greatly depending on the location within the city itself. Adding features on the condition, as well as information on the location within the city and proper encoding of this information, could greatly improve the quality of the predictions even for special cases and can be included at later stages of the project. Also, on the current stage this project needed to be limited for Top-100 German cities only. One idea for expanding the geography of the project to the whole country could be to cluster small towns and villages around certain points on the map. These points would have to be defined in such a way that they would reflect the situation on the property market in the region in the most realistic way. Knowing the addresses of all properties, a measure of distance to these points could be introduced and towns and villages within a given range around these points could be included in these clusters. All these extensions can be further investigated and added in the future stages.

7. References

[1] R.-T. Mora-Garcia, M.-F. Cespedes-Lopez, and V. R. Perez-Sanchez, 'Housing Price Prediction Using Machine Learning Algorithms in COVID-19 Times', *Land*, vol. 11, no. 11, p. 2100, Nov. 2022, doi: 10.3390/land11112100.

[2] Q. Truong, M. Nguyen, H. Dang, and B. Mei, 'Housing Price Prediction via Improved Machine Learning Techniques', *Procedia Computer Science*, vol. 174, pp. 433–442, 2020, doi: 10.1016/j.procs.2020.06.111.

[3] A. B. Adetunji, O. N. Akande, F. A. Ajala, O. Oyewo, Y. F. Akande, and G. Oluwadara, 'House Price Prediction using Random Forest Machine Learning Technique', *Procedia Computer Science*, vol. 199, pp. 806–813, 2022, doi: 10.1016/j.procs.2022.01.100.

[4] W. Coleman, B. Johann, N. Pasternak, J. Vellayan, N. Foutz, and H. Shakeri, 'Using Machine Learning to Evaluate Real Estate Prices Using Location Big Data', 2022, doi: 10.48550/ARXIV.2205.01180.

[5] 'Mehrfamilienhäuser als Kapitalanlage 2023. Der Marktreport für Groß- und Mittelstädte in Deutschland'. [Online]. Available: https://www.engelvoelkers.com/de-de/mehrfamilienhaus/#lokal

[6] D. Bera, R. Pratap, and B. D. Verma, 'Dimensionality Reduction for Categorical Data', *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3658–3671, Apr. 2023, doi: 10.1109/TKDE.2021.3132373.

[7] T. Jiang, J. L. Gradus, and A. J. Rosellini, 'Supervised Machine Learning: A Brief Primer', *Behavior Therapy*, vol. 51, no. 5, pp. 675–687, Sep. 2020, doi: 10.1016/j.beth.2020.05.002.

[8] K. A. Venkatesh, K. Mohanasundaram, and V. Pothyachi, 'Regression tasks for machine learning', in *Statistical Modeling in Machine Learning*, Elsevier, 2023, pp. 133–157. doi: 10.1016/B978-0-323-91776-6.00009-9.

[9] L. E. Melkumova and S. Ya. Shatskikh, 'Comparing Ridge and LASSO estimators for data analysis', *Procedia Engineering*, vol. 201, pp. 746–755, 2017, doi: 10.1016/j.proeng.2017.09.615.

[10] F. Pedregosa, G. Varoquaux, and A. Gramfort, 'Scikit-learn: Machine Learning in Python', vol. 12, pp. 2825–2830, Nov. 2011.

[11] *Handbook Statistical foundations of machine learning*.

[12] A. Amro, M. Al-Akhras, K. E. Hindi, M. Habib, and B. A. Shawar, 'Instance Reduction for Avoiding Overfitting in Decision Trees', *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 438–459, Jan. 2021, doi: 10.1515/jisys-2020-0061.

[13] M. R. Machado, S. Karray, and I. T. De Sousa, 'LightGBM: an Effective Decision Tree Gradient Boosting Method to Predict Customer Loyalty in the Finance Industry', in *2019 14th International Conference on Computer Science & Education (ICCSE)*, Toronto, ON, Canada: IEEE, Aug. 2019, pp. 1111–1116. doi: 10.1109/ICCSE.2019.8845529.

[14] G. Shanmugasundar, M. Vanitha, R. Čep, V. Kumar, K. Kalita, and M. Ramachandran, 'A Comparative Study of Linear, Random Forest and AdaBoost Regressions for Modeling Non-Traditional Machining', *Processes*, vol. 9, no. 11, p. 2015, Nov. 2021, doi: 10.3390/pr9112015.

[15] M. Aria, C. Cuccurullo, and A. Gnasso, 'A comparison among interpretative proposals for Random Forests', *Machine Learning with Applications*, vol. 6, p. 100094, Dec. 2021, doi: 10.1016/j.mlwa.2021.100094.

[16] R. Wang, 'AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review', *Physics Procedia*, vol. 25, pp. 800–807, 2012, doi: 10.1016/j.phpro.2012.03.160.
[17] H. Drucker, 'Improving Regressors using Boosting Techniques', presented at the ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning, Jul. 1997, pp. 107–115.
[18] J. Elith, J. R. Leathwick, and T. Hastie, 'A working guide to boosted regression trees', *Journal of Animal Ecology*, vol. 77, no. 4, pp. 802–813, Jul. 2008, doi: 10.1111/j.1365-2656.2008.01390.x.
[19] 'LightGBM's documentation'. Copyright 2023, Microsoft Corporation. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/index.html
[20] K. Guolin, Q. Meng, and T. Finley, 'LightGBM: a highly efficient gradient boosting decision tree', presented at the NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, Dec. 2017, pp. 3149–3157.
[21] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 103. in Springer Texts in Statistics, vol. 103. New York, NY: Springer New York, 2013. doi: 10.1007/978-1-4614-7138-7.
[22] J. J. Salazar, L. Garland, J. Ochoa, and M. J. Pyrcz, 'Fair train-test split in machine learning: Mitigating spatial autocorrelation for improved prediction accuracy', *Journal of Petroleum Science and Engineering*, vol. 209, p. 109885, Feb. 2022, doi: 10.1016/j.petrol.2021.109885.
[23] M. Steurer, R. J. Hill, and N. Pfeifer, 'Metrics for evaluating the performance of machine learning based automated valuation models', *Journal of Property Research*, vol. 38, no. 2, pp. 99–129, Apr. 2021, doi: 10.1080/09599916.2020.1858937.
[24] M. Saarela and S. Jauhiainen, 'Comparison of feature importance measures as explanations for classification models', *SN Appl. Sci.*, vol. 3, no. 2, p. 272, Feb. 2021, doi: 10.1007/s42452-021-04148-9.
[25] J. Pek, O. Wong, and A. C. M. Wong, 'How to Address Non-normality: A Taxonomy of Approaches, Reviewed, and Illustrated', *Front. Psychol.*, vol. 9, p. 2104, Nov. 2018, doi: 10.3389/fpsyg.2018.02104.

Appendix

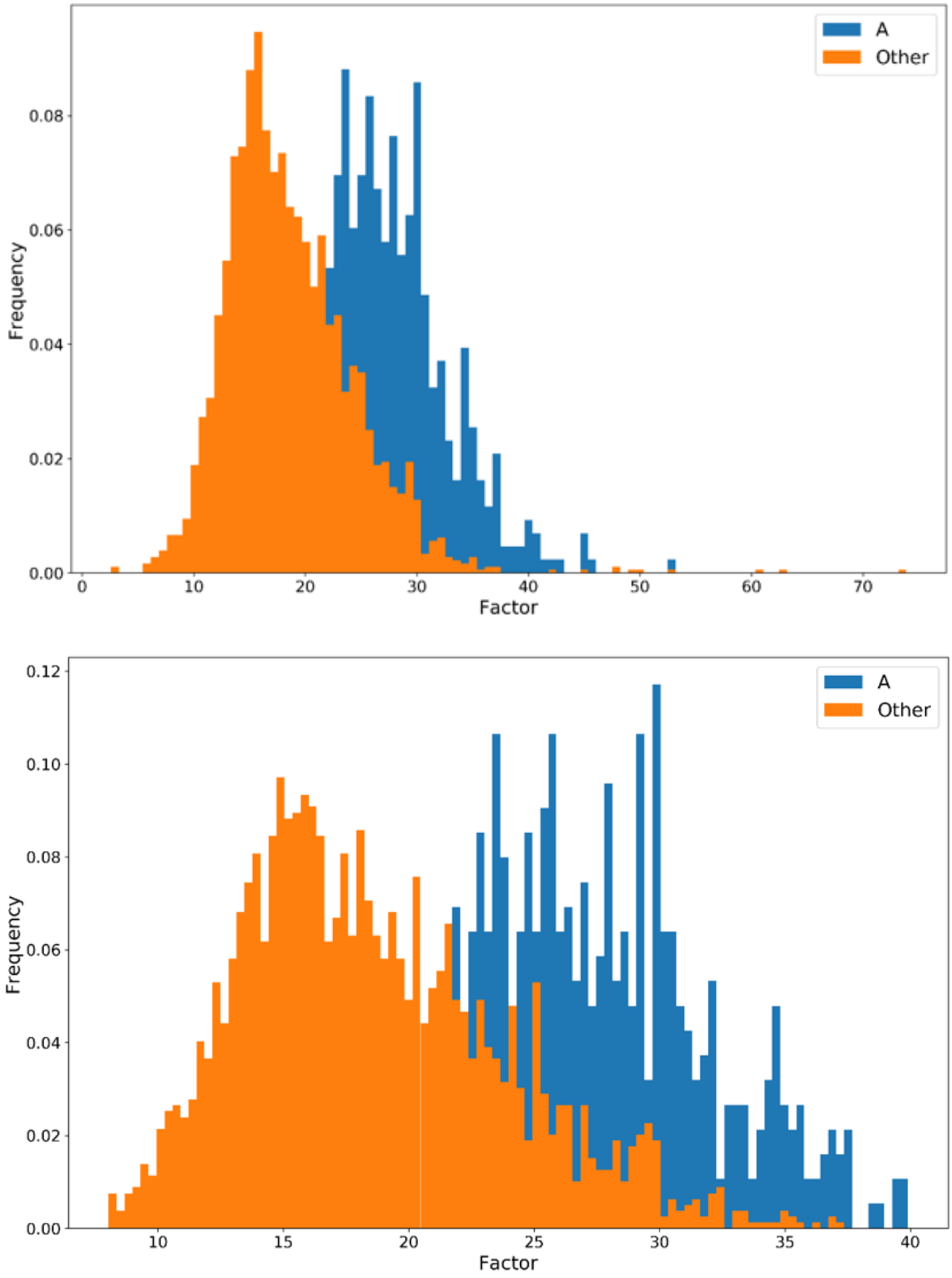

Figure 4. *Factor distribution of all objects in the dataset with outliers (above) and after filtering of outliers (below). The city class is colour-coded by orange for B, C and D class cities and blue for A class cities.*

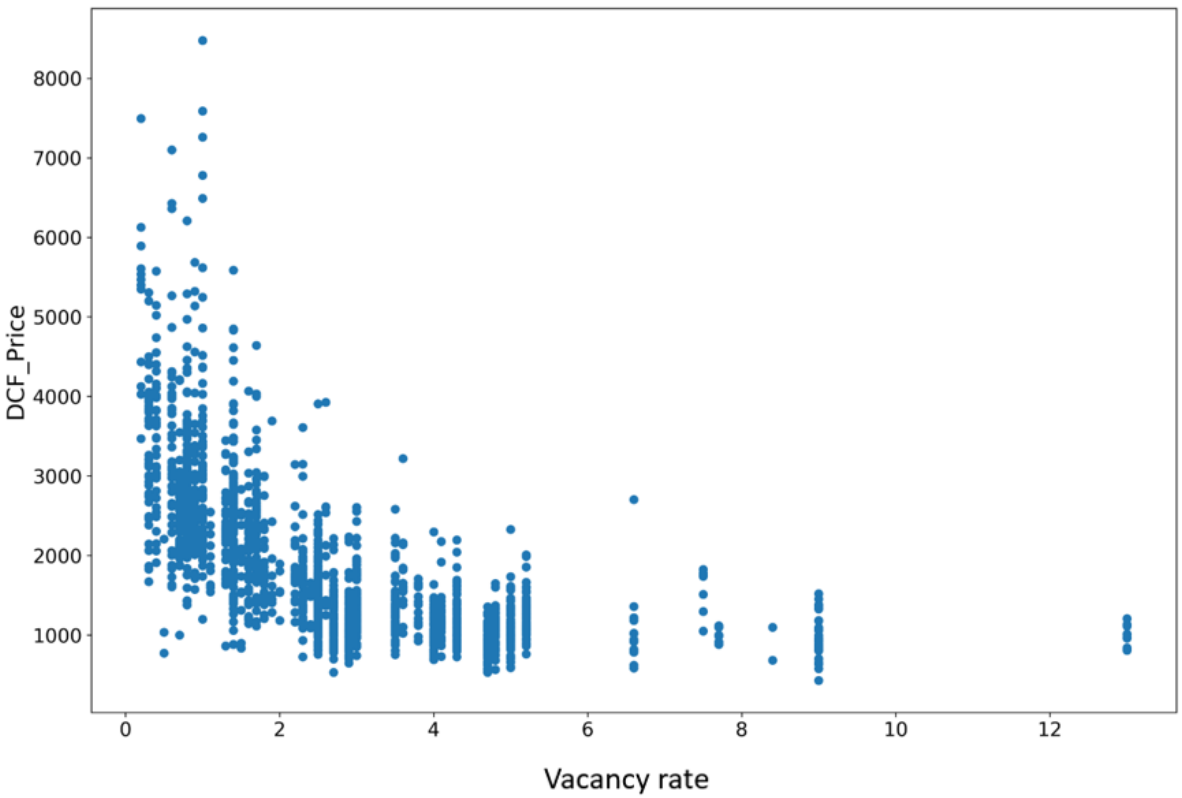

Figure 5. *Dependency between vacancy rate feature (one of the most important features) and target. This relationship is not linear, but rather exponentially decreasing.*

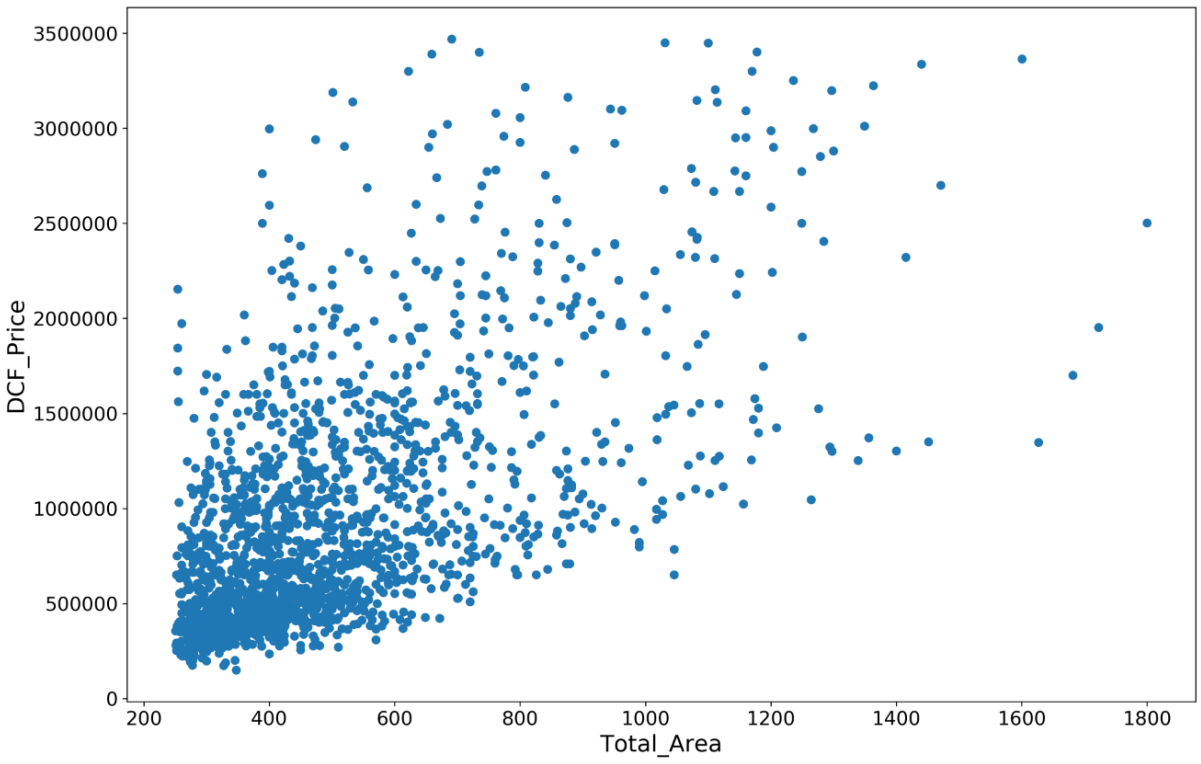

Figure 6. *Dependency between Total area and DCF-Price (represented for the whole area). Although this relationship seems to be linear, the data are extremely spread.*

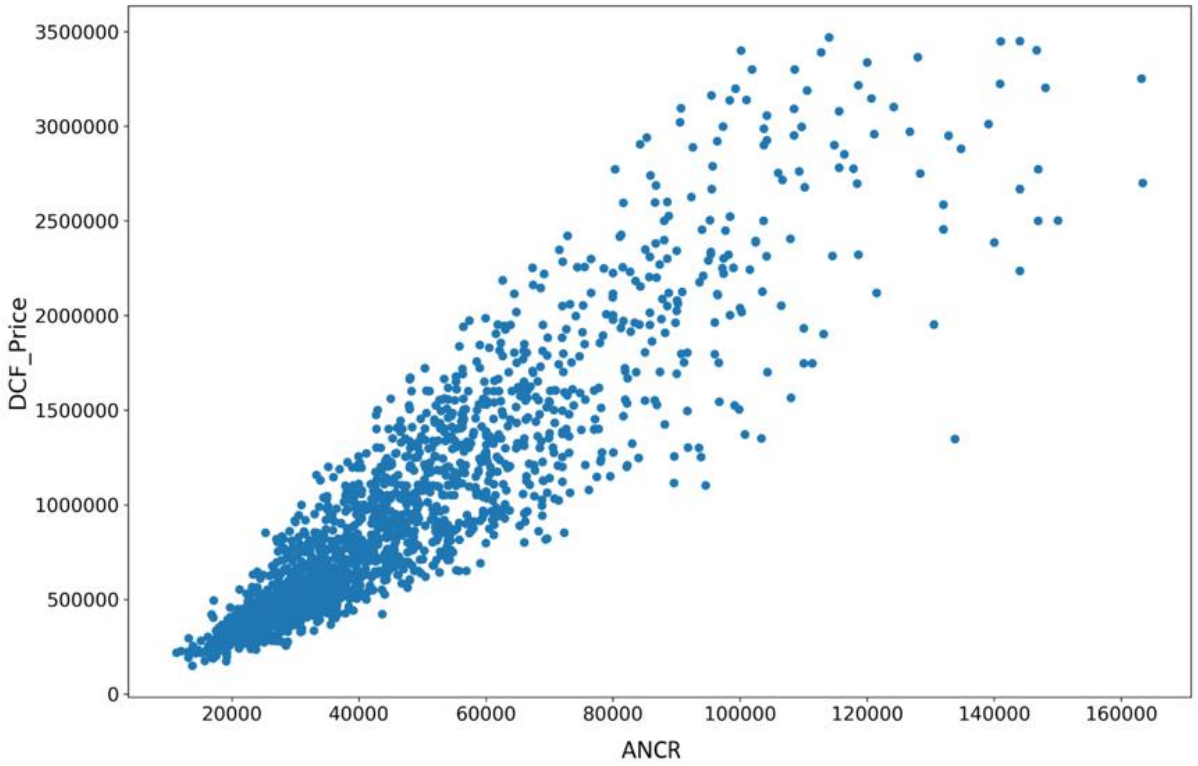

Figure 7. *Dependency between ANCR and DCF-Price (represented for the whole area). Although this relationship is linear, the spread in the data is higher, what leads to the higher error.*

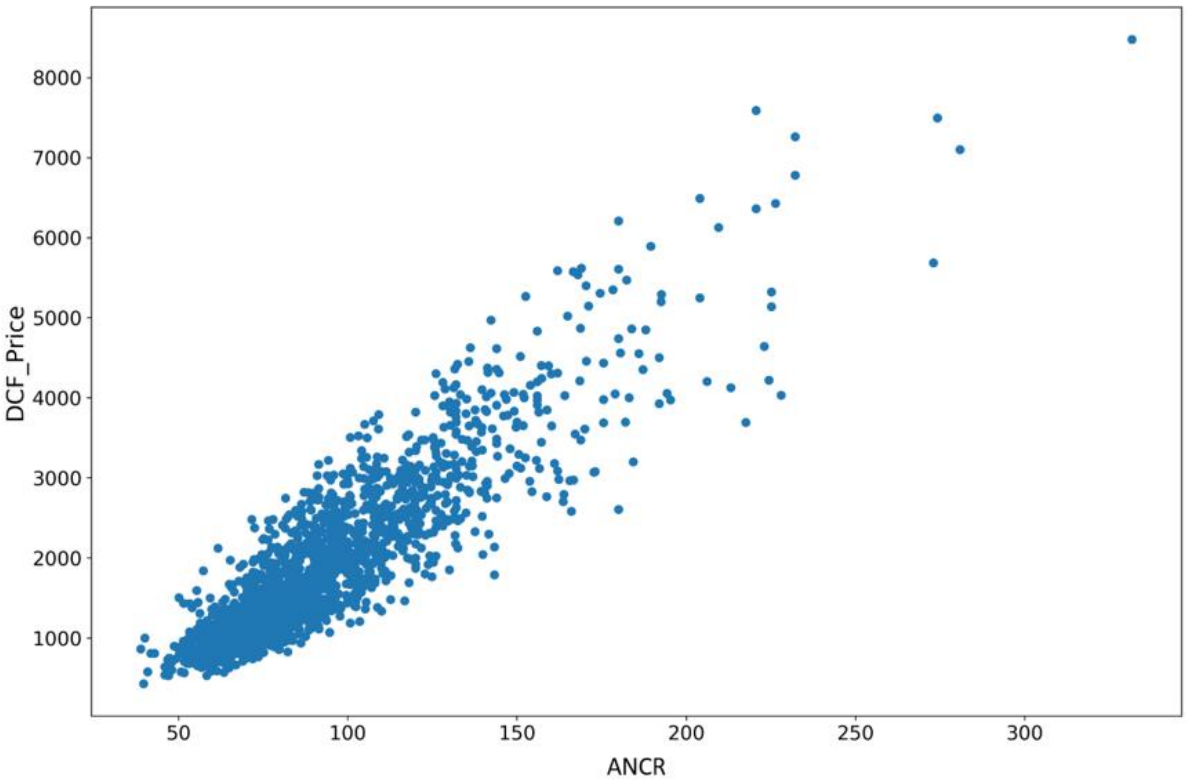

Figure 8. *Dependency between ANCR and DCF-Price (represented for the sqm). This relationship is strictly linear, and the spread in the data is less significant.*

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.54	15.8	0.6	0.62
LinReg	8.95	9.16	0.86	0.85
Lasso	8.95	9.6	0.86	0.84
Ridge	8.91	9.44	0.86	0.84
DecTree	10.07	11.03	0.81	0.78
Random Forest	9.13	9.96	0.85	0.81
AdaBoost	11.67	11.61	0.75	0.73
LightGBM	7.89	9.02	0.89	0.86

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.42	15.76	0.61	0.58
LinReg	7.82	8.73	0.9	0.85
Lasso	8.03	8.52	0.89	0.87
Ridge	8.02	8.3	0.89	0.87
DecTree	10.36	11.37	0.81	0.76
Random Forest	9.4	11.2	0.84	0.8
AdaBoost	11.59	11.53	0.76	0.74
LightGBM	7.97	8.72	0.89	0.85

Table 10. *The performance comparison without (above) and with (below) one-hot encoding of valuation date (in quarters). As it can be seen, this one-hot encoded representation of valuation date leads as expected to slight performance increase for linear models.*

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.6	15.34	0.61	0.61
LinReg	8.05	8.51	0.9	0.86
Lasso	8.07	8.55	0.89	0.86
Ridge	8.08	8.56	0.89	0.85
DecTree	11.21	12.64	0.77	0.72
Random Forest	10.37	11.66	0.81	0.77
AdaBoost	13.03	13.01	0.71	0.7
LightGBM	8.21	9.04	0.88	0.85

Model	MDAPE _Train	MDAPE _Test	20%_Train	20%_Test
Baseline	15.42	15.76	0.61	0.58
LinReg	7.82	8.73	0.9	0.85
Lasso	8.03	8.52	0.89	0.87
Ridge	8.02	8.3	0.89	0.87
DecTree	10.36	11.37	0.81	0.76
Random Forest	9.4	11.2	0.84	0.8
AdaBoost	11.59	11.53	0.76	0.74
LightGBM	7.97	8.72	0.89	0.85

Table 11. *The performance comparison without (above) and with (below) vacancy rate and market sqm-prices feature. Since the dependency between vacancy rate and DCF-price is not linear, there is no significant performance improvement for linear models, but ensemble models and decision tree show better performance by taken into account these features.*

Model	MAE_Train	MAE_Test	MSE_Train	MSE_Test	MDAE_Train	MDAE_Test
Baseline	359.52	365.24	286771.76	317597.01	254.53	266.26
LinReg	176.92	201.13	67112.68	87567.82	120.93	135.34
Lasso	182.32	192.88	70833.23	76829.51	126.82	131.07
Ridge	179.4	192.05	68152.36	81676.63	125.65	127.63
DecTree	231.52	255.9	112266.07	147667.28	165.99	177.72
Random Forest	210.19	245.89	94377.93	127931.43	149.52	182.21
AdaBoost	244.98	270.54	104595.28	153961.7	191.33	201.9
LightGBM	187.65	204.94	85782.84	104131.89	129.05	130.45

Model	MDAPE_Train	MDAPE_Test	MAPE_Train	MAPE_Test	5%_Train	5%_Test
Baseline	15.42	15.76	20.57	21.17	0.16	0.17
LinReg	7.82	8.73	9.68	10.89	0.33	0.29
Lasso	8.03	8.52	9.93	10.6	0.32	0.31
Ridge	8.02	8.3	9.87	10.29	0.33	0.32
DecTree	10.36	11.37	12.93	14.06	0.25	0.24
Random Forest	9.4	11.2	11.73	13.59	0.29	0.23
AdaBoost	11.59	11.53	15.03	15.57	0.23	0.23
LightGBM	7.97	8.72	10.02	11.11	0.33	0.31

Model	10%_Train	10%_Test	20%_Train	20%_Test	R2_Train	R2_Test
Baseline	0.34	0.33	0.61	0.58	0.73	0.7
LinReg	0.61	0.56	0.9	0.85	0.93	0.92
Lasso	0.59	0.56	0.89	0.87	0.93	0.92
Ridge	0.6	0.58	0.89	0.87	0.94	0.92
DecTree	0.49	0.45	0.81	0.76	0.89	0.86
Random Forest	0.53	0.44	0.84	0.8	0.91	0.88
AdaBoost	0.44	0.44	0.76	0.74	0.9	0.86
LightGBM	0.6	0.55	0.89	0.85	0.92	0.89

Table 12. *Comparison of model performances based on all metrics considered in this project. The data were pre-processed following the example described in the third experiment in Part 5.*