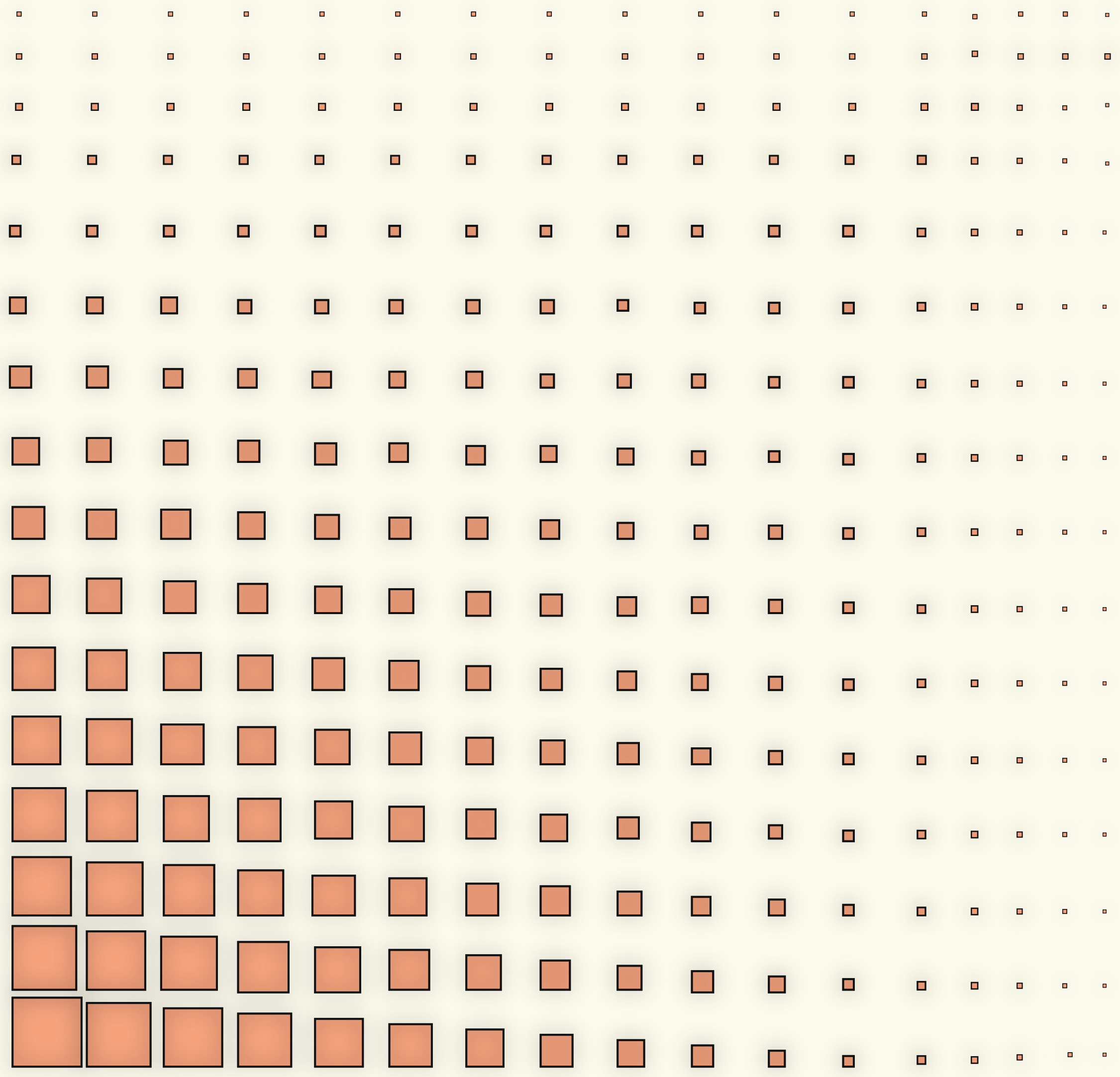
Hyperparameter Optimisation for Real Estate Prediction Models

Tobias Klein

Bachelor's Thesis
April 2019



A L B E R T - L U D W I G S - U N I V E R S I T Ä T   F R E I B U R G   I M   B R E I S G A U

Fachprüfungsausschuss Volkswirtschaftslehre (B. Sc.)

Data Mining:
Hyperparameter Optimisation for Real Estate Prediction Models

Bachelor's Thesis

Advisor
Prof. Dr. Dirk Neumann

Author
Tobias Klein
3804799
Mainz

Timeframe
Feb. 6 2019 – May 6 2019

To my mother Astrid Klein, who is sadly missed.

Contents

Abstract

English Version

Combining a highly scalable and customisable process, with very accurate prediction results using machine learning models, is what this work proposes. The customisation is guided by what information the user seeks to gain from the process. This makes the process applicable for a variety of sectors, such as Banking & Finance, Marketing and urban development among others. It evaluates the process of using self-acquired data from an online real estate platform, gained from deploying a custom web scraping algorithm. This data is then combined with several spatial features for predicting the base rent for apartments on a validation dataset. The analysis and predictions are made for rental apartment listings within the Hanseatic City of Hamburg. The spatial features originate from sources other than that of the apartments data and have to be adapted to it first, therefore. Predictions are made using state of the art machine learning models, in the form of a Lasso Regression model and a XGBoost Regressor model. The Hyperparameter Optimisation techniques grid search and random search are compared, during the optimisation process. The focus is on maximising prediction accuracy of the models. The best scores, expressed in RMSE, are 190.68 for the Lasso and 115.39 for the XGBoost Regressor. Differences in complexity and interpretability between the models are discussed and associated with it, the strengths and weaknesses of the respective model are pointed out.

German Version

Die Kombination eines hoch skalierbaren und anpassbaren Prozesses mit sehr präzisen Vorhersageergebnissen unter Verwendung von Machine Learning Modellen ist der vorgeschlagene Ansatz dieser Arbeit. Die Anpassung richtet sich danach, welche Informationen der Benutzer aus dem Prozess gewinnen möchte. Dadurch ist der Prozess für eine Vielzahl von Sektoren anwendbar, wie zum Beipiel Bankwesen & Finanzen, Marketing und Stadtentwicklung. Es bewertet den Prozess der Verwendung von selbst gewonnenen Daten, die durch den Einsatz eines eigenen Web-Scraping-Algorithmus von einer Online-Immobilienplattform gewonnen wurden. Diese Daten werden dann mit mehreren räumlichen Merkmalen kombiniert, um die Kaltmiete für Wohnungen auf einem Validierungsdatensatz vorherzusagen. Die Analysen und Prognosen werden für Mietwohnungsangebote in der Hansestadt Hamburg erstellt. Die räumlichen Merkmale stammen aus anderen Quellen als denen der Wohnungsdaten und müssen daher zunächst an diese angepasst werden. Die Vorhersagen werden mit Hilfe modernster Machine Learning Modelle in Form eines Lasso-Regressionsmodells und eines XGBoost Regressor-Modells getroffen. Die Hyperparameter Optimierungstechniken Grid Search und Random Search werden während des Optimierungsprozesses verglichen. Der Fokus liegt auf der Maximierung der Vorhersagegenauigkeit der Modelle. Die besten Ergebnisse, ausgedrückt in RMSE, sind $190,68$ für das Lasso und $115,39$ für den XGBoost Regressor. Unterschiede in der Komplexität und Interpretierbarkeit zwischen den Modellen werden diskutiert und damit verbunden, die Stärken und Schwächen des jeweiligen Modells aufgezeigt.

1 | Introduction

Motivation Why is it of any importance, if using open source data enables economic actors to make predictions regarding the base rent (Kaltmiete), in a similar way to how is done here? Are there not enough official reports and statistics regarding this matter already? These questions are legitimate and so they should be answered.
Given that this work was entirely done using the open source language `Python` [89], it can be said that the open source nature gives a high degree of independence regarding which tools are used and how they are used. Further the diversity and the number of tools available for a programming language like `Python`, is hard to match for a single commercial software solution. This kind of open source approach makes it possible to analyse city dynamics, such as the prediction of base rent, rent increases over time, socioeconomic dynamics, the proximity to amenities such as hospitals or public transport, for different districts of the city. These results can then be compared to one another or they can be tested for their predictive importance regarding the prediction of the dependent variable of interest, using machine learning techniques. The importance of it then is, that it can enable an economic actor to conduct their own analysis, customised to give the information they are looking for. This nevertheless depends on the information that is available. This in particular used to be a limiting factor for such independent research but has changed with the emergence of open source data specifically [7]. Considering the question whether there is not enough official information available already, the answer is that it depends on the country the research area is in [64] and the specific research question. In their paper, (Rondinelli and Veronese 2011) write:

> Empirical analysis on rent dynamics is still rather scant for the Euro area, mainly because of a lack of data. [64]

They note that Germany is the exception though, with a sufficient amount of data available for research [64], as demonstrated by (Hoffmann and Kurz 2002) [35]. Since 2011 this might have changed, nevertheless this issue is described in the literature and especially for China there are reports that there is a lack of data for city dynamics and land use analysis [21, 37, 91]. Therefore, one finds a high concentration of these open source techniques together with the acquisition of data by means of web scraping or by using image recognition algorithms to analyse satellite images [21, 37, 91]. For an organisation or company, for example in the Banking & Finance sector, looking for information regarding a non standard metric not found in official reports, the need for these independent and open source approaches, regardless of the quality and quantity of official statistics, is a given. From the newspaper article in the *International Business Times UK*, (Deep learning and big data 2017):

> In the world of finance, the new data paradigm entails applying predictive analytics to new datasets that are collected from non-traditional financial data sources to discover novel and consistently predictive features, and potentially useful patterns about the entity in question beyond what is easily available from traditional financial data sources. [24]

Many of the points made in this quote, are part of the here proposed process and are described in the following chapters. In this work it is demonstrated how official statistics in the form of geospatial attributes can be joined with the existing dataset of the project. This statement from (Barham 2017), originally made regarding characteristics of *big data*, can be applied here as well:

> *Collecting the right kind of data*: This task is really tricky, getting and trying to aggregate all the available data means time and resources costs, however, capturing minimal amount of data, could mean losing hidden values that were not known before it was captured (Tole 2013) (Naimi and Westreich 2014) (Detwiler 2015). [10]

The conclusion then is that, as long as there are features not included in any official report that are possibly relevant, that might help a company gain a competitive edge, a process similar to the one explored in this work is of relevance.

1.1 Scope & Research Questions

The main goal of this work is to predict the value of base rent (dependent variable) on unseen data using a set of predictive variables (regressors). The scope of this work is the urban area of the Hanseatic City of Hamburg (Hamburg). In particular, it concerns rental apartments and not commercial properties or short-term rentals, as is the case with *Airbnb*, for example.

The first part of the research question is how these regressors can be constructed from the following independent data sources:

- Geospatial features (based on the location of the listing):
  - Exposure to street noise [46]
  - Distance to the next subway station (U-Bahn Station) [48]
  - Distance to the next suburban train station (S-Bahn Station) [47]
  - Status class of the statistical area the listing is in [78]
  - Socio-spatial trend for the statistical area the listing is in [78]
- Core variables from listings on the real estate portal Immobilienscout24.de [39]

The second part is concerned with the question, if using Hyperparameter Optimisation on the models `XGBoost` and `Lasso` improves prediction results compared to using their default parameters.

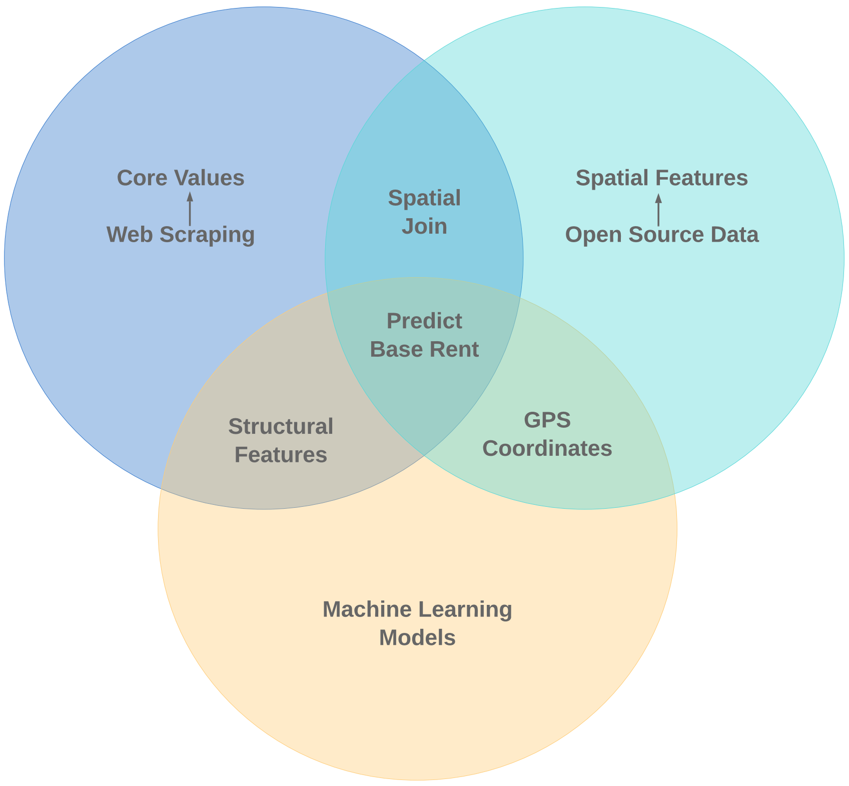


Figure 1.1. Figure showing the data sources web scraping and open source data together. These inputs are used for training the machine learning models and all parts together make it possible to predict the variable base rent on unseen data. The overlapping areas represent the links between the different parts.

The overlapping parts in Figure 1.1 describe the links between the three parts. In detail going clockwise from *Spatial Join*: A *Spatial Join* was used to connect the environmental data with the core variables of the listings from the *web scraping*. All spatial features are given as inputs to the machine learning models. The one associated closest with each listing is its GPS coordinates [31]. The models get the structural variables for each listing in the form of the core variables. The inputs from the clean and completely spatially joined data sources make it possible to train and ultimately predict the value of base rent on unseen data for apartments within Hamburg.

1.2 Outline

An outline of Chapters 2 to 8 is given. It gives a short summary of the contents of each Chapter.

Chapter 2 gives an overview of the existing literature regarding the prediction of real estate prices in Section 2.1. Literature with a focus on describing city dynamics and structure using similar methods is also included. It is specifically mentioned how data from the Internet and especially open source data have already been used in the existing literature. The focus is on literature that includes both structural and environmental variables as inputs to the model. In this context, the *Hedonic Price Model* is described and applied to this situation in Section 2.2. An introduction to *Hyperparameter Optimisation* with references in the literature is also part of Section 2.2.

Chapter 3 specifies the here found prediction problem in Section 3.1 and the model selection in Section 3.2 together with a short description of each model. It continues with the collection of the data in Section 3.3, describing the web scraping algorithm and the core variables obtained by it. Section 3.4 describes the cleaning steps used to transform the data into a *clean structure* for further analysis. In Section 3.5, the external variables are described together with the process of integrating them into the existing dataset. Section 3.6 is the last section of the chapter. It gives an overview together with a description of all the final variables that are used in the following chapters. This table serves as a reference as well, giving the reader information about what each variable describes if needed at any point.

Chapter 4 is about the visualisation of the variables in the dataset. It begins with an explanation of the univariate distribution for each variable in the dataset in Section 4.1. The correlations found between the variables is discussed in Section 4.2. In the same section, the noise data is analysed for both day and night and its impact on base rent specifically. What follows in Section 4.3, is a detailed heat map of the distribution of variable base rent by statistical area, plotted onto a map of Hamburg. The last section is Section 4.4. The general preprocessing done for machine learning is explained and it discusses standardisation of the data, for the machine learning part that follows in Chapter 5.

Chapter 5 documents the process of fitting, optimising and interpreting the models. First, the evaluation metric used here for measuring the accuracy of the models, is explained and important declarations are made. The following Section 5.1 deals with the Hyperparameter Optimisation. The `Lasso` model is first optimised, then the `XGBoost` model. An overview of the RMSE values by model, method and set is found in Section 5.2. The coefficients of the `Lasso` are interpreted in detail in Section 5.3 and the feature importance plot of the final `XGBoost` model is described in Section 5.4.

Chapter 6 reviews the results from Chapter 5 and compares the results of both models in Section 6.1. The results of the Hyperparameter Optimisation are summarised in Section 6.2. Finally, the limitations are discussed, and possible future work is presented in Section 6.3.

Chapter 7 discusses to what extent the results obtained answer the research questions posed in Chapter 1 and gives a conclusion in this respect. As part of this, the integrated geospatial features are checked for their relevance with regard to the prediction of the base rent in Section 7.1. Section 7.2 gives the answer to the question posed in Chapter 4, whether standardising the data can aid better prediction results. It then presents, the final accuracy scores for the base rent predictions of both models. Subsequently, the Hyperparameter Optimisation procedure for both models is discussed in review and conclusions are drawn.

Chapter 8 is the final chapter of this work. It reviews the methods and the thinking process used during the here proposed process, in Section 8.1. Section 8.2 concludes this work with the identification of economic sectors and fields in which the process presented here can be applied and for which it is therefore relevant. Exemplarily, possible practical applications are also mentioned.

2 | Related Work & Foundation

This chapter gives an overview of the existing literature regarding the prediction of real estate prices in Section 2.1. Literature with a focus on describing city dynamics and structure using similar methods is also included. It is specifically mentioned how data from the Internet and especially open source data have already been used in the existing literature. The focus is on literature that includes both structural and environmental variables as inputs to the model. In this context, the Hedonic Price Model is described and applied to this situation in Section 2.2. An introduction to Hyperparameter Optimisation with references in the literature is also part of Section 2.2.

2.1 Related Work

Geospatial Analysis

There are many different factors involved in determining the value, of a property. For clarification, value here is equivalent to price, it can be the price for a house or apartment or the monthly base rent. This value is the *equilibrium value*, as written in the paper (Sirmans et al. 1991):

> The standard approach to modeling housing markets is to employ the familiar hedonic pricing framework exposited by Rosen (1974) **to estimate the equilibrium value of houses as functions of their specific objective characteristics** without drawing out explicit connections to real estate broker service markets. [72, 65]

The following takes up the *specific objective characteristics* found in the quotation and explains them in more detail.
They can be grouped by their associated level [91]. The first level being the household level or the set of properties of an apartment. These are the main attributes of the apartment such as the living space, number of rooms, whether it has a fitted kitchen among others. (Sirmans et al. 1991) [72] call these features *physical characteristics*, in their case for houses, and they use the variables: "Living area", "Other square footage",

"Number of bedrooms", "Number of baths", "Age of house". These variables are included here as well, except for "Other square footage". It was not applicable in this case. There are studies that take into account the socioeconomic sphere with variables such as, the economic sectors the tenants work in, the household income. This data can come from survey data, as is the case for (Wu et al. 2013) [91]. These variables were not available for this work, so they are not included. The second level and above contain variables from the zone that the apartment is in. There is an abundance of variables that possibly affect the value of the dependent variable at the zone level. Such as, what facilities are close to the apartment and the effect the distance between apartment and facility has on the dependent variable. With that comes the risk of an omitted variable bias. This can cause a very strong bias in a Hedonic Price Model [57]. It can make it impossible to estimate the effect that a regressor has on the dependent variable [4]. Common variables from the zone level and above include proximity of the property to the next minor or major access road [4, 55, 91], proximity to a school [55, 91], distance to the city center [4, 55, 64, 91]. Many of these variables are only available through the widespread use of online platforms for rental type and purchasing or selling transactions [21, 36, 62]. A technique used in the papers (Chen et al. 2016) and (Hu et al. 2016) for areas in China, is using *Landsat Images* (satellite imagery of the ground) and open source map data to analyse rent and city dynamics [21, 37].

2.2 Fundamentals

Rent Prices and House Prices are different values and qualities of real estate objects being studied in the literature and often times the Hedonic Price Model is used in studies that either focus on the value of a property given by its sale price or estimated sale price that is derived from the inherent qualities of the property and its features from the zone level. Less often the model is used to explain rental dynamics. This difference is mentioned, because in this work only the effect on the rent of an apartment is studied. The theory of the Hedonic Price Model is applicable in this case as well, since there is a solid and persistent relation between the price and the rent of a property [44].

The Hedonic Price Model

> The Hedonic Price Model is a model of product differentiation based on the hedonic hypothesis that goods are valued for their utility-bearing attributes or characteristics. [65]

Specifically, it describes a relation between the value of the base rent and the aforementioned main attributes that are of structural nature and closely linked to the physical attributes and amenities the apartment offers together with the features provided by the zone around the apartment.

Spatial Autocorrelation in this context is referred to as the "coincidence of value similarity with locational similarity" [5]. In this case it means that apartments that are close to one another tend to have similar values for the base rent per square meter variable. This variable normalises the value of the base rent, so that the base rent of different apartments can be compared regardless of their living space values. The living space tends to positively correlate with the base rent value, as will be discussed later in Chapter 4.

Spatial Heterogeneity describes the phenomenon, that apartment rent prices are dependent on the location the apartment is in. These differences can have many causes and it is related to such factors as the distance to the city centre from the apartment, the neighbourhood the apartment is in. Other factors can be, the exposure to noise or its proximity to the next subway or suburban train station. Spatial heterogeneity is found when houses with similar features regarding, for example lot size, number of rooms, interior quality, number of bedrooms have different sale prices in different parts of the city [8, 28, 56].

Hyperparameter Optimisation

To lay the foundation of what Hyperparameter Optimisation is, as well as to give an overview of commonly used techniques, an introduction is given. A definition of what hyperparameters are and what some of their characteristics are, is referenced from (Bergstra et al. 2015):

> Most implementations of machine learning algorithms have a set of configuration variables that the user can set which have various effects on how the training is done. Often there is no configuration that is optimal for all problem domains, so the best configuration will depend on the particular application. These configuration variables are called hyperparameters. [12]

The optimisation of these parameters is of particular interest, as "often there is no configuration that is optimal for all problem domains", as stated in the quotation. It is about tuning these parameters, so they are optimal for the specific problem. Setting the values of these hyperparameters used to be mainly a manual task, but has become a domain for algorithm based solutions over the last years, according to (Bardenet et al. 2013):

> Hyperparameter learning has traditionally been a manual task because of the limited number of trials. Todays computing infrastructures allow bigger evaluation budgets, thus opening the way for algorithmic approaches. [9]

Popular Methods for Hyperparameter Optimisation A grid search is a method where the user passes a grid with an exhaustive set of values to be tested to the algorithm as input. A set of values is passed on as input for the grid search algorithm, for each parameter to be optimised during the grid search. Here, the grid search was implemented for optimising the hyperparameters of the models, in an effort to better their prediction results in Chapter 5. The underlying problem with the grid search is, that each hyperparameter can have a large value range for its possible values. An example is a parameter with a continuous value range between 0 and 1. This range containing all *machine numbers* between 0 and 1 could not be tested in a grid search, as there are too many values in this range. Oftentimes only a small subset of all hyperparameters in a model and only a small subset of the respective value ranges for each parameter are of relevance for the value of the evaluation metric [11, 13]. However, the number of models can still become extremely high. Consider 15 hyperparameters and for illustration purposes assume each one has 40 possible values. If a 5 fold cross validation is used to evaluate the models, the total number of models to build is given by $15 \cdot 40 \cdot 5 = 3000$. To put it into perspective, with an estimated time of 0.68 seconds that it takes to build one model on the here used machine, to build all 3000 models would take $3000 \cdot 0.68 = 34$ minutes. While more computational power in the form of better hardware is a solution up to a certain point, using one of the following methods can be comparably more efficient [11, 13, 59]. Considering these weaknesses of the grid search procedure, there are alternatives available.
Among the main alternatives used for Hyperparameter Optimisation, are the following algorithms. The random search [11], which is used here and explained in detail in Section 5.1. Other more complex approaches include Bayesian Optimisation [38, 59, 77] and iterated F-racing [53, 14].

3 | Methods & Data

A brief specification of the problem is given in Section 3.1 and the underlying mechanics of the models used here are briefly described in Section 3.2. It continues with the collection of the data in Section 3.3, describing the web scraping algorithm and the core variables obtained by it. Section 3.4 describes the cleaning steps used to transform the data into a *clean structure* for further analysis. In Section 3.5, the external variables are described together with the process of integrating them into the existing dataset. Section 3.6 is the last section of the chapter. It gives an overview together with a description of all the final variables that are used in the following chapters. This table serves as a reference as well, giving the reader information about what each variable describes if needed at any point.

3.1 Specification of the Problem

The aim is to predict the value of the base rent variable and to try to identify important variables for the value of base rent. This requires relevant independent variables which deliver consistent, meaningful and standardised values. These should then be used to train the model to predict the dependent variable in order to finally estimate the base rent for apartments, unknown to the model.
It is the estimation of a dependent variable that has continuous values that can be ordered and compared with one another. A single numerical value is what is being estimated by the model. In order to make a statement about the effect that individual independent variables have on the dependent variable, oftentimes several assumptions regarding data and residuals are made a priori. Whether these assumptions are fulfilled can only be assessed once the data is *clean* and only to a limited extent in general.

3.2 Model Selection

It should be noted that both models are *supervised learning* models, meaning that they are "taught by example". This entails that the models get as inputs all the variables of the final dataset, except for the base rent variable. This variable is to be predicted and the method used for *fitting* the data is a stochastic gradient descent in the case of the `XGBoost` model [32, 45] and a coordinate descent [73] for the `Lasso`.

`Lasso` The first model used in this work is the Lasso Regression Model. Lasso is an acronym for: *Least Absolute Selection and Shrinkage Operator* [84]. It was first introduced in the paper (Tibshirani 1996) [83]. It is trained with ℓ_1 as regulariser in its default configuration with $\alpha = 1$. What this means is that the sum of the absolute values of the coefficients (w) is added to the *ordinary least squares* term [33, 85, 84]. The complete term is to be minimised regarding w by the model during coordinate descent [73]. The objective function to minimise is:

$$\min_{w} \frac{1}{2n} ||Xw - y||_2^2 + \alpha ||w||_1$$

$$\text{Euclidean Norm}: \ \|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} \quad , \quad \ell_1\text{-Norm}: \ \|x\|_1 := \sum_{i=1}^{n} |x_i|$$

With n the number of observations, X the vector of the independent variables with coefficients vector w and y the dependent variable. The model solves the minimisation problem regarding w imposed by the added penalty term $\alpha ||w||_1 \geq 0$ [1]. Where $\alpha \geq 0$ is a constant and $||w||_1$ is the ℓ_1-Norm of w [32]. The value of α can be altered and $\alpha = 1$ gives the ℓ_1-Norm of w and thus tends to *shrink* many coefficients. The applied *shrinkage* gives coefficients with values that are either equal to zero or close to zero (sparse coefficients) [34, 86, 84, 85]. The lower bound is $\alpha = 0$, which eliminates the penalty term completely and gives an ordinary least squares linear regression [73]. Hence the usage of the model not only for predictions, but also as a feature selector with $\alpha > 0$. A special case where ordinary least squares linear regression fails and `Lasso` is used instead, is the following. Consider X as a matrix $X \in \mathbb{R}^{n \times p}$ with the number of rows given by n (number of observations) and the number of columns given by p (number of regressors). `Lasso` is frequently used when there are far more regressors than there are observations, that is if $p \gg n$ [33, 63, 71]. As with a general regression model, the assumption is that the observations are either independent or that the y_is are conditionally independent given the x_{ij}s [84].

XGBoost The second model used here is the GXBoost model. XGBoost stands for *eXtreme Gradient Boosting* and it is an implementation of gradient boosting machines [19]. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community (DMLC) [25]. A short explanation of how gradient boosting works is given based on the book of (Kuhn and Johnson 2013) [45]. The three main elements are:

1. A loss function that must be optimised (*here: mean squared error*)
2. A weak learner that makes predictions (*here: regression trees*)
3. An additive model, such that weak learners can be added to minimise the loss function (*boosting*)

The loss function depends on the type of problem at hand and since a regression type of problem is given here, a mean squared error function is used inside the model. In gradient boosting decision trees are used as weak learners in general, and regression trees more specifically if a regression problem is given. These give a floating-point number as output for each leaf and the output of all leaves is summed up to give the final score of one tree. As illustrated in Figure 3.1, the output values of the individual trees are summed up at the end to give the final value. The construction of the trees happens in a greedy manner. A greedy algorithm is characterised by only considering which next step has the highest instant reward, for the evaluation of which step to take next [49].
Constraints are used to keep the trees being weak learners. A characteristic of the model is that trees are added to the model, one at a time and existing weak learners are not altered. A stochastic gradient descent is used to minimise the loss, so when adding a new tree, it must reduce the loss, follow the gradient. The output for the new tree is then added to the output of the existing trees in an effort to improve the overall performance of the model. There are two ways this can come to an end. 1. The fixed number of trees is reached. 2. The addition of a new tree does not better the result of the model on a validation dataset or the loss function reaches an acceptable level.
These are the main elements of the ensemble method gradient boosting. One difference of XGBoost compared to the classical gradient boosting algorithm is its ability to use the available computational resources very well, and specifically its ability to parallelise parts of the training process during fitting of the data [19]. One consequence of this is that the trees are not added individually after each iteration as described in the boosting process, but sequentially as part of the parallelisation of the process [19].
A formal, more mathematical definition of the main elements of the XGBoost algorithm follows. The following is a mere reproduction with an adaptation of the labels

of the occurring variables, in order to establish uniformity with the previously declared variables, during the description of the Lasso model. It is about the description of the XGBoost algorithm as can be found in the paper (Chen and Guestrin 2016) [19], Section "2.1 Regularized Learning Objective":

> For a given dataset with n observations and p features $\mathcal{D} = \{(x_i, y_i)\}$, $(|\mathcal{D}| = n, x_i \in \mathbb{R}^p, y_i \in \mathbb{R})$, a tree ensemble model uses K additive functions to predict the output.
>
> $$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F},$$
>
> with $\mathcal{F} = \{f(x) = w_{q(x)}\}$, $(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ the space of regression trees. Here q gives the structure of each tree that maps an observation to the corresponding leaf index. T denotes the number of leaves in the tree. Each f_k corresponds to an independent tree structure q and leaf weights w. Unlike decision trees, each regression tree contains a continuous score on each of the leaves. It is given by w_i for the i-th leaf. What this means is that for a given observation the decision rules, given by q, of the tree are used and their respective outputs then are projected to the leaves. These scores (w_i) for each leaf are summed up and the final score (w) is the result. To learn the set of functions used in the model, the following *regularized* objective is minimised.
>
> $$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$
> $$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$
>
> Here l is a differentiable convex loss function that measures the difference between the prediction $\hat{y}_i$ and the target y_i. The second term Ω penalises the complexity of the model. The additional regularisation term helps to smooth the final learnt weights to avoid over-fitting.

One point to mention is that the number of parameters, as counted by the author on the official website [92], that the XGBoost model has, is 22. While not all might be relevant and some might have only one possible value for a given problem, the settings of these parameters can be of great importance concerning the accuracy of the predictions [22, 29, 30]. The fact that there are correlations between some of them regarding their settings, should be acknowledged when doing Hyperparameter Optimisation [9, 13, 20, 32].

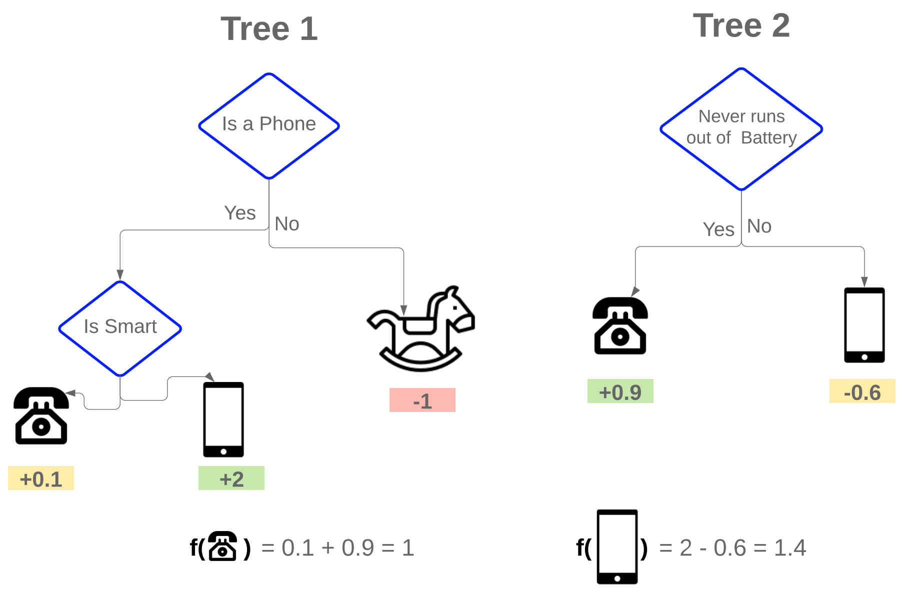


Figure 3.1. An exemplary illustration of how the scores in the respective leaves of Tree 1 and Tree 2 are added up to give the final score.

3.3 Collection of Data

The listings dataset is created with data from a large real estate portal Immobilienscout24 [39]. The portal is amongst the most popular in Germany. Specifically, from its Hamburg rental listings section is where the web scraping algorithm collected the core data for this work. The data is from the period between July 2016 and November 2018.

Web Scraping is an integral part of this thesis, since it is the tool used to acquire all data about the apartments and their main attributes. One definition of the term web scraping algorithm is:

> A more recent variant of the web crawler is the web scraper, which looks for certain kinds of information - prices of particular goods from various online stores for instance - and then aggregates it into new web pages, often using a database. [3]

It is important to understand that the goal is to acquire the values of specific variables contained in the `HTML` code of each listing. The algorithm has to be written, so that it

locates these variables reliably and extracts the corresponding values for each instance (listing). Detailed analysis of the extracted variables follows in Chapter 4. Here, a brief description of the most important tools used in the process is given.
A package that lets the user send HTTP requests and that captures the response. A HTML Parser to navigate and search the acquired HTML tree with its many tags. It is also the main tool to find the attribute-value pairs in the HTML code that hold the values of the desired variables. If the correct tag is specified, the value of the associated

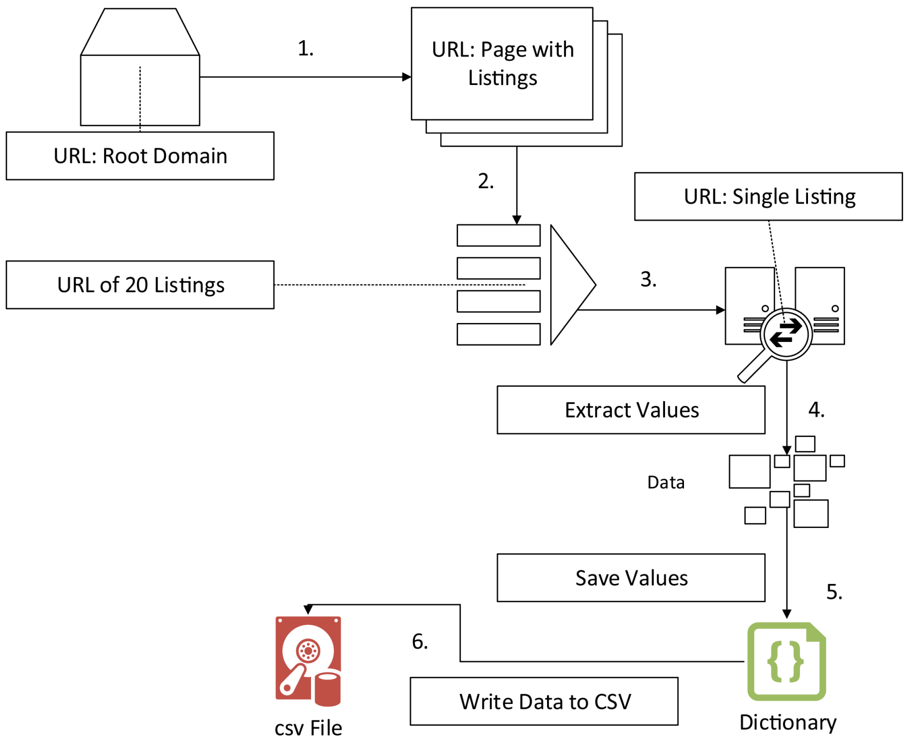


Figure 3.2. The steps of collecting the data with the web scraping algorithm.

variable is the output. Some script tags of type `<script type="text/javascript">` have nested JSON elements in them, which also have an attribute-value pair structure. However, extracting values from these nested JSON elements was not as easy as with HTML. The problem is that they can not be reliably separated from the surrounding text in the script tag. The starting index varies depending of the content before the JSON element and the ending index varies as well, depending on the content of the JSON element. Without being able to extract the JSON element with its associated structure, it is unstructured text. In order to get reliable results for the large number of listings on the platform, regular expressions were used. That is, after the conversion of HTML to string format. These are the main tools used here, all from within Python.

The acquisition of the data for the predictions is done using a self-written web scraping algorithm. The structure of the algorithm follows a simple principle in order to extract all the relevant data from every single listing in the set of listings. Figure 3.2 illustrates the process for one loop of twenty listings. The terms used here to explain the process are the same as the ones used in Figure 3.2. The algorithm collects data from roughly 12000 Single Listings (Step 3. Figure 3.2) and 610 Pages with Listings (Step 1. Figure 3.2), each with 20 Single Listings. The steps are: 0. get the URL Root Domain (Start in Figure 3.2). 1. get the URL of the first/next page with the URL of 20 Listings (Step 1. Figure 3.2). 2. get URL of each Single Listing, with 20 per page (Step 2. Figure 3.2). 3. go to URL of each Single Listing, one by one (Step 3. Figure 3.2). 4. Extract the specified values from each Single Listing (Step 4. Figure 3.2). 5. Temporarily Save Values from the previous step in a Dictionary (Step 5. Figure 3.2). 6. Repeat steps 1 to 5 for all Pages and Listings. Save all values from step 4 in the Dictionary for each Single Listing. 7. Convert the Dictionary with the values to a csv File (Step 6. Figure 3.2). A list of all received variables can be found in Table 1 of the Annex.

Table 3.1. Names of the variables used in the further steps in the left column with a description of each one in the right column.

Variable	**Description**
kitchen	Listing has a fitted kitchen yes/no
elevator	Building has an elevator yes/no
ancil_costs	Costs that are not included in the base rent (Eur)
lat	Latitude value of the listing
lng	Longitude value of the listing
const	The year the building was constructed
base_rent	The base rent of the listing (Eur)
sqm	The living space of the listing (m^2)
no_room	The number of rooms the listing has
balcony	Listing has a balcony yes/no
floor	The floor the listing is on inside the building
number_pics	The number of images uploaded for the listing
dl_speed	The internet download speed (mbit/s)
ul_speed	The internet upload speed (mbit/s)
time_gap	The time (days) the listing was online

In Table 3.1, only the variables ultimately used are listed for further reference.

3.4 Preparation of Data

In the following section the steps taken to transform the raw data into clean data, suitable for detailed analysis and in the right format for training the models with it, are explained. To aid the understanding of what characteristics tidy data has, a definition follows. A "tidy" dataset fulfils the following principles [50, 90]:

1. Each variable forms a column
2. Each observation forms a row
3. Each value has its own cell

Data Types Tidy datasets enable quick data analysis and visualisation. Further most machine learning models do not cope well with missing data, outputting an error message only. As a result, all the data in a given column have the same data type, that is appropriate for the variable of the column. The most common data types in this work are: 1. string or character (e.g. "base rent","well kept") 2. numeric (e.g. 2, 5.7,-15) 3. datetime64 (e.g. "2018-08-01").

First Cleaning Steps To gain a tidy dataset several case dependent cleaning steps have to be performed on the raw dataset from the web scraping part. The cleaning steps used here for the core variables are illustrated in Table 3.3. Here, a description of these steps is given. The **split** step refers to splitting the content of a cell, by a fixed character such as "," or white space " ". Splitting on white space was used for the variable `plz` for example, as shown in Table 3.2. **Stripping** is removing any kind

Table 3.2. Illustration of variables before and after the cleaning process.

Variable	Variable before	Variable after
lat	['lat: 53.547867859440416,']	53.547867859440416
plz	20459 Hamburg Neustadt	20459

of specified characters before or after the data of interest. It overlaps with the **extract** method where specific characters or types are extracted from the entry. This was used to get the latitude and longitude values, as shown in Table 3.2 for variable latitude (`lat`). **Replace** was mainly used to delete unwanted white space or change separators. Oftentimes the decimal separator used in raw data is comma (","). That is a problem, since for this analysis a float type number must have a dot (".") as a separator. Interestingly all the variables from parts of the `HTML` data that can not be seen on the actual webpage of the listing, are already in this format and all the ones visible to the visitor have comma as a separator.

Table 3.3. Shows core variables, cleaning steps and what each variable measures once cleaned. The entry "x" indicates that this step was used for processing the variable.

Variable	**split**	**strip**	**replace**	**extract**	**conversion**	**measures**
const	x	x	x	x	string	year built
plz	x				string	area code
lat/lng			x		float	part of location
onlinesince			x		datetime64	date posted
offlinesince			x		datetime64	date taken offline
rooms			x		float	number of rooms
pic			x		integer	number of images
elevator			x		binary	elevator yes/no
kitchen			x		binary	fitted kitchen yes/no
download speed			x	x	float	internet download speed
upload speed			x	x	float	internet upload speed

Type Conversions The raw data have always the format of `string`. It is appropriate for the values of some variables, mainly of type nominal categorical. An example is `plz`. While it contains only numbers, it does not make sense to order its values, e.g. from small to large. Each unique value represents a geographically bounded area, each with a variety of different features. Other features of ordinal nature have to be converted to either `integer` or `float`. The conversion is usually the last step in the cleaning process, since the data have to be in the right format, as seen in Table 3.2 in the "variable after" column for example. There is a special type of numeric variable in the form of a binary variable. The domain of these variables d_i with $i \in L$ and L denominating the set of all listings, is $\{0, 1\}$. The value of d_i is 0, if the measured feature is absent in listing i and 1 if it is present in listing i. This type of variable was used in the core set of variables, to measure whether the listing has a fitted kitchen and whether an elevator is in the building (see Table 3.3). For a given listing, the variable measuring whether there is a fitted kitchen in the apartment (`kitchen`), has value `kitchen` $= 0$ given there is no fitted kitchen and value `kitchen` $= 1$, if there is one.
Another type conversion is from `string` to `datetime64`. It was used to create the variables `onlinesince` and `offlinesince`. The output is a date and time for each valid entry in the data series, which then can be used to create a `timedelta` object. In this work the `timedelta` variable measures the time in days that the listing was online on the platform, before it was unlisted.

Further Cleaning Steps

Dropped Data All rows consisting of only `nan` values were dropped, with nan an acronym for *not a number* and a placeholder for missing values in the data series. The same goes for duplicate rows. All negative values were set to `nan`, apart from variables that could contain such values.

One of the most important variables is the geolocation of the listing. There are 9421 valid entries in `lat` and `lng` which have missing values in identical index values. That leaves 2902 missing entries. For 59 of these there is address data. Using the geocoder library with its `komoot API` [6] with the services of komoot [82], these 59 values were filled, 9480 entries are valid.
Since there was no way to acquire the remaining missing data of `lat` and `lng` coordinates and since imputing the data would possibly corrupt the up to this point valid data for `lat` and `lng`, all the rows with missing values in these two columns were dropped. As a result, the dataset now consists of 9480 observations. For variable `const` all values before the year 1850 were set to missing values. `kitchen`, `elevator` values that were missing were set to 0, as the original variable had either value "elevator" or "kitchen" respectively or `nan`. Other columns with more than 10% missing values were dropped, since there was no way to impute the missing values that does not risk altering the distribution of the data. Missing values were imputed for the following columns, the first number in brackets is the number of non missing values, the second is the percentage of missing values in relation to the number of all observations: `const` (8743, 7.8), `ul_speed` (8961, 5.5), `dl_speed` (9005, 5), `ancil_costs` (9370, 1.2). The method used to fill missing values was the mean of all the valid values in the column.

3.5 Spatial Joins

In addition to the core variables mentioned above, the longitude and latitude data from the listings was used to create several other features that measure attributes at the zone level. Each longitude and latitude pair forms the Global Positioning System (GPS) [31] coordinates of the corresponding object. In this case, each listing can be placed on a map according to its GPS coordinates and the spatial relations to other objects with known GPS coordinates can be calculated. Calculating these relationships is a type of spatial join.

For this work it is made possible by open source data available for the Hamburg area. In particular the GPS coordinates of all subway stations (U-Bahn Stationen) [48] and suburban train stations (S-Bahn Stationen) [47] from the Hamburg area are collected. The collected data has to be altered and the GPS coordinates have to be extracted from the surrounding data.

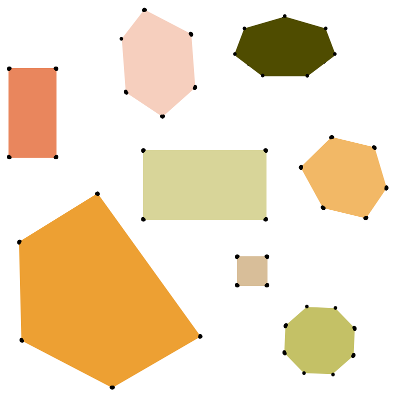


Figure 3.3: Illustration of possible polygon shapes. The objects are characterised by the coordinates of their corners marked in black. A multipolygon would consist of these polygons as array entries.

In a second step the values in the format "degrees, minutes, seconds" have to be converted to decimal degrees. This is necessary, since the listings use the decimal degrees format (epsg:4326). Another variable collected is the "noise data" [46]. In this dataset, areas within Hamburg are classified according to their exposure to noise in decibel (dB(A)). These areas are marked by "multipolygon" shapes. These multipolygons mark the boundaries of an area, that is spanned by all the contained arrays of polygons [18]. The polygons themselves have an outer ring marking the borders of the polygon by the coordinates of the corners, as illustrated in Figure 3.3. There can be elements within that ring in the form of their own coordinates [18]. The spatial join is performed after converting the "noise data" coordinates from epsg:25832 to epsg:4326 format, so they match the format of the listings. In the actual joining step it is checked for each listing whether it is within one of the areas of the "noise data". The areas contained in the "noise data" are all exposed to noise levels above the respective threshold. The same process is used to assign the values from the "neighbourhood data" [78] to each listing.

Distance to Station

The acquired GPS coordinates of all subway stations and suburban train stations from 3.5 is used to create the following features:

- `min_dist`: The minimum distance from each listing to either a subway station or a suburban train station in kilometres (km), whatever is closer gives the value
- `min_subway`: The minimum distance from each listing to the next subway station (km)

- `min_train`: The minimum distance from each listing to the next suburban train station (km)
- `smaller_0.6`: A binary variable defined as

$$\texttt{smaller\_0.6}_i = \begin{cases} 1, & \text{if } \texttt{min\_dist}_i < 0.6\,\text{km} \\ 0, & \text{if } \texttt{min\_dist}_i \geq 0.6\,\text{km} \end{cases}$$

  With $i \in L$, i is the i-th listing and L is the set of all listings in the sample.

The value of the threshold (0.6km) is chosen, in order to indicate whether the closest public transport station is within a walking distance of roughly 10 minutes at maximum [41, 87]. This is the number that the operator Hamburger Verkehrsverbund (HVV) uses for its own calculations of the catchment area for any suburban and subway station [26]. There is a reported increase in the valuation of a neighbourhood and the average rent per square meter, when it gains access to public transport [40]. This is part of the reason this data was joined with the listings data, so its predictive importance could be assessed.

Noise Data

The noise data gives the dB(A) values for the areas exposed to street noise for daytime and for night time as well. There is a difference between the two, as the minimum dB(A) value needed to be considered noise is $> 60 - 65\ dB(A)$ for the day time, while for the night time it is lower with $> 50 - 55\,dB(A)$. Since the dB(A) scale is logarithmic and there is a steep increase when moving up the scale, this is a substantial difference between the two entry level noise levels. There is evidence that street noise is more "annoying" at night than it is during the day [68] and it seems rational, given that most people sleep during night time. This data was added to the dataset of the listings as a categorical attribute for both day (`noise_day`) and night (`noise_night`). All entries not exposed to noise levels within the range of table 3.4 were marked with the value 0 in the corresponding column.

Table 3.4: The noise levels by day and night, starting with the lowest values at the top and increasing towards the bottom.

Day	**Night**
$> 55 - 60\,dB\,(\text{A})$	$> 50 - 55\,dB\,(\text{A})$
$> 60 - 65\,dB\,(\text{A})$	$> 55 - 60\,dB\,(\text{A})$
$> 65 - 70\,dB\,(\text{A})$	$> 60 - 65\,dB\,(\text{A})$
$> 70 - 75\,dB\,(\text{A})$	$> 65 - 70\,dB\,(\text{A})$
$> 75\,dB\,(\text{A})$	$> 70\,dB\,(\text{A})$

Neighbourhood Data

The neighbourhood data is a product of the "Sozialmonitoring Bericht" [79] published by the Urban Development and Housing Authority of the City of Hamburg. This is a translation of the main purpose and methodology behind the annual report:

> The Social Monitoring Report analyses and describes socio-spatial developments within the Free and Hanseatic City of Hamburg on an annual basis. The aim is to identify socio-spatial differences within the city and to identify neighbourhoods potentially in need of support. For this purpose, a small-scale analysis of selected indicators is carried out at the level of the 941 statistical areas of the City of Hamburg. In this way, sub-regions can be observed, compared and statistical areas identified in which cumulative problem situations may be suspected. [79]

Original German version in the Annex, Page 78.

There are two metrics from it that are used in this work. The first one is the actual "Statusindex" (`status`) which is has a categorical value for each of the 941 statistical areas. Its values are from low to high: "very low", "low","mediocre","high". This variable was converted to type categorical (a subset of type `string` variable) and the values were converted as follows: `{"very low": "1","low": "2","mediocre": ↩ "3","high": "4"}`, keeping the original order of the categories. The second one is the "Dynamikindex" (`dynamic`) with "-","0","+" as values describing the socio-spatial trend for each area, where "-" indicates a downward trend, "0" a stable situation and "+" an upward trend. These were converted to type categorical as follows:
`{"-": "-1","0": "0","+": "1"}`

3.6 Overview - All Variables

Table 3.5. A table showing all final variables in column 1 and their respective descriptions in column 2.

Variable	Description
base_rent	The base rent (Kaltmiete) of the listing
kitchen	Listing has a fitted kitchen yes/no
elevator	Building has an elevator yes/no
ancil_costs	Costs that are not included in the base rent
lat	Latitude value of the listing
lng	Longitude value of the listing
sqm	The living space of the apartment in m^2
balcony	Listing has a balcony yes/no
floor	The floor the listing is on inside the building
number_pics	The number of images uploaded for the listing
dl_speed	The internet download speed in mbit/s
ul_speed	The internet upload speed in mbit/s
time_gap	The number of days the listing was online
status	The status index of the statistical area the listing is in
dynamic	The trend of the status value for the statistical area
noise_day	The Exposure to noise during day
noise_night	The Exposure to noise during night
smaller_0.6	distance to next subway or suburban train station smaller 0.6km yes/no
min_dist	The minimum distance to the next subway or suburban train station
min_subway	The minimum distance to the next subway station
min_train	The minimum distance to the next suburban train station
const	The year the building was built

4 | Exploratory Data Analysis

Chapter 4 is about the visualisation of the variables in the data. It begins with an explanation of the univariate distribution for each variable in the dataset in Section 4.1. The correlations found between the variables is discussed in Section 4.2. In the same section, the noise data is analysed for both day and night and its impact on base rent specifically. What follows in Section 4.3, is a detailed heat map of the distribution of variable base rent by statistical area, plotted onto a map of Hamburg. The last section is Section 4.4. The general preprocessing done for machine learning is explained and it discusses standardisation of the data, for the machine learning part that follows in Chapter 5.

In order to understand the final data better, either visualisations or summary statistics were created. It is about describing the distribution of the variables and about revealing the correlation between them. Strong correlation between pairs of independent variables can be a sign of the existence of redundant variables. If the correlation is extreme, it may be a sign of collinearity. Both cases pose problems for the analysis of causal relationships between the affected variables and the dependent variable. The figures showing histograms or bar chart type plots, often have a truncated x-axis range. This range is narrower than the true range of values between the respective minimum and maximum of the variable. This was done, since the omitted values were not visible in the original plots. The scale of the plots would have to be bigger, for them to be visible, with the constraint from the paper size this was not possible. Please refer to Table 4.1 for the complete range of values.

Categorical & Numerical Variables

There are two different types of variables in this work. Namely discrete categorical variables with few possible values and continuous numerical variables. The categorical variables used here are `kitchen`, `elevator`, `balcony`, `status`, `dynamic`, `noise_day`, `noise_night`, `smaller_0.6`, `const`, `floor`. This was discussed in Section 3.4 and Section 3.5. This kind of variable has categories as values that have no natural order or scale [45] and each listing can fall in exactly one of the categories for each variable. Since all, but noise_day, noise_night were converted to a numerical categorical variable type, only noise_day, noise_night are excluded from Table 4.1. It is not possible to give the summaries found in Table 4.1 for string type variables. The remaining variables have a natural, numeric scale and are therefore continuous numerical variables [45].

4.1 Univariate Distributions

The variables are shown in Table 4.1 with their summary statistics. Here is a description from left to right of what Table 4.1 shows. Count is the same for all variables, since this is the final, clean dataset. The mean is the mean of the values given in the unit of the variable.

Binary Categorical Variables In the case of the binary variables (kitchen, elevator, balcony, smaller_0.6) its value measures the percentage of listings that have the feature within the dataset. What can be observed for these variables is:

- kitchen: mean = 0.61 $\implies$ 61% of the listings have a fitted kitchen, 39% don't.
- elevator: mean = 0.21 $\implies$ 21% of the listings have access to an elevator, 79% don't.
- balcony: mean = 0.68 $\implies$ 68% of the listings have a balcony, 32% don't.
- smaller_0.6: mean = 0.48 $\implies$ 48% of the listings have access to either a subway or a suburban train station within a radius smaller than 600m, for 52% the distance is equal or greater than 600m.

Continuous Categorical Variables Continuing with the other categorical variables shown in Figure 4.2 it is visible that the distribution of floor has a mean of close to 2 and has a positive skew. floor is for the majority of observations 1 or 2. The status index is 3 for around 6000 listings and the majority of listings therefore and is close to the actual values of the "Sozialmonitoring Bericht 2018" [79]. The same goes for the dynamic index with almost all listing being in areas that have a stable situation in terms of their status class, again these values are close to the ones found in the "Sozialmonitoring Bericht 2018". Looking at const the bulk of buildings was built in the period after World War II (WW II), between 1950 and 1975 roughly. During WW II (1939-1945) large parts of the city were completely destroyed and so during the post-war period many buildings had to be rebuilt or newly constructed [16]. It might show that base_rent of apartments in buildings that were built during that time have a low

Table 4.1. A table showing all numerical variables in column 1 and their summary statistics in columns 2-11. For an explanation of the variable names, please see Table 3.5.

	count	**mean**	**std**	**min**	**10%**	**25%**	**50%**	**75%**	**90%**	**max**
kitchen	9480	0.61	0.49	0	0	0	1	1	1	1
elevator	9480	0.21	0.41	0	0	0	0	0	1	1
ancil_costs	9480	159.08	83.13	0	73	100	144.96	200	265	950
lat	9480	53.57	0.05	53.4	53.49	53.55	53.58	53.6	53.62	53.71
lng	9480	10.01	0.09	9.74	9.9	9.96	10.01	10.07	10.13	10.3
base_rent	9480	752.89	446.9	148.97	363.23	451	633	906	1318.37	5600
sqm	9480	66.2	26.71	15	37.21	49	62	77.93	97.01	350
no_room	9480	2.41	0.88	1	1	2	2	3	3.5	8
balcony	9480	0.68	0.47	0	0	0	1	1	1	1
floor	9480	1.98	1.79	-1	0	1	1	3	4	24
number_pics	9480	8.36	5.58	0	2	5	7	11	15	64
dl_speed	9480	83.81	25.92	6	50	50	100	100	100	200
ul_speed	9480	31.11	13.91	2.4	10	10	40	40	40	100
time_gap	9480	22.8	45.78	0	0	1	7	26	59	924
status	9480	2.84	0.7	1	2	3	3	3	4	4
dynamic	9480	0.02	0.28	-1	0	0	0	0	0	1
min_dist	9480	0.92	0.84	0.01	0.23	0.37	0.63	1.16	2.06	9.44
min_subway	9480	1.79	1.54	0.03	0.4	0.67	1.27	2.46	3.94	9.44
min_train	9480	2.36	3.03	0.01	0.26	0.44	0.92	2.99	7.63	14.17
smaller_0.6	9480	0.48	0.5	0	0	0	0	1	1	1
const	9480	1966.56	31.95	1850	1920	1953	1964	1992	2013	2019

`base_rent` value compared with apartments built in different periods, with similar features. The consideration comes from a simple supply and demand logic.

There is a sharp downward trend around 1975 to 1990 in the number of buildings built during that period with a pickup in the Late-1990s. For several years between 2000-2010 the number decreases once again, to reach levels as high as the ones seen around 1960 for the period 2010-2018. Right now, there is indication that a relatively high number of residential buildings are constructed from this sample. This does not completely align with the data in Figure 4.1, as according to the Statistikamt Nord [80] the maximum of apartments built between 1990-2017 was in 1995 and the pickup in construction after 2010 can not reach the levels of that period. The report of the Statistikamt Nord, however, includes non residential buildings and construction measures on existing buildings as well [80]. This may hinder a direct comparison between the here found data from the sample of all listings and their data. The overall trend however is similar across both data sources. There are reports that rent prices are rising at the moment in Hamburg [54].

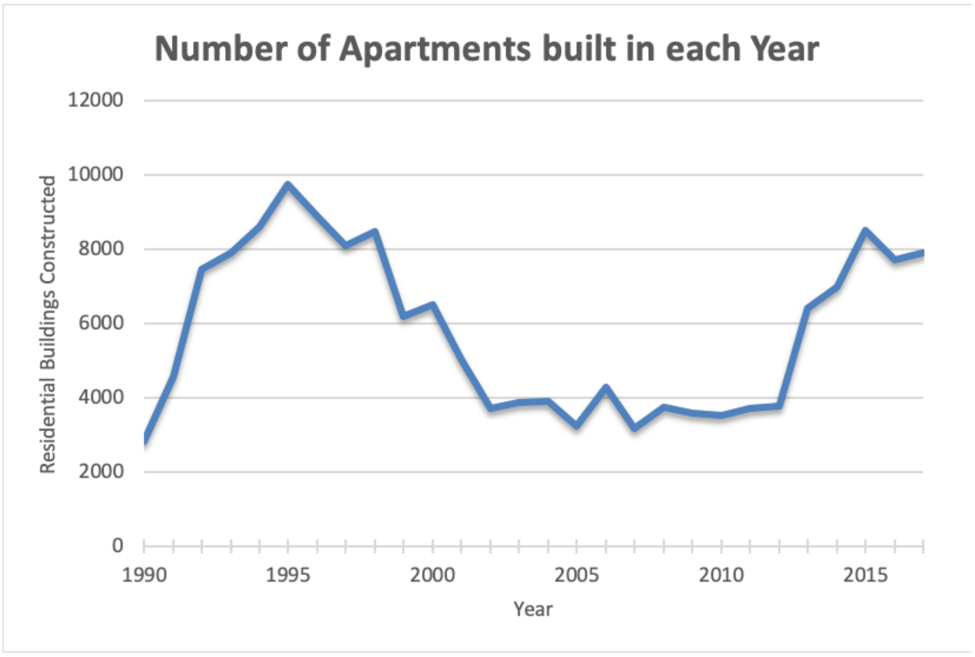


Figure 4.1. Line Chart showing the number of apartments constructed, in residential and non residential buildings, incl. construction measures on existing buildings (source: Statistikamt Nord) [80] for the years 1990-2017.

However, not as much as in other major cities in Germany possibly. This is also due to the fact of a relatively high number of apartments being constructed at the moment and a stable population in Hamburg, at least according to the Deutsche Bank German housing market 2018 (Deutscher Häuser- und Wohnungsmarkt 2018):

> In Hamburg, housing prices in the portfolio have risen by more than 70% since 2009. Rents are growing at a below-average rate compared with other major cities. The relatively brisk construction activity and the stable number of inhabitants are dampening rental dynamics. Low interest rates could therefore be the main driver for Hamburg's housing and house prices. Correspondingly, interest rate sensitivity could be higher than in other metropolises. In our baseline scenario, we expect mortgage rates to rise only slightly in 2018. House and apartment prices in Hamburg should therefore continue to rise strongly in the current year. [52]

Original German version in the Annex, Page 78.

For the prediction important is that there might be relevant rent increases for comparable apartments between the beginning of the timeframe in July 2016 and the end in November 2018, that might degrade prediction accuracy. `smaller_0.6`, the last variable in Figure 4.2 is discussed at the beginning of Section 4.1 with the other binary variables.

Continuous Numerical Variables Continuing with the numerical variables and their histograms in Figure 4.3. The histogram of base_rent shows a pronounced positive skew in the data series. The mean ($\bar{x}$) of 753 Euro for the series is low considering the value range of 149-5600 Euro. However, extreme values above 1318.37 Euro only account for 10% of the data. The standard deviation of 447 Euro is a better metric to show the large spread of this variable. The one Sigma (s) interval has limits (Eur.) $[306 = \bar{x} - s, \bar{x} + s = 1200]$, with the lower limit $\approx 40\%$ of $\bar{x}$ and the upper limit $\approx 160\%$ of $\bar{x}$. A similar pattern is observed for ancillary costs (ancil_costs), with the distribu-

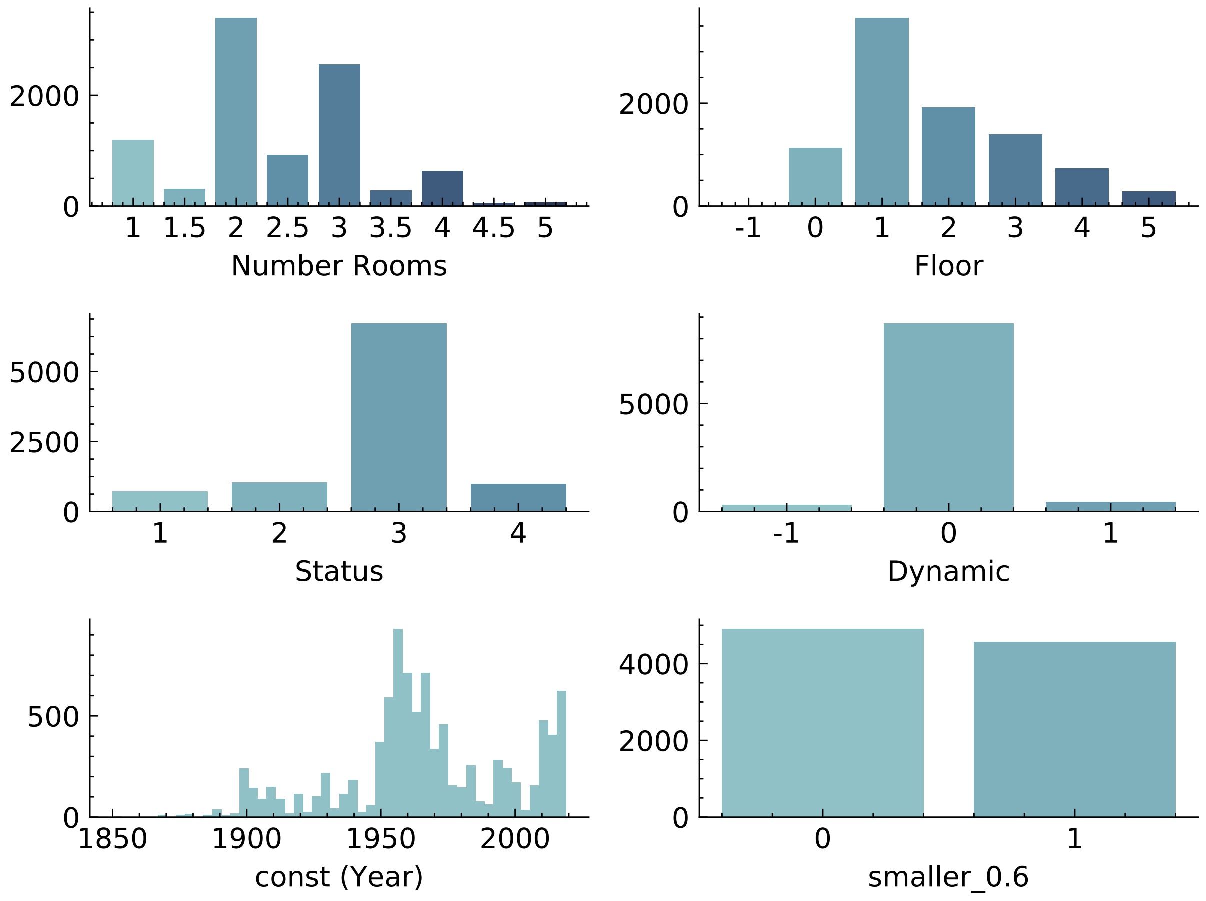


Figure 4.2. Distribution of the categorical variables with the variable name and its value on the x-axis and the absolute count on the y-axis. For an explanation of the variable names, please see Table 3.5 .

tion exhibiting a bit less skew and a more linear falloff past the 130 Euro mark. Since the determination of ancillary costs is not uniformly established regarding the costs contained within them, it is difficult to consider them as a consistent variable. Square meters (sqm) has a gaussian distribution like value distribution with a slight positive skew. The mean is $\bar{x} = 66$ with the one Sigma interval, given in m^2, of $[39, 93]$. This echoes the slight skewness for the series. The number of images that are uploaded to describe the apartment seem to be important, since the highest frequency found around 5 to 10 images and not zero. There are less than 10% of observations with 0 images, as can be seen in Table 4.1. The download (dl_speed) and upload speed (ul_speed) variables show that for both variables, observations gather around a few mbit/s values. With these findings they have to be considered categorical variables, as the value range is not continuous for the listings in the series. For dl_speed these are (mbit/s, count): (100, 6404), (50, 2105), (83, 484), (16, 453), (25, 36), (200, 5), (6, 2). For ul_speed: (40, 6404), (10, 2105), (31, 528), (2.4, 447), (100, 5). It is noticeable that for the majority of observations tuples can be formed from values of both variables that have near identical counts in the series. The variable measuring the days a listing was online on the platform shows that there was strong demand for the apartments in this series, since 25% of the data series was listed and delisted within two days, thus having the time_gap = 1 day value. 10% have a value of 0 days, meaning that they were online for less than 24 hours. The median is one week, see Table 4.1. There are values as high as 924 days in the series, however the portion of values 59 $\leq$ time_gap $\leq$ 924 makes up the highest 10% of all values for time_gap. Therefore, the impact is expected to be small on the prediction results, if there were outliers amongst these values. The minimum distance (min_dist) to either of subway or suburban train station shows that for 70.5% of observations the distance is $\leq$ 1 km and the median (50% quantile) is 0.63 km. This is in accordance with the observations described for variable smaller_0.6, where 48% had a distance of less than 0.6 km. With the 50% quantile at 0.63 km it shows, that the binary variable smaller_0.6 splits the set of observations into two almost equal parts. The distributions of the variables min_train, min_subway show a very similar picture. Since the variable min_dist is set to the smaller value of the two, its distribution is a concentration of the smallest values of the two. One can see that the distance to the nearest subway station has a high concentration of values close to zero and a sudden drop after approximately 1 km. In the case of the nearest suburban train station the distribution is shifted towards the mean of 2.36 km and falls off more slowly afterwards.

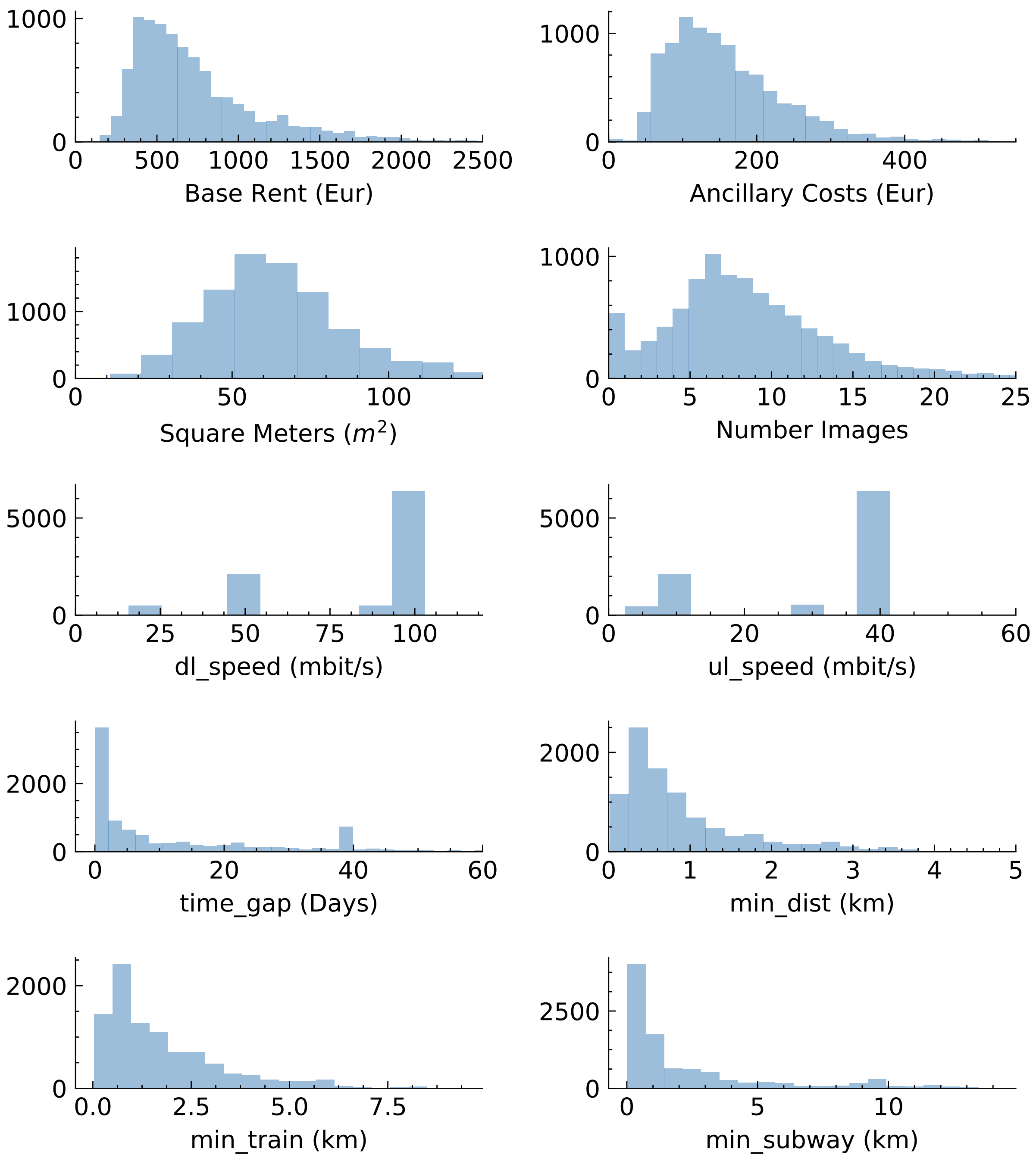


Figure 4.3. Distribution of the numerical variables with the variable name and its value on the x-axis and the absolute count in the data series on the y-axis. For an explanation of the variable names, please see Table 3.5.

4.2 Correlation

Hypothesis Test The distributions of all variables were checked for whether they may have a heavy tail. This was done by comparing them to the exponential distribution, as minimum requirement for possibly being heavy tailed. The reason is definitional: "The typical quantitative definition of a "heavy-tail" is that it is not exponentially bounded." [27]. It is a prerequisite for the use of the Pearson Correlation Coefficient (ρ) that the covariance for the bivariate distribution of the features exists and the respective standard deviation as well, entailing that there must not be heavy tails in the univariate distributions of the variables [70]. The detailed results can be found in the Annex, Table 2. Variables where the test results do not confirm the null hypothesis of: The distribution fulfils the requirements of ρ, at the $\alpha = 0.05 = 5\%$ level are: base_rent ($p \approx 0.004$), floor ($p \approx 0.0004$). ρ can still be used as metric, as long as the relationship between variables is a linear one. This is the case with only continuous relationships observed so far and the Lasso regression model showing it is capable of predicting base_rent well (Section 5.1), being a linear model. An assurance that the correlations between the regressors are of a linear nature was gained from the aforementioned hypothesis tests, with only variable floor fulfilling the minimum requirements to have a heavy tailed distribution. The values of ρ for the two aforementioned variables have to be critically assessed [27].

Pearson Correlation Coefficient Therefore, the Pearson Correlation Coefficient is used to calculate the correlations between the variables. Using this metric correlation between two variables is defined on the domain $[-1,1]$, where values of the coefficient ρ between -1 and 0, including -1, excluding 0, ($\rho \in [-1,0[$) show a negative correlation between the two assessed variables. The opposite is the case for values of ρ between 0 and 1 ($\rho \in]0,1]$), with a positive correlation. The strength of the correlation increases with increasing values of ρ. The negative correlation gets stronger with values of ρ closer to -1. In this data series one finds the strongest negative correlation with a value of -0.64 between smaller_0.6 and min_dist. This is a result of the definition of the two variables. Once min_dist goes below 0.6 km, smaller_0.6 switches from value 0 to 1. In the final model only smaller_0.6 is kept, since the other three variables min_dist, min_subway, min_train show strong correlation with it. The strongest positive correlation is observed for ul_speed and dl_speed with a value of $\rho = 0.9$, consequently only dl_speed is kept. This follows the discussion in the context of Figure 4.3 with nearly identical ratios between the two across observations. For sqm and no_room ρ is 0.81, which signals that the number of rooms does not contain information not already given by the living space variable (sqm). It is dropped therefor. Notable ρ values are:

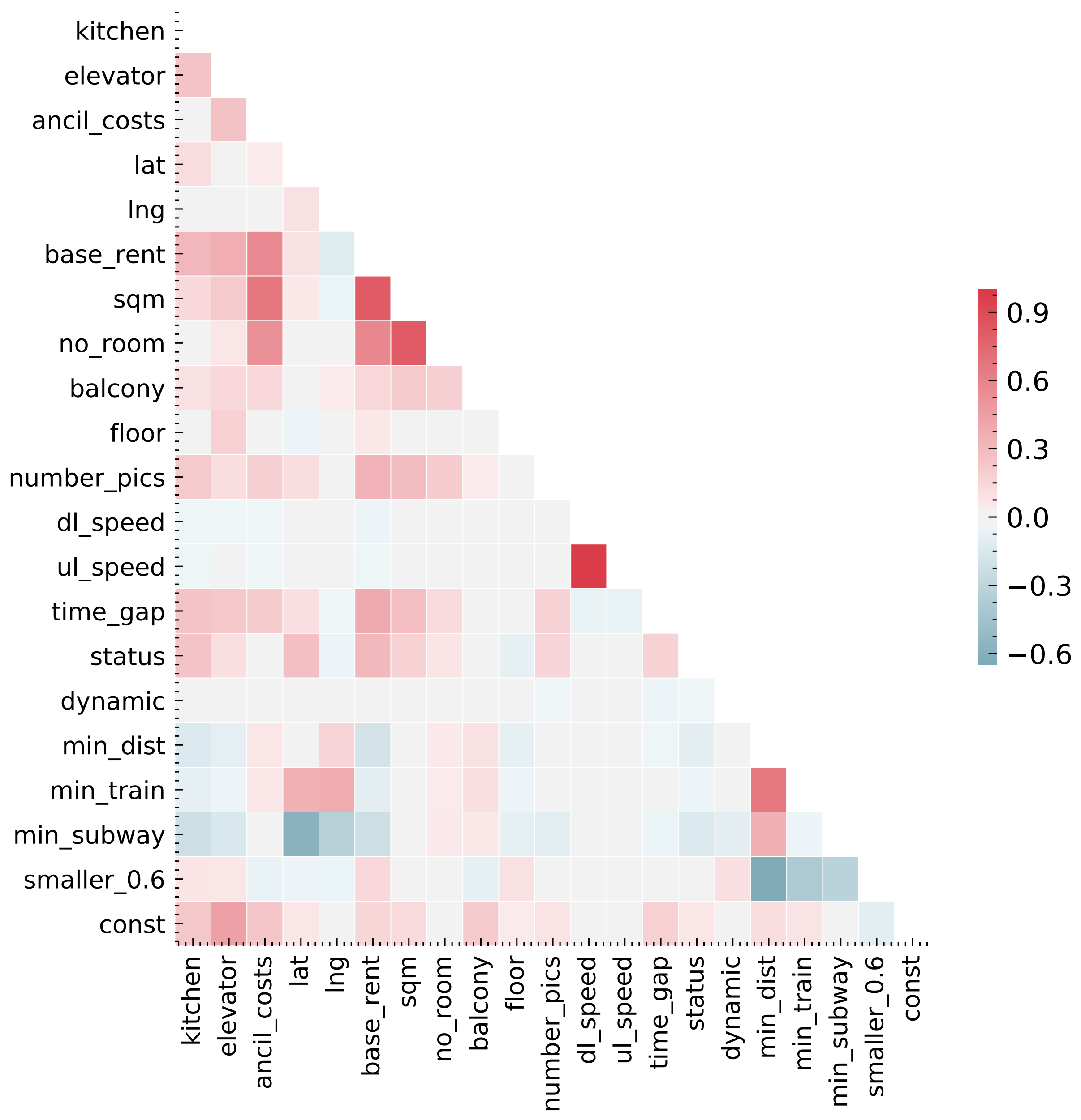


Figure 4.4. A correlation matrix with the correlation between the variables of the series. Values are calculated using the Pearson Correlation Coefficient. For an explanation of the variable names, please see Table 3.5.

1. `const`:

   - The year the building/apartment was built has $\rho = 0.43$ for `elevator`. It is plausible that younger buildings are more likely to have an elevator than older ones.

   - For `ancil_costs`: $\rho = 0.24$. There seems to be an increase in ancillary costs as the year of construction of the property increases.

   - For `kitchen`: $\rho = 0.23$. This suggests that it is more likely to find a fitted kitchen in an apartment with the increase in year of construction.

- Importantly there is a correlation between it and `base_rent` ($\rho = 0.16$). It is slight, but the positive value suggests a positive correlation between the two.

2. `base_rent`:

   - The highest $\rho = 0.82$ is for `sqm`. It is common that base rent and living space are correlated. The Hamburg housing market seems to be homogeneous regarding the relationship between these two variables for most part of the `base_rent` price range, in which the living space is very important for the amount of the base rent.
   - It is closely followed by the number of rooms ($\rho = 0.58$) and ancillary costs ($\rho = 0.56$). Given the strong correlation between `base_rent` and `sqm`, an increase in ancillary costs with an increase in the number of square meters therefore seems appropriate. Larger apartments often need more energy for heating and with increasing `sqm`, an increase in base rent is associated. Heating costs are mostly settled through the ancillary costs and are therefore part of these. The same considerations apply to the number of rooms.
   - A notable mention is the ($\rho = 0.38$) for `time_gap`, indicating that listings for expensive apartments are online for longer, compared to affordable apartments.

Variables beside these that have predominantly positive ρ values are `status`, `time_gap`, `number_pics`, `balcony`, `sqm`. There is a slight positive correlation indicated between `lat` and `base_rent` and a negative one for `lng` and `base_rent`. Since longitude measures the horizontal movement (West-East-axis) and its values get larger going eastwards, the negative correlation suggests that the base rent is lower in the eastern part of Hamburg and higher in the western part. With latitude measuring the vertical movement (North-South-axis) and rising values going towards North a positive correlation hints at higher base rent values in the northern part of Hamburg than in the southern Part. These interpretations might not be true, since it is unclear how many samples there are in the series for each fraction of `lat`, `lng` that is part of the Hamburg area and whether the listings in the sample are spatially distributed in a way around Hamburg, so these correlations are based on solid grounds.
With the elimination of variables `ul_speed`, `no_room` and by only keeping `smaller_0.6`, redundant variables are dropped. Collinearity is also not the case amongst the independent variables judging by ρ, after these steps are taken.

Noise Data

Noise Day The ridge plots of `base_rent` by `noise_day` show that most of the data is exposed to less than >55-60 dB (A) (row 1 in Figure 4.5) of noise during the daytime. There are only 840 entries in total that make up the data for the other rows. The non zero entries have the distributions (row, zone : entries): (2, >55-60 : 165), (3, >60-65 : 203), (4, >65-70 : 269), (5, >70-75 : 170), (6, >75 : 33). One notices from looking at Figure 4.5, that for each zone most of the `base_rent` entries are slightly left of their mean of 753 Euro and that the upper quantile of `base_rent` entries tends to decrease with increasing noise levels. However, there are some outliers that do not follow this pattern, as is the case in row 5 around the 2000 Euro mark.

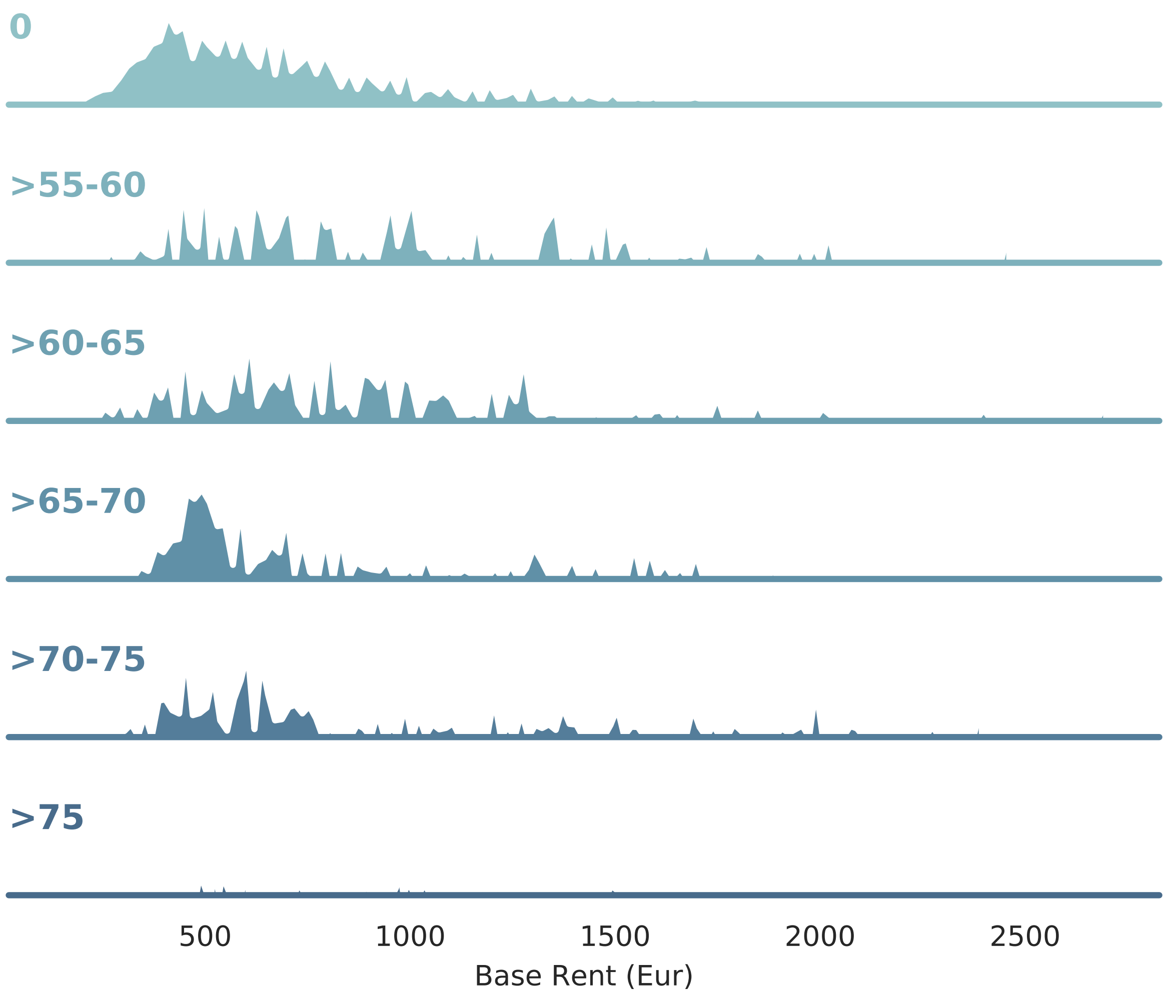


Figure 4.5. The relationship between base rent (Eur) on the x-axis and its frequency by noise level during day (dB (A)) on the y-axis. The noise level increases from top to bottom.

Since the sample size for all but row 1 is very small, sample variation could influence these findings.

Noise Night For the night time data there are around 600 non zero entries and the observed pattern is similar to the one found in Figure 4.5. The ridge plot for `noise_night` can be found in the Annex, Figure 1.

4.3 Heat Map of Base Rent

Using the GPS coordinates of the listings it is possible to make a plot (heat map) that shows the spatial distribution of the normalised `base_rent` variable on a map of Hamburg. To show the univariate distribution of base rent without the influence of living space a variable "base rent per square meter" (`rent_sqm`) that gives the value of base rent normalised by the size of the apartment is created. Formally it is for the i-th listing of the set of all observations (L):

$$\texttt{rent\_sqm}_i = \frac{\texttt{base\_rent}_i}{\texttt{sqm}_i}, \quad i \in L$$

Table 4.2: The 10 highest values for rent per square meter in the right most column grouped by statistical area first and by district second.

stat. area	district	rent_sqm
6004	Hammerbrook	29.05
37006	Harvestehude	23.11
27006	Othmarschen	22.52
37008	Harvestehude	19.97
2002	HafenCity	19.14
36009	Rotherbaum	18.86
27003	Othmarschen	18.67
43012	Stellingen	17.68
45011	Eppendorf	17.51
35003	Eimsbüttel	17.47

In Figure 4.6 it can be seen by the grey areas, that not all areas inside Hamburg are represented in the series. There are 719 statistical areas (definition, see Section 3.5, Sozialmonitoring) with data and 222 without. The median count of observations per statistical area is 78 and the standard deviation is 100. So for many areas there is a large number of observations and the mean value of `rent_sqm` is meaningful. It is evident, that the mean of rent per square meter by statistical area has a tendency to increase from East to West. It is higher for areas closer to the city centre, than for ones further away from it. The following values are given in Euro per m^2. The minimum for `rent_sqm` is 3.77, the mean 11.27 and the median is 10.93 with a maximum value of 39.2. See Annex, Figure 2 for a box plot.

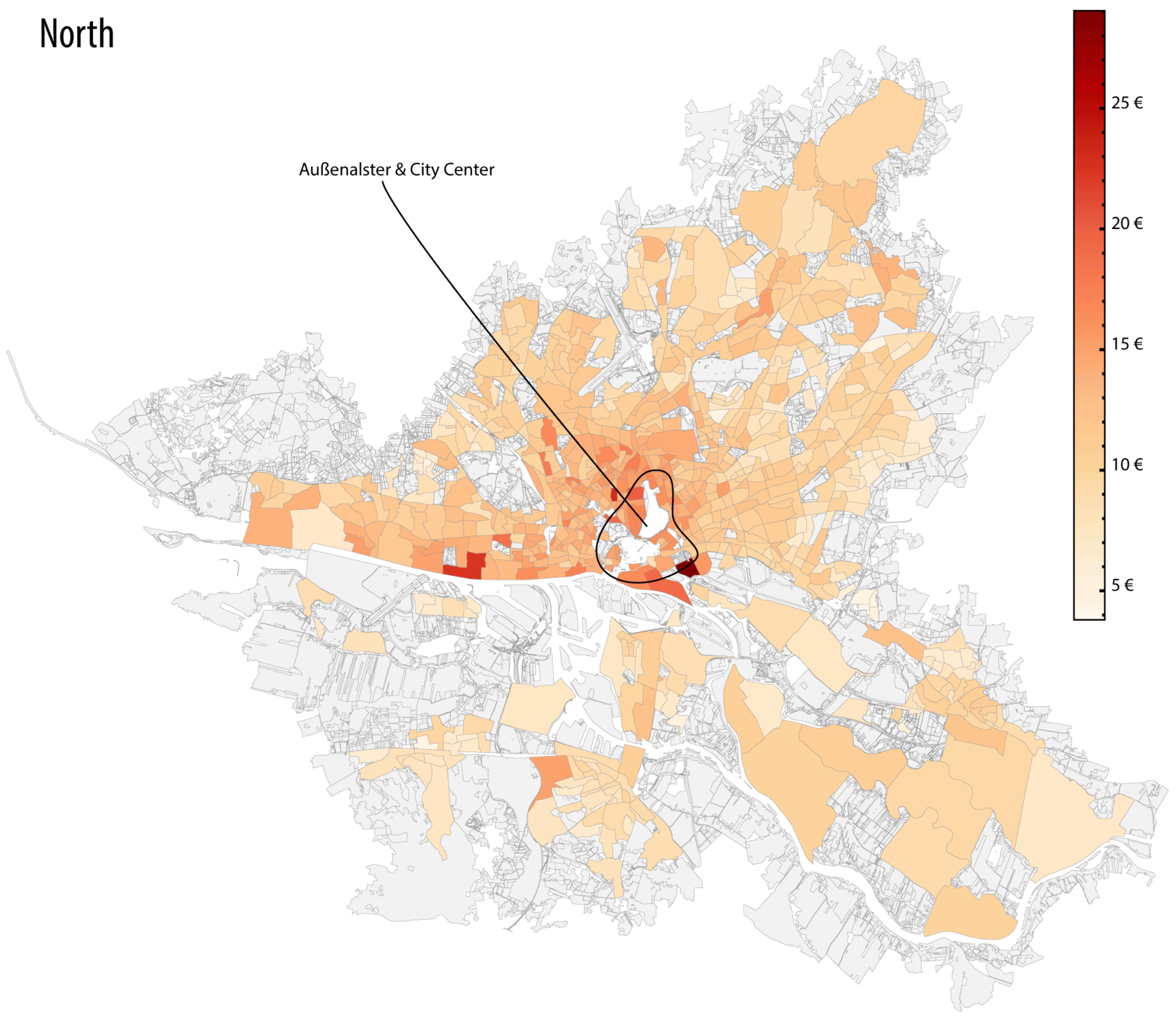


Figure 4.6. The heat map of the mean value of base rent per square meter (Eur.) aggregated at the statistical area level to give one uniform colour shade per area. The legend shows the corresponding shade for each rent per square meter value in Euro. Grey areas mark neighbourhoods in Hamburg from which no listings are in the series.

4.4 Preprocessing

The dataset was split into two parts for the machine learning. The first one is the training set, which contains 80% of the listings data in the dataset. The training set is used to train and evaluate the models, during the initial fitting and the subsequent Hyperparameter Optimisation process. A 3 fold cross-validation is used to measure the performance of the different model configurations during the Hyperparameter Optimisation process. With 3 folds, the training set is randomly split into 3 subsets (folds). The model is trained on 2 folds and evaluated on the remaining fold (test set). This process is repeated 3 times, so that each combination of folds was once used for training and that each fold was once the test set. The result is 3 scores per parameter combination tested. Each model configuration tested during the optimisation is assessed solely on the mean of the individual accuracy scores for each of the 3 repetitions. The second set that makes up 20%, is called a validation set. It is only used once at the end of the optimisation process and is meant to give an estimate of the performance one can expect from a model on unseen data. Sampling was done randomly with a constant random seed to enable comparable results between sessions.

A Need for Standardisation?

For the machine learning part, a standardised and normalised version (standardised data) of the up to this point described dataset (non standardised data) was created to compare the prediction results between the two versions. This was done, as the original dataset has many different units in its variables that have very large differences in the value range in some cases. An example for non normalised value ranges is the latitude variable `lat` whose values range from 53.4 to 53.71 (Table 4.1) and `dl_speed` which values range from 6 to 200. There are also variables in the dataset with very different distributions, as is the case with `dl_speed` with a strong negative skew and `base_rent` with a strong positive skew. To fix this all variables are transformed so that they have a mean value of 0 and a standard deviation of 1. The function `StandardScaler` from `sklearn.preprocessing` was used for this [76]. The standardisation was made after splitting the data into training and validation sets. This was done to ensure the training data is not influenced by the scale of the data in the validation set.
Another technique used is *binning*. Here the value range of a variable is split into a user specified number of bins. It was used for variable `const`, the year the building was constructed. The value range of this variable is very broad with values from year 1850 to year 2019. Each bin size was chosen to contain a period of 20 years. This means that variable `const` was split into 9 groups. This number was chosen, so that the coefficients of the `Lasso` model could possibly give a better explanation of the effects that different time ranges of `const` have on the value of the dependent variable `base_rent`.

5 | Machine Learning

This chapter documents the process of fitting, optimising and interpreting the models. First, the evaluation metric used here for measuring the accuracy of the models is explained and important declarations are made. The following Section 5.1 deals with the Hyperparameter Optimisation. The `Lasso` model is first optimised, then the `XGBoost` model. An overview of the RMSE values by model, method and set is found in Section 5.2. The coefficients of the `Lasso` are interpreted in detail in Section 5.3 and the feature importance plot of the final `XGBoost` model is described in Section 5.4.

Evaluation Metric The evaluation metric used is the Root-Mean-Square-Error (RMSE) and its values are in Euro. It is defined as:

$$\mathsf{RMSE} = \sqrt{\frac{\sum_{i \in L} (\hat{y}_i - y_i)^2}{L}}$$

With L being the set of listings and i one listing in L. The RMSE then gives the square root of the mean value of the squared deviations of the predicted results $\hat{y}_i$ from the true values y_i. The goal is to minimise the value of RMSE on the respective left out fold for each of the 3 times repetitions, by using Hyperparameter Optimisation on a model. The model should make predictions $\hat{y}_i$ as close to the true values y_i as possible, after the Hyperparameter Optimisation procedure. This minimises the term $\sum_{i \in L} (\hat{y}_i - y_i)^2$ and as a result the value of RMSE. The validation set is then used to estimate the performance of the final model. If the results on the validation set are much worse than the results during the optimisation process, this can be a sign of *overfitting*. The values mentioned here are rounded to two digits, since there is no information gain from having more decimal places for the objectives of this work. Calculations were made without any rounding of numbers.

Declarations For the following these declarations are made: Numerical values in brackets, like (149.45), for either model refer to the RMSE value of the particular configuration of the model on non standardised data. Values of the form (149.54, 190.45) denote the RMSE values for a model configuration on non standardised data, standardised data (RMSE on non standardised data, RMSE on standardised data). Also, *data options* will refer to both versions of the data, the non standardised and the standardised data together.

The models are fitted and optimised using the `Python` library `scikit-learn` [69] (`sklearn`). `Lasso` is part of the `sklearn` toolkit [73], while `XGBoost` is accessed through `sklearn` using an API [61]. Like that there is a wrapper for `XGBoost` so it can be handled like any other `sklearn` model.

5.1 Hyperparameter Optimisation

During the initial fitting and testing the `XGBoost` model gave better predictions than the `Lasso` with values of (145.75, 144.8) for `XGBoost` and (195.62, 196.02) for `Lasso`. This difference in accuracy between the two models was expected though, as `XGBoost` is an ensemble based model and it proved to be a very successful one in various competitions [66].

> As the winner of an increasing amount of Kaggle competitions, XGBoost showed us again to be a great all-round algorithm worth having in your toolbox.

Winners Interview, Mad Professors. [23]

These values mark the baseline by which changes to the parameters of both models will be compared with.

Optimisation `Lasso`

For `Lasso` there is only one parameter alpha (α) that can be optimised. Here a grid search was used for the optimisation, given the low dimension of the search space for this single hyperparameter. This parameter has a value range between 0 (`Linear ↩ Regression`) and 1 (default for `Lasso`), see Section 3.2. This parameter is also called λ in other libraries. A grid search was used with a list of values for α in the range of $\alpha \in [0, 1.2]$ in increments of 0.1. The smallest values for RMSE were found for $\alpha = 0$ on the training set without any issues. Using this value on the valuation set had the model not converge during the coordinate descent. On the `scikit-learn` [69] website the following can be found regarding alpha:

> **alpha : float, optional**
> Constant that multiplies the L1 term. Defaults to 1.0. alpha = 0 is equiva-

> lent to an ordinary least square, solved by the LinearRegression object. **For numerical reasons, using alpha = 0 with the Lasso object is not advised**. Given this, you should use the LinearRegression object. [73]

Since the value 0.1 for α showed identical values for RMSE (see Annex Figure 3), this value was chosen for the model to converge. With $\alpha = 0.1$ the predictions on the validation set gave (190.68, 190.64). This is an improvement over the initial scores on the training set and even over the best scores during the grid search: `Best: ↩ 195.160000 using {'alpha': 0.0}`. These were identical for both *data options*.

Listing 5.1: Specification of the final Lasso model with parameter names from the sklearn library.

```
from sklearn.linear_model import Lasso

Lasso(alpha=0.1, copy_X=True, fit_intercept=True, max_iter=1000,
  normalize=False, positive=False, precompute=False, random_state=42,
  selection='cyclic', tol=0.0001, warm_start=False)
```

The model started already with a high accuracy and was able to use the input variables well for prediction. There was little increase in performance found with the grid search when one compares the initial training score of 196.02 with the best training score of 195.16, but one has to keep in mind that only alpha can be optimised within a small range and that the model generally has its strength in explaining the results as well. The different *data options* had very little impact on the performance of this model. There was no result in the entire process of training, optimisation and evaluation of the model where the difference in RMSE was greater than 1 unit for the different *data options*.

Optimisation `XGBoost`

With the `XGBoost` model there are more parameters that can be set by the user, see Section 3.2. It is more complex than `Lasso` in this regard.

Description of Parameters The following parameters were optimised using structured experiments. The total number of trees is given by `n_estimators` and the maximum depth of each tree by `max_depth`. The `learning_rate` limits the information gained from a newly added tree by adding a weighting factor to its correction. These two parameters can have a strong interdependence. If `learning_rate` has a low value, the corrective output value (or information gain) of the newly added tree can not be fully included in the model. This could potentially mean, that improving the RMSE requires adding more trees when the learning rate is low compared to when the learning rate is high. An analogous relationship may also be plausible for `n_estimators` and `max_depth`. Here the assumption would be that shallow trees are associated with less information gain per tree and trees with more splits can generate a higher information gain from the inputs. According to this logic, a smaller number of deep trees would then be required compared to shallow trees in order to achieve the same information gain. The remaining parameters are part of the stochastic gradient boosting procedure [45]. `colsample_bylevel` is the sample size of the bootstrap sample, in percent of the entire set of columns in the dataset, that is randomly sampled when constructing each level (split) of a tree. `colsample_bytree` is the fraction of columns that is used for the construction of an entire tree. `subsample` gives the percentage of rows that are used in each sample to construct a tree.

Grid Search

Protocols of Structured Tests For the tests, only the selected parameters are changed from their default values, all others have their default values. Detailed protocols for each test together with the idea behind each test specification are given in the following. The interaction between the questions asked for each test and the results gained from it is explained as well. They are grouped by parameter names and sorted in chronological order. They are summarised by the bullet points, to give the structure and the key insights for each of the conducted tests.

Number of Trees

The experiments used for the parameter `n_estimators` (which defaults to `n_estimators` $=$ 100) are:

- Test 1: Maximum number of trees in the interval $[50, 950]$ with steps of 50. The lowest RMSE value was achieved with 950 trees. Since it is the maximum in the test range another test is conducted.
- Test 2: Interval $[800, 1750]$ with steps of 50. Best result with 1750. It is the highest value in the testing range again and a very high optimal value. Adding a lot of trees can lead to overfitting and therefor the parameter is tested with other ones in conjunction and no further single testing is done.
- Tests were conducted for both *data options* and results were identical for large numbers of trees. Since only a large amount of trees gave optimal results, at least for `n_estimators` both *data options* are identical.

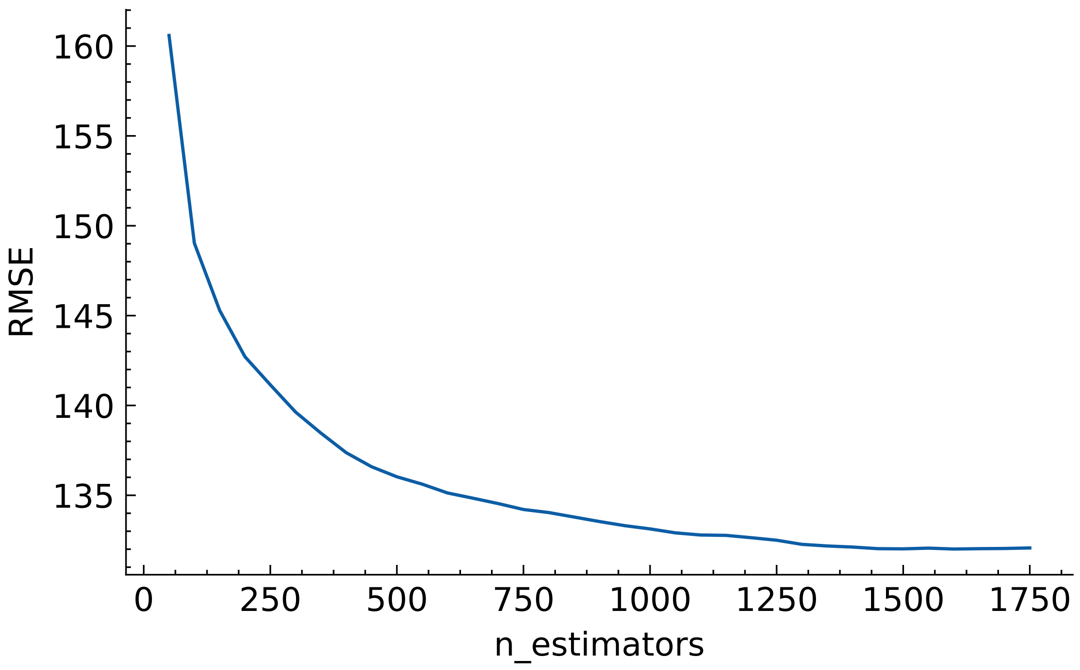


Figure 5.1. Plot showing the RMSE for number of trees tested.

Maximum Depth of Trees

For `max_depth` (default `max_depth` = 3), the test is:

- The interval with values is $[1, 9]$ with a step size of 2. Best result is found with `max_depth` = 5. This value is in the middle of the testing range and no further testing for this parameter by itself is done.
- There was no difference in the optimum value of `max_depth` between *data options*.

Number of Trees & Maximum Depth of Trees

In order to test how good the results from the individual tests for `n_estimators` and `max_depth` are, an experiment testing both simultaneously was conducted. The specifications are:

- For `max_depth` the interval is $[1, 9]$, step size is 1. The step size was lowered, to test whether optimal values could be found for even values excluded from the individual test of `max_depth`.
- For `n_estimators` the interval is $[50, 950]$, step size is 50. It was believed that the number of trees should not exceed 1000, because of possible overfitting and so the interval was not changed for this test.

- Results were `max_depth` = 5, `n_estimators` = 950.
- This was treated as a sign that `max_depth` = 5 is optimal. It shows that more testing is needed to find the optimum for `n_estimators`.
- Only non standardised data was used, since results were identical for both individual tests of the selected parameters.

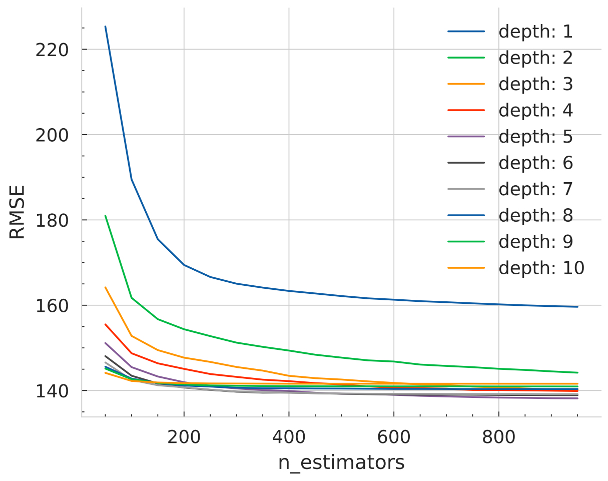


Figure 5.2. Plot showing the RMSE values for different values of `max_depth` and `n_estimators`, tested together.

Learning Rate

An experiment was also done for `learning_rate`. It was an experiment that was hoped to give insight to the very high number of trees in the tests before. The default for `learning_rate` is 0.1. The hypothesis was that, if `learning_rate` is too low and the model can not gain enough information from a newly added tree, then it might need a high number of trees as observed in the tests for `n_estimators`.

- To test the hypothesis, higher values for `learning_rate` were included in this grid search. The range of values is `[0.0001,0.001,0.01,0.1,0.2,0.3,0.4]`.
- Best result is achieved with `learning_rate` = 0.2.
- The results are in line with the hypothesis, since the optimum value found in this test is higher than the default value of 0.1 for `learning_rate`.
- Findings do not differ for the different *data options*.

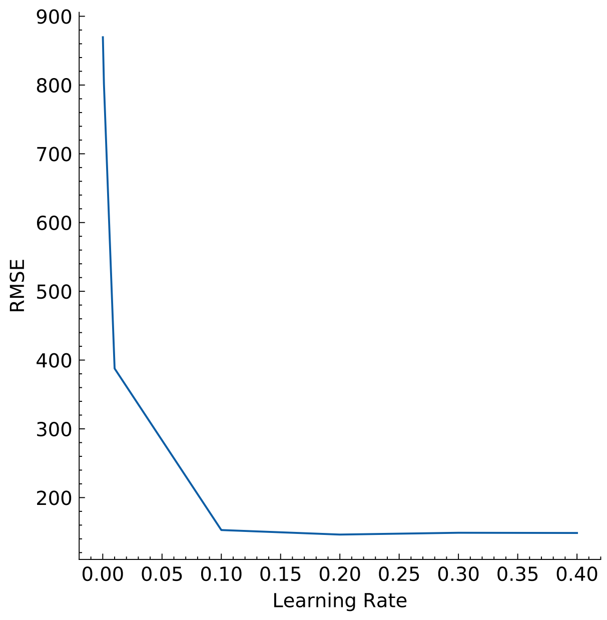


Figure 5.3. Plot showing the RMSE values for different values of `learning_rate`, during isolated testing.

Number of Trees & Learning Rate

A joint test for `n_estimators` and `learning_rate` was performed, since the 0.2 value for `learning_rate` suggests that the default of 0.1 is too low.

- To test this a broad range of values for `n_estimators` is used with $[100, 1750]$ and a step size of 50. This includes the entire range that was tested during the two individual tests for `n_estimators`.
- The range for `learning_rate` is `[0.0001,0.001,0.01,0.1,0.2,0.3,0.4]`.
- Best results are for (`n_estimators`,`learning_rate`) with values (1150, 0.2).
- This can be interpreted as the learning rate of 0.2 giving the model more information gain per tree and thus with this learning rate of 0.2 not ≥ 1750, but a lower number of 1150 `n_estimators` give best results.
- Findings do not differ for the different *data options*.

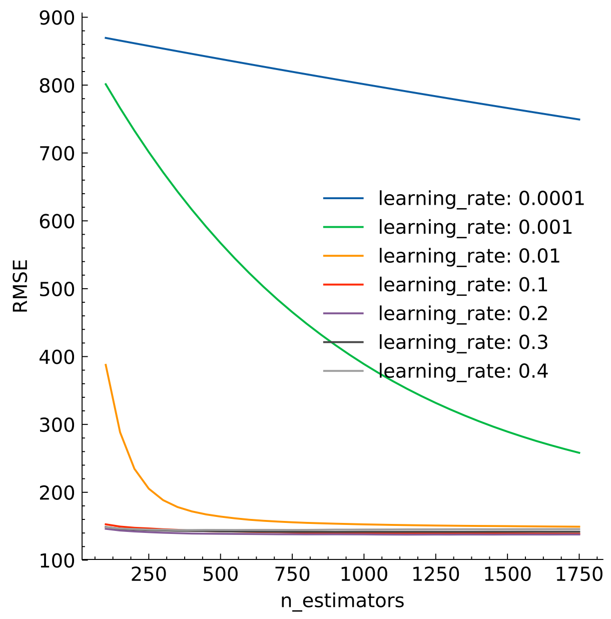


Figure 5.4. Plot showing the RMSE values for different values of `learning_rate` and `n_estimators`, tested together.

Subsampling, Subsampling by Column

For the remaining parameters `subsample`, `colsample_bytree` and `colsample_bylevel` individual tests were conducted.

- The value range is `[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,1]` for all of them.
- The optimum value is 0.5 for all of them.
- These results are consistent across the *data options*.

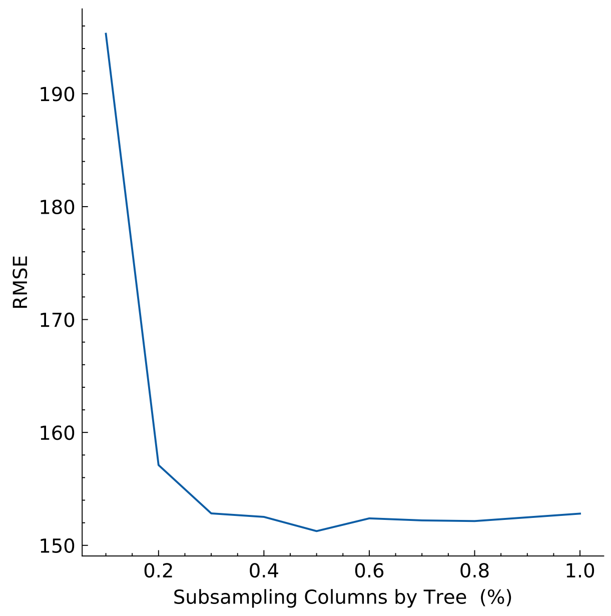


Figure 5.5. Plot showing the RMSE values for different values of `colsample_bytree`, during isolated testing.

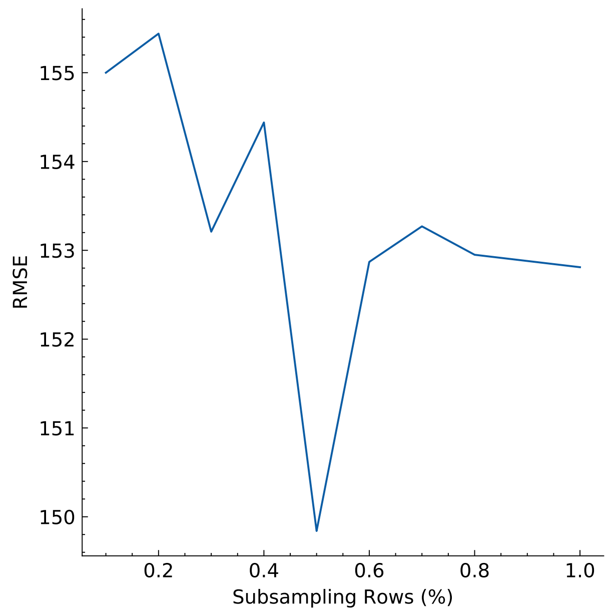


Figure 5.6. Plot showing the RMSE values for different values of `subsample`, during isolated testing.

Listing 5.2: Output of test subsample, to illustrate the raw print() output received from any of the tests in Python.

```
# subsampling: ‘[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,1]‘
# non standardised data (had better results than the standardised ↩
    data, by approx. 1 RMSE unit)
# all other parameters have their default values

Best: 149.840000 using {'subsample': 0.5}
The RMSE is 155.0 with: {'subsample': 0.1}
The RMSE is 155.44 with: {'subsample': 0.2}
The RMSE is 153.21 with: {'subsample': 0.3}
The RMSE is 154.44 with: {'subsample': 0.4}
The RMSE is 149.84 with: {'subsample': 0.5}
The RMSE is 152.87 with: {'subsample': 0.6}
The RMSE is 153.27 with: {'subsample': 0.7}
The RMSE is 152.95 with: {'subsample': 0.8}
The RMSE is 152.81 with: {'subsample': 1}
```

Summary The final model with the optimal parameter settings is shown in Listing 5.3. A very high number of trees of 1150 (default 100) showed to be optimal together with a learning rate of 0.2 (default 0.1). The interdependence of `n_estimators` and `learning_rate` as mentioned above showed over the testing process. Values up to 1750 were tested, after successive tests gave optimal results at the highest values in the range of values of the respective test for this parameter. From the joint experiment of these two parameters is where the final values come from. There the optimal number of trees was in the middle of the value range with 1150. The maximum depth of each tree stayed consistent throughout the tests with a value of 5, not supporting the aforementioned hypothesis regarding `n_estimators` and `max_depth`. All subsampling parameters were found to have their optimum with a value of 0.5.
The tests were either identical for the *data options* or the difference was only ±1 RMSE unit.

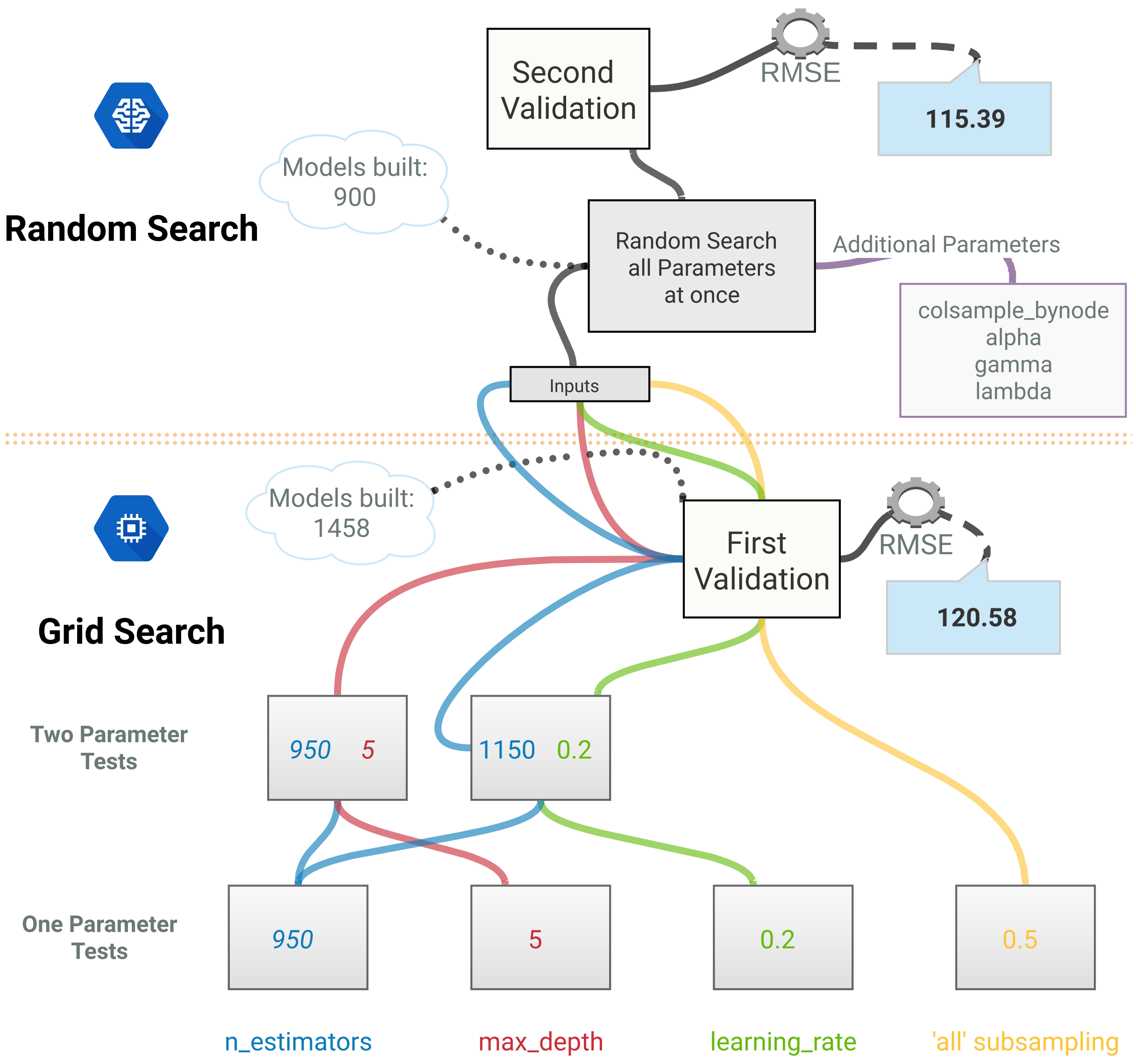


Figure 5.7. Graphic showing the process and the characteristics, as well as the outcomes of grid search and random search in this work. From bottom to top: The first row shows all one parameter tests and declares the respective colour associated with each parameter. Coloured lines show where a test result of a parameter value was used as an input for a test in a higher row. The second row gives the two parameter tests that were conducted. The validations give the final RMSE score for each approach. "Models built" gives the total number of models built for each approach. The dotted yellow double line separates the two approaches.

Validation Results The validation results were expressed in RMSE: 131.68 for the standardised dataset and 120.58 for the non standardised dataset. For the `XGBoost` model the standardised data showed a tendency to give worse results compared to the non standardised data during training. The spread on the validation data was much greater than on the training set.

Listing 5.3: Specification of the final XGBRegressor model after the grid search with parameter names from the xgboost Python library.

```
from xgboost import XGBRegressor

model = XGBRegressor(random_state=42, n_estimators=1150,
                     learning_rate=0.2, max_depth=5, subsample=0.5,
                     colsample_bylevel=0.5, colsample_bytree=0.5)
```

Random Search

Method There are many parameters that can be set on the `XGBoost` model [92]. Since the numerous possible value combinations for all parameters make it difficult to test efficiently, another approach to optimising the results was explored as well. Ten parameters were picked out for optimising the results using a randomised search. The difference between a grid search and a random search, based on the documentation section of the `sklearn` library for grid search [2] and random search [75] respectively, is that for a grid search the user gives an exhaustive list of values for each parameter to the grid search method. The grid search algorithm then builds models for all possible combinations of the values entered by the user. Therefore, the complexity of the search increases, with the number of values entered by the user. As an example, take 3 parameters a, b, c. For each one the user enters 10 values. The result is that during the grid search $10 \cdot 10 \cdot 10 = 1000$ models get built, regardless of how many of these combinations actually yield good accuracy scores. To combat this increase in complexity by increase in input values a random search can be used. Here distributions of variables are passed on to the algorithm instead of fixed values. Continuous distributions can be used to pass on non exhaustive sets of test values. The algorithm then samples the distributions given by the user and only builds models for some of the sampled input values. It looks for the combinations of parameters that yield a distinct and different result compared to all other models already built [11]. Thus there is no direct link between the number of input variables and the number of models that get built. The algorithm is given a "computational" budget in the form of how many models it is allowed to build in total during the test.

Listing 5.4: Specification of the final XGBRegressor model after the random search with parameter names from the xgboost Python library. All parameters were optimised in one single test where a total of 900 models were built.

```
from xgboost import XGBRegressor

"XGBRegressor(alpha=0.2, base_score=0.5, booster='gbtree',
       colsample_bylevel=0.6511245191281928, colsample_bynode=0.4001,
       colsample_bytree=0.890805910634995, gamma=1.3744079987707487,
       importance_type='gain', lambda=1.4200000000000008,
       learning_rate=0.013686923721625832, max_delta_step=0, ↩
          max_depth=10,
       min_child_weight=1, missing=None, n_estimators=1447, n_jobs=1,
       nthread=None, objective='reg:linear', random_state=42, ↩
          reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
       subsample=0.4529439356235432)"
```

Results In Listing 5.4, the parameters of the final XGBoost model can be found. The grid of parameter distributions given to the algorithm, is shown in Listing 5.5. With this method a total of 900 models were built and the validation scores are expressed in RMSE, for the standardised dataset 125.72 and for the non standardised dataset 115.39. In this example random search was better than a manual grid search (120.58) by a difference of RMSE = 4.19.

Listing 5.5: The parameter distributions used for the random search test.

```
import scipy.stats
import numpy as np

def get_truncated_normal(mean, sd, low, upp):
    return scipy.stats.truncnorm((low - mean) / sd, (upp - mean) / ↩
       sd, loc=mean, scale=sd)

param_dist = {
      'n_estimators'      : scipy.stats.randint(400, 1700),
      'learning_rate'     : np.random.random_sample(100),
      'max_depth'         : scipy.stats.randint(4, 12),
      'subsample'         : get_truncated_normal(mean=0.6, sd=0.4, ↩
         low=0, upp=1),
      'colsample_bylevel': get_truncated_normal(mean=0.7, sd=0.4, ↩
         low=0, upp=1),
      'colsample_bytree'  : get_truncated_normal(mean=0.6, sd=0.4, ↩
         low=0, upp=1),
      'colsample_bynode'  : ↩
         np.random.choice(np.arange(0.0001,1,000.1),size=300,replace=True),
      'lambda'            : ↩
         np.random.choice(np.arange(0.5,1.5,0.01),size=300,replace=True),
      'alpha'             : ↩
         np.random.choice(np.arange(0,1,000.1),size=300,replace=True),
      'gamma'             : scipy.stats.expon.rvs(size=500)}
```

It should be pointed out that the optimal values found with the grid search were used as a starting point for the random search. This could limit the comparability of the results in terms of how computationally expensive they were to get. Without these initial values it could be that more than 900 iterations with random search optimisation are needed to get the result of RMSE = 115.39. However, there is still a big gap in favour of the random search in this regard between the two procedures in terms of how many models were built. This is illustrated in Figure 5.7. During the grid search a total of 1458 models were built, while for the random search a total of 900 were built.

5.2 Overview - RMSE Values

A Table showing all RMSE values. From left to right, with the respective column names in brackets, the RMSE values are categorised by model ("Model"), the optimisation method where the respective RMSE value resulted from ("Method"), the RMSE score itself ("RMSE") and whether the training set or the validation set was used ("Training or Validation"). Method "baseline" describes the initial fitting of the models with no optimisation method applied. Values are for the non standardised data option, since RMSE values where either identical (±1) for both *data options* or better on the non standardised data.

Table 5.1. Table showing all RMSE values, from left to right: For Model, Method, the RMSE score and whether it was with the training set or the validation set.

Model	**Method**	**RMSE**	**Training or Validation**
`Lasso`	baseline	195.62	training
	grid search	190.68	validation
`XGBoost`	baseline	145.75	training
	grid search	120.58	validation
	random search	115.39	validation

5.3 Lasso - Interpretation of Coefficients

The coefficients of the `Lasso` model can be interpreted in the same way as the coefficients of a simple linear regression model. It cannot be ensured that causal relationships between the regressors and the dependent variable are described by the estimates of the coefficients by the model. For this it should be possible to rule out the possibility of an *omitted variable bias*. This is not possible due to the many factors potentially relevant for the value of the base rent, see Section 2.1. Nevertheless, the correlations between the regressors and the dependent variable are described below based on the coefficients estimated by the model. Detailed values for all weights for both *data options* can found in Table 5.2. Figure 5.8 shows the estimated weights in a bar plot for the non standardised data. Non standardised data was chosen here, as the coefficients of the continuous variables can be interpreted easier than with standardised data. There are no differences regarding whether a weight estimate has a positive or a negative sign between the *data options*. It is the values that can differ.

A foundation regarding the interpretation of regressors follows. The regression func-

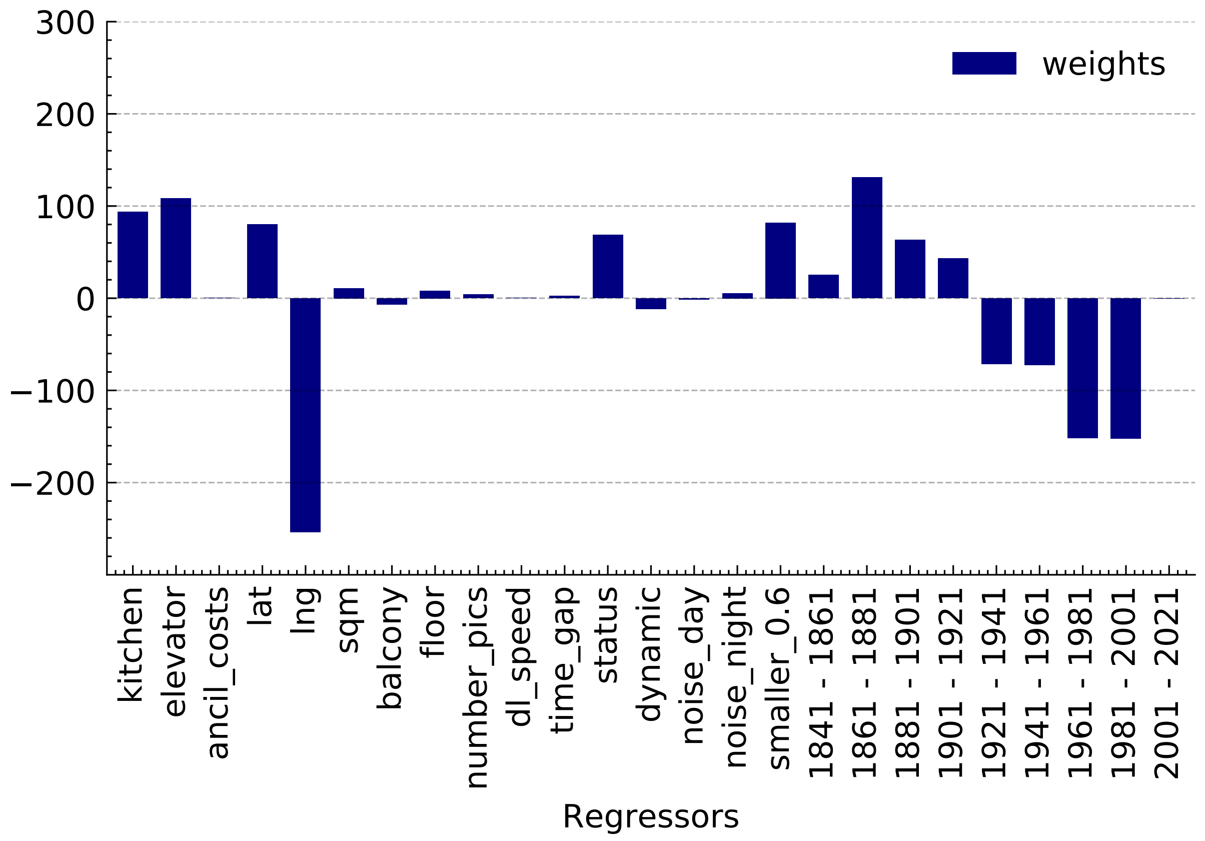


Figure 5.8. Plot showing the estimates for weights w_i with the optimal $\alpha = 0.1$. All independent variables are listed on the x-axis with the estimated weights on the y-axis. For an explanation of the variable names, please see Table 3.5 and Section 4.4.

tion is $y = b + m \cdot x$ and its partial derivative regarding $x_{i,j}$ is $\frac{\partial y}{\partial x_{i,j}} = \partial x_{i,j} \cdot w_j$. Let dummy regressor (variable) and its estimated weight be $x_{i,j}, w_j, i \in L, j \in X$ with L the sample of all listings, X the set of all regressors in the model. x_i has value $x_i = 0$, if the feature is absent that x_i describes and $x_i = 1$ if it is present. The relationship between x_i and y then is such, that x_i measures the difference in value of y between $x_i = 0$ and $x_i = 1$.

> The partial derivative $\frac{\partial Y}{\partial X}$ is the rate at which dependent variable Y changes per change in independent variable X, net of the effects of other variables in the model that influence Y. [81]

This means that the value of w_j translates to the amount that the dependent variable is higher, if the feature is present and $w_j > 0$, compared to when it is not. If $w_i < 0$, the weight gives the amount that the dependent variable is lower, if the feature is present, compared to when it is not. In the following for binary variables the part "compared to when it is not present", is not explicitly added, but this is always the basis for comparison. Also, all numbers expressed are in Euro, all statements are made *ceteris paribus* and they only describe effects on average between regressor and dependent variable.

The significance level of the estimated coefficient values can not be easily computed, since the standard error is difficult to calculate for the `Lasso` model [17, 83]. Therefore, it is unknown to what significance level the coefficients are not null here. In the following, the significance level $\alpha_{sig} = 5\%$ is used as a basis, if the significance level is not explicitly mentioned in the respective statement. With the low value of $\alpha = 0.1$, it is possible that little shrinkage has been applied to the coefficients by the model. There is only one weight that is set to zero by the model, see Table 5.2. To assess the level of shrinkage applied, the coefficients should be compared with the ones from an ordinary least squares regression model [30, 86].

Table 5.2: Weights of the `Lasso` for the standardised/non standardised data in column "Stand."/"Non Stand.".

Regressor	Stand.	Non Stand.
kitchen	93.3098	93.5189
elevator	108.9329	108.2650
ancil_costs	26.0164	0.3135
lat	6.239149	80.289014
lng	-23.7540	-253.5753
sqm	292.8326	10.9200
balcony	-6.4847	-6.6686
floor	14.9768	8.1879
number_pics	23.3877	4.1946
dl_speed	2.9967	0.1200
time_gap	41.1305	2.6461
status	47.8749	68.4892
dynamic	-3.6751	-11.7444
noise_day	-2.3626	-1.2076
noise_night	4.7951	5.5244
smaller_0.6	81.9303	81.9833
1841 - 1861	25.4141	25.3295
1861 - 1881	132.4772	131.1990
1881 - 1901	63.4769	63.2924
1901 - 1921	42.8391	43.1351
1921 - 1941	-71.5610	-71.5296
1941 - 1961	-72.6018	-72.6182
1961 - 1981	-152.4916	-151.8401
1981 - 2001	-153.4999	-152.4364
2001 - 2021	0.0000	0.0000

Interpretation - Binary Variables The interpretation of the weights for the binary variables is then: `kitchen` with a value of 93.52 suggests that `base_rent` is 93.52 higher, if the apartment has a fitted kitchen. `elevator` shows that base rent is 108.27 higher, when there is an elevator in the building. `balcony` points towards a lower base rent of -6.7, if there is a balcony. `smaller_0.6` suggests an 81.99 increase in base rent, if there is either subway or a suburban train station within a 600m radius of the apartment. The year the building was constructed was split into 9 groups, see Section 4.4. It is not possible to evaluate the coefficients of these like the binary variables before, due to the fact that all listings are in buildings that were built at some point. Therefore, the comparison with "feature is absent" can not be made. Nevertheless, correlations and the strength of them, approximated by the values of the estimated weights, can be analysed. The highest positive value is given for $1861-1881$ with 131.2 suggesting that houses built during that period have apartments that are more expensive compared to all the other 8 groups. The lowest positive value of all groups has $1841-1861$ with 25.33 This large difference between the two adjacent time periods $1841-1861$ and $1861-1881$ would require further analysis. Possibly binning bias plays are role or there was an asymmetry in the sample regarding the number of samples for each period or a historical event might play a role. Given that the significance levels of the estimated coefficients are not known, the coefficient estimate for either or for both of these periods, might not be valid. There could also be other factors that explain this. One notices that all groups from $1921-1941$ onwards up to $1981-2001$ have negative coefficients associated with them. This suggests that for buildings in the sample constructed within these periods the base rents are lower compared to buildings of the other groups. The values go from -71.53, -72.62 for the period $1921-1941$, $1941-1961$ down to -151.84, -152.44 for $1961-1981$, $1981, 2001$. These results backup the hypothesis made in Section 4.1, that the `base_rent` is comparatively low for apartments in buildings that were built during the highest construction periods in the sample. The only coefficient set to zero was the one for $2001-2021$, indicating that not much predictive importance is given to this category by the model.

Interpretation - Continuous Variables For the regressors with continuous values the interpretation of the weights is that for every marginal increase in x_j the dependent variable y changes in the magnitude equal to the value of w_j. Regarding whether y increases or decreases when x_j increases, it depends on the sign of w_j as mentioned earlier. Here marginal changes are based on the 1 Euro level. `ancil_costs` has a positive weight (0.31) indicating that an increase in ancillary costs by 1 Euro brings with it an increase in `base_rent` by 0.31 Euro. This is close to the mean value of the ratio $\frac{\texttt{ancil\_costs}}{\texttt{base\_rent}} = 0.25$ for all listings in the series. The latitude (`lat`) variable has a large positive coefficient (80.29). However, the marginal increase here is based on one additional latitude unit. In the series the maximum distance between two listings is ≈ 0.31 degrees of latitude or ≈ 33.97 kilometres. The function `lat_dist()`, used to convert between distance in degrees and kilometres can be found in the Annex, Listing 2. For the given data then, the degrees of latitude per kilometre is ≈ 0.00899. Multiplying with the coefficient value then gives $80.29 \cdot 0.00899 \approx 0.72$. The latitude values mark the position of an object in the North-South direction and its values increase on the northern hemisphere towards North. What this translates to then is that for each kilometre further North within the listings range `base_rent` increases by 0.72 Euro. For the entire latitude range of the sample in kilometres (≈ 33.97), this effect can at most cause a difference in `base_rent` of $33.97 \cdot 0.72 \approx 24.46$ Euro, by this estimation of the

coefficient. For the conversion between distance in degrees of longitude and kilometres the custom function `lng_dist()` was created, see Annex Listing 3. Longitude values increase from West towards East. The coefficient is -253.58. To better visualise the indicated relationship between longitude (`lng`) and `base_rent`, the maximum difference in longitude, kilometres is 0.56, 54.3. The degrees of longitude per kilometre is ≈ 0.01031. Given the coefficient this gives a decrease in `base_rent` per kilometre going eastwards of ≈ -2.61 Euro and the maximum difference possible, given the coordinate range of all listings in the series, is ≈ -141.96 Euro. These findings support the results in Section 4.3, Heat Map of Base Rent. There it was observed, that base rent per square meter has a tendency to increases towards the western part of Hamburg. It seems, that the here included and with `base_rent` highly correlated variable square meters (`sqm`) does not distort these finding from Section 4.3. There, the influence of `sqm` on `base_rent` was removed by combining it with `base_rent` to give `rent_sqm`.
The coefficient of square meters (`sqm`) is 10.92. It describes an increase in `base_rent` by 10.92 per additional square meter and could therefore have the predictive importance indicated by $\rho = 82$, Figure 4.4. With each floor (`floor`) higher, on which the apartment is located, so the model estimates the `base_rent` rises by 8.19 Euro. With each additional image (`number_pics`) uploaded for the listing, so the model estimates the `base_rent` rises by 4.19 Euro. Going to a higher download speed (`dl_speed`) category, is estimated to increases `base_rent` by 0.12, indicating a low predictive importance of this variable. Each additional day the listing is online on the platform (`time_gap`) raises base rent by 2.65, this gives a basis for the hypothesis that more expensive apartments are longer online on the platform, compared to less expensive ones. The same was indicated by Figure 4.4. Each increase in the `status` category with its 4 levels is estimated to raise `base_rent` by 68.49. For `dynamic` the sign of the coefficient is not what one would expect with a value of -11.7444. This categorical variable has 3 levels `[-1,0,1]`, where a higher value is associated with a better socio-spatial situation for any neighbourhood, see Section 3.5. Here each increase in `dynamic` is estimated to lower `base_rent` by -11.74. The noise exposure during day (`noise_day`) lowers `base_rent` only very little (-1.21) as the noise level increases from one category to another. An increase in noise level during night (`noise_night`) is estimated to increase `base_rent` by 5.52. Since these noise categories only have 6 categories each and since less than 10% of all listings are exposed to noise levels above the respective minimum threshold, these two categories have little impact overall, judging by the estimates. The positive value of the estimated coefficient for `noise_night`, if significant, should be further explored. It could be tested, if a tradeoff between good location of an apartment and higher noise levels, in favour of the location, is the case for a subset of the listings in the dataset. See Section 4.2 for the analysis of the noise data.

5.4 XGBoost - Feature Importance

The `XGBoost` model has a method called `feature_importances_` to evaluate the importance of each feature that was passed on to the model as a predictor. To understand the values, a short explanation of the here relevant structures is given. As explained in Section 3.2 for each iteration the model adds a new split to either an existing tree that is not yet fully grown or it creates a new tree with 1 split during the stochastic gradient descent. Actually, the model adds them sequentially, because it can parallelise this process [19], but for illustration purposes adding just one tree per iteration is assumed.

Since each new split is created to minimise the loss function, the splits are chosen for features that account for an "important part" of the loss score [67]. Since a greedy algorithm is used during this process, choosing to split on the respective chosen feature, yielded the highest improvement regarding the loss score for the given iteration [19, 29, 49]. With this theoretical background the importance of a feature is equivalent to the number of splits that were made for it. A feature with a high number of splits in the model is one that was regarded as the "most important" for many iterations during the greedy search [19]. The ranking of importance in Figure 5.9 is the result of comparing the total number splits made for each feature in the tree structure. For the final output the features are sorted by number of splits made in descending order. The values in Figure 5.9 are from the model that resulted from the random search procedure (see Listing 5.4 for the specification), since this one had the lowest RMSE score of all tested models. One notices in Figure 5.9, that the latitude (`lat`) and longitude (`lng`) variables rank highest in total splits, each with around 4000 splits. Given the very high number of 1447 trees in the final model and maximum depth of each tree of 10, a high number of splits made is

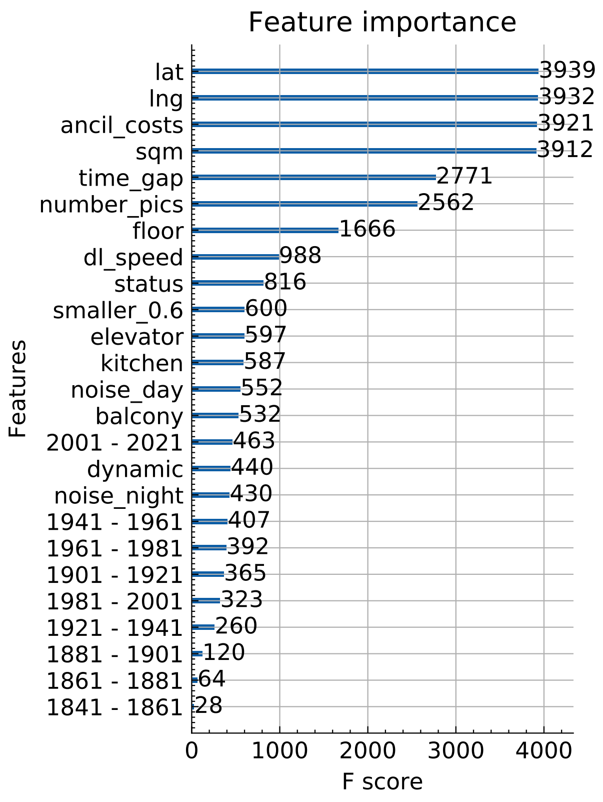


Figure 5.9: The feature importance plot from the final `XGBoost` model with the features on the y-axis and their respective scores on the x-axis. For an explanation of the variable names, please see Table 3.5 and Section 4.4.

possible in Figure 5.9. Ancillary costs (ancil_costs) and square meters (sqm) have almost the same feature importance with (3921, 3912) splits each. Then follows variable time_gap with over 1000 splits less compared to the aforementioned variables with 2771 splits. Interestingly the number of images upload for the listing (number_pics), while not having a broad variety of values with 90% of all its values (integers) in the range of 0 to 15 (Table 4.1), ranks high with 2562 splits. Next is variable floor, describing the floor the apartment is on, with 1666 splits. Again, a variable with limited value range, seems to be important in combination with other variables. These variables make up the group of variables with more than 1000 splits. Please see Figure 5.9 for a graphic including the number of splits made for all the predictive features in the model.

6 | Evaluation

This chapter reviews the results from Chapter 5 and compares the results of both models in Section 6.1. The results of the Hyperparameter Optimisation are summarised in Section 6.2. Finally, the limitations are discussed and possible future work is presented in Section 6.3.

6.1 Comparison of Results

An unrestricted comparison of the models cannot be made due to the very different types of operation of the models. It starts with the fact that the `Lasso` model is a relatively simple model compared to the `XGBoost`. There is a trade-off described in the literate between easy to interpret, simple models that show low accuracy and complex, hard to interpret models with high accuracy. [15, 42, 43, 88]. Low and high here means in comparison to the respective other model group in the statement. From (Johansson et al. 2011), a statement regarding the matter:

> In many cases, there is a clear trade-off between accuracy on the one hand and interpretability on the other. Models exhibiting the former property are many times more complex and opaque. These models are hard to interpret, while simpler and transparent, that is, interpretable, models may lack the necessary accuracy. This trade-off has been frequently observed in the machine learning and predictive modeling communities. [42]

This trade-off is observed here as well between the two models used for prediction. With the `Lasso` model the coefficients were well interpretable. It was possible to depict a possible relationship for each regressor with `base_rent` from analysing the sign and absolute value of each estimated coefficient. For the `XGBoost` model this was not the case. Only a raw metric [60] was available to evaluate the importance of a regressor during the stochastic gradient descent. This importance of a feature only states how important a feature is in the model to lower the loss function during the stochastic gradient descent [19, 51, 67]. It does not provide information about the relationship between the regressor and the dependent variable, whether there is a positive or negative correlation between the two [45]. Nor is there any information about the estimated quantitative relationship as obtained in a regression analysis [45, 32].

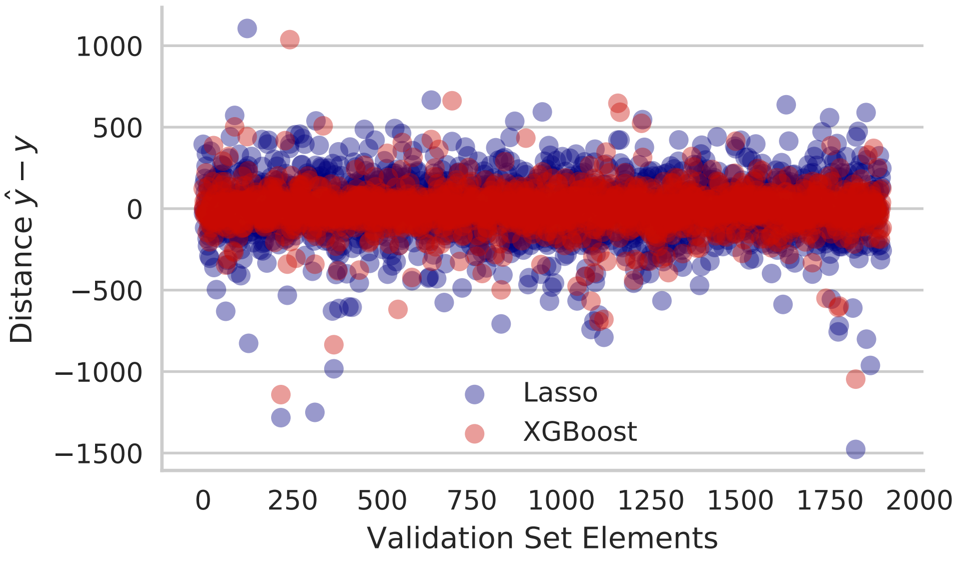


Figure 6.1. Plot comparing the residuals of both models. Blue dots designate the residuals of Lasso and red dots the residuals of XGBoost. On the x-axis one finds the elements of the validation set, for which the distance between the predicted value $\hat{y}$ to the true value y is calculated. This distance is plotted on the y-axis.

Where the XGboost shines is prediction accuracy. The best RMSE for the Lasso was 190.64 and 115.39 for the XGBoost. The difference in the residuals of both models is visible in Figure 6.1. The residuals of the Lasso have a wider spread than the ones of XGBoost. One finds this to be true, especially for large negative residuals, which describe the instances where the model predicted a value much smaller than the true value. In this area most of the residuals belong to Lasso, with only few belonging to XGBoost. The mean of the residuals is almost identical for both models with −3.77 for XGBoost and −3.97 for Lasso.

6.2 Hyperparameter Optimisation - Results

`Lasso`

The Hyperparameter Optimisation for the `Lasso` involved one test for the only configurable parameter alpha (α). The grid search algorithm was given a list with test values for parameter alpha and the best mean score on the respective test sets was found for $\alpha = 0$. During validation the model, using $\alpha = 0$, did not converge (see Section 5.1). With $\alpha = 0.1$ it did converge and since tests showed that the RMSE was identical on the training set using this value (see. Annex, Figure 3), $\alpha = 0.1$ was chosen as the optimal parameter. The (best) scores for the model then were in RMSE: Training set (baseline) 196.02, training set (grid search) 195.16, validation set 190.64.

`XGBoost`

The `XGBoost` model required many tests for the numerous parameters it has. Using a grid search proved to be computationally expensive and required specific structured tests. These involved finding good values for each parameter during single parameter tests first, and then testing, if they are still optimal when combined with a second parameter. 1458 models were built during the procedure and the baseline RMSE of 144.8 was lowered as a result to 120.58. A randomised search was performed after the grid search which optimised all parameters it was given as inputs at once. For this the inputs are distributions for each parameter rather than specific value ranges for each parameter, as is the case for a grid search. This method gave the lowest RMSE of 115.39 for all procedures and models in this work with 900 models built.

6.3 Limitations & Future Work

Limitations

The following limitations were identified for this work. The order of the description follows the order of the steps taken in this work.
The data, as it was collected from one real estate platform could have a selection bias. Firstly, it is not known what the true population of rental apartments in Hamburg is. Secondly, the here found sample might be a subset of the population which shares common values, this would imply that the sample is not representative of the population. For example, the listings in the sample could be concentrated around a few areas in Hamburg, could have a lower variance overall regarding base rent values and the

sample distribution of base rent might be different from that of the population or of another sample taken from a different platform or time range.
Given that an omitted variable bias can not be out ruled, causal relationships between the regressors and the dependent variable were not assessed in this work.
The interpretation of the coefficients of the `Lasso` model was carried out only under the condition that the coefficients are significantly different from `0` and that the assumptions made in advance for the model are true.

Future Work

Because of the aforementioned limitations, the generalisation of the here found results should be tested further in future work.
Regarding the prediction results, the `Lasso` model should be compared to other models, such as Linear Regression or Ridge Regression regarding the RMSE scores of the three models.
For interpretability a `Lasso` with an α closer to `1` should be tested and the estimated coefficients compared to the ones discussed in Section 5.3. Another alternative would be to use a Linear Regression model and to analyse its coefficients incorporating a significance test.
The structure of the final `XGBoost` model should be further analysed regarding the high number of `n_estimators`. This should include the analysis of the splits made for variables `lat` and `lng` in the tree structure, to evaluate whether the model recognises some sort of spatial autocorrelation. The role of parameter `learning_rate` in this regard should also be explored further.
More features should be created and added, and it should be tested which ones can aid even better RMSE results.
Adding a feature that takes into account the date the listing was posted on the platform and streamlining it to become a regressor was under construction but was not included here. It was difficult to find a suitable aggregated metric associated with the date a listing was put online. High volatility for comparable time ranges was found for any metric tested, and more testing would have been needed to find a solution. This should be further explored in future work.

7 | Discussion & Conclusion

Chapter 7 discusses to what extent the results obtained answer the research questions posed in Chapter 1 and gives a conclusion in this respect. As part of this, the integrated geospatial features are checked for their relevance with regard to the prediction of the base rent in Section 7.1. Section 7.2 gives the answer to the question posed in Chapter 4, whether standardising the data can aid better prediction results. It then presents, the final accuracy scores for the base rent predictions of both models. Subsequently, the Hyperparameter Optimisation procedure for both models is discussed in review and conclusions are drawn.

7.1 Dataset Construction

The integration of the geospatial features into the tabular data with the core variables from the listings, to form the final dataset, was the first research question. It was asked regarding how the different data could be combined to give relevant predictors for the regression. The prerequisite for this is consistent and valid data for longitude and latitude values of the listings, as acquired from the real estate portal Immobilienscout24.de [39]. These values were critical for joining all of the spatial features, namely the exposure to street noise, distance to the next subway or suburban train station and status class, dynamic of the statistical area the listing is in. These features were aligned with the core variables using a spatial join. There were 9480 listings that could be joined from the ≈ 12000 gained by the web scraping algorithm. The exposure to street noise and the status index had to be extracted from large `gml` files, converted to match the `crs` format of the listings and finally joined. It was here that it showed, that the GPS data of the listings was mostly valid, causing only a few errors. These errors occurred during the conversion of values from tuples with float numbers to `Shapely.geometry` objects. After that the values had to be converted to numerical values keeping the categorial order of the original data. Status data variable was given a high positive value in the `Lasso` model of 68.49 considering its four unique values. The status data showed to split the data only little, with almost all listings (and most

statistical areas in Hamburg [79]) having the second highest status value of the range. This might have hindered it being even more important in the end. It was a similar case with the exposure to noise data. Less than 1000 listings were found to be within any of the areas marked as having a relevant exposure to noise. These two variables therefore might not have given much predictive input to the models. However, it is possible that though they separated only few listings with their own categories, they might have been important separating instances of listings that were otherwise hard to predict correctly for the remaining regressors alone [45]. Variable distance to the next subway or suburban train station was converted to a binary variable and a distance of 0.6 kilometres, which is roughly equivalent to a 10 minute walking distance between listing and the closest station [41, 87], was chosen as threshold. This split the series into two almost equal parts. The estimated coefficient for it in the `Lasso` model was 81.99 and therefore one of the highest in the model, for binary variables. The `XGBoost` model ranked it low in the upper half of features. It is unclear what that entails regarding the relationship of the variable with the dependent variable. It might be that the feature did not require many splits with its value range of 0 and 1 for the model to reduce its loss function regarding this feature.
Overall the integration was a success and the features proved to be of relevance to the models, judging by the available tools that the models give to make such interpretations. This shows that including these features in any prediction of rent prices or property prices can further refine the given model, either by using already supplied and clean spatial features (e.g. from official reports) or by means of feature engineering as was done here.

7.2 Hyperparameter Optimisation - Discussion

Standardisation

The standardisation of the data showed to be of very little importance for `Lasso` and it consistently gave worse RMSE scores when used for the `XGBoost` model, compared with non standardised data. In the literature the following can be found from (Friedman et al. 2010) regarding the ability of the `Lasso` to scale to both *data options*:

> Our algorithms generalize naturally to the unstandardized case. [30]

This supports the findings in this work, that both *data options* can be used with the `Lasso`. As pointed out in Section 5.3, the coefficients of the continuous variables can be interpreted easier with non standardised data than with standardised data. This

comes from the fact that the standardised data do not have their original units anymore.
Regarding the question, how well the base rent of rental apartments in Hamburg can be predicted, using the remaining features in the dataset, the final results prove that it was accomplished with an extremely high accuracy on the validation set.
Using baseline parameters already gave very accurate results for both models. Both models yielded predictions with a high accuracy and low volatility during the testing.

> Another metric, apart from RMSE, commonly used is the explained variance by the model [74].
>
> **The best result of the `Lasso` was 81%, while the `XGBoost` reached an extremely high value of 93% in the final configuration. The optimisation efforts paid off here, raising the baseline score of 86% by 7%.**

`Lasso`

The choice of $\alpha = 0.1$ as the optimal value for the `Lasso` is discussed in the following. Opinions in the literate are also referred to in the discussion. The choice of $\alpha = 0.1$ was made purely based on looking for the parameter, that is most likely to give the lowest RMSE score on the validation set, as indicated by the results of the grid search. In review the tradeoff between a low shrinkage of coefficients, as is the case with $\alpha = 0.1$ and the marginal gain in RMSE score might not have been worth it. With $\alpha = 0.1$ the RMSE score was less than 1 RMSE unit better compared to the score received, using the default $\alpha = 1$. This is illustrated in the Annex, Figure 3. This consideration comes from the fact that with the chosen alpha the interpretability of the coefficients was degraded und more uncertain, compared to a model with an α value closer to the default of 1, which gives the ℓ_1-penalty. (Hastie et al. 2016) write:

> In the last 10 to 15 years, it has become clear that the ℓ_1-penalty has a number of good properties, which can be summarized as follows: Interpretation of the final model: The ℓ_1-penalty provides a natural way to encourage or enforce sparsity and simplicity in the solution. [...] [33]

`Lasso` **Conclusion**

To conclude the Hyperparameter Optimisation for the `Lasso` model, the bettering of the RMSE score was very small, with less than 1 unit between the default α value and the optimal value of 0.1 on the training set. The optimisation removed some of the good characteristics associated with the model and made interpretation of the estimated coefficients more uncertain. A solution might have been to consider another model, such as Linear Regression or Ridge Regression and compare the RMSE scores of the three models. For interpretability a `Lasso` with an α closer to 1 should be tested and the estimated coefficients compared to the ones discussed in Section 5.3.

`XGBoost`

A discussion of the way that the grid search was implemented here and a comparison of its results to the ones from the random search follows. Opinions in the literate are also taken into account regarding these topics. Since only 6 out of all parameters for the model (≈ 22) were included in the grid search, its results seemed unlikely to be optimal. This is given the small fraction of all value combinations explored, by only including the limited subset of parameters in the search. However, there are possibly diminishing gains to be had for each further optimisation effort using a grid search [11]. Refining the grid search is done by lowering the distance between each included test value further or by expanding the range of test values or by including more parameters in the test. These three can also be combined to further refine the search space. Soon further refinement can become prohibitively computationally expensive and take more time compared to random search [11]. Here a total of 1458 models were built during grid search. In their paper, (Bergstra and Bengio 2012) write:

> Grid search experiments are common in the literature of empirical machine learning, where they are used to optimize the hyper-parameters of learning algorithms. It is also common to perform multistage, multi-resolution grid experiments that are more or less automated, because a grid experiment with a fine-enough resolution for optimization would be prohibitively expensive. [11]

Another problem associated with including only a limited number of parameters in a test is that possible interdependencies between the parameter values of the parameters remain unconsidered.
Where single parameter tests showed to be insufficient was for all the subsampling parameters. This was found after learning that imposing a restriction using a true subset of all available columns or rows, depending on the parameter, with a single parameter has repercussions for the other subsampling parameters. To optimise these parameters, one has to include all of them in one test, otherwise it can happen, that the cumulative effect of these parameters gives the final model insufficient sample sizes to construct the trees in an optimal way [92]. Because of the aforementioned shortcomings of the here implemented grid search a random search was also conducted. A

limit was set for the random search procedure with a maximum um 900 models that could be built during testing (see Section 5.1 for more details on the procedure). A gain over the grid search result was found. The difference however, was much smaller between the two procedures with 4.19 RMSE units than between baseline and grid search (24.22). The initial values for the random search were the optimal values resulting from the grid search and distributions for the values in the random search were centred on these values as well. An observation made throughout the optimisation process is that the number optimal number of trees (`n_estimators`) was always very high with 1150 for the grid search and 1447 for the random search. While the learning rate (`learning_rate`) was higher than the default value for the grid search (0.2), it was lower for the random search (0.013686923721625832). Given that the highest number of splits were made for the latitude (`lat`) and longitude (`lng`) variables, with both having almost identical counts, this might be a sign of the model recognising some sort of localised spatial autocorrelation (see Section 2.2) and integrating it into its tree structure. There was no overfitting observed, as a result of these high numbers for `n_estimators`.

`XGBoost` Conclusion

To conclude the Hyperparameter Optimisation for the `XGBoost` model, the difference in starting points might lessen the comparability of the two results in terms of how many models had to be built, to get to the final parameter values with their associated RMSE scores. However, as mentioned before, from the total number of models built for the random search (900), a total of give or take 550 models more could have been built, until the total number built for this procedure matches the one of the grid search. This gap is, at least in this work, in favour of the random search being more efficient and more accurate than the grid search. A factor that plays an important role is the *experience level of the user* however [11, 45]. Had the author known better what parameters to include in the grid search and what value ranges to pick, the results could have come from building less models during the grid search. To some extend the same is true for the random search, as the choice of which distributions to pick for the sampling of parameter values, during random search, becomes more refined, the results might come after fewer iterations here as well. With the grid search this *experience level of the user* factor is more important than for the random search with the many choices that have to be made during the process of optimisation. Random search given its fewer input choices does not profit as much from this [11].
For both models no overfitting was observed by the comparison of RMSE scores between training and validation sets.

8 | Summary

This chapter is the final one of this work. It reviews the methods and the thinking process used during the here proposed process, in Section 8.1. Section 8.2 concludes this work with the identification of economic sectors and fields in which the process presented here can be applied and for which it is therefore relevant. Exemplarily, possible practical applications are also mentioned.

8.1 Review

In review the results proved that it is possible to combine core variables describing rental apartments listed on a real estate platform with independent geospatial features from different data sources. During the process it was shown how a web scraping algorithm was used to collect the core variables. The importance of cleaning and carefully joining these core variables with the geospatial features to retain existing and to create new consistent and valid features was emphasised. The understanding of the data and based on it, the selection of features with either a high predictive importance or a high importance for interpretability of the final model was included. New findings were iteratively integrated into the process and suitable solutions sought and implemented within the context of grid search and random search procedures. The best values regarding the prediction accuracy were extremely high in the end. However, little attention has been paid to the absolute value of accuracy metrics. Rather, the focus was on a methodically clean and structured approach and continued critical thinking. This also included the statements made being based on solid foundations. The premise was made that this approach should always be applied, regardless of whether very good results were achieved in the predictions from the outset. The focus then, was on doing each step of the process well, not primarily the outcome.

8.2 Applicability of the Process

Going back to the quotation from the newspaper article published in the *International Business Times UK* in Chapter 1, (Deep learning and big data 2017):

> In the world of finance, the new data paradigm entails applying predictive analytics to new datasets that are collected from non-traditional financial data sources to discover novel and consistently predictive features, and potentially useful patterns about the entity in question beyond what is easily available from traditional financial data sources. [24]

The need of the financial industry, as outlined in this quotation, matches the here described process for the most part. Therefore, several use cases in the financial sector should be found, where this process is of relevance. While the statement is clearly made for the financial industry in the original source, the author believes that it is applicable in other sectors as well. What characterises this sector, as described in the paper (Porter and Millar 1985), is:

> The Banking and newspaper industries have a high information-technology content in both product and process. [58]

While the paper is dated, its statement regarding the Banking industry is still valid. With this in mind the here described process should be of relevance to other sectors with similar characteristics. Potential sectors, where the proposed process can be applied to solve problems that have a similar structure to the one described in the quotation from article (Deep learning and big data 2017), are: 1. Banking & Finance, reasoning as explained in the preceding paragraphs. 2. Marketing, for example Retail. The first step would be to adapt the process to the business needs. An example would be predicting demand incorporating external factors such as weather and location of a store. 3. Healthcare, for example to find factors relevant for the development of personalised medication and treatments and to make a prediction of patient response using a well interpretable model in this context. 4. Policy, for example to understand socio-spatial developments in the different neighbourhoods of the city. A possible task could be to find relevant factors for the prediction of which investment in public goods gives the highest utility for the inhabitants of a city, incorporating spatial features. 5. Urban Development, for example to build more efficient cities regarding travel durations within the city. A possible task could be to identify factors important to the travel durations and predict the durations between important locations within the city. 6. Research, for example in fields such as Econometrics, Sociology or Environmental Sciences.

Bibliography

[1] *1.1. Generalized Linear Models — Scikit-Learn 0.20.3 Documentation* [n.d.], https://scikit-learn.org/stable/modules/linear_model.html#lasso.

[2] *3.2. Tuning the Hyper-Parameters of an Estimator — Scikit-Learn 0.20.3 Documentation* [n.d.], https://scikit-learn.org/stable/modules/grid_search.html.

[3] Adams, A. A. and McCrindle, R. [2008], *Pandora's Box: Social and Professional Issues of the Information Age*, John Wiley & Sons.

[4] Ahlfeldt, G. [2011], 'If Alonso Was Right: Modeling Accessibility and Explaining the Residential Land Gradient', *Journal of Regional Science* **51**(2), 318–338.

[5] Anselin, L. [2001], 'Spatial Effects in Econometric Practice in Environmental and Resource Economics', *American Journal of Agricultural Economics* **83**(3), 705–710.

[6] *API Overview — Geocoder 1.38.1 Documentation* [n.d.], https://geocoder.readthedocs.io/api.html#forward-geocoding.

[7] Arribas-Bel, D. [2014], 'Accidental, open and everywhere: Emerging data sources for the understanding of cities', *Applied Geography* **49**, 45–53.

[8] Bailey, T. C. and Gatrell, A. C. [1995], *Interactive Spatial Data Analysis*, Vol. 413, Longman Scientific & Technical Essex.

[9] Bardenet, R., Brendel, M., Kégl, B. and Sebag, M. [2013], Collaborative hyperparameter tuning, *in* 'Proceedings of the 30 Th International Conference on Machine Learning', Vol. 30, Atlanta, Georgia, USA, p. 9.

[10] Barham, H. [2017], Achieving Competitive Advantage Through Big Data: A Literature Review, *in* '2017 Portland International Conference on Management of Engineering and Technology (PICMET)', pp. 1–7.

[11] Bergstra, J. and Bengio, Y. [2012], 'Random Search for Hyper-Parameter Optimization', *Journal of Machine Learning Research* **13**(281-305), 25.

[12] Bergstra, J., Komer, B., Eliasmith, C., Yamins, D. and Cox, D. D. [2015], 'Hyperopt: A Python library for model selection and hyperparameter optimization', *Computational Science & Discovery* **8**(1), 014008.

[13] Bergstra, J. S., Bardenet, R., Bengio, Y. and Kégl, B. [n.d.], 'Algorithms for Hyper-Parameter Optimization', p. 9.

[14] Birattari, M., Yuan, Z., Balaprakash, P. and Stützle, T. [2010], F-Race and Iterated F-Race: An Overview, *in* T. Bartz-Beielstein, M. Chiarandini, L. Paquete and M. Preuss, eds, 'Experimental Methods for the Analysis of Optimization Algorithms', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 311–336.

[15] Bohanec, M. and Bratko, I. [1994], 'Trading accuracy for simplicity in decision trees', *Machine Learning* **15**(3), 223–250.

[16] Brakman, S., Garretsen, H. and Schramm, M. [2004], 'The strategic bombing of German cities during World War II and its impact on city growth', *Journal of Economic Geography* **4**(2), 201–218.

[17] Buja, A. and Brown, L. [n.d.], 'DISCUSSION: "A SIGNIFICANCE TEST FOR THE LASSO"', p. 10.

[18] Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S. and Schaub, T. [2016], The GeoJSON Format, Technical Report RFC7946, RFC Editor.

[19] Chen, T. and Guestrin, C. [2016], 'XGBoost: A Scalable Tree Boosting System', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16* pp. 785–794.

[20] Chen, T. and He, T. [2015], Higgs boson discovery with boosted trees, *in* 'NIPS 2014 Workshop on High-Energy Physics and Machine Learning', pp. 69–80.

[21] Chen, Y., Liu, X., Li, X., Liu, Y. and Xu, X. [2016], 'Mapping the fine-scale spatial pattern of housing rent in the metropolitan area by using online rental listings and ensemble learning', *Applied Geography* **75**, 200–212.

[22] Coors, S. [n.d.], 'Automatic Gradient Boosting', p. 96.

[23] *Dato Winners' Interview: 1st Place, Mad Professors* [2015], http://blog.kaggle.com/2015/12/03/dato-winners-interview-1st-place-mad-professors/.

[24] *Deep Learning and Big Data: Wall Street and the New Data Paradigm* [2017], https://www.ibtimes.co.uk/deep-learning-big-data-wall-street-new-data-paradigm-1648448.

[25] *DMLC* [n.d.], http://dmlc.ml/.

[26] *Einzugsbereiche von HVV-Haltestellen - GovData* [n.d.], https://www.govdata.de/daten/-/details/einzugsbereiche-von-hvv-haltestellen.

[27] *Exponential Change of Measure* [2003], *in* S. Asmussen, ed., 'Applied Probability and Queues', Stochastic Modelling and Applied Probability, Springer New York, New York, NY, pp. 352–379.

[28] Fik, T. J., Ling, D. C. and Mulligan, G. F. [2003], 'Modeling spatial variation in housing prices: A variable interaction approach', *Real Estate Economics* **31**(4), 623–646.

[29] Friedman, J. H. [2001], 'Greedy Function Approximation: A Gradient Boosting Machine', *The Annals of Statistics* **29**(5), 1189–1232.

[30] Friedman, J. H., Hastie, T. and Tibshirani, R. [2010], 'Regularization Paths for Generalized Linear Models via Coordinate Descent', *Journal of Statistical Software* **33**(1), 1–22.

[31] *GPS: The Global Positioning System* [n.d.], https://www.gps.gov/.

[32] Hastie, T., Tibshirani, R. and Friedman, J. H. [2009], *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, 2nd ed edn, Springer, New York, NY.

[33] Hastie, T., Tibshirani, R. and Wainwright, M. [2016], *Statistical Learning with Sparsity, The Lasso and Generalizations*, number 143 *in* 'Monographs on Statistics and Applied Probability'.

[34] Hepp, T., Schmid, M., Gefeller, O., Waldmann, E. and Mayr, A. [2016], 'Approaches to Regularized Regression – A Comparison between Gradient Boosting and the Lasso', *Methods of Information in Medicine* **55**(5), 422–430.

[35] Hoffmann, J. and Kurz, C. [2002], 'Rent indices for housing in West Germany 1985 to 1998', *ECB Working Paper* **116**, 56.

[36] Hogan, B. and Berry, B. [2011], 'Racial and Ethnic Biases in Rental Housing: An Audit Study of Online Apartment Listings', *City & Community* **10**(4), 351–372.

[37] Hu, T., Yang, J., Li, X. and Gong, P. [2016], 'Mapping Urban Land Use by Using Landsat Images and Open Social Data', *Remote Sensing* **8**(2), 151.

[38] Hutter, F., Hoos, H. H. and Leyton-Brown, K. [2011], Sequential Model-Based Optimization for General Algorithm Configuration, *in* C. A. C. Coello, ed., 'Learning and Intelligent Optimization', Vol. 6683, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 507–523.

[39] *Immobilien, Wohnungen Und Häuser Bei ImmobilienScout24* [n.d.], https://www.immobilienscout24.de/.

[40] Indra, I. [2008], "U-Bahn Gleich Aufwertung?", Dipl, uniwien, wien.

[41] Jermann, J. [2004], GIS-basiertes Konzept zur Modellierung von Einzugsbereichen auf Bahn-Haltestellen, PhD thesis, ETH Zurich.

[42] Johansson, U., Sönströd, C., Norinder, U. and Boström, H. [2011], 'Trade-off between accuracy and interpretability for predictive *in silico* modeling', *Future Medicinal Chemistry* **3**(6), 647–663.

[43] Jordan, M. I. and Mitchell, T. M. [2015], 'Machine learning: Trends, perspectives, and prospects', *Science* **349**(6245), 255–260.

[44] Kishor, N. K. and Morley, J. [2015], 'What factors drive the price–rent ratio for the housing market? A modified present-value analysis', *Journal of Economic Dynamics and Control* **58**, 235–249.

[45] Kuhn, M. and Johnson, K. [2013], *Applied Predictive Modeling*, Springer New York, New York, NY.

[46] *Lärmkarten Hamburg* [n.d.], https://www.hamburg.de/laermkarten/.

[47] *Liste der Bahnhöfe der S-Bahn Hamburg* [2018], *Wikipedia* .

[48] *Liste der Hamburger U-Bahnhöfe* [2018], *Wikipedia* .

[49] Loulou, R. and Michaelides, E. [1979], 'New Greedy-Like Heuristics for the Multidimensional 0-1 Knapsack Problem', *Operations Research* **27**(6), 1101–1114.

[50] Malley, B., Ramazzotti, D. and Wu, J. T.-y. [2016], Data Pre-processing, *in* MIT Critical Data, ed., 'Secondary Analysis of Electronic Health Records', Springer International Publishing, Cham, pp. 115–141.

[51] Mayr, A., Binder, H., Gefeller, O. and Schmid, M. [2014], 'The Evolution of Boosting Algorithms - From Machine Learning to Statistical Modelling', *Methods of Information in Medicine* **53**(06), 419–427.

[52] Möbert, J. [2018], Deutscher Häuser- und Wohnungsmarkt 2018, Technical report, Deutsche Bank Research.

[53] Montes de Oca, M. A., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M. and Dorigo, M. [2011], 'Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making', *Swarm Intelligence* **5**(3-4), 305–327.

[54] NDR [n.d.], 'Aktionstag: Wird Wohnen in Hamburg unbezahlbar?', /nachrichten/hamburg/Aktionstag-Wird-Wohnen-in-Hamburg-unbezahlbar-,wohneninhh102.html.

[55] Oduwole, H. K. and Eze, H. T. [2013], 'A hedonic pricing model on factors that influence residential apartment rent in Abuja satellite towns', *Mathematical Theory and Modeling* **3**(12), 65–73.

[56] Pellegrini, P. A. and Fotheringham, A. S. [2002], 'Modelling spatial choice: A review and synthesis in a migration context', *Progress in human geography* **26**(4), 487–510.

[57] Phil Graves, a., James C. Murdoch, a., Mark A. Thayer, a. and Don Waldman, a. [1988], 'The Robustness of Hedonic Price Estimation: Urban Air Quality', *Land Economics* **64**(3), 220–231.

[58] Porter, M. E. and Millar, V. E. [1985], 'How Information Gives You Competitive Advantage', p. 24.

[59] Probst, P., Bischl, B. and Boulesteix, A.-L. [2018], 'Tunability: Importance of Hyperparameters of Machine Learning Algorithms'.

[60] *Python API Reference Feature Importance — Xgboost 0.83.Dev0 Documentation* [n.d.], http://tinyurl.com/y6gn7u3s.

[61] *Python API Reference — Xgboost 0.81 Documentation* [n.d.], https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn.

[62] Rae, A. [2015], 'Online Housing Search and the Geography of Submarkets', *Online Housing Search and the Geography of Submarkets* .

[63] Raman, S., Fuchs, T. J., Wild, P. J., Dahl, E. and Roth, V. [2009], The Bayesian group-Lasso for Analyzing Contingency Tables, *in* 'Proceedings of the 26th Annual International Conference on Machine Learning', ICML '09, ACM, New York, NY, USA, pp. 881–888.

[64] Rondinelli, C. and Veronese, G. [2011], 'Housing rent dynamics in Italy', *Economic Modelling* **28**(1), 540–548.

[65] Rosen, S. [1974], 'Hedonic prices and implicit markets: Product differentiation in pure competition', *Journal of political economy* **82**(1), 34–55.

[66] *Scalable, Portable and Distributed Gradient Boosting (GBDT, GBRT or GBM) Library, for Python, R, Java, Scala, C++ and More. Runs on Single Machine, Hadoop, Spark, Flink and DataFlow: Dmlc/Xgboost* [2019], Distributed (Deep) Machine Learning Community.

[67] Schapire, R. E. [1990], 'The strength of weak learnability', *Machine Learning* **5**(2), 197–227.

[68] Schreckenberg, D., Schuemer-Kohrs, A., Schuemer, R., Griefahn, B. and Moehler, U. [1999], 'An interdisciplinary study on railway and road traffic noise: Annoyance differences', *The Journal of the Acoustical Society of America* **105**(2), 1219–1219.

[69] *Scikit-Learn: Machine Learning in Python — Scikit-Learn 0.20.3 Documentation* [n.d.], https://scikit-learn.org/stable/.

[70] Serinaldi, F. [2008], 'Analysis of inter-gauge dependence by Kendall's τK, upper tail dependence coefficient, and 2-copulas with application to rainfall fields', *Stochastic Environmental Research and Risk Assessment* **22**(6), 671–688.

[71] Simon, N., Friedman, J. and Hastie, T. [n.d.], 'A Blockwise Descent Algorithm for Group-penalized Multiresponse and Multinomial Regression', p. 15.

[72] Sirmans, C. F., Turnbull, G. K. and Benjamin, J. D. [1991], 'The markets for housing and real estate broker services', *Journal of Housing Economics* **1**(3), 207–217.

[73] *Sklearn.Linear_model.Lasso — Scikit-Learn 0.20.3 Documentation* [n.d.], https://tinyurl.com/yxlm6otb.

[74] *Sklearn.Metrics.Explained_variance_score — Scikit-Learn 0.20.3 Documentation* [n.d.], http://tinyurl.com/y39ovrd6.

[75] *Sklearn.Model_selection.RandomizedSearchCV — Scikit-Learn 0.20.3 Documentation* [n.d.], http://tinyurl.com/y66cmlpl.

[76] *Sklearn.Preprocessing.StandardScaler — Scikit-Learn 0.20.3 Documentation* [n.d.], https://tinyurl.com/ya3puadx.

[77] Snoek, J., Larochelle, H. and Adams, R. P. [n.d.], 'Practical Bayesian Optimization of Machine Learning Algorithms', p. 9.

[78] *Sozialmonitoring* [n.d.], https://www.hamburg.de/sozialmonitoring/.

[79] *Sozialmonitoring Bericht 2018* [n.d.], Technical report, Behörde für Stadtentwicklung und Wohnen Amt für Wohnen, Stadterneuerung und Bodenordnung, Hamburg.

[80] *Statistik Einwohner Haushalte Wohnungen Wohnungsbau* [n.d.], https://www.mieterverein-hamburg.de/de/aktuelles/statistiken-wohnen-hamburg/.

[81] Stolzenberg, R. M. [1980], 'The Measurement and Decomposition of Causal Effects in Nonlinear and Nonadditive Models', *Sociological Methodology* **11**, 459.

[82] *The Best Route Planner for Cycling, Walking, Hiking and Running* [n.d.], https://www.komoot.com/plan.

[83] Tibshirani, R. [1996], 'Regression Shrinkage and Selection Via the Lasso', *Journal of the Royal Statistical Society: Series B (Methodological)* **58**(1), 267–288.

[84] Tibshirani, R. [n.d.*a*], 'Modern regression 2: The lasso', p. 24.

[85] Tibshirani, R. [n.d.*b*], 'Sparsity, the Lasso, and Friends', p. 34.

[86] Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J. and Tibshirani, R. J. [2012], 'Strong rules for discarding predictors in lasso-type problems: Strong Rules for Discarding Predictors', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **74**(2), 245–266.

[87] Walther, K. [2013], *Nachfrageorientierte Bewertung der Streckenführung im öffentlichen Personennahverkehr*, Springer-Verlag.

[88] Wang, G., Ma, J. and Yang, S. [2014], 'An improved boosting based on feature selection for corporate bankruptcy prediction', *Expert Systems with Applications* **41**(5), 2353–2361.

[89] *Welcome to Python.Org* [n.d.], https://www.python.org/.

[90] Wickham, H. [2014], 'Tidy Data', *Journal of Statistical Software* **59**(1), 1–23.

[91] Wu, W., Zhang, W. and Dong, G. [2013], 'Determinant of residential location choice in a transitional housing market: Evidence based on micro survey from Beijing', *Habitat International* **39**, 16–24.

[92] *XGBoost Parameters — Xgboost 0.83.Dev0 Documentation* [n.d.], https://xgboost.readthedocs.io/en/latest/parameter.html.

Annex

German Version of the Sozialmonitoring Bericht 2018 Quote in Section 3.5

Der Sozialmonitoring-Bericht analysiert und beschreibt jährlich sozialräumliche Entwicklungen innerhalb der Freien und Hansestadt Hamburg. Ziel ist es, sozialräumliche Unterschiede innerhalb der Stadt zu erkennen und potenziell unterstützungsbedürftige Quartiere zu identifizieren. Hierfür erfolgt eine kleinräumige Analyse ausgewählter Indikatoren auf der Ebene der 941 Statistischen Gebiete der Stadt Hamburg. So können Teilräume beobachtet, miteinander verglichen und Statistische Gebiete identifiziert werden, in denen ggf. kumulierte Problemlagen zu vermuten sind. [79]

German Version of the Deutsche Bank Research Quote in Section 4.1

In Hamburg stiegen die Wohnungspreise im Bestand seit dem Jahr 2009 um mehr als 70%. Die Mieten wachsen im Vergleich zu den anderen Metropolen unterdurchschnittlich. Die relativ rege Bauaktivität und die stabile Einwohnerzahl dämpfen die Mietdynamik. Die Niedrigzinsen könnten deshalb der Haupttreiber für Hamburgs Wohnungs- und Hauspreise sein. Entsprechend könnte die Zinssensitivität höher sein als in anderen Metropolen. In unserem Basisszenario erwarten wir 2018 nur leicht steigende Hypothekenzinsen. Daher dürften in Hamburg die Haus- und Wohnungspreise im laufenden Jahr weiter kräftig zulegen. [52]

Table 1. The table shows all variables that were collected from the web scraping. Note that the json duplicates of some variables were collected to cross check the values collected from the respective counter part or to fill in missing values where one of the pair did not return a value, but the other one did.

Variable	**Description of what it measures**
kitchen	Listing has a fitted kitchen yes/no
json_hasKitchen	Listing has a fitted kitchen yes/no from the json data
elevator	Building has an elevator yes/no
json_lift	Building has an elevator yes/no from the json data
base_rent	The base rent of the listing
json_baseRent	The base rent of the listing from the json data
sqm	The living space of the apartment in (m^2)
json_livingSpace	The living space of the apartment in (m^2) from the json data
ancil_costs	Costs that are not included in the base rent
total_rent	Total rent per month consisting of base rent + ancillary costs
lat	Latitude value of the listing
lng	Longitude value of the listing
gps	The combined data from lng and lat as tuples in the form of (lng,lat)
str	Street and number of the listing
use_a	Usable area associated with the listing
no_room	The number of rooms the listing has
no_bed	The number of bedrooms the listing has
no_bath	The number of bathrooms in the listing
plz	The area code of the listing
regio	Raw data column that contains area code, city and neighbourhood of the listing
parking	Describes the type and number of parking spots that are associated with the listing
online_since	The date the listing was put posted
offline_since	The date the listing was taken off the platform
time_gap	The gap in days between online_since and offline_since
const	The year the building was constructed
json_yearConstructed	The year the building was constructed from the json data
cond	The condition the listing is in
json_condition	The condition the listing is in from the json data
json_interiorQual	The interior quality of the listing from the json data
type	The type of the listing
floor	The floor the listing is on inside the building
heating_type	The heating type of the listing
json_heatingType	The type of heating from the json data
rel_en	The relevant energy sources of the listing
fin_en	Amount of energy needed per month
hcost	The heating costs per month
balcony	Listing has a balcony yes/no
json_balcony	Listing has a balcony yes/no from the json data
json_number_pics	The number of images uploaded for the listing from the json data
json_telekomDownloadSpeed	The internet download speed (Provider is Telekom) from the json data in mbit/s
json_telekomUploadSpeed	The internet upload speed (Provider is Telekom) from the json data in mbit/s
json_electricityBasePrice	The base price of electricity for the listing
json_electricityKwhPrice	The electricity Kwh price for the listing from the json data
cellar	The listing has a basement compartment yes/no
json_cellar	The listing has a basement compartment yes/no from the json data
json_petsAllowed	Pets are allowed in the listing yes/no from the json data
pets	Pets are allowed in the listing yes/no
city_neigh	The city and neighbourhood that the listing is in
json_totalRent	Total rent per month consisting of base rent + ancillary costs from the json data
json_firingTypes	The firing type used in the listing for heating
json_yearConstructedRange	The range in which the building of the listing was constructed in from the json data

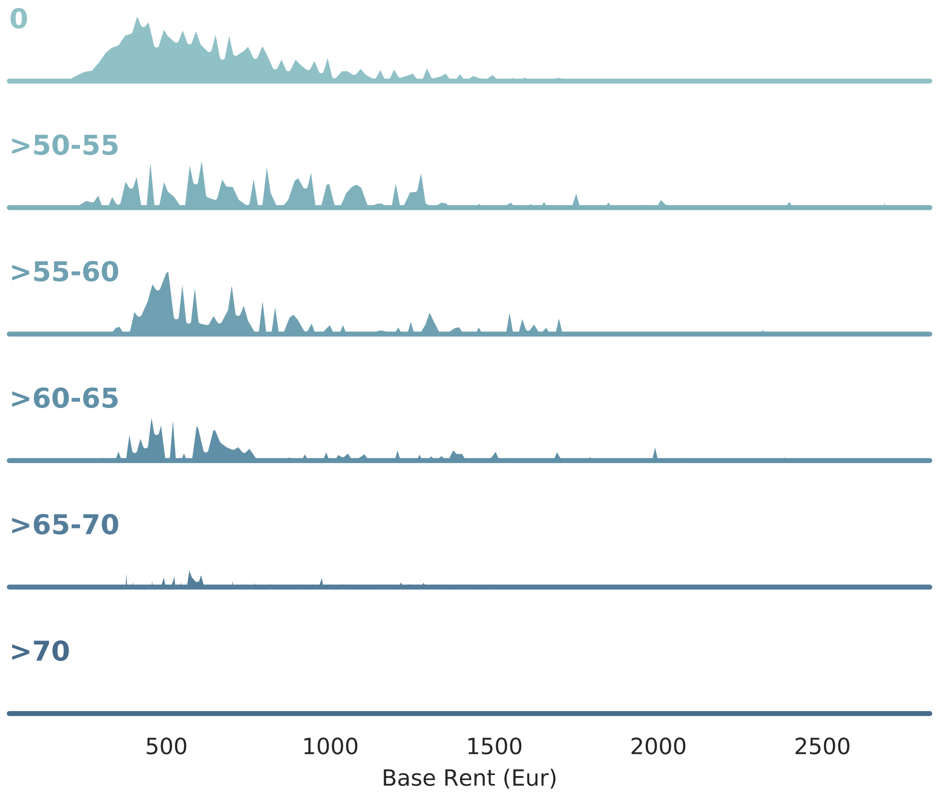


Figure 1. The relationship between base rent (Eur) on the x axis and its frequency by noise level during night (dB (A)) on the y axis. The noise level increases from top to bottom.

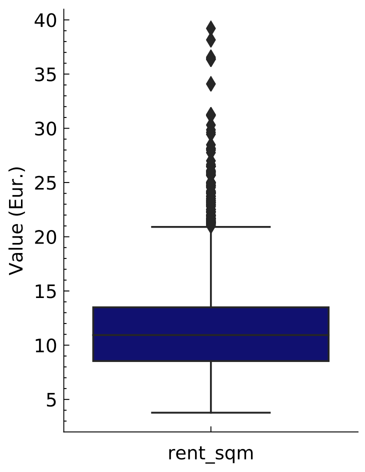


Figure 2. Box Plot of the variable `rent_sqm` showing the value on the y-axis in Euro per square meter .

Table 2. Results of comparing the distributions of the variables with the Exponential Distribution in order to check for heavy tails. The Exponential Distribution gives the minimum requirements for the length of a tail, in order for a distribution to be possibly 'heavy tailed' (See 4.2). Thus, results indicating that a power distribution is not a better fit than an Exponential Distribution make it highly unlikely for the distribution to have a heavy tail. In the tuple the sign of first element indicates whether a power distribution (> 0) is is a better fit or a non power distribution (< 0). H_0 is that a non power distribution is the better fit. The second element is the Pearson Correlation Coefficient. Here the significance level was set to 5%, so values smaller 0.05 here indicate that H_0 should be dismissed. The value 0 indicates no likelihood for either direction was the result.

Variable	**(Log-likelihood Ratio, *p*-Value)**
kitchen	(0, 1)
elevator	(0, 1)
ancil_costs	(0.8633620531244639, 0.38793843629339375)
lat	(nan, nan)
lng	(-894.2756317300255, 0.0)
base_rent	(2.863144900365252, 0.00419458690403231)
sqm	(-0.9036353423406421, 0.36618879440105934)
no_room	(-1.212361705403381, 0.2253739530242902)
balcony	(0, 1)
floor	(3.53516182742496, 0.000407525499472497)
number_pics	(-1.886985742467091, 0.05916224334355868)
dl_speed	(-228.043408947026, 0.0)
ul_speed	(-125.44752544664827, 0.0)
time_gap	(-15.833074808018742, 1.840051927320293e-56)
status	(-31.145734011188, 5.793603111328023e-213)
dynamic	(0, 1)
min_train	(-2.310867364093777, 0.020840182209689636)
min_subway	(-33.3742635115118, 3.239950438616581e-244)
smaller_0.6	(0, 1)
const	(nan, nan)

Protocols of Grid Search

Lasso

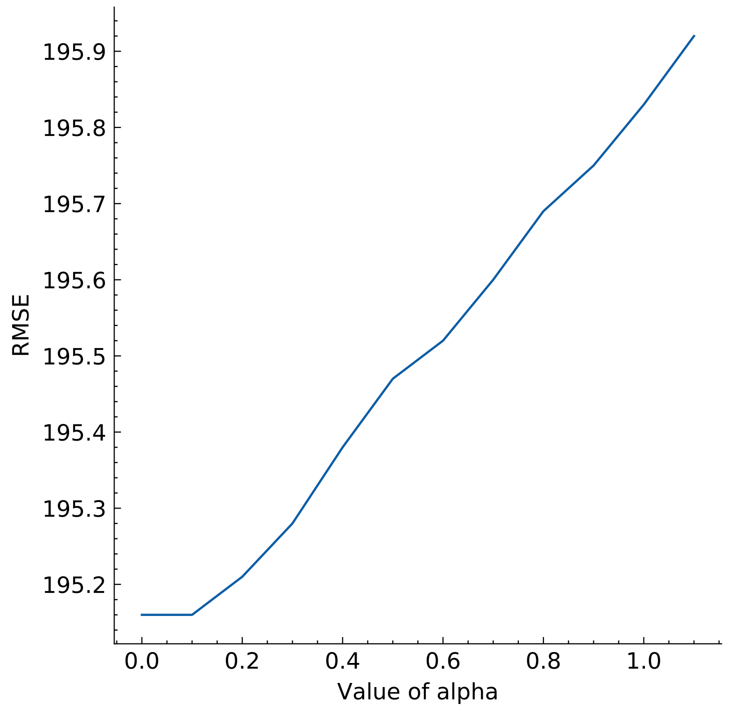


Figure 3. Plot showing the RMSE values (y-axis) for the range of tested alpha values (x-axis) during the grid search.

XGBoost

Baseline

Listing 1: Output of testing predictions with the default parameters on non standardised and standardised data.

```
# Baseline

# Non standardised data

# Results
The RMSE of the model is:(145.74874692689227)

# Standardised data

# Results
The RMSE of the model is:(144.80089959581352)
```

Custom Conversion Functions

Listing 2: Custom function used to convert the latitude values to kilometres. Since the distance between a fixed distance in degrees between 2 latitude values is constant the distance between two latitude values only has to be multiplied by the constant.

```
import numpy as np

def lat_dist(lat1, lat2):
    diff = np.abs(lat1 - lat2)
    diff = 111.19 * diff # constant = 111.19
    print('The difference in Kilometres is ' + str(diff) + ' km')
    return (diff)
```

Listing 3: Custom function used to convert the distance from degrees of longitude to kilometres. Since the distance between a fixed difference in degrees of longitude is not constant factor c has to be calculated for the actual conversion. Here the median of all latitude values found in the data series is used for the calculation of c.

```
import numpy as np

def lng_dist(lng1, lng2, lat):

  # approximate value of the circumference of the Earth in km
    r = 9367

  # calculation of factor c, as distance between longitude values ↩
      varies by latitude
    c = r * np.cos(lat * 0.0174533)

  # the actual distance calculation
    dist = c * np.abs((lng2 * 0.0174533) - (lng1 * 0.0174533))
    print('The difference in Kilometres between the longitude values ↩
       is ' + '\n' + str(dist) + ' km')
    return (dist)
```

Source of Listing 4:

https://github.com/dmlc/xgboost/blob/master/python-package/xgboost/core.py#L953] [1]

Listing 4: Underlying code of the get_fscore method.

```
def get_fscore(self, fmap=''):
    """Get feature importance of each feature.
    Parameters
    ----------
    fmap: str (optional)
       The name of feature map file
    """
    trees = self.get_dump(fmap)  ## dump all the trees to text
    fmap = {}
    for tree in trees:              ## loop through the trees
        for line in tree.split('\n'):      # text processing
            arr = line.split('[')
            if len(arr) == 1:              # text processing
                continue
            fid = arr[1].split(']')[0]     # text processing
            fid = fid.split('<')[0]        # split on the ↩
               greater/less(find variable name)

            if fid not in fmap:   # if the feature id hasn't been seen ↩
               yet
                fmap[fid] = 1     # add it
            else:
                fmap[fid] += 1    # else increment it
    return fmap                   # return the fmap, which has the ↩
       counts of each time a  variable was split on
\endinput
```

List of Figures

List of Tables

Listings

I certify that the thesis at hand was made without unauthorized help and that I only used the tools denoted. All statements literally or logically taken from publications are marked as quotes.

Freiburg, April 17, 2019

. . . .

(Signature Tobias Klein)